Assessment (ECE2111)

Lab 01 Result Document

MONASH University

Name: Tan Jin Chun

Student ID: 32194471

ECE2111 lab1 results document:

Name: *Tan Jin Chun*
Names of all students you discussed this lab with: *Loh Jia Quan, Chong Yen Juin, Kelvin Ku Yew Siang, Huan Meng Hui*

Section 1:
Output from item 1: 0.0000 + 1.0000i

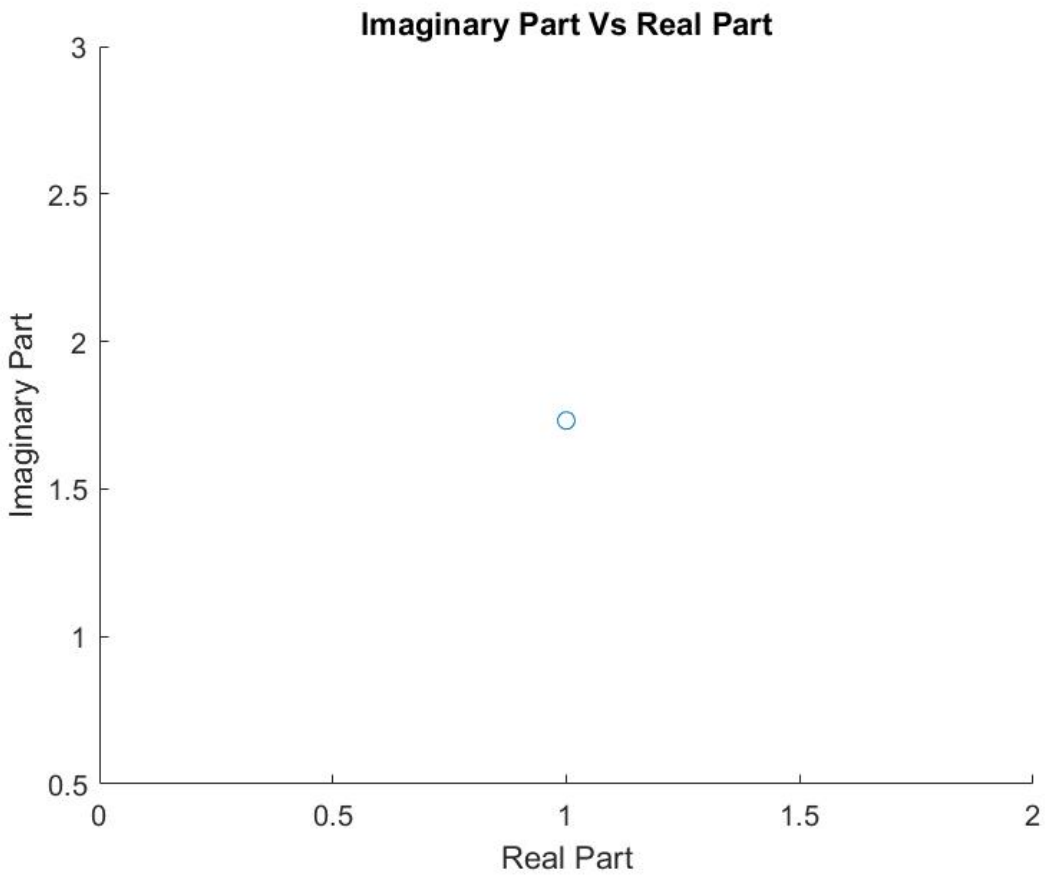
Output from item 2: 0.0000 + 2.0000i
0.0000 + 2.0000i

