Course Number	532
Course Title	ELE
Semester/Year	Fall 2023
Instructor	Javad Alirezaie
TA Name	Xiaodan Bi

Lab/Tutorial Report No.	Lab 1

Report Title	Lab #1 Report

Section No.	11
Group No.	-
Submission Date	October 4, 2022
Due Date	October 4, 2022

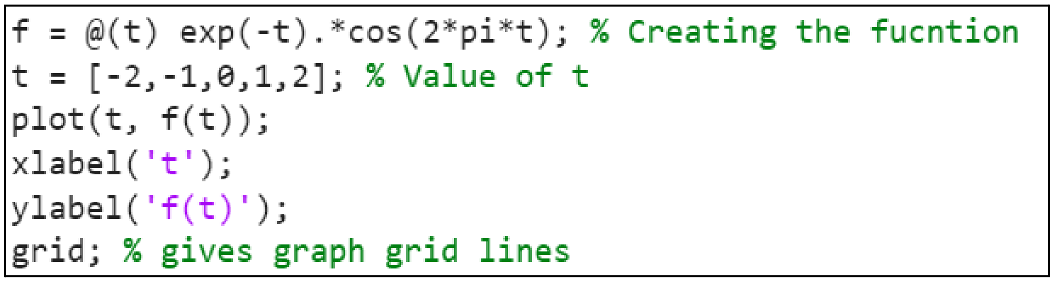
Student Name	Student ID	Signature*
Hanana Gohir	500946695	[illegible]

(Note: remove the first 4 digits from your student ID)

Part A:

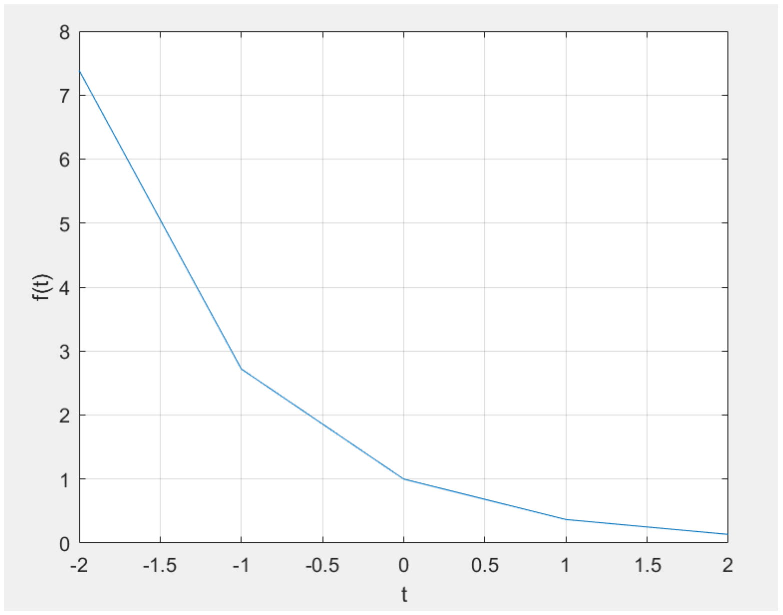
Problem A.1: *Section 1.11-1, page 126. Generate and plot the graphs as shown in Figures 1.46 and 1.47 on page 127.*

<u>Matlab code for figure 1.46</u>

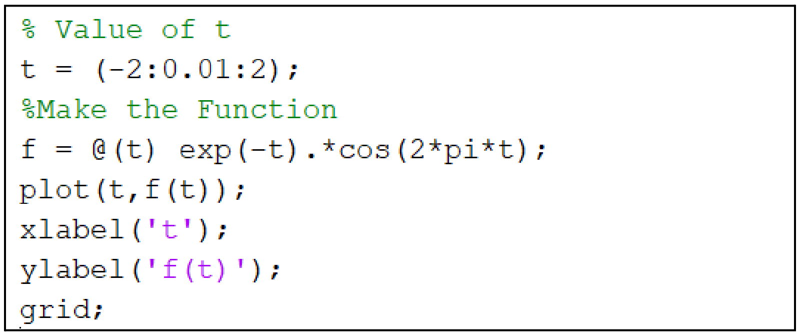


```
f = @(t) exp(-t).*cos(2*pi*t); % Creating the fucntion
t = [-2,-1,0,1,2]; % Value of t
plot(t, f(t));
xlabel('t');
ylabel('f(t)');
grid; % gives graph grid lines
```

<u>Graph 1.46 on Matlab</u>

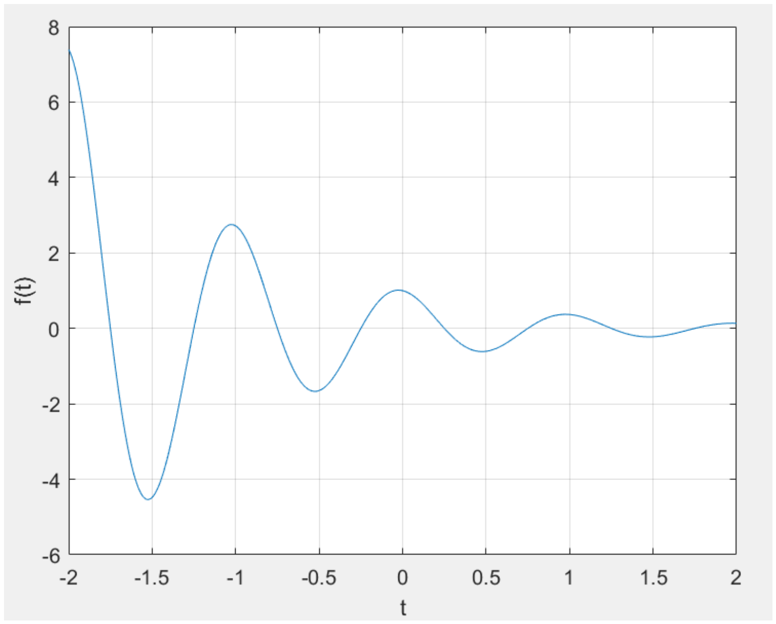


<u>Matlab code for figure 1.47</u>



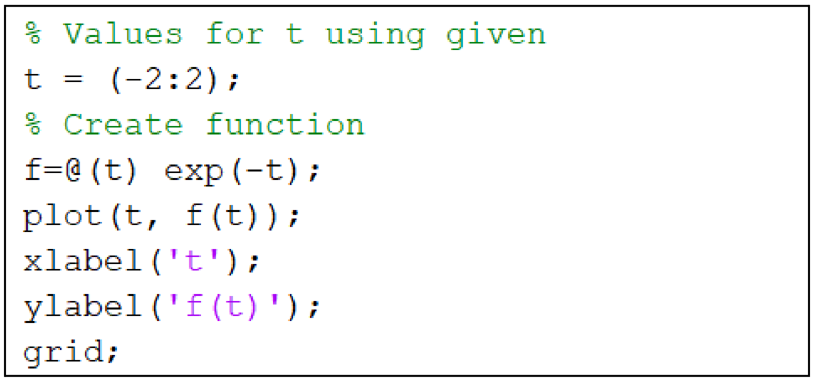
```
% Value of t
t = (-2:0.01:2);
%Make the Function
f = @(t) exp(-t).*cos(2*pi*t);
plot(t,f(t));
xlabel('t');
ylabel('f(t)');
grid;
```

