

12.2.a.1)

Given:

$$\dot{x}_1 = x_1 + x_2$$

$$\dot{x}_2 = 3x_1^2 x_2 + x_1 + u$$

$$y = -x_1^3 + x_2$$

Find:

Design a state feedback controller to stabilize the origin.

Solution:

Since we are doing state feedback, ignore output  $y$ .

$$\dot{x} = \begin{bmatrix} x_1 + x_2 \\ 3x_1^2 x_2 + x_1 \end{bmatrix} + \begin{bmatrix} 0 \\ u \end{bmatrix}$$

$\underbrace{\hspace{2cm}}_{f(x)}$        $\underbrace{\hspace{2cm}}_{g(x)}$

Note that:

 $n = 2$  for this problem

$$\text{ad}^0 g(x) = g(x)$$

$$\text{ad}^1 g(x) = [f, g](x) = \nabla g(x) f(x) - \nabla f(x) g(x)$$

$$\text{ad}^2 g(x) = [f, \text{ad}^1 g](x)$$

⋮

$$\Rightarrow \text{ad}^0 g(x) = \begin{bmatrix} 0 \\ u \end{bmatrix} \quad \text{ad}^1 g(x) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 6x_1 x_2 + 1 & 3x_1^2 \end{bmatrix}$$

$$G(x) = \begin{bmatrix} 0 & -1 & -1 \\ u & -6x_1 x_2 - 1 & -3x_1^2 \end{bmatrix} \quad \text{has } \underline{\text{rank}(G(x)) = 2}$$

And

$$\Delta := \text{span}\{g(x), \text{ad}^1 g\} \subseteq P$$

↳ Involutive

To find a satisfactory  $h(x)$  we must satisfy

$$\nabla L^0 h = 0 \Rightarrow \nabla(h(x))g(x) = 0$$

$$\nabla L^0 h \neq 0 \Rightarrow \nabla(\nabla h(x)f(x))g(x) \neq 0$$

$$h(0) = 0$$

$$\nabla h(x)g(x) = \left[ \begin{array}{cc} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \end{array} \right] \left[ \begin{array}{c} 0 \\ 1 \end{array} \right] = \frac{\partial h}{\partial x_2} = 0$$

$h(x)$  is independent of  $x_2$

$$\Rightarrow \nabla h(x)f(x) = \left[ \begin{array}{cc} \frac{\partial h}{\partial x_1} & 0 \end{array} \right] \left[ \begin{array}{c} x_1 + x_2 \\ 3x_1 x_2 + x_1 \end{array} \right] = \frac{\partial h}{\partial x_1} (x_1 + x_2)$$

$$\Rightarrow \left[ \begin{array}{cc} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_1} \end{array} \right] \left[ \begin{array}{c} 0 \\ 1 \end{array} \right] = \frac{\partial h}{\partial x_1} \neq 0$$

$$\text{Choose } h(x) = x_1 \Rightarrow h(0) = 0$$

$$\underline{z}_1 = h(x) = x_1 ; \underline{z}_2 = \dot{z}_1 = \dot{x}_1 = x_1 + x_2$$

$$\Rightarrow \dot{z}_1 = \dot{z}_2 = \underbrace{x_1 + x_2}_{\bar{u}} ; \dot{z}_2 = \dot{x}_1 + \dot{x}_2 = x_1 + x_2 + 3x_1^2 x_2 + x_1 + u \\ = 3x_1^2 x_2 + 2x_1 + x_2 + u$$

$$\bar{u} = -(3x_1^2 x_2 + 2x_1 + x_2) + v$$

We are left w/

$$\dot{z} = Az + Bu = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v$$

Choose a  $V$  such that  $A - BK$  is Hurwitz.

$$V = -K^2 \text{ and let } K = [10 \ 10]$$

$$\Rightarrow \dot{z} = (A - BK)z \Rightarrow \text{eig}(A - BK) = -1.12, -8.87$$

$$\Rightarrow u = \bar{u} - V = -3x_1^2x_2 + 2x_1 + x_2 - Kz$$

12.2.3.2)

Given:

$$\dot{x}_1 = x_1 + x_2$$

$$\dot{x}_2 = x_1 x_2^2 - x_1 + x_3$$

$$\dot{x}_3 = u$$

Find:

See (12.2.2.1)

Solution:

This time lets try linearization

$$J = \frac{\partial f}{\partial x} \Delta x + \frac{\partial f}{\partial u} \Delta u$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 1 & 1 & 0 \\ x_2^2 - 1 & 2x_1 x_2 & 1 \\ 0 & 0 & 0 \end{bmatrix} \Delta x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} v$$

$A \qquad \qquad \qquad B$

$$\Rightarrow \dot{\bar{x}} = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \bar{x} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} v \quad \begin{array}{l} \text{Let } v = -K\bar{x} \\ \text{and } K = \begin{bmatrix} 10 & 10 & 5 \end{bmatrix} \end{array}$$

$$\Rightarrow \dot{\bar{x}}_1 = (A - BK)\bar{x} \Rightarrow \text{eg}(A - BK) = -0.82 \pm 1.202i \\ -2.353$$

