ECE 450 - Homework #3

Collin Heist

September 12, 2019

1 ECE 450 - Homework #3

1.1 Package Imports

```
In [1]: import numpy as np
        import seaborn as sns
        import pandas as pd
        import matplotlib.pyplot as plt
```

1.2 Generic Function for a State Variable Solver

```
In [11]: def state_solver(A_matrix, B_matrix, force_function, initial_conditions,
                          time_range=[0, 10], dt=0.01):
             time_values = np.arange(time_range[0], time_range[1], dt)
             x_vals = np.array(initial_conditions)
             state_variables = [[] for _ in range(len(initial_conditions))]

             # Loop through each instance in time, calculate the state variable at that time
             for time in time_values:
                 if isinstance(force_function(time), np.ndarray):
                     x_vals = x_vals + dt * (A_matrix @ x_vals) + dt * (B_matrix @ force_functio
                 else:
                     x_vals = x_vals + (dt * (A_matrix @ x_vals)) + dt * (B_matrix * force_funct
                 for index, _ in enumerate(state_variables):
                     state_variables[index].append(x_vals[index])

             return state_variables, time_values
```

1.3 Generic Function for Creating `DataFrames` out of State Variables

```
In [3]: def create_combined_df(state_variables, time, names):
            df_list = []
            for state_var, name in zip(state_variables, names):
                df = pd.DataFrame(list(zip(time, state_var)), columns=["Time", "x(t)"])
                df["Solution"] = name
                df_list.append(df)
```

```
        return pd.concat(df_list, ignore_index=True, axis=0)
```

1.4 Problem 6.2.1

Perform a state space simulation of the following

$$\frac{d^2y(t)}{dt^2} + 3\frac{dy(t)}{dt} + 2y(t) = u(t)$$

Where $y(0) = 1$ and $y'(0) = 2$

1.4.1 Solution

Find the state space equation of the above differential equation. Start by choosing: $x_1 = y, x_2 = \dot{x}_1 = \dot{y}$. This generates the following:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ f(t) \end{bmatrix}, where \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

