Table of Contents


```
% A.1 Figure 1.46
t = (-2:2);
f = @(t) exp(-t).*cos(2*pi*t);
plot(t,f(t));
grid;
title('Figure 1.46');
xlabel('t');
ylabel('f(t)');
figure;

% A.1 Figure 1.47
t2 = (-2:0.01:2);
f2 = @(t2) exp(-t2).*cos(2*pi*t2);
plot(t2,f2(t2));
grid;
title('Figure 1.47');
xlabel('t2');
ylabel('f2(t2)');
```

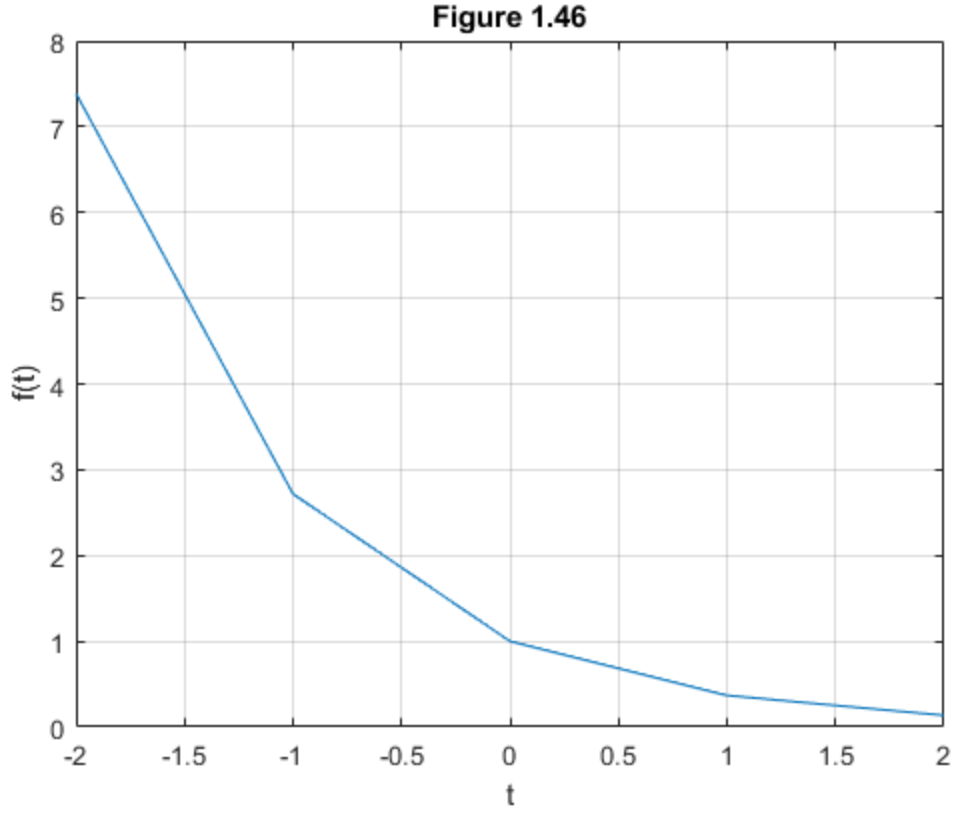


Figure 1.46
f(t)
t
8
7
6
5
4
3
2
1
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