14.31)

Given:

$$\dot{x}_1 = x_2 + a + (x_1 - a^{1/3})^3$$

$$\dot{x}_2 = x_1 + u$$

a is a known constant

Find:

Using backstepping, find a state feedback controller to  
globally stabilize the system

Solution:

$$\eta = x_1 ; \xi = x_2$$

$$f_a(\eta) = a + (x_1 - a^{1/3})^3 ; g_a(\eta) = 1$$

$$f_b(\eta) = x_1 ; g_b(\eta, \xi) = 1$$

Suppose that there is a smooth function  $\xi = \phi(\eta) ; 0 = \phi(0)$   
 s.t.

$$\dot{\eta} = f_a(\eta) + g_a(\eta) \phi(\eta)$$

is asymptotically stable.

To make life easy lets start by saying

$$\phi(\eta) = \underbrace{-a - (x_1 - a^{1/3})^3}_{\text{in}} + \psi(\eta)$$

 $g_a(\eta) = 1$  so this cancels out  $f_a(\eta)$ Now let  $\psi(\eta) = -x_1$  to make  $\dot{\eta} = f_a + g_a \phi$  A.S.

$$\Rightarrow \phi(\eta) = -a - (x_1 - a^{1/3})^3 - x_1$$

Now suppose there is a Lyapunov function  $V_a$  and positive definite function  $W$  s.t.

$$\forall \eta \in D, \nabla V_a(\eta) [f_a(\eta) + g_a(\eta) \phi(\eta)] \leq -W(\eta)$$

$$\text{Let } V(x) = \frac{1}{2}x_1^2 \Rightarrow \dot{V}(x) = x_1 \dot{x}_1$$

$$\text{where } \dot{x}_1 = -x_1 \Rightarrow \dot{V}(x) = \underline{-x_1^2} \leq 0$$

To backstep define

$$\begin{aligned} z := \xi - \phi(\eta) &= x_2 + \alpha + (x_1 - 2^{1/3})^3 + x_1 \\ &= x_1 + x_2 + 3\alpha^{2/3}x_1 + x_1^3 - 3\alpha^{1/3}x_1^2 \end{aligned}$$

$$\Rightarrow \dot{z} = \dot{x}_1 + \dot{x}_2 + 3\alpha^{2/3}\dot{x}_1 + 3x_1^2\dot{x}_1 - 6\alpha^{1/3}x_1\dot{x}_1$$

$$\text{If } \alpha = 0$$

$$\dot{z} = \dot{x}_1 + \dot{x}_2 + 3x_1^2\dot{x}_1 = \underline{x_2 + x_1 + u + 3x_1^2x_2}$$

$$\text{Let } V = \frac{1}{2}x_1^2 + \gamma_2 z^2$$

$$\Rightarrow \dot{r} = x_1 \dot{x}_1 + 2\dot{z} = x_1 x_2 + 2(z(x_2 + x_1 + u + 3x_1^2x_2))$$

$$\text{Let } u = -x_1 - x_2 - 3x_1^2x_2 - z$$

$$\text{Then } \dot{r} = -x_1^2 - z^2 \leq 0$$

(4.34)

Given:

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

$$\dot{x}_2 = x_1 - x_2 - x_1 x_3 + u$$

$$\dot{x}_3 = x_1 + x_1 x_2 - 2x_3$$

Find:

- a) Starting w/  $x_1$  and stepping back into  $x_2$ , design a state feedback controller  $u = \psi(x)$  s.t.  $\dot{x} = [0]$  is G.A.S.
- b) Show what the controller found causes the system to be G.A.S.

Hint: Use input-to-state properties of the third equation

Solution:

$$\eta = x_1 \quad \xi = x_2$$

$$f_a(\eta) = -\eta \quad ; \quad g_a(\xi) = 1$$

$$f_b(\eta, \xi) = \eta - \xi - \eta \xi \quad ; \quad g_b(\eta, \xi) = 1$$

$$\dot{\eta} = f_a(\eta) + g_a(\eta) \phi(\eta) \quad ; \quad \text{Let } \phi(\eta) = 0$$

$$\Rightarrow \dot{\eta} = -\eta$$

$$\text{Let } V(x) = \frac{1}{2} \eta^2 \Rightarrow V(x) = \eta \dot{\eta} = -\eta^2 \leq 0$$

$$z := \xi - \phi(\eta) = \xi - 0 = \xi$$

$$\Rightarrow \dot{z} = \dot{\xi} = \eta - \xi - \eta \xi + u$$

$$\text{Let } V(x) = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2$$

$$\Rightarrow \dot{V}(x) = x_1\dot{x}_1 + x_2\dot{x}_2 = -x_1^2 + 2(x_1 - x_2 - x_1x_3 + u)$$

$$\Rightarrow u = -x_1 + x_2 + x_1x_3 - 2$$

$$\Rightarrow \dot{V}(x) = -x_1^2 + 2(-2) = -x_1^2 - x_2^2$$

Therefore the system

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

$$\dot{x}_2 = -x_2$$

is G.A.S

b) To show the system is G.A.S begin by observing

$$\dot{x}_3 = x_1 + x_1x_2 - 2x_3$$

where we have guaranteed  $x_1 \rightarrow 0$  and  $x_2 \rightarrow 0$ . The uncontrolled variable  $x_3$  is what remains.

$$V(x) = \frac{1}{2}x_3^2 \Rightarrow x_3\dot{x}_3 = x_3(x_1 + x_1x_2 - 2x_3) = \dot{V}(x)$$

$$\Rightarrow \dot{V}(x) = \cancel{x_3}\dot{x}_1 + \cancel{x_1}\dot{x}_2 - 2x_3^2 \leq 0$$

Therefore the system is G.A.S.

