Lab 4, EC401

Varsha Singh, [redacted]

Problem 4.1

(a)

```
% Define row frequencies (Hz)
fRow1 = 679;
fRow2 = 770;
fRow3 = 852;
fRow4 = 941;

% Define column frequencies (Hz)
fCol1 = 1209;
fCol2 = 1336;
fCol3 = 1447;

Fs = 8192;          % Sampling frequency 8.192 kHz
T = 1;              % Duration of tone in sec = 1000ms
N = round(Fs*T);    % Number of samples at Fs
t = (0:N-1)/Fs;     % Corresponding time vector: N samples
                    % separated by 1/Fs seconds

% Create signals/sinusoids/tones for each row
sRow1 = sin(2*pi*fRow1*t);

% Create signals/sinusoids/tones for each column
sCol1 = sin(2*pi*fCol1*t);
```

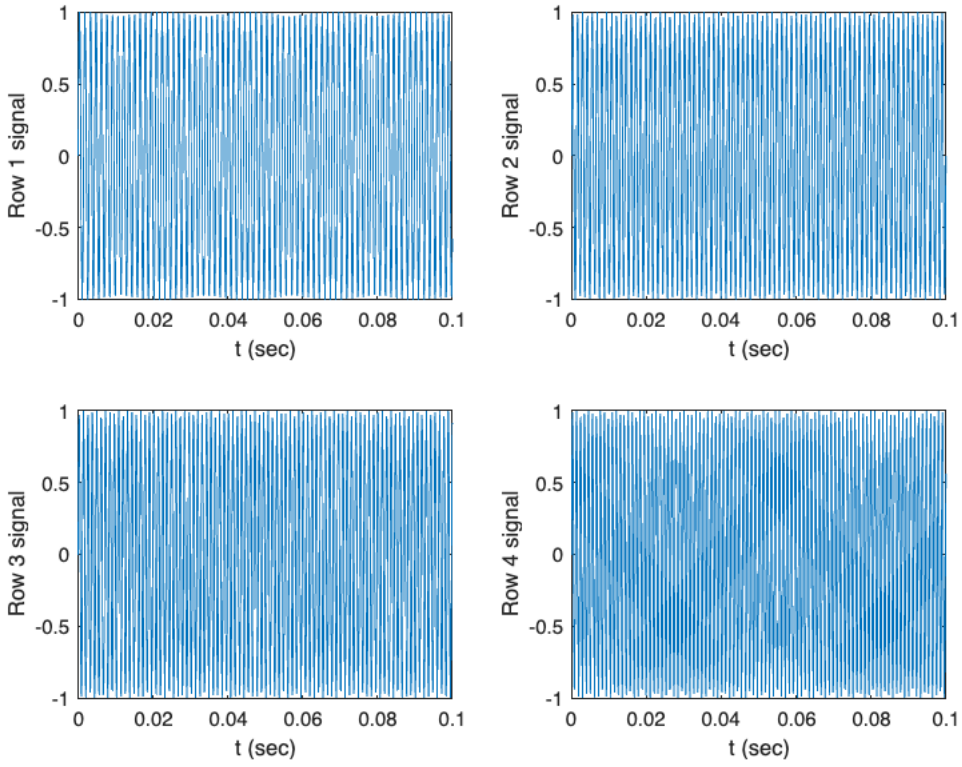
a.1) In a similar way, create the signals for the remaining row signals: sRow2, sRow3, sRow4 and the remaining column signals: sCol1, sCol2, sCol3.

```
% Create signals/sinusoids/tones for each row
sRow2 = sin(2*pi*fRow2*t);
sRow3 = sin(2*pi*fRow3*t);
sRow4 = sin(2*pi*fRow4*t);

% Create signals/sinusoids/tones for each column
sCol2 = sin(2*pi*fCol2*t);
sCol3 = sin(2*pi*fCol3*t);
```

a.1) Plot all the signals using subplot and display the first 10ms (0.01 sec)

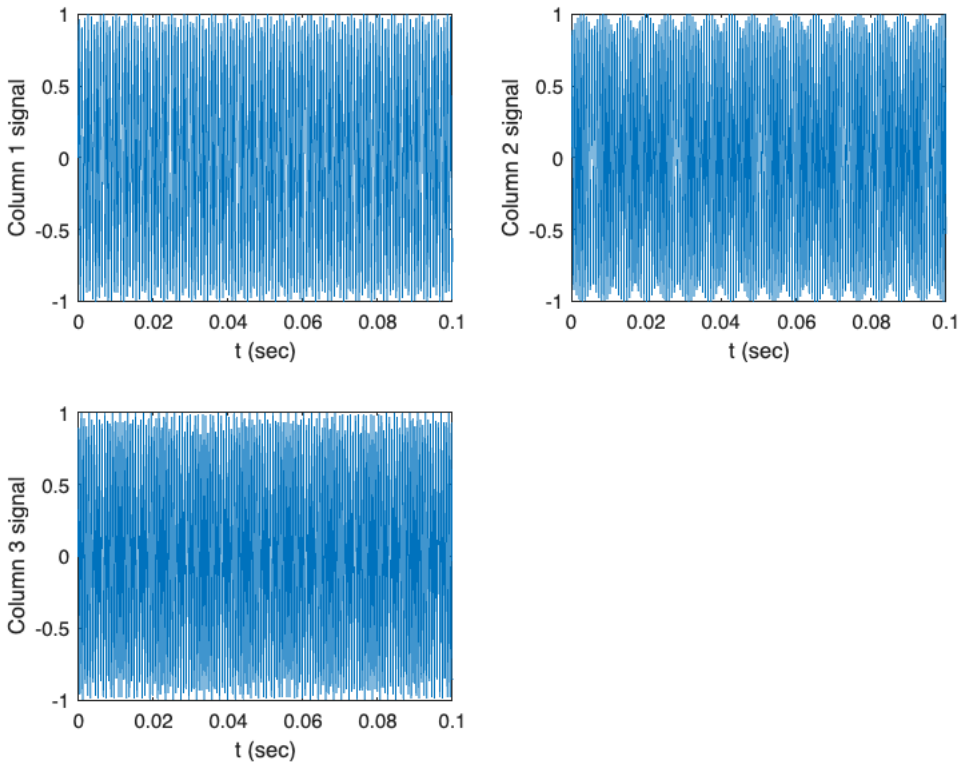
```
% Plot row signals
figure;
subplot(2, 2, 1);
plot(t, sRow1);
xlabel('t (sec)');
ylabel('Row 1 signal');
```

```
axis([0 0.1 -1 1]);

subplot(2, 2, 2);
plot(t, sRow2);
xlabel('t (sec)');
ylabel('Row 2 signal');
axis([0 0.1 -1 1]);

subplot(2, 2, 3);
plot(t, sRow3);
xlabel('t (sec)');
ylabel('Row 3 signal');
axis([0 0.1 -1 1]);

subplot(2, 2, 4);
plot(t, sRow4);
xlabel('t (sec)');
ylabel('Row 4 signal');
axis([0 0.1 -1 1]);
```




```
% Plot column signals
figure;
subplot(2, 2, 1);
plot(t, sCol1);
xlabel('t (sec)');
ylabel('Column 1 signal');
axis([0 0.1 -1 1]);
```

```
subplot(2, 2, 2);
plot(t, sCol2);
xlabel('t (sec)');
ylabel('Column 2 signal');
axis([0 0.1 -1 1]);

subplot(2, 2, 3);
plot(t, sCol3);
xlabel('t (sec)');
ylabel('Column 3 signal');
axis([0 0.1 -1 1]);
```