Graph 1.47 on Matlab



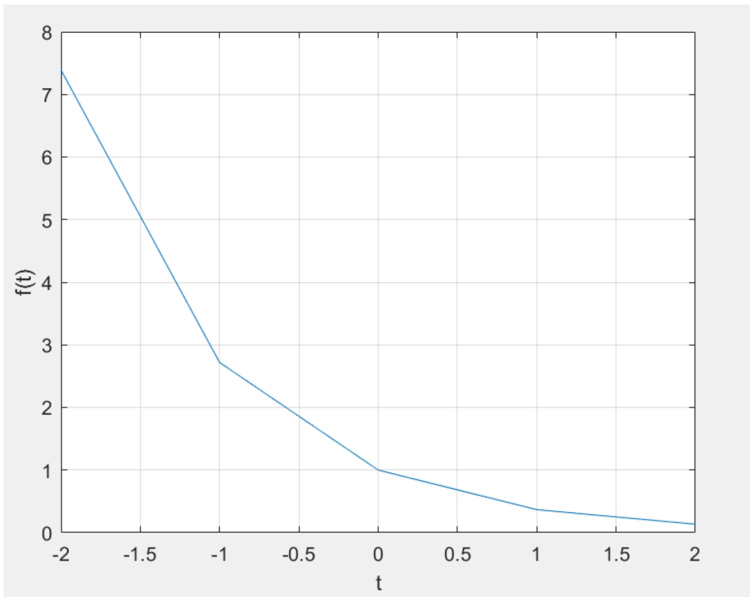
Problem A.2: *Plot the function e −t for t taking on integer values contained in [−2, -1, 0, 1, 2]; you can generate these values using the Matlab command t=[-2:2].*

Matlab code for function e −t contained in [−2, -1, 0, 1, 2]



```
% Values for t using given
t = (-2:2);
% Create function
f=@(t) exp(-t);
plot(t, f(t));
xlabel('t');
ylabel('f(t)');
grid;
```

Graph for function e −t contained in $-2 \leq t \leq 2$



Problem A.3: *Compare the results of Problem A.2 with Figure 1.46 in Problem A.1.*

The graphs of both A.2 and 1.46 are identical. This means that the functions used in both graphs are equal to one another. Therefore, `f = @(t) exp(-t).*cos(2*pi*t);` and `f=@(t) exp(-t);` are the same graph as they both exponentially decrease with three slope changes of the same magnitude at t= -1, t=0 and t=1.

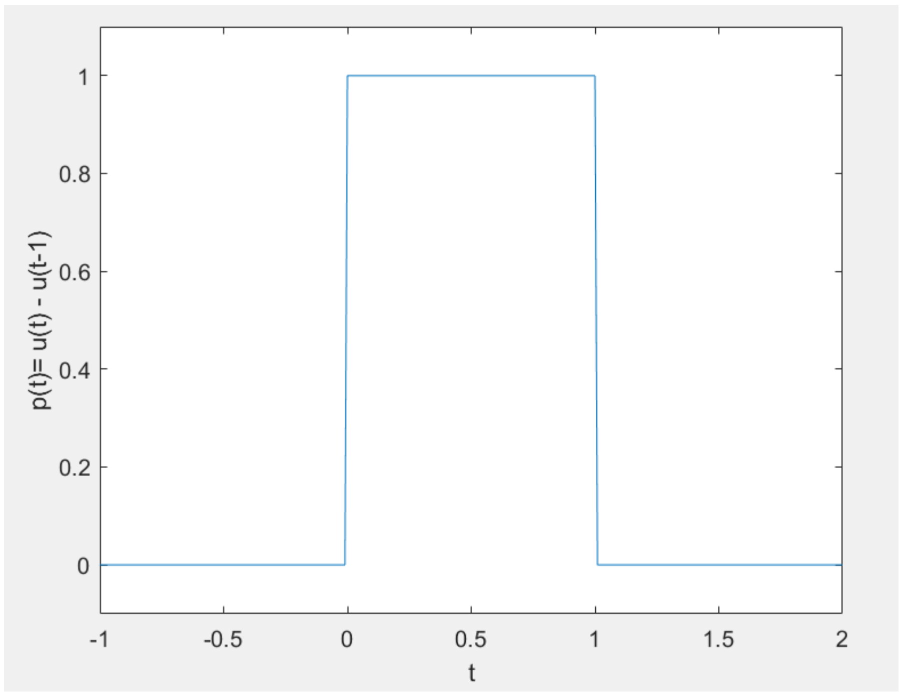
Part B

Problem B.1: *Section 1.11-2, page 128. Generate and plot p(t) as shown in Figure 1.50 on page 129.*

Matlab Code for Figure 1.50

```
%Set the value for t
t= (-2:0.01:2);
%Create the fucntion
p =  @(t) 1.0*((t>=0) & (t<=1));
plot(t,p(t));
axis([-1 2 -0.1 1.1]);
xlabel('t');
ylabel('p(t)= u(t) - u(t-1)');
```

Matlab Graph for Figure 1.50

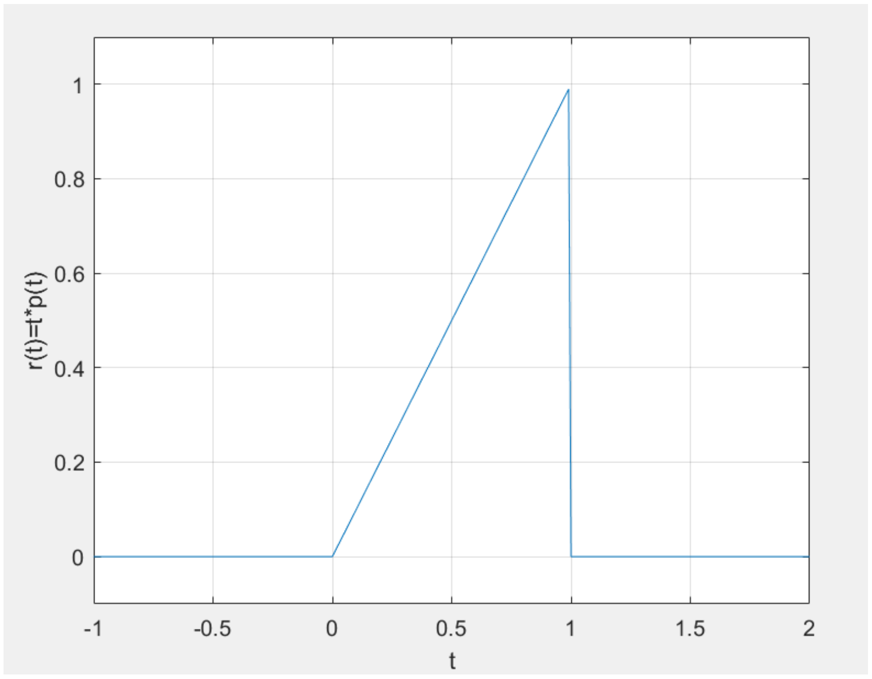


Problem B.2: *Use p(t) defined in Problem B.1 to generate and plot functions r(t) = tp(t) and n(t) = r(t) + r(−t + 2).*