14.42)

Given:

$$\dot{x} = Ax + Bu$$

$$PA + A^T P \leq 0 \text{ and } P > 0$$

Find:

A globally stabilizing feedback law  $u = -\Psi(x)$  such that

$\|\Psi(x)\| < K + x$  where  $K$  is given as a positive constant

Solution:

$$\dot{x} = Ax + Bu ; y = Cx$$

$$\text{Let } V(x) = \frac{x^T P x}{2}$$

$$\begin{aligned}\Rightarrow \dot{V}(x) &= x(P + P^T) \dot{x} = x^T (P + P^T)(Ax + Bu) \\ &= x^T (PAx + PBu + P^TAx + P^T Bu) \\ &= x^T (PA + P^T A)x + x^T (PB + P^T B)u\end{aligned}$$

$$\text{Note } A^T B = B^T A \text{ if symmetric}$$

$$= x^T (PA + A^T P)x + x^T P B u = -B^T P u$$

$$\Rightarrow x^T (PA + A^T P)x + x^T P B u \leq B^T P x u = y^T u$$

$$\text{Let } B^T P x = C \text{ then}$$

$$x^T (PA + A^T P)x + x^T P B u \leq y u \quad \therefore \text{The system is passive}$$

Also note that if  $u = 0$  then when  $x = 0$ ;  $y = 0$

$\therefore$  The system is zero state observable.

$$\text{Choose } u = -K y$$

14.43)

Given:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -x_1^3 + \psi(u)$$

$\psi$  is:

- Locally Lipschitz

- $\psi(0) = 0$   $u\psi(u) > 0$   $\forall u \neq 0$

Find:

Design a globally stabilizing state feedback control.

Solution:

$$\text{Let } V(x) = \frac{1}{2}x^T x \Rightarrow \dot{V}(x) = x^T \dot{x} = x_1 \dot{x}_1 + x_2 \dot{x}_2$$

$$= x_1 x_2 + x_2 (-x_1^3 + \psi(u))$$

$$\Rightarrow \dot{V}(x) = x_1 x_2 - x_1^3 x_2 + x_2 \psi(u)$$

$$\Rightarrow -x_1 x_2 + x_1^3 x_2 = x_2 \psi(u) \quad \text{Let } x_2 = y$$

Therefore the system is passive

If  $u=0$  and  $y=0 \Rightarrow x_2=0 \Rightarrow \dot{x}_2=0 \Rightarrow x_1=0$

$\therefore$  The system is zero state observable

Let  $u = -y^2$

## Code/Plots

The problems were integrated using `solve_ivp` from the `scipy` library in Python.

### 12.2.2.1)

---

```
1 #!/bin/python
2
3 ##### Libraries
4 import matplotlib.pyplot as plt
5 import numpy as np
6
7
8 from scipy.integrate import solve_ivp
9
10 from plot import plot
11
12 #### Functions
13
14
15 #####
16 #
17 def model(t,x):
18     """
19         input
20             t: time step
21             x: state
22         output
23             dx: differential output of ode
24
25     """
26     # Extract states
27     x1, x2, u = x
28
29     # Calculate z
30     z1 = x1
31     z2 = x1 + x2
32     z = np.array([[z1], [z2]])
33
34     # Calculate control
35     k = np.array([10, 10])
36     u = -3*(x1**3)*x2 + 2*x1 + x2 - k[0]*z
37
38     # Calculate ODEs
39     xd1 = x1 + x2
40     xd2 = 3*(x1**2)*x2 + x1 + u
41
42     # Bundle ODEs
43     dx = [xd1, xd2, u]
44
45     return dx
46
47 #####
48 # Script
49
50 # Plotting/Calculating variables
```

```

51
52 ## Time horizon
53 t0 = 0.0
54 tf = 15.0
55 t = [t0, tf]
56
57 ## Initial conditions
58 x0 = [1, 0.1, 0]
59
60
61 # Solve ode model
62 sol = solve_ivp(model, t, x0, method='RK45', dense_output=True)
63
64 # Plot solution
65 plot(sol, tf=tf, title="12.2.2.1", legend=['x1', 'x2', 'u'])