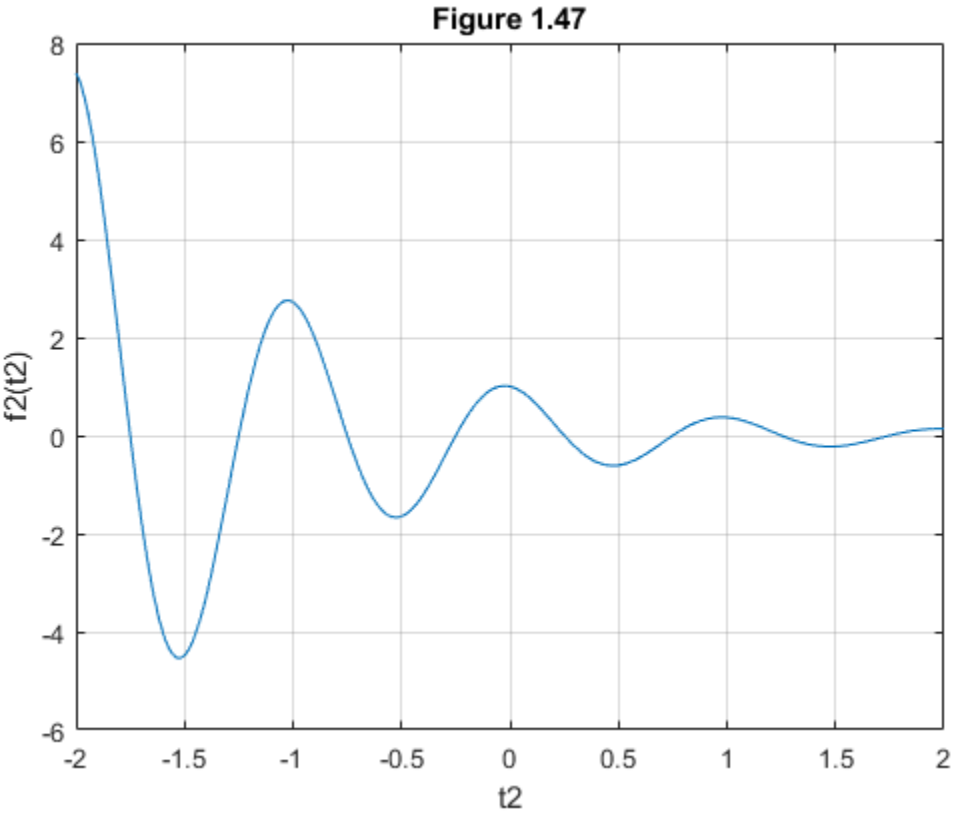
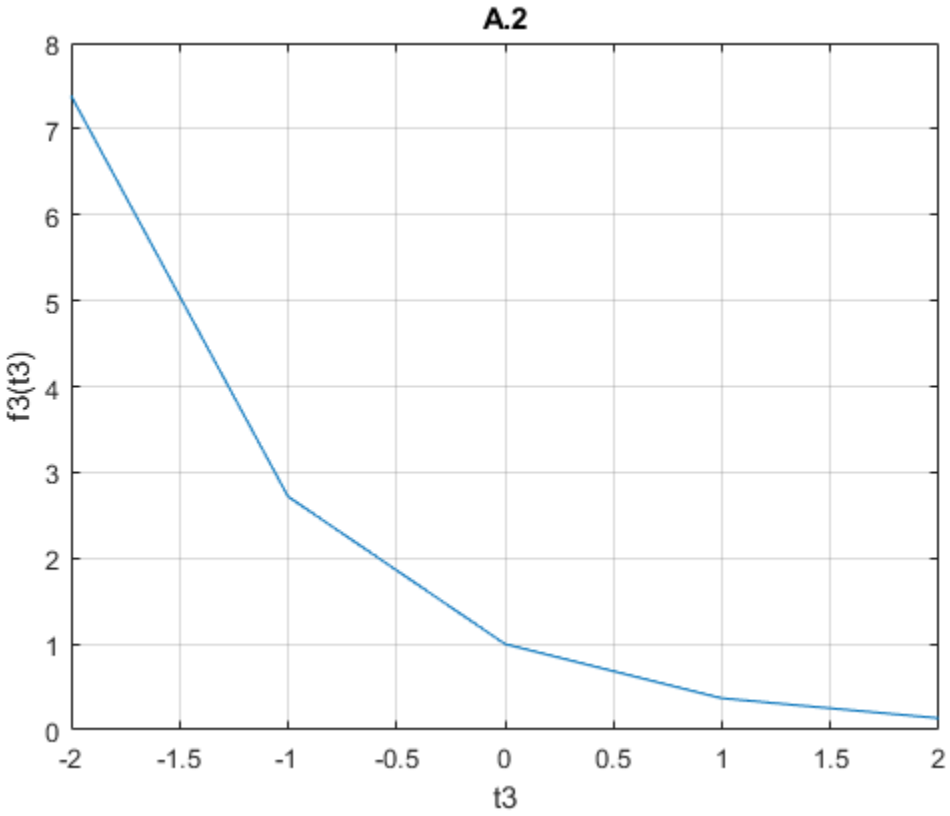


Figure 1.47
f2(t2)
t2
8
6
4
2
0
-2
-4
-6
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

A.2

```
t3 = (-2:1:2);
f3 = @(t3) exp(-t3);
plot(t3,f3(t3));
grid;
title('A.2');
xlabel('t3');
ylabel('f3(t3)');
```