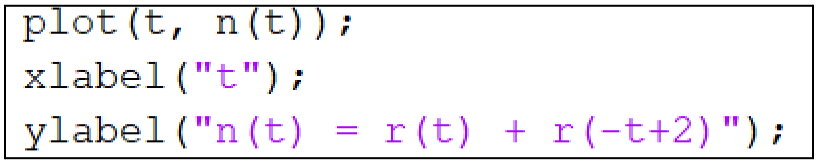
Matlab Code for r(t) = tp(t)

```
%Set value for t
t=(-2:0.01:2);
%Create the function
u = @(t)1.0.*(t>=0);
p = @(t) u(t)-u(t-1);
r= @(t) t.*p(t);
n= @(t) r(t) + r(-t+2);
plot(t,r(t));
xlabel('t');
ylabel('r(t)=t*p(t)');
axis([-1 2 -0.1 1.1]);
grid;
```

Matlab Graph for r(t) = tp(t)

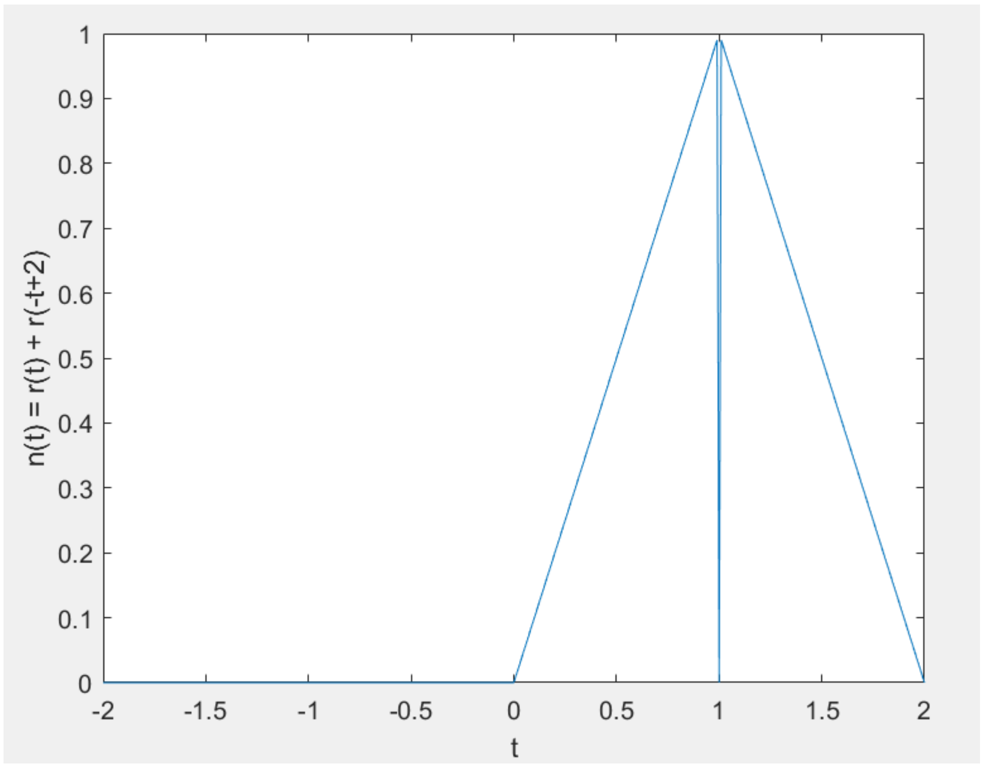


Matlab Code for *n(t) = r(t) + r(−t + 2).*



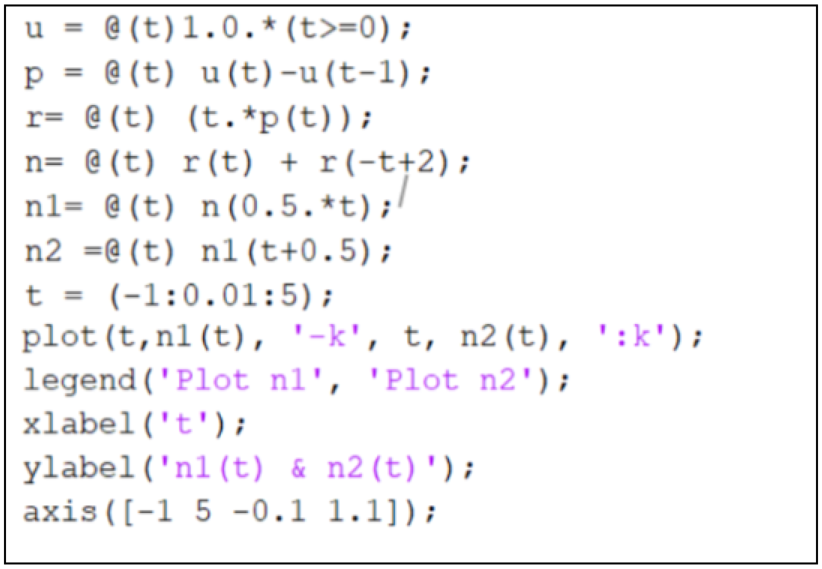
```
plot(t, n(t));
xlabel("t");
ylabel("n(t) = r(t) + r(-t+2)");
```

