# Lab 1: Working with MATLAB, Visualization of Signals

By: Syed Yousuf and Khaled Hashem

# A.

**A.1:**

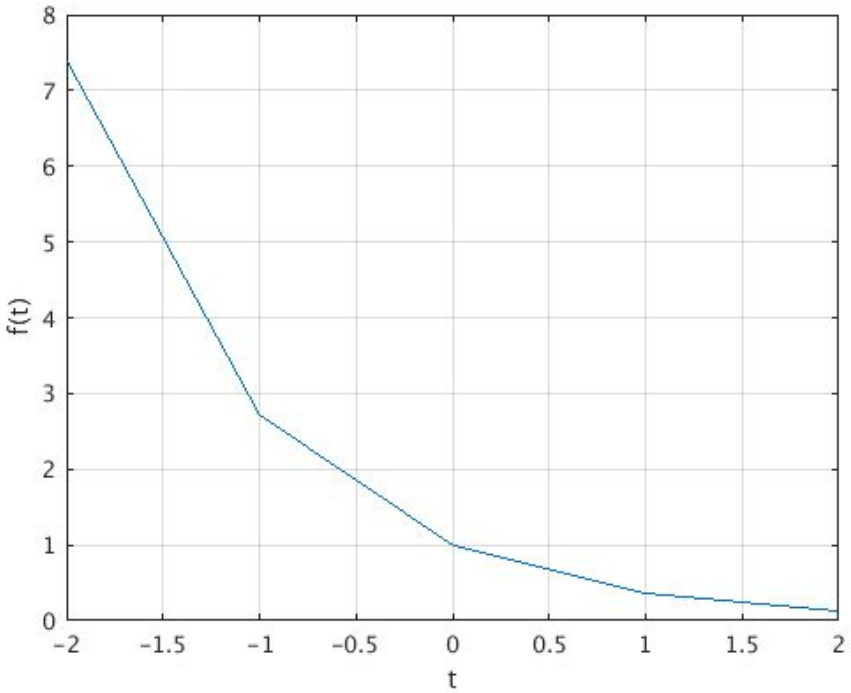
Mathlab code for generating and plotting graph 1.46:

---

```
% Set the value of t
t = (-2:2);

% Create function
f = @(t) exp(-t).*cos(2*pi*t);

plot(t,f(t));
xlabel('t');
ylabel('f(t)');
Grid;
```

---

**Graph 1.46**



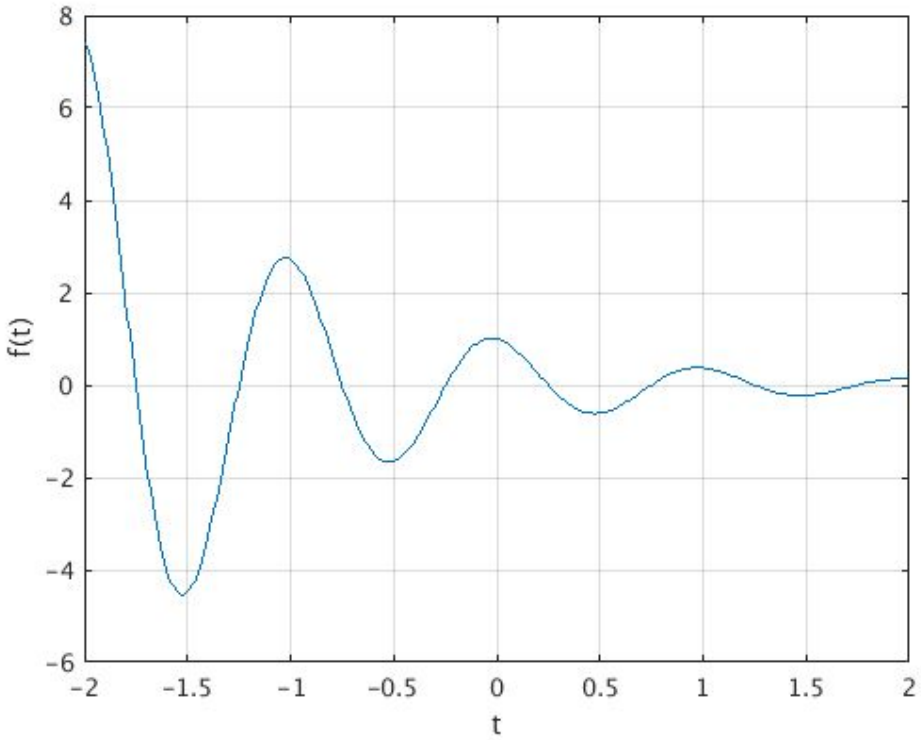
Mathlab code for generating and plotting graph 1.47:

---

```
% Set the value of t
t = (-2:0.01:2);

% Create function
f = @(t) exp(-t).*cos(2*pi*t);

plot(t,f(t));
xlabel('t');
ylabel('f(t)');
grid;
```

---

## Graph 1.47



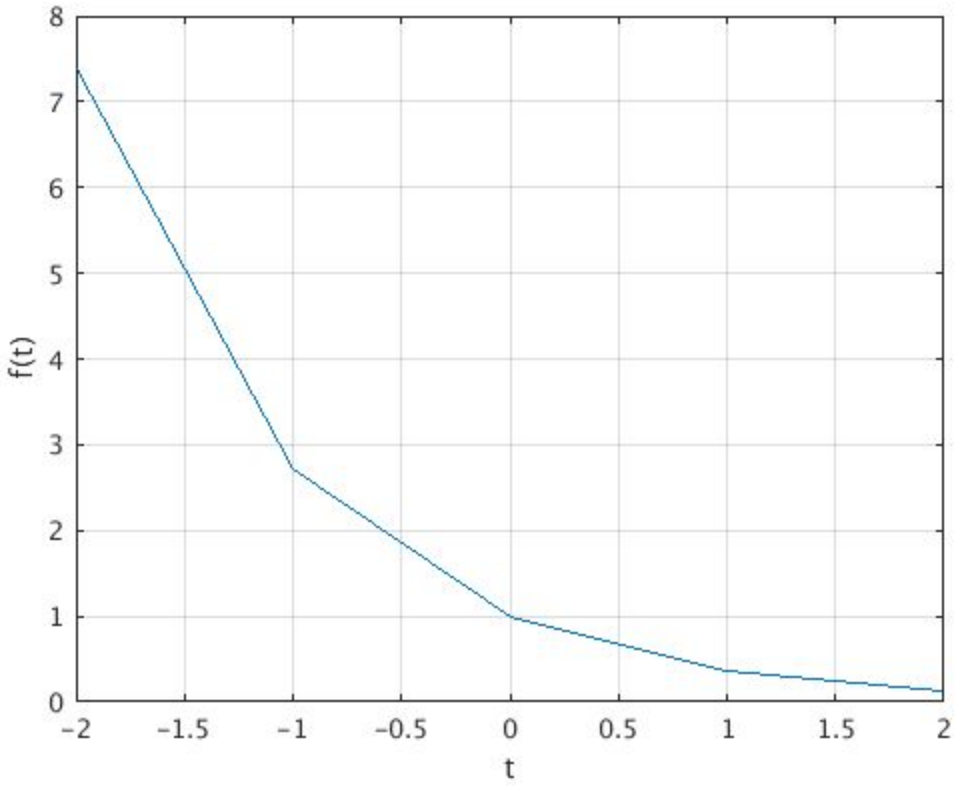
**A.2:**

Matlab code

---

```
% Set the value of t
t = (-2:2);

% Create function
f = @(t) exp(-t);

plot(t,f(t));
xlabel('t');
ylabel('f(t)');
grid;
```

---



**A3:**

When the graph for question A.2 with graph 1.46, we observe the following. We observe that both graphs are identical. Thus, f1 = @(t) exp(-t) produces the same graph as f2 = @(t) exp(-t).*cos(2*pi*t). Additionally, all the graphs decrease exponentially. The jagged edges we observe in graph1.46 and question A.2 is due to the large step size that was set for these two graphs.

