Project 2

By: Tasha Asyiqin, Mikael Pineda, & Diego Rufino

Problem 1 – A PWM Motor Drive

1a. By doing KVL for the electrical circuit, it resulted in the following differential equation:

$$-V_A + V_R + V_L + k\Omega = 0$$

When organized, it results in the final differential equation for the electrical circuit:

$$V_a = I_a R + L\frac{di_a}{dt} + k\Omega$$

1b. Using the mechanical and electrical equation and fourier transform formulas, the following transfer function was found:

$$H(w) = \frac{\Omega(w)}{V_a(w)} = \frac{K}{RJjw + \beta R + LJ(jw)^2 + L\beta jw + k^2}$$

1c. The following MATLAB code was done to find the magnitude of the transfer function:

```
%% part c (plotting the magnitude of the transfer function)
omega = logspace(-2,4,10000); %% c.a.

L = 0.01;
R = 3.38;
K = 0.029;
J = 2e-4;
beta = 0.5e-5;

s = j.*omega;
```

```
%% vector for H(w)

tau = L/R; %% for an RL circuit
H = K ./ ((R*J*s)+(beta*R)+(L*J*(s).^2)+(L*beta*s)+(K).^2);

figure(1);
semilogx(omega,20*log10(abs(H))); %% c.c.

title('Transfer Function Magnitude');
xlabel('Frequency');
ylabel('Magnitude');
```

1d. Finding the area of the square wave allowed to determine the DC value via duty ratio by doing the following:

$$V_a(0) = \alpha_0 = \frac{1}{T}\int_0^T V_a(t)dt = \frac{1}{T}(12)(DT) = 12D$$

1e. The following formula is used and converted to find average speed (x = 0):

$$\Omega(0) = H(0)V_a(0)$$

The transfer function was found to be 30.6 by looking at w(0) in the plot and making x-value, the frequency, 0. Omega was given as 250, so now V_a(0) had to be found.

$$250 = 30.6 * V_a(0)$$

$$V_a(0) = 8.17$$

1f. In part D, the DC value was already found as 12D, so inputting what V(0) was found was used to find the duty ratio.

$$D = \frac{V(0)}{12} = \frac{8.17}{12} = 0.68$$

1g. To find the amplitude of the frequency, a piecewise function is done to find the input voltage signal at the square wave:

$$V_a(t) = 12, 0 \leq t \leq DT$$

$$V_a(t) = 0, DT < t \leq T$$

This results in the integral to find the value of the fundamental frequency is written as shown:

$$\alpha_1 = \frac{1}{T}\int_0^T V_a(t)e^{-j\omega_0 t}dt = \frac{1}{T}\int_0^{DT} 12e^{-j\omega_0 t}dt = \frac{12\left(e^{-j\omega_0 DT} - 1\right)}{-j\omega_0 T}$$

Implementing Tw_0 = 2pi from T = 2pi/w_0 results in:

$$\alpha_1 = \frac{12\left(e^{-j\omega_0 D} - 1\right)}{-j\omega_0 DT} = \frac{12\left(e^{-j2\pi D} - 1\right)}{-j2\pi}$$

D was found in part f as 0.68, so it was found to be:

$$\alpha_1 = -1.728 - 2.723j$$

By doing C_n = 2|α_n|, the result is:

$$C_1 = 6.45$$

1h. The formula is given to find the frequency so that the amplitude of the speed variation at the fundamental frequency will be less than 1% of the average speed:

$$2|\Omega(\omega_0)| < 0.01|\Omega(0)|$$

When plugging the average speed, 250, it results in:

$$2|\Omega(\omega_0)| < 2.5$$

$$|H(\omega_0)| < 1.25$$

Doing the complex function after finding average speed at 0 results in:

$$|H(\omega_0)| < \frac{1.25}{|V_a(\omega_0)|}$$

Implementing the alpha magnitude of V_a is 3.225. This results in:

$$|H(\omega_0)| < \frac{1.25}{3.225} = 0.388$$

When converted into decibels, it results in:

$$|H(\omega_0)| < -8.223$$

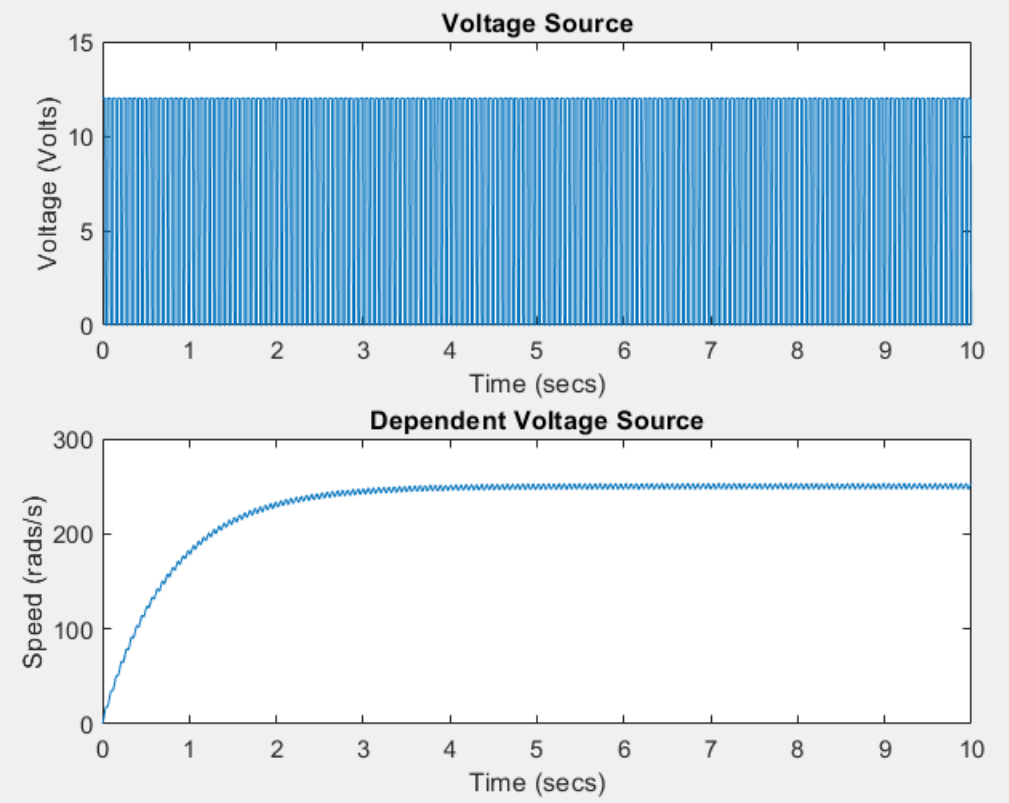
Looking at the Bode plot, choosing 106 results in a transfer function magnitude closest to the range. This means w_0 = 106.

$$H(106) = 0.338$$

1i. The square wave waveform that was defined involves the code for simulating the motor response:

```
%% part i
sw = 106; % switching frequency
Tsw = 2*pi/sw; % period

delta_t = Tsw/10000;
t = [0:delta_t:10];

Va = 6*square(sw*t, 61.63) + 6;

omega = [];
I_a = [];

for n = 1:1:length(t)-1
    if n == 1
        omega(n) = 0;
        I_a(n) = 0;
    end
        omega(n+1) = (delta_t/J)*(K*I_a(n) - beta*omega(n)) + omega(n);
        I_a(n+1) = (delta_t/L) *(Va(n) - K*omega(n) - R*I_a(n)) + I_a(n);
end

averageval = mean(omega(end - 10000:end));
p2pval = peak2peak(omega(end - 10000:end));

subplot(2, 1, 1);
plot(t, Va);
xlabel('Time (secs)');
ylabel('Voltage (Volts)');
title('Voltage Source');

subplot(2, 1, 2);
plot(t, omega);
xlabel('Time (secs)');
ylabel('Speed (rads/s)');
title('Dependent Voltage Source');
```

