12.2.2.1)

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.

$$\dot{x} = \underbrace{\begin{bmatrix} x_1 + x_2 \\ 3x_1^2 x_2 + x_1 \end{bmatrix}}_{f(x)} + \underbrace{\begin{bmatrix} 0 \\ u \end{bmatrix}}_{g(x)}$$

Note that: $n = 2$ for this problem

$$ad_f^0 g(x) = g(x)$$

$$ad_f g(x) = [f, g](x) = \nabla g(x) f(x) - \nabla f(x) g(x)$$

$$ad_f^2 g(x) = [f, ad_f g](x)$$

$$\vdots$$

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

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

and

$$\Delta := \text{span}\{g(x), ad_f g\} \in D$$

$\Rightarrow$ Involutive

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

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

$L_g L_f h \neq 0 \Rightarrow \nabla(\nabla h(x) f(x)) g(x) \neq 0$

$h(0) = 0$

$$\nabla h(x) g(x) = \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{\partial h}{\partial x_2} = 0$$

$h(x)$ is independent of x_2

$$\Rightarrow \nabla h(x) f(x) = \begin{bmatrix} \frac{\partial h}{\partial x_1} & 0 \end{bmatrix} \begin{bmatrix} x_1 + x_2 \\ 3x_1^2 x_2 + x_1 \end{bmatrix} = \frac{\partial h}{\partial x_1}(x_1 + x_2)$$

$$\Rightarrow \begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_1} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{\partial h}{\partial x_1} \neq 0$$

Choose $h(x) = x_1 \Rightarrow h(0) = 0$

$z_1 = h(x) = x_1 \; ; \; z_2 = \dot{z}_1 = \dot{x}_1 = x_1 + x_2$

$\Rightarrow \dot{z}_1 = z_2 = \underbrace{x_1 + x_2}_{\bar{u}} \; ; \; \dot{z}_2 = \dot{x}_1 + \dot{x}_2 = x_1 + x_2 + 3x_1^2 x_2 + x_1 + u$

$= 3x_1^2 x_2 + 2x_1 + x_2 + u$

$u = -(3x_1^2 x_2 + 2x_1 + x_2) + v$

We are left w/

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

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

$V = -kz$ and let $k = [10 \;\; 10]$

$\Rightarrow \quad \dot{z} = (A - Bk)z \quad \Rightarrow \quad \text{eig}(A - Bk) = -1,12 \;,\; -8,87$

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

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 (12.2.3.1)

Solution:

This time lets try linearization

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

$$\frac{\partial f}{\partial x} = \underbrace{\begin{bmatrix} 1 & 1 & 0 \\ x_2^2 - 1 & 2x_1x_2 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{A}\delta x + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{B} V \Big|_{\bar{x}=[0]}$$

$$\Rightarrow \dot{\bar{x}} = \underbrace{\begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{A}\bar{x} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{B} V$$

Let $V = -K\bar{x}$

and $K = [10 \quad 10 \quad 5]$

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

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

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

$g_a(\eta) = 1$ 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.

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

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

Where $\dot{x}_1 = -x_1 \rightarrow \dot{V}(x) = -x_1^2 \leq 0$

To backstep define

$$z := \xi - \phi(\eta) = x_2 + a + (x_1 - a^{1/3})^3 + x_1$$

$$= x_1 + x_2 + 3a^{2/3}x_1 + x_1^3 - 3a^{1/3}x_1^2$$

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

If $a = 0$

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

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

$$\Rightarrow \dot{V} = x_1\dot{x}_1 + z\dot{z} = x_1x_2 + z(x_2 + x_1 + u + 3x_1^2x_2)$$

Let $u = -x_1 - x_2 - 3x_1^2x_2 - z$

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

14.34)

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/ $\dot{x}_1$ and stepping back into $\dot{x}_2$, design a state feedback controller $u = \psi(x)$ s.t. $x = [0]$ is G.A.S.

b) Show that the controller found causes the system to be G.A.S.