Matlab Graph for *n(t) = r(t) + r(−t + 2).*



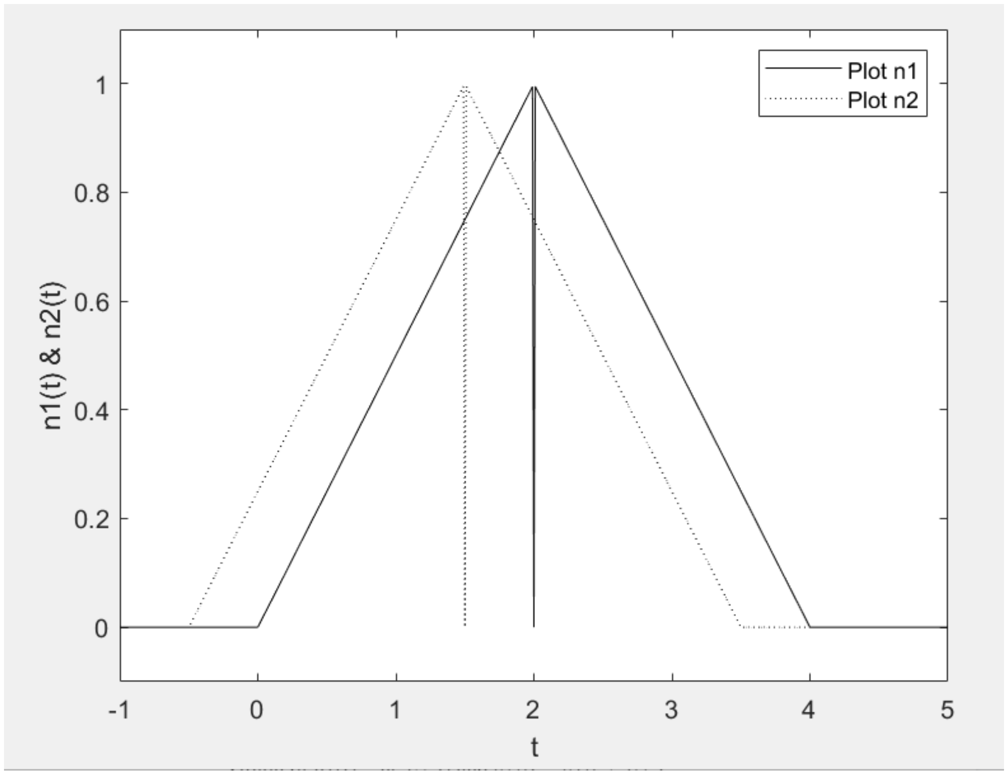
Problem B.3: *Plot the following two signals: n1(t) = n(1/2 t), n2(t) = n1(t + 1/2).*

Matlab Code for n1(t) = n(1/2 t) and *n2(t) = n1(t + 1/2).*



```
u = @(t)1.0.*(t>=0);
p = @(t) u(t)-u(t-1);
r= @(t) (t.*p(t));
n= @(t) r(t) + r(-t+2);
n1= @(t) n(0.5.*t);
n2 =@(t) n1(t+0.5);
t = (-1:0.01:5);
plot(t,n1(t), '-k', t, n2(t), ':k');
legend('Plot n1', 'Plot n2');
xlabel('t');
ylabel('n1(t) & n2(t)');
axis([-1 5 -0.1 1.1]);
```

Graph of n1(t) = n(1/2 t) and *n2(t) = n1(t + 1/2).*

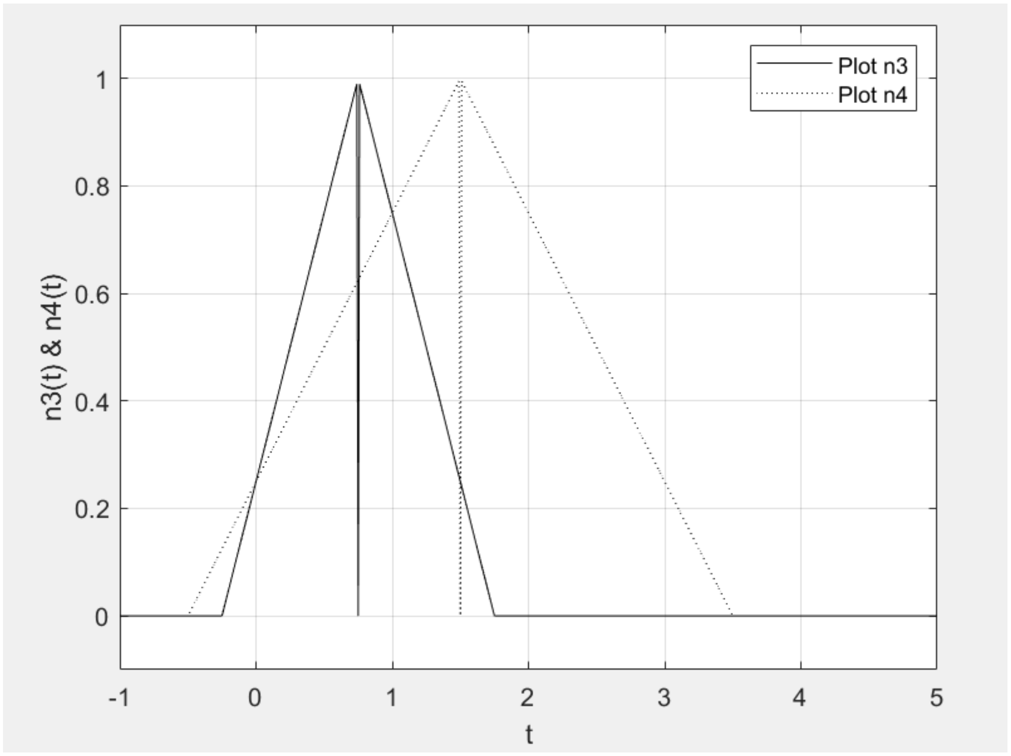


Problem B.4: *Plot the following two signals: n3(t) = n(t + 1/4), n4(t) = n3(1/2 t).*

<u>Matlab Code for *n3(t) = n(t + 1/4), n4(t) = n3(1/2 t).*</u>

```
>> r = @(t) (t.*p(t));
>> u = @(t) 1.0.*(t>=0);
p = @(t) u(t)-u(t-1);
r = @(t) (t.*p(t));
n = @(t) r(t)+r(-t+2);
n3 = @(t) n(t+0.25);
n4 = @(t) n3(0.5.*t);
t = (-1:0.01:5);
>> plot(t,n3(t), '-k',t,n4(t),':k');
>> legend ('Plot n3' ,'Plot n4');
>> xlabel('t');
>> ylabel('n3(t) & n4(t)');
>> axis([-1 5 -0.1 1.1]);
>> grid;
```

<u>Graph of *n3(t) = n(t + 1/4), n4(t) = n3(1/2 t).*</u>



Problem B.5: *Compare n4(t) and n2(t); explain any observed differences and/or similarities.*

The graphs of n4(t) and n2(t) are identical. this means that the functions used in both graphs are equal to one another. Therefore, n4(t) = n3(1/2 t) is equivalent to *n2(t) = n1(t + 1/2).*Furthermore, both graphs start with an increasing slope of the same magnitude at (-0.48, 0.01), reach a maximum at (0.74, 0.99) and then have a negative slope of the same magnitude at (1.74, 0.01).