The following plots were found:

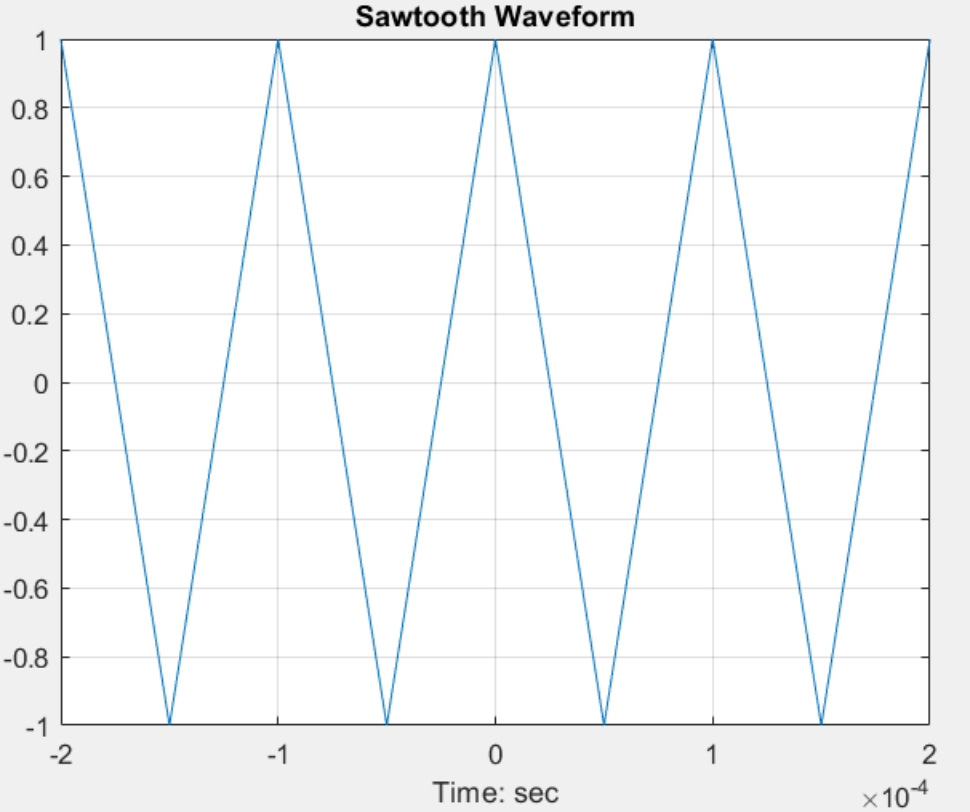


Problem 2 – Filtering a Signal for Control

2a. The signal was found in the following code:

```
%% part a
T = 10^-4; %% period, a.a
f = 1/T;
w = (2*pi*f);
delta_t = T/10000;
t = [-2*T:delta_t:2*T]; %% time step, a.c
x = -sawtooth(2*pi/T*t, 0.5); %%% sawtooth command & phase, a.b & a.d

plot(t, x);
figure(1);
title('Sawtooth Waveform');
xlabel('Time: sec');
grid on;
```

This resulted in the following sawtooth signal:



Then, finding the corresponding signals was done in the following code:

```
%% determine the corresponding signals, part a
alpha_1 = 0;
alpha_2 = 0;
alpha_3 = 0;
for n = 1:1:10000
     alpha_1 = alpha_1 + x(n)*exp(-j*1*w*t(n))*delta_t; %% create the waveform x
     alpha_2 = alpha_2 + x(n)*exp(-j*2*w*t(n))*delta_t;
     alpha_3 = alpha_3 + x(n)*exp(-j*3*w*t(n))*delta_t;
end

alpha_1 = alpha_1/T;
alpha_2 = alpha_2/T;
alpha_3 = alpha_3/T;
abs(alpha_1);
abs(alpha_2);
abs(alpha_3);
```