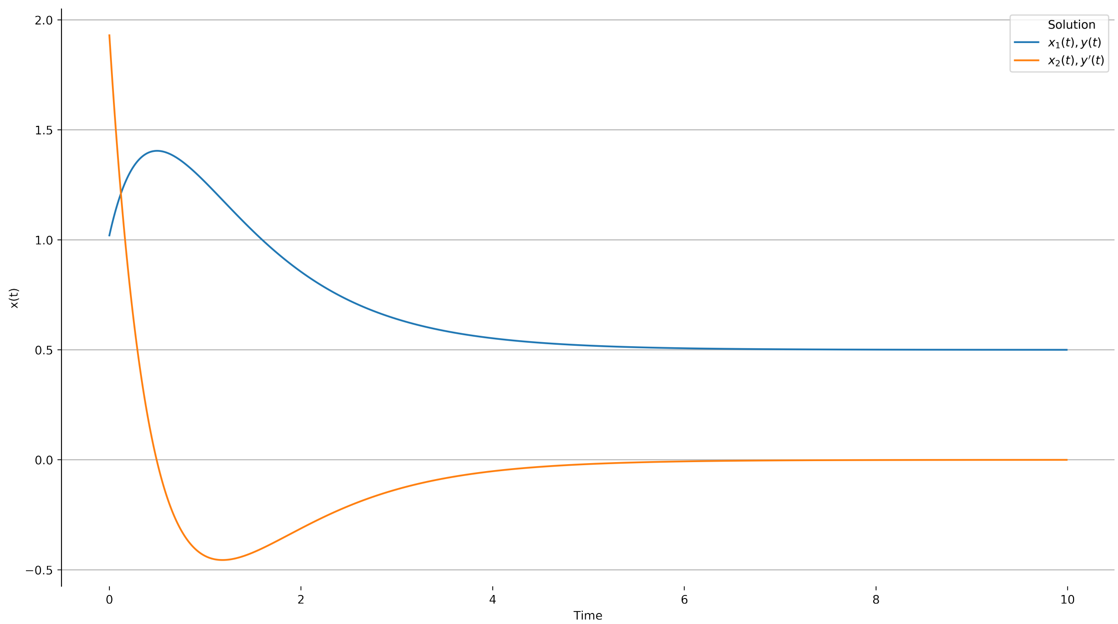
Where $f(t) = u(t)$

Perform the finite different simulation

```
In [4]: A_matrix = np.array([[0, 1], [-2, -3]])
        B_matrix = np.array([0, 1])
        init_state = [1, 2]
        force_function = lambda x: 0 if x < 0 else 1

        state_vars, time = state_solver(A_matrix, B_matrix, force_function, init_state)
```

Plot the state space simulation

```
In [5]: prob1_names = ["$x_1(t), y(t)$", "$x_2(t), y'(t)$"]
        prob1_df = create_combined_df(state_vars, time, prob1_names)

        fig = plt.figure(figsize=(16, 9), dpi=500)
        ax = sns.lineplot(x="Time", y="x(t)", data=prob1_df, hue="Solution")
        ax.grid(False)
        ax.yaxis.grid(True)
        sns.despine(top=True, right=True, bottom=True)
```



1.5 Problem 6.2.2

Simulate the response system described by

$$H(s) = \frac{2s^2 + 8s + 6}{s^3 + 8s^2 + 16s + 6}$$

to the input: $f(t) = u(t) - u(t-2)$.

1.5.1 Solution

Start by ignoring the numerator of the transfer function. Then, cross multiply the definition of $H(s) = \frac{Y(s)}{X(s)}$:

$$X(s) = Y(s) \cdot (s^3 + 8s^2 + 16s + 6)$$

Now, take the inverse laplace of the above equation and substitute in the following state variables:

$$x(t) = \frac{d^3y(t)}{dt^3} + 8\frac{d^2y(t)}{dt^2} + 16\frac{dy(t)}{dt} + 6y(t)$$
$$x_1 = y(t) \rightarrow \dot{x}_1 = x_2$$
$$x_2 = \frac{dy(t)}{dt} \rightarrow \dot{x}_2 = x_3$$
$$x_3 = \frac{d^2(t)}{dt^2} \rightarrow \dot{x}_3 = -6x_1 - 16x_2 - 8x_3 + f(t)$$

Using these state variables, the A, and B matrices can be created and a simulation can be run:

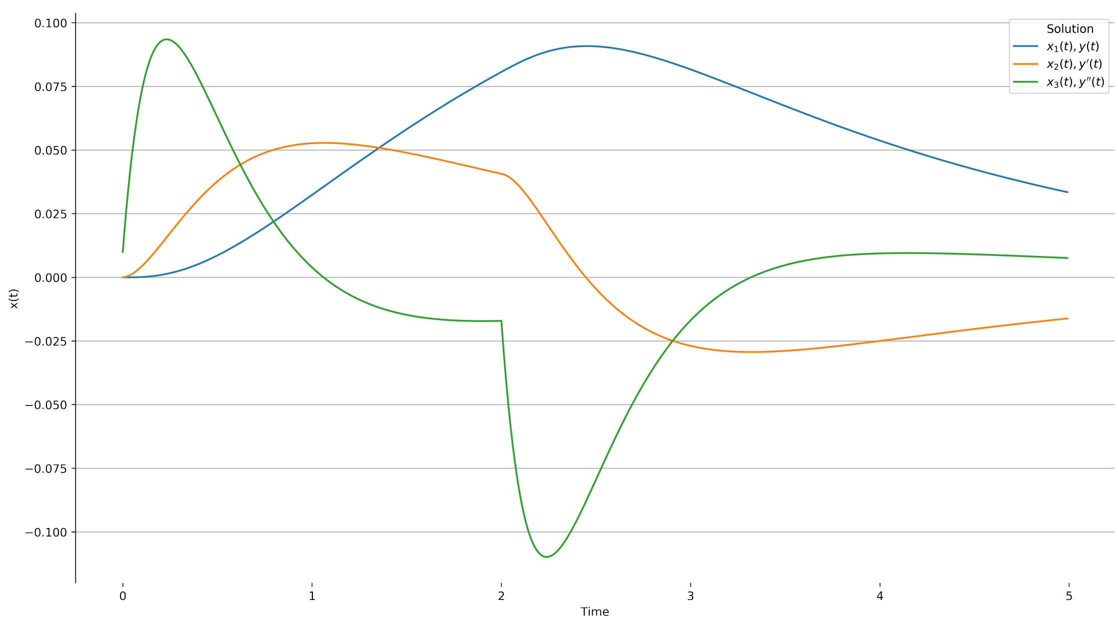
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -6 & -16 & -8 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ f(t) \end{bmatrix}, where \begin{bmatrix} x_1(0) \\ x_2(0) \\ x_3(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Compute the state variables over time

```
In [18]: A_matrix = np.array([[0, 1, 0], [0, 0, 1], [-6, -16, -8]])
         B_matrix = np.array([0, 0, 1])
         init_state = [0, 0, 0]
         input_function = lambda x: 0 if x < 0 or x > 2 else 1

         state_vars, time = state_solver(A_matrix, B_matrix, input_function, init_state, [0, 5])
```

