Ryerson University

Faculty of Engineering and Architectural Science

Department of Electrical Engineering

Lab 1

Shane Sigesmund 500817399, Abrar Alam 500725366

Professor: Dr. Soosan Beheshti

TA: Salar Razavi

Date: 23rd September 2020

Contents

List of Figures

Chapter 1

Part A

1.1 A.1

```
%Part A.1a
u = @(t) 1.0.*(t>=0);

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

t = (-2:2);

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


%Part A.1b

t = (-2:0.01:2);

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

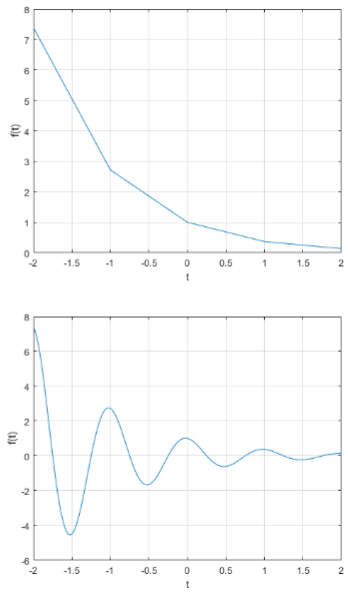

Figure 1.1: Part A.1a & A.1b graphs

1.2 A.2

```
%Part A.2
t = (-2:2);
plot(t,f(t)); xlabel('t'); ylabel('f(t) = exp(-t)'); grid;
```

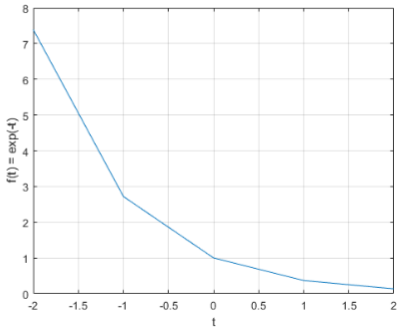

Figure 1.2: A.2 graph

1.3 A.3

The comparison between the two graphs fig 1.1(first graph), and fig 1.2 are strikingly similar as both of them follow a graph of a common exponential to the power of negative t.

Chapter 2

Part B

2.1 B.1

```
p = @(t) 1.0.*((t>=0)&(t<1));
t = (-1:0.01:2); plot(t,p(t));

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

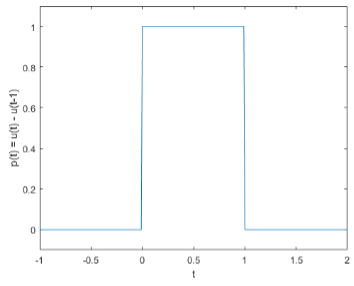

Figure 2.1: B.1 graph

2.2 B.2

```
p = @(t) 1.0.*((t>=0)&(t<1));

r = @(t) t.*p(t);

n = @(t) (r(t) + r(-t + 2));

t = (-1:0.01:2); plot(t,r(t));

%Part 1
xlabel('t'); ylabel('r(t) = t*p(t)');
axis([-1 2 -0.1 1.1]);

t = (-1:0.01:2); plot(t,n(t));

%Part 2
xlabel('t'); ylabel('n(t) = r(t) + r(-t + 2)');
axis([-1 2 -0.1 1.1]);
```

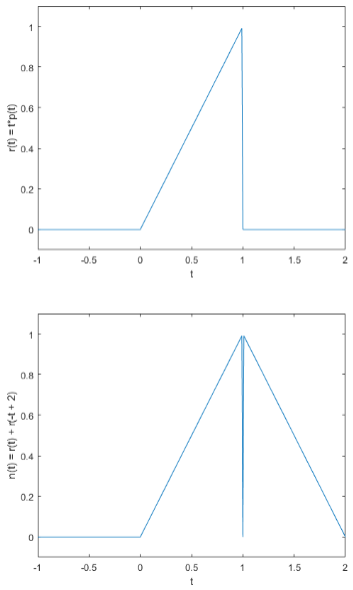

Figure 2.2: B.2 Part A graph

2.3 B.3

```
p = @(t) 1.0.*((t>=0)&(t<1));

r = @(t) t.*p(t);

n = @(t) (r(t) + r(-t + 2));

n1 = @(t) n((1/2)*t);

n2 = @(t) n1(t + 1/2);


%Comment out one of the plots to see the other one

%Part 1
t = (-1:0.01:2); plot(t,n1(t));
xlabel('t'); ylabel('n1(t) = n((1/2)*t)');
axis([-1 2 -0.1 1.1]);

%Part 2
t = (-1:0.01:2); plot(t,n2(t));
xlabel('t'); ylabel('n2(t) = n1(t + 1/2)');
axis([-1 2 -0.1 1.1]);
```