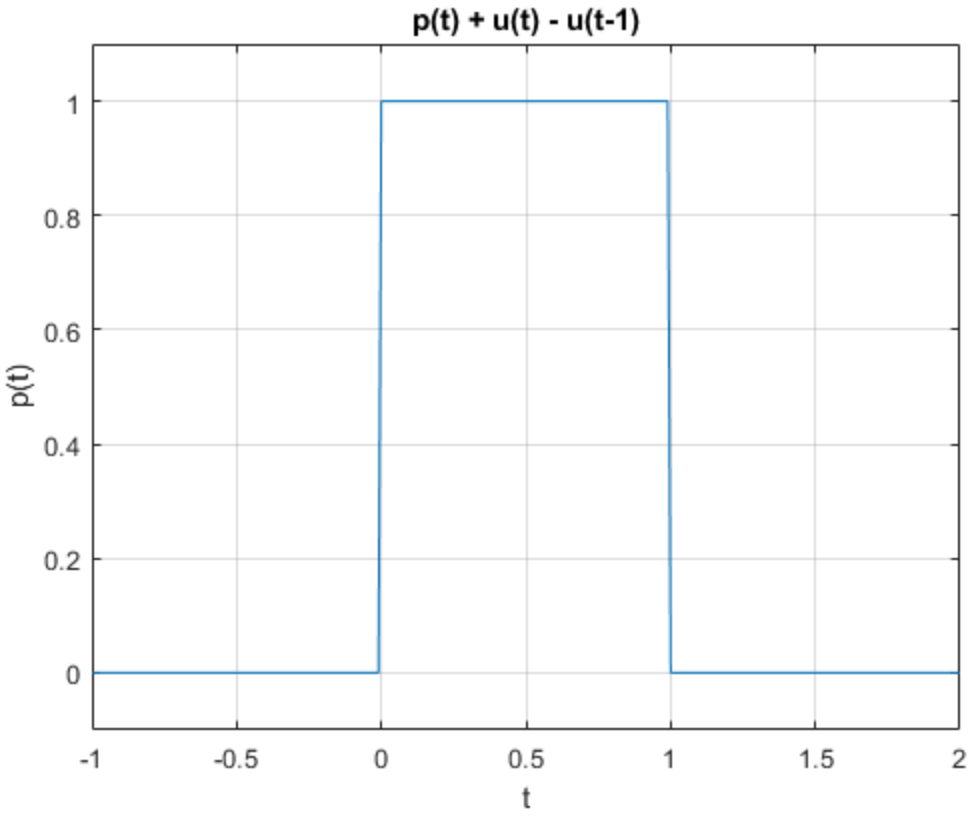
A.3

Comparing A.2 with Figure 1.46, they look identical despite being completely different functions. This is because in A.2, the interval is so large that the accuracy is lost which results in a function looking like Figure 1.46

B.1

```
t4 = (-2:0.01:2);
p = @(t4) 1.0 .* ((t4>=0) & (t4<1));
plot(t4, p(t4));
grid;
axis([-1 2 -.1 1.1]);
title('p(t) + u(t) - u(t-1)');
xlabel('t');
```

