ECGR 2254 – Project 2
Due: 11/25/19
Name: Colby Sheppard
Collaborators: Grant Caskey, William Shaughnessy, James Tallett

Problem 1: A test file was provided with a variable x containing a list of points for a sinusoidal voltage.

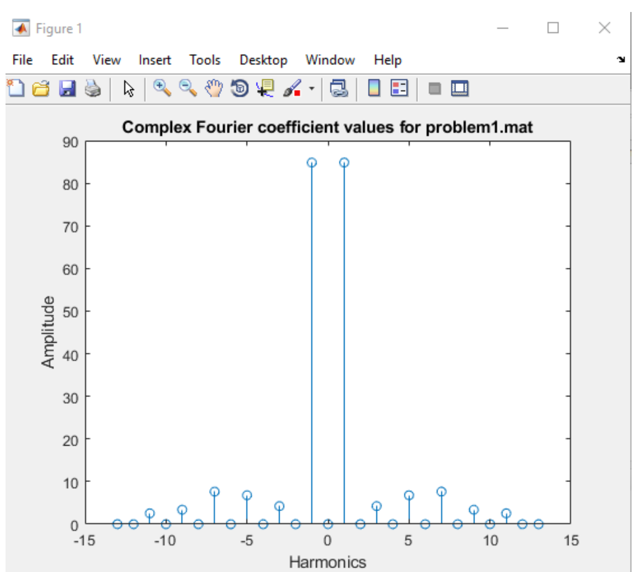
a. Given a voltage signal, the complex coefficients of its sine and cosine components were calculated using MATLAB. Two nested For-Loops were needed to do this without any built-in MATLAB functions. The outer loop iterates for the required number of harmonics, 27 in total, while the inner loop iterates based on numeric integration using the Fourier Series complex formula:

$$C_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t)e^{-jnwt}dt$$

The results of a Fourier Series are equivalent to a periodic waveforms' Fourier Transform.

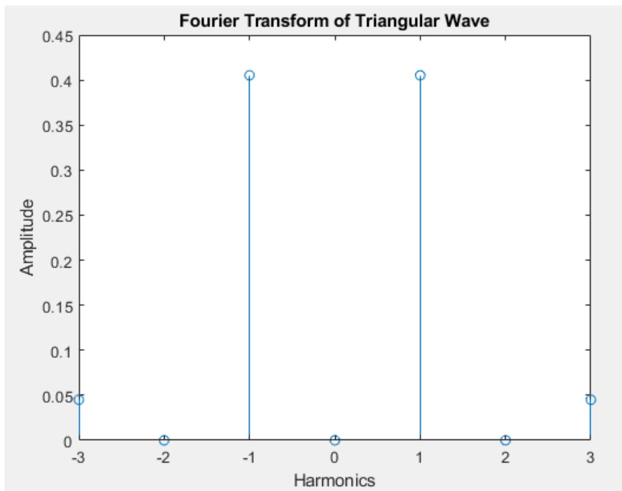
b. The transform coefficients were then plotted in MATLAB using the stem function to provide discrete point results for each harmonic point calculated in part a.

```
load('problem1.mat');

T = 1/60;
w_0 = 2*pi/T;
T_s = 1/60000;
t = [0:T_s:T];

n = [-13:1:13];

alpha = zeros(27,1);

for k = 1:1:length(n)

    for i = 1:1:1000
          alpha(k) = alpha(k) + (x(i)* T_s) * exp(-j * w_0 * n(k) * t(i));
    end

alpha(k) = alpha(k) *(1/T);
end

figure(1);
clf;
stem(n,abs(alpha))
```



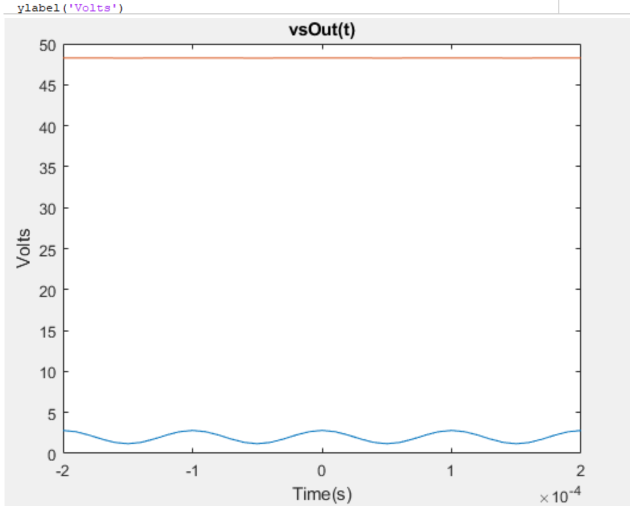
<u>Problem 2:</u> A 60Hz signal corrupted by a higher frequency sawtooth wave was provided. An appropriate filter and corner frequency were then chosen to reduce the signal as much as possible without shifting the phase of the original signal.

a. The sawtooth wave was reproduced using the sawtooth command in MATLAB. Using numeric integration similar to project 1, the Fourier Coefficients for positive and negative harmonics 1-3 were calculated. Those coefficients were then turned into shifted cosine form by computing their angle and magnitude in MATLAB, then multiplying the magnitude by two.



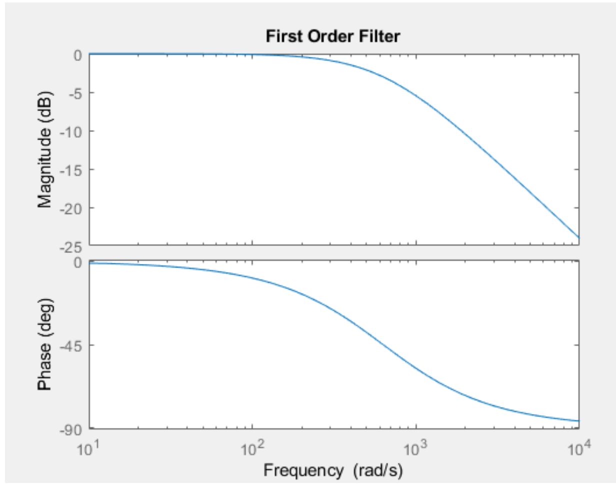

```
T = 1e-4;%period of triangle wave
w_0 = 2*pi/T;
fs =  10000000; %sample rate
t = -2e-4: 1/fs : 2e-4; %x axis
x = sawtooth(w_0 * t + pi , 1/2);

plot (t,x)

w= [.01:.01:15000];
wc = 3175*(2*pi);

%filter
H = tf([wc^4], [1 2.6132*wc 3.41430612*wc^2 2.6132*wc^3 wc^4]);

n = [-3:1:3];%harmonics
T_s = T/1000; %time step(delta t)
t1 = [0:T_s:T];%time vector
alpha = zeros(7,1);%result storage

m = (1/2)*T : T_s : (3/2)*T; %one period of function split into 1000 pieces


for k = 1:1:length(n) %loop for number of harmonics

    for i = 1:1:1000 %loop one thousand times over one period
         alpha(k) = alpha(k) + (x(i)* T_s)* exp(-j * w_0 * n(k) * t(i));
    end
     alpha(k) = alpha(k) * (1/T);
end

amplitude = zeros(7,1);

for f = 1:1:7
    amplitude(f) = abs(alpha(f));
end

angle7 = zeros(7,1);

for q = 1:1:7
    angle7(q) = angle(alpha(q));
end

figure(1);
stem(n,abs(alpha))
```

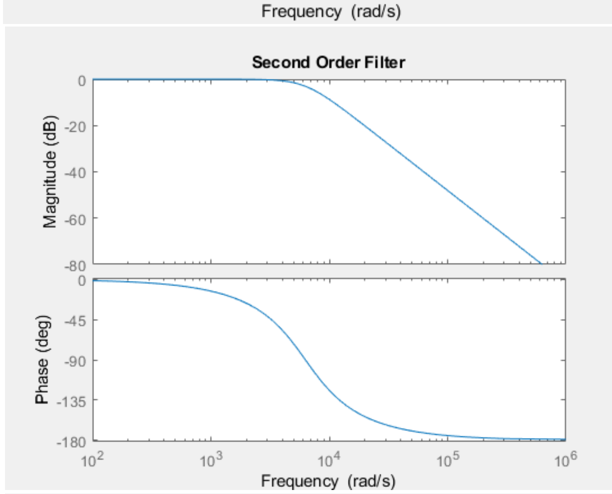
```
wc = 3175*(2*pi);
w0 = 2*pi*10000;
t = [-2e-4: 1/100000 : 2e-4];

n1 = 0.8106*cos(2*pi*10000*t + 4.336e-16);
n2 = 3.808e-6*cos(2*pi*20000*t + 2.434);
n3 = 9.007e-15*cos(2*pi*30000*t + 2.874e-15);
og60hzterm = 2*cos(2*pi*60*t);

Vin = og60hzterm + n1 + n2 + n3

%plugging fundamental frequency of n(t) into H(w)
filter_at_w0 = (wc^4)/(((j*w0)^2+j*0.7654*wc*w0+wc^2)*((j*w0)^2+j*1.8478*wc*w0+wc^2))

filterMagnitude = abs(filter_at_w0)
filterAngle = angle(filter_at_w0) *(180/pi)

vin_new2 = (Vin *filterMagnitude) + filterAngle;

figure;
plot(t,Vin);
hold on;
plot(t,vin_new2);
title('vsOut(t)')
xlabel('Time(s)')
ylabel('Volts')
```