Output from item 3: Magnitude = 1.4142, Phase = 0.7854, Real = 1, Imaginary = 1

Answer to item 4: Radians

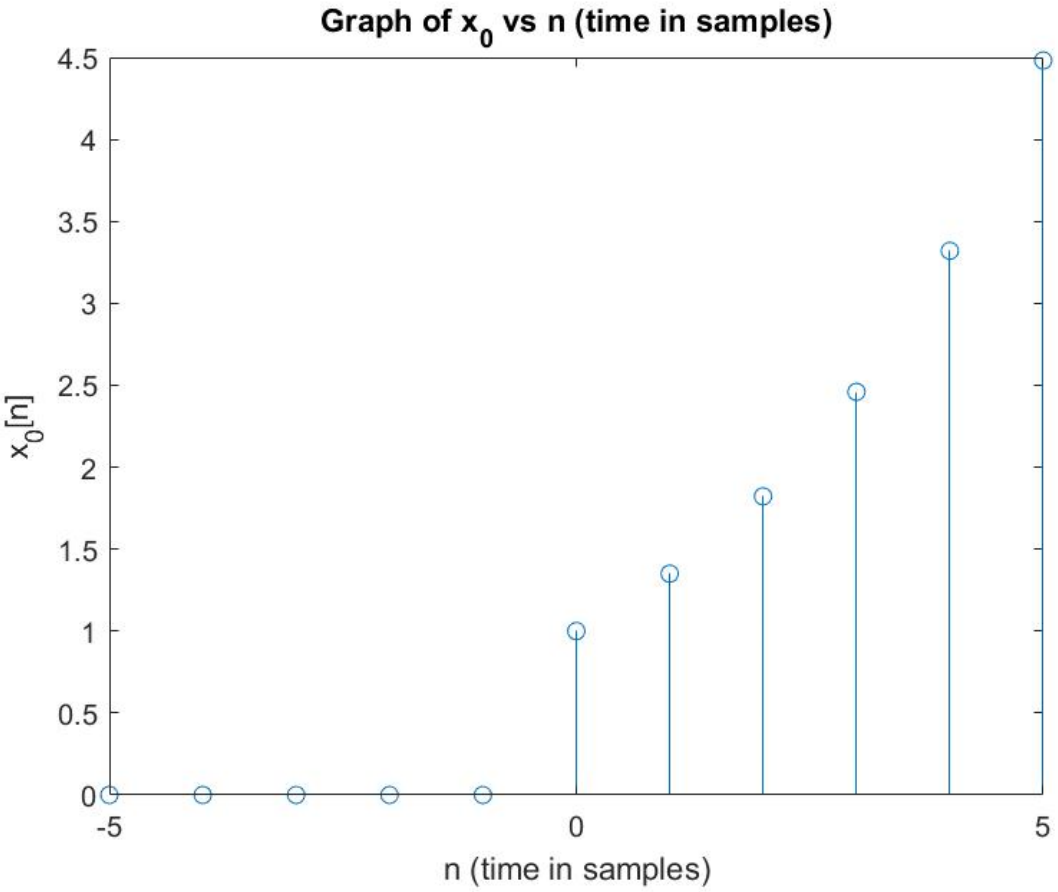
Output from item 5: 3.3859

Plot from item 6:

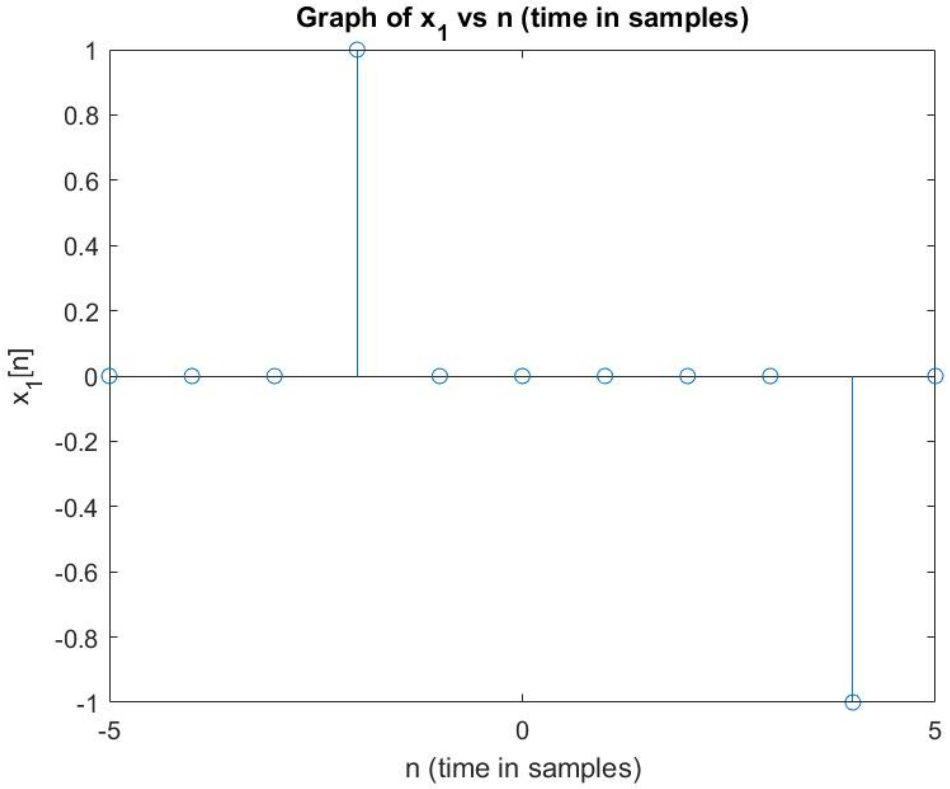


Section 2:
Include your figures for parts (a), (b), and (c), below.

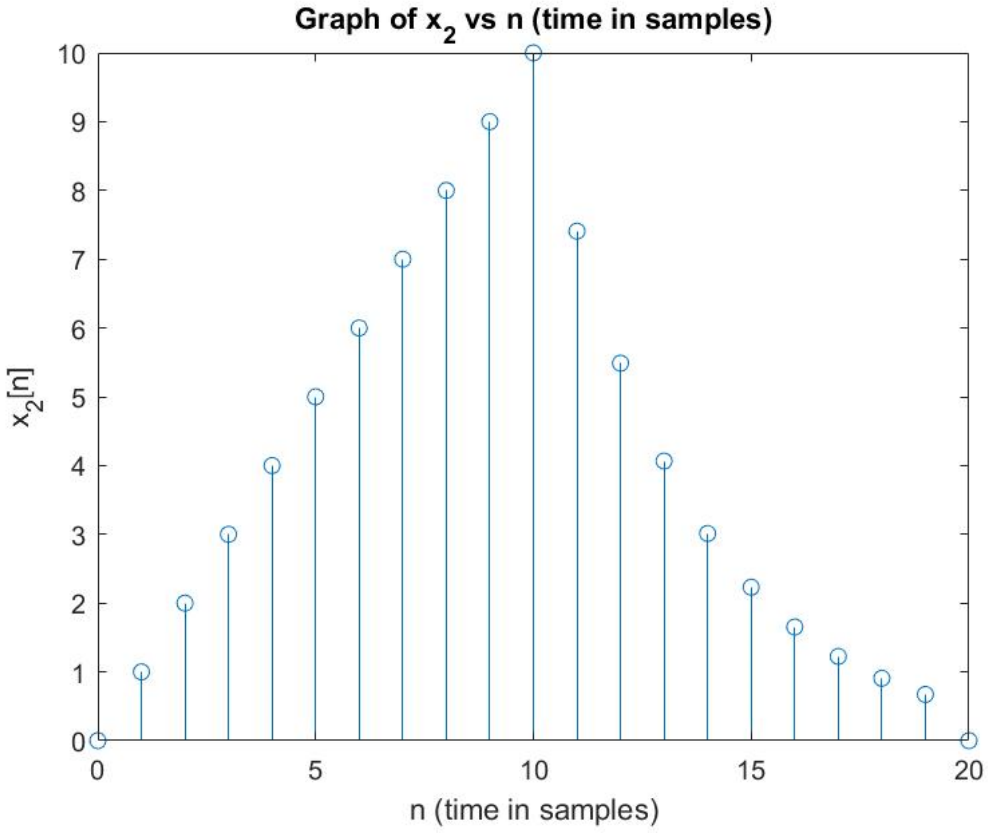
Parts (a)



Parts (b)