a.3) Play the signals through the speakers

```
% Row
sound(sRow1,Fs);
pause(0.1);
clear sound;

sound(sRow2,Fs);
pause(0.1);
clear sound;

sound(sRow3,Fs);
pause(0.1);
clear sound;
```

```
sound(sRow4,Fs);
pause(0.1);
clear sound;

% Col
sound(sCol1,Fs);
pause(0.1);
clear sound;

sound(sCol2,Fs);
pause(0.1);
clear sound;

sound(sCol3,Fs);
pause(0.1);
clear sound;
```

b)

b.1) create signals for all keys

```
s1 = (sRow1 + sCol1)/2;
s2 = (sRow1 + sCol2)/2;
s3 = (sRow1 + sCol3)/2;

s4 = (sRow2 + sCol1)/2;
s5 = (sRow2 + sCol2)/2;
s6 = (sRow2 + sCol3)/2;

s7 = (sRow3 + sCol1)/2;
s8 = (sRow3 + sCol2)/2;
s9 = (sRow3 + sCol3)/2;

sStar = (sRow4 + sCol1)/2;
s0 = (sRow4 + sCol2)/2;
sPound = (sRow4 + sCol3)/2;
```

b.2) play each key signal

```
sound(s1,Fs);
pause(0.2);
clear sound;

sound(s2,Fs);
pause(0.2);
clear sound;

sound(s3,Fs);
pause(0.2);
clear sound;
```

```
sound(s4,Fs);
pause(0.2);
clear sound;

sound(s5,Fs);
pause(0.2);
clear sound;

sound(s6,Fs);
pause(0.2);
clear sound;

sound(s7,Fs);
pause(0.2);
clear sound;

sound(s8,Fs);
pause(0.2);
clear sound;

sound(s9,Fs);
pause(0.2);
clear sound;

sound(sStar,Fs);
pause(0.2);
clear sound;

sound(s0,Fs);
pause(0.2);
clear sound;

sound(sPound,Fs);
pause(0.2);
clear sound;
```

b.3) create a 100ms pause

```
Tpause = 0.1;   % 0.1s pause
p = zeros(1, round(Fs*Tpause));
```

c) assemble a phone number

```
sNumber = [s5 p s1 p s2 p s7 p s7 p s9 p s0 p s2 p s6 p s0];

sound(sNumber, Fs);
```

d) create a time vector to the number

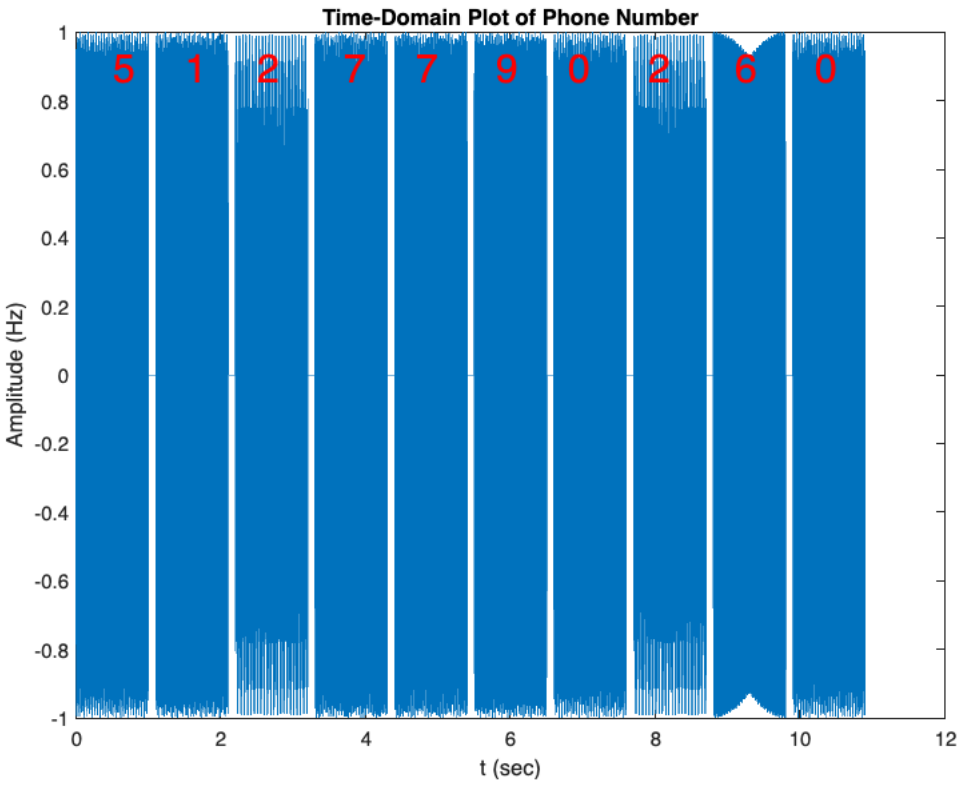
```
tNumber = [0:length(sNumber)-1]/Fs;
```

```
tNumber = 1×89291
         0    0.0001    0.0002    0.0004    0.0005    0.0006    0.0007    0.0009 ···
```

```
figure;
plot(tNumber, sNumber);
xlabel('t (sec)');
ylabel('Amplitude (Hz)');
title('Time-Domain Plot of Phone Number');

% labels
digit_times = [0.5, 1.5, 2.5, 3.7, 4.7, 5.8, 6.8, 7.9, 9.1, 10.2];
digits = ['5' '1' '2' '7' '7' '9' '0' '2' '6' '0'];
fprintf("%d",length(digit_times))
```

```
10
```

```
for i = 1:length(digit_times)
    text(digit_times(i), max(sNumber) * 0.9, digits(i), 'FontSize', 20, 'Color', 'r');
end
```