# B.

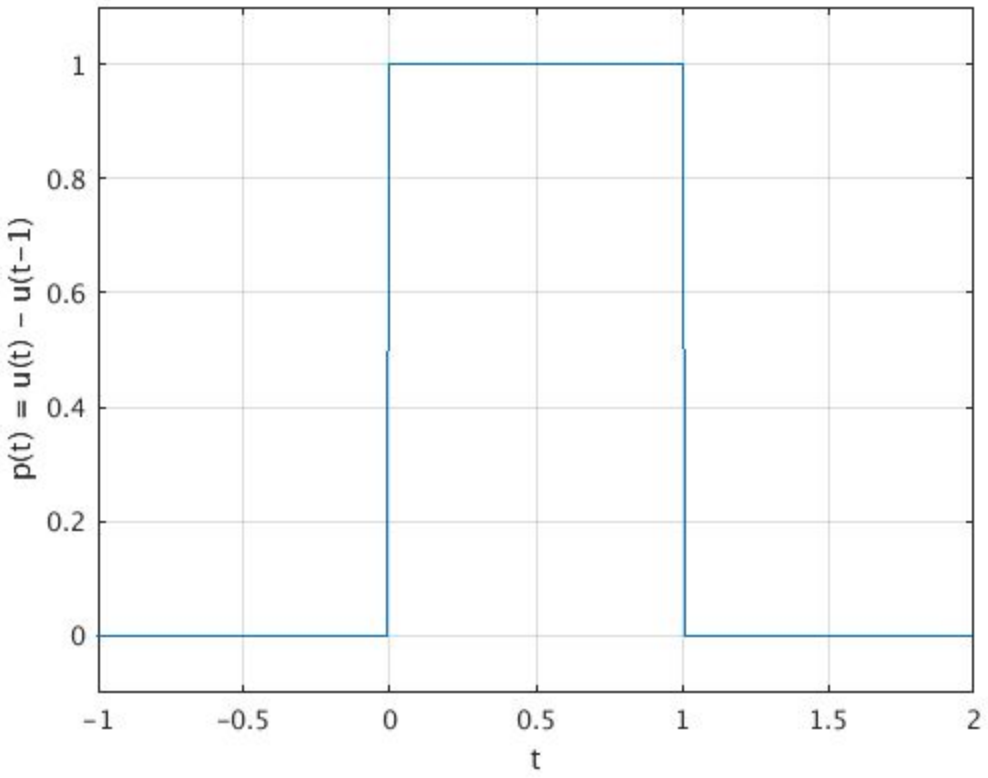
**B1.:**

Matlab code:

---

```
% Set the value of t
t = (-2:0.01:2);

% Create function
p = @(t) 1.0 * ((t >= 0) & (t <= 1));

plot(t,p(t));
xlabel('t');
ylabel('p(t) = u(t) - u(t-1)');
axis([-1 2 -0.1 1.1]);

grid;
```

---




**B.2:**

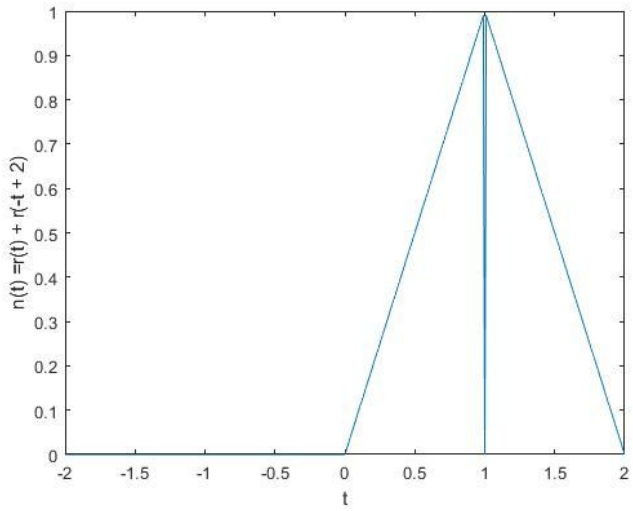
Matlab code for r(t)