```

---

## 12.2.2.2)

---

```

1#!/bin/python
2
3#####
4# Libraries
5import matplotlib.pyplot as plt
6import numpy as np
7
8from scipy.integrate import solve_ivp
9
10from plot import plot
11
12#####
13# Functions
14
15#####
16#
17def model(t,x):
18    """
19        input
20            t: time step
21            x: state
22        output
23            dx: differential output of ode
24
25    """
26    # Extract state
27    x1, x2, x3, u = x
28    x           = np.array([x1,x2,x3])
29
30    # Calculate control
31    k = np.array([10, 10, 5])
32    u = -k@x[0:4]
33
34    # Calculate ODEs
35    xd1 = x1 + x2
36    xd2 = x1*x2**2 - x1 + x3
37    xd3 = u

```

```

38     # Bundle ODEs
39     dx = [xd1, xd2, xd3, u]
40
41     return dx
42
43
44 #####=====
45 # Script
46
47 # Plotting/Calculating variables
48
49 ## Time horizon
50 t0 = 0.0
51 tf = 15.0
52 t = [t0, tf]
53
54 ## Initial conditions
55 x0 = [0.5, 0.1, 0.1, 0]
56
57
58 # Solve ode model
59 sol = solve_ivp(model, t, x0, method='RK45', dense_output=True)
60
61 # Plot solution
62 plot(sol, tf=tf, title="12.2.2.2", legend=['x1', 'x2', 'u'])

```

---

### 14.31)

---

```

1 #!/bin/python
2
3 ####=====
4 # Libraries
5 import matplotlib.pyplot as plt
6 import numpy as np
7
8 from scipy.integrate import solve_ivp
9
10 from plot import plot
11
12 ####=====
13 # Functions
14
15 ##-----
16 #
17 def model(t,x):
18     """
19         input
20             t: time step
21             x: state
22         output
23             dx: differential output of ode
24     """
25
26     # Extract states
27     x1, x2, u = x

```

```

28     a      = 0
29
30     # Calculate z
31     z = x1 + x2 + a + (x1 - a**1/3)**3 + x1
32
33     # Calculate control
34     u = -x1 - x2 - 3*(x1**2)*x2 - z
35
36     # Calculate ODEs
37
38     xd1 = x2 + a + (x1 - a**1/3)**3
39     xd2 = x1 + u
40
41     # Bundle ODEs
42     dx = [xd1, xd2, u]
43
44     return dx
45
46 #####=====
47 # Script
48
49 # Plotting/Calculating variables
50
51 ## Time horizon
52 t0 = 0.0
53 tf = 8.0
54 t = [t0, tf]
55
56 ## Initial conditions
57 x0 = [0.5, 0.6, 0]
58
59
60 # Solve ode model
61 sol = solve_ivp(model, t, x0, method='RK45', dense_output=True)
62
63 # Plot solution
64 plot(sol, tf=tf, title="14.31", legend=['x1', 'x2', 'u'])

```

---

## 14.34)

---

```

1 #####=====
2 # Libraries
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 from scipy.integrate import solve_ivp
7
8 from plot import plot
9
10 ####=====
11 # Functions
12
13 ##
14 #
15 def model(t,x):

```

```

16     """
17     input
18         t: time step
19         x: state
20     output
21         dx: differential output of ode
22
23     """
24     # Extract states
25     x1, x2, x3, u = x
26
27     # Calculate z
28     z = x2
29
30     # Calculate control
31     u = -x1 + x2 + x1*x3 - z
32
33     # Calculate ODEs
34     xd1 = -x1 + x2
35     xd2 = x1 - x2 - x1*x3 + u
36     xd3 = x1 + x1*x2 - 2*x3
37
38     # Bundle ODEs
39     dx = [xd1, xd2, xd3, u]
40
41     return dx
42
43 #####=====
44 # Script
45
46 # Plotting/Calculating variables
47
48 ## Time horizon
49 t0 = 0.0
50 tf = 8.0
51 t = [t0, tf]
52
53 ## Initial conditions
54 x0 = [0.1, 0.2, 0.3, 0]
55
56
57 # Solve ode model
58 sol = solve_ivp(model, t, x0, method='RK45', dense_output=True)
59
60 # Plot solution
61 plot(sol, tf=tf, title="14.34", legend=['x1', 'x2', 'x3', 'u'])

```

---

### 14.15 a)

---

```

1 #!/bin/python
2
3 #####=====
4 # Libraries
5 import matplotlib.pyplot as plt
6 import numpy as np

```

```

7
8  from scipy.integrate import solve_ivp
9
10 from plot import plot
11
12 #####=====
13 # Functions
14
15 ####-
16 #
17 def model(t,x):
18     """
19         input
20             t: time step
21             x: state
22         output
23             dx: differential output of ode
24
25     """
26     # Extract states
27     x1, x2, u = x
28
29     # Calculate control
30     k = 0.25
31     u = -k*x2
32
33     # Calculate ODEs
34
35     xd1 = x2
36     xd2 = -x1**3 + u
37
38     # Bundle ODEs
39     dx = [xd1, xd2, u]
40
41     return dx
42
43 #####=====
44 # Script
45
46 # Plotting/Calculating variables
47
48 ## Time horizon
49 t0 = 0.0
50 tf = 100
51 t = [t0, tf]
52
53 ## Initial conditions
54 x0 = [0.5, 0.6, 0]
55
56
57 # Solve ode model
58 sol = solve_ivp(model, t, x0, method='RK45', dense_output=True)
59
60 # Plot solution
61 plot(sol, tf=tf, title="14.15_a", legend=['x1', 'x2', 'u'])