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

Solution:

a)

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

$$f_a(\eta) = -x_1 \; ; \; g_a(\xi) = 1$$

$$f_b(\eta, \xi) = x_1 - x_2 - x_1 x_3 \; ; \; g_b(\eta, \xi) = 1$$

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

$$\Rightarrow \dot{\eta} = -x_1$$

Let $V(x) = \frac{1}{2}x_1^2 \Rightarrow \dot{V}(x) = x_1 \dot{x}_1 = \underline{-x_1^2 \leq 0}$

$$z := \xi - \phi(\eta) = x_2 - 0 = x_2$$

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

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

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

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

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

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

$\dot{x}_3 = x_1 + x_1x_3 - 2x_3$

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

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

$\Rightarrow \dot{V}(x) = \cancel{x_3x_1}^{\to 0} + \cancel{x_1x_2}^{\to 0}x_3 - 2x_3^2 \leq 0$

Therefore the system is G.A.S.

14.42)

Given:

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

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

Find:

A globally stabilizing feedback law $u = -\psi(x)$ such that $\|\psi(x)\| < k \; \forall x$ where k is given as a positive constant

Solution:

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

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

$$\Rightarrow \dot{V}(x) = x(P + P^T)\dot{x} = x^T(P + P^T)(Ax + Bu)$$

$$= x^T(PAx + PBu + P^T Ax + P^T Bu)$$

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

Note $A^T B = B^T A$ if symmetric

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

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

Let $B^T P x = C$ then

$x^T(PA + A^T P)x + x^T P B u \leq yu$ $\therefore$ The system is passive

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

$\therefore$ The system is zero state observable.

Choose $u = -ky$

14.43)

Given:

$\dot{x}_1 = x_2$

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

ψ is:

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

Find:

Design a globally stabilizing state feedback control

Solution:

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

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

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

$\Rightarrow -x_1 x_2 + x_1^3 x_2 = x_2 \psi(u)$ 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 $u = -ky^2$

Code/Plots

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

12.2.2.1)

```python
#!/bin/python

##============================================================================
# Libraries
import matplotlib.pyplot as plt
import numpy as np

from scipy.integrate import solve_ivp

from plot import plot

##============================================================================
# Functions

##----------------------------------------------------------------------------
#
def model(t,x):
    """
    input
        t: time step
        x: state
    output
        dx: differential output of ode

    """
    # Extract states
    x1, x2, u = x

    # Calculate z
    z1 = x1
    z2 = x1 + x2
    z  = np.array([[z1], [z2]])

    # Calculate control
    k = np.array([10, 10])
    u = -3*(x1**3)*x2 + 2*x1 + x2 - k@z

    # Calculate ODEs
    xd1 = x1 + x2
    xd2 = 3*(x1**2)*x2 + x1 + u

    # Bundle ODEs
    dx = [xd1, xd2, u]

    return dx

##============================================================================
# Script

# 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)

```python
#!/bin/python

##====================================================================================
# Libraries
import matplotlib.pyplot as plt
import numpy as np

from scipy.integrate import solve_ivp

from plot import plot

##====================================================================================
# Functions

##------------------------------------------------------------------------------------
#
def model(t,x):
    """
    input
        t: time step
        x: state
    output
        dx: differential output of ode

    """
    # Extract state
    x1, x2, x3, u = x
    x           = np.array([x1,x2,x3])

    # Calculate control
    k = np.array([10, 10, 5])
    u = -k@x[0:4]

    # Calculate ODEs
    xd1 = x1 + x2
    xd2 = x1*x2**2 - x1 + x3
    xd3 = u

    # Bundle ODEs
    dx = [xd1, xd2, xd3, u]

    return dx

##====================================================================================
# Script

# Plotting/Calculating variables

## Time horizon
t0 = 0.0
tf = 15.0
t = [t0, tf]

## Initial conditions
```

```
x0 = [0.5, 0.1, 0.1, 0]


# Solve ode model
sol = solve_ivp(model, t, x0, method='RK45', dense_output=True)

# Plot solution
plot(sol, tf=tf, title="12.2.2.2", legend=['x1', 'x2', 'u'])
```

14.31)

```
#!/bin/python

##====================================================================================
# Libraries
import matplotlib.pyplot as plt
import numpy as np

from scipy.integrate import solve_ivp

from plot import plot

##====================================================================================
# Functions

##------------------------------------------------------------------------------------
#
def model(t,x):
    """
    input
        t: time step
        x: state
    output
        dx: differential output of ode

    """
    # Extract states
    x1, x2, u = x
    a     = 0

    # Calculate z
    z = x1 + x2 + a + (x1 - a**1/3)**3 + x1

    # Calculate control
    u = -x1 - x2 - 3*(x1**2)*x2 - z

    # Calculate ODEs

    xd1 = x2 + a + (x1 - a**1/3)**3
    xd2 = x1 + u

    # Bundle ODEs
    dx = [xd1, xd2, u]

    return dx

##====================================================================================
# Script