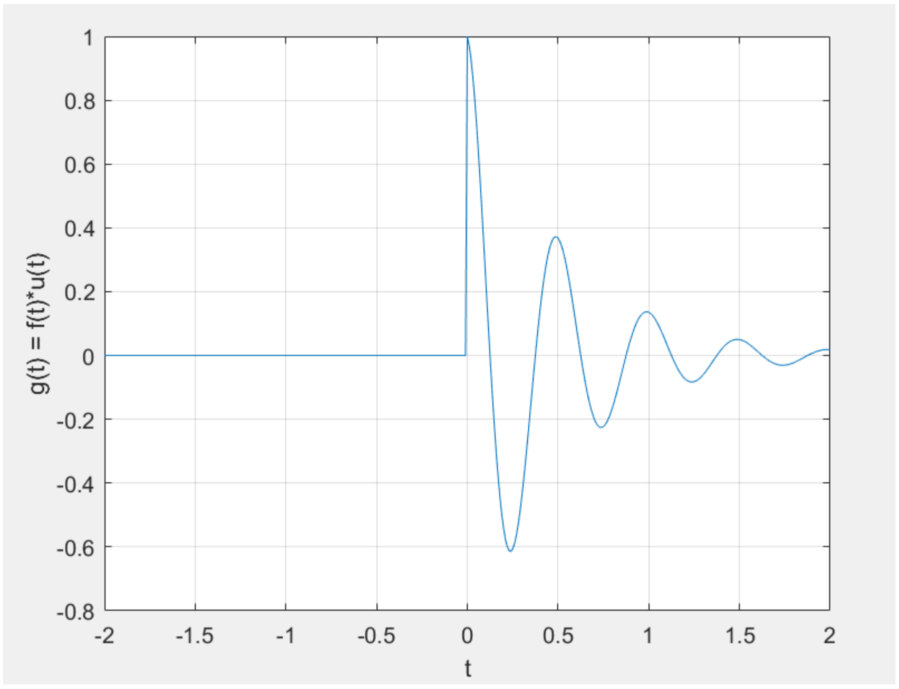
Part C

Problem C.1: s] *Section 1.11-3, Lathi, Matlab Session 1, page 130. Follow the steps in Section 1.11-3, but instead generate g(t) = f(t)u(t) where f(t) = e −2t cos 4πt.*

Matlab Code for *g(t) = f(t)u(t) where f(t) = e −2t cos 4πt.*

```
%Create f(t)
f = @(t) exp(-2.*t).*cos(4*pi*t);
t=(-2:0.01:2);
%Create u(t)
u = @(t) 1.0.*(t>=0);
axis([-2 2 -0.1 1.1]);
%Create g(t)
g = @(t) f(t).*u(t);
t = (-2:0.01:2);
plot(t,g(t));
xlabel('t');
ylabel('g(t) = f(t)*u(t)');
grid;
```

Graph of *g(t) = f(t)u(t) where f(t) = e −2t cos 4πt.*

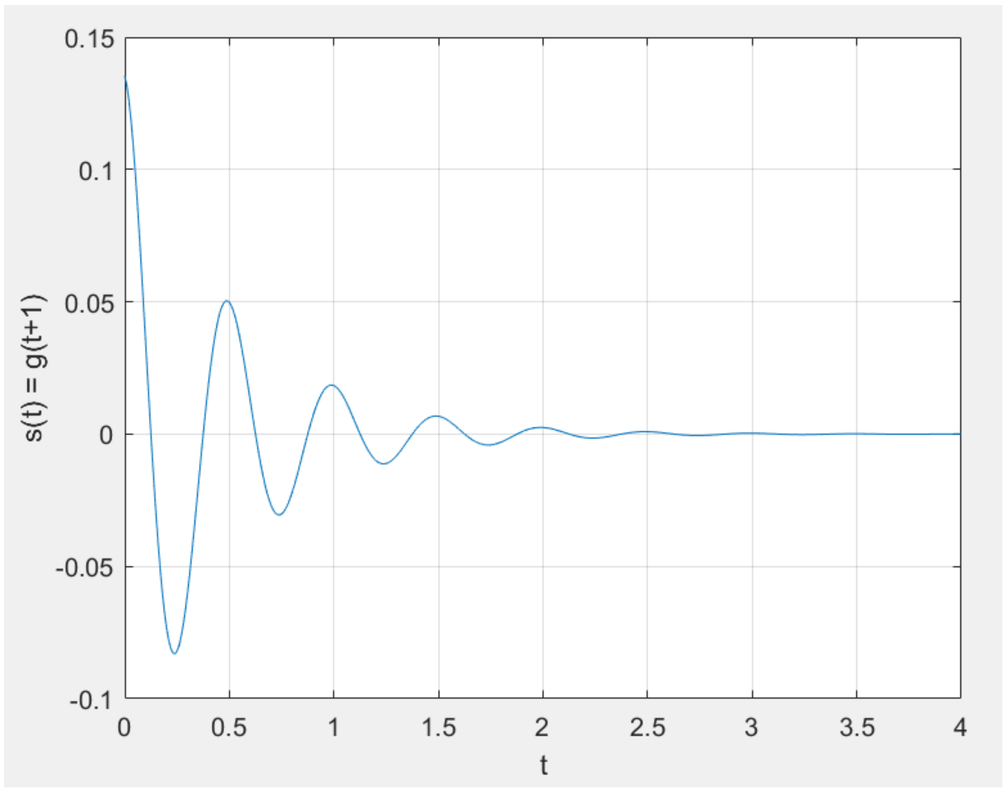


Problem C.2: *Using g(t) as described in Problem C.1, generate and plot s(t) = g(t + 1) for t = [0 : 0.01 : 4].*

Matlab Code for *s(t) = g(t + 1) for t = [0 : 0.01 : 4]*

```
>> %Create f(t)
f = @(t) exp(-2.*t).*cos(4*pi*t);
t=(-2:0.01:2);
%Create u(t)
u = @(t) 1.0.*(t>=0);
axis([-2 2 -0.1 1.1]);
%Create g(t)
g = @(t) f(t).*u(t);
t = (-2:0.01:2);
%Create s(t)
s = @(t) g(t+1);
t = (0:0.01:4);
plot(t,s(t));
xlabel("t");
ylabel('s(t) = g(t+1)');
grid;
```