The values were found to be the following:

alpha_1	0.4053 + 0.0000i
alpha_2	5.0193e-17 - 8.7637e-17i
alpha_3	0.0450 + 0.0000i

2b. The constraints were written in the following code:

```
%% part b
h1w = @(wc,w) 1/(1+((j*w)/wc)); %% h1w takes in wc and w and outputs H1 @ omega
h2w = @(wc, w) (wc)^2./( (s).^2 + (s .* wc .* ( 2/sqrt(2) ) ) + (wc)^2 ); %% h2w takes in wc and w and outputs H2 @ omega
h4w = @(wc, w) (wc)^4./( ((s).^2 + (s .* wc .* 0.7654) + (wc)^2) .*  ( (s).^2 + (s.*wc*1.8478) + (wc)^2 )); %% h4w takes in wc and w and outputs H4 @ omega

wc = 2*pi*100; %%TO DO: play around with omega c and why the first 2 won't work and why 3rd one will.

w0 = (w);
w2pi60 = (2*pi*60);

hw0 = h1w(wc,w0);
hw2pi60=h1w(wc, w2pi60);

error = 0.01;


if abs( abs(hw0) - 0.01 ) / 0.01 <= error %% experimental value - expected value / expected value = percent error
    disp('passes const. 1');
else
    disp('fails const. 1');
end

if abs( angle(hw2pi60) ) <= 4 %% experimental value < 4
    disp('passes const. 2');
else
    disp('fails const. 2');
end

if abs( abs(hw2pi60) - 1 ) <= error
    disp('passes const. 3');
else
    disp('fails const. 3');
end
```

2c. The transfer functions were written in the following code:

```
%% part c
omega = logspace(1,6,10000); %% go all the way to the frequency



s = j.*omega;
H1W = 1./(1+(s./wc)); %% H(1)W
H2W = (wc)^2./( (s).^2 + (s .* wc .* ( 2/sqrt(2) ) ) + (wc)^2 ); %% H(2)W
H4W = (wc)^4./( ((s).^2 + (s .* wc .* 0.7654) + (wc)^2) .*  ( (s).^2 + (s.*wc*1.8478) + (wc)^2 )); %% H(4)W
```

```
%% H(1)W
subplot(2, 1, 1);
semilogx(omega, 20*log10(abs(H1W))); %% magnitude plot
title('H1 Magnitude');
xlabel('Frequency');
ylabel('Magnitude');
subplot(2, 1, 2);
semilogx(omega, rad2deg(abs(H1W))); %% bode plot
title('H1 Angle');
xlabel('Frequency');
ylabel('Angle');
```

```
%% H(2)W
subplot(2, 1, 1);
semilogx(omega, 20*log10(abs(H2W))); %% magnitude plot
title('H2 Magnitude');
xlabel('Frequency');
ylabel('Magnitude');
subplot(2, 1, 2);
semilogx(omega, rad2deg(abs(H2W))); %% angle plot
title('H2 Angle');
xlabel('Frequency');
ylabel('Angle');
```

```
%% H(4)W
subplot(2, 1, 1);
semilogx(omega, 20*log10(abs(H4W))); %% magnitude plot
title('H4 Magnitude');
xlabel('Frequency');
ylabel('Magnitude');
subplot(2, 1, 2);
semilogx(omega, rad2deg(angle(H4W))) %% angle plot
title('H4 Angle');
xlabel('Frequency');
ylabel('Angle');
```

2d. The following code is found to generate a plot of 20log_10|H(w)| versus log_10w:

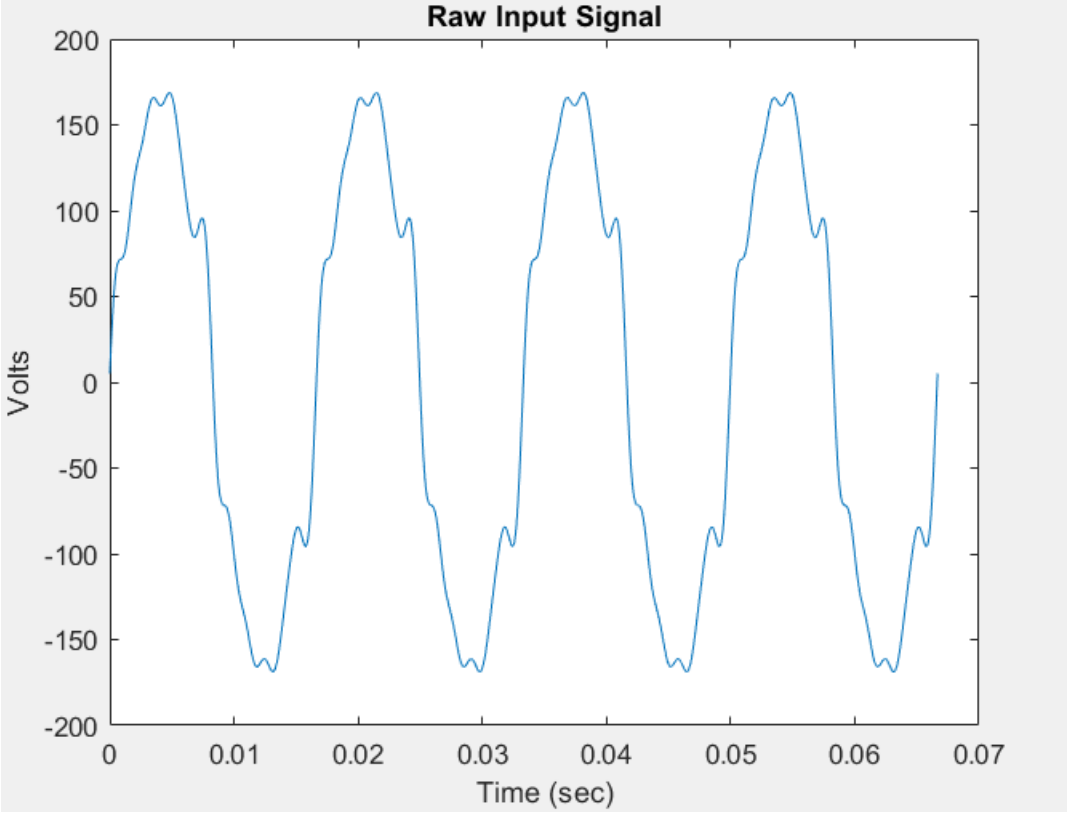
```
%% part d, filter 1 and 2 angles immediately go down. you'll meet 2 constraints but not your 3rd
alpha_4 = 1; %% constructs the a_n values for the Vin signal

h3w = @ (w, wc) (wc)^4./( ((s).^2 + (s .* wc .* 0.7654) + (wc)^2) .*  ( (s).^2 + (s.*wc*1.8478) + (wc)^2 )); %% New filter

% construct alpha values for Vout
h1 = h3w(w, wc);
h2 = h3w(2*w, wc);
h3 = h3w(2*pi*60, wc);
h4 = h3w(3*w, wc);
% create the alpha values
v_a1 = (alpha_1 * h1);
v_a2 = (alpha_2 * h2);
v_a3 = (alpha_3 * h3);
v_a4 = (alpha_4 * h4);
% create V_out
V_out1 = 2 * abs(v_a1) * cos(w * t + angle(v_a1));
V_out2 = 2 * abs(v_a2) * cos(2* w * t + angle(v_a2));
V_out3 = 2 * abs(v_a3) * cos(3 * w * t + angle(v_a3));
V_out4 = 2 * abs(v_a4) * cos(2 * pi * 60 * t + angle(v_a4));

V_out = V_out1 + V_out2 + V_out3 + V_out4;


figure(6);
subplot(2, 1, 1);
plot(t, V_out);
title('Filtered Voltage Out');
xlabel('Time: sec');
ylabel('Voltage');
subplot(2, 1, 2);
plot(t, Vs);
title('Unfiltered Voltage Source');
xlabel('Time: sec');
ylabel('Voltage');
```