# Plotting/Calculating variables

## Time horizon
t0 = 0.0
tf = 8.0
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)

```
##=====================================================================================
# Libraries
import matplotlib.pyplot as plt
import numpy as np

from scipy.integrate import solve_ivp

from plot import plot

##=====================================================================================
# Functions

##-------------------------------------------------------------------------------------
#
def model(t,x):
    """
    input
        t: time step
        x: state
    output
        dx: differential output of ode

    """
    # Extract states
    x1, x2, x3, u = x

    # Calculate z
    z = x2

    # Calculate control
    u = -x1 + x2 + x1*x3 - z

    # Calculate ODEs
    xd1 = -x1 + x2
    xd2 = x1 - x2 - x1*x3 + u
    xd3 = x1 + x1*x2 - 2*x3

    # Bundle ODEs
    dx = [xd1, xd2, xd3, u]

    return dx

##=====================================================================================
# Script

# Plotting/Calculating variables

## Time horizon
t0 = 0.0
tf = 8.0
t = [t0, tf]

## Initial conditions
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)

```python
#!/bin/python

##====================================================================================
# Libraries
import matplotlib.pyplot as plt
import numpy as np

from scipy.integrate import solve_ivp

from plot import plot

##====================================================================================
# Functions

##------------------------------------------------------------------------------------
#
def model(t,x):
    """
    input
        t: time step
        x: state
    output
        dx: differential output of ode

    """
    # Extract states
    x1, x2, u = x

    # Calculate control
    k = 0.25
    u = -k*x2

    # Calculate ODEs

    xd1 = x2
    xd2 = -x1**3 + u

    # Bundle ODEs
    dx = [xd1, xd2, u]

    return dx

##====================================================================================
# Script

# Plotting/Calculating variables

## Time horizon
t0 = 0.0
tf = 100
t = [t0, tf]

## Initial conditions
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)

```python
#!/bin/python

##====================================================================================
# Libraries
import matplotlib.pyplot as plt
import numpy as np

from scipy.integrate import solve_ivp

from plot import plot

##====================================================================================
# Functions

##------------------------------------------------------------------------------------
#
def model(t,x):
    """
    input
        t: time step
        x: state
    output
        dx: differential output of ode

    """
    # Extract states
    x1, x2, u = x

    # Calculate control
    k = 0.25
    u = -(2*k/np.pi)*np.arctan(x2)

    # Calculate ODEs

    xd1 = x2
    xd2 = -x1**3 + u

    # Bundle ODEs
    dx = [xd1, xd2, u]

    return dx

##====================================================================================
# Script

# Plotting/Calculating variables

## Time horizon
t0 = 0.0
tf = 100
t = [t0, tf]

## Initial conditions
x0 = [0.1, 0.1, 0.0]
```