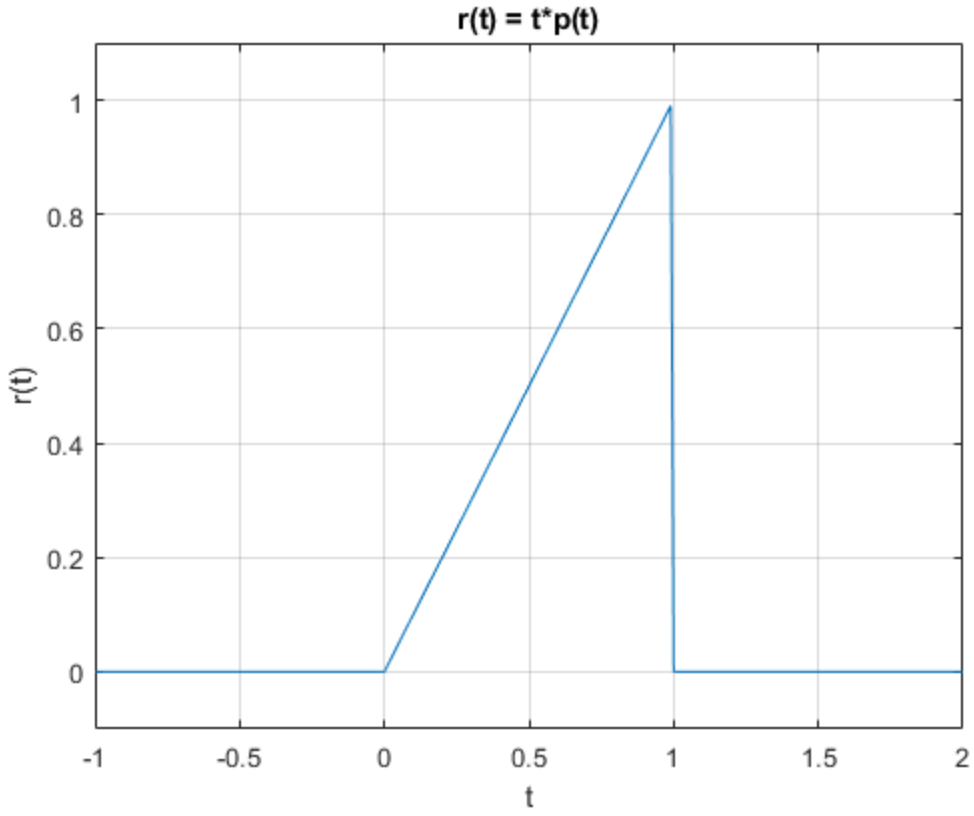
```
ylabel('p(t)');
```

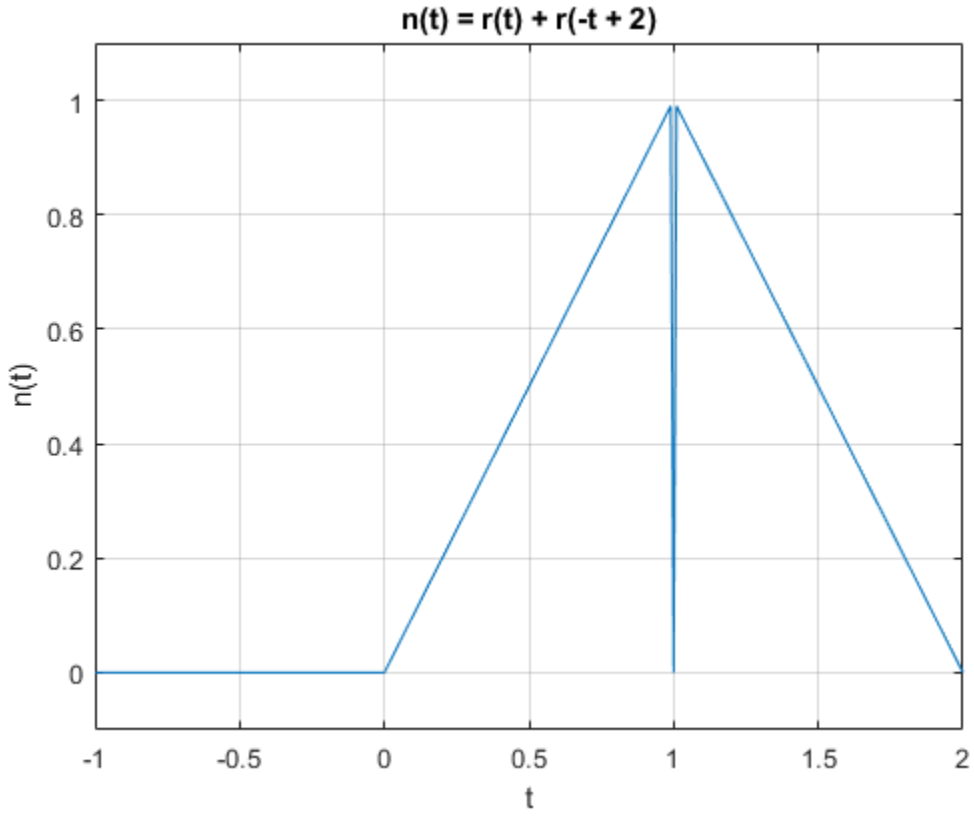


B.2

r(t)

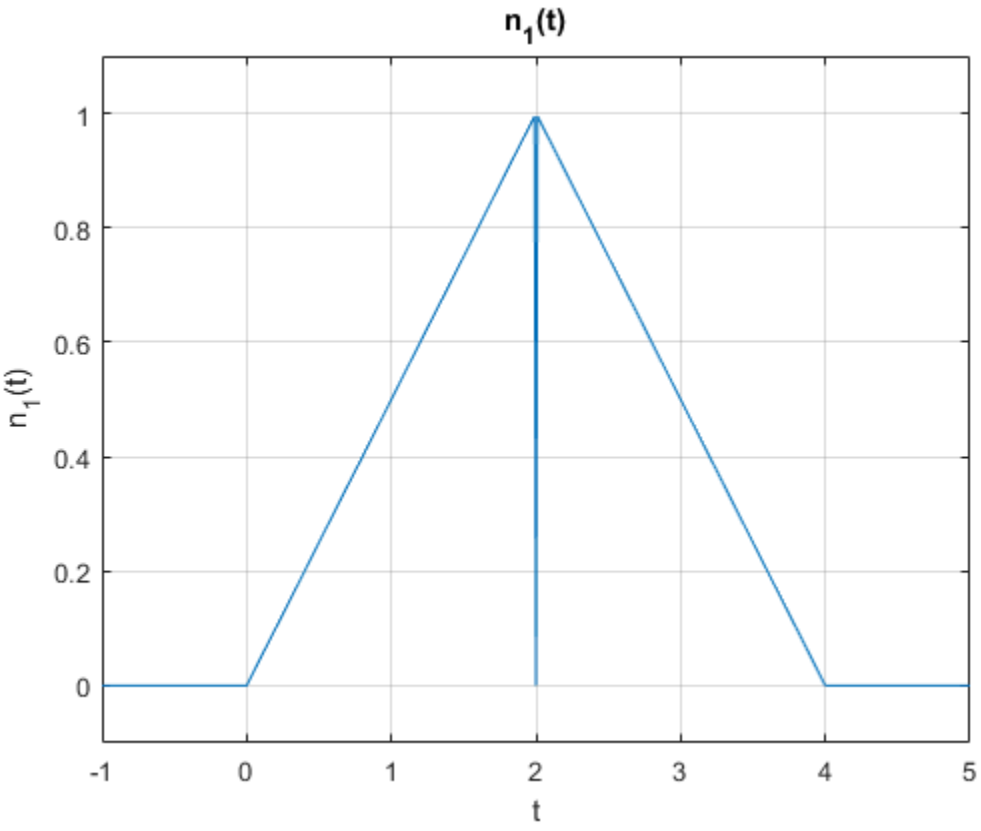
```
r = @(t4) (t4 .* p(t4));
plot(t4, r(t4));
grid;
axis([-1 2 -.1 1.1]);
title('r(t) = t*p(t)');
xlabel('t');
ylabel('r(t)');
figure;

% n(t)
n = @(t4) (r(t4) + r(-t4 + 2));
plot(t4, n(t4));
grid;
axis([-1 2 -.1 1.1]);
title('n(t) = r(t) + r(-t + 2)');
xlabel('t');
ylabel('n(t)');
```