Plot the state variables over time

```
In [19]: prob2_names = ["$x_1(t), y(t)$", "$x_2(t), y'(t)$", "$x_3(t), y''(t)$"]
         prob2_df = create_combined_df(state_vars, time, prob2_names)

         fig = plt.figure(figsize=(16, 9), dpi=500)
         ax = sns.lineplot(x="Time", y="x(t)", data=prob2_df, hue="Solution")
         ax.grid(False)
         ax.yaxis.grid(True)
         sns.despine(top=True, right=True, bottom=True)
```

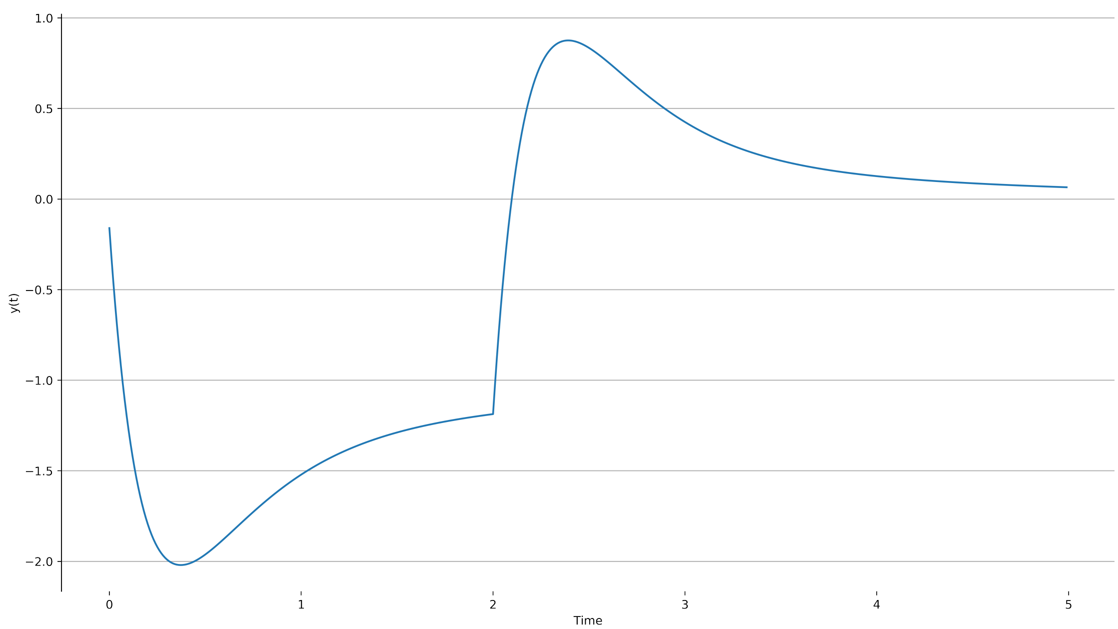


Now, to find the output function. This output function needs to account for the denominator that was ignored earlier. The above output will be referred to as y_0. Using this:

$$2s^2 + 8s + 6 \rightarrow y_{new} = 2\ddot{y}_0 + 8\dot{y}_0 + 6y_0$$
$$y_0 = x_1, \dot{y}_1 = x_2, \ddot{y}_2 = -6x_1 - 16x_2 - 8x_3$$
$$y_{new} = 2(-6x_1 - 16x_2 - 8x_3) + 8(x_2) + 6(x_1)$$
$$y_{new} = -6x_1 - 24x_2 - 16x_3$$

Define and plot the output function

```
In [20]: output_func = lambda states: [-6*x1 - 24*x2 - 16*x3 for x1, x2, x3 in zip(states[0], st

         output = output_func(state_vars)
         output_df = pd.DataFrame(list(zip(time, output)), columns=["Time", "y(t)"])

         fig = plt.figure(figsize=(16, 9), dpi=500)
         ax = sns.lineplot(x="Time", y="y(t)", data=output_df)
         ax.grid(False)
         ax.yaxis.grid(True)
         sns.despine(top=True, right=True, bottom=True)
```



1.6 Problem 6.2.4

1.6.1 Solution

Starting at the beginning of the system, with $H_1(s)$, where the output of that block is X_2. I chose Y_3 as the state variable for this block:

$$X_2 = F(s) \cdot \frac{1}{s+a}$$
$$\dot{X}_2 + aX_2 = F$$
$$Y_3 = X_2 \rightarrow \dot{Y}_3 = \dot{X}_2 = -aY_3 + F(s)$$

Now for $H_2(s)$, whose output is X_3 and the state variable I use is Y_4:

$$X_3 = G(s) \cdot \frac{s+1}{s+b}$$
$$X_3^0 = \frac{G(s)}{s+b}$$
$$\dot{X}_3^0 + bX_3^0 = G(s)$$
$$Y_4 = X_3^0 \rightarrow \dot{Y}_4 = \dot{X}_3^0 = -bY_4 + G(s)$$

Now, to account for the (previously ignored) numerator in $H_2(s)$, X_3 needs to be mapped from the state variable, Y_4.

$$X_3 = \dot{Y}_4 + Y_4 \rightarrow X_3 = Y_4(1-b) + G(s)$$

Finally, $H_3(s)$, whose output is X_1, and it's corresponding state variables are Y_1, Y_2:

$$X_1 = \frac{s+c}{s^2+ds+e} \cdot (X_2 + X_3)$$
$$X_1^0 = \frac{X_2 + X_2}{s^2+ds+e}$$
$$\ddot{X}_1^0 + d\dot{X}_1^0 + eX_1^0 = X_2 + X_3$$
$$Y_1 = X_1^0 \rightarrow \dot{Y}_1 = Y_2$$
$$Y_2 = \dot{X}_1^0 \rightarrow \dot{Y}_2 = -eY_1 - dY_2 + (X_2 + X_3) \rightarrow \dot{Y}_2 = -eY_1 - dY_2 + Y_3 + Y_4(1-b) + G(s)$$

With all of these equations, the following state space matrices can be created:

$$\begin{bmatrix} \dot{Y}_1 \\ \dot{Y}_2 \\ \dot{Y}_3 \\ \dot{Y}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -0.03 & -0.05 & 1 & 0.98 \\ 0 & 0 & -0.01 & 0 \\ 0 & 0 & 0 & -0.02 \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & g(t) \\ f(t) & 0 \\ 0 & g(t) \end{bmatrix}$$

Where the output of the entire system is:

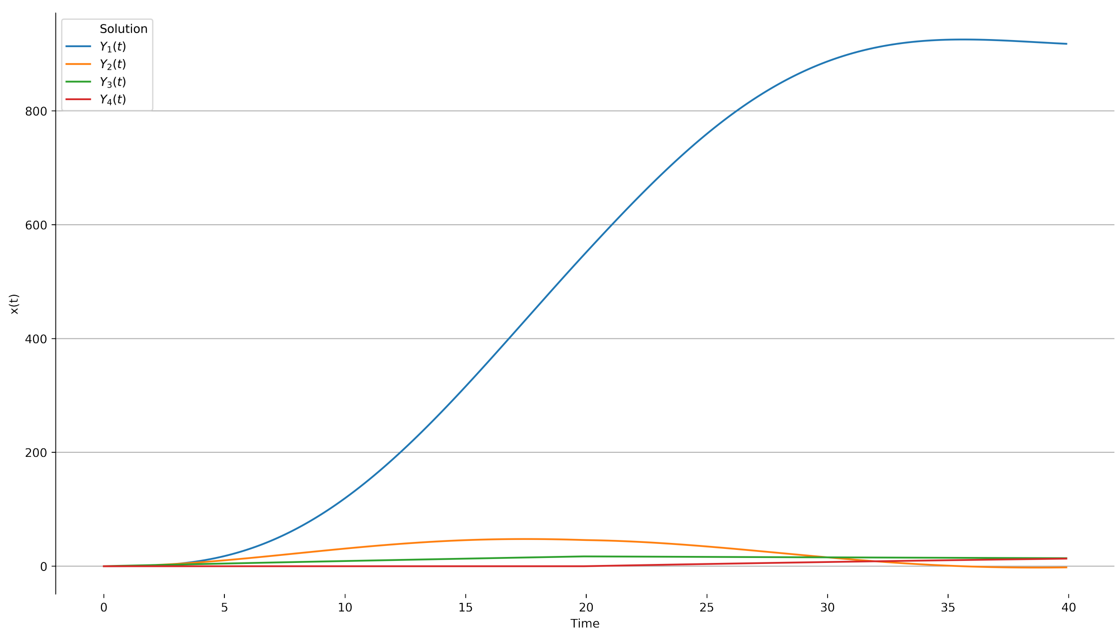
$$y = \begin{bmatrix} 0.1 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} + 0$$

Compute the state variables over time

In [12]:
```
A_matrix = np.array([[0,      1,     0,     0],
                     [-0.03, -0.05, 1,     0.98],
                     [0,      0,    -0.01, 0],
                     [0,      0,     0,    -0.02]])
B_matrix = np.array([[0, 0], [0, 1], [1, 0], [0, 1]])
init_state = [0, 0, 0, 0]

def input_function(t):
    step = lambda time: 0 if time < 0 else 1
    f = np.sin(2 * np.pi * 0.3) * (step(t) - step(t - 20))
    g = np.sin(2 * np.pi * 0.15) * (step(t - 20) - step(t - 40))

    return np.array([f, g])

state_vars, time = state_solver(A_matrix, B_matrix, input_function, init_state, [0, 40]
```

Plot the state variables over time

In [13]:
```
prob3_names = ["$Y_1(t)$", "$Y_2(t)$", "$Y_3(t)$", "$Y_4(t)$"]
prob3_df = create_combined_df(state_vars, time, prob3_names)

fig = plt.figure(figsize=(16, 9), dpi=500)
ax = sns.lineplot(x="Time", y="x(t)", data=prob3_df, hue="Solution")
ax.grid(False)
ax.yaxis.grid(True)
sns.despine(top=True, right=True, bottom=True)
```



Define and plot the output function

```
In [14]: output_func = lambda states: [0.1*x1 +x2 for x1, x2, x3 in zip(states[0], states[1], st

         output = output_func(state_vars)
         output_df = pd.DataFrame(list(zip(time, output)), columns=["Time", "y(t)"])

         fig = plt.figure(figsize=(16, 9), dpi=500)
         ax = sns.lineplot(x="Time", y="y(t)", data=output_df)
         ax.grid(False)
         ax.yaxis.grid(True)
         sns.despine(top=True, right=True, bottom=True)
```