4.2: Analyzing Telephone Keypad Tones with the Fourier Transform

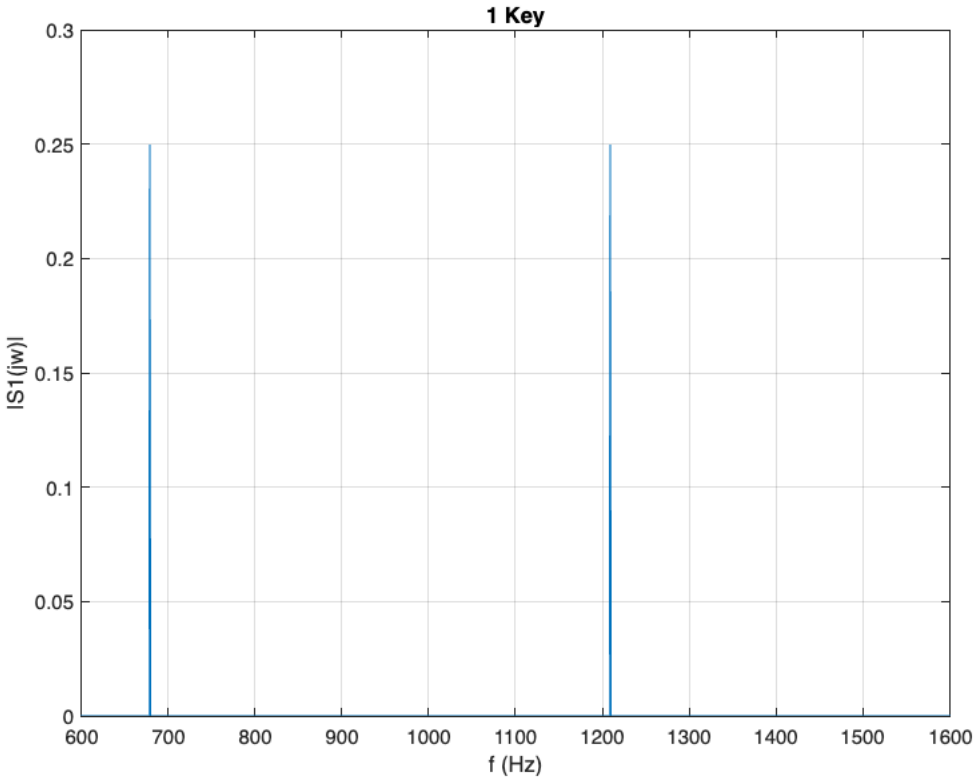
a) using ctft, find the Fourier transform of each of the key pad signals

```
[S1, f] = ctft(s1, Fs);;
[S2, f] = ctft(s2, Fs);
```

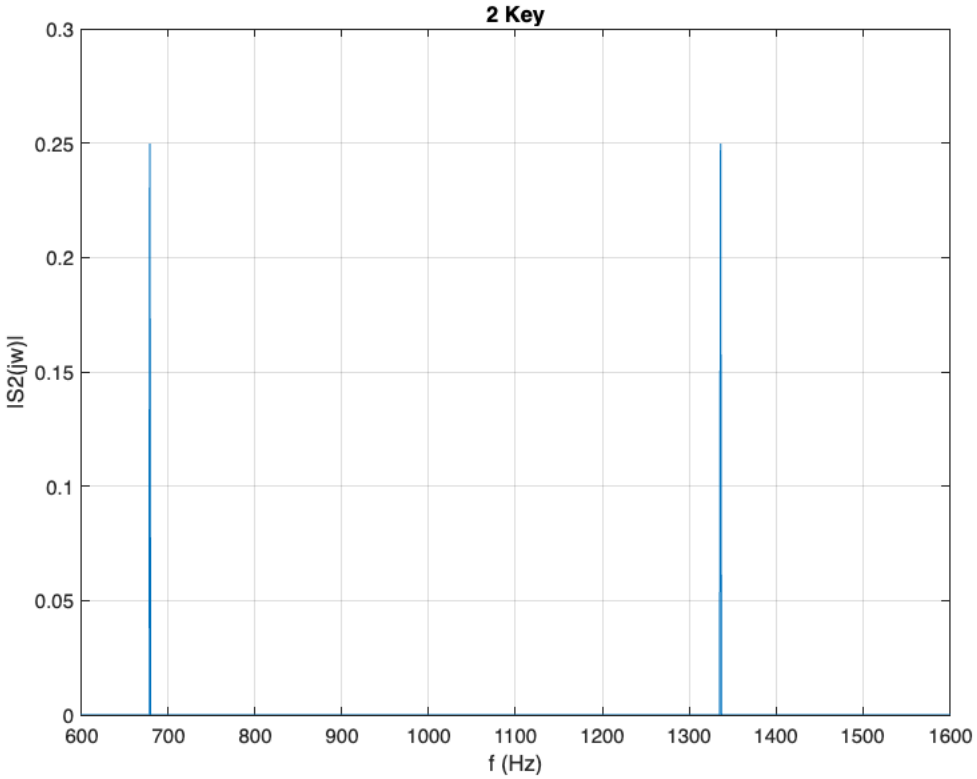
```
[S3, f] = ctft(s3, Fs);
[S4, f] = ctft(s4, Fs);
[S5, f] = ctft(s5, Fs);
[S6, f] = ctft(s6, Fs);
[S7, f] = ctft(s7, Fs);
[S8, f] = ctft(s8, Fs);
[S9, f] = ctft(s9, Fs);
[SStar, f] = ctft(sStar, Fs);
[S0, f] = ctft(s0, Fs);
[SPound, f] = ctft(sPound, Fs);
```