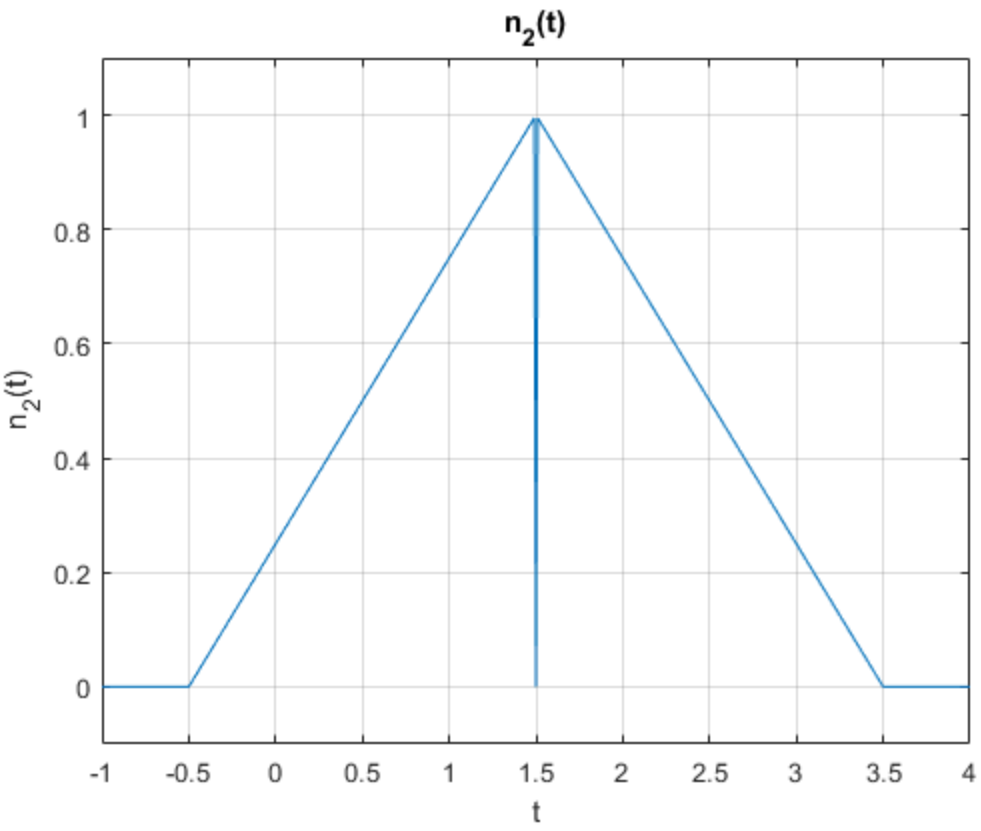

r(t) = t*p(t)
r(t)
t
1
0.8
0.6
0.4
0.2
0
-1
-0.5
0
0.5
1
1.5
2


n(t) = r(t) + r(-t + 2)
n(t)
t
1
0.8
0.6
0.4
0.2
0
-1
-0.5
0
0.5
1
1.5
2

B.3

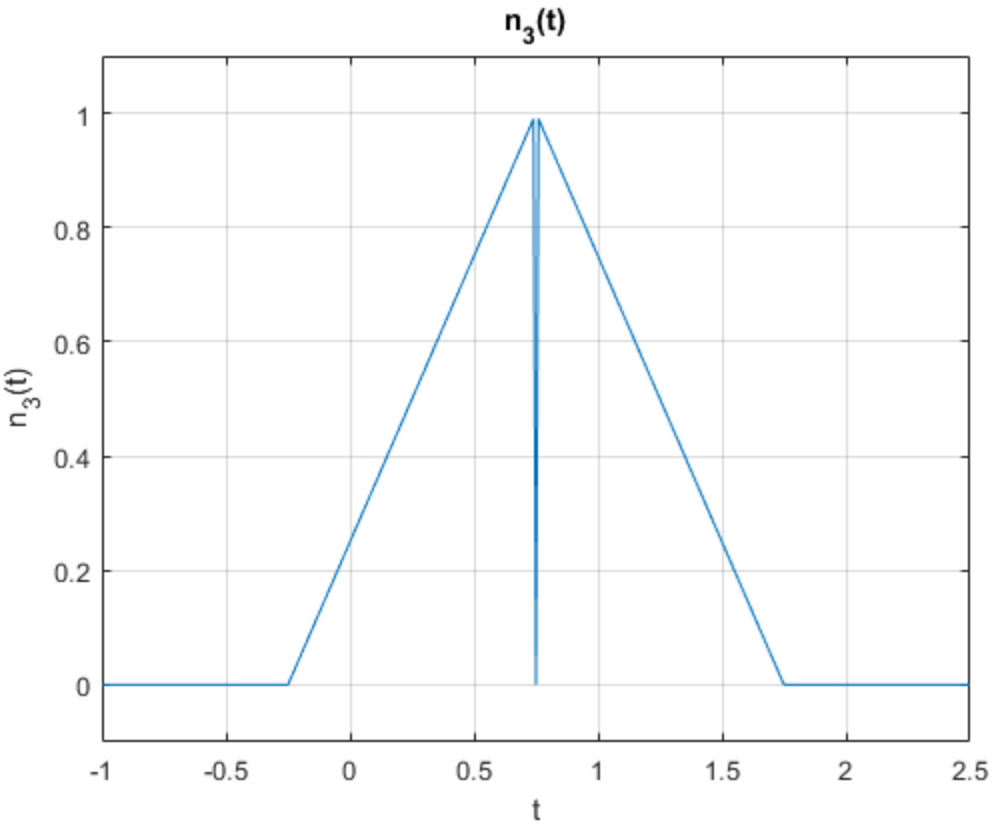
```
t5 = [-5:0.01:5];
n1 = @(t5) (n(0.5 .* t5));
plot(t5, n1(t5));
grid;
title("n_{1}(t)");
xlabel('t');
ylabel('n_{1}(t)');
axis([-1 5 -0.1 1.1]);
figure;
n2 = @(t5) (n1(t5 + 0.5));
plot(t5, n2(t5));
grid;
title("n_{2}(t)");
xlabel('t');
ylabel('n_{2}(t)');
axis([-1 4 -0.1 1.1]);
```

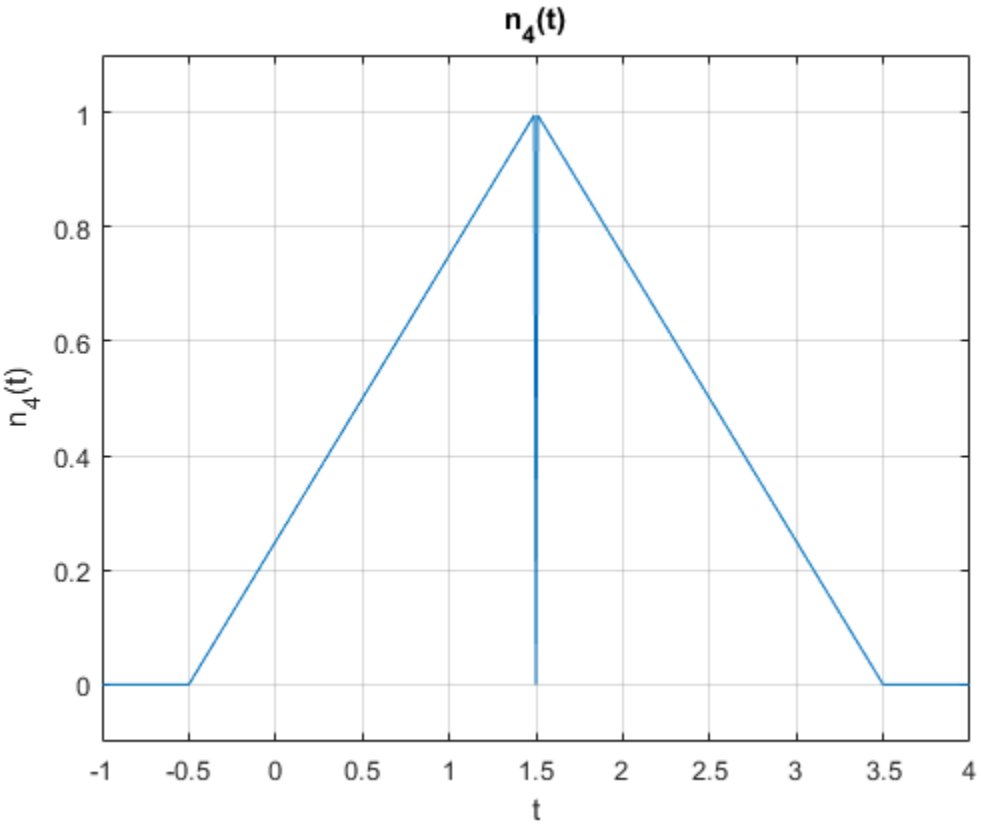




B.4

```
n3 = @(t5) (n(t5 + (1 ./ 4)));
plot(t5, n3(t5));
grid;
title("n_{3}(t)");
xlabel('t');
ylabel('n_{3}(t)');
axis([-1 2.5 -0.1 1.1]);
figure;
n4 = @(t5) (n3(0.5 .* t5));
plot(t5, n4(t5));
grid;
title("n_{4}(t)");
xlabel('t');
ylabel('n_{4}(t)');
axis([-1 4 -0.1 1.1]);
```