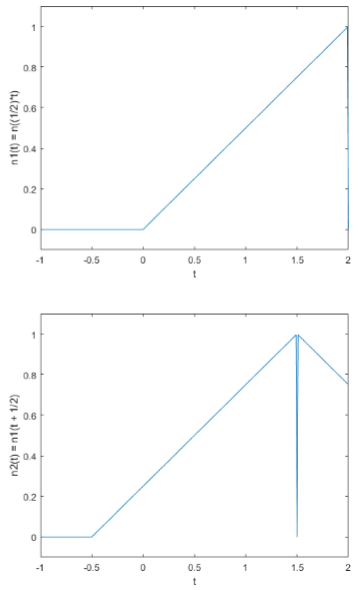

Figure 2.3: B.3 Part A and B graph

2.4 B.4

```
p = @(t) 1.0.*((t>=0)&(t<1));

r = @(t) t.*p(t);

n = @(t) (r(t) + r(-t + 2));

n3 = @(t) n(t + 1/4);

n4 = @(t) n3(t.*(1/2));

%Comment out one of the plots to see the other one

%Part 1
t = (-1:0.01:2); plot(t,n3(t));
xlabel('t'); ylabel('n3(t) = n(t + 1/4)');
axis([-1 2 -0.1 1.1]);

%Part 2
t = (-1:0.01:2); plot(t,n4(t));
xlabel('t'); ylabel('n4(t) = n3(t * 1/2)');
axis([-1 2 -0.1 1.1]);
```

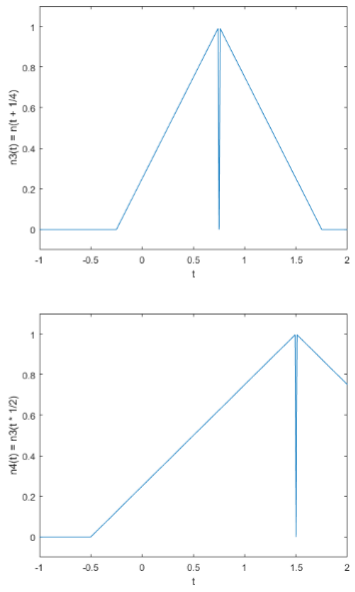

Figure 2.4: B.4 Part A and B graph

2.5 B.5

As seen in section A.3, there seems to be a considerable similarity between the two graphs. Both of the graphs look exactly the same, however, the graph for n_4(t) has the function $(t * \frac{1}{2})$ while n_2(t) has the function $(t + \frac{1}{2})$.

Chapter 3

Part C

3.1 C.1

The signal $g(t) = f(t)u(t)$ is generated by following the instructions in the textbook in page 130. However, in this case we chose $f(t) = e^{\neg 2t}\cos 4\pi t$

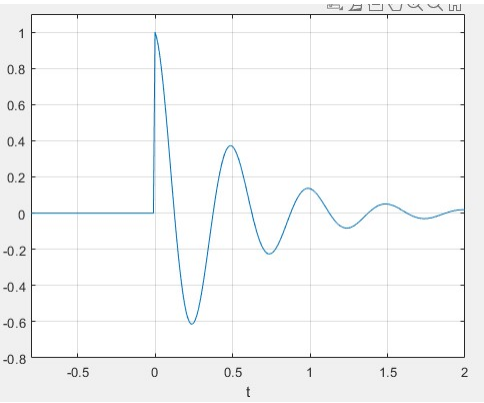

Figure 3.1: C.1 graph

3.2 C.2

Let's now generate $s(t) = g(t+1)$ for $t = [0 : 0.01 : 4]$. Please refer to figure 3.2 and 3.3 for the code and graph respectively.

```
t = (0 : 0.01 : 4);
%unit step signal
u = @(t) 1.0 .* (t >= 0);
%p = @(t) u(t) - u(t-1);
%plot(t, p(t));
%Our exp. periodic original signal below
f = @(t) exp(-2*t).* cos(4*pi*t);
% below is our final causal signal
g = @(t) f(t) .* u(t);
% Our shifted version
s = @(t) g(t+1);
plot(t, s(t));
xlabel('t'); ylabel('s(t) = g(t+1)'); grid;
clear all;
```

Figure 3.2: C.2 code

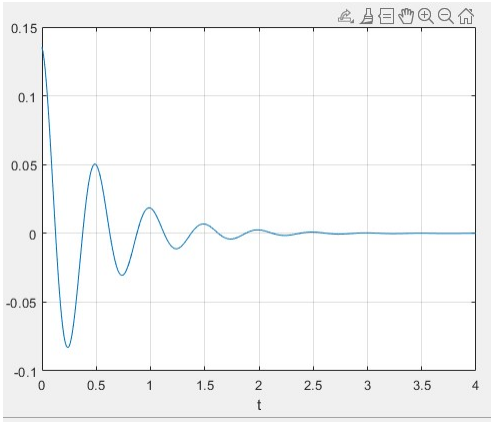

Figure 3.3: C.2 graph

3.3 C.3

$s_\alpha(t) = e^{\neg 2} e^{\neg \alpha t} \cos(4\pi t) u(t)$ is plotted for $\alpha \in \{1,3,5,7\}$ in one figure for $t = [0 : 0.01 : 4]$. Figure 3.4, and 3.5 represents the codes for generating these same graphs using algorithm

vectorization, and for loop structure, respectively. Besides, figure 3.6, and 3.7 illustrates the equivalence between the graphs using these very different methods.

```
alpha = (1 : 2 : 7);
%lets create a column vector as t
t = (0 : 0.01 : 4)';
T = t * ones(1, 4);
% defining the u(t)
u = @(T)1.0 .* (T >= 0);

%k_demo = T * diag(alpha).*u(T-1);
%dummy_2 = diag(alpha);
%plot (t, k_demo);
H = exp(-2)*exp(-T * diag(alpha)).* cos(4*pi*T).*u(T);
plot(t, H);
xlabel('t'); ylabel('exp(-2)exp(-at)cos(4*pi*t)u(t)');grid;
clear all;
```

Figure 3.4: C.3 algorithm vectorization code

```
t = (0 : 0.01 : 4);
% defining the u(t)
u = @(t)1.0 .* (t >= 0);

for i = 0:3
    plot(t, exp(-2)*exp(-1*((2*i) + 1)*t).*cos(4*pi*t).*u(t));
    grid;
    xlabel('t');
    ylabel('exp(-2)exp(-at)cos(4*pi*t)u(t)');
    hold on;
end

hold off;
clear all;
```

Figure 3.5: C.3 for loop version code

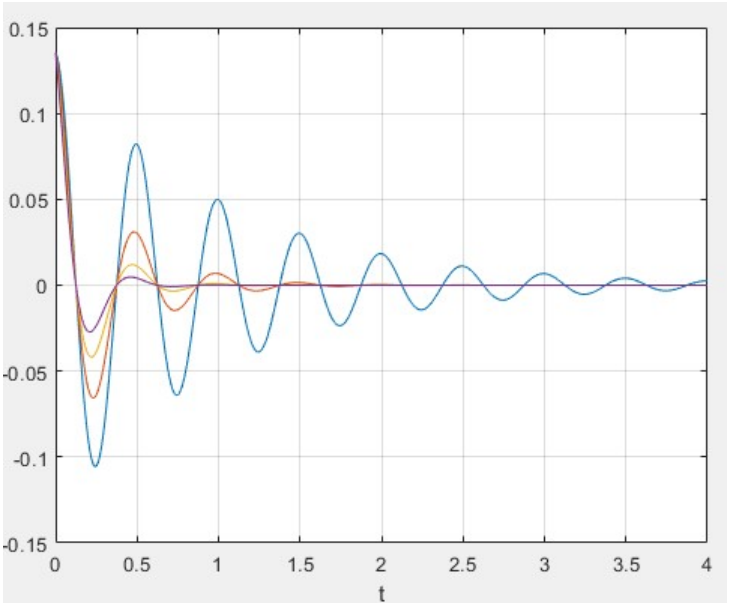

Figure 3.6: C.3 graph using vectoriztion

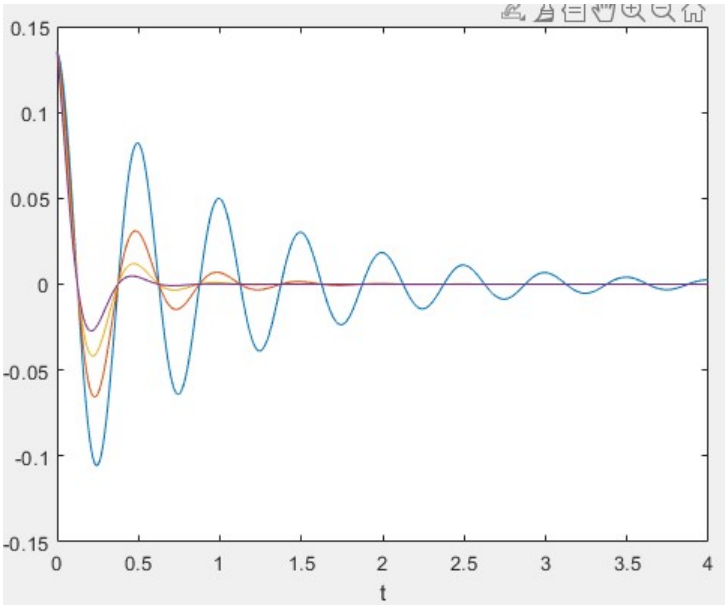

Figure 3.7: C.3 graph using for loop structure

3.4 C.4

To determine size of any 2D matrix H, we use the 'size(H)' function, which returns another 1x2 matrix with row and column numbers. The code and result are presented in figure 3.8, and 3.9 respectively.

```
alpha = (1 : 2 : 7);
%lets create a column vector as t
t = (0 : 0.01 : 4)';
T = t * ones(1, 4);
% defining the u(t)
u = @(T)1.0 .* (T >= 0);
H = exp(-2)*exp(-T * diag(alpha)).* cos(4*pi*T).*u(T);
dimension = size(H)
```

Figure 3.8: C.4 determination of the size of the matrix $s(t)$, namely 'H'

```
dimension =

   401     4
```

Figure 3.9: C.4 The result showing the number of rows and columns of the matrix $s(t)$ ('H' according to our code).

Chapter 4

Part D

4.1 D.1

1. The outcome of A(:) in general, reshapes a matrix into a column vector.

2. When using the command A([2 4 7]), it takes the column vector generated in part a and only outputs index 2, 4, 7.

3. The command [A ¿= 0.2] takes all the values that are larger than or equal to 0.2 and switches them to a 1 while changing all the values that do not conform to ¿= 0.2, to 0.

4. The command A([A ¿= 0.2]) takes all the values that are larger than or equal to 0.2 and puts them into a column vector.

5. The command A([A¿= 0.2]) = 0 takes all the values that are larger than or equal to 0.2 and makes them 0.

4.2 D.2

The below image, shows section D.2 parts a-c.

```
%%%%%%%%%%%Part 1
numrows = size(B,1)
numcols = size(B,2)
for i = 1:numrows
for j = 1:numcols
if(abs(B(i,j)) > 0.1)
B(i,j) = 0;
end
end
end

%%%%%%%%%%%Part 2
B([B <= 0.01]) = 0

%%%%%%%%%%%Part 3
tic
t1= toc
for i = 1:numrows
for j = 1:numcols
if(abs(B(i,j)) > 0.1)
B(i,j) = 0;
end
end
end
toc

tic
B([B <= 0.01]) = 0
t2 = toc
toc

fprintf("Time for first %f\n", t1)
fprintf("Time for second %f ", t2)
```

Figure 4.1: Part D.2 Parts A-C

4.3 D.3

```
sound(x_audio,8000)

X = x_audio

X([X <= 0.001]) = 0

J = nnz(~x_audio)

sound(X,8000)
```

```
X = 20000×1
    0.2030
    0.1639
    0.1327
    0.0760
    0.0238
   -0.0438
   -0.0800
   -0.0878
   -0.1073
   -0.1190
    ⋮

X = 20000×1
    0.2030
    0.1639
    0.1327
    0.0760
    0.0238
         0
         0
         0
         0
         0
    ⋮

J = 58
J = 58
```

Figure 4.2: Part D.3