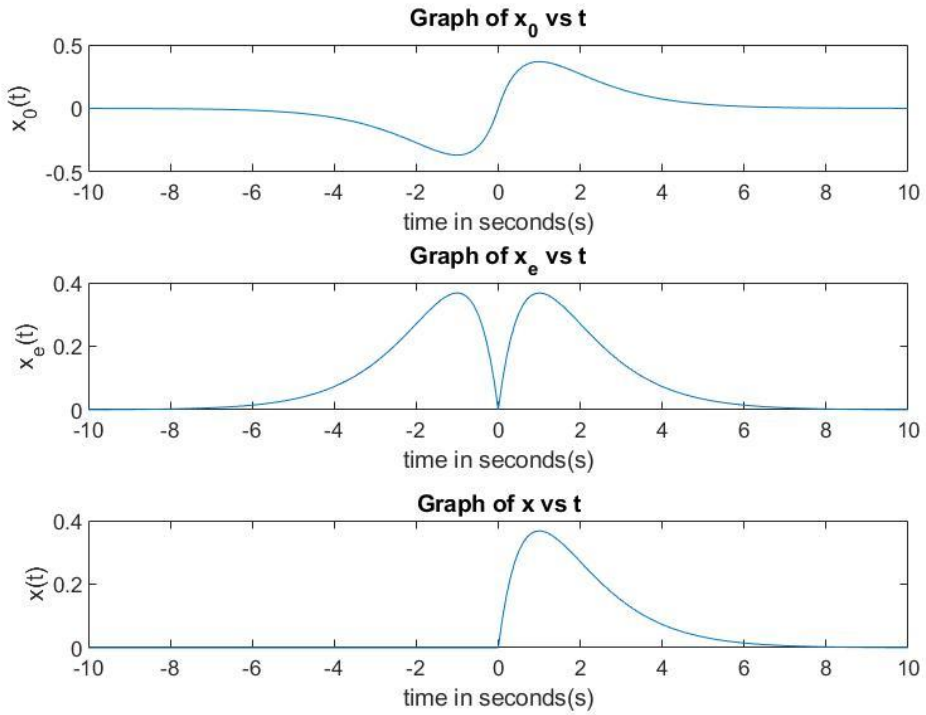


Parts (c)



Section 3:

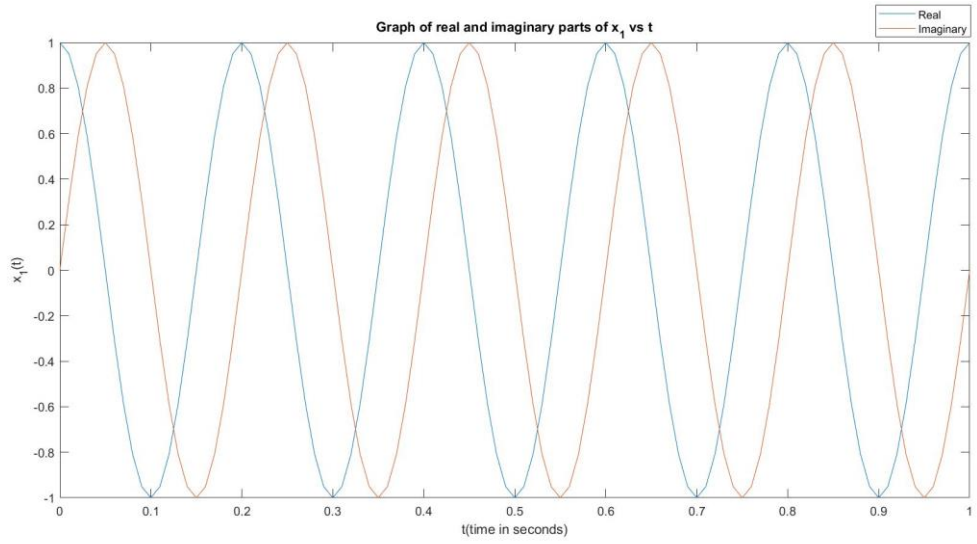
Include your figure from item 3, below:



4.What special property does signal xe have? What about the signal xo?