n3(t)
1
0.8
0.6
0.4
0.2
0
n3(t)
-1
-0.5
0
0.5
1
1.5
2
2.5
t

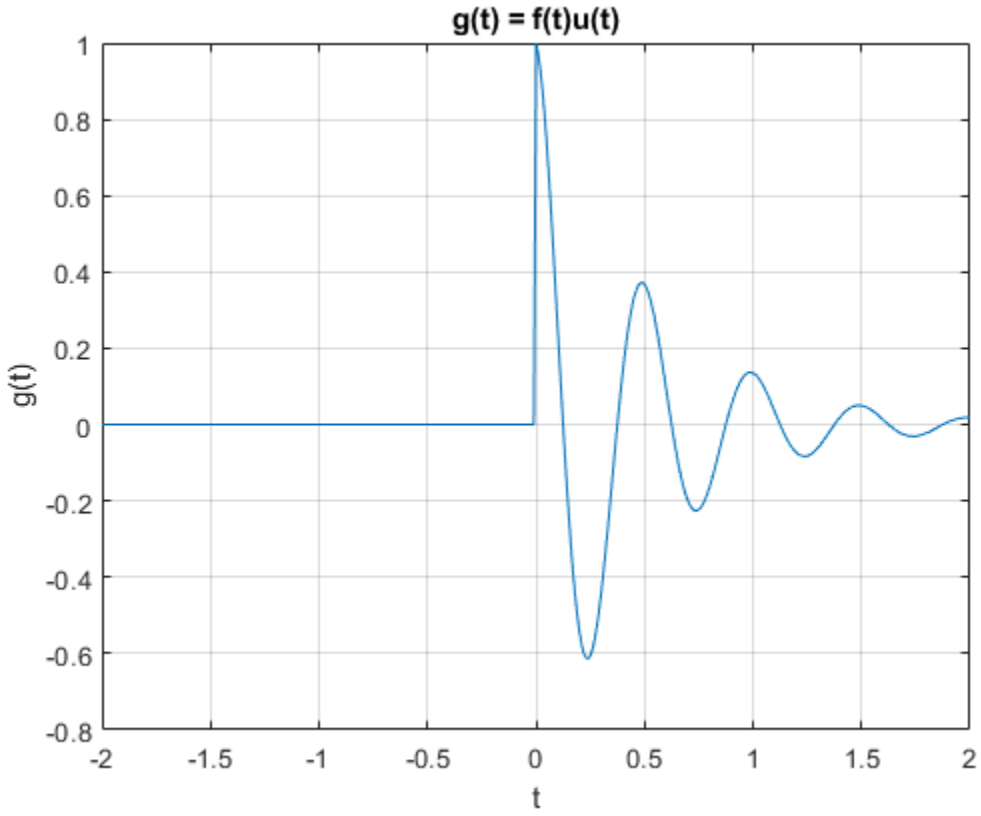

n4(t)
1
0.8
0.6
0.4
0.2
0
n4(t)
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
t

B.5

both n4(t) and n2(t) are identical as seen in the figures above and this is because both the functions are identical. This can be proven by simply expanding and simplifying both the equations

C.1

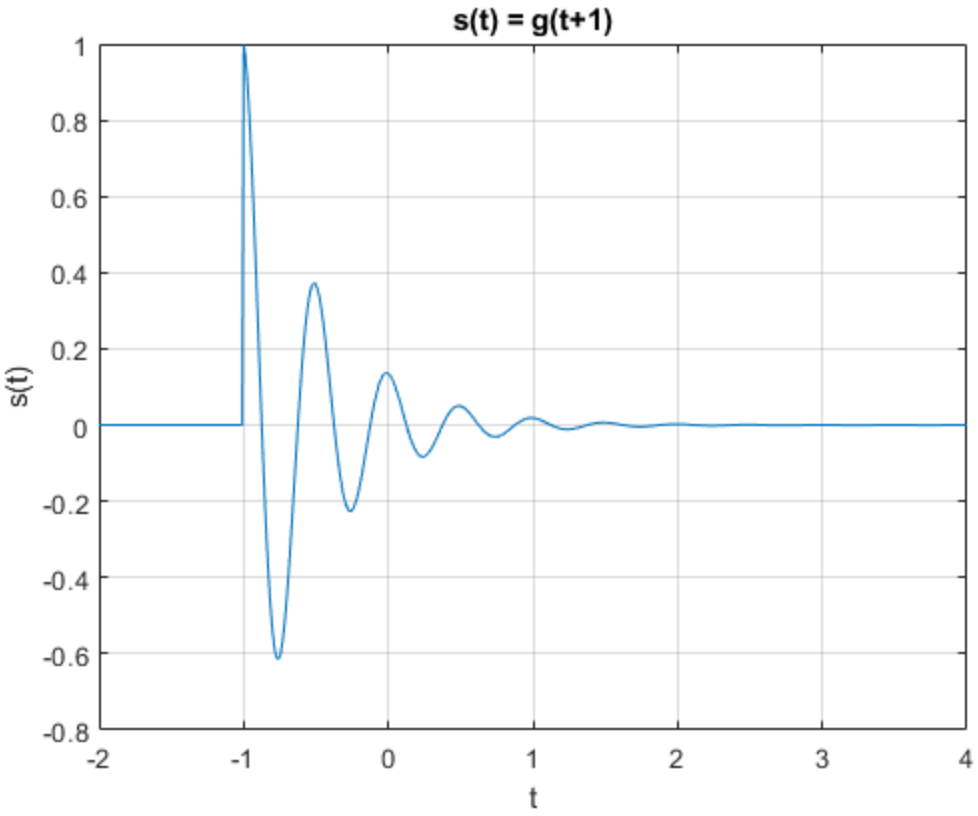
```
t6 = [-2:0.01:2];
u = @(t6) 1.0.*(t6>=0);
f = @(t6) (exp(-2 .* t6) .* cos(4 .* pi .* t6));
g = @(t6) f(t6) .* u(t6);
plot(t6, g(t6));
grid;
title("g(t) = f(t)u(t)");
xlabel('t');
ylabel('g(t)');
```