a.2) plot magnitude of the Fourier Transforms

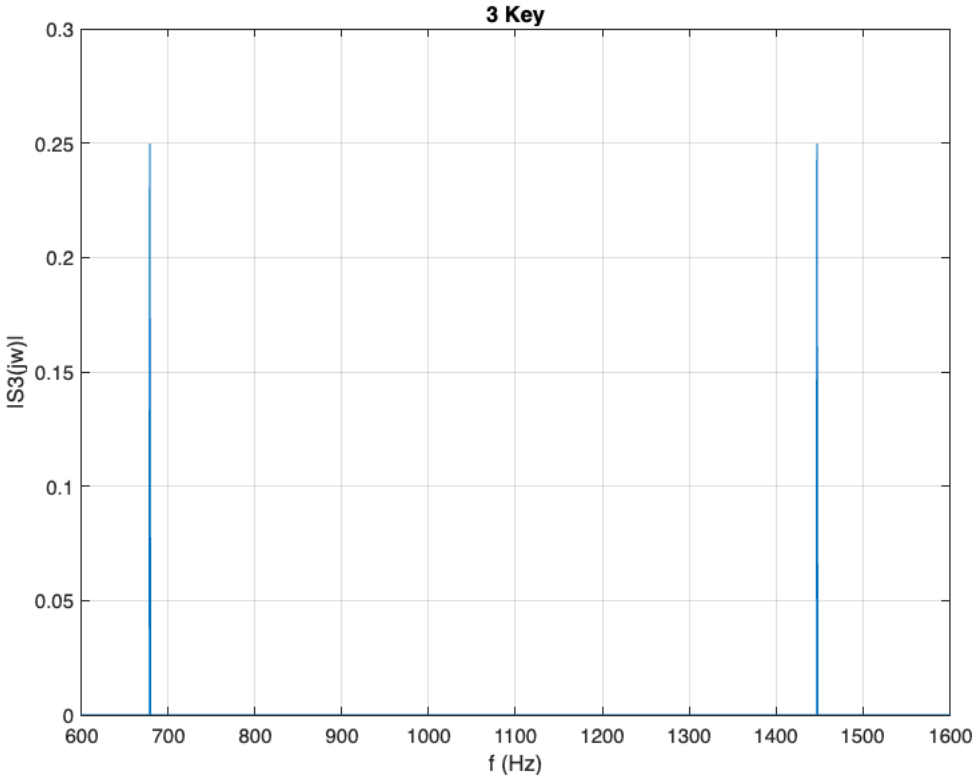
```
plot(f, abs(S1));
grid;
xlabel('f (Hz)')
ylabel('|S1(jw)|')
title('1 Key')
axis([600 1600 0 0.3])
```



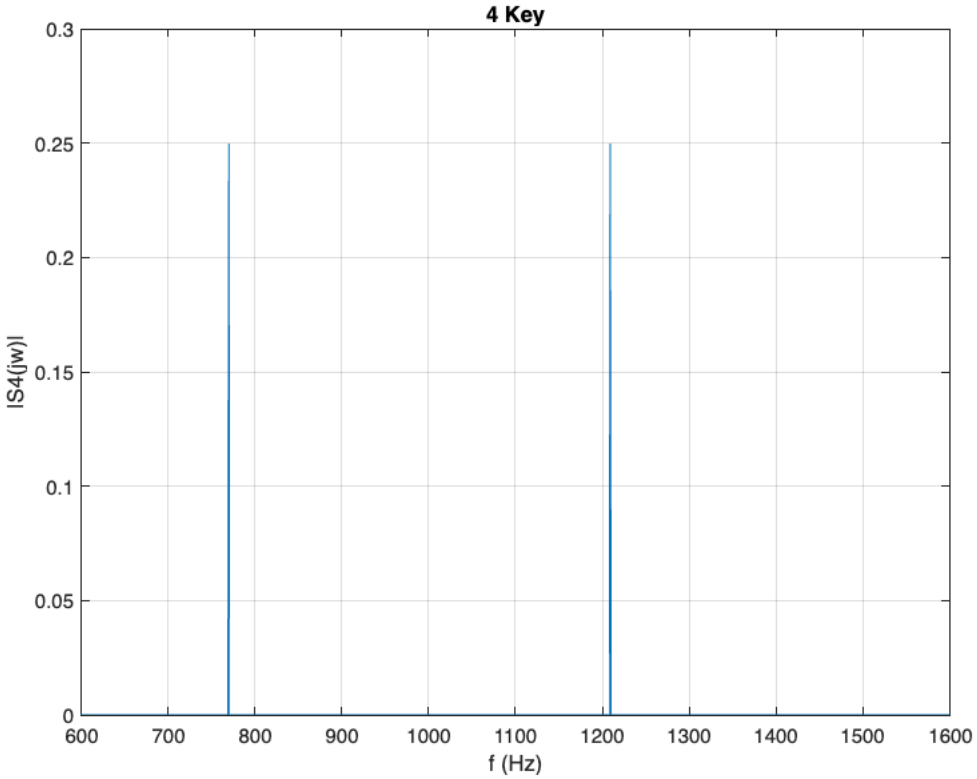

```
plot(f, abs(S2));
grid;
xlabel('f (Hz)')
ylabel('|S2(jw)|')
title('2 Key')
axis([600 1600 0 0.3])
```



