ELE632 Lab 1 Report

Table of Contents

Authors: Mai Abdelhameed & Omar Ahmad

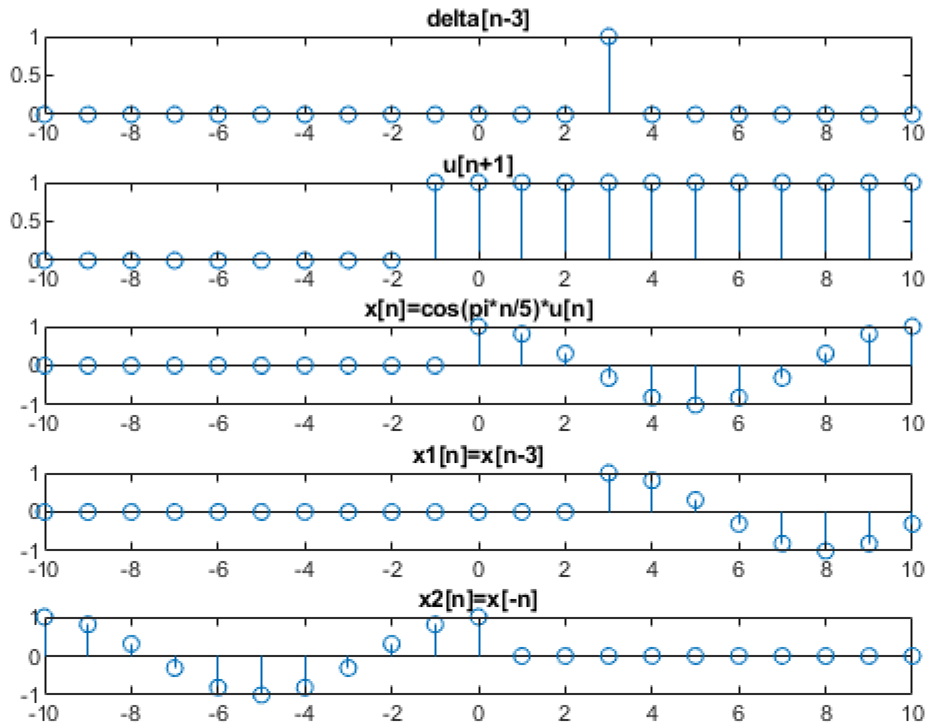
Part A: Signal Transformation

A.1:

```
n = [-10:10];   %Creates the range of n, (-10 < n < 10) with steps
                %of 1 in between

% A.1-I
impulse = @(n) (n == 0) * 1.0 .* (mod(n, 1) == 0);
a = impulse(n-3);

%A.1-II
u = @(n) (n >= 0) * 1.0 .* (mod(n,1)==0);
    %Creates a unit step function that is usable by
    %calling u(n), the (mod(n,1)==0) term forces the function to be a
    %discrete time function by only saving the values of integer
 values
    %of n
b = u(n+1);

%A.1-III
x = @(n) u(n) .* cos((n .* pi) / 5);
c = x(n);

%A.1-IV
x1 = @(n) x(n-3);
d = x1(n);

%A.1-V
x2 = @(n) x(-n);
e = x2(n);

plots = {a,b,c,d,e}; % a cell of objects that holds variables
titles = {"delta[n-3]", "u[n+1]", "x[n]=cos(pi*n/5)*u[n]", ...
            "x1[n]=x[n-3]", "x2[n]=x[-n]"};
```

```
figure
for i = 1:length(plots)
 subplot(length(plots),1,i);
 stem(n,plots{i}); %indexing into a cell using {i} to get
                      %the i'th element
    title(titles{i});
end

disp('X1[n] is being time shifted. X2[n] is being time reversed.')
```

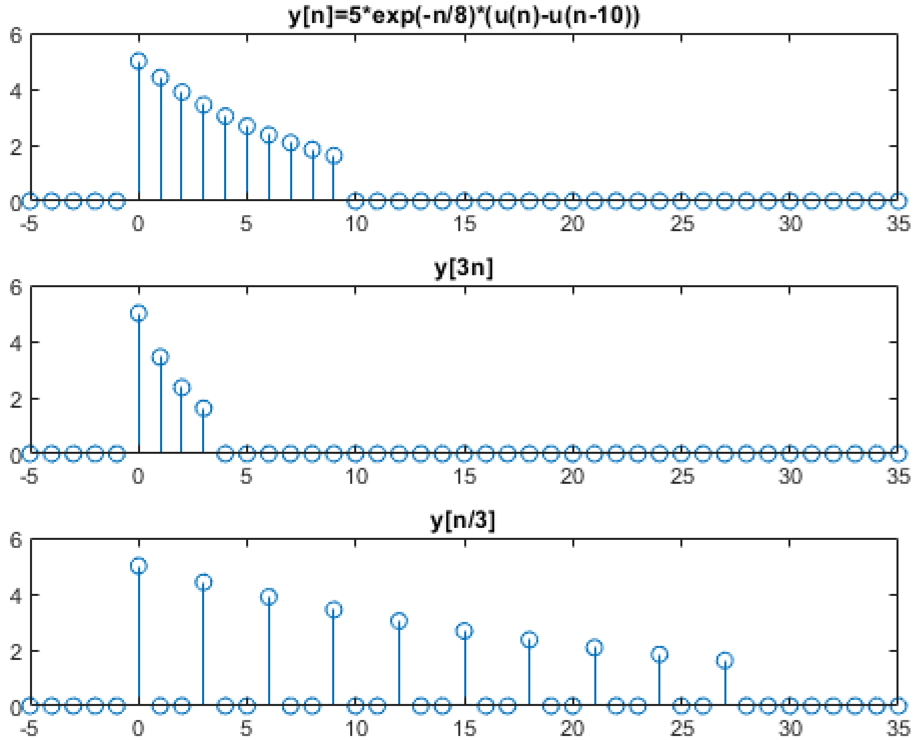
X1[n] is being time shifted. X2[n] is being time reversed.



A.2

```
clear;

%Note that the range was shortened to -5:35 instead of the original
%-10:70 since there was a lot of unnecessary empty space.
n = [-5:35];

%A.2-I
u = @(n) (n >= 0) * 1.0 .* (mod(n,1)==0);
y = @(n) 5*exp(-n/8).*(u(n)-u(n-10));
a = y(n);
```

```
%A.2-II
y1 = @(n) y(3*n);
b = y1(n);

%A.2-III
y2 = @(n) y(n/3);
c = y2(n);

plots = {a,b,c};
titles = {"y[n]=5*exp(-n/8)*(u(n)-u(n-10))","y[3n]","y[n/3]"};

figure
for i = 1:length(plots)
 subplot(length(plots),1,i);
 stem(n,plots{i});
    title(titles{i});
end

%y1[n] and y2[n] are time scaling transforms, a compression and
 expansion
%respectively
```

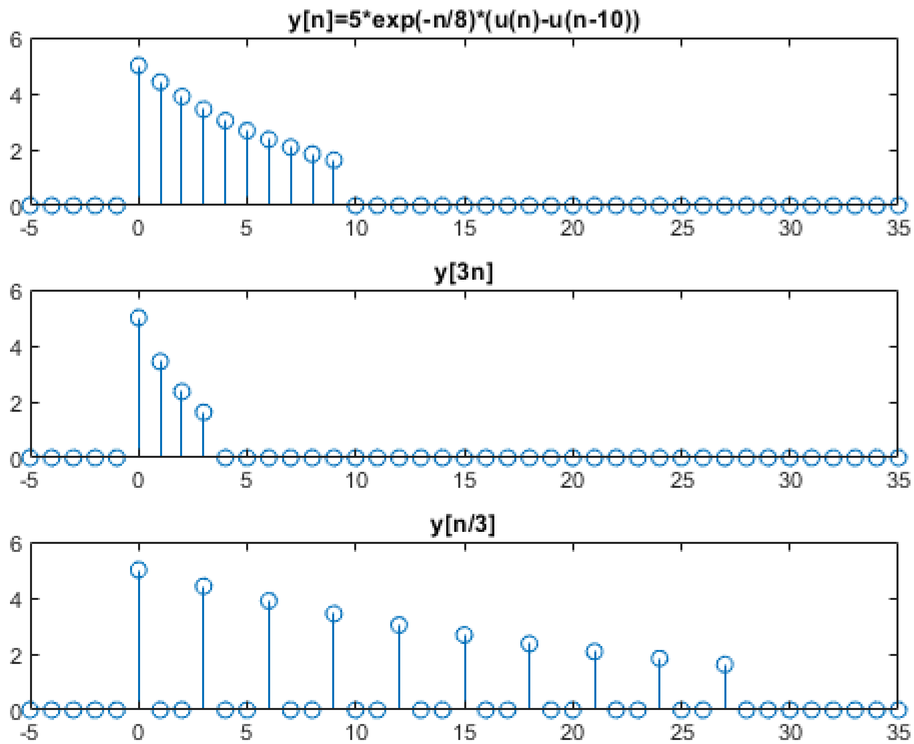


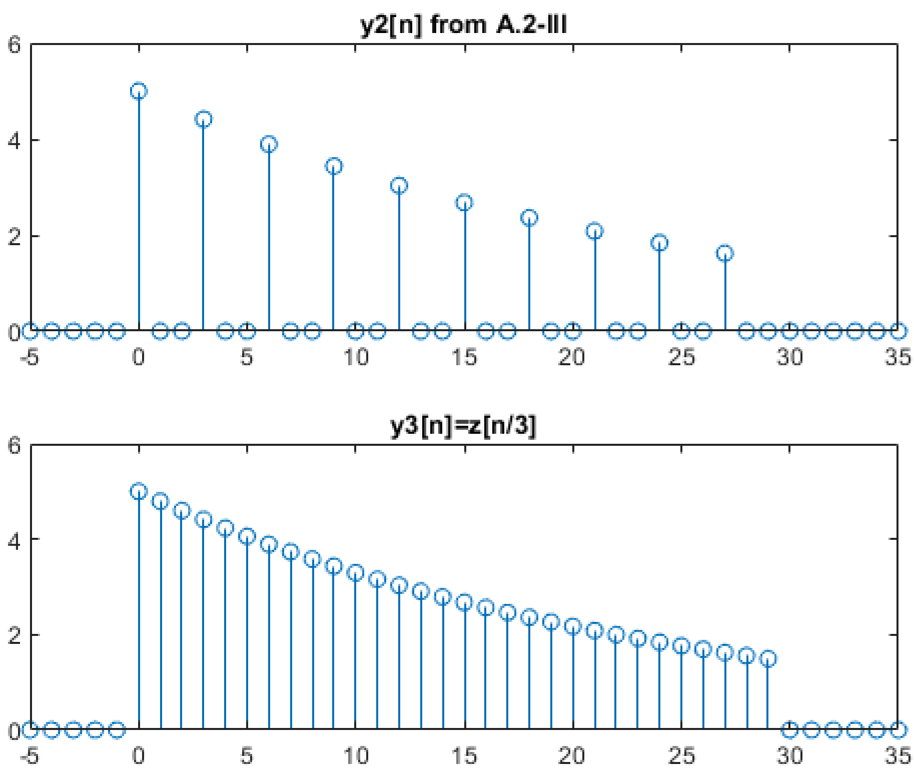
A.3

```
clear;
```

```
u = @(n) (n >= 0) * 1.0 .* (mod(n,1)==0);
y = @(n) 5*exp(-n/8).*(u(n)-u(n-10));
y2 = @(n) y(n/3);

%A.3-I
u1 = @(n) (n >= 0) * 1.0;
z = @(n) 5*exp(-n/8).*(u1(n)-u1(n-10));
y3 = @(n) z(n/3) .* (mod(n,1)==0);
n = [-5:1:35];

figure
subplot(2,1,1);
stem(n,y2(n));
title("y2[n] from A.2-III");

subplot(2,1,2);
stem(n,y3(n));
title("y3[n]=z[n/3]");

disp('We notice that y3[n] has more data values than y2[n] because of
 the');
disp('fact that the signal transformation was applied to the
 continuous');
disp('signal first, allowing the sampling to sample values that NOW
 exist');
disp('in discrete integer values, which previously didnt before
 stretching' );
disp('the continuous function.');
```

We notice that y3[n] has more data values than y2[n] because of the fact that the signal transformation was applied to the continuous signal first, allowing the sampling to sample values that NOW exist in discrete integer values, which previously didnt before stretching the continuous function.


y[n]=5*exp(-n/8)*(u(n)-u(n-10))
y[3n]
y[n/3]
0
2
4
6
-5
0
5
10
15
20
25
30
35


y2[n] from A.2-III
y3[n]=z[n/3]
0
2
4
6
-5
0
5
10
15
20
25
30
35

Part C: Design a Filter: N-point maximum filter

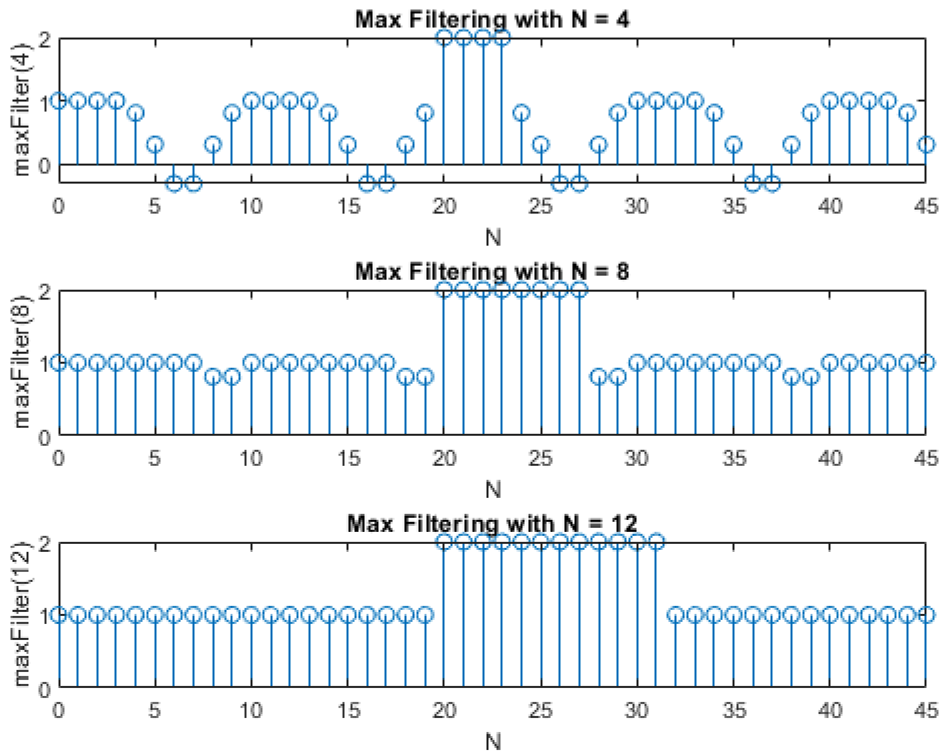
C.1

The code below is the contents of the function written inside maxFilter.m

```
function output = maxFilter(N)

%This function filters a function
%with steps equalling N

n = [0:45];
impulse = @(n) (n == 0) * 1.0 .* (mod(n, 1) == 0);
x = @(n) (cos(pi*n/5)+impulse(n-20)-impulse(n-35))...
    * 1.0 .* (mod(n,1)==0);

f = x(n); %creates an array of output values
f1 = [zeros(1,(N-1)) f]; %pads the beginning with the appropriate
                         %number of zeroes
output = [];
for i = 1:length(f1)-(N-1)
    temp = f1(1:1, i:i+(N-1));
    m = max(temp);
    output = [output m];
end
```

C.2

```
clear;

n = [0:45];

figure;
subplot(3,1,1);
stem(n,maxFilter(4));
title("Max Filtering with N = 4");
xlabel("N")
ylabel("maxFilter(4)");
subplot(3,1,2);
stem(n,maxFilter(8));
title("Max Filtering with N = 8");
xlabel("N");
ylabel("maxFilter(8)");
subplot(3,1,3);
stem(n,maxFilter(12));
title("Max Filtering with N = 12");
xlabel("N");
ylabel("maxFilter(12)");
```



Published with MATLAB® R2018b