Logan Shaffer MA345: Honors Project April 24, 2024

1 Statement of Purpose:

The purpose of this project is to model the IVP differential equation for the mass spring system: $x'' + 3x' + 2x = 0$, $x(0) = x_0$, $x'(0) = x_1$. This project sought to set-up a general homogeneous solution with the initial conditions, then plugging in $x(0) = x_0$, $x'(0) = 1$ and graphing with three different x_0 with given initial velocity. Solving the non-homogeneous portion letting $f(t) = 4cost$, finding a general solution $y = y_h + y_p$ and letting $x(0) = x_0$, $x'(0) = 1$ and graphing with three different x_0. The solution $y = y_h + y_p$ is also to be graphed after letting $x(0) = 2$, $x'(0) = x_1$ using four different x_1 with the given initial displacement conditions. Finally, the homogeneous and non-homogeneous portions are to be solved using Laplace Transformations for letting $x(0) = 2$, $x'(0) = 1$.

2 Homogeneous Solution:

To find a homogeneous solution, I first found the auxiliary equation of $x'' + 3x' + 2x = 0$ to be $r'' + 3r' + 2r = 0$. Then solving for r, I found there were two real solutions of $r = -2, -1$. After, I plugged the r values into the homogeneous equation of $x_h = C_1e^{rt} + C_2e^{rt}$ to get $x_h = C_1e^{-2t} + C_2e^{-t}$. I applied the initial conditions of $x(0) = x_0$, $x'(0) = x_1$ to the homogeneous solution and solved for C_1 and C_2.

$$x_h' = -2C_1e^{-2t} - C_2e^{-t}$$
$$x(0) = x_0 = C_1e^0 + C_2e^0$$

$$x_0 = C_1 + C_2 \qquad x_1 = 2C_1 - C_2$$
$$C_1 = x_0 - C_2 \qquad C_2 = -2C_1 - x_1$$
$$C_1 = x_0 - (2C_1 - y_1) \qquad C_2 = -2(-x_0 - x_1) - x_1$$
$$C_1 = -(x_0 + x_1) \qquad C_2 = 2x_0 + x_1$$

Plugging back into the homogeneous equation, the new equation is

$$x_h = -(x_0 + x_1)e^{-2t} + (2x_0 + x_1)e^{-t}$$

Which after plugging in $x(0) = x_0$, $x'(0) = 1$, the equation becomes

$$x_h = -(x_0 + 1)e^{-2t} + (2x_0 + 1)e^{-t}$$

I graphed the equation with three different x_0 initial conditions in Microsoft Visual Code Studio using Python and Jupyter Notebook. The graph and code for the graph can is as follows:

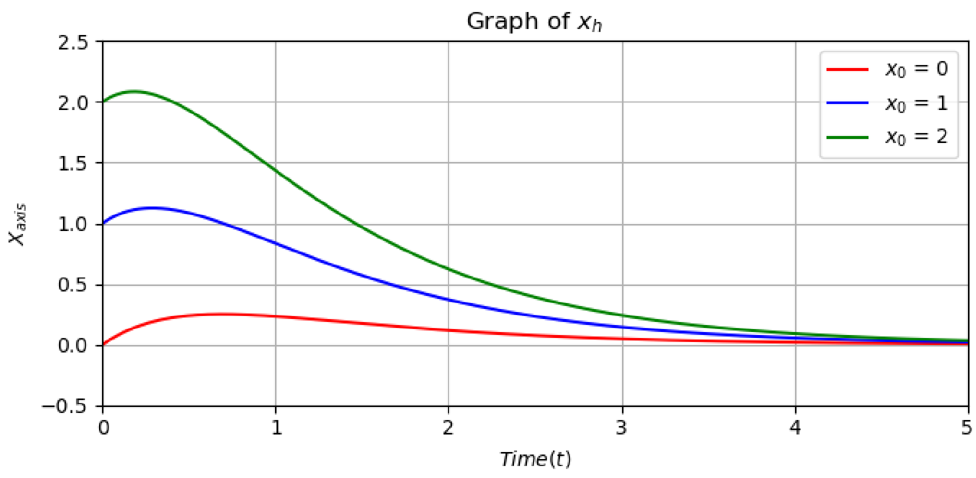
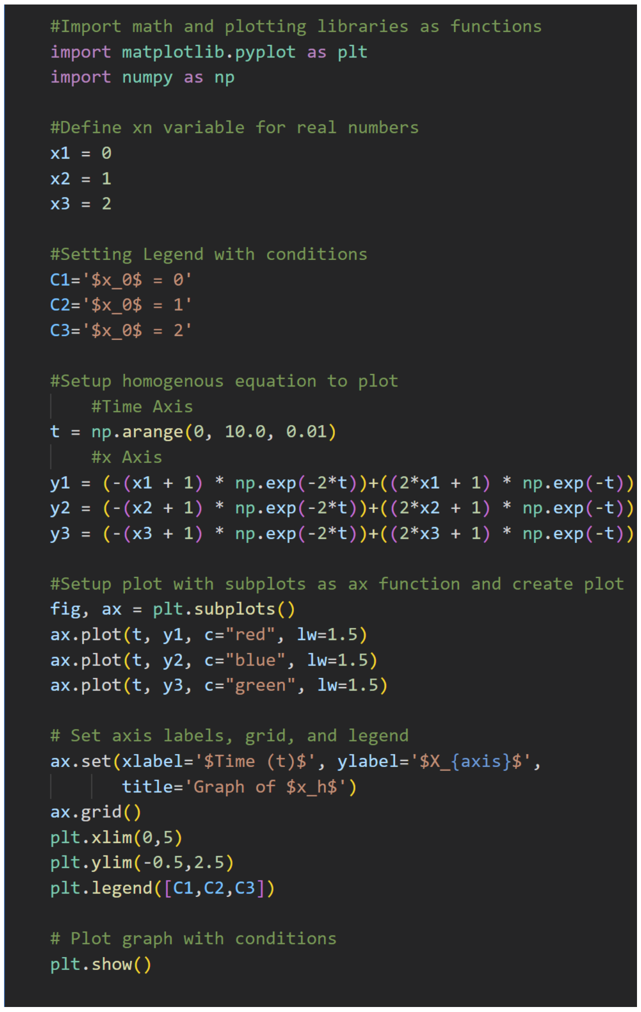


Figure 1: Homogeneous graph with initial conditions



```
#Import math and plotting libraries as functions
import matplotlib.pyplot as plt
import numpy as np

#Define xn variable for real numbers
x1 = 0
x2 = 1
x3 = 2

#Setting Legend with conditions
C1='$x_0$ = 0'
C2='$x_0$ = 1'
C3='$x_0$ = 2'

#Setup homogenous equation to plot
    #Time Axis
t = np.arange(0, 10.0, 0.01)
    #x Axis
y1 = (-(x1 + 1) * np.exp(-2*t))+((2*x1 + 1) * np.exp(-t))
y2 = (-(x2 + 1) * np.exp(-2*t))+((2*x2 + 1) * np.exp(-t))
y3 = (-(x3 + 1) * np.exp(-2*t))+((2*x3 + 1) * np.exp(-t))

#Setup plot with subplots as ax function and create plot
fig, ax = plt.subplots()
ax.plot(t, y1, c="red", lw=1.5)
ax.plot(t, y2, c="blue", lw=1.5)
ax.plot(t, y3, c="green", lw=1.5)

# Set axis labels, grid, and legend
ax.set(xlabel='$Time (t)$', ylabel='$X_{axis}$',
       title='Graph of $x_h$')
ax.grid()
plt.xlim(0,5)
plt.ylim(-0.5,2.5)
plt.legend([C1,C2,C3])

# Plot graph with conditions
plt.show()
```

