Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Signals & Systems I
Course Number:	ELE 532
Semester/Year (e.g.F2016)	F2023

Instructor:	Dr. Alirezai

Assignment/Lab Number:	Lab 2
Assignment/Lab Title:	Continuous-Time Convolution

Submission Date:	October 22,2023
Due Date:	October 22,2023

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Ally	Jahmil	501045419	5	JAlly

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Lab Assignment

Imports, Functions and Variables

You, 3 minutes ago | 1 author (You)

```
#Lab1 ELE532
#Jahmil Ally (501045419)

#Imports
import os
import time
import numpy as np
import sympy as sp
import scipy.io as sci
import matplotlib.pyplot as plt
import sounddevice as sd        You, 2 weeks ago • Create lab2main.py
import matlab.engine

#Variable Declaration
color="g"
eng = matlab.engine.start_matlab()

#Defining a generic function for plotting
def plot(f_t, t, newGraph=True, figsize=(12.0, 6.0), title="", functionLabel="", xLabel="t", yLabel="f(t)"):
    global color
    if newGraph:
        plt.figure(figsize=figsize)
        color="g"

    #Configure Colour
    if  color == "g" and  newGraph==False:
        color="r"
    elif color == "r" and  newGraph==False:
        color="y"
    elif color == "y" and  newGraph==False:
        color="b"
    elif color == "b" and  newGraph==False:
        color="o"
    elif color == "o" and  newGraph==False:
        color="p"

    #Configurable titles/ Labels
    plt.plot(t, f_t, color=color, label=functionLabel)
    if title !="":
        plt.title(title)
    if xLabel !="":
        plt.xlabel(xLabel)
    if yLabel !="":
        plt.ylabel(yLabel)

    #Visuals
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
```

A. Impulse Response

- **Problem A.1**

  Code:

```
#Part A:
#Problem A.1
#Define values
R = [1e4, 1e4, 1e4]
C = [1e-6, 1e-6]

#Coefficients for the characteristic equation
A1 = [1, (1/R[0] + 1/R[1] + 1/R[2]) / C[1], 1 / (R[0] * R[1] * C[0] * C[1])]

#Characteristic roots
lambda_values = np.roots(A1)

print("\nA.1) Roots:" + str(lambda_values[0]) + "," + str(lambda_values[1]) + "\n")
```

  Results:

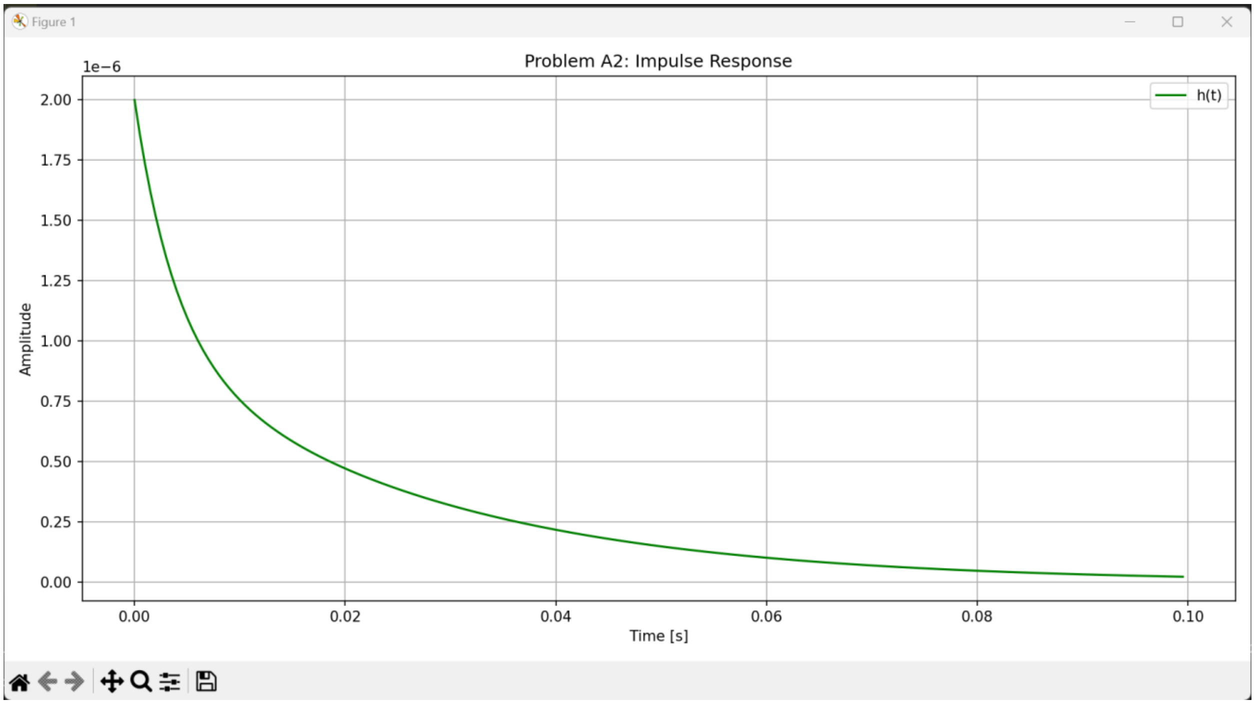
```
A.1) Roots:-261.8033988749895,-38.19660112501052
```

- **Problem A.2**

  Code:

```
#Problem A.2
#Time vector
t = np.arange(0, 0.1, 0.0005)

#Unit step function: u(t)
u = lambda t: 1.0 * (t >= 0)

#h(t) using the characteristic roots
h = lambda t: (C[0] * np.exp(lambda_values[0] * t) + C[1] * np.exp(lambda_values[1] * t)) * (u(t))

#h(t) Plotted
plot(h(t), t, title="Problem A2: Impulse Response", functionLabel="h(t)", xLabel="Time [s]", yLabel="Amplitude")
```

  Results:



- **Problem A.3**
  Code:

```
#Problem A.3
def CH2MP2(R, C):
    #Coefficients for the characteristic equation
    A = [1, (1/R[0] + 1/R[1] + 1/R[2]) / C[1], 1 / (R[0] * R[1] * C[0] * C[1])]

    #Characteristic roots
    roots = np.roots(A)

    return roots

lambda_ = CH2MP2([1e4, 1e4, 1e4],[1e-9, 1e-6])

print("\nA.3) Roots:" + str(lambda_[0]) + "," + str(lambda_[1]) + "\n")

plt.show()
```

  Results:

```
A.3) Roots:(-150.00000000000006+3158.7180944174174j),(-150.00000000000006-3158.7180944174174j)
```

B. Convolution.

- **Problem B.1**
  Code:

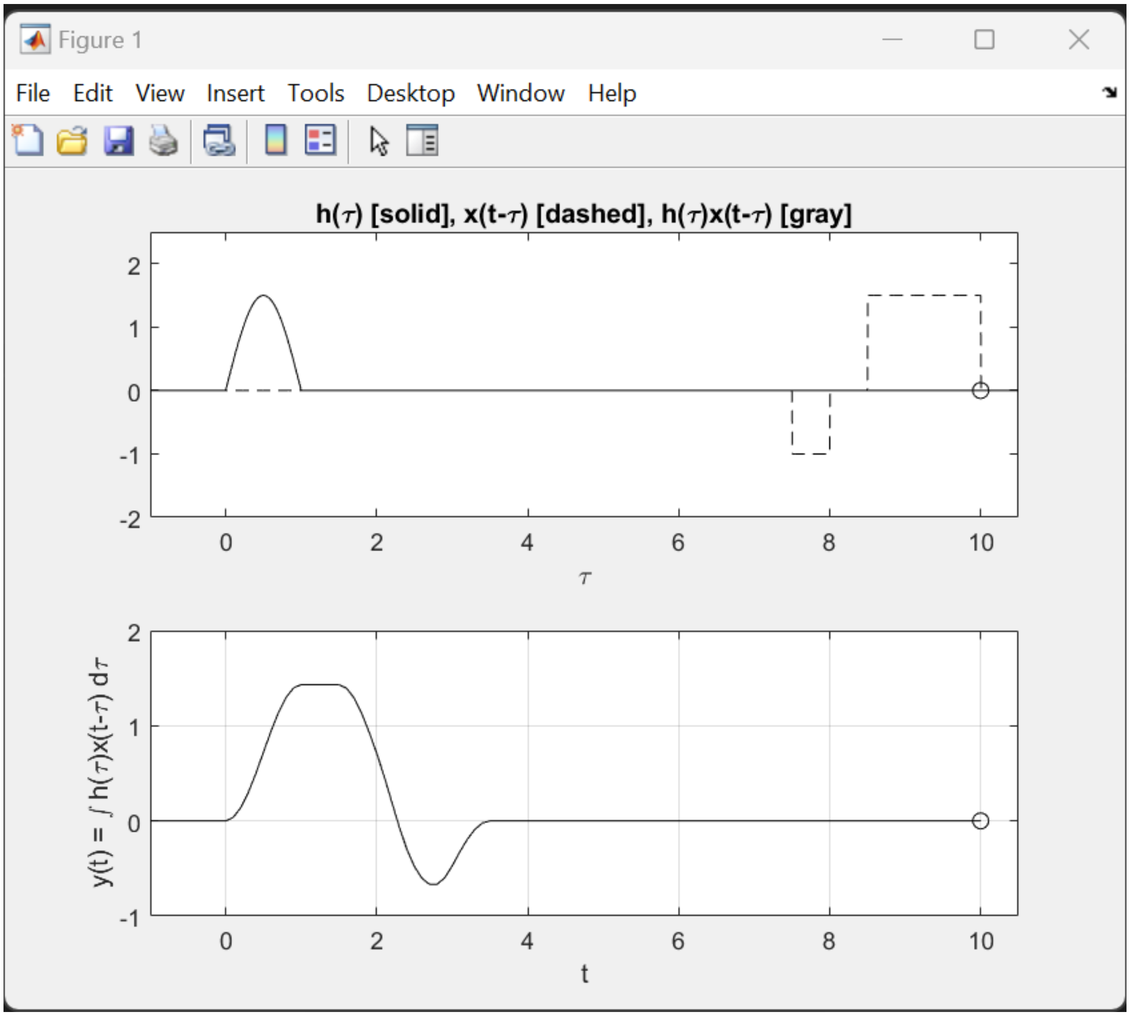
```python
#Part B:
#Problem B.1
#Specify the primary and alternate file paths
primary_script_path = "/Users/jah/Documents/GitHub/ELE532/lab2/CH2MP4.m"
alternate_script_path = "C:\\Users\\Jahmil\\Desktop\\Coding_Projects\\ELE532\\lab2\\CH2MP4.m"

#Check if the file exists in the primary location
if os.path.exists(primary_script_path):
    script_path = primary_script_path
else:
    #If not, use the alternate location
    script_path = alternate_script_path

eng = matlab.engine.start_matlab()
eng.eval(f"run('{script_path}')", nargout=0)
```

lab2 > CH2MP4.m

You, 3 minutes ago | 1 author (You)

```matlab
% CH2MP4.m : Chapter 2, MATLAB Program 4
% Script M-file graphically demonstrates the convolution process.figure(1)
% Create figure window and make visible on screen
u = @(t) 1.0*(t>=0);
x = @(t) 1.5*(u(t)-u(t-1.5))-u(t-2)+u(t-2.5);
h = @(t) 1.5*sin(pi*t).*(u(t)-u(t-1));
dtau = 0.005;
tau = -1:dtau:10.5;ti = 0;
tvec = -1:.1:10; y = NaN*zeros(1,length(tvec));

% Pre-allocate memory        You, 2 weeks ago • Lab2
for t = tvec,
    ti = ti+1; % Time index
    xh = x(t-tau).*h(tau);
    lxh = length(xh);
    y(ti) = sum(xh.*dtau);
    % Trapezoidal approximation of convolution integral
    subplot(2,1,1),plot(tau,h(tau),"k-",tau,x(t-tau),"k--",t,0,"ok");
    axis([tau(1) tau(end) -2.0 2.5]);
    patch([tau(1:end-1);tau(1:end-1);tau(2:end);tau(2:end)],...
        [zeros(1,lxh-1);xh(1:end-1);xh(2:end);zeros(1,lxh-1)],...
        [.8 .8 .8],"edgecolor","none");
    xlabel("\tau"); title("h(\tau) [solid], x(t-\tau) [dashed], h(\tau)x(t-\tau) [gray]");
    c = get(gca,'children'); set(gca,'children',[c(2);c(3);c(4);c(1)]);
    subplot(2,1,2),plot(tvec,y,"k",tvec(ti),y(ti),"ok");
    xlabel("t"); ylabel("y(t) = \int h(\tau)x(t-\tau) d\tau");
    axis([tau(1) tau(end) -1.0 2.0]); grid;

    if abs(t - 2.25) < 0.01 % Check if t is close to 2.25
        pause; % Pause at t = 2.25
    else
        pause(0.001); % Pause for other time points
    end
end
```

Results:

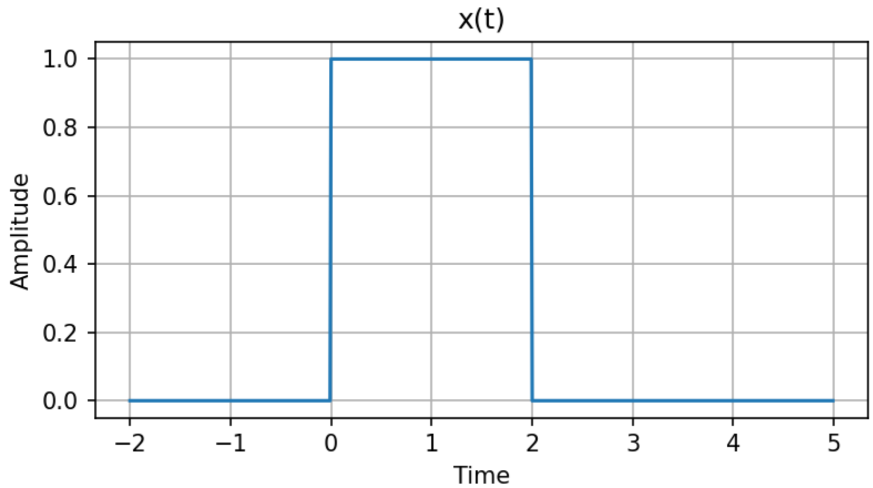


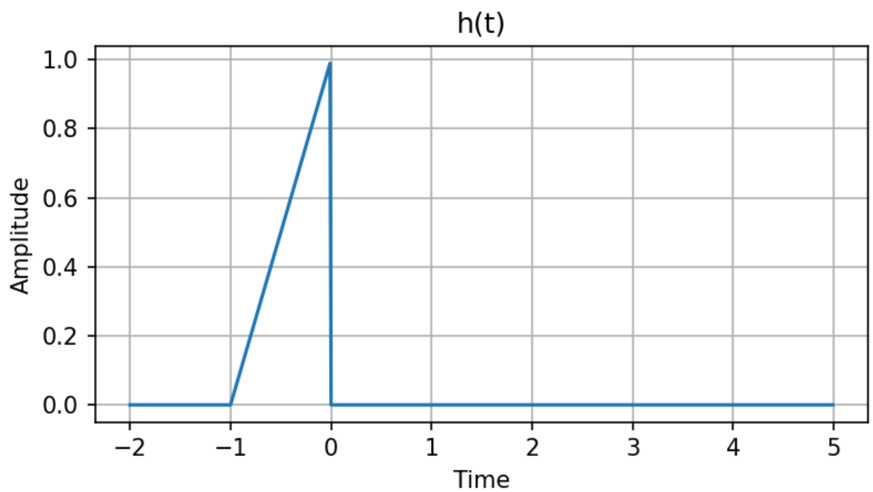
- **Problem B.2**

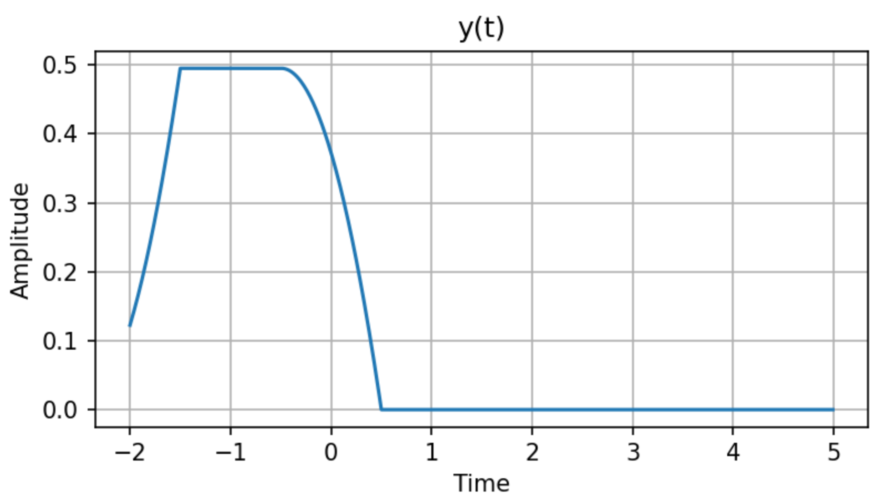
  Code:

```python
#Problem B.2
#Defining functions
x = lambda t: np.heaviside(t , 1) - np.heaviside(t - 2, 1)
h = lambda t: (t+1) * (np.heaviside(t + 1, 1) - np.heaviside(t, 1))

#Defining time vector
t = np.arange(-2, 5, 0.01)

x_t = x(t)
h_t = h(t)

y_t = np.convolve(x_t, h_t, "same") * 0.01

plt.figure(figsize=(6, 14))

#Subplot for x(t)
plt.subplot(3, 1, 1)
plt.plot(t, x_t, label="x(t)")
plt.title("x(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

#Subplot for h(t)
plt.subplot(3, 1, 2)
plt.plot(t, h_t, label="h(t)")
plt.title("h(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

#Subplot for y(t)
plt.subplot(3, 1, 3)
plt.plot(t, y_t)
plt.title("y(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

plt.subplots_adjust(hspace=0.5)

plt.show()
```

Results:


x(t)
Amplitude
Time


h(t)
Amplitude
Time


y(t)
Amplitude
Time

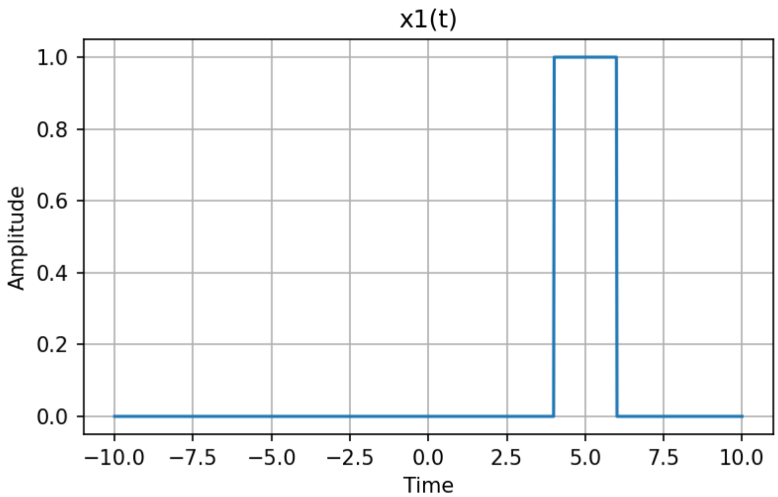
- **Problem B.3**

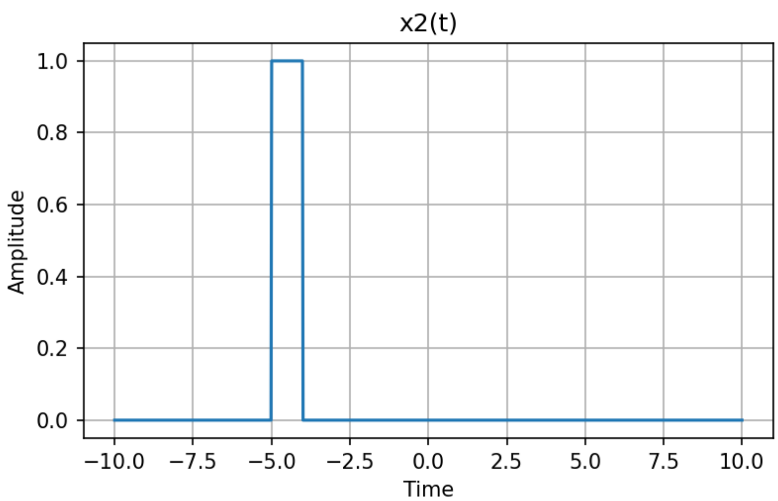
  Part A:

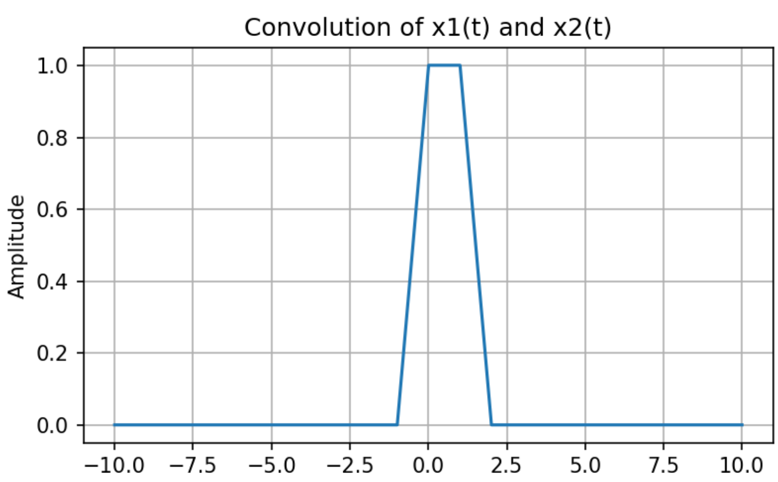
  Code:

```
#Problem B.3
#Part a)
t = np.linspace(-10, 10, 1000)
x1 = lambda t: (np.heaviside(t - 4, 0.5) - np.heaviside(t - 6, 0.5))
x2 = lambda t: (np.heaviside(t + 5, 0.5) - np.heaviside(t + 4, 0.5))

x1_t = x1(t)
x2_t = x2(t)

convolution = np.convolve(x1_t, x2_t, "same") * (t[1]-t[0])  # Multiply by dt for integration

plt.figure(figsize=(6, 14))

#Subplot for x1(t)
plt.subplot(3, 1, 1)
plt.plot(t, x1_t)
plt.title("x1(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

#Subplot for x2(t)
plt.subplot(3, 1, 2)
plt.plot(t, x2_t)
plt.title("x2(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

#Subplot for convolution
plt.subplot(3, 1, 3)
plt.plot(t, convolution)
plt.title("Convolution of x1(t) and x2(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

plt.subplots_adjust(hspace=0.5)
```

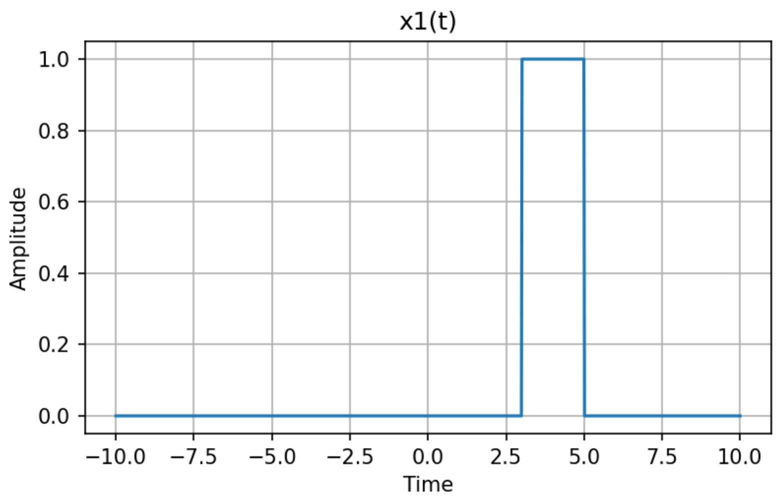
  Results:

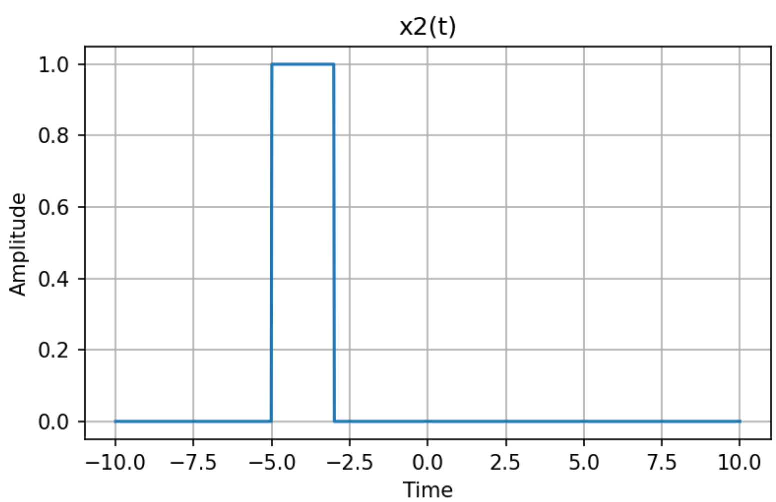

x1(t)
Amplitude
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Time

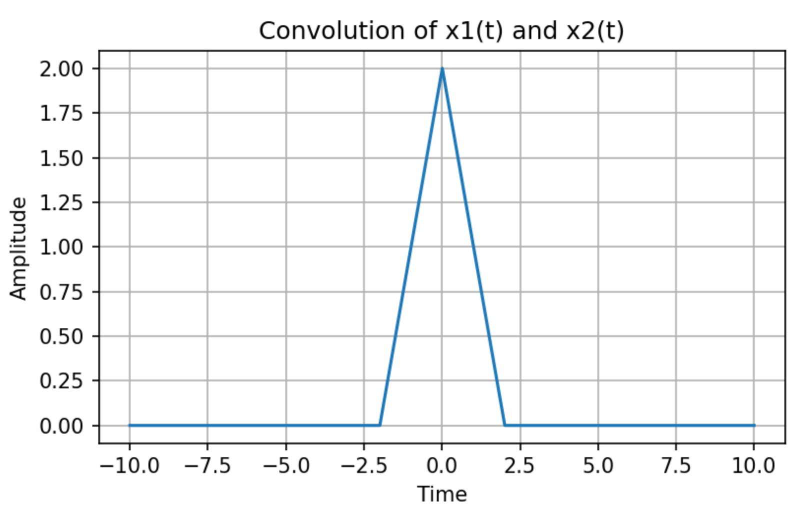

x2(t)
Amplitude
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Time


Convolution of x1(t) and x2(t)
Amplitude
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Time

Part B:
Code:
```
#Part b)
t = np.linspace(-10, 10, 1000)
x1 = lambda t: (np.heaviside(t - 3, 0.5) - np.heaviside(t - 5, 0.5))
x2 = lambda t: (np.heaviside(t + 5, 0.5) - np.heaviside(t + 3, 0.5))

x1_t = x1(t)
x2_t = x2(t)

convolution = np.convolve(x1_t, x2_t, "same") * (t[1]-t[0])  # Multiply by dt for integration

plt.figure(figsize=(6, 14))

#Subplot for x1(t)
plt.subplot(3, 1, 1)
plt.plot(t, x1_t)
plt.title("x1(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

#Subplot for x2(t)
plt.subplot(3, 1, 2)
plt.plot(t, x2_t)
plt.title("x2(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

#Subplot for convolution
plt.subplot(3, 1, 3)
plt.plot(t, convolution)
plt.title("Convolution of x1(t) and x2(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)
plt.subplots_adjust(hspace=0.5)
```
Results:


x1(t)
Amplitude
Time
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

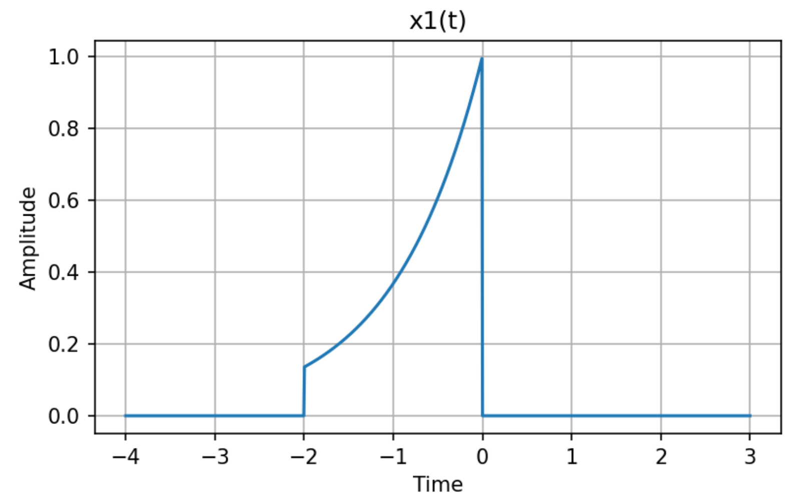

x2(t)
Amplitude
Time
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


Convolution of x1(t) and x2(t)
Amplitude
Time
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

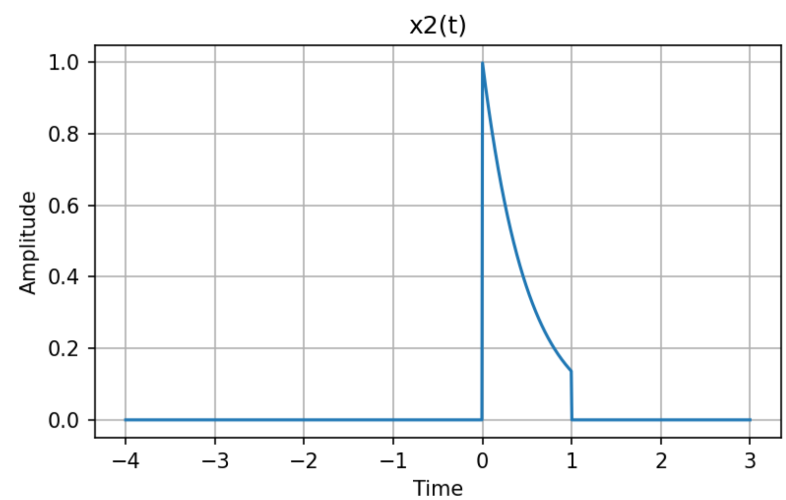
Part H:
Code:

```
#Part h)
t = np.linspace(-4, 3, 1000)  #Extended to capture both functions
x1 = lambda t: np.exp(t) * (np.heaviside(t + 2, 0.5) - np.heaviside(t, 0.5))
x2 = lambda t: np.exp(-2 * t) * (np.heaviside(t, 0.5) - np.heaviside(t - 1, 0.5))

x1_t = x1(t)
x2_t = x2(t)

convolution = np.convolve(x1_t, x2_t, "same") * (t[1]-t[0])  # Multiply by dt for integration

plt.figure(figsize=(6, 14))

# Subplot for x1(t)
plt.subplot(3, 1, 1)
plt.plot(t, x1_t)
plt.title("x1(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

# Subplot for x2(t)
plt.subplot(3, 1, 2)
plt.plot(t, x2_t)
plt.title("x2(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

# Subplot for convolution
plt.subplot(3, 1, 3)
plt.plot(t, convolution)
plt.title("Convolution of x1(t) and x2(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)
plt.subplots_adjust(hspace=0.5)

plt.show()
```

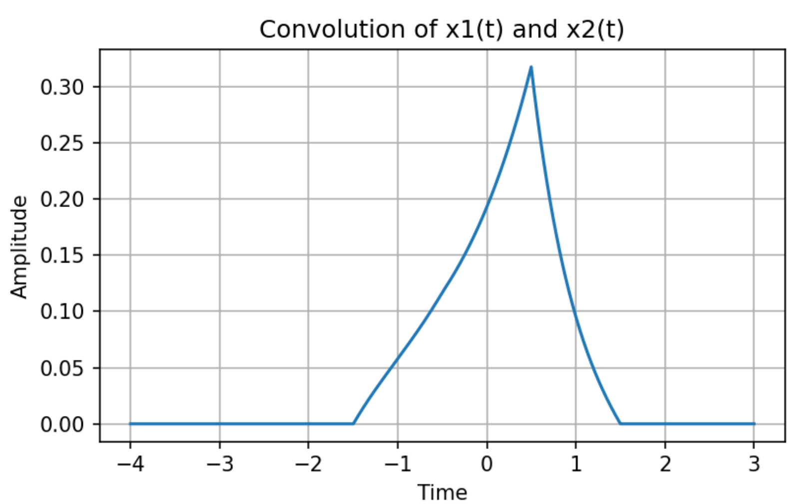
Results:



x1(t)
Amplitude
Time



x2(t)
Amplitude
Time



Convolution of x1(t) and x2(t)
Amplitude
Time

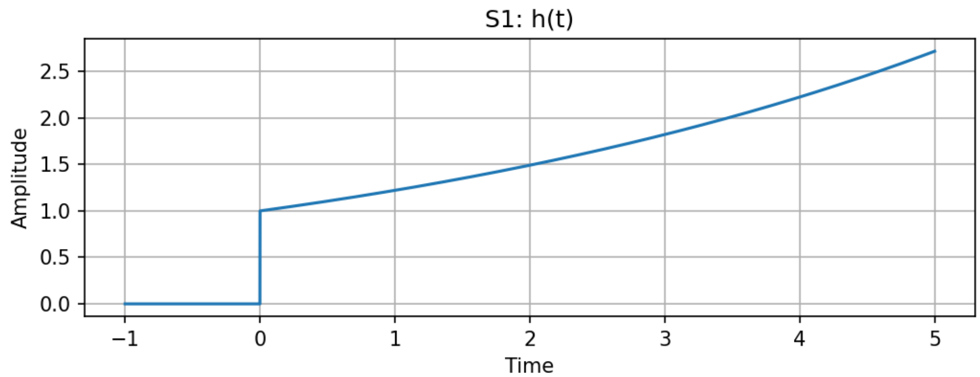
C. System Behavior and Stability.

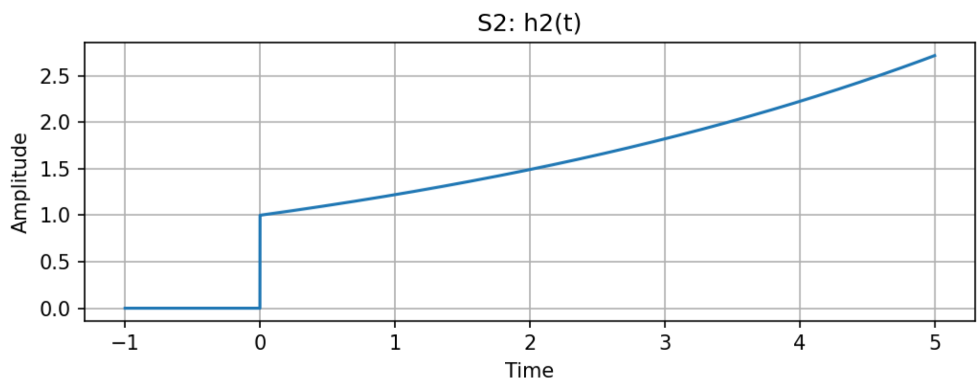
- **Problem C.1**

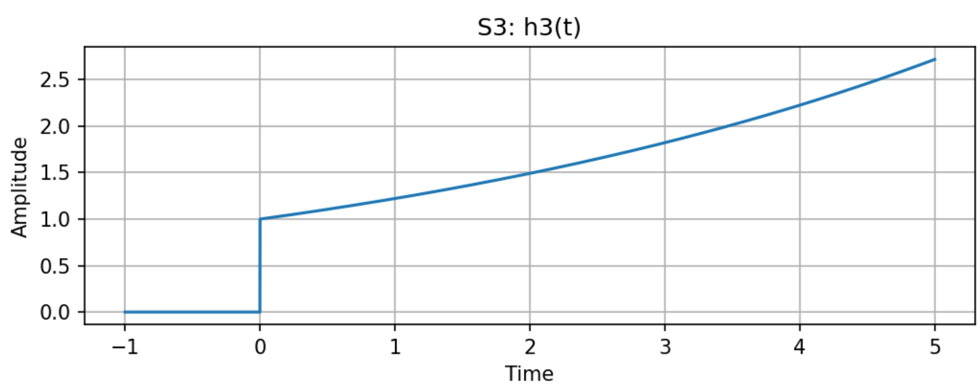
  Code:

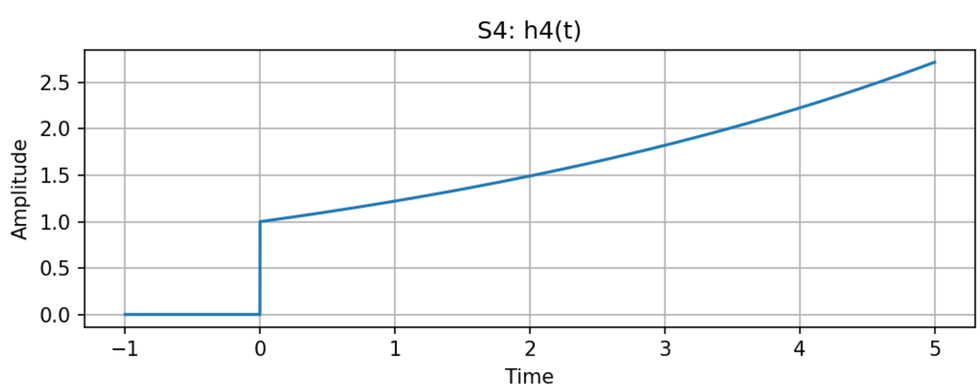
```
#Part C:
#Problem C.1
#Defining Functions
t = np.arange(-1, 5, 0.001)
h1 = lambda t: np.exp(t/5) * np.heaviside(t, 1)
h2 = lambda t: 4*np.exp(-t/5) * np.heaviside(t, 1)
h3 = lambda t: 4*np.exp(-t) * np.heaviside(t, 1)
h4 = lambda t: 4*(np.exp(-t/5) - np.exp(-t)) * np.heaviside(t, 1)


h1_t= h1(t)
h2_t= h1(t)
h3_t= h1(t)
h4_t= h1(t)


plt.figure(figsize=(8, 16))

# Subplot for h1(t)
plt.subplot(4, 1, 1)
plt.plot(t, h1_t, label="e^(t/5) * u(t)")
plt.title("S1: h(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

# Subplot for h2(t)
plt.subplot(4, 1, 2)
plt.plot(t, h2_t, label="4e^(-t/5) * u(t)")
plt.title("S2: h2(t)")
plt.xlabel("Time")        You, 9 minutes ago • Uncommitted changes
plt.ylabel("Amplitude")
plt.grid(True)

# Subplot for h3(t)
plt.subplot(4, 1, 3)
plt.plot(t, h3_t, label="4e^-t * u(t)")
plt.title("S3: h3(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

# Subplot for h4(t)
plt.subplot(4, 1, 4)
plt.plot(t, h4_t, label="4(e^(-t/5) - e^(-t)) * u(t)")
plt.title("S4: h4(t)")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.grid(True)

plt.subplots_adjust(hspace=0.5)

plt.show()
```

Results:


S1: h(t)
Amplitude
Time
2.5
2.0
1.5
1.0
0.5
0.0
−1
0
1
2
3
4
5


S2: h2(t)
Amplitude
Time
2.5
2.0
1.5
1.0
0.5
0.0
−1
0
1
2
3
4
5


S3: h3(t)
Amplitude
Time
2.5
2.0
1.5
1.0
0.5
0.0
−1
0
1
2
3
4
5


S4: h4(t)
Amplitude
Time
2.5
2.0
1.5
1.0
0.5
0.0
−1
0
1
2
3
4
5

- **Problem C.2**

  Results:

```
#Problem C.2
#S1:
eigenvalue_1 = 1/5

#S2:
eigenvalue_2 = -(1/5)

#S3:
eigenvalue_3 = -1

#S4:
eigenvalue_4a = -(1/5)
eigenvalue_4b = -1
```

- **Problem C.3**

  Code:

```
#Problem C.3
#Specify the primary and alternate file paths
primary_script_path = "/Users/jah/Documents/GitHub/ELE532/lab2/C3.m"
alternate_script_path = "C:\\Users\\Jahmil\\Desktop\\Coding_Projects\\ELE532\\lab2\\C3.m"

#Check if the file exists in the primary location
if os.path.exists(primary_script_path):
    script_path = primary_script_path
else:
    #If not, use the alternate location
    script_path = alternate_script_path

eng = matlab.engine.start_matlab()
eng.eval(f"run('{script_path}')", nargout=0)
```

C3.m U ×

lab2 > C3.m

```matlab
%Define the u(t) function
u = @(t) 1.0.* (t>=0);

%Define the x(t) function.
x = @(t) sin(5*t).*(u(t) - u(t - 3));

%Truncate each impluse response funtion.
h1 = @(t) exp(t/5).*(u(t)-u(t-20));
h2 = @(t) 4*exp(-t/5).*(u(t)-u(t-20));
h3 = @(t) 4*exp(-t).*(u(t)-u(t-20));
h4 = @(t) 4*(exp(-t/5)-exp(-t)).*(u(t)-u(t-20));

%Modified CH2MP4 from B.1
dtau = 0.005;
tau = 0:dtau:20; ti = 0;
tvec = 0:.1:20; y = NaN*zeros(1,length(tvec));


%Change This to see Each Function
h=h1;

% Pre-allocate memory
for t = tvec,
    ti = ti+1; % Time index
    xh = x(t-tau).*h(tau);
    lxh = length(xh);
    y(ti) = sum(xh.*dtau);
    % Trapezoidal approximation of convolution integral
    subplot(2,1,1),plot(tau,h(tau),"k-",tau,x(t-tau),"k--",t,0,"ok");
    axis([tau(1) tau(end) -2.0 2.5]);
    patch([tau(1:end-1);tau(1:end-1);tau(2:end);tau(2:end)],...
        [zeros(1,lxh-1);xh(1:end-1);xh(2:end);zeros(1,lxh-1)],...
        [.8 .8 .8],"edgecolor","none");
    xlabel("\tau"); title("h(\tau) [solid], x(t-\tau) [dashed], h(\tau)x(t-\tau) [gray]");
    c = get(gca,'children'); set(gca,'children',[c(2);c(3);c(4);c(1)]);
    subplot(2,1,2),plot(tvec,y,"k",tvec(ti),y(ti),"ok");
    xlabel("t"); ylabel("y(t) = \int h(\tau)x(t-\tau) d\tau");
    axis([tau(1) tau(end) -1.0 2.0]); grid;

    drawnow;
end


```

Results:

h1:

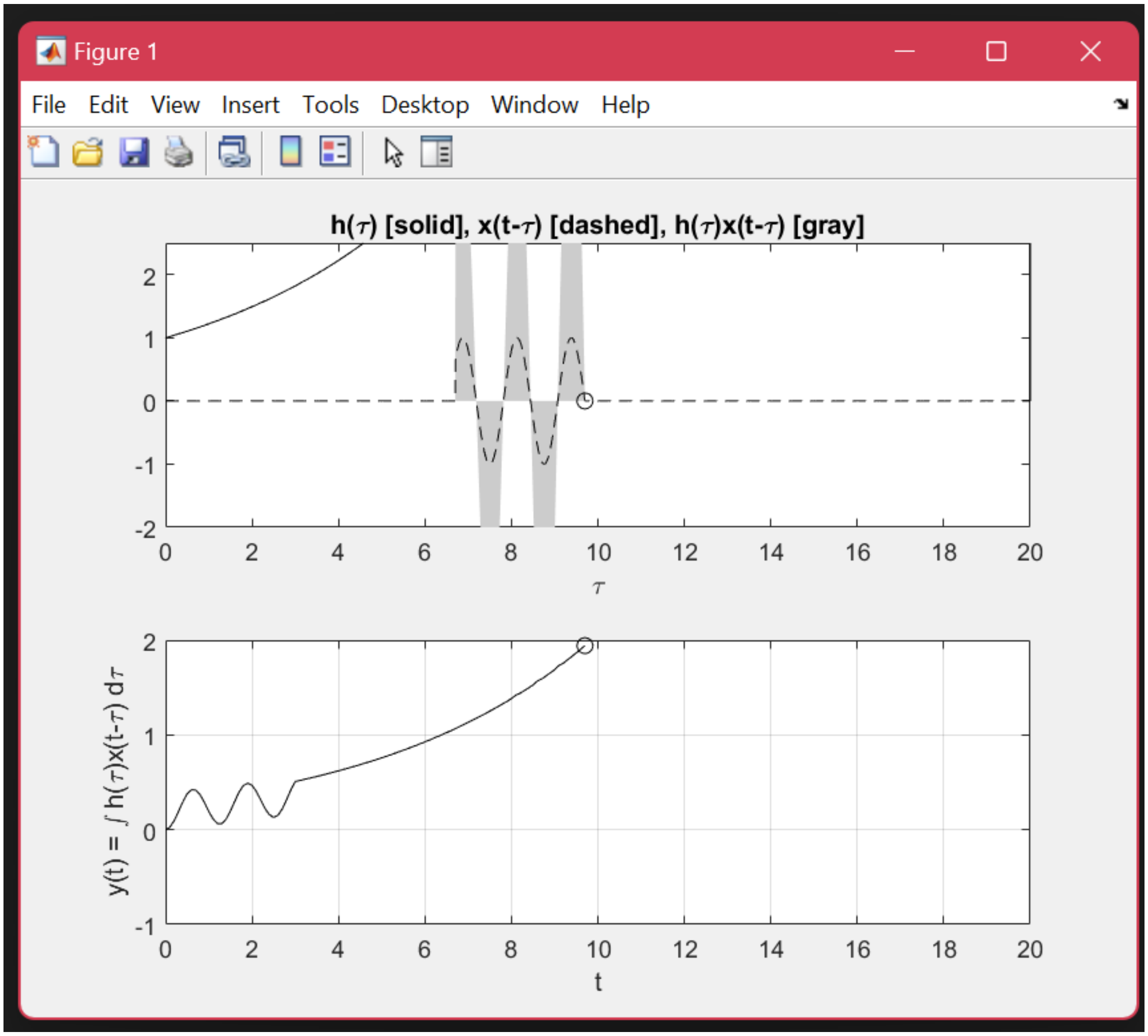


Figure 1
File Edit View Insert Tools Desktop Window Help
h(τ) [solid], x(t-τ) [dashed], h(τ)x(t-τ) [gray]
2
1
0
-1
-2
0 2 4 6 8 10 12 14 16 18 20
τ
y(t) = ∫ h(τ)x(t-τ) dτ
2
1
0
-1
0 2 4 6 8 10 12 14 16 18 20
t

h2:

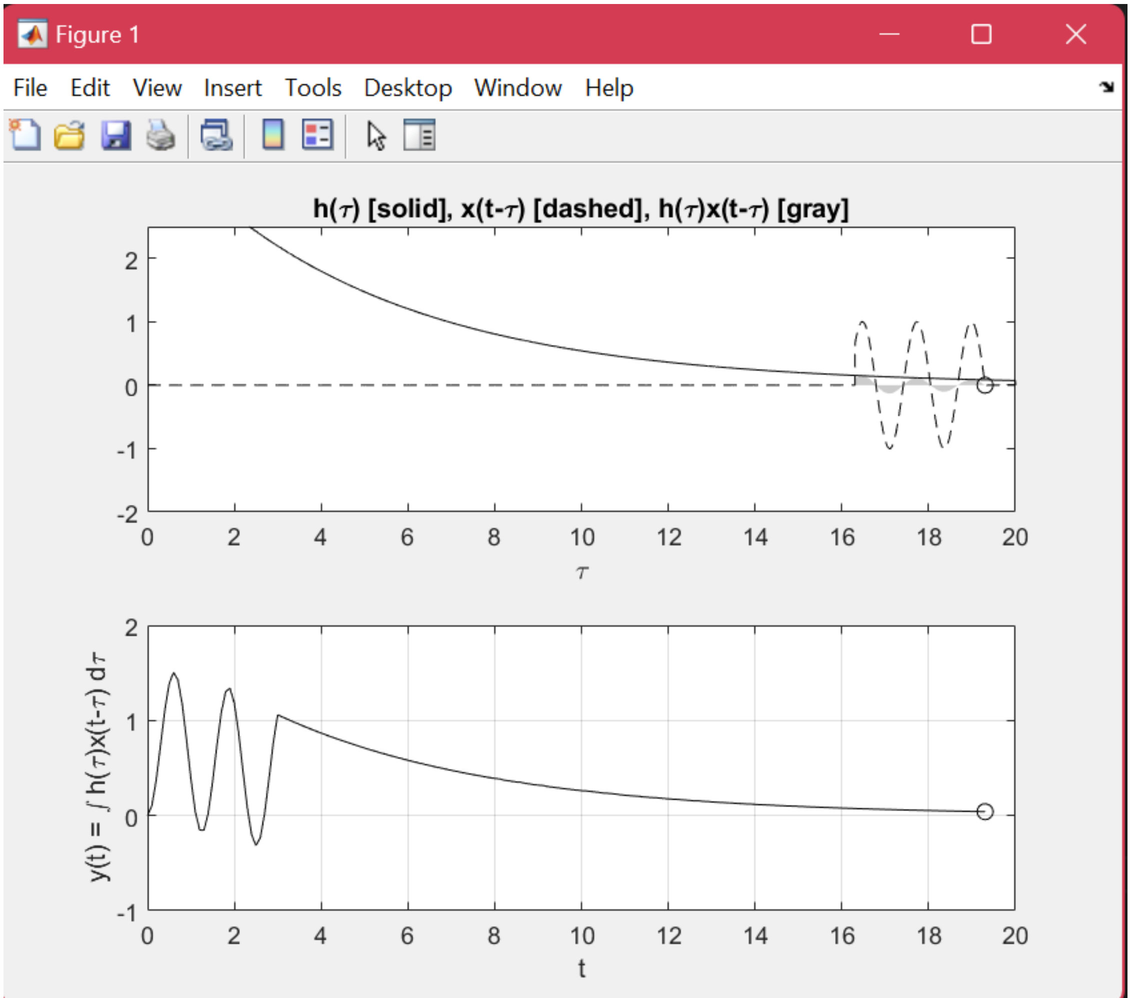


Figure 1
File Edit View Insert Tools Desktop Window Help
h(τ) [solid], x(t-τ) [dashed], h(τ)x(t-τ) [gray]
τ
y(t) = ∫ h(τ)x(t-τ) dτ
t

h3:

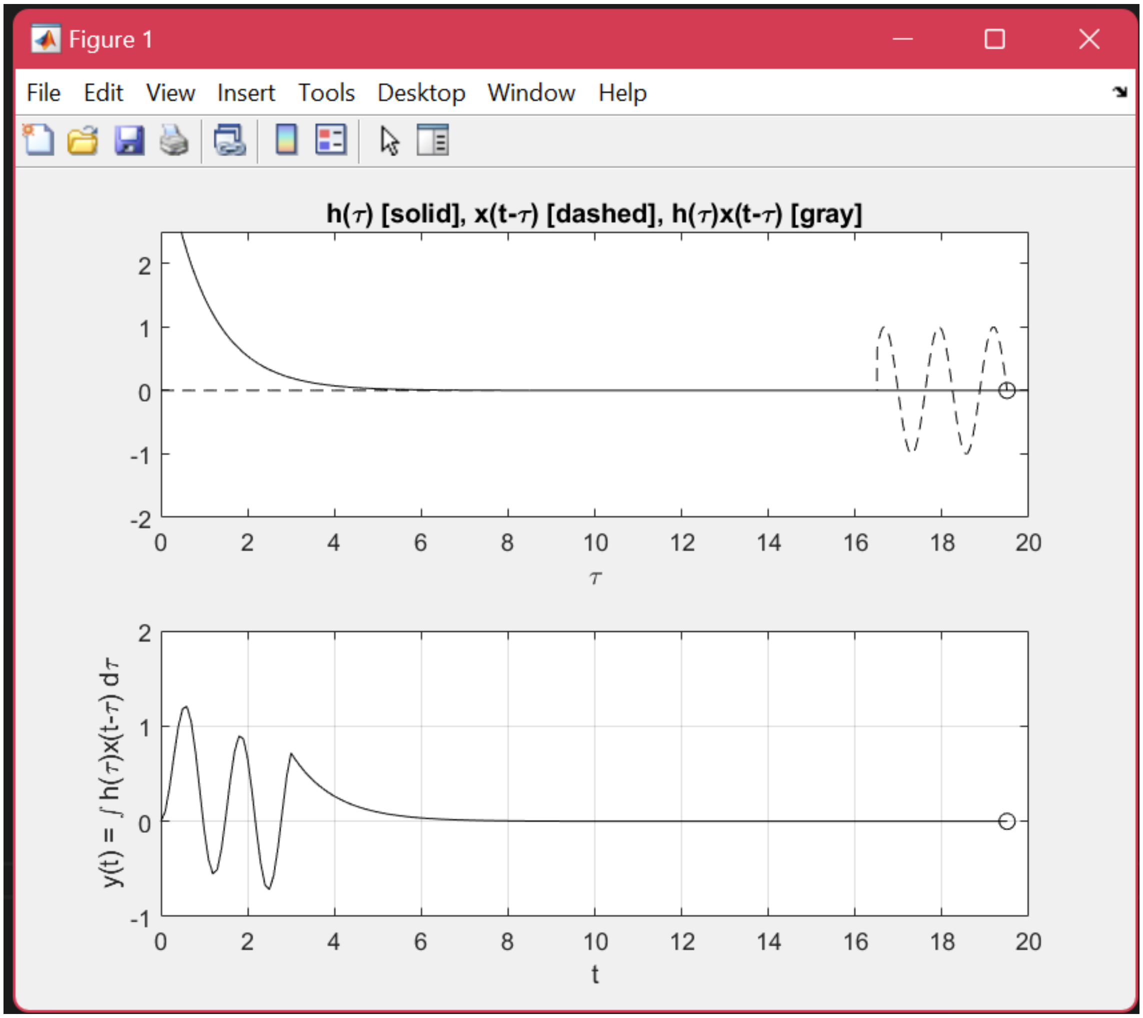
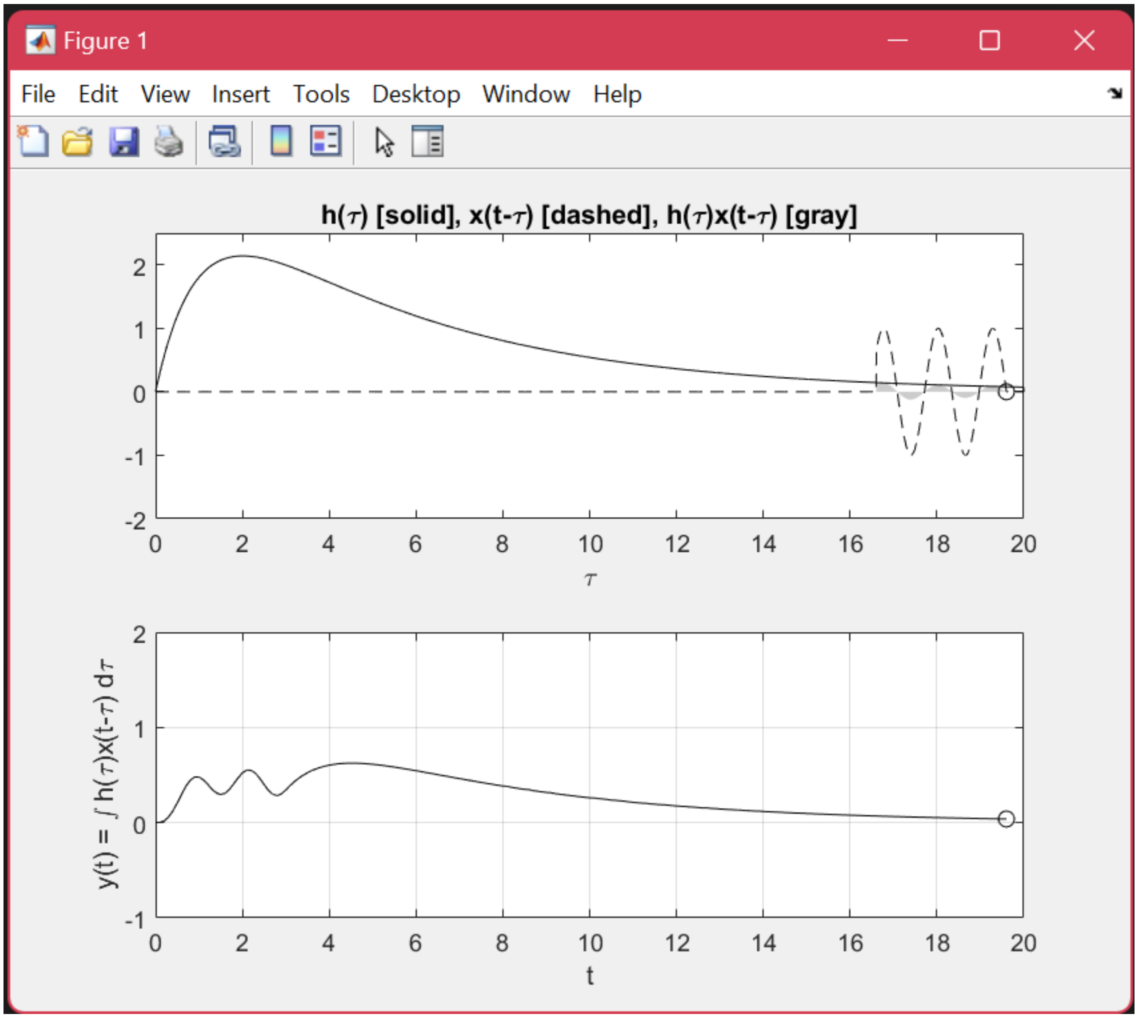


Figure 1
File Edit View Insert Tools Desktop Window Help
h(τ) [solid], x(t-τ) [dashed], h(τ)x(t-τ) [gray]
τ
y(t) = ∫ h(τ)x(t-τ) dτ
t

h4:



D. Discussion.

- **Problem D.2**
  Results: When two signals are convolved, the resulting signal's duration is equal to the sum of the durations of the original functions.