```

---

## 14.15 b)

---

```
1 #!/bin/python
2
3 ##### Libraries
4 import matplotlib.pyplot as plt
5 import numpy as np
6
7 from scipy.integrate import solve_ivp
8
9 from plot import plot
10
11 #####
12 # Functions
13
14 #####
15 #####
16 #
17 def model(t,x):
18     """
19         input
20             t: time step
21             x: state
22         output
23             dx: differential output of ode
24
25     """
26     # Extract states
27     x1, x2, u = x
28
29     # Calculate control
30     k = 0.25
31     u = -(2*k/np.pi)*np.arctan(x2)
32
33     # Calculate ODEs
34
35     xd1 = x2
36     xd2 = -x1**3 + u
37
38     # Bundle ODEs
39     dx = [xd1, xd2, u]
40
41     return dx
42
43 #####
44 # Script
45
46 # Plotting/Calculating variables
47
48 ## Time horizon
49 t0 = 0.0
50 tf = 100
51 t = [t0, tf]
52
53 ## Initial conditions
54 x0 = [0.1, 0.1, 0.0]
```

```

55
56
57 # Solve ode model
58 sol = solve_ivp(model, t, x0, method='RK45', dense_output=True)
59
60 # Plot solution
61 plot(sol, tf=tf, title="14.15_b", legend=['x1', 'x2', 'u'])

```

---

## Plotter

---

```

1 #!/bin/python
2
3 ##### Libraries
4 import matplotlib.pyplot as plt
5 import numpy as np
6
7
8 #####
9 #
10 def plot(solution,
11           tf      : float   = 15,
12           x_lbl   : str     = "t",
13           legend  : np.array = ['x','y'],
14           title   : str     = "Model",
15           show_plot: bool    = False):
16     """
17     Plotting module for ODE's.
18
19     input
20         solution: Solution from ode
21
22     output
23         NONE
24     """
25
26
27     # Extract
28     t = np.linspace(0,tf,300)
29     z = solution.sol(t)
30
31     # Plot
32     plt.plot(t, z.T)
33     plt.xlabel(x_lbl)
34     plt.legend(legend, shadow=True)
35     plt.title(title)
36
37     # Determine whether to display or save plot
38     if show_plot:
39         plt.show()
40     else:
41         plt.savefig("img/"+title+".png")
42
43     return