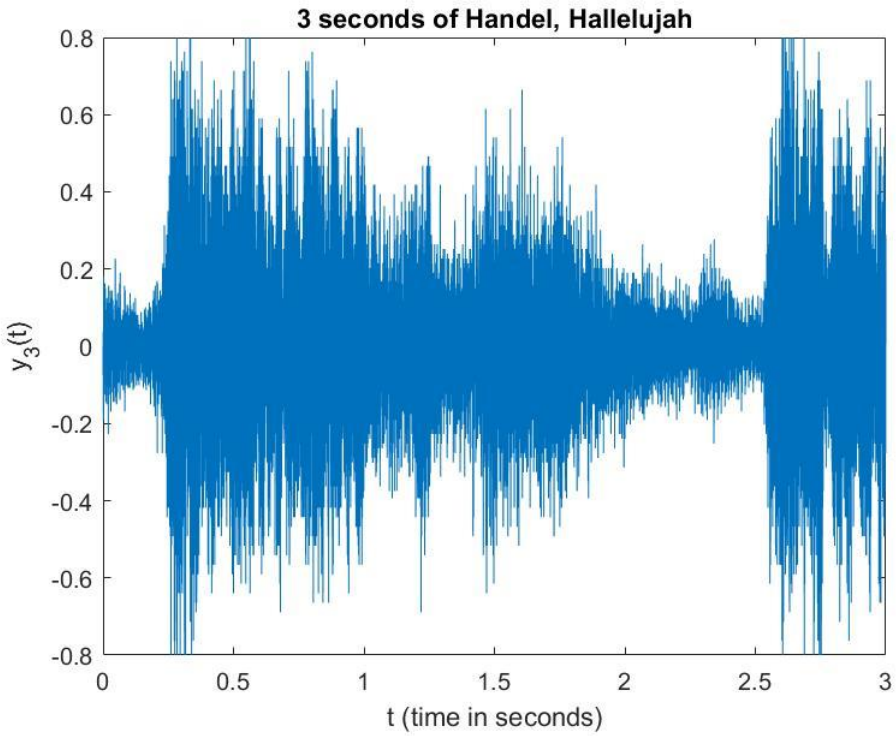
Signal xe is an even signal since the signal can be reflected about the origin, a bounded signal and aperiodic signal.
Signal xo is an odd signal since the signal can be reflected about the y-axis, a bounded signal and aperiodic signal.

Include your figure from item 6, below:



Section 4

Include your figure from item 6, below:



Question 2: Is the sound higher or lower than before? Why do you think this is happening?
The sound is lower than before. This is because the sampling frequency of the sound is lower than the normal sampling frequency. When the sampling frequency is lower, the pitch of the sound will be lower. Thus, the sound will be lower than before.

Question 3a: Use `sound()` *to play the signal ys = 2xs. What do you hear? Why do you think you hear this?*
The sound should have a similar pitch (frequency) with sound xs but with some distortion. At each point in time, the amplitude at each signal is scaled by 2. Thus, the signal will be louder since a larger amplitude corresponds to a larger sound. However, If the magnitude exceeds by 1. The sound built-in function will set a limit between 1 and -1, causing distortion in the sound

Question 3b: Use `soundsc()` *to play the signal ys = 2xs. What do you hear? Use* `help soundsc` *to explain this.*
The sound is the same as the sound for xs. Soundsc will only scale the value for signal ys to fit in the range (-1.0 to 1.0). This would then revert ys signal back to xs signal.

Question 4: How many samples correspond to the first three seconds of sound of the `y` *signal (including the sound at time t = 0 and time t = 3)?*

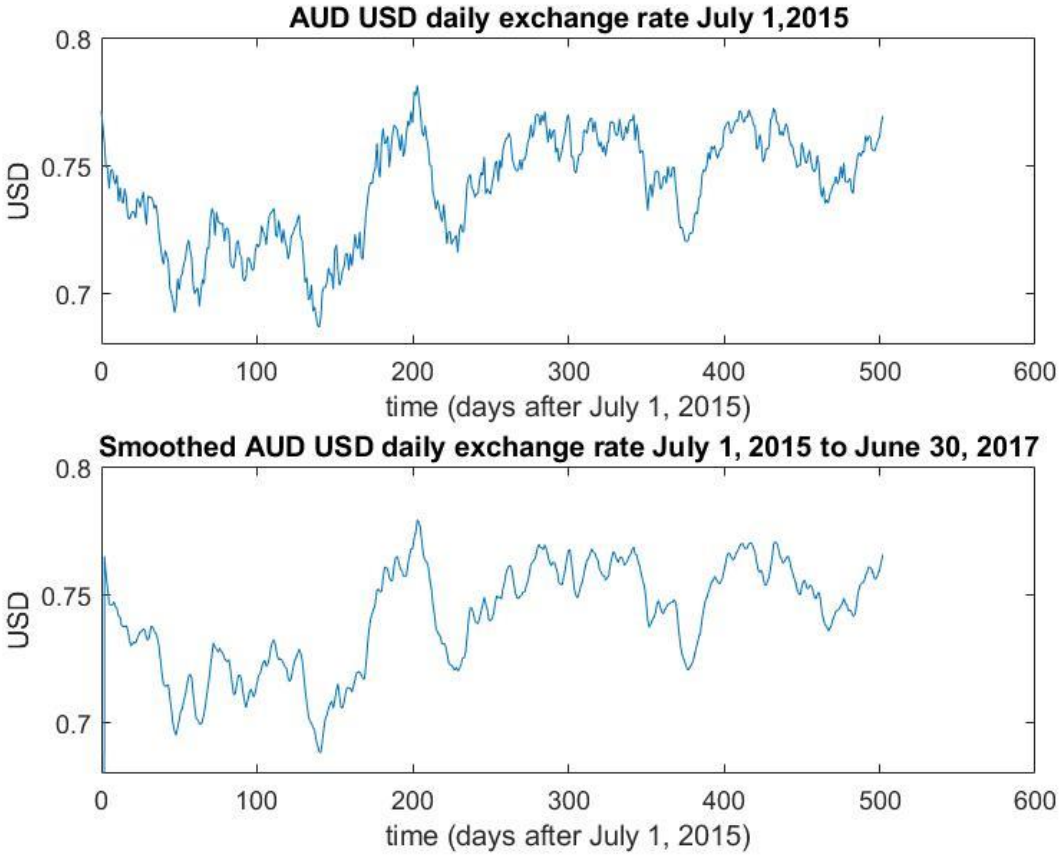
Sampling Period = 8192(Samples/Second)
*Total Number of Samples = 3*8192 + 1 (Include the signal when t = 0) = 24577 Samples*

Section 5:

Question: Does your `sumsys()` *system also work for continuous-time signals?*

Yes, the sumsys() system works for continuous time signals as well if the assumption made is that both of the signals are of the same length.

Include your plot of `aud` *and* `audout` *from item 6(b), below:*



Code for section 1:

Code from section 1.1:

```
Paste your code in here.
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01T01

clear all;close all;clc

% Question 1
% Entering sqrt(1) to display the result
x = sqrt(-1);

% Question 2
% Enter 1+j and 1i + 1j and display the result
y = i + j;
z = 1i + 1j;

% Question 3
% Defining z1 = 1 + j by entering z1 = 1+1j

% Initialising z as 1 + j
z_1 = 1 + j;

% Magnitude
Magnitude = abs(z_1);

% Angle
Angle = angle(z_1);

% Real Number
Real = real(z_1);

% Imaginary Number
Imaginary = imag(z_1);

% Question  4
% The phase is in radians. Angle(z_1)returns the phase angle in the
% interval [-?,?] for each element of a complex array z .

% Question 5
z_2 = 2*exp((1j*pi)/3);

% Displaying the magnitude of z1+z2
z_ans = z_1 + z_2;
abs_ans = abs(z_ans);

% Question 6
% Plotting the complex number z2 = 2*exp(j*pi/3)
z_3 = 2*exp((1j*pi)/3);
figure
x = real(z_3);
y = imag(z_3);
scatter(x,y);
title("Imaginary Part Vs Real Part")
```

```
xlabel("Real Part")
ylabel("Imaginary Part")
```

Code for section 2

Code for function `dtstep`:

Paste your code here.

```
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01T02
% The name of this function is dtstep
% The purpose of this function is to return a representation of the
signal
% x[n] = delta[n-n_o] for the time n1<=n<=n2

% Only need to change one line of the given code (dtimpulse.m)

function [x,n] = dtstep(n0,n1,n2)
    % dtstep: returns discrete-time unit step function
    % centered at n0 over time range n1:n2
    % [x, n] = dtstep(n0,n1,n2)
    % where n0 n1 and n2 are integers with n1 <= n0 <= n2,
    % produces signal x[n] = u[n-n0] for n1<=n<=n2.
    n = n1:n2;
    x = zeros(1,length(n));

    % Just change this line from dtimpulse function file
    % Explanation: Based on the lecture notes, the discrete time
unit step
    % u[n] will have a value of one if n is bigger or equal to 0
    x(n>=n0) = 1;

end
```

Code for section 2 item 2 (script to plot three discrete-time signals):