Graph of *s(t) = g(t + 1) for t = [0 : 0.01 : 4]*

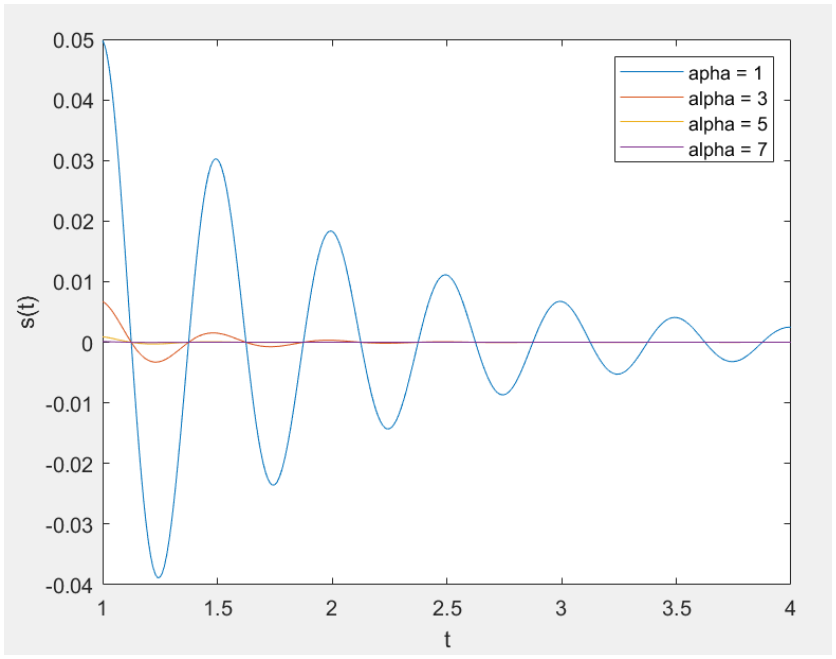


Problem C.3: *Plot sα(t) = e −2 e −αt cos(4πt)u(t) for α ∈ {1, 3, 5, 7} in one figure for t = [0 : 0.01 : 4]. For this plot you can use the for command for a loop structure (to learn more about this command type help for at the Matlab prompt). Also try to use matrix and vector operations to generate and plot the desired functions by following the steps of Section B.7-6, page 49.*

<u>Matlab Code for *sα(t) = e −2 e −αt cos(4πt)u(t) for α ∈ {1, 3, 5, 7}*</u>

```
%Create u(t) function
u = @(t) 1.0.*(t>=0);
t = (1:0.01:4);
%Create a matrix of zeros that are 401by4 matrix
matrix = zeros(401,4);
for alpha = 1:2:7
%loop and then the fucntion is made
s = @(t)exp(-2).*exp(-alpha.*t).*cos(4*pi*t).*u(t);
%plot fucntion
plot(t,s(t));
xlabel('t');
ylabel('s(t)');
hold on;
end
legend('apha = 1', 'alpha = 3','alpha = 5','alpha = 7');
hold off;
```

<u>Graph of *sα(t) = e −2 e −αt cos(4πt)u(t) for α ∈ {1, 3, 5, 7}*</u>



Problem C.4: *Determine the size of the matrix s(t) generated in Problem C.3.*

The size of the matrix s(t) is 401x4 or 1604. This is found from the line matrix = zeros(401,4) which makes a 401 by 4 matrix of zeros.

Part D

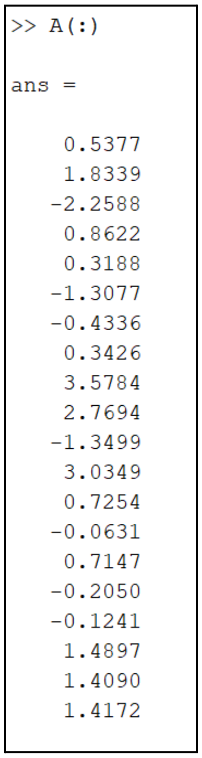
Note: The Matlab data file ELE532 Lab1 Data.mat contains all data arrays (arrays A, B and x audio) referenced in this section. You can download the data file from the course homepage on Blackboard. Alternatively, if you are using Matlab on any of departmental computers, typing load ELE532 Lab1 Data at the Matlab prompt will load all data arrays into your current Matlab workspace.

Problem D.1: *Let A be a 5 × 4 matrix array with real-valued elements:*

$$\mathbf{A} = \begin{bmatrix} 0.5377 & -1.3077 & -1.3499 & -0.2050 \\ 1.8339 & -0.4336 & 3.0349 & -0.1241 \\ -2.2588 & 0.3426 & 0.7254 & 1.4897 \\ 0.8622 & 3.5784 & -0.0631 & 1.4090 \\ 0.3188 & 2.7694 & 0.7147 & 1.4172 \end{bmatrix}$$

For the matrix A in Equation (1) implement the following operations:

(a) A(:)

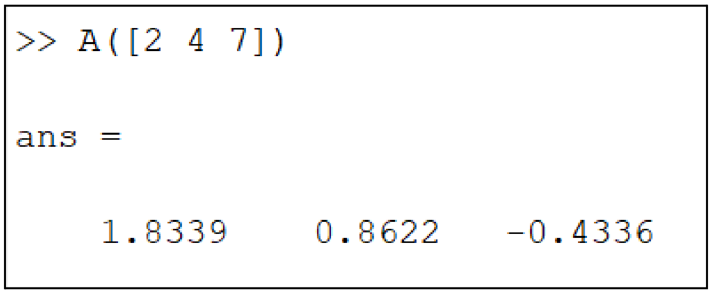


```
>> A(:)

ans =

    0.5377
    1.8339
   -2.2588
    0.8622
    0.3188
   -1.3077
   -0.4336
    0.3426
    3.5784
    2.7694
   -1.3499
    3.0349
    0.7254
   -0.0631
    0.7147
   -0.2050
   -0.1241
    1.4897
    1.4090
    1.4172
```

(b) A([2 4 7])

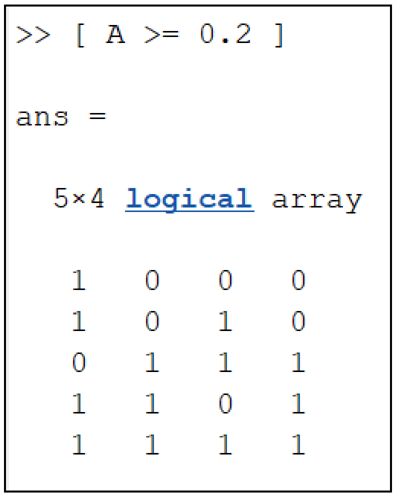


```
>> A([2 4 7])

ans =

    1.8339    0.8622   -0.4336
```

(c) [A >= 0.2]



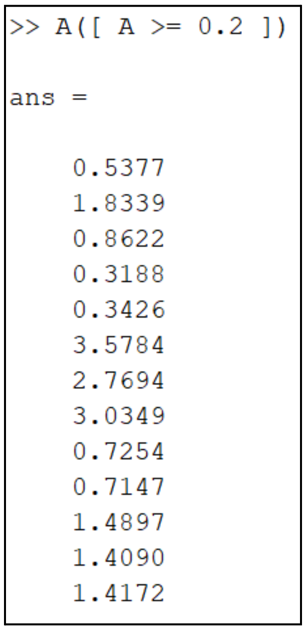
```
>> [ A >= 0.2 ]

ans =

  5×4 logical array

   1   0   0   0
   1   0   1   0
   0   1   1   1
   1   1   0   1
   1   1   1   1
```

(d) A([A >= 0.2])

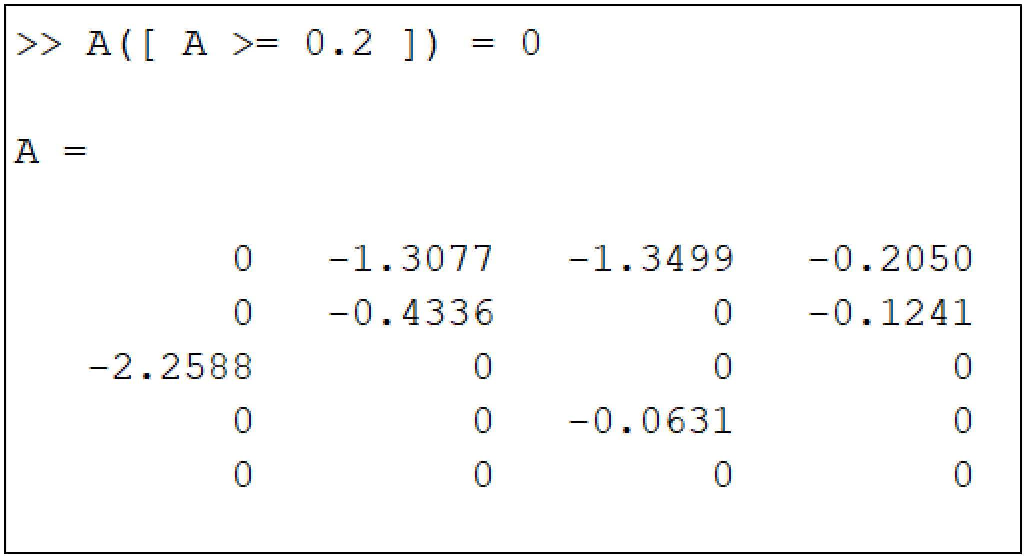


```
>> A([ A >= 0.2 ])

ans =

    0.5377
    1.8339
    0.8622
    0.3188
    0.3426
    3.5784
    2.7694
    3.0349
    0.7254
    0.7147
    1.4897
    1.4090
    1.4172
```

(e) A([A >= 0.2]) = 0



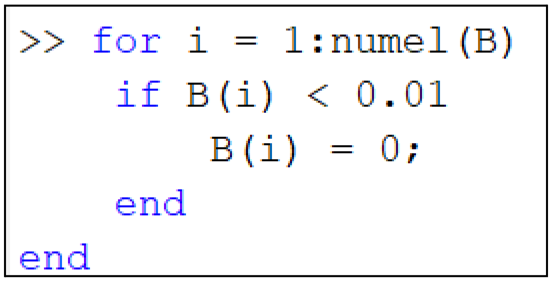
```
>> A([ A >= 0.2 ]) = 0

A =

         0   -1.3077   -1.3499   -0.2050
         0   -0.4336         0   -0.1241
   -2.2588         0         0         0
         0         0   -0.0631         0
         0         0         0         0
```

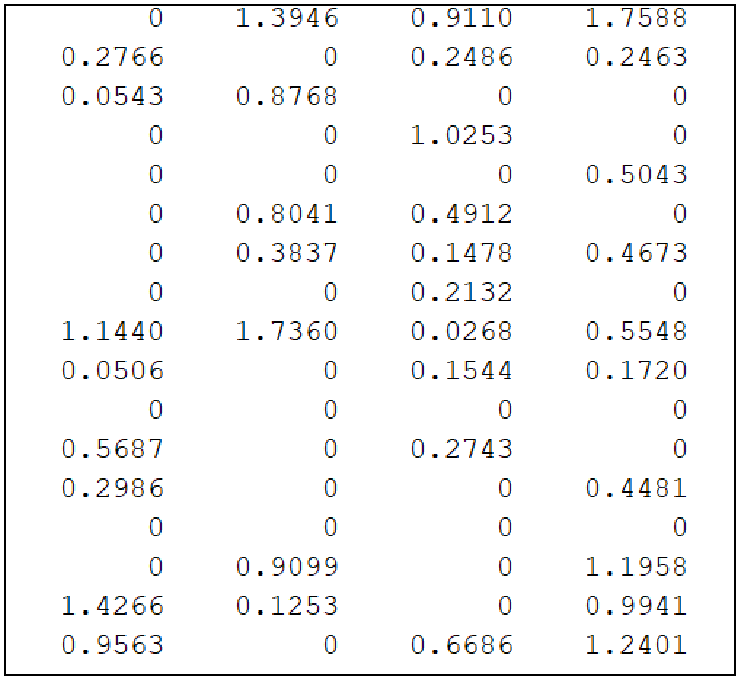
Problem D.2: *Let B be a 1024 × 100 data matrix representing 100 blocks of non-overlapping 1024-element input samples from a particular data source.*

(a) Write a simple Matlab program using two nested for loops that will set all elements of the data matrix B with magnitude values below 0.01 to zero: B(i, j) = 0, if |B(i, j)| < 0.01, (2) where B(i, j) is element of the data matrix B in i-th row and j-th column.



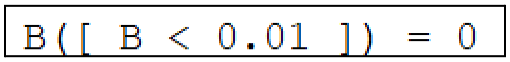
```
>> for i = 1:numel(B)
    if B(i) < 0.01
        B(i) = 0;
    end
end
```

Sample of B Matrix with element > 0.01 = 0 using the above-nested loop in Matlab



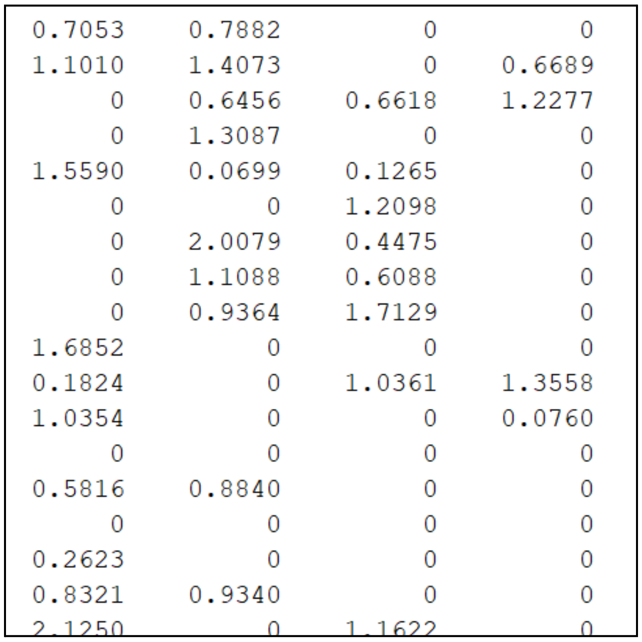
```
         0    1.3946    0.9110    1.7588
    0.2766         0    0.2486    0.2463
    0.0543    0.8768         0         0
         0         0    1.0253         0
         0         0         0    0.5043
         0    0.8041    0.4912         0
         0    0.3837    0.1478    0.4673
         0         0    0.2132         0
    1.1440    1.7360    0.0268    0.5548
    0.0506         0    0.1544    0.1720
         0         0         0         0
    0.5687         0    0.2743         0
    0.2986         0         0    0.4481
         0         0         0         0
         0    0.9099         0    1.1958
    1.4266    0.1253         0    0.9941
    0.9563         0    0.6686    1.2401
```

(b) Repeat part (a) using Matlab's indexing features as described in Problem D.1.



```
B([ B < 0.01 ]) = 0
```

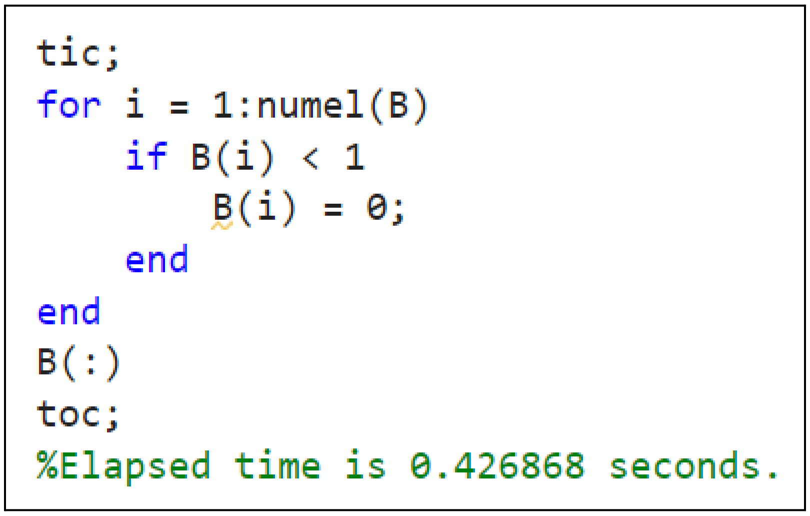
Sample of B Matrix with element > 0.01 = 0, using above indexing feature code



```
0.7053    0.7882         0         0
1.1010    1.4073         0    0.6689
     0    0.6456    0.6618    1.2277
     0    1.3087         0         0
1.5590    0.0699    0.1265         0
     0         0    1.2098         0
     0    2.0079    0.4475         0
     0    1.1088    0.6088         0
     0    0.9364    1.7129         0
1.6852         0         0         0
0.1824         0    1.0361    1.3558
1.0354         0         0    0.0760
     0         0         0         0
0.5816    0.8840         0         0
     0         0         0         0
0.2623         0         0         0
0.8321    0.9340         0         0
2.1250         0    1.1622         0
```

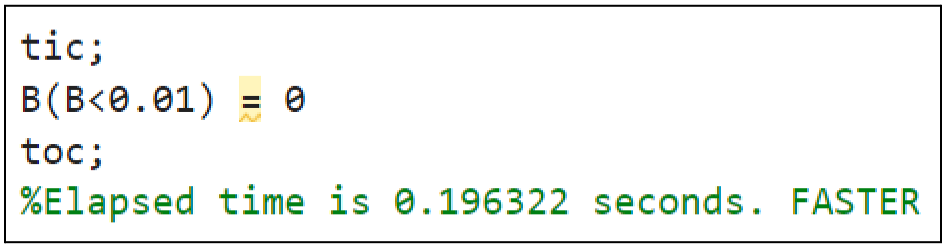
(c) Use the Matlab commands tic and toc to compare the execution time of the code you wrote in parts (a) and (b).

Here is the code for part a code:



```
tic;
for i = 1:numel(B)
    if B(i) < 1
        B(i) = 0;
    end
end
B(:)
toc;
%Elapsed time is 0.426868 seconds.
```

Here is the code for part b code:



```
tic;
B(B<0.01) = 0
toc;
%Elapsed time is 0.196322 seconds. FASTER
```

The execution time for part b is faster!

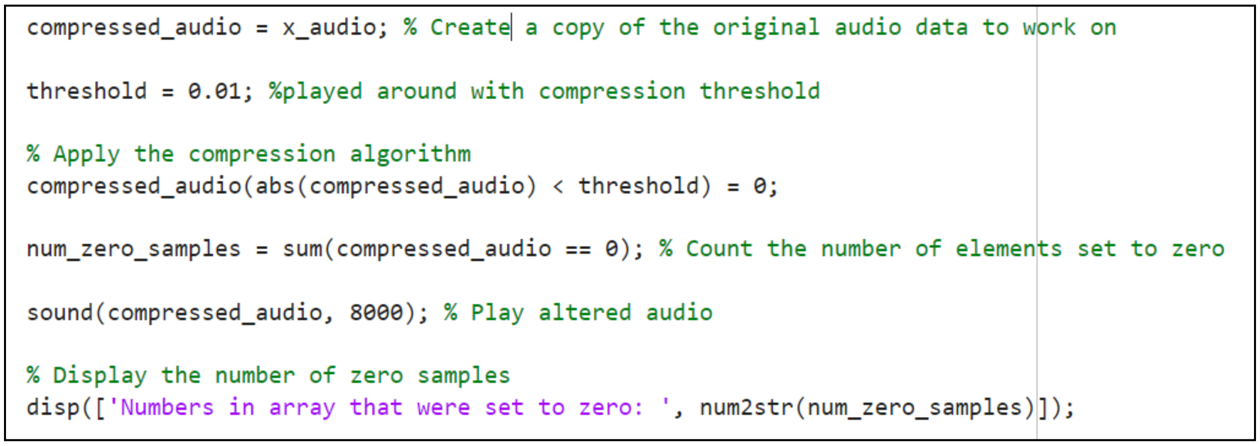
Problem D.3: *Let **x_audio** be a 20,000 sample-long row vector representing 2.5 sec of an audio signal sampled at 8 kHz. A simple data compression algorithm can be implemented by setting all elements of the data array **x_audio** with magnitude values below a threshold to zero.*

Note: *The actual compression algorithm will code the zero-valued samples more efficiently thus achieving a certain degree of compression. In this exercise, however, we* *only want to investigate the compressibility of the data array* ***x_audio*** *as measured by the number of samples with magnitude values below the threshold and the resulting sound quality as a function of the threshold.*

Write such a data compression algorithm and listen to the processed audio file. You may want to consider the following points:

- *Do not work directly on the data array* ***x_audio****; otherwise, after each process, you will need to reload the data file. Instead, copy the data array* ***x_audio*** *to another working array and process that array.*
- *Devise a simple method of counting the number of elements of the data array that are set to zero.*
- *You can listen to an audio array by using the Matlab command sound. For example, if you want to listen to the original, unprocessed data array* ***x_audio,*** *issue the command sound(****x_audio****,8000) at the Matlab prompt.*

My algorithm:



```
compressed_audio = x_audio; % Create a copy of the original audio data to work on

threshold = 0.01; %played around with compression threshold

% Apply the compression algorithm
compressed_audio(abs(compressed_audio) < threshold) = 0;

num_zero_samples = sum(compressed_audio == 0); % Count the number of elements set to zero

sound(compressed_audio, 8000); % Play altered audio

% Display the number of zero samples
disp(['Numbers in array that were set to zero: ', num2str(num_zero_samples)]);
```

What does the code do?

Because I changed the threshold number and tweaked it, now, the code allows for the audio which is played to be clear.