```

---

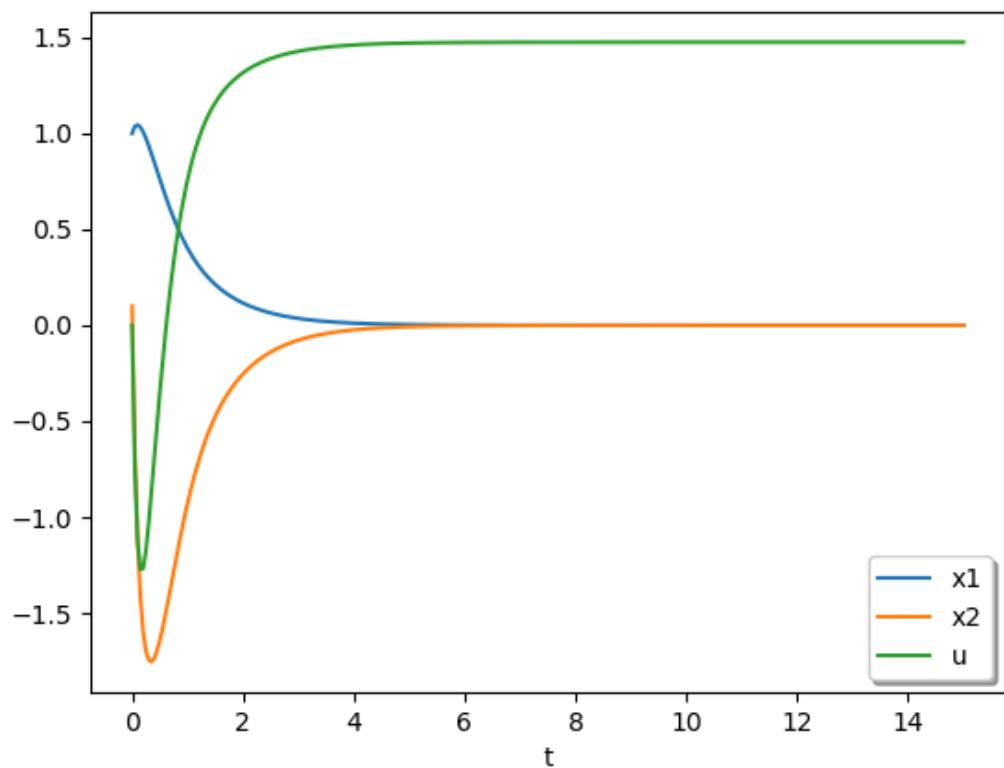
## Script Runner

---

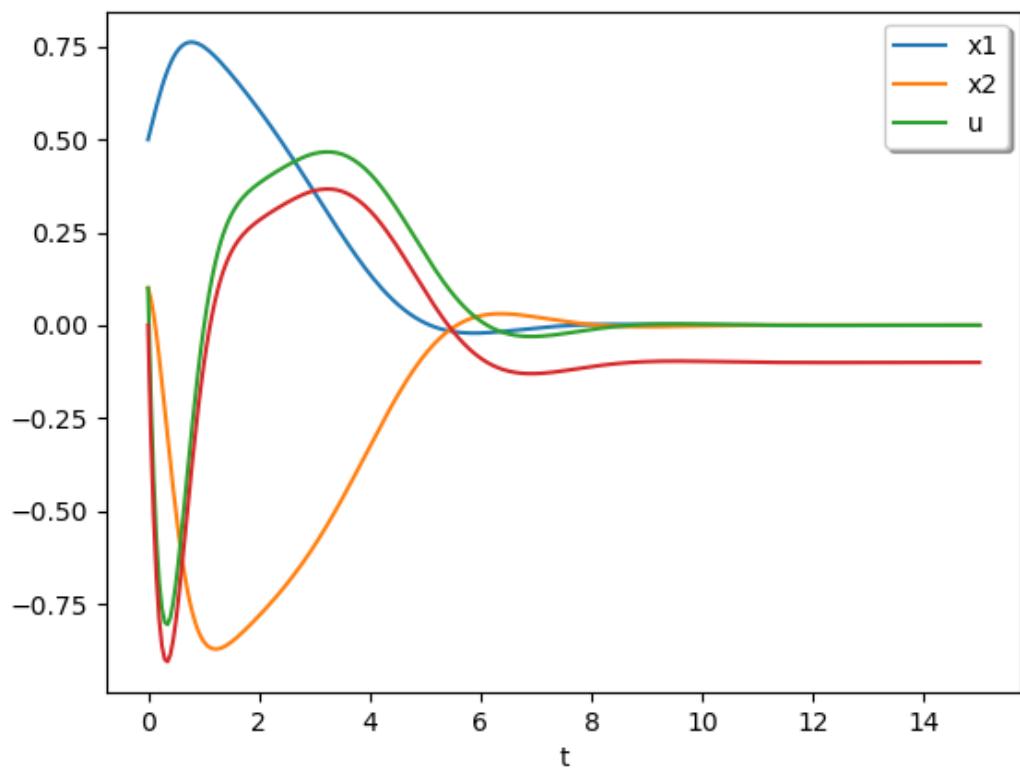
```
1 #!/bin/sh
2
3 # Create a fancy little progress bar
4 # https://stackoverflow.com/questions/238073/how-to-add-a-progress-bar-to-a-shell-script
5 function ProgressBar
6 {
7     # Process data
8     let _progress=(${1}*100/${2}*100)/100
9     let _done=(${_progress}*4)/10
10    let _left=40-$_done
11
12    # Build progressbar string lengths
13    _fill=$(printf "%${_done}s")
14    _empty=$(printf "%${_left}s")
15
16    # 1.2 Build progressbar strings and print the ProgressBar line
17    # 1.2.1 Output example:
18    # 1.2.1.1 Progress : [#####
19    printf "\rProcessing $s: [${_fill// /#}${_empty// /-}] ${_progress}%% \r"
20 }
21 # Variables
22 i=1
23
24 # Create list of scripts to be ran
25 declare -a scripts=(`find . -type f -name "*.py" ! -name "plot.py"`)
26
27 # Loop through each script
28 for s in "${scripts[@]}"
29 do
30     ## Update progress bar
31     ProgressBar $i ${#scripts[@]}
32
33     ## Process script
34     python $s
35
36     ## Update Index
37     i=$((i+1))
38 done
39
40 printf '\nFinished!\n'
```

---

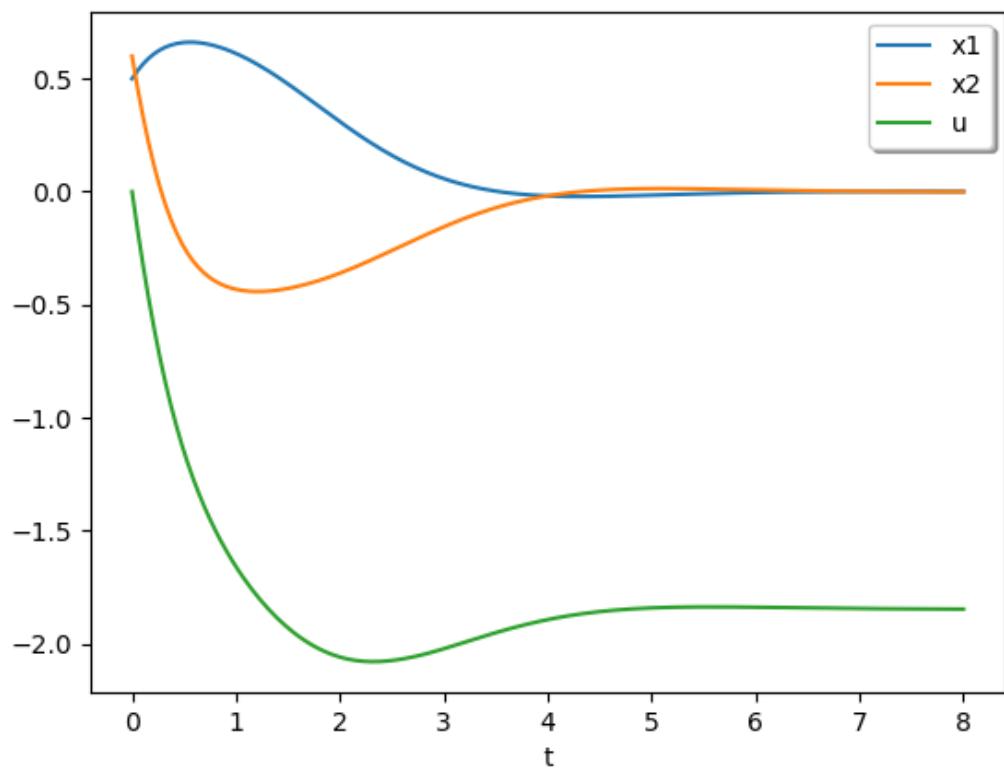
12.2.2.1



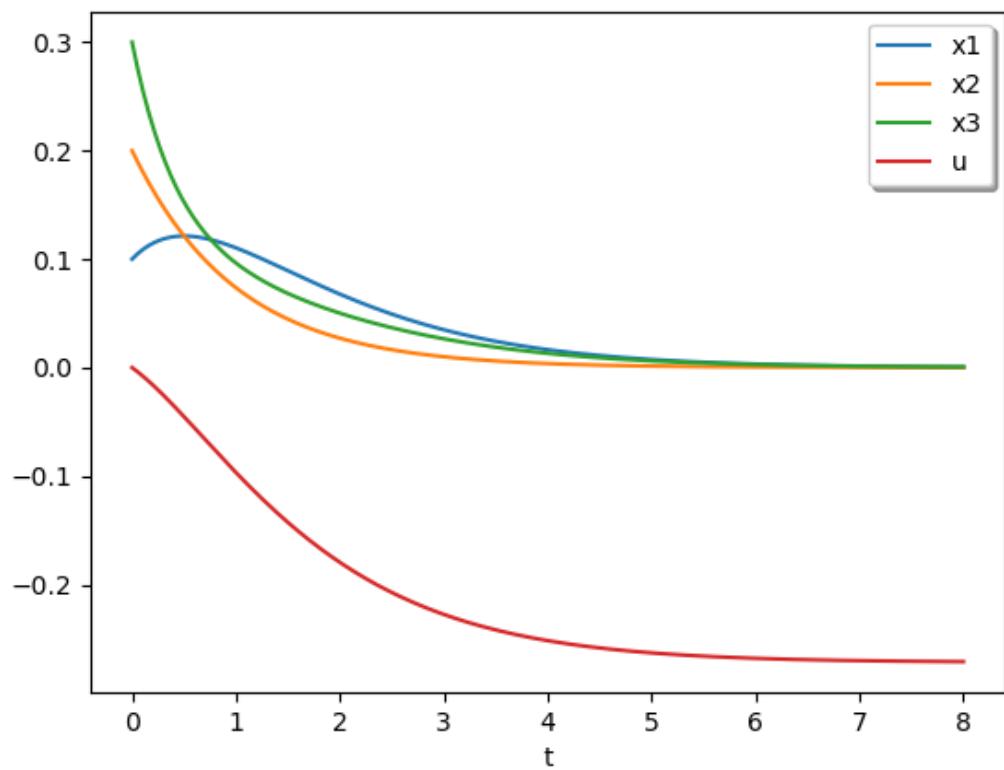
12.2.2.2



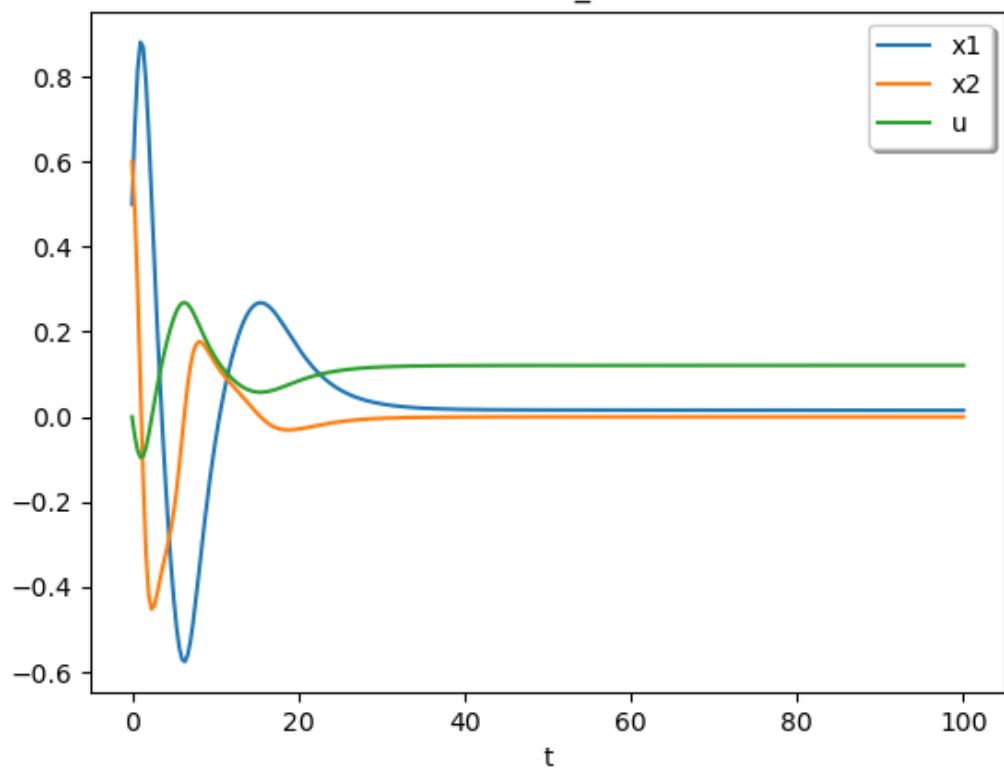
14.31

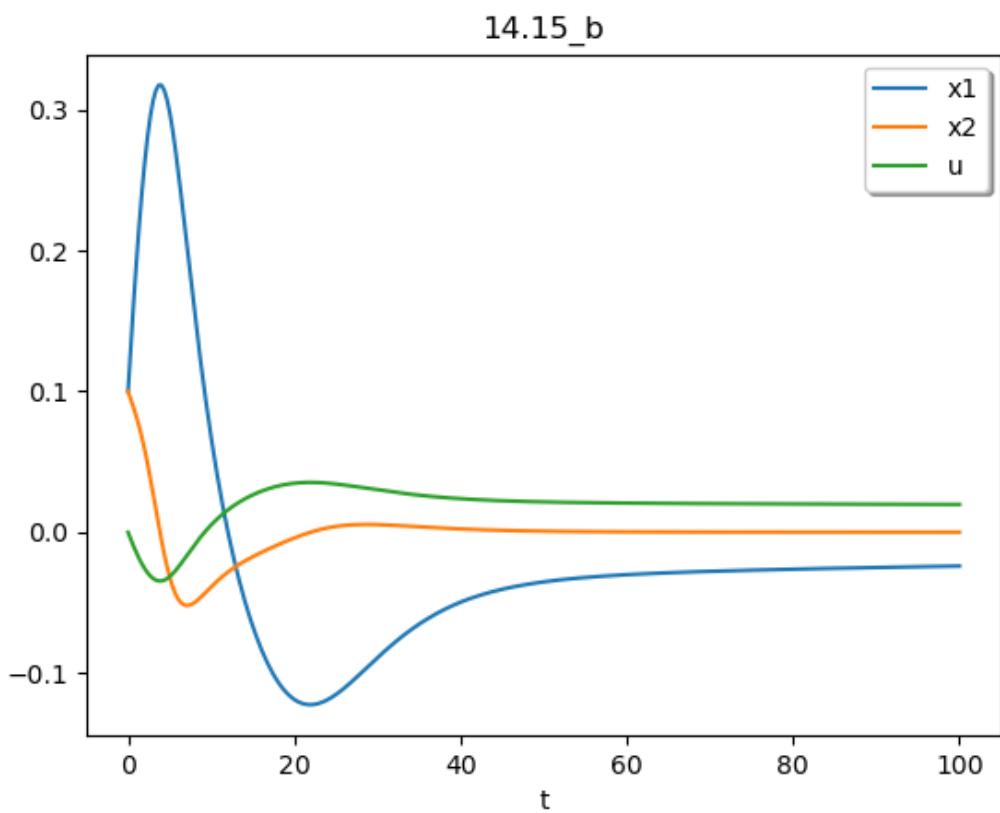


14.34



14.15\_a





### A note on 14-15-a and 14-15-b

Comparing the two plots, it can be noted that a has a better state response with more control applied.  
b converges slower, but has less overshoot with and less control applied.

(x) such that  $\|\psi(x)\| < k \forall x$  where  $k$  is a positive constant.

---

**Solution:**

Begin by defining the linear system

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx\end{aligned}$$

Let  $V(x) = x^T Px$ , therefore

$$\begin{aligned}\dot{V}(x) &= x^T(P + P^T)\dot{x} \\ &= x^T(PA + P^TA)x + x^T(PB + P^TB)u \\ &= x^T(PA + A^TP)x + x^T(PB + B^TP)u \\ &= x^T(PA + A^TP)x + x^T(PB)u = -B^TPxu\end{aligned}$$

Let  $C = B^TPx$ , therefore

$$\dot{V} = x^T(PA + A^TP)x + x^T(PB)u = yx$$

Which means signifies that the system is passive. Furthermore, setting  $u = 0$  and stating  $x = 0 \implies y = 0$  which means that the system is zero state observable. Therefore we choose the control to by  $u = -ky$  where  $k > 0$ .

---

---