Paste your code here.

```
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01T02

clear all;close all;clc

% Question 2(a)
% Initialising the variable
n1 = -5;
n2 = 5;

% Centered at 0 , so n0 = 0, between -5 and 5
% Calling the function dtstep
[x,n] = dtstep(0,n1,n2);
x0 = exp(0.3.*n).*x;

% Plotting the graph
figure
stem(n,x0);
title("Graph of x_0 vs n (time in samples)");
xlabel("n (time in samples)");
ylabel("x_0[n]");

% Question 2(b)
% Initialising the variable
n1 = -5;
n2 = 5;

% Calling the function dtstep
% For delta[n+2]
% n -(-2)
[u0,n] = dtimpulse(-2,n1,n2);

% For delta[n-4]
% n-(4)
[u1,n] = dtimpulse(4,n1,n2);
x1 = u0-u1;

% Plotting the graph
figure
stem(n,x1);
title("Graph of x_1 vs n (time in samples)");
xlabel("n (time in samples)");
ylabel("x_1[n]");

% Question 2(c)
% Initialising the variable
n1 = 0;
n2 = 20;

% Calling the function dtstep
[u2,n] = dtstep(0,n1,n2);
[u3,n] = dtstep(10,n1,n2);
```

```
[u4,n] = dtstep(20,n1,n2);

% From the formula
x2 = n.*(u2-u3) + 10.*exp(-0.3.*(n-10)).*(u3 - u4);

% Plotting the graph
figure
stem(n,x2);
title("Graph of x_2 vs n (time in samples)");
xlabel("n (time in samples)");
ylabel("x_2[n]");
```

Code for section 3

Code from section 3.2:

Paste your code in here.

```
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01T03

clear all;close all;clc

% Initialsing the variables
t = linspace(-10,10,201);
x0 = t.*exp(-1*abs(t));
xe = abs(t).*exp(-1*abs(t));
x = 0.5*(x0+xe);

% Plotting the graph in a subplot
subplot(3,1,1);
plot(t,x0);
title("Graph of x_0 vs t")
xlabel("time in seconds(s)")
ylabel("x_0(t)")
subplot(3,1,2);
plot(t,xe);
title("Graph of x_e vs t")
xlabel("time in seconds(s)")
ylabel("x_e(t)")
subplot(3,1,3);
plot(t,x)
title("Graph of x vs t")
xlabel("time in seconds(s)")
ylabel("x(t)")

% Question 5
t1 = linspace(0,1,101);
x1 = exp(1j*10*pi*t1);

% Plotting the graph
figure
% Plotting the real part of the graph
plot(t1,real(x1))

% Plotting the imaginary part of the graph
hold on
plot(t1,imag(x1))
hold off
title("Graph of x_1 vs t")
xlabel("time in seconds(s)")
ylabel("x_1(t)")
```

Code for section 4:

Code from section 4.2:

Paste your code in here.

```
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01T04

clear all;close all;clc

% Part 1
% Initialising the variables
% 24577 samples are needed since 3*8192+1 = 24577
t = linspace(0,3,24577);
xs = cos(2*pi*440.*t);
fs = 8192;

% Playing the sound
sound(xs,fs);
pause

% Part 2
% The sound will have a lower pitch when played using a lower
frequency
sound(xs,8000);
pause

% Part 3
% Creating a new signal with a factor of 2
ys = 2*xs;

% Playing the sound again
sound(ys,fs);
pause

% I hear a beep sound together with the original sound

% Playing the sound using soundsc()
soundsc(ys,fs);

% The data is scaled so that the sound is played as loud as possible
with
% no clipping.

% Part 4
% Loading handel
load handel;

% Part 5
% Song is Hallelujah
y3 = y(1:length(t));
sound(y3,Fs);

% Part 6
% Plotting the continuous time signal
ty3 = linspace(0,3,length(y3));
```

```
plot(ty3,y3)
title("3 seconds of Handel, Hallelujah")
xlabel("t (time in seconds)")
ylabel("y_3(t)")
```

Code for section 5:

Code for the function `sumsys()`

```
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01Section5Part1
% The name of this function is sumsys
% The purpose of this function is to implement a sum system
(addition)

function [y, nout] = sumsys(x1, nin1, x2, nin2)
% gainsys: implements sum system
% [y, nout] = gainsys(x1,nin1,x2,nin2)
% where x1 and nin1 are row vectors of the same length and the same
goes
% for x2 and nin2
% produces signal y obtained by adding x1 by x2
% This system works for both discrete time and continuous time as
long as both of the vector are of the same length
% time

    if length(x1) ~= length(nin1)
        error("arguments 1 and 2 should have the same length");
    end

    if length(x2) ~= length(nin2)
        error("arguments 3 and 4 should have the same length");
    end

    % Preallocating for speed
    nout = 1:length(nin2);
    y = 1:length(nin2);

    % Using a for loop to loop through the vectors
    for i = 1:length(nin2)
        % Getting the sum of the signals
        nout(i) = nin1(i);
        y(i) = x1(i) + x2(i);

    end

end
```

Code for the function `delaysys()`

```
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01Section5Part3
% The name of this function is delaysys
% The purpose of this function is to implement a delay system

function [y, ny] = delaysys(N, x, nx)
% gainsys: implements delay system
% [y, ny] = gainsys(N, x, nx)
% where x and nx are row vectors of the same length, and N is a
scalar,
% produces signal y obtained by delaying x by N and having the same
time indices.

    if length(x) ~= length(nx)
        error("arguments 1 and 2 should have the same length");
    end

    if ~isscalar(N)
        error("argument 1 should be a scalar");
    end

    % Getting the delay of the signals
    ny = nx;

    % Preallocating for speed
    y = zeros(1,length(x));

    % Using a for loop to go through the vector
    for i = 1:length(x)
        if (i - N >0 && i - N <=length(x))
            % Delaying the signal by N times (shifting the signal by
N)
            y(i) = x(i-N);
        end
    end
end
```

Code for the function `threepointaverage()`

```
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01Section5
% The name of this function is threepointaverage
% The purpose of this function is to implement a delay system

function [y, ny] = threepointaverage(x, nx)
% gainsys: implements a three point average system
% [y, ny] = threepointaverage(x, nx)
% where x and nx are row vectors of the same length
% produces an output y after going through a three point average
system

    % Unneccesary to check the length since the sumsys and delaysys
will
    % check the function

    % Delay x by 1 to get x1
    [x1, nx1] = delaysys(1, x, nx);

    % Delay x by 1 to get x2
    [x2, nx2] = delaysys(1,x1,nx1);

    % Gain x, x1 and x2 by 1/3
    [u, ux] = gainsys(1/3,x,nx);
    [u1, ux1] = gainsys(1/3,x1,nx1);
    [u2, ux2] = gainsys(1/3,x2,nx2);

    % Adding the three signals together
    [w, wout] = sumsys(u, ux, u1, ux1);
    [y, ny] = sumsys(w, wout, u2, ux2);

    % There is of course a more efficient way to do this. I am just
    % following the block diagram given
    % Three Point Average
    y(1:2) = 0;
end
```

Code for the script:

```
% Written by Tan Jin Chun
% Last Modified: 25/7/2021
% Lab01T05

clear all;clc;close all

% Part 2
% Checking if my sumsys() function is working
nin1 = 0:3;
nin2 = 0:3;
x1 = [1 0 0 0];
x2 = [1 2 3 4];

% Calling the sumsys() function
[y1, nout] = sumsys(x1, nin1, x2, nin2);

% Part 4
% Checking if my delaysys() function is working or not
N = 2;
x = [1 0 0 0];
nx = 0:3;

% Calling the delaysys() function
[y2, ny] = delaysys(N, x, nx);

% If N=4, the signal will be all zero
N = 4;

% Calling the delaysys() function again
[y3, ny1] = delaysys(N, x, nx);

% Part 6
% Loading the .mat file
load AUDUSD

% Testing the threepointaverage() function
[y, ty] = threepointaverage(aud, taud);

% Plotting the subplot
subplot(2,1,1);
plot(taud,aud);
title("AUD USD daily exchange rate July 1,2015");
xlabel("time (days after July 1, 2015)");
ylabel("USD");

% Setting the vertical axis limit
ylim([0.68, 0.8]);

% Plotting the next subplot
subplot(2,1,2);
plot(ty,y);
title("Smoothed AUD USD daily exchange rate July 1, 2015 to June 30,
2017");
xlabel("time (days after July 1, 2015)");
```

```
ylabel("USD");

% Setting the vertical axis limit
ylim([0.68,0.8])
```