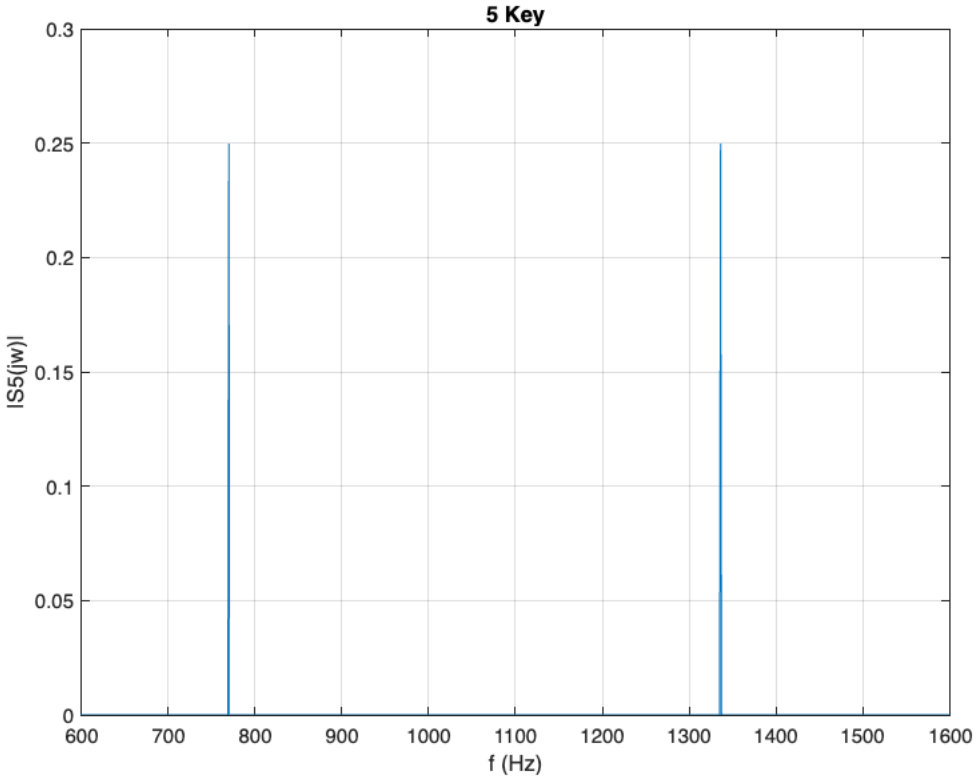

```
plot(f, abs(S3));
grid;
xlabel('f (Hz)')
ylabel('|S3(jw)|')
title('3 Key')
axis([600 1600 0 0.3])
```



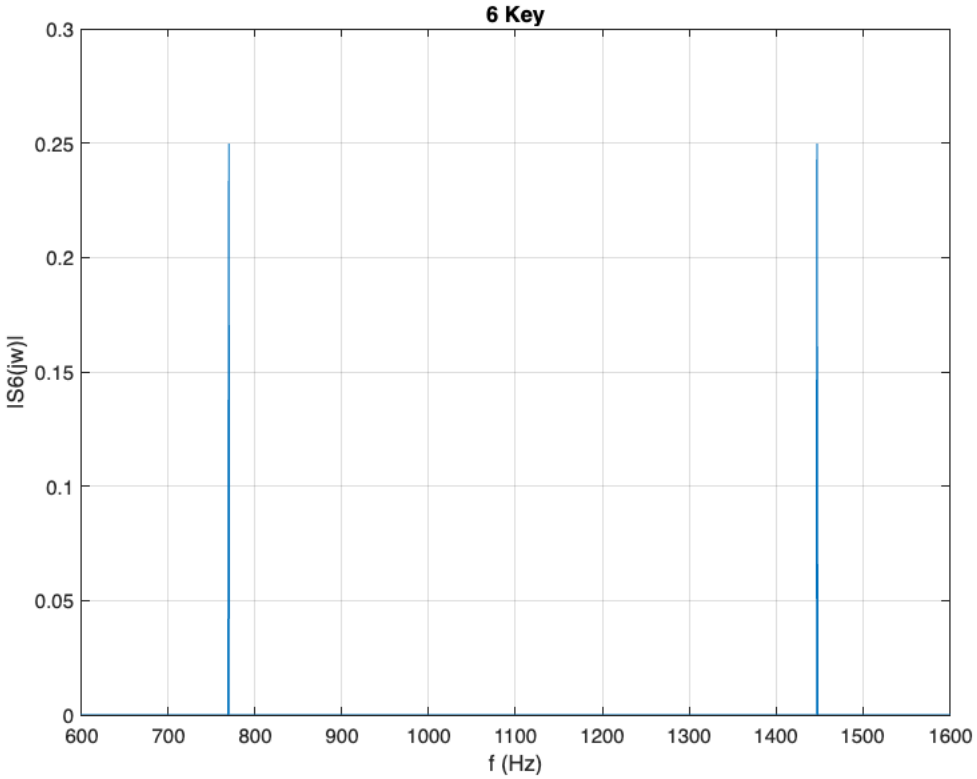

```
plot(f, abs(S4));
grid;
xlabel('f (Hz)')
ylabel('|S4(jw)|')
title('4 Key')
axis([600 1600 0 0.3])
```