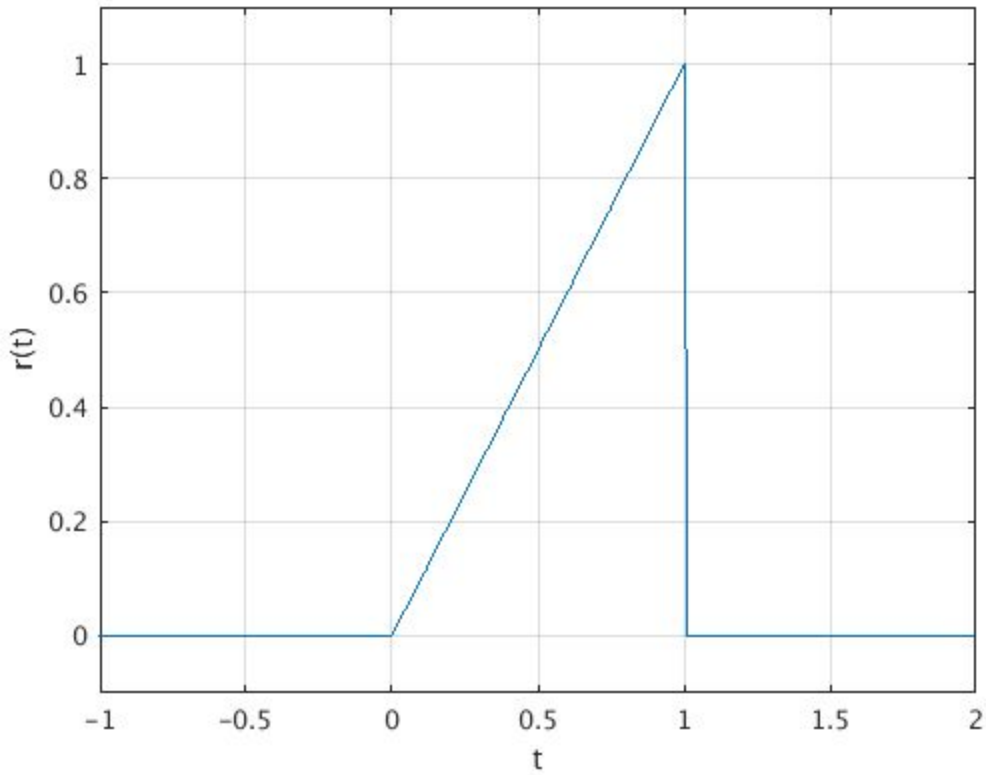
---

```
% Set the value of t
t = (-2:0.01:2);

% Create 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;
```