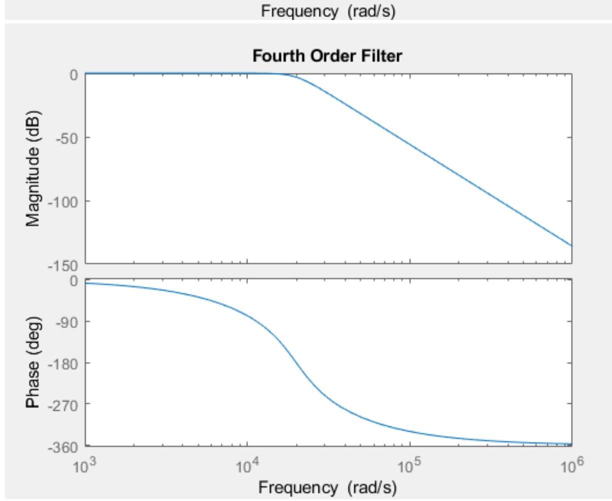


b. Given three options of filters and two constraints, the appropriate filter and corresponding corner frequency were chosen. The filters were tested within MATLAB by plugging in corner frequencies into their given transfer functions. The bode() MATLAB command was then used to analyze the performance of the filter. After plugging in corner frequency values for all filters, it was quite clear that the fourth order filter would be the most helpful in completing the task of the original problem with the smallest phase shift and a corner frequency of $2\pi 3175$ rads/sec.

```
w = [.01:.01:15000];

wc1 = (10000*2*pi)/100;
wc2 = (10000*2*pi)/10;
wc3 = (3175*2*pi);

%Filter 1
H1 = tf([1], [1/wc1 1]);
figure(1);
bode(H1);
title('First Order Filter');

%Filter 2
H2 = tf([wc2^2], [1 (2/sqrt(2)*wc2) wc2^2]);
figure(2);
bode(H2);
title('Second Order Filter');

%Filter 3
H3 = tf([wc3^4], [1 2.6132*wc3 3.41430612*wc3^2 2.6132*wc3^3 wc3^4]);
figure(3);
bode(H3);
title('Fourth Order Filter');
```





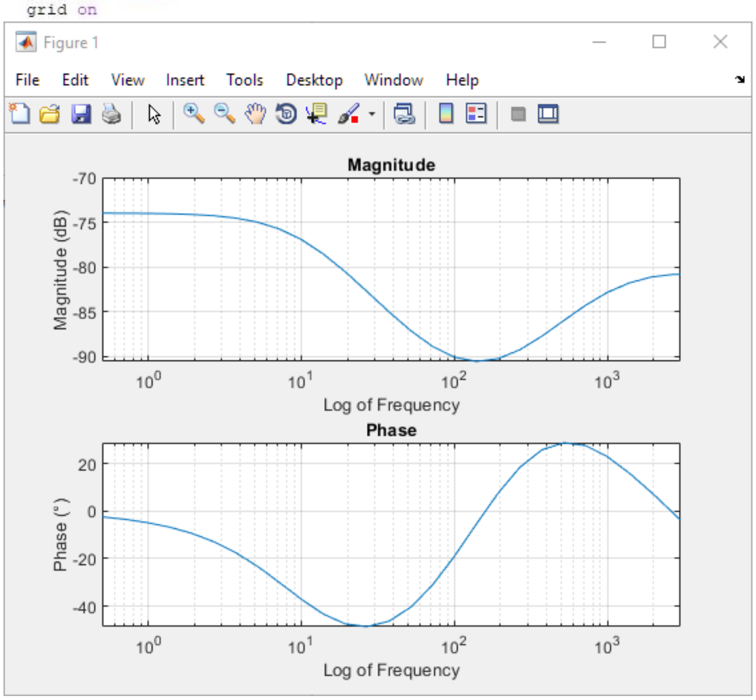


<u>Problem 3:</u> A model for a DC motor, a differential equation for its mechanical circuit components and a periodic PWM waveform were provided.

a. Referred back to Project 1, Problem 4 to reuse the differential equation of the identical DC motor's mechanical circuit.

b. Both differential equations were computed algebraically to eliminate the term of the current in transfer function orientation. The DC motor speed represented output and the voltage represented input.
c. The frequency response diagram was computed using user-defined code in MATLAB rather than the pre-defined bode function as instructed.

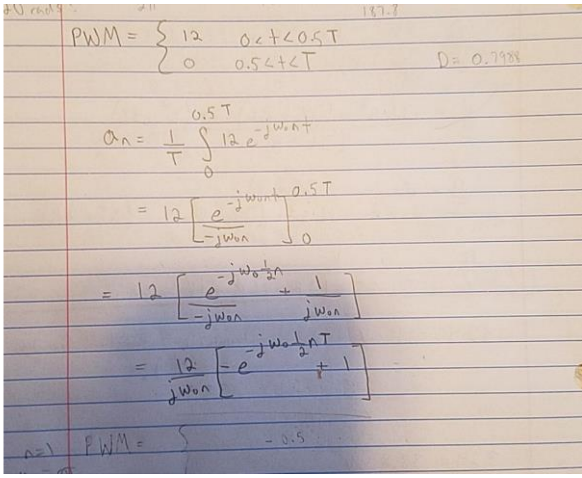
```
clc
clear all;

L = .01;
R = 3.38;
K = .029;
J = 2e-4;
B = .5e-5;

w = logspace(-3,4);

X1 = ((j.*w+100).*(j.*w+200))./((j.*w+10).*(j.*w+1000).*(j.*w+10000)); % Blue
X2 = K ./ (((J*L)*j.*w.^2).*(((J*R)+(B*L))*j.*w).*((K*K)+(B*R)));
subplot(2,1,1)
semilogx(w,20*log10(abs(X1)));
title('Magnitude')
ylabel('Magnitude (dB)')
xlabel('Log of Frequency')
set(gca, 'XLim',[0.5 3000])
% ylim([-1 10]);
grid on
subplot(2,1,2)
semilogx(w,angle(X1)*180/pi);
title('Phase')
ylabel('Phase (°)')
xlabel('Log of Frequency')
set(gca, 'XLim',[0.5 3000])
grid on
```

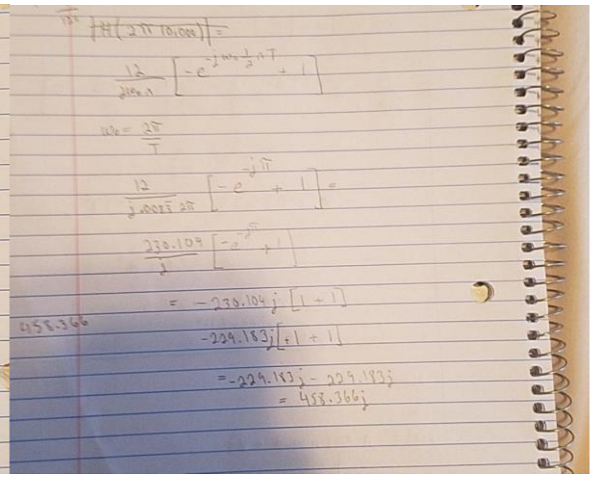


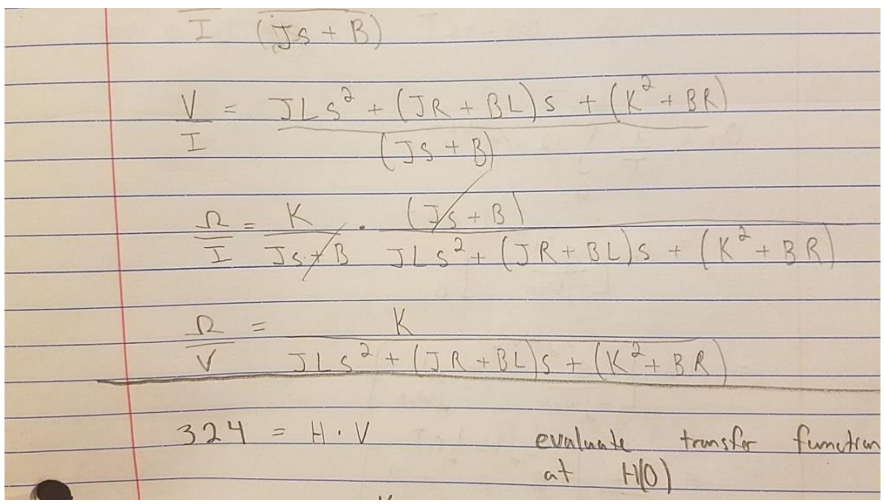
d. The frequency response plot from C was used to determine that the switching frequency should be set to 120 rad/sec, which would allow the amplitude of the signal to be about 1% of the DC value of the PWM. To determine the DC voltage required to turn the motor at 324 rad/sec, the transfer function was set to 0 frequency and solved to be 9.586V.

e. To compute the DC term of the Fourier Series as a function of D, the 9.586V was set equal to 12D, producing D = 0.7988.

f. See Below







g. The code from Project 1 and the square function (with a duty ratio of 0.7988 and an amplitude of 12) to simulate the PWM was used to simulate the motor's behavior. The average value of the speed in steady-state was 177.366, which was quite far from the approximations. The peak-to-peak value at the switching frequency was 0.9. One possible reason for the difference in the peak-to-peak variation of the switching frequency is that the prediction used a duty cycle of 0.5 but the actual PWM waveform had a duty cycle of 0.7988. Another possible explanation is a possible programming error for the MATLAB model of the DC motor.

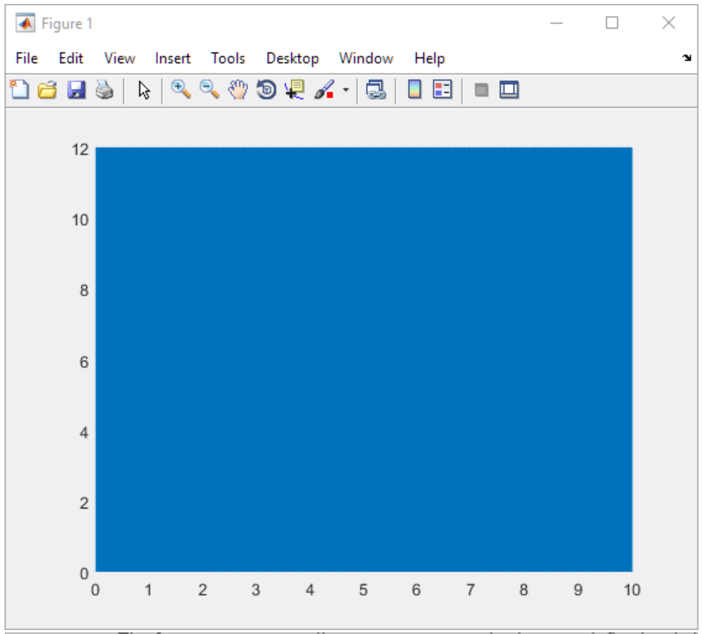


Figure 1
File Edit View Insert Tools Desktop Window Help

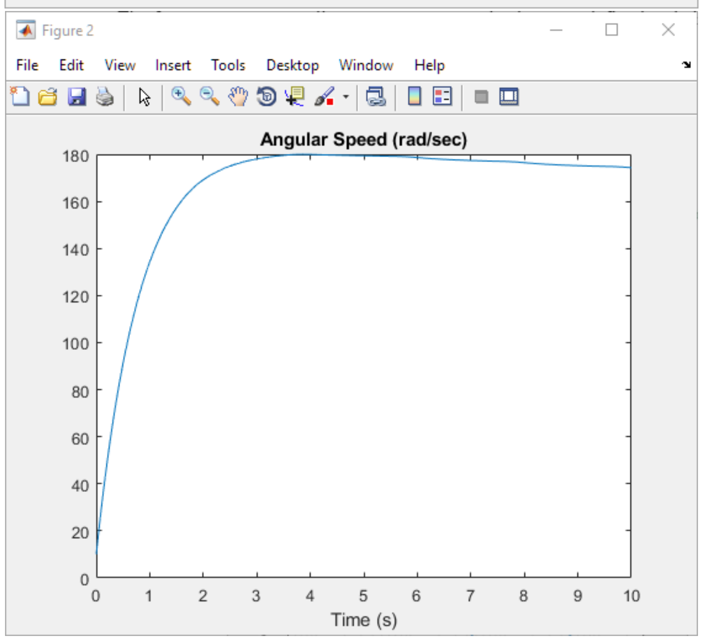


Figure 2
File Edit View Insert Tools Desktop Window Help
Angular Speed (rad/sec)
Time (s)

Editor - C:\Users\cgree\Downloads\prob3partg.m

EDITOR | PUBLISH | VIEW

asymp.m | prob3partc.m | prob3partg.m

```matlab
clc
clear all;
f = 120;
T = 1/f;
t = 0:T/10000:1200*T;
y = (6*square(2*pi*120*t,79.88) + 6);

figure(1)
plot(t,y)

K = 0.029;
J = 2e-4;
beta = 0.5e-5;
L = 0.01;
R = 3.38;
x1 = 1.5;
x2 = 0;
dt = .2/1000;
stop = round(10/dt);
t = [0:1:stop]*dt;
v_in = y;

for a = 1:1:stop

    x1(a+1) = dt*x2(a) + x1(a);
    x2(a+1) = dt*(v_in(a) - 338*x2(a) - 421*x1(a)) + x2(a);

end

omega = [10];

for b = 1:1:stop

    omega(b+1) = dt*((K*x1(b)) - (beta*omega(b)))/J + omega(b);

end

theta = [0];

for c = 1:1:stop

    theta(c+1) = dt*omega(c) + theta(c);

end



figure(2)
plot(t,omega);
title('Angular Speed (rad/sec)');
xlabel('Time (s)');

sum = 0;
for t = 20000:1:50000

    sum = sum + omega(t)

end

average = sum/30000;
```

script | Ln 1 | Col 4

h. To attenuate the amplitude of the speed to 0.01% of the DC value, a switching frequency of 12000 rad/sec was chosen because it is a factor of 100 different than the previous switching frequency for the 1% value at 120 rad/sec.

Problem 4: A sample signal with three different sample periods was provided. DFT was introduced and used to find Fourier Transforms for given signals.

a. Using subplot, 3 graphs were shown for each sample period given, plotted over one period each.

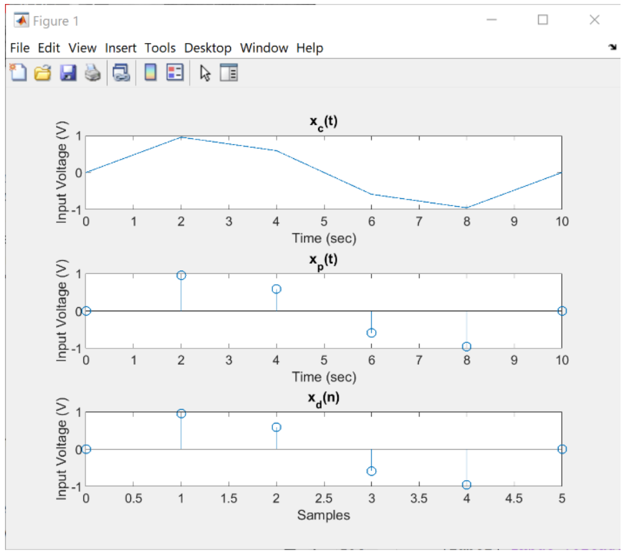


Figure 1
File Edit View Insert Tools Desktop Window Help
$x_c(t)$
Input Voltage (V)
Time (sec)
$x_p(t)$
Input Voltage (V)
Time (sec)
$x_d(n)$
Input Voltage (V)
Samples

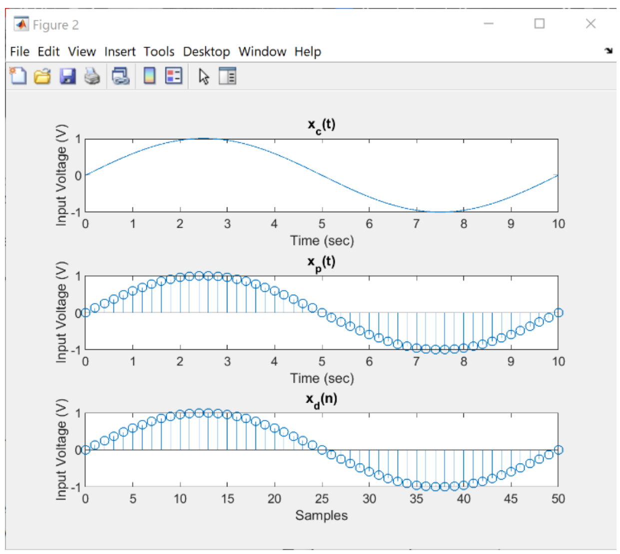


Figure 2
File Edit View Insert Tools Desktop Window Help
$x_c(t)$
Input Voltage (V)
Time (sec)
$x_p(t)$
Input Voltage (V)
Time (sec)
$x_d(n)$
Input Voltage (V)
Samples

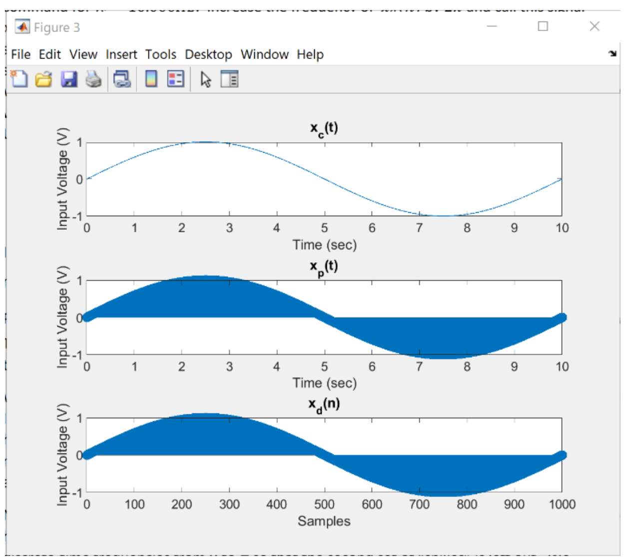
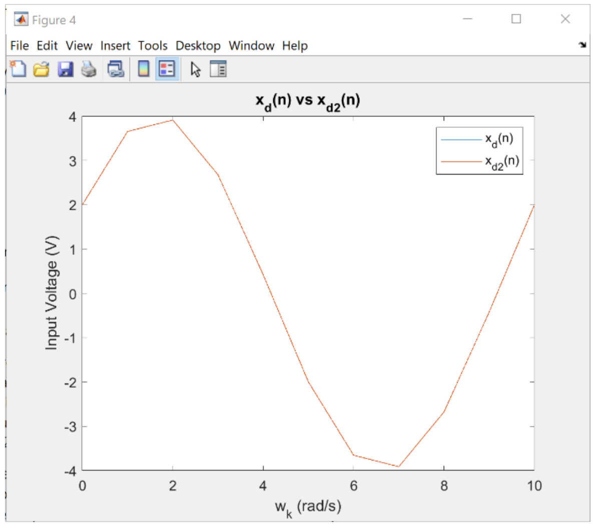


Figure 3
File Edit View Insert Tools Desktop Window Help
$x_c(t)$
Input Voltage (V)
Time (sec)
$x_p(t)$
Input Voltage (V)
Time (sec)
$x_d(n)$
Input Voltage (V)
Samples

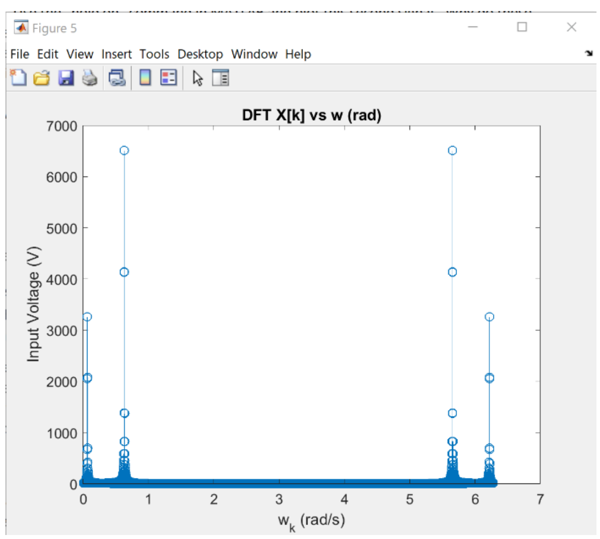
b. $N_1 = 10$, $omega_{k1} = 6.2832 * 10^{-4}$ rad/s, $N_2 = 100$, $omega_{k2} = 6.2832 * 10^{-5}$rad/s

c. For both signals, the frequency is the same, but the graphs end up resting on top of each other because increasing the frequency by 2π only causes it to be another period ahead of the signal. This causes the graphs to look the same as both are sinusoidal graphs that have a fundamental period of 2π.

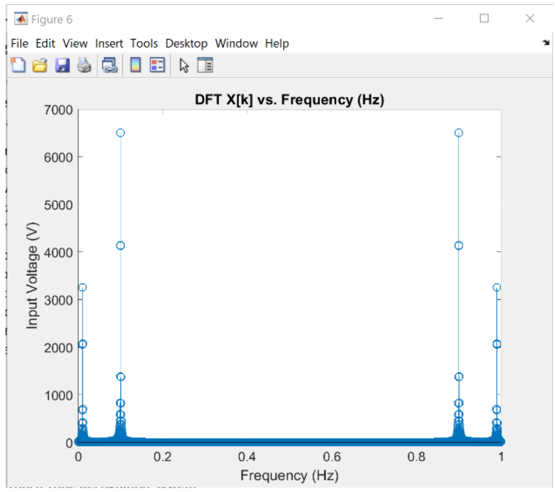


d. SEE IN CODE.

e. The relative values of the spikes for each wave make sense because at some point the input voltage will spike due to the input signal being real, thus being conjugate-symmetric within the frequency domain. If the input signal was imaginary, only one spike would be seen per wave.



f. Plotted for frequency vs DFT X[k].




```
clear;

T = 10;
T_s1 = 2;
T_s2 = .2;
T_s3 = .01;

f = 1/T;
f_s1 = 1/T_s1;
f_s2 = 1/T_s2;
f_s3 = 1/T_s3;

n1 = [0:1:(1/(f/f_s1))];
n2 = [0:1:(1/(f/f_s2))];
n3 = [0:1:(1/(f/f_s3))];

t1 = n1*T_s1;
t2 = n2*T_s2;
t3 = n3*T_s3;

x_c1 = sin(2*pi()*f*t1);
x_d1 = sin(2*pi()*(f/f_s1)*n1);

x_c2 = sin(2*pi()*f*t2);
x_d2 = sin(2*pi()*(f/f_s2)*n2);

x_c3 = sin(2*pi()*f*t3);
x_d3 = sin(2*pi()*(f/f_s3)*n3);
```

```
figure;
subplot(3,1,1);
plot(t1,x_c1);
title('x_c(t)');
hold on;
xlabel('Time (sec)');
ylabel('Input Voltage (V)');
hold off;
subplot(3,1,2);
stem(t1,x_c1);
title('x_p(t)');
hold on;
xlabel('Time (sec)');
ylabel('Input Voltage (V)');
hold off;
subplot(3,1,3);
stem(n1,x_d1);
title('x_d(n)');
hold on;
xlabel('Samples');
ylabel('Input Voltage (V)');
hold off;
```

```
figure;
subplot(3,1,1);
plot(t2,x_c2);
title('x_c(t)');
hold on;
xlabel('Time (sec)');
ylabel('Input Voltage (V)');
hold off;
subplot(3,1,2);
stem(t2,x_c2);
title('x_p(t)');
hold on;
xlabel('Time (sec)');
ylabel('Input Voltage (V)');
hold off;
subplot(3,1,3);
stem(n2,x_d2);
title('x_d(n)');
hold on;
xlabel('Samples');
ylabel('Input Voltage (V)');
hold off;
```

```
figure;
subplot(3,1,1);
plot(t3,x_c3);
title('x_c(t)');
hold on;
xlabel('Time (sec)');
ylabel('Input Voltage (V)');
hold off;
subplot(3,1,2);
stem(t3,x_c3);
title('x_p(t)');
hold on;
xlabel('Time (sec)');
ylabel('Input Voltage (V)');
hold off;
subplot(3,1,3);
stem(n3,x_d3);
title('x_d(n)');
hold on;
xlabel('Samples');
ylabel('Input Voltage (V)');
hold off;
```

```
f1 = 1000;
f2 = 100;
fs1 = 10e3;
fs2 = 100e3;
omegak1 = (2*pi())/fs1;
omegak2 = (2*pi())/fs2;
ns1 = [0:1:(1/(f1/fs1))];
ns2 = [0:1:(1/(f1/fs2))];
phase1 = 30*(pi()/180);
phase2 = 20*(pi()/180);

xd1 = 4*sin((2*pi()*(f1/fs1)*ns1) + phase1) + 2*cos((2*pi()*(f2/fs1)*ns1) + phase2);
xd2 = 4*sin((2*pi()*(f1/fs2)*ns2) + phase1) + 2*cos((2*pi()*(f2/fs2)*ns2) + phase2);

xd3 = 4*sin((2*pi()*(f1/fs1)*ns1) + phase1);
xd4 = 4*sin((2*pi()*(f1/fs1)*ns1) + phase1 + (2*pi()));

figure
plot(ns1, xd3);
hold on;
plot(ns1, xd4);
hold off;
title('x_d(n) vs x_d_2(n)');
legend('x_d(n)','x_d_2(n)');
ylabel('Input Voltage (V)');
xlabel('w_k (rad/s)');


n = [0:1:39999];

xd5 = 4*sin((2*pi()*(f1/fs1)*n) + phase1) + 2*cos((2*pi()*(f2/fs1)*n) + phase2);

dft(xd5,6500);
```

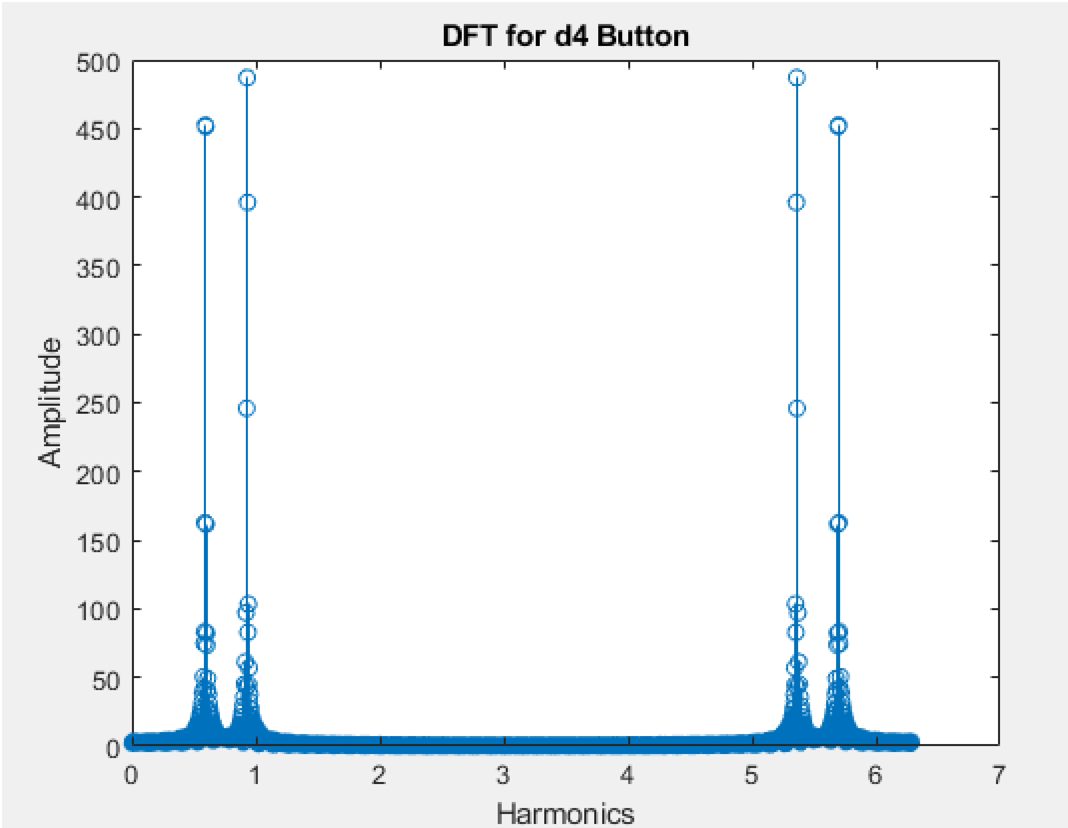
```
function d = dft(din,N_p)
    dx = [din zeros(1,N_p-1000)];
    X = zeros(1,N_p);
    for k = 1:1:N_p
        for L = 1:1:N_p/2
            X(k) = X(k)+(dx(L)*exp((-j)*(2 * pi / N_p)*(L-1)*(k-1)));
        end
    end
    K = 1:1:N_p;
    wk = 2*pi*K / N_p;
    figure;
    stem(wk,abs(X))
    title('DFT X[k] vs w (rad)')
    ylabel('Input Voltage (V)')
    xlabel('w_k (rad/s)')
    d = X;

    freqk = wk/(2*pi());

    figure;
    stem(freqk,abs(X))
    title('DFT X[k] vs. Frequency (Hz)')
    ylabel('Input Voltage (V)')
    xlabel('Frequency (Hz)')

end
```

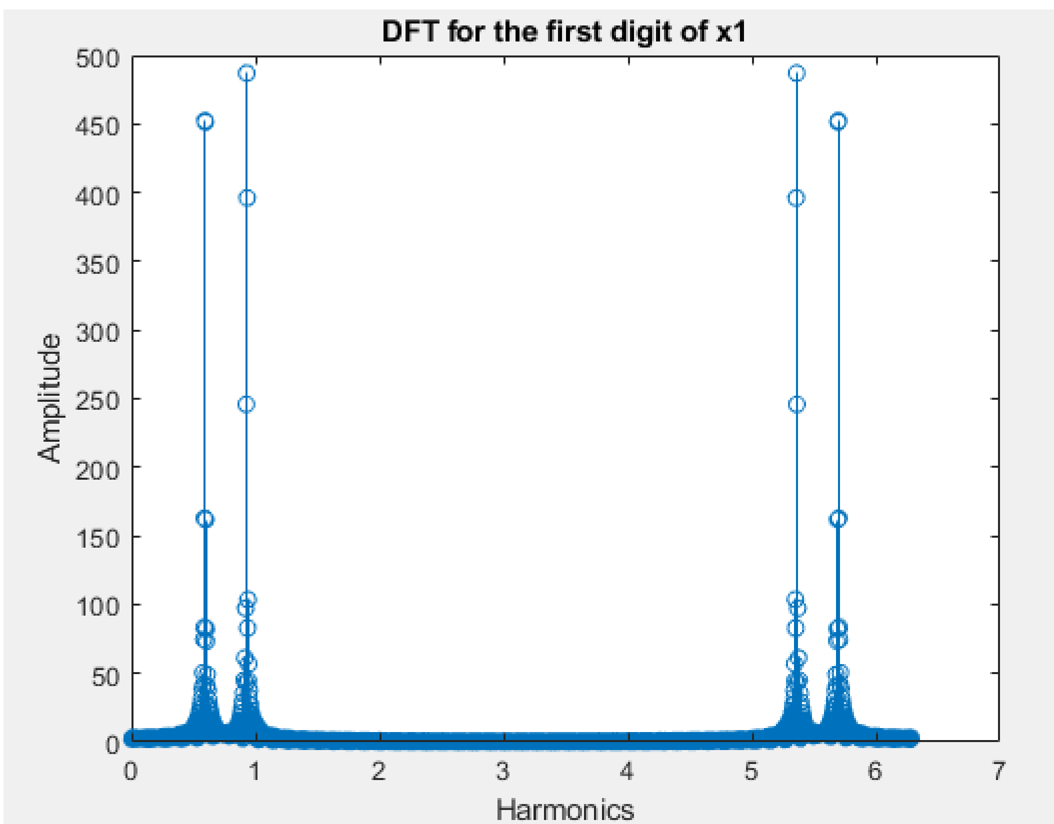
Problem 5: A vector consisting of seven digits of a mock phone number with each digit being 1000 samples separated by vectors of space was given.

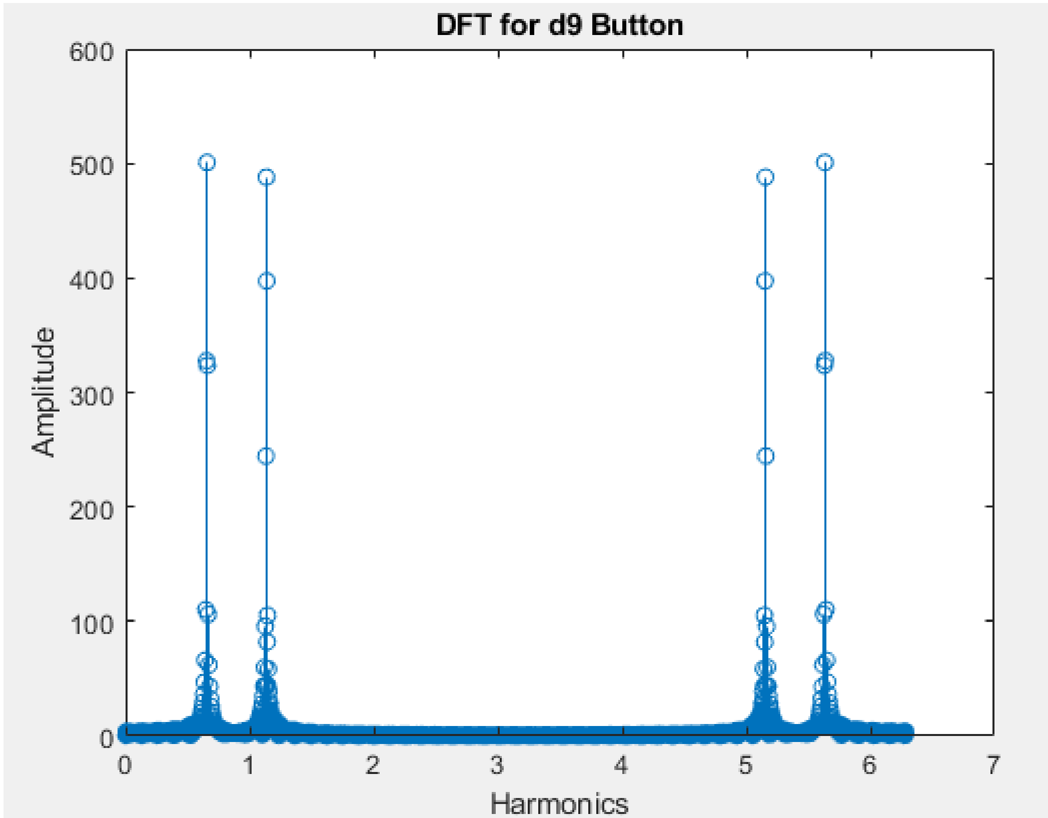
a. To represent the digits 0-9, ten vectors were created, each consisting of a sum of two sine waves with differing frequencies between each digit. The first sinusoidal frequency is the row of the touchpad, the second frequency is the column of the touchpad.
b. Sound(d3,fs) was used to play the frequency for #3, using d3 as the signal and fs as the sample frequency of 8192 Hz.
c. A zero-vector called space was used to delay the signals in between playing each tone, adding the effect of someone dialing a telephone. The space vector can easily be lengthened or shortened to increase or decrease the dial tone spacing.
d. Each digit was analyzed using a double nested loop in order to represent their digital Fourier Transforms. This resulted in plotting using stem to identify each digit by plot.
e. The touch.mat file supplied was loaded into the script in MATLAB and each digit was isolated and had its DFT graphed to compare it to the known digital signals from the given dial tone frequencies. By analyzing each number individually, we were able to cut down on inaccuracy by confusing plots and were able to come up with the correct phone number 491-5877, which correctly adds up to 41 as expected.

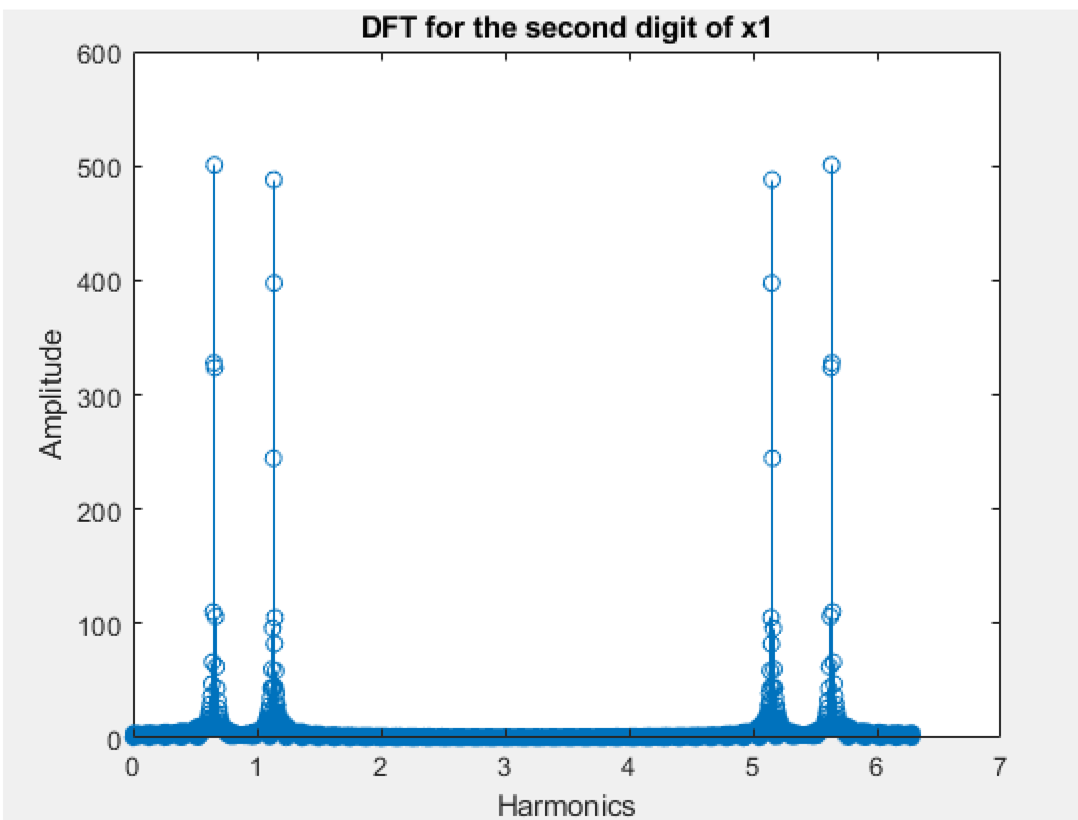
```
clc
clear all;
Fs = 8192;
T = 1000;
n = [0:1:999];
d0 = sin(((2*pi*941)/8192)*n)+sin(((2*pi*1336)/8192)*n);
d1 = sin(((2*pi*697)/8192)*n)+sin(((2*pi*1209)/8192)*n);
d2 = sin(((2*pi*697)/8192)*n)+sin(((2*pi*1336)/8192)*n);
d3 = sin(((2*pi*697)/8192)*n)+sin(((2*pi*1477)/8192)*n);
d4 = sin(((2*pi*770)/8192)*n)+sin(((2*pi*1209)/8192)*n);
d5 = sin(((2*pi*770)/8192)*n)+sin(((2*pi*1336)/8192)*n);
d6 = sin(((2*pi*770)/8192)*n)+sin(((2*pi*1477)/8192)*n);
d7 = sin(((2*pi*852)/8192)*n)+sin(((2*pi*1209)/8192)*n);
d8 = sin(((2*pi*852)/8192)*n)+sin(((2*pi*1336)/8192)*n);
d9 = sin(((2*pi*852)/8192)*n)+sin(((2*pi*1477)/8192)*n);
sound(d2,8192)
space = zeros(size(100));
x = [d8 space d4 space d5 space d6 space d2 space d5 space d9 space d4 space d7 space d8];
sound(x,8192)
load('touch.mat')
```

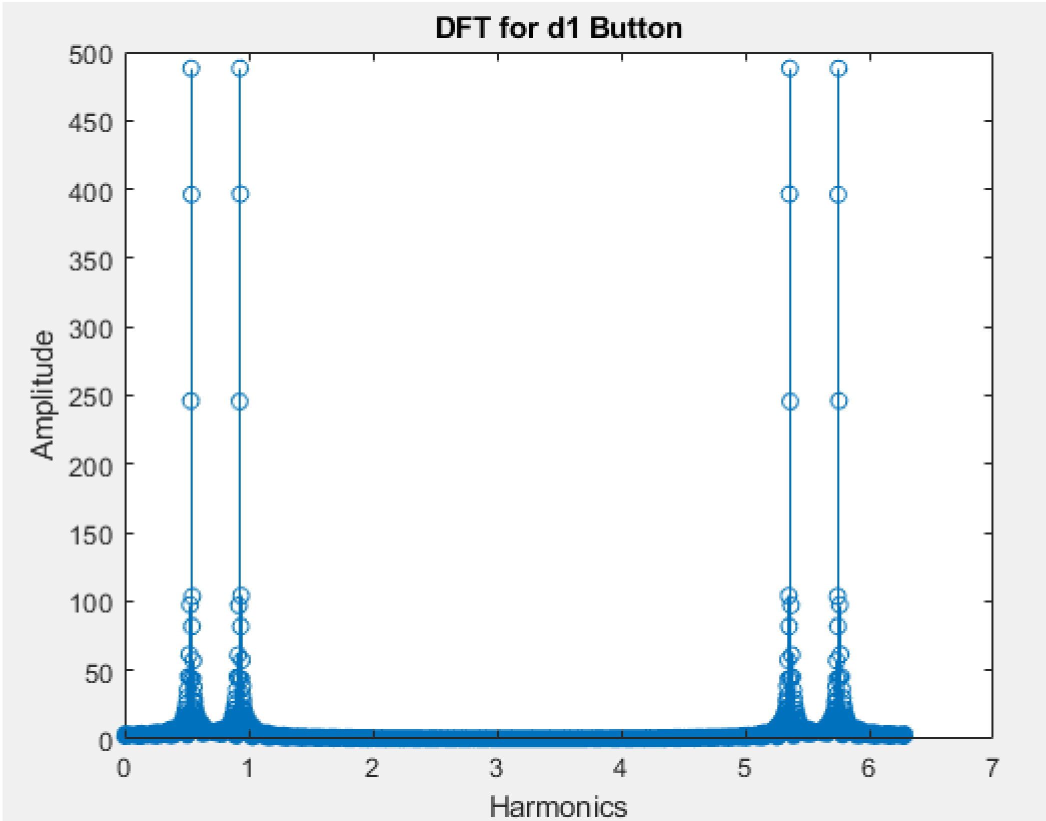
```
N_p = 2048;
d4 = [d4 zeros(1,1048)];
for k = 1:1:2048
    X(k) = 0;
    for i = 1:1:2048
        X(k) = X(k)+(d4(i)*exp((-j)*(2 * pi / N_p)*(i-1)*(k-1)));
    end
end
K = 1:1:N_p;
WK = 2*pi*K / N_p;
figure(1);
%stem(WK,abs(X))
title('DFT for d4')
%Evaluates DFT of x1 for number 7
for q = 1:1:2048
    Y(q) = 0;
    for m = 1:1:1000
        Y(q) = Y(q)+(x1(m)*exp((-j)*(2 * pi / N_p)*(m-1)*(q-1)));
    end
end
figure(2)
stem(WK,abs(Y))
title('DFT for the fourth digit of x1')
```

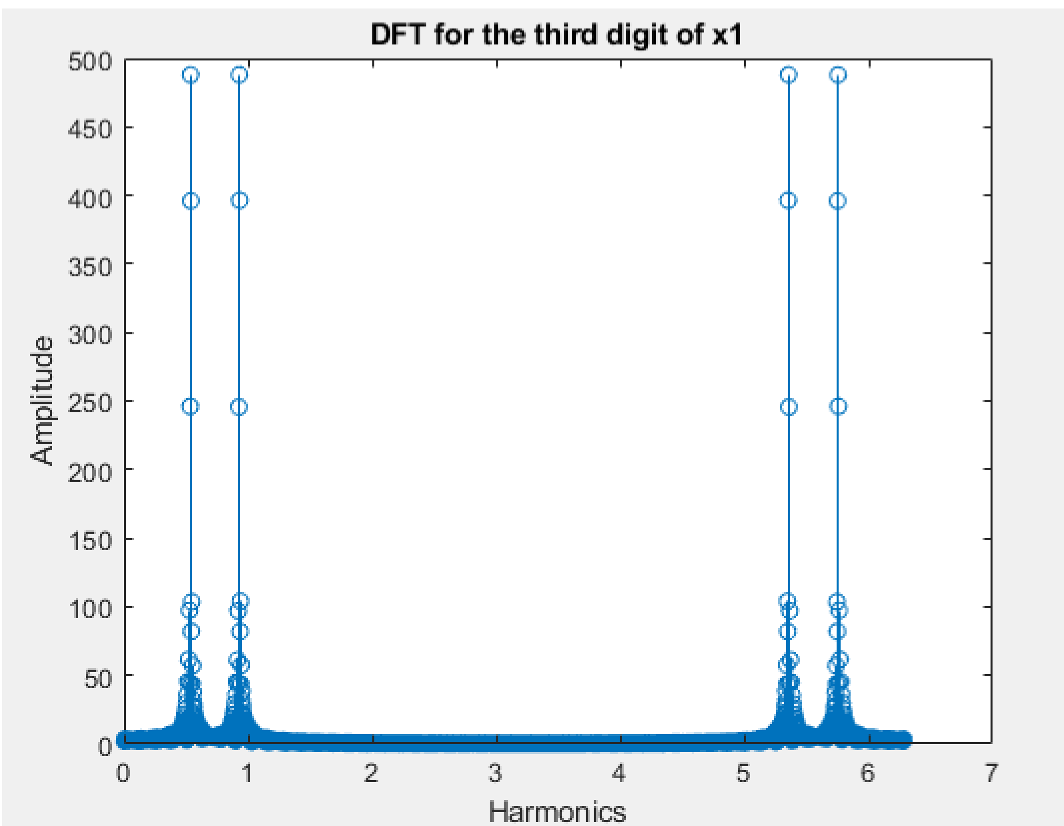

DFT for d4 Button
Amplitude
Harmonics
0
50
100
150
200
250
300
350
400
450
500
0
1
2
3
4
5
6
7

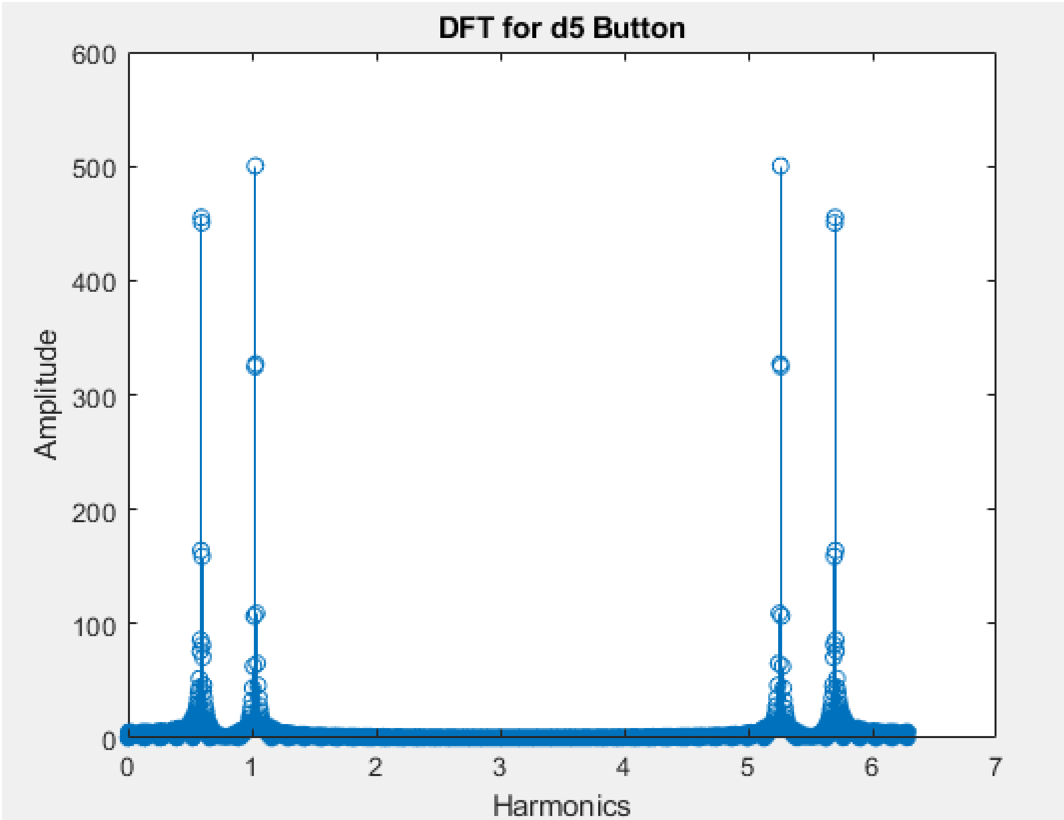

DFT for the first digit of x1
Amplitude
Harmonics
0
50
100
150
200
250
300
350
400
450
500
0
1
2
3
4
5
6
7

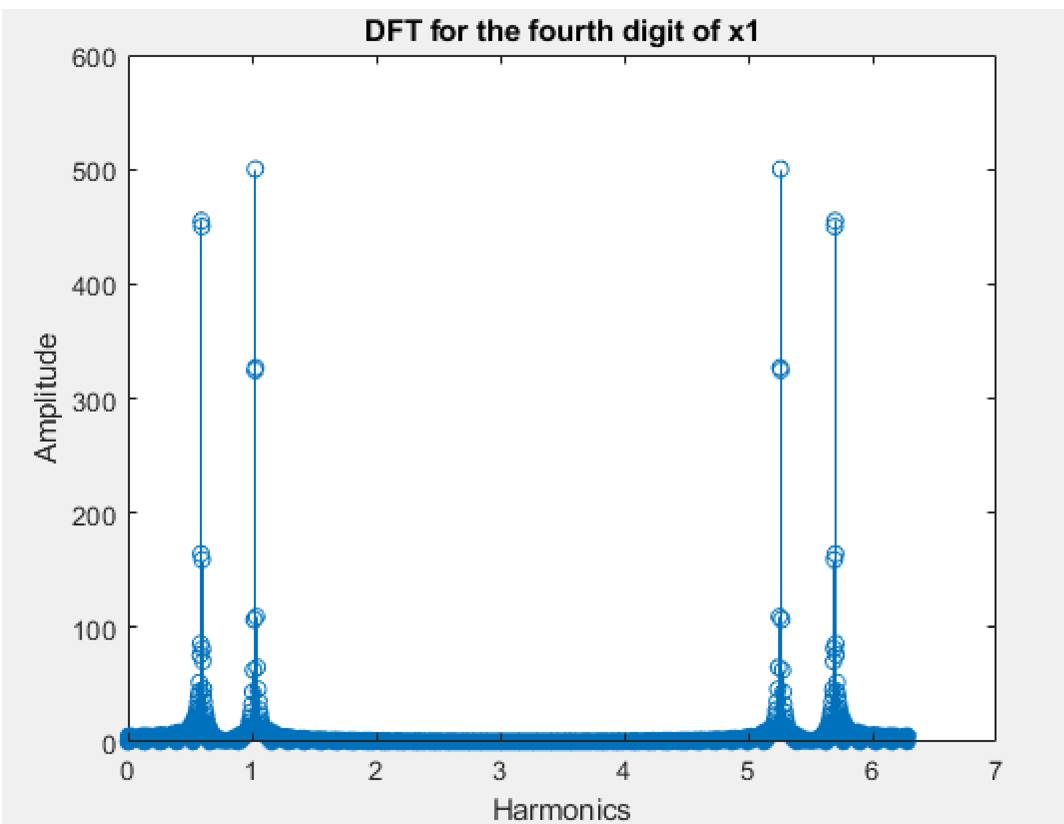

DFT for d9 Button
Amplitude
Harmonics
600
500
400
300
200
100
0
0
1
2
3
4
5
6
7

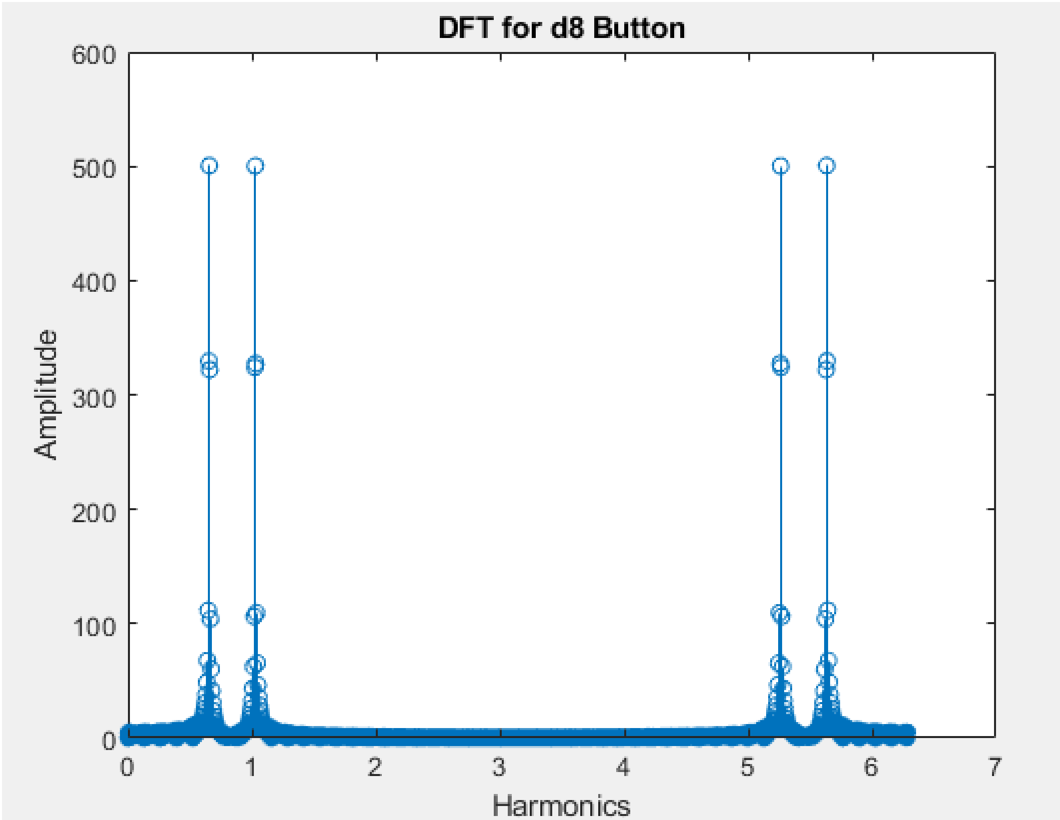

DFT for the second digit of x1
Amplitude
Harmonics
600
500
400
300
200
100
0
0
1
2
3
4
5
6
7

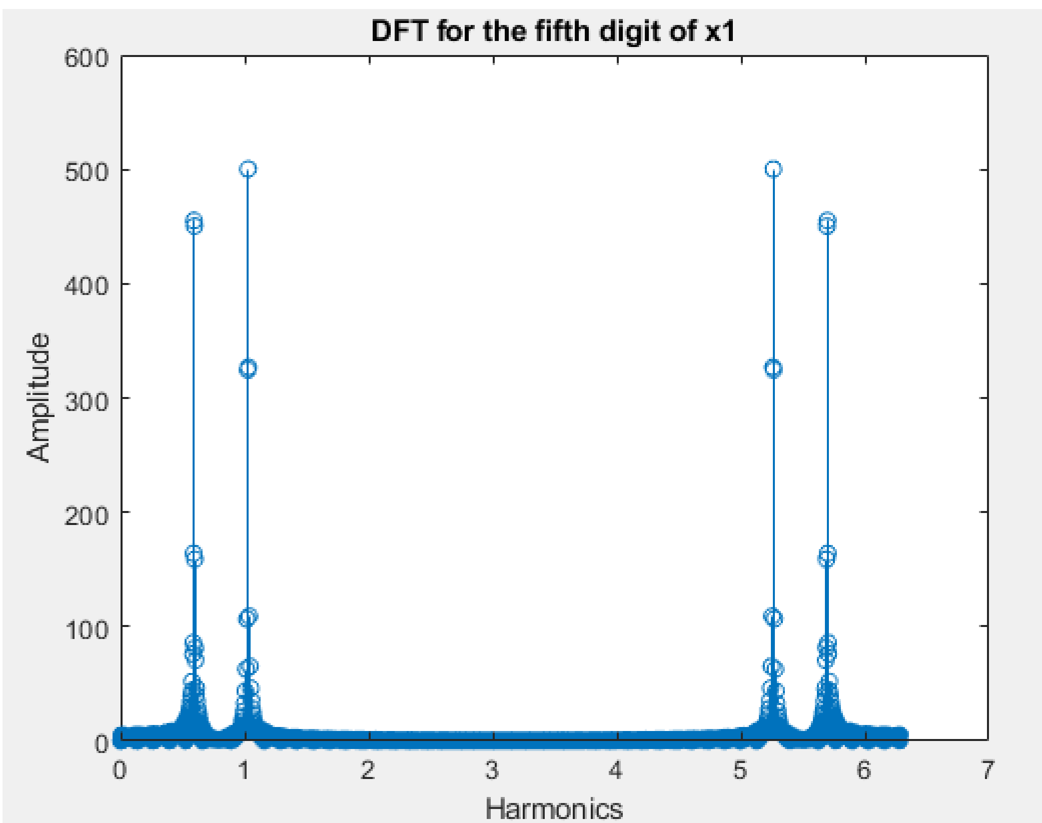

DFT for d1 Button
Amplitude
Harmonics
0
50
100
150
200
250
300
350
400
450
500
0
1
2
3
4
5
6
7

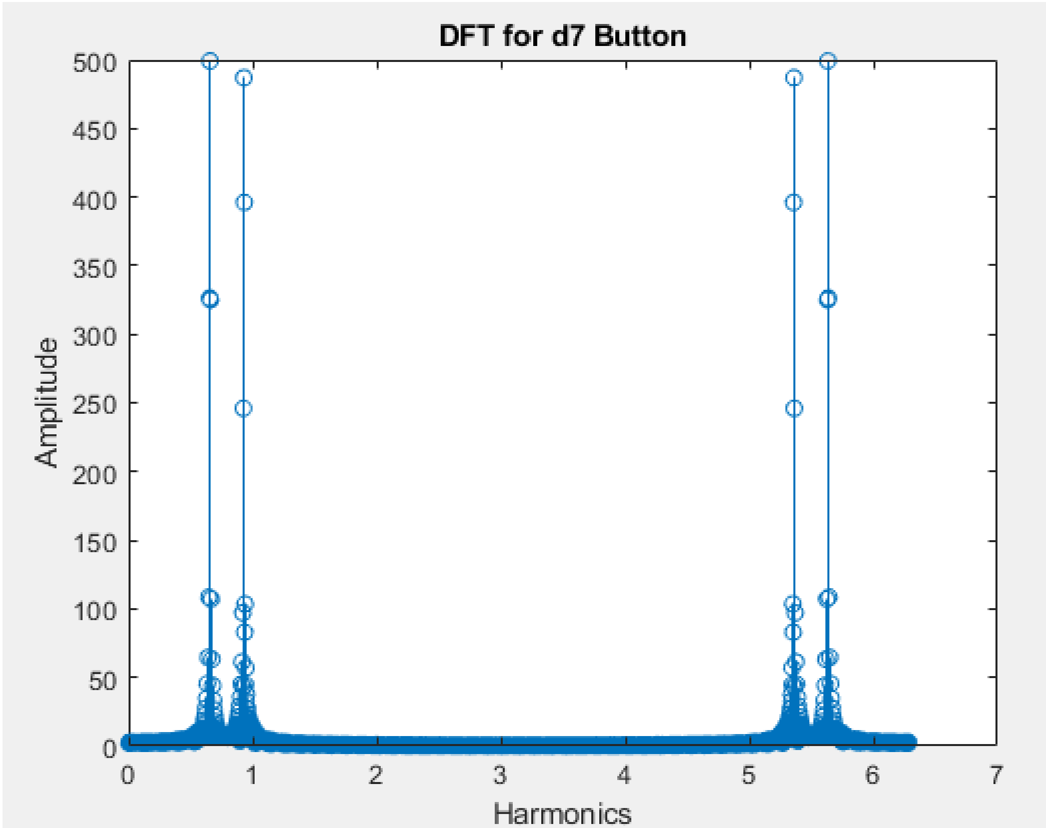

DFT for the third digit of x1
Amplitude
Harmonics
0
50
100
150
200
250
300
350
400
450
500
0
1
2
3
4
5
6
7

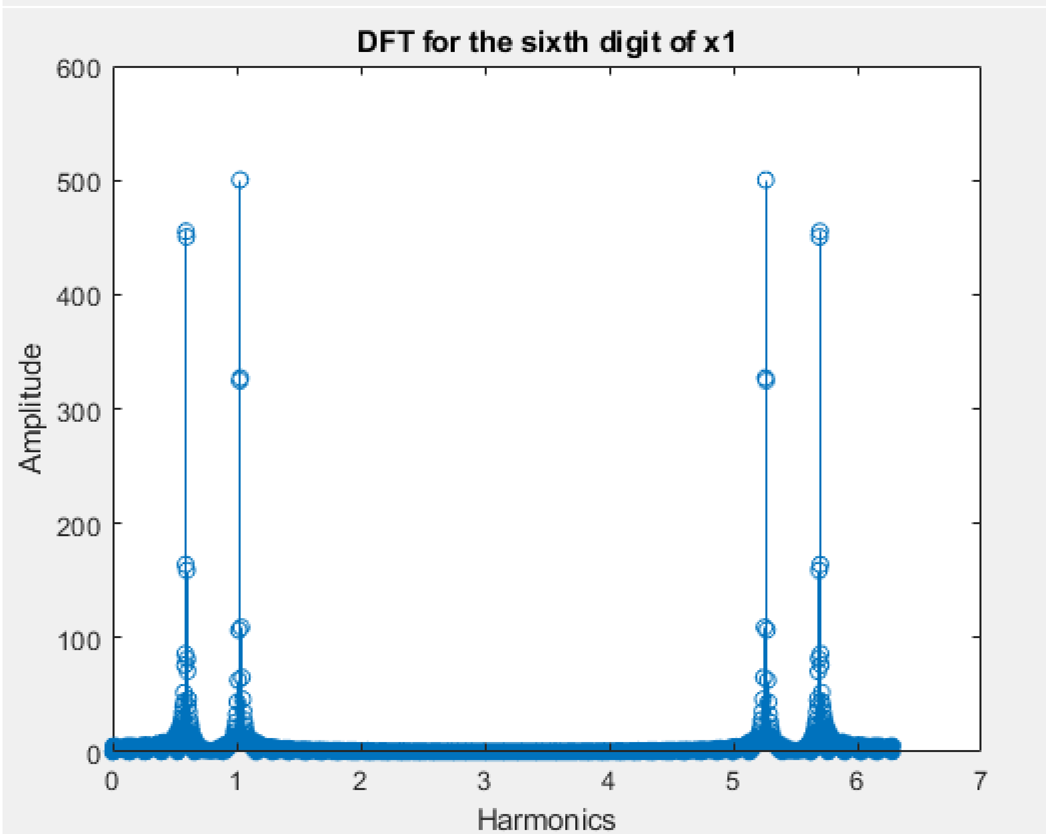

DFT for d5 Button
Amplitude
Harmonics
0
100
200
300
400
500
600
0
1
2
3
4
5
6
7

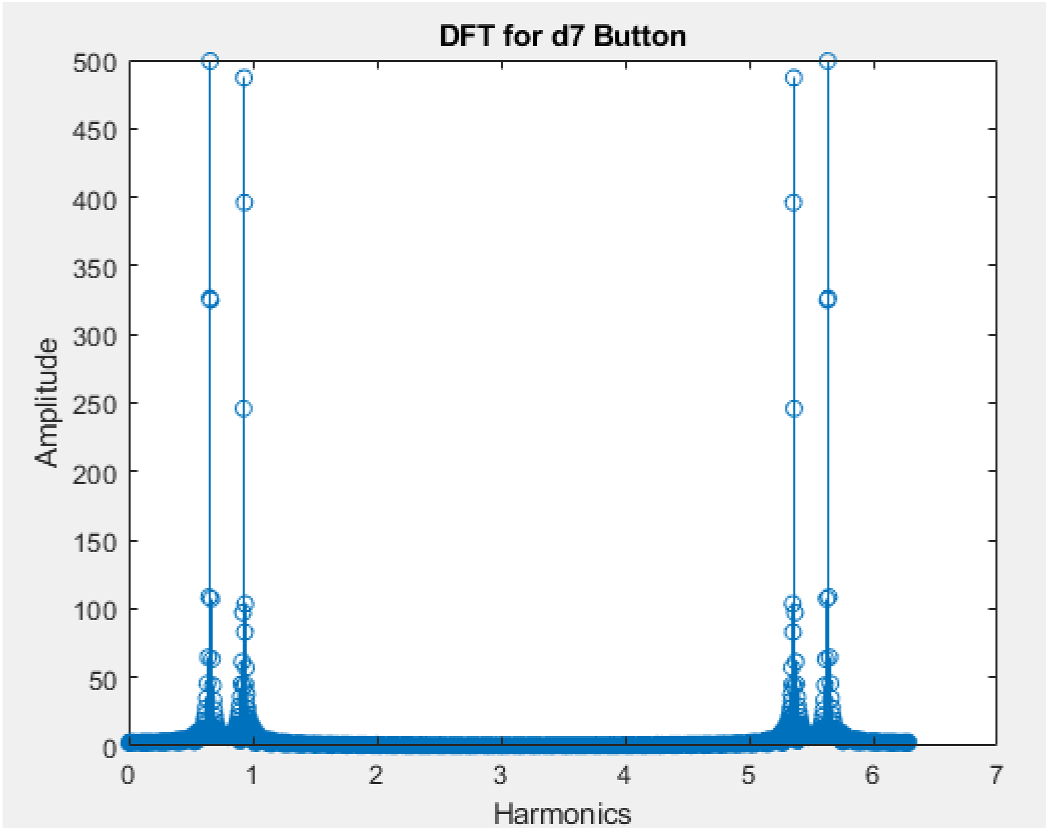

DFT for the fourth digit of x1
Amplitude
Harmonics
0
100
200
300
400
500
600
0
1
2
3
4
5
6
7


DFT for d8 Button
Amplitude
Harmonics
600
500
400
300
200
100
0
0
1
2
3
4
5
6
7

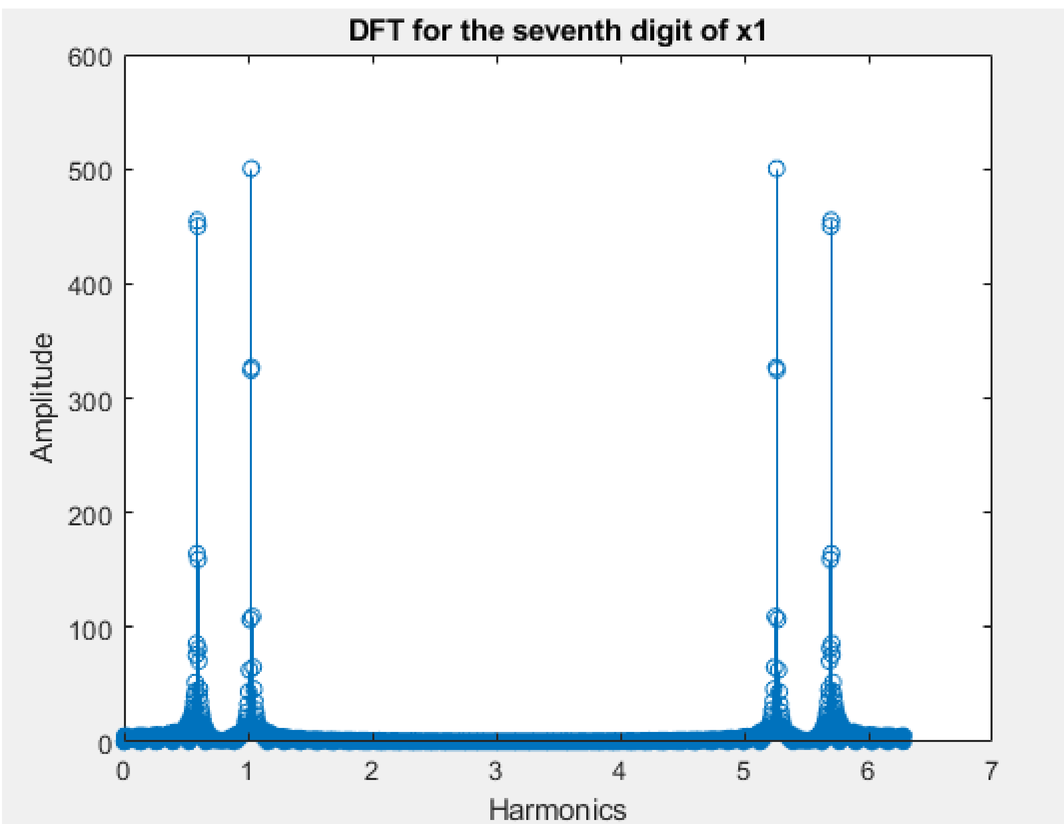

DFT for the fifth digit of x1
Amplitude
Harmonics
600
500
400
300
200
100
0
0
1
2
3
4
5
6
7


DFT for d7 Button
Amplitude
Harmonics
0
50
100
150
200
250
300
350
400
450
500
0
1
2
3
4
5
6
7


DFT for the sixth digit of x1
Amplitude
Harmonics
0
100
200
300
400
500
600
0
1
2
3
4
5
6
7



DFT for d7 Button
Amplitude
Harmonics
0
50
100
150
200
250
300
350
400
450
500
1
2
3
4
5
6
7



DFT for the seventh digit of x1
Amplitude
Harmonics
0
100
200
300
400
500
600
1
2
3
4
5
6
7

For each pair of graphs above, the first DFT graph shows the signal of the number and corresponds it to the matching frequency from the phone number within touch.mat.