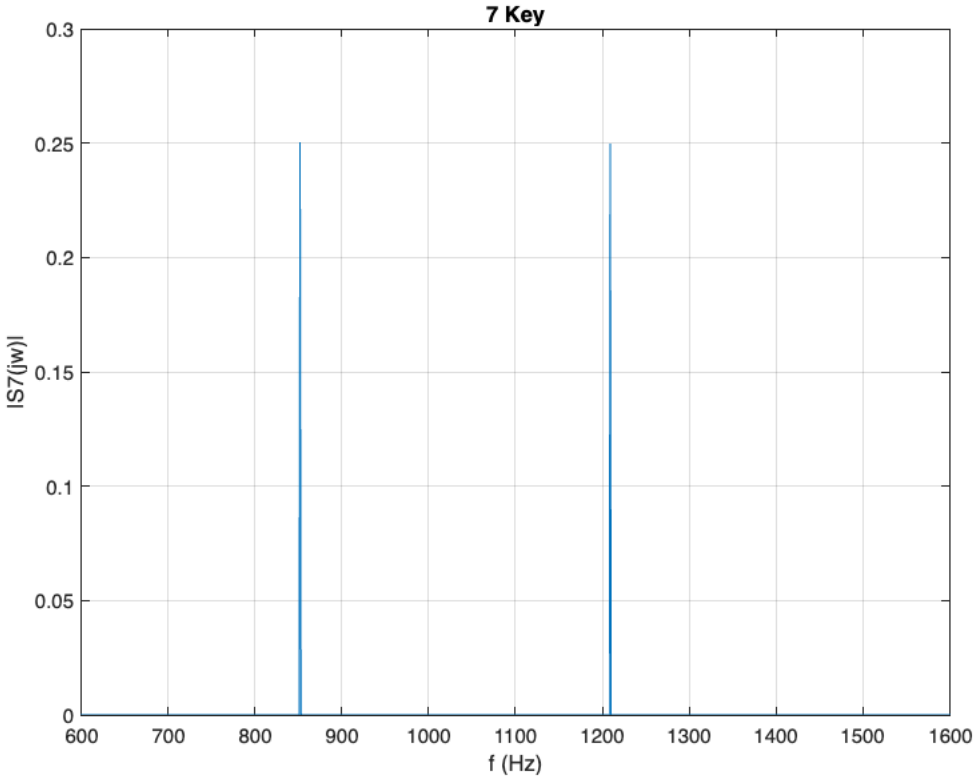

```
plot(f, abs(S5));
grid;
xlabel('f (Hz)')
ylabel('|S5(jw)|')
title('5 Key')
axis([600 1600 0 0.3])
```



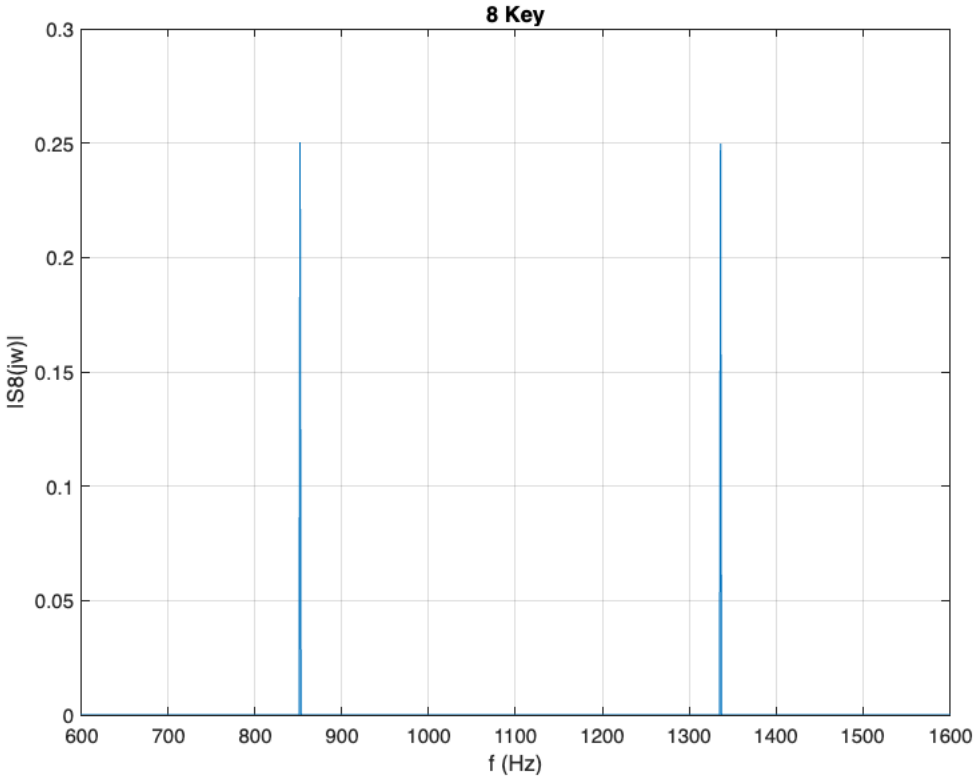

```
plot(f, abs(S6));
grid;
xlabel('f (Hz)')
ylabel('|S6(jw)|')
title('6 Key')
axis([600 1600 0 0.3])
```



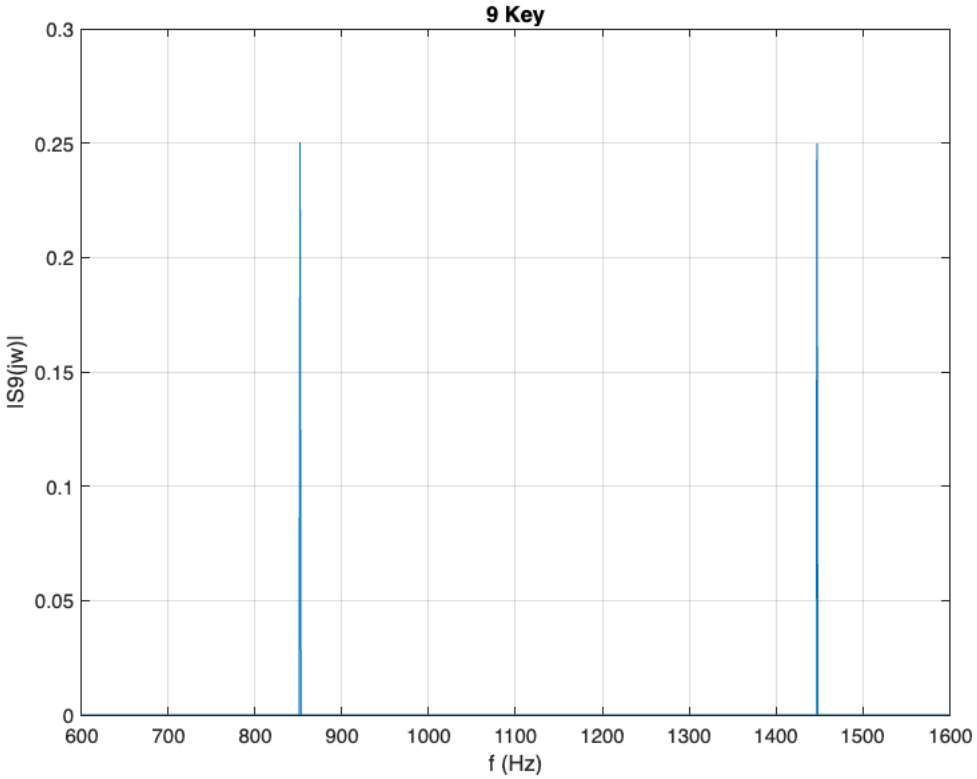

```
plot(f, abs(S7));
grid;
xlabel('f (Hz)')
ylabel('|S7(jw)|')
title('7 Key')
axis([600 1600 0 0.3])
```