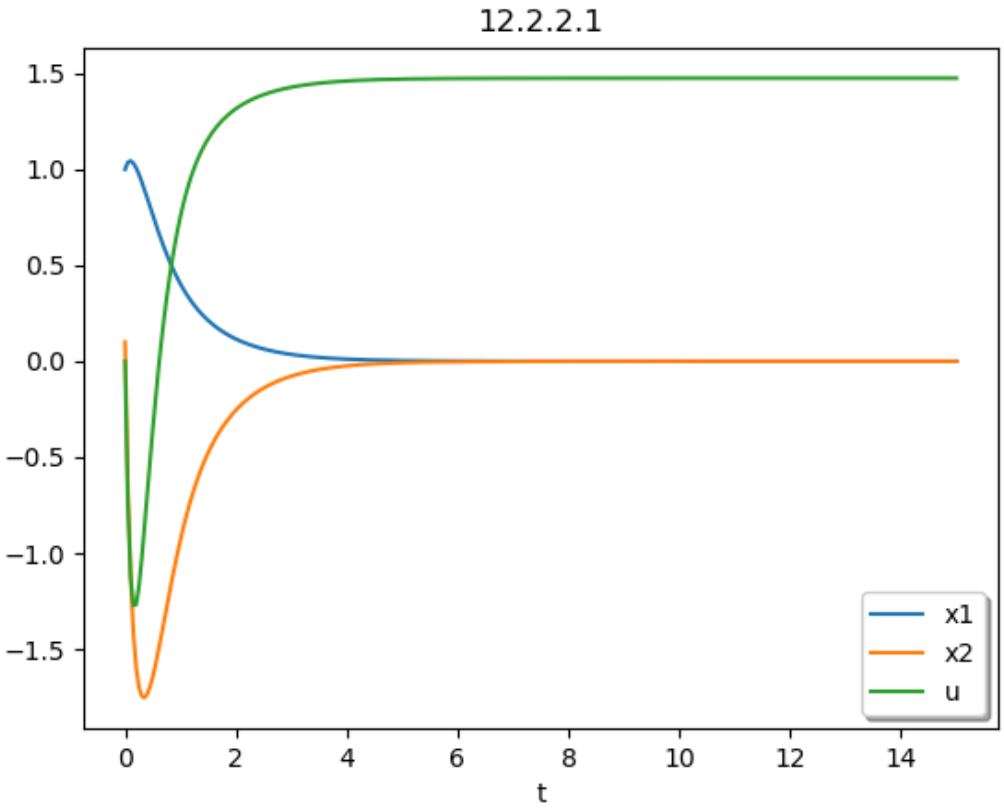
```
55
56
57 # Solve ode model
58 sol = solve_ivp(model, t, x0, method='RK45', dense_output=True)
59
60 # Plot solution
61 plot(sol, tf=tf, title="14.15_b", legend=['x1', 'x2', 'u'])
```

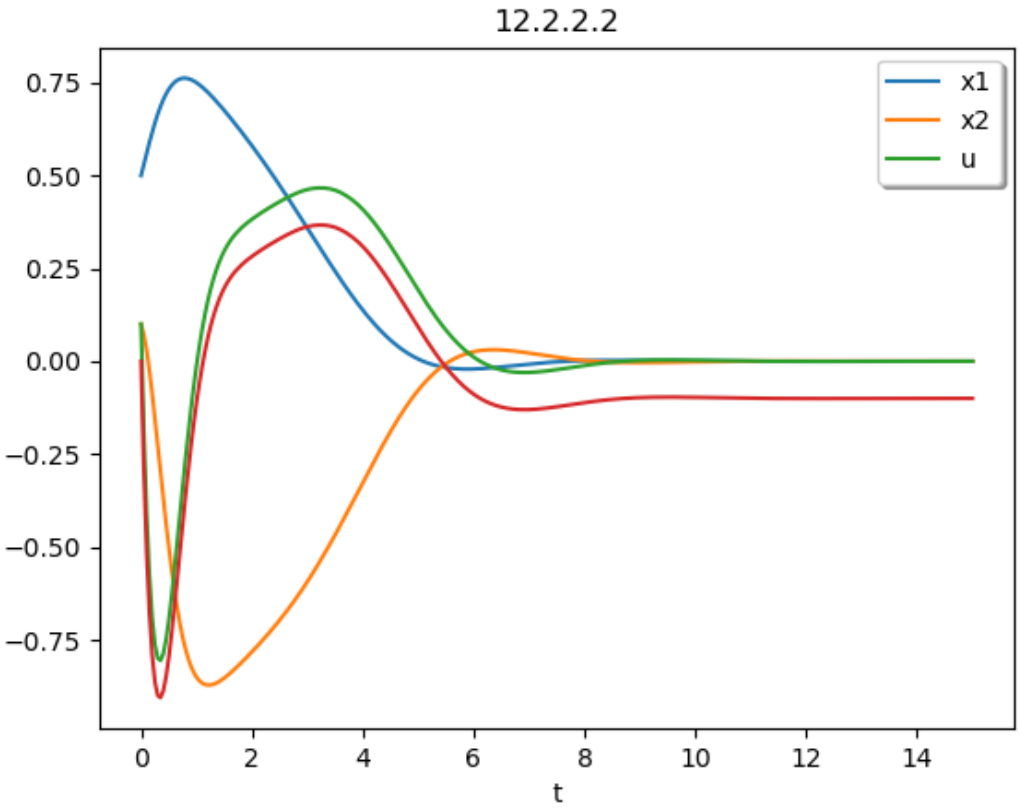
Plotter

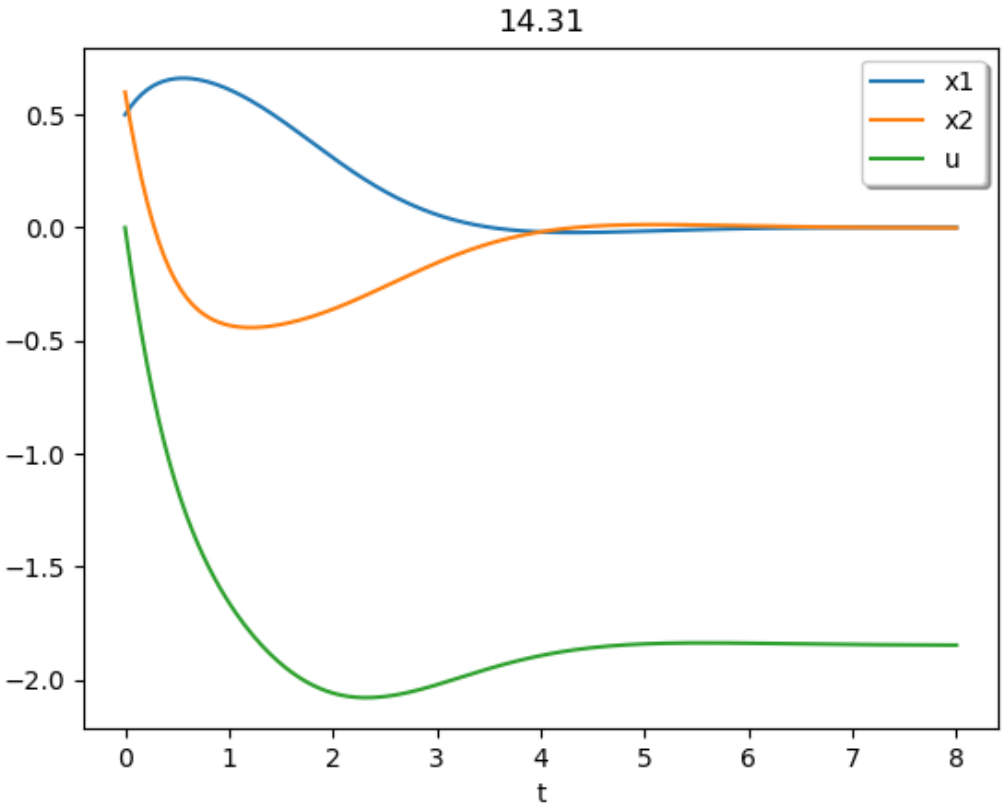
```python
#!/bin/python

##============================================================================
# Libraries
import matplotlib.pyplot as plt
import numpy as np


##============================================================================
#
def plot(solution,
        tf       : float   = 15,
        x_lbl    : str     = "t",
        legend   : np.array = ['x','y'],
        title    : str     = "Model",
        show_plot : bool  = False):
    """
    Plotting module for ODE's.

    input
        solution: Solution from ode

    output
        NONE
    """

    # Extract
    t = np.linspace(0,tf,300)
    z = solution.sol(t)

    # Plot
    plt.plot(t, z.T)
    plt.xlabel(x_lbl)
    plt.legend(legend, shadow=True)
    plt.title(title)

    # Determine whether to display or save plot
    if show_plot:
        plt.show()
    else:
        plt.savefig("img/"+title+".png")

    return
```

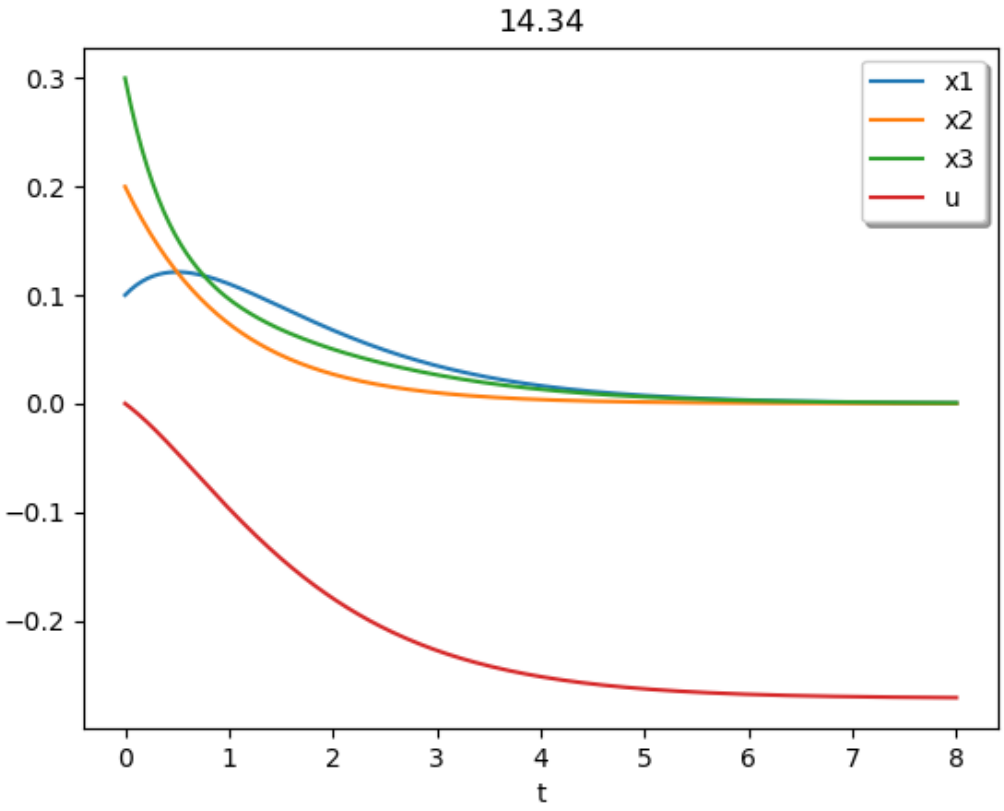
Script Runner

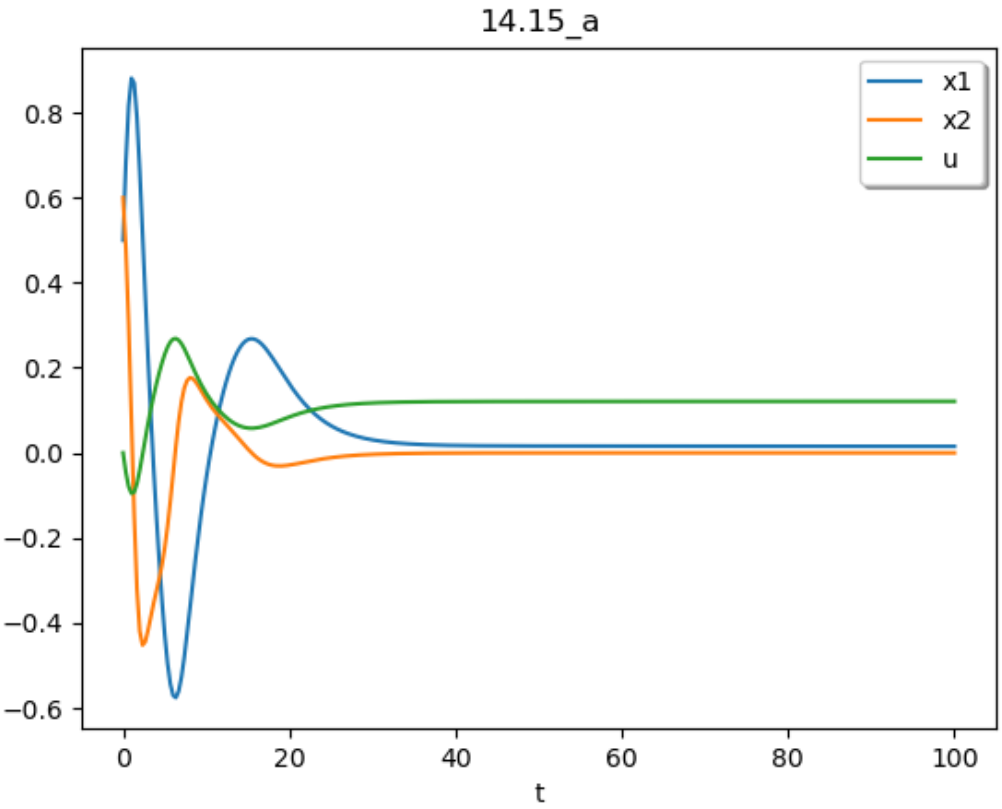
```
#!/bin/sh

# Create a fancy little progress bar
# https://stackoverflow.com/questions/238073/how-to-add-a-progress-bar-to-a-shell-script
function ProgressBar
{
    # Process data
    let _progress=(${1}*100/${2}*100)/100
    let _done=(${_progress}*4)/10
    let _left=40-$_done

    # Build progressbar string lengths
    _fill=$(printf "%${_done}s")
    _empty=$(printf "%${_left}s")

    # 1.2 Build progressbar strings and print the ProgressBar line
    # 1.2.1 Output example:
    # 1.2.1.1 Progress : [########################################] 100%
    printf "\rProcessing $s: [${_fill// /#}${_empty// /-}] ${_progress}%% \r"
}
# Variables
i=1

# Create list of scripts to be ran
declare -a scripts=(`find . -type f -name "*.py" ! -name "plot.py"`)

# Loop through each script
for s in "${scripts[@]}"
do
    ## Update progress bar
    ProgressBar $i ${#scripts[@]}

    ## Process script
    python $s

    ## Update Index
    i=$((i+1))
done

printf '\nFinished!\n'
```

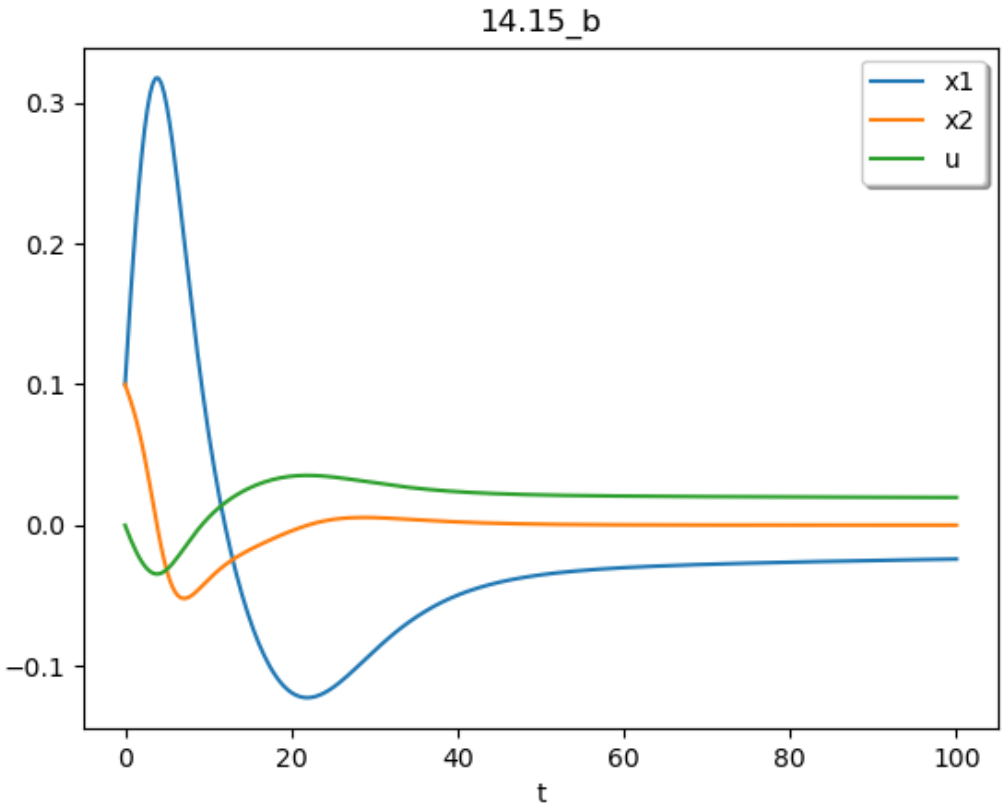

12.2.2.1
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
2
4
6
8
10
12
14
t
x1
x2
u


12.2.2.2
x1
x2
u
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
2
4
6
8
10
12
14
t


14.31
x1
x2
u
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5
6
7
8
t


14.34
0.3
0.2
0.1
0.0
−0.1
−0.2
0
1
2
3
4
5
6
7
8
t
x1
x2
x3
u


14.15_a
x1
x2
u
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20
40
60
80
100
t



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.

14.42

Given:

$$\dot{x} = Ax + Bu$$
$$PA + A^T P \leq 0$$
$$P > 0$$

Find:

A globally stabilizing feedback law $u = -\psi(x)$ such that $||\psi(x)|| < k \forall x$ where k is a positive constant.

Solution:

Begin by defining the linear system

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

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

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

Let $C = B^T P x$, therefore

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

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 $k > 0$.