Figure 2: Code for plotting homogeneous equation with initial conditions

3 Non-homogeneous Solution:

To find a non-homogeneous solution, I found the particular solution of $f(t) = 4cost$ and the homogeneous solution of $x_h = C_1e^{-2t} - C_2e^{-t}$. To do this, I setup the general particular solution for $f(t)$, which is $x_p = t^s(Acost + Bsint)$ where s=0. So this simplifies to:

$$x_p = Acost + Bsint.$$

Then I took the first and second derivative with respect to t to get:

$$x'_p = -Acost + Bsint.$$
$$x''_p = -Acost - Bsint.$$

Then plugging the derivatives of the particular into $x'' + 3x' + 2x = 4cost$ resulted in:

$$(-Acost - Bsint) + 3(-Asint - Bcost) + 2(Acost + Bsint) = 4cost.$$

Which simplifies to $A + 3B(cost) + B - 3A(sint) = 4cost + 0sint$. Setting the coefficient of $4cost$ and $0sint$ equal to $A + 3B$ and $B - 3A$ respectively, results with $B = 6/5$, $A = 2/5$. Which plugging into x_p results in: $x_p = 2/5cost + 6/5sint$. Where $x(t) = x_h + x_p$ is equal to:

$$x(t) = C_1e^{-2t} - C_2e^{-t} + 2/5cost + 6/5sint$$

I applied the initial conditions of $x(0) = x_0$, $x'(0) = x_1$ to $x(t)$ and solved for C_1 and C_2

$$x_0 = C_1e^0 - C_2e^0 + 2/5cos(0) + 6/5sin(0)$$
$$x_0 = C_1 - C_2 + 2/5$$
and
$$x'_0 = -2C_1e^{-2t} - C_2e^{-t} + 2/5sint + 6/5cost$$
$$x_1 = -2C_1e^0 - C_2e^0 + 2/5sin(0) + 6/5cos(0)$$
$$x_1 = -2C_1 - C_2 + 6/5$$

$C_1 = x_0 + C_2 - 2/5$
$C_1 = x_0 - (x_1 + 2x_0 - 2) - 2/5$
$C_1 = -x_1 - x_0 + 8/5$
$C_1 = -(x_1 + x_0) + 8/5$

$x_1 = -2(x_0 - C_2 - 2/5) - C_2 + 6/5$
$x_1 = -2x_0 + 2C_2 + 4/5 - C_2 + 6/5$
$x_1 = -2x_0 + C_2 + 2$
$C_2 = x_1 + 2x_0 - 2$

Plugging back in, the new equation is

$$x(t) = [-(x_1 + x_0) + \tfrac{8}{5}]e^{-2t} + [x_1 + 2x_0 - 2]e^{-t} + \tfrac{2}{5}cost + \tfrac{6}{5}sint$$

Which after plugging in $x(0) = x_0$ and $x'(0) = 1$, the equation becomes

$$x(t) = [-(1 + x_0) + \tfrac{8}{5}]e^{-2t} + [2x_0 - 1]e^{-t} + \tfrac{2}{5}cost + \tfrac{6}{5}sint$$

I graphed three different x_0 with the given initial velocity using this equation as shown below:

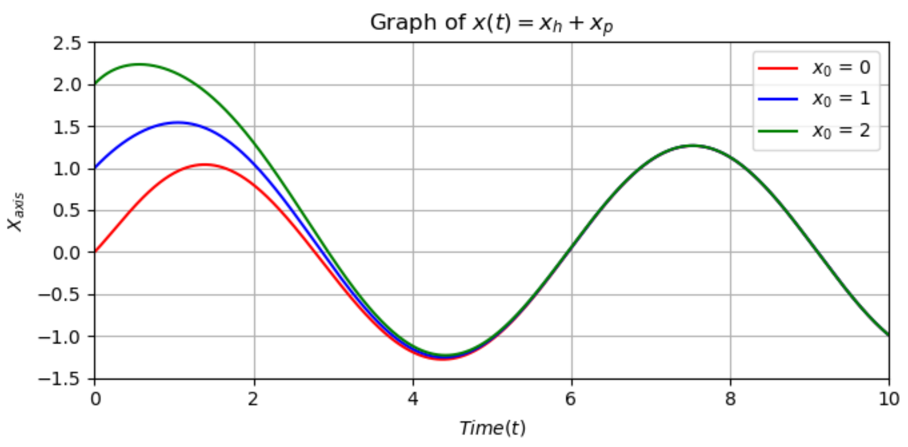
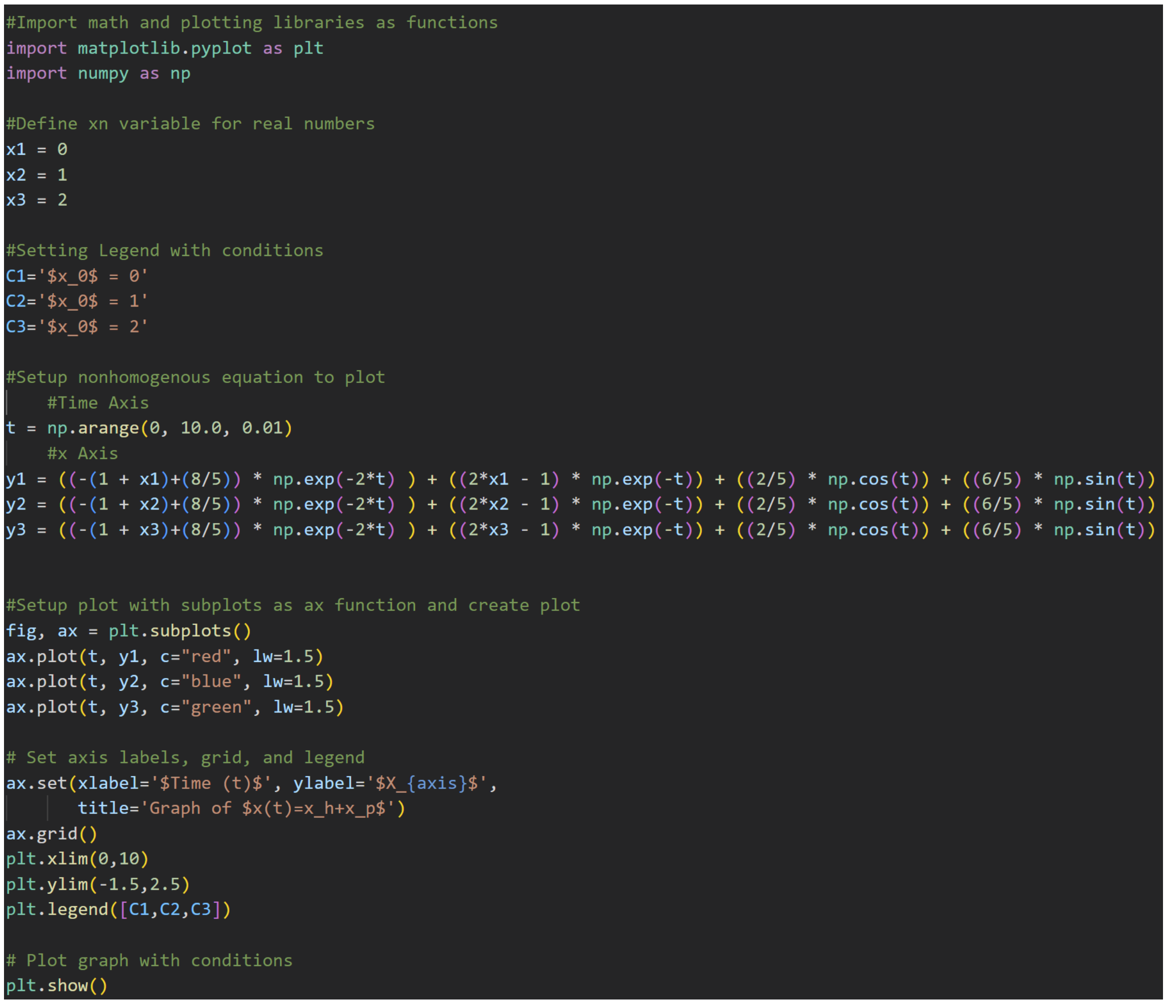


Figure 3: Non-homogeneous graph with initial conditions



```
#Import math and plotting libraries as functions
import matplotlib.pyplot as plt
import numpy as np

#Define xn variable for real numbers
x1 = 0
x2 = 1
x3 = 2

#Setting Legend with conditions
C1='$x_0$ = 0'
C2='$x_0$ = 1'
C3='$x_0$ = 2'

#Setup nonhomogenous equation to plot
    #Time Axis
t = np.arange(0, 10.0, 0.01)
    #x Axis
y1 = ((-(1 + x1)+(8/5)) * np.exp(-2*t) ) + ((2*x1 - 1) * np.exp(-t)) + ((2/5) * np.cos(t)) + ((6/5) * np.sin(t))
y2 = ((-(1 + x2)+(8/5)) * np.exp(-2*t) ) + ((2*x2 - 1) * np.exp(-t)) + ((2/5) * np.cos(t)) + ((6/5) * np.sin(t))
y3 = ((-(1 + x3)+(8/5)) * np.exp(-2*t) ) + ((2*x3 - 1) * np.exp(-t)) + ((2/5) * np.cos(t)) + ((6/5) * np.sin(t))


#Setup plot with subplots as ax function and create plot
fig, ax = plt.subplots()
ax.plot(t, y1, c="red", lw=1.5)
ax.plot(t, y2, c="blue", lw=1.5)
ax.plot(t, y3, c="green", lw=1.5)

# Set axis labels, grid, and legend
ax.set(xlabel='$Time (t)$', ylabel='$X_{axis}$',
       title='Graph of $x(t)=x_h+x_p$')
ax.grid()
plt.xlim(0,10)
plt.ylim(-1.5,2.5)
plt.legend([C1,C2,C3])

# Plot graph with conditions
plt.show()
```

Figure 4: Code for plotting Non-homogeneous equation with initial conditions

4 Results of Homogeneous and Non-homogeneous:

When the inital condition from Figure 1 and 3 are changed, the initial height of the oscillation is either increased or decreased from its initial position. Depending on how far the initial condition is changed, the amplitude of the oscillation could be extremely high, low, or negative. In the end, all of the lines eventually converge to form a single line or oscillation.

The homogeneous IVP model is an over damped oscillation since the solution has two real roots with no oscillation or imaginary roots. The steady state solution is $x_p = 2/5cost + 6/5sint$ and the transient solution is $x_h = C_1e^{-2t} - C_2e^{-t}$

5 Graphing $x = x_h + x_p$:

Now I took the homogeneous plus non-homogeneous solution from section 2 after plugging in $x(0) = 2$ and $x'(0) = x_1$, where the equation becomes

$$x = [-(x_1 + 2) + \frac{8}{5}]e^{-2t} + [x_1 + 2]e^{-t} + \frac{2}{5}cost + \frac{6}{5}sint$$

I graphed it with four different x_1 with the given initial displacement condition.

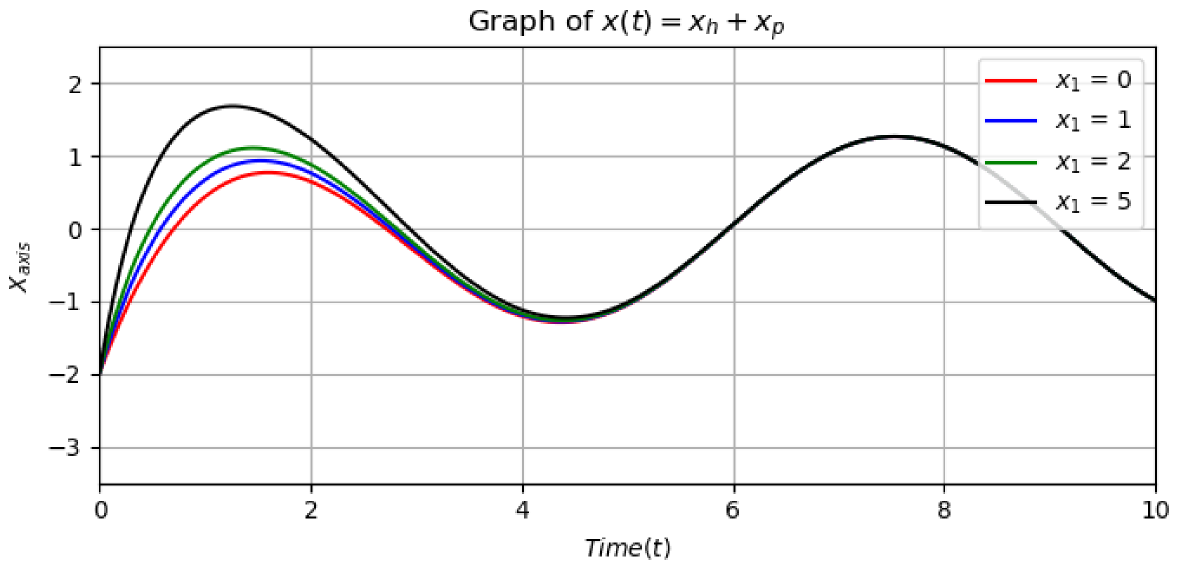
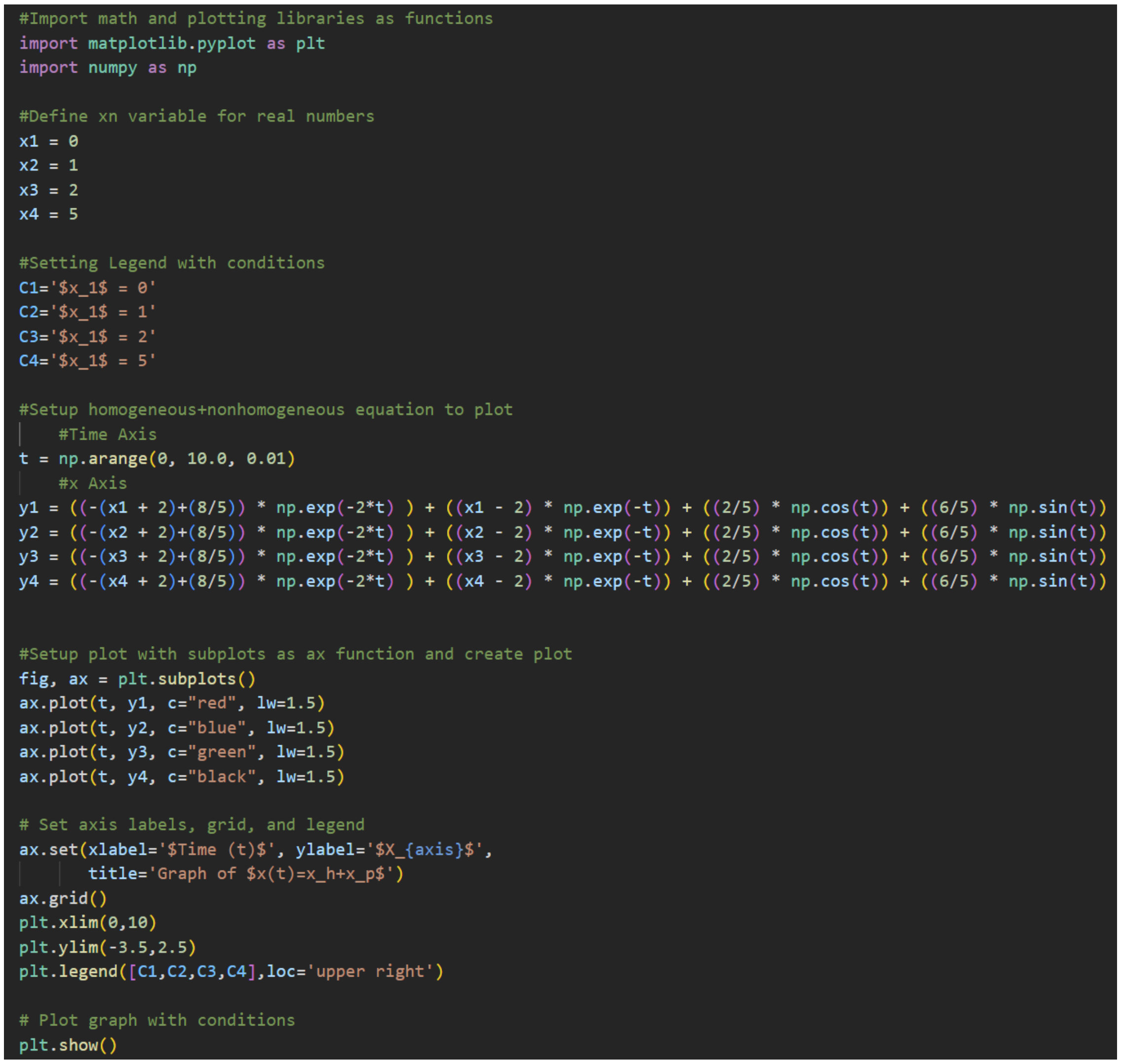