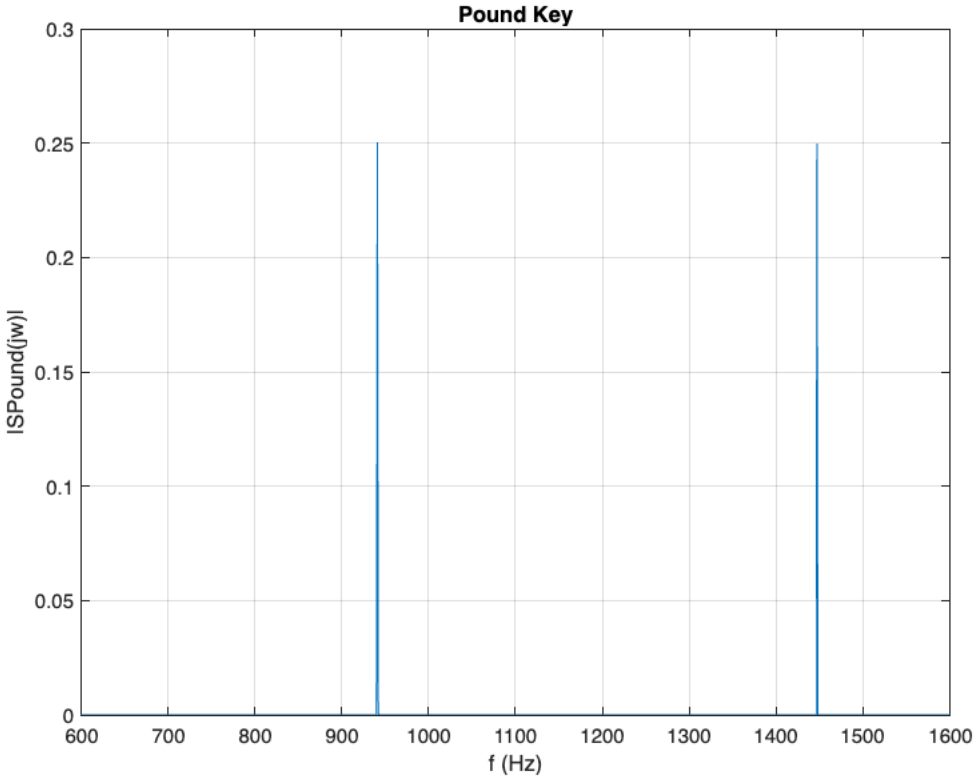

```
plot(f, abs(S8));
grid;
xlabel('f (Hz)')
ylabel('|S8(jw)|')
title('8 Key')
axis([600 1600 0 0.3])
```



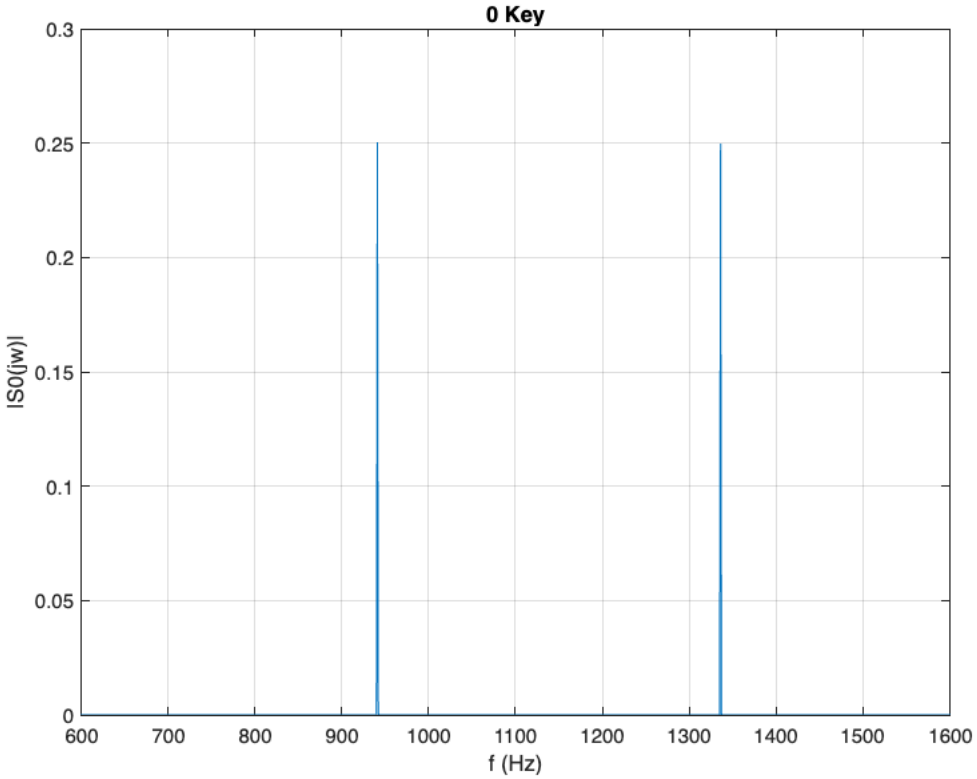

```
plot(f, abs(S9));
grid;
xlabel('f (Hz)')
ylabel('|S9(jw)|')
title('9 Key')
axis([600 1600 0 0.3])
```



