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 1
Assignment/Lab Title:	Working with Matlab, Visualization of Signals

Submission Date:	October 1,2023
Due Date:	October 1,2023

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

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

#Imports
import time
import numpy as np
import scipy.io as sci
import matplotlib.pyplot as plt
import sounddevice as sd

#Variable Declaration
color='g'

#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()
```

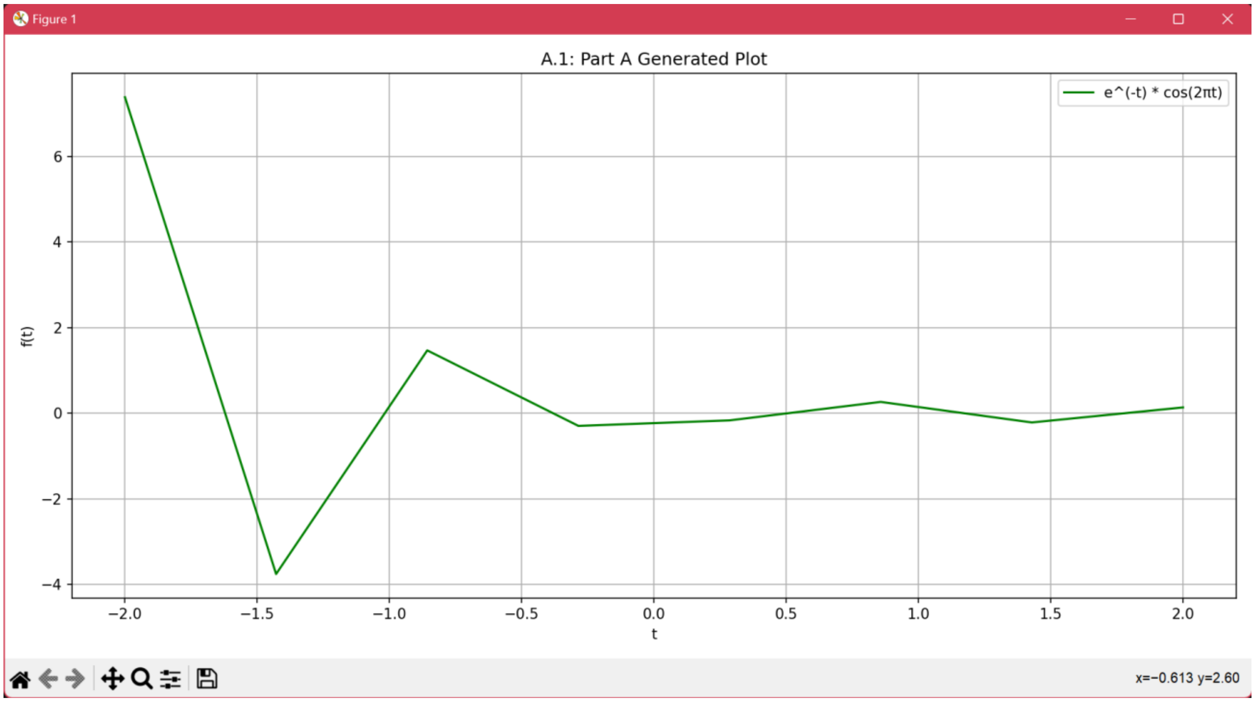
A. Anonymous functions and plotting continuous functions.

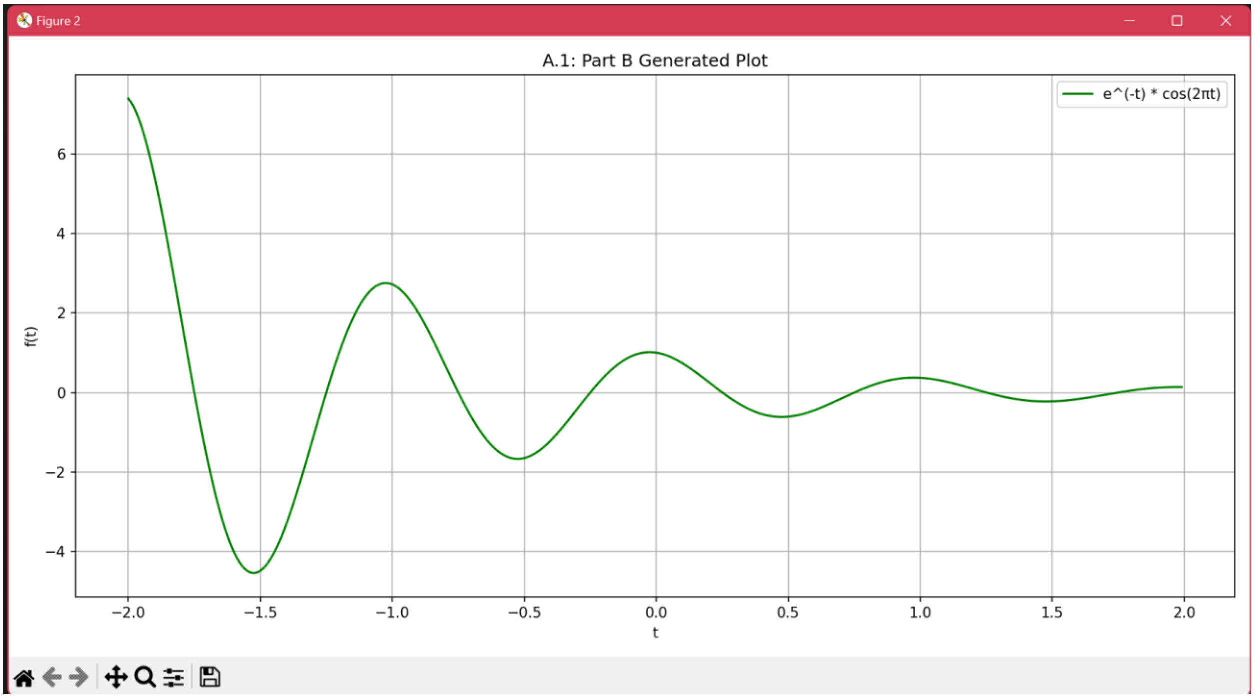
- **Problem A.1**
  Code:

```python
# Problem A.1:
# Figure 1.46: Plotting f(t) = e^(-t) * cos(2πt)
t = np.linspace(-2, 2, 8)
f_t = np.exp(-t) * np.cos(2 * np.pi * t)
plot(f_t, t, title='A.1: Part A Generated Plot', functionLabel='e^(-t) * cos(2πt)')

# Figure 1.47: Additional Points
t = np.arange(-2, 2, 0.01)
f_t = np.exp(-t) * np.cos(2 * np.pi * t)
plot(f_t, t, title='A.1: Part B Generated Plot', functionLabel='e^(-t) * cos(2πt)')
```

Results:



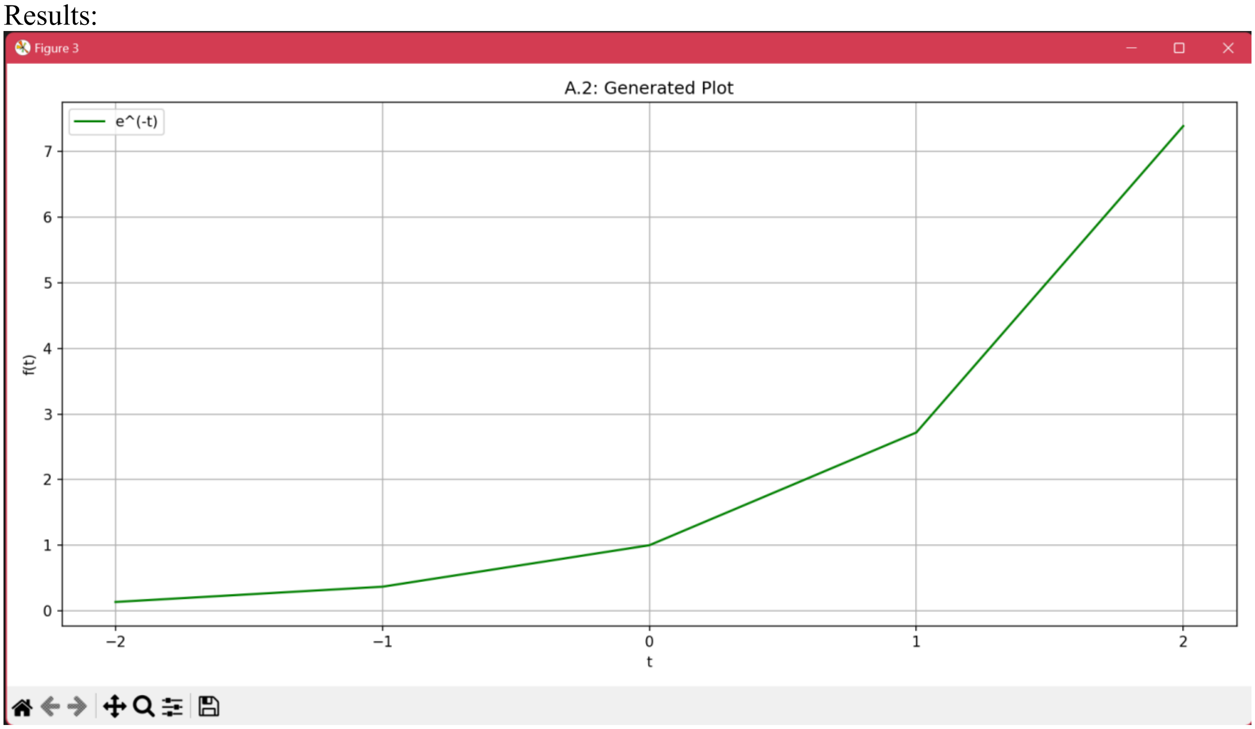


- **Problem A.2**

Code:

```
# Problem A.2:
t = np.linspace(-2, 2, 5)        You, last week • The Start …
f_t = np.exp(t)
plot(f_t, t, title='A.2: Generated Plot', functionLabel='e^(-t)')
plt.xticks(np.arange(-2, 2.01, 1))
```

Results:

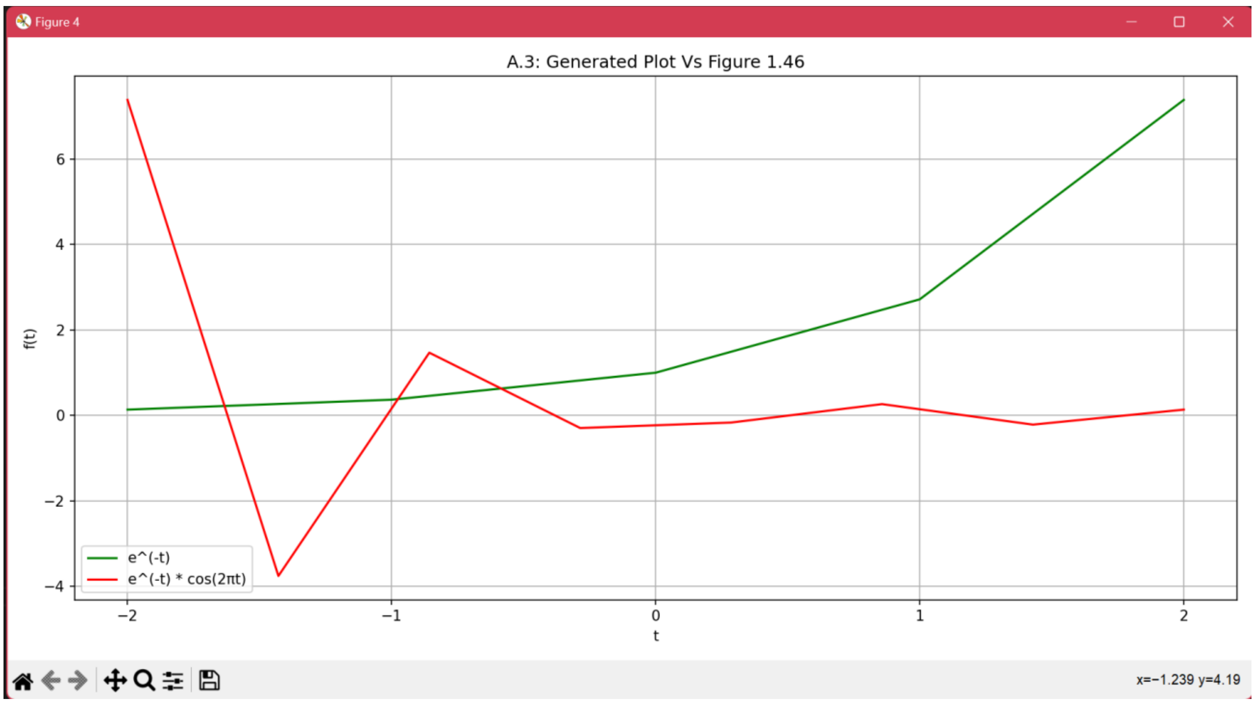


- **Problem A.3**

Code:

```
# Problem A.3:
# A.2 plot
t = np.linspace(-2, 2, 5)
f_t = np.exp(t)
plot(f_t, t, title='A.3: Generated Plot Vs Figure 1.46', functionLabel='e^(-t)')
plt.xticks(np.arange(-2, 2.01, 1))
# SuperImpose Figure 1.46 from A,1
t = np.linspace(-2, 2, 8)
f_t = np.exp(-t) * np.cos(2 * np.pi * t)
plot(f_t, t,newGraph=False, functionLabel='e^(-t) * cos(2πt)')
```

Results:



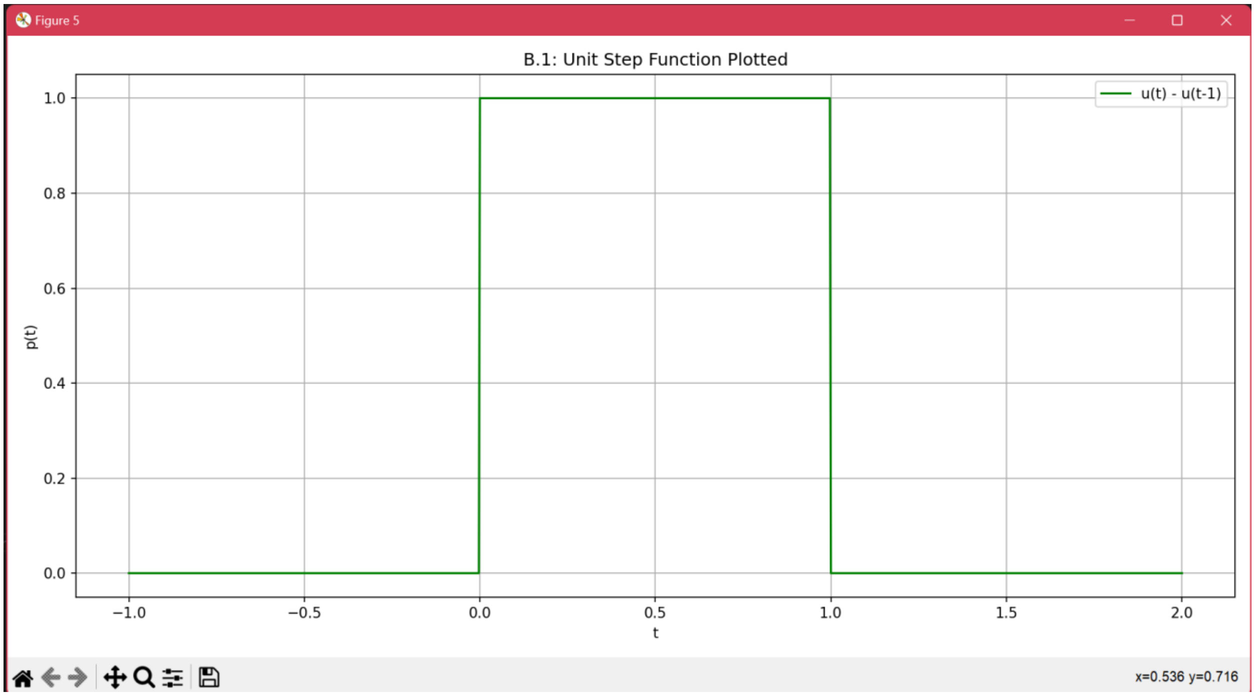
B. Time shifting and time scaling.

- **Problem B.1**

Code:

```
#Problem B.1:
t = np.linspace(-1, 2, 1000)
p_t = np.heaviside(t, 1) - np.heaviside(t - 1, 1)
plot(p_t, t, title='B.1: Unit Step Function Plotted', functionLabel='u(t) - u(t-1)', yLabel='p(t)')
```

Results:



- **Problem B.2**

  Code:

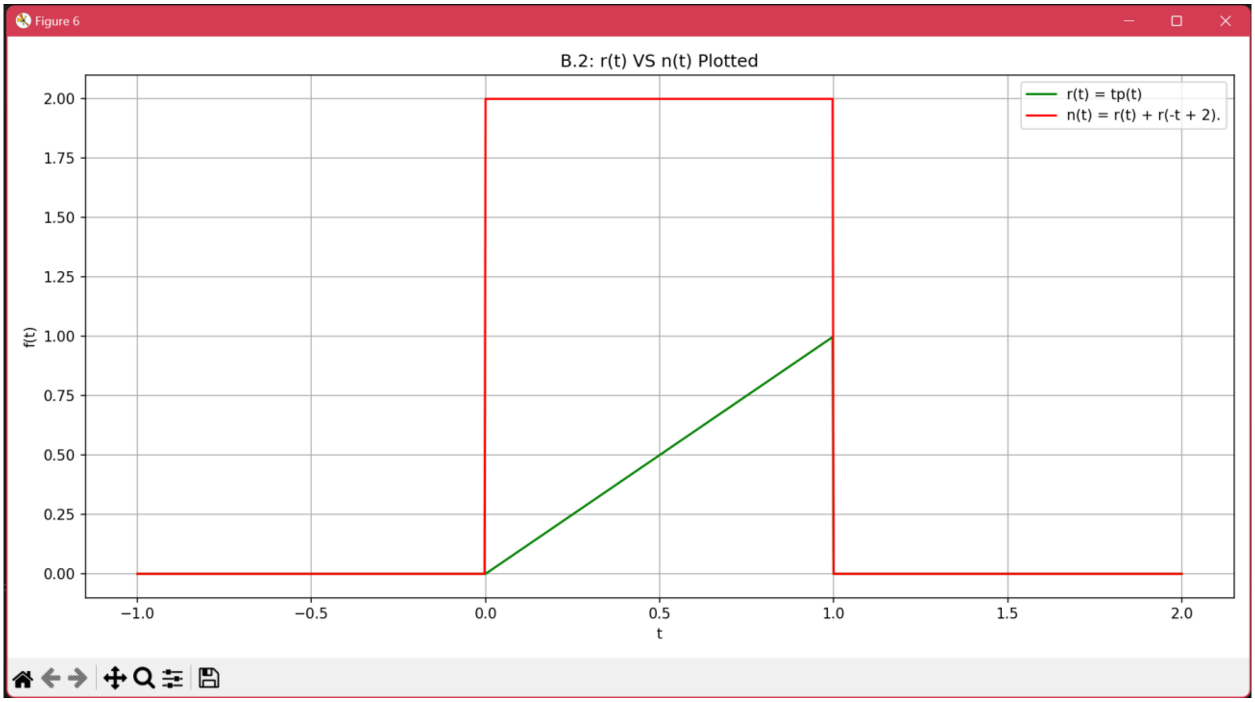
```
#Problem B.2:        You, last week • Update lab1main.py …
def r(t):
    return t * p_t

def n(t):
    return r(t) + r(-t + 2)

r_t = r(t)
n_t = n(t)

plot(r_t, t, title='B.2: r(t) VS n(t) Plotted', functionLabel='r(t) = tp(t)', yLabel='r(t)/n(t)')
plot(n_t, t, newGraph=False, functionLabel= 'n(t) = r(t) + r(-t + 2).')
```

  Results:

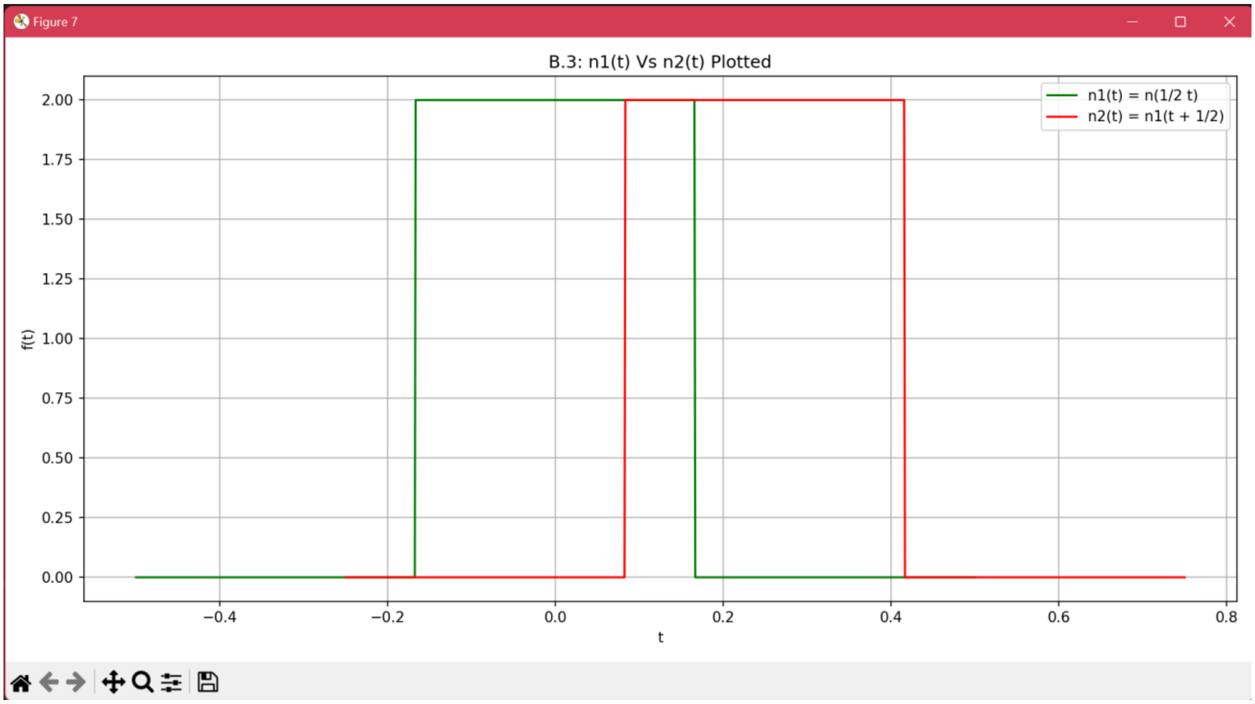


- **Problem B.3**

Code:

```
#Problem B.3:
#n1(t)
t = np.linspace(-1, 1, 1000)  # Reduce time values for n1 and n2
t = 0.5*t
n1_t = n(t)
plot(n1_t, t, title='B.3: n1(t) Vs n2(t) Plotted', functionLabel='n1(t) = n(1/2 t)', yLabel='n1(t)/n2(t)')
#n2(t)
t = np.linspace(-1, 1, 1000)
t = 0.5*(t + (1/2))
n2_t = n(t)  # Adjust the time values for n2
plot(n2_t, t, newGraph=False, functionLabel='n2(t) = n1(t + 1/2)')
```

Results:

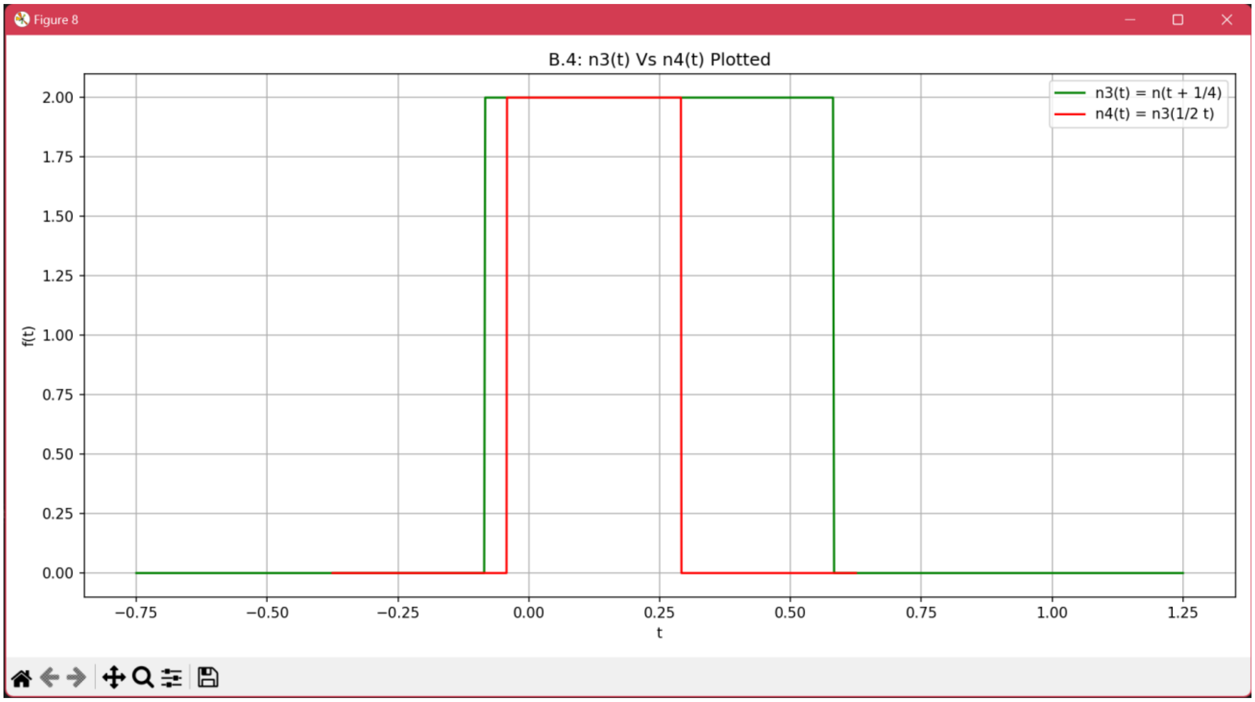


- **Problem B.4**

  Code:

```
#Problem B.4:
#n3(t)
t = np.linspace(-1, 1, 1000)
t = t+(1/4)
n3_t = n(t)
plot(n3_t, t, title='B.4: n3(t) Vs n4(t) Plotted', functionLabel='n3(t) = n(t + 1/4)', yLabel='n3(t)/n4(t)')
#n4(t)
t = np.linspace(-1, 1, 1000)
t = 1/2*(t + 1/4)
n4_t = n(t)
plot(n4_t, t, newGraph=False, functionLabel='n4(t) = n3(1/2 t)')
```

  Results:

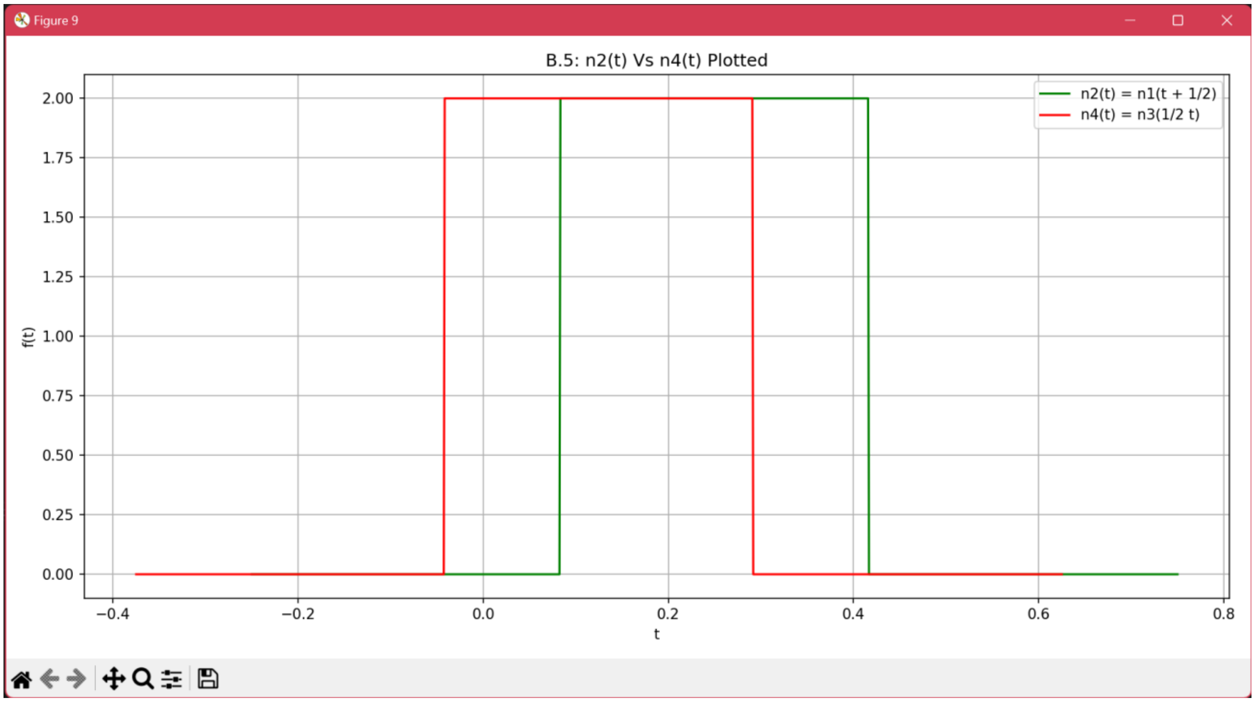


- **Problem B.5**

Code:

```
#Problem B.5:
#n2(t)
t = np.linspace(-1, 1, 1000)
t = 0.5*(t + (1/2))
n2_t = n(t)  # Adjust the time values for n2
plot(n2_t, t, title='B.5: n2(t) Vs n4(t) Plotted', functionLabel='n2(t) = n1(t + 1/2)', yLabel='n2(t)/n4(t)')
#n4(t)
t = np.linspace(-1, 1, 1000)
t = 1/2*(t + 1/4)
n4_t = n(t)
plot(n4_t, t, newGraph=False, functionLabel='n4(t) = n3(1/2 t)')
```

Results:



The unit step functions, share all qualities except that they are out of phase with one another.

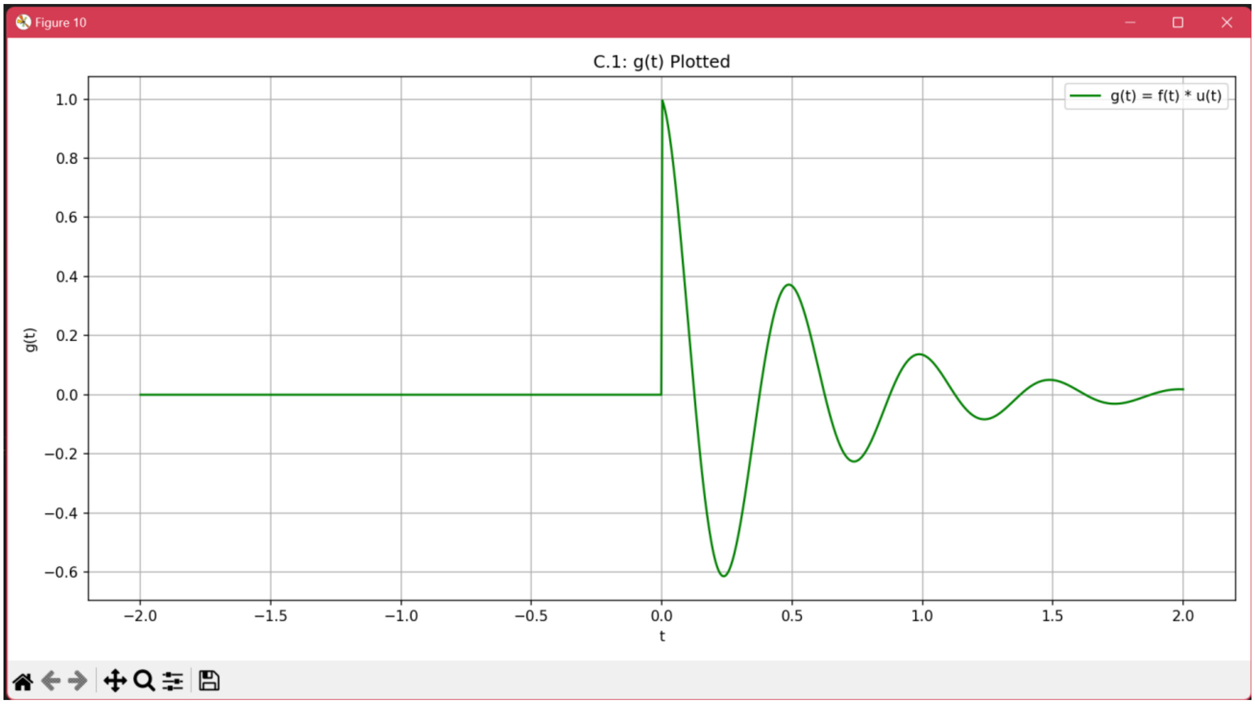
C. Visualizing operations on the independent variable and algorithm vectorization.

- **Problem C.1**

Code:

```
#Problem C.1 : g(t) = u(t) * f(t)
t = np.linspace(-2, 2, 1000)
f_t = np.exp(-2 * t) * np.cos(4 * np.pi * t)
u_t = np.heaviside(t, 1)
g_t = u_t * f_t
plot(g_t, t, title='C.1: g(t) Plotted', functionLabel= "g(t) = f(t) * u(t)", xLabel='t', yLabel='g(t)')
```

Results:

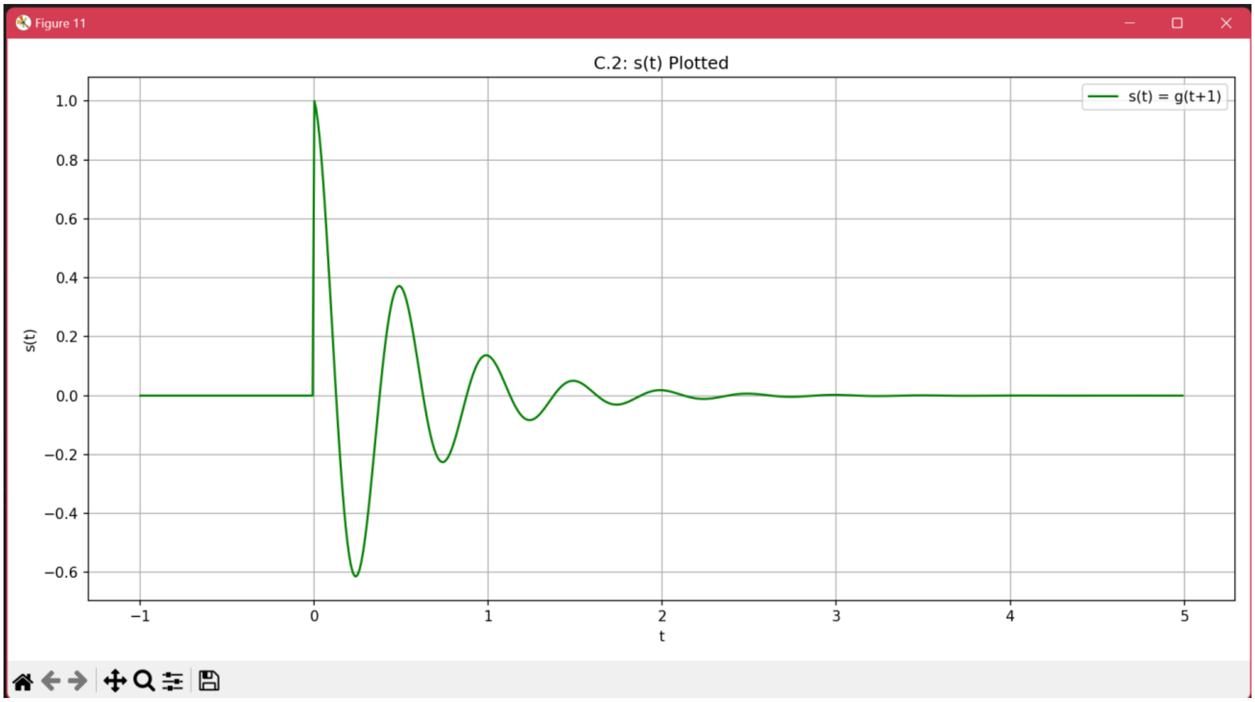


- **Problem C.2**

  Code:

```
#Problem C.2: Plotting s(t)
t = np.arange(-2, 4, 0.01)
t = t + 1
f_t = np.exp(-2 * t) * np.cos(4 * np.pi * t)
u_t = np.heaviside(t, 1)
g_t = u_t * f_t
plot(g_t,t, title='C.2: s(t) Plotted', functionLabel= "s(t) = g(t+1)", xLabel='t', yLabel='s(t)')
```

  Results:



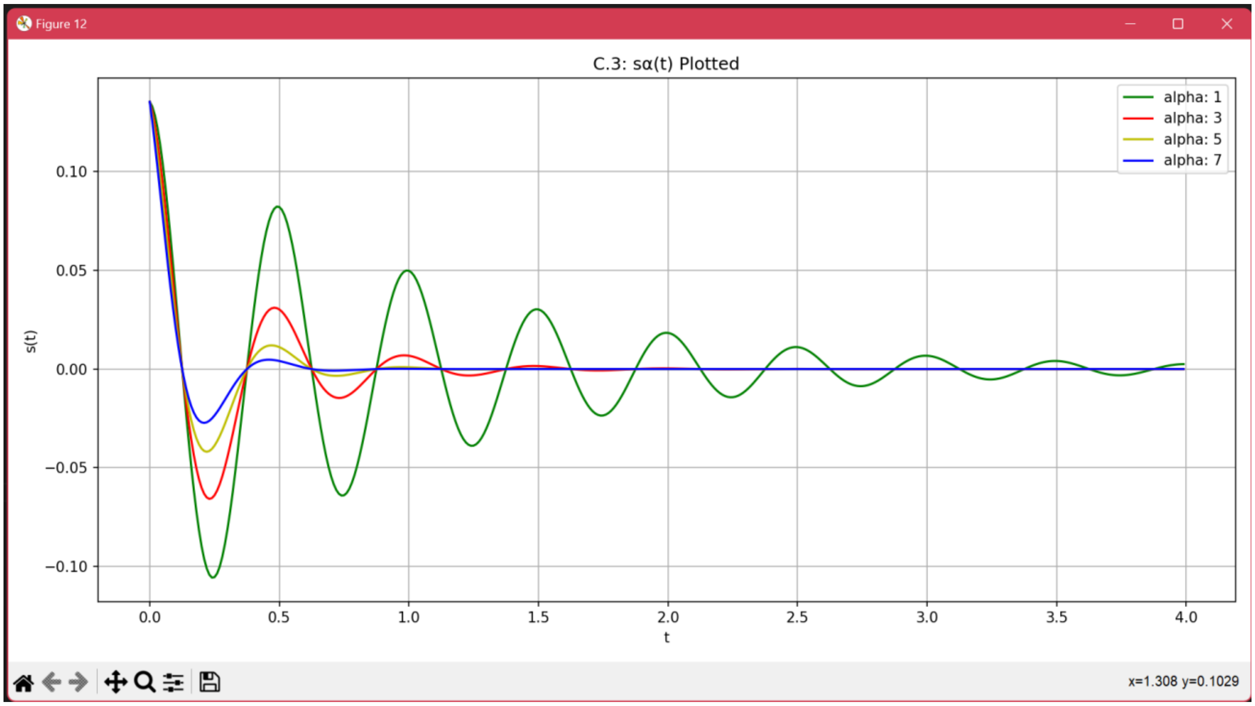
- **Problem C.3**

  Code:

```
#Problem C.3: g(t) for different alpha values
t = np.arange(0, 4, 0.01)
u_t = np.heaviside(t, 1)
alpha = [1, 3, 5, 7]

newGraph = True
for i in range(0, 4):
    if(i > 0):
        newGraph = False
    f_t = np.exp(-2) * np.exp(-1 * alpha[i] * t) * np.cos(4 * np.pi * t)
    s_t = f_t * u_t
    functionLabel = "alpha: " + str(alpha[i])
    plot(s_t, t, newGraph=newGraph, title='C.3: sα(t) Plotted', functionLabel= functionLabel , xLabel='t', yLabel='s(t)')
```

  Results:



D. Array indexing.

```
#Read MATLAB data file
data = sci.loadmat('lab1\ELE532_Lab1_Data.mat')

#Load data
A = data['A']
B = data['B']
x_audio = data['x_audio']
```

A

- **Problem D.1**
  Code:

```python
#Problem D.1 : Modifications to Array A
#(a) A(:)
a = A.flatten()
print("(a) Flattened A to 1D Array:")
print(a)

#(b) A([ 2 4 7 ])
b_indexes = np.array([1, 3, 6])
b = A.flatten()[b_indexes]
print("\n(b) Extracted elements based on index:")
print(b)

#(c) [ A >= 0.2 ]
c_mask = A >= 0.2
print("\n(c) Boolean mask for all elements >= 0.2:")
print(c_mask)

#(d) A([ A >= 0.2 ])
d = A[c_mask]
print("\n(d) Extracted elements where A >= 0.2:")
print(d)

#(e) A([ A >= 0.2 ]) = 0        You, now • Uncommitted changes
A[c_mask] = 0
print("\n(e) Setting elements >= 0.2 to 0:")
print(A)
```

Results:

```
The Length of the Matrix is 4 Cells

(a) Flattened A to 1D Array:
[ 0.5377 -1.3077 -1.3499 -0.205   1.8339 -0.4336  3.0349 -0.1241 -2.2588
  0.3426  0.7254  1.4897  0.8622  3.5784 -0.0631  1.409   0.3188  2.7694
  0.7147  1.4172]

(b) Extracted elements based on index:
[-1.3077 -0.205   3.0349]

(c) Boolean mask for all elements >= 0.2:
[[ True False False False]
 [ True False  True False]
 [False  True  True  True]
 [ True  True False  True]
 [ True  True  True  True]]

(d) Extracted elements where A >= 0.2:
[0.5377 1.8339 3.0349 0.3426 0.7254 1.4897 0.8622 3.5784 1.409  0.3188
 2.7694 0.7147 1.4172]

(e) Setting elements >= 0.2 to 0:
[[ 0.      -1.3077 -1.3499 -0.205 ]
 [ 0.      -0.4336  0.      -0.1241]
 [-2.2588  0.      0.       0.    ]
 [ 0.      0.      -0.0631  0.    ]
 [ 0.      0.      0.       0.    ]]
```

- **Problem D.2**

Code:

```python
#Problem D.2
rows, cols = B.shape

#(a) Set values below 0.01 to zero
start = time.time()
for i in range(rows):
    for j in range(cols):       You, 16 hours ago • Finished Code …
        if abs(B[i, j]) < 0.01:
            B[i, j] = 0
end = time.time()
nestedTime = end - start

#(b) Indexing approach
start = time.time()
B[B < 0.01] = 0
end = time.time()
IndexTime = end - start

#(c) Check execution time
print("\nNested Time: " + str(nestedTime) + "\nIndex  Time: " + str(IndexTime) + "\n")
```

Results:

```
Nested Time: 0.020335674285888672
Index  Time: 0.0004963874816894531
```

- **Problem D.3**

  Code:

```
#Problem D.3
#Create a Copy of the data
audio_X = np.copy(x_audio)

#Threshold for compression
threshold = 0.1

#Initialize counter for zero-valued samples
zero_samples = 0

#Nested Loop to iterate through the array and apply the compression
for i in range(len(audio_X)):
    if abs(audio_X[i]) < threshold:
        audio_X[i] = 0
        zero_samples += 1

#Print the number of zero-valued samples
print(f"\nNumber of zero-valued samples: {zero_samples}")

#Play the original audio
sd.play(x_audio, 8000)
sd.wait()

#Play the compressed audio
sd.play(audio_X, 8000)
sd.wait()


#EOF CONTENT

#Display all plots
plt.show()

#End of code
```

  Results:

```
Number of zero-valued samples: 12193
```