Figure 5: Homogeneous plus Non-homogeneous graph with initial conditions



```
#Import math and plotting libraries as functions
import matplotlib.pyplot as plt
import numpy as np

#Define xn variable for real numbers
x1 = 0
x2 = 1
x3 = 2
x4 = 5

#Setting Legend with conditions
C1='$x_1$ = 0'
C2='$x_1$ = 1'
C3='$x_1$ = 2'
C4='$x_1$ = 5'

#Setup homogeneous+nonhomogeneous equation to plot
    #Time Axis
t = np.arange(0, 10.0, 0.01)
    #x Axis
y1 = ((-(x1 + 2)+(8/5)) * np.exp(-2*t) ) + ((x1 - 2) * np.exp(-t)) + ((2/5) * np.cos(t)) + ((6/5) * np.sin(t))
y2 = ((-(x2 + 2)+(8/5)) * np.exp(-2*t) ) + ((x2 - 2) * np.exp(-t)) + ((2/5) * np.cos(t)) + ((6/5) * np.sin(t))
y3 = ((-(x3 + 2)+(8/5)) * np.exp(-2*t) ) + ((x3 - 2) * np.exp(-t)) + ((2/5) * np.cos(t)) + ((6/5) * np.sin(t))
y4 = ((-(x4 + 2)+(8/5)) * np.exp(-2*t) ) + ((x4 - 2) * np.exp(-t)) + ((2/5) * np.cos(t)) + ((6/5) * np.sin(t))


#Setup plot with subplots as ax function and create plot
fig, ax = plt.subplots()
ax.plot(t, y1, c="red", lw=1.5)
ax.plot(t, y2, c="blue", lw=1.5)
ax.plot(t, y3, c="green", lw=1.5)
ax.plot(t, y4, c="black", lw=1.5)

# Set axis labels, grid, and legend
ax.set(xlabel='$Time (t)$', ylabel='$X_{axis}$',
       title='Graph of $x(t)=x_h+x_p$')
ax.grid()
plt.xlim(0,10)
plt.ylim(-3.5,2.5)
plt.legend([C1,C2,C3,C4],loc='upper right')

# Plot graph with conditions
plt.show()
```

Figure 6: Code for Homogeneous plus Non-homogeneous graph with initial conditions

In Figure 5, the system shows the variation of initial conditions with starting amplitudes, but as time continues they converge to the steady state and transient solution. So no matter how large a number is input for the initial displacement, they will fall into a predictable oscillation modeled by $x(t)$.

6 Laplace Transforms:

I solved the homogeneous and non-homogeneous systems with Laplace transforms for $x(0) = 2$ and $x'(0) = 1$. First, I used the homogeneous with initial conditions to get

$$x'' + 3x' + 2x = 0; x(0) = 2, x'(0) = 1$$

Then I applied the Laplace transform with the differentiation property to each term of the homogeneous solution:

$$s^2X - 2s - 1 + 3sX - 6 + 2X = 0$$

Which simplifies to

$$X[s^2 + 3s + 2] = 7 + 2s$$

After dividing, I applied partial fractions to solve for A and B:

$$X = \frac{7+2s}{(s+2)(s-1)}$$

$$7 + 2s = A(s + 1) + B(s + 2)$$

Plugging in s=0,1, I solved for A=-3 and B=-4

$$X = \frac{-3}{s+2} - \frac{4}{s+1}$$

Then taking the inverse Laplace of both sides gets the final homogeneous solution with conditions of:

$$\mathscr{L}^{-}1(X) = -3e^{-2t} - 4e^{-t}$$

Which matches the homogeneous solution of x_h with conditions of $x(0) = 2, x'(0) = 1$

$$x_h = -3e^{-2t} - 4e^{-t}$$

I then applied the Laplace transform again to the non-homogeneous portion solving with the initial conditions:

$$x'' + 3x' + 2x = 4cost; x(0) = 2, x'(0) = 1$$

Then I applied the Laplace transform with the differentiation property to each term of the homogeneous solution

$$s^2X - 2s - 1 + 3sX - 6 + 2X = \frac{4s}{s^2+1}$$

Which simplifies to

$$X[s^2 + 3s + 2] = 7 + 2s + \frac{4s}{s^2+1}$$

After dividing, I applied partial fractions to solve for A,B,C,D:

$$X = \frac{2s^3+7s^2+6s+7}{(s^2+1)(s^2+3s+2)}$$

$$\mathscr{L}^{-}1(X) = [\frac{2s}{s(s^2+1)} + \frac{6}{5(s^2+1} + \frac{3}{s+1} - \frac{7}{s(s+2)}]$$

Which gives the final solution of: $x = \frac{2}{5}cost + \frac{6}{5}sint + 3e^{-t} - \frac{7}{5}e^{-2t}$

Which matches the non-homogeneous solution of $x(t)$ with conditions of $x(0) = 2, x'(0) = 1$

$$x(t) = \frac{2}{5}cost + \frac{6}{5}sint + 3e^{-t} - \frac{7}{5}e^{-2t}$$

7 Summary:

I was successful in modeling the IVP differential equation of the given mass spring system: $x'' + 3x' = 2x = 0$ letting the non-homogeneous portion equal $f(t) = 4cost$. I fully solved for all given initial conditions and graphed with several different initial displacements to show the changing graphs with the conditions. I was able to identify the oscillation, steady state, and transient solutions. Finally, I took the Laplace transform for the homogeneous and non-homogeneous systems with given initial conditions.

You might see an over damped oscillation in the "real world" where a system or mechanism slowly returns to its original position without oscillating. A perfect example of a design that uses an over damped oscillation is the closing of a automatic handicap door. A door that slowly returns to its original position can be modeled with an over damped oscillation, as you do not want the door to oscillate through its rest position, but if displaced either forward or back, to eventually return to its zero position.

Having done this project, I have gained more experience in Python, Visual Code Studio and using Jupyter Notebook. All of these skills, I have acquired from my journey with this project, and I have learned many new skills and techniques that I will be able to showcase and apply in my future studies.

I would recommend that the project could be altered to further either the student's solving or application skills. The project could feature each "look" of oscillation from over damped, critically, under, and no damping to be graphed. Although, this would introduce more work for the student, other sections such as Laplace transforms, or less conditions, could be altered in length or removed to distribute the workload more evenly.