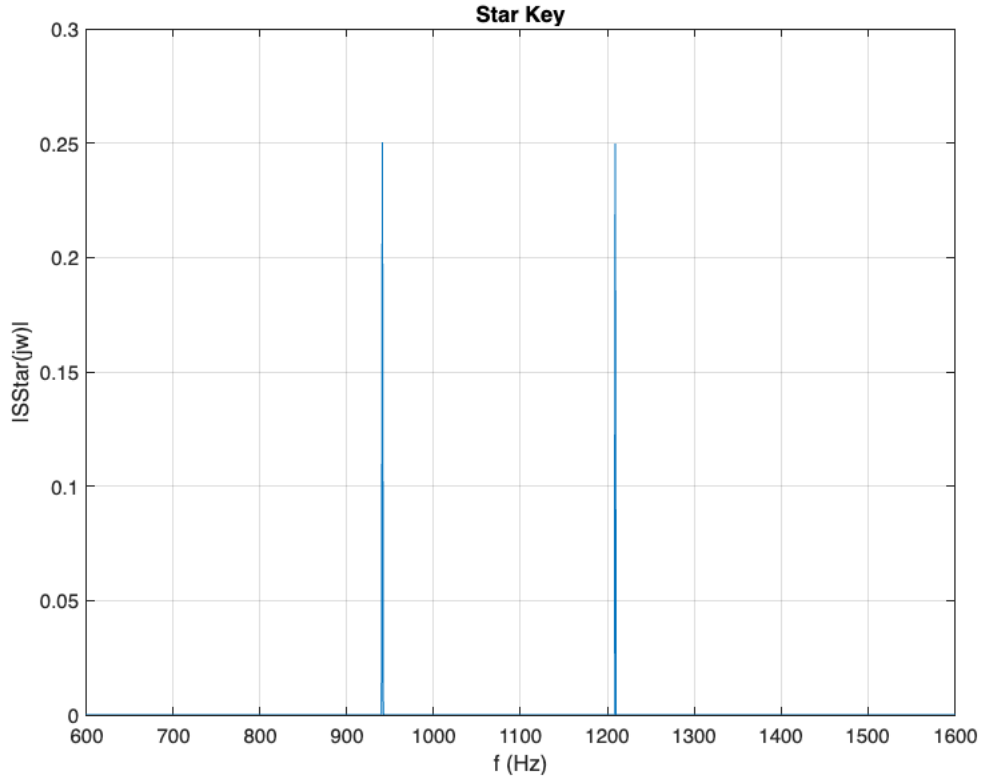

```
plot(f, abs(SPound));
grid;
xlabel('f (Hz)')
ylabel('|SPound(jw)|')
title('Pound Key')
axis([600 1600 0 0.3])
```




```
plot(f, abs(S0));
grid;
xlabel('f (Hz)')
ylabel('|S0(jw)|')
title('0 Key')
axis([600 1600 0 0.3])
```

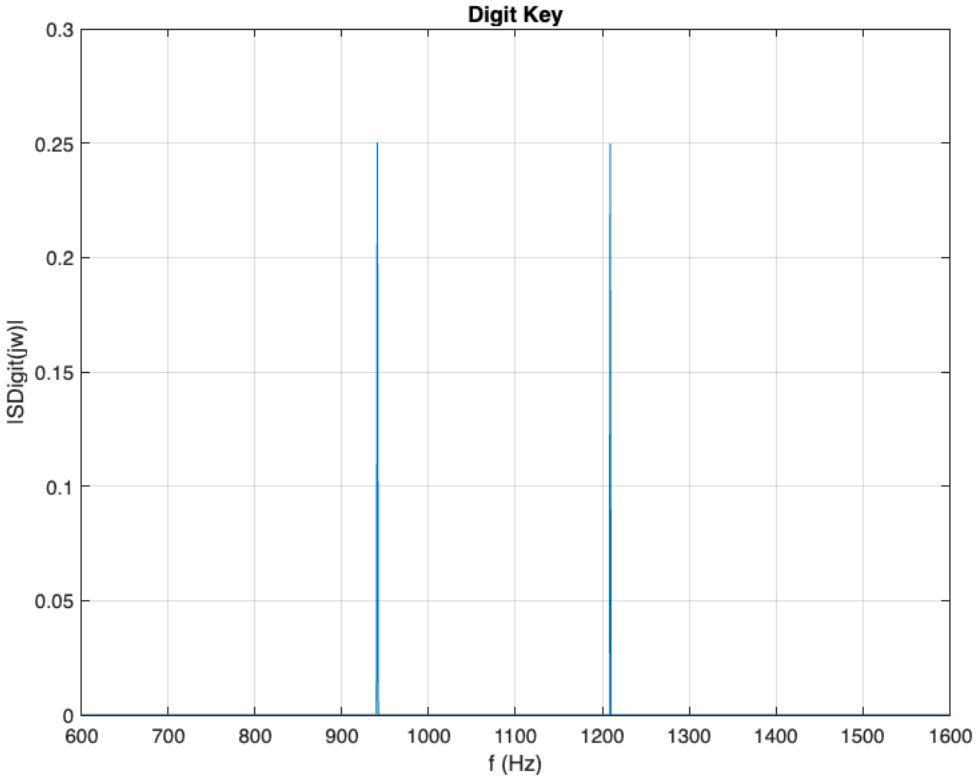



```
plot(f, abs(SStar));
grid;
xlabel('f (Hz)')
ylabel('|SStar(jw)|')
title('Star Key')
axis([600 1600 0 0.3])
```



b) use a to decome a phone number from DTMF signal

```
[SDigit, f] = ctft(sDigit, Fs);
load decode

plot(f, abs(SDigit));
grid;
xlabel('f (Hz)')
ylabel('|SDigit(jw)|')
title('