Problem 3 – A Simple Power Quality Analyzer

3a. The harmonic numbers were determined by the following code:

```
%% part a
f = 60;
T = 1/f;
omega = 2*pi/T;

fs = 60000;
Ts = 1/fs;
t = [0:length(x)-1]*Ts;


N =[-13:1:13];
alpha_1 = 0;

for n = 1:1:length(t)
    alpha_1 = alpha_1 + x(n)*exp(-j*1*omega*t(n))*Ts;
end

alpha_1 = alpha_1/T;

figure(1);
plot(t, x);
title('Raw Input Signal');
xlabel('Time (sec)');
ylabel('Volts');
```

The following graph is shown:



Problem 4 – A Touch Tone Recognition System

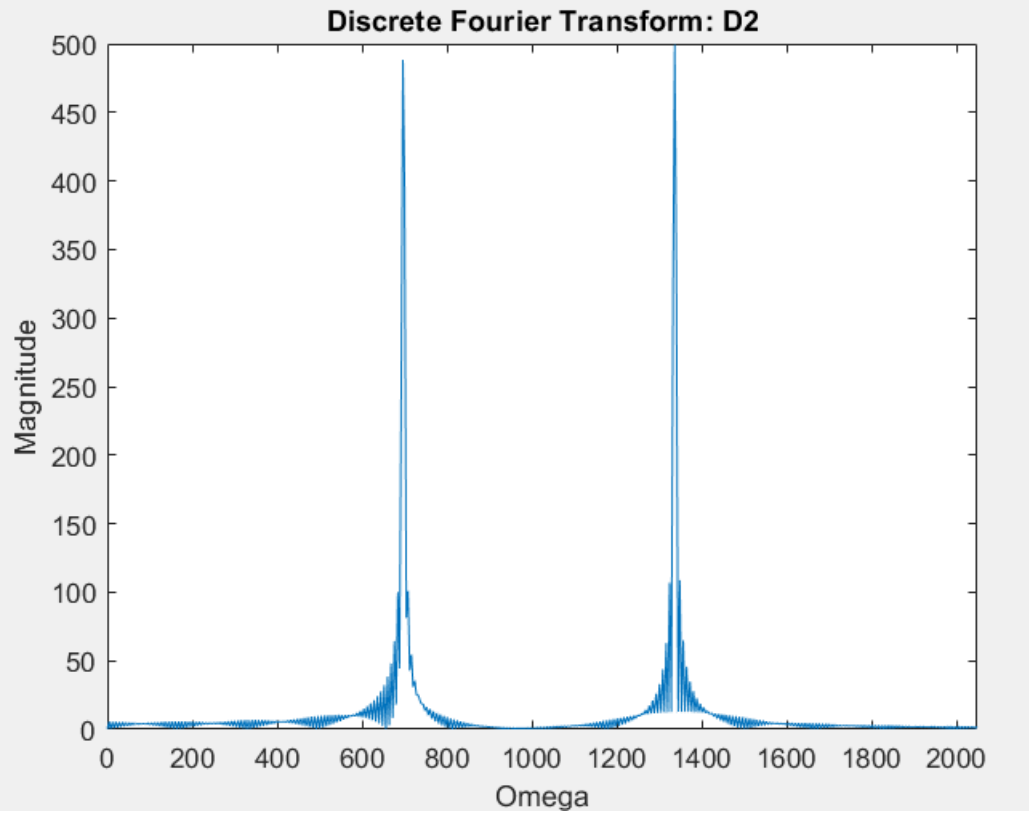
4a, 4b, 4c. The signals in corresponding discrete-time and continuous-time are shown in the following code:

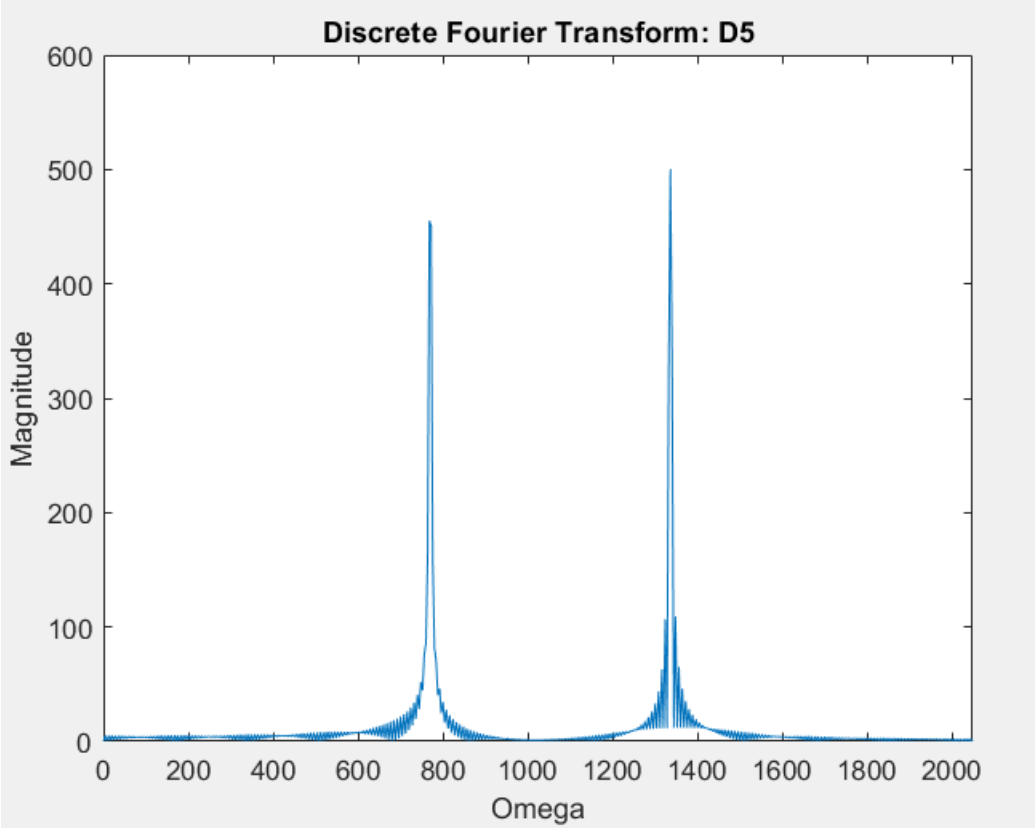
```
%% part a - c
Fs = 8192; %% sampled at 8192 Hz
t = [0:1:999]; %% time interval
sample = 100; %% sample
deltat = 1/Fs;
n_discretized = t*deltat; %% discretized time

d1 = sin(2*pi*697*n_discretized) + sin(2*pi*1209*n_discretized); %% 1
d2 = sin(2*pi*697*n_discretized) + sin(2*pi*1336*n_discretized); %% 2
d3 = sin(2*pi*697*n_discretized) + sin(2*pi*1477*n_discretized); %% 3
d4 = sin(2*pi*770*n_discretized) + sin(2*pi*1209*n_discretized); %% 4
d5 = sin(2*pi*770*n_discretized) + sin(2*pi*1336*n_discretized); %% 5
d6 = sin(2*pi*770*n_discretized) + sin(2*pi*1477*n_discretized); %% 6
d7 = sin(2*pi*852*n_discretized) + sin(2*pi*1209*n_discretized); %% 7
d8 = sin(2*pi*852*n_discretized) + sin(2*pi*1336*n_discretized); %% 8
d9 = sin(2*pi*852*n_discretized) + sin(2*pi*1477*n_discretized); %% 9
d0 = sin(2*pi*947*n_discretized) + sin(2*pi*1336*n_discretized); %% 0

space = zeros(size(n_discretized)); %% vector

x = [d7 space d0 space d4 space d4 space d2 space d1 space d3 space d4 space d4 space d6];
sound(x, Fs);
```