C.2

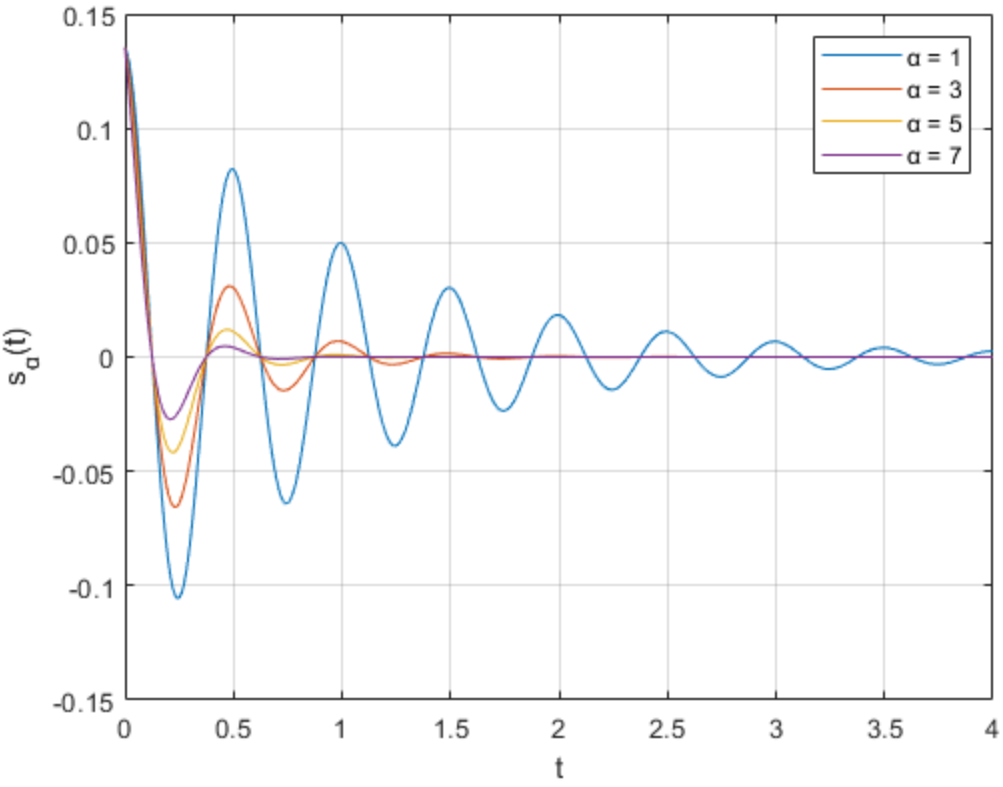
```
t7 = [-2:0.01:4];
s = @(t7) g(t7 + 1);
plot(t7, s(t7));
grid;
title("s(t) = g(t+1)");
```

```
xlabel('t');
ylabel('s(t)');
```



C.3

```
u = @(t8) 1.0 .* (t8 >= 0);
t8 = (0:0.01:4);
array_size = 0;

for a = (1:2:7)
    s = @(t8) exp(-2) .* exp(-a .* t8).*cos(4*pi*t8).*u(t8);
    plot(t8, s(t8));
    xlabel('t');
    ylabel('s_{#}(t)');
    hold on;
    array_size = array_size + size(t8);
end

grid;
legend('# = 1', '# = 3', '# = 5', '# = 7');
hold off;
```



C.4

```
array_size
% The size of the array is 1604 (4x401)
```

```
array_size =

           4        1604
```

D.1

```
load('ELE532_Lab1_Data.mat');

% a)
% all elements in the matrix array column by column
A(:)

% b)
% elements at indexes 2, 4, 7 going column by column
A([2 4 7])

% c)
% shows 1 if element in the matrix is greater than or equal to 0.2
```

```
% and shows 0 if it is not
[A >= 0.2]

% d)
% lists the element values that are greater than or equal to 0.2
% going column by column
A([A >= 0.2])

% e)
% shows matrix A but with all elements that are greater than or
% equal to 0.2 replaced with 0.
A([A >= 0.2]) = 0
```

```
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


ans =

    1.8339    0.8622   -0.4336


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
```

```
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


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
```

D.2 a)

```
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
```

D.2 b)

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

D.2 c) P1

```
tic
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
toc
```

Elapsed time is 0.002013 seconds.

D.2 c) P2

```
tic
B([abs(B) <= 0.01]) = 0;
toc

% Therefore the MATLAB indexing is faster than the nested for loop
```

Elapsed time is 0.000491 seconds.

D.3

```
load('ELE532_Lab1_Data.mat');

audio_file = x_audio;

audio_file((audio_file) <= 0) = 0;

num_rows = size(audio_file,1);
num_cols = size(audio_file,2);
num_zeros = 0;

for i = 1:num_rows
    for j = 1:num_cols
        if((audio_file(i, j)) == 0)
            num_zeros = num_zeros + 1;
        end
    end
end

num_zeros
sound(x_audio,8000)
pause(3)
sound(audio_file,8000)

% The processed sound appears to be more muffled than the original sound
```

num_zeros =

10073

Published with MATLAB® R2023a