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

---



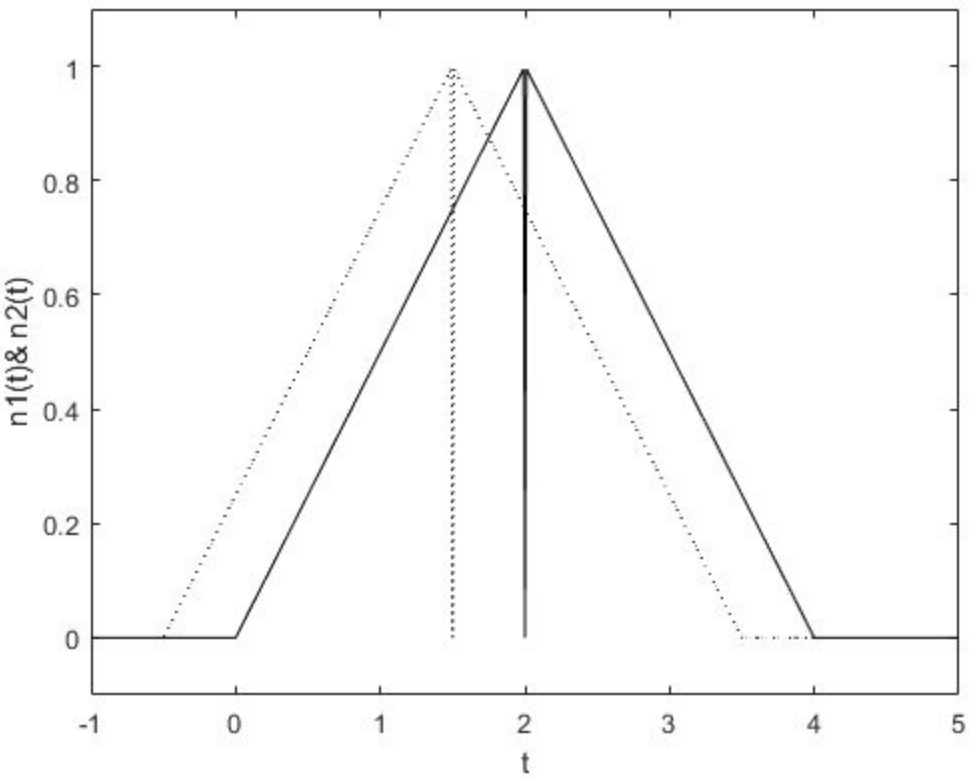



**B.3:**

Matlab code

```
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);

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');
xlabel('t'); ylabel('n3(t)& n4(t)');
axis([-1 5 -.1 1.1]);
grid;
```

---

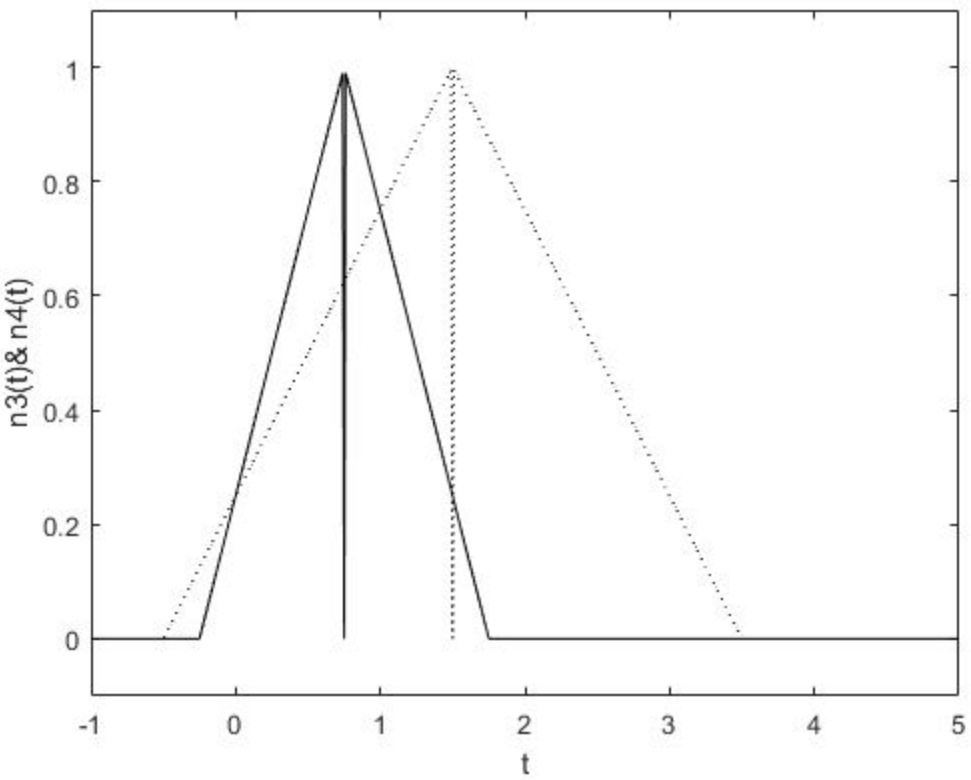


**B.4:**

Matlab code:

```
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');
xlabel('t'); ylabel('n1(t)& n2(t)');
axis([-1 5 -.1 1.1]);

grid;
```

---



**B.5:**

When we compare n4(t) and n2(t), we observe that both of them produce the exact same graph.

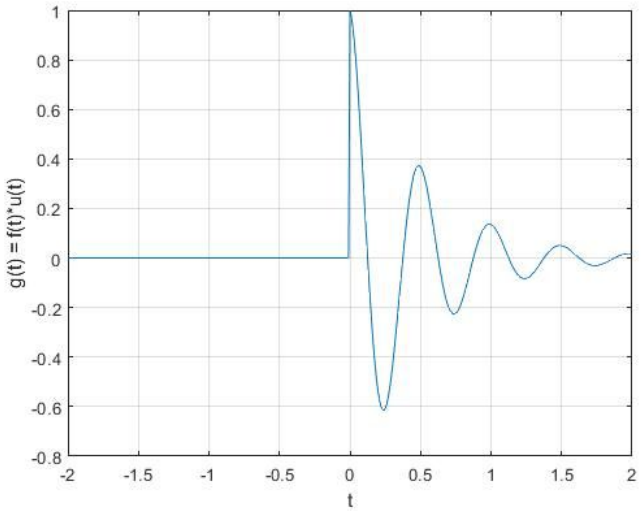
# C:

**C.1:**

Matlab:

```
% Create f(t) function
f = @(t) exp(-2.*t).*cos(4*pi*t);
t = (-2:0.01:2);

% Create u(t) function
u = @(t) 1.0.*(t>=0);
axis([-2 2 -0.1 1.1]);

% Create a g(t) function
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();
```

---

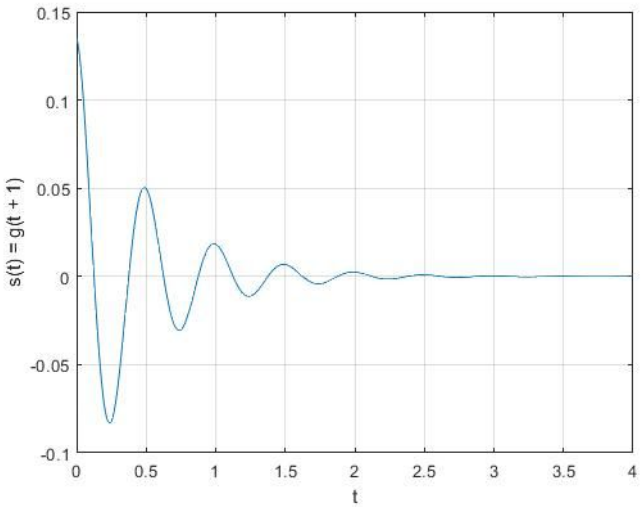


**C.2:**

Matlab:

```
% Create f(t) function
f = @(t) exp(-2.*t).*cos(4*pi*t);
t = (-2:0.01:2);

% Create u(t) function
u = @(t) 1.0.*(t>=0);
axis([-2 2 -0.1 1.1]);

% Create a g(t) function
g = @(t) f(t).*u(t);
t = (-2:0.01:2);

% Create the s(t) function
s = @(t) g(t + 1);
t = (0: 0.01: 4);

%plot
plot(t,s(t));
xlabel("t");
ylabel("s(t) = g(t + 1)");
grid();
```

---

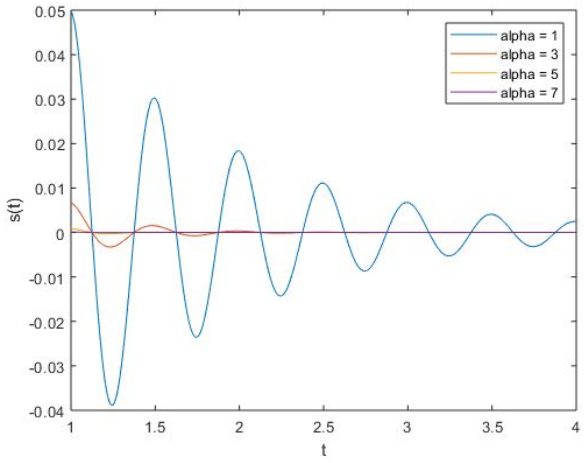



C.3:

Matlab:

```
% Create u(t) function
u = @(t) 1.0.*(t>=0);
t = (1 :0.01: 4);
matrix = zeros(401, 4);

for alpha = 1:2:7
    %Create each function.
    s = @(t) exp(-2).*exp(-alpha.*t).*cos(4*pi*t).*u(t);
    plot(t,s(t));
    xlabel("t");
    ylabel("s(t)");

    %Make sure to hold off plotting it on the figure.
    hold on;
end

%Create the legend
legend('alpha = 1', 'alpha = 3', 'alpha = 5', 'alpha = 7');
%Plot all the functions on one figure.
hold off;
```

---




**C.4:**

The size of the array generated in C.3 for s(t) is 1604.

# D

## D.1

a) Operation lists the components of the matrix array vertically, starting from the leftmost column.
b) Elements at row 2 (column 1), row 4 (column 1) , and at row 2 (column 2), which is technically the 7th element, are all listed when this operation is performed.
c) Creates a logical matrix array (5 x 4) , that has values 0 and 1, the position in which the matrix has 0 refers to the part of the original matrix array A where the value of the element is not greater than or equal to 0.2, and the opposite holds if this generated logical matrix array holds 1 at parts of its indexes.
d) This operation , on the other hand, lists the elements that are greater than or equal to 0.2 in array A.
e) This operation places zeros at the indexes of A where the elements are greater than or equal to 0.2.

## D.2

a)

```
load('ELE532_Lab1_Data.mat')
num_rows = size(B,1);
num_cols = size(B,2);

for i=1:1:num_rows
      for j=1:1:num_cols
            if (abs(B(i,j)) < 0.01)
                        B(i,j)=0;
            end
      end
end
```

b)

```
load('ELE532_Lab1_Data.mat')
B([ abs(B) >= 0.01]) = 0
```

c) i)

```
tic
load('ELE532_Lab1_Data.mat')
num_rows = size(B,1);
num_cols = size(B,2);

for i=1:1:num_rows
      for j=1:1:num_cols
            if (abs(B(i,j)) < 0.01)
                  B(i,j)=0;
            end
      end
end

fprintf('\nD2:\nFor part a:\n')
toc
```

ii)

```
tic
load('ELE532_Lab1_Data.mat');
B([ abs(B) >= 0.01]) = 0

fprintf('\nD2: \nFor part b: Elapsed time is :\n ');
toc

D.3:
% Load the data.
load('ELE532_Lab1_Data.mat');

% Copy the data array x_audio into audio.
audio = x_audio;

num_rows = size(audio,1);
num_cols = size(audio, 2);

number_of_zeros = 0;

for i = 1: num_rows
      for j = 1: num_cols
      if(abs(audio(i,j) == 0))
            number_of_zeros = number_of_zeros + 1;
```

```
      end
      end
end

% How many elements are zero in the
fprintf("\n" + number_of_zeros);

% Now play the sound.
sound(audio,8000)
```