4d. The following code was done to find the Discrete Fourier Transform:

```
%% part d
N_p = 2048;
X = zeros(1, N_p);

d2p = [d2 zeros(1,2048 - length(d2))]; %% 1000 -> 2048 padding
d5p = [d5 zeros(1, 2048 - length(d5))]; %% 1000 -> 2048 padding

D2P = zeros(size(d2p));
D5P = zeros(size(d5p));

for k = 1:1:N_p  %% fourier transform
    for n = 1:1:N_p
            D2P(k) = D2P(k) + d2p(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
            D5P(k) = D5P(k) + d5p(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
    end
end

k = [0:1:N_p - 1];
omega = k * 2 * pi/N_p;
w = k * Fs/N_p; %% convert radians a sec to Hz

figure(2);
plot(w, abs(D2P));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform: D2');

figure(3);
plot(w, abs(D5P));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform: D5');
```

This results in the following plots:





4e. Using the touch.mat file, the Discrete Fourier Transform used the same format as part d, except accounting for the vectors of X1.

```
%% part e
X1 = x1(1:1000);
X2 = x1(1101:2100);
X3 = x1(2201:3200);
X4 = x1(3301:4300);
X5 = x1(4401:5400);
X6 = x1(5501:6500);
X7 = x1(6601:7600);

DX1P = [X1 zeros(1,2048 - length(X1))]; %% 1000 -> 2048 padding
DX2P = [X2 zeros(1, 2048 - length(X2))]; %% 1000 -> 2048 padding
DX3P = [X3 zeros(1, 2048 - length(X3))]; %% 1000 -> 2048 padding
DX4P = [X4 zeros(1, 2048 - length(X4))]; %% 1000 -> 2048 padding
DX5P = [X5 zeros(1, 2048 - length(X5))]; %% 1000 -> 2048 padding
DX6P = [X6 zeros(1, 2048 - length(X6))]; %% 1000 -> 2048 padding
DX7P = [X7 zeros(1, 2048 - length(X7))]; %% 1000 -> 2048 padding

dx1p = zeros(size(DX1P));
dx2p = zeros(size(DX2P));
dx3p = zeros(size(DX3P));
dx4p = zeros(size(DX4P));
dx5p = zeros(size(DX5P));
dx6p = zeros(size(DX6P));
dx7p = zeros(size(DX7P));


for k = 1:1:N_p  %% fourier transform
    for n = 1:1:N_p
            dx1p(k) = dx1p(k) + DX1P(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
            dx2p(k) = dx2p(k) + DX2P(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
            dx3p(k) = dx3p(k) + DX3P(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
            dx4p(k) = dx4p(k) + DX4P(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
            dx5p(k) = dx5p(k) + DX5P(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
            dx6p(k) = dx6p(k) + DX6P(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
            dx7p(k) = dx7p(k) + DX7P(n) * exp(-j * (2*pi)/N_p * (k-1) * (n-1));
    end
end
```

```
subplot(4, 2, 1);
plot(w, abs(dx1p));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform of the 1st Digit'); %% makes the number 4


subplot(4, 2, 2);
plot(w, abs(dx2p));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform of the 2nd Digit'); %% makes the number 9


subplot(4, 2, 3);
plot(w, abs(dx3p));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform of the 3rd Digit'); %% makes the number 1


subplot(4, 2, 4);
plot(w, abs(dx4p));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform of the 4th Digit'); %% makes the number 5


subplot(4, 2, 5);
plot(w, abs(dx5p));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform of the 5th Digit'); %% makes the number 8


subplot(4, 2, 6);
plot(w, abs(dx6p));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform of the 6th Digit'); %% makes the number 7


subplot(4, 2, 7);
plot(w, abs(dx7p));
xlim([0, 2048]);
xlabel('Omega');
ylabel('Magnitude');
title('Discrete Fourier Transform of the 7th Digit'); %% makes the number 7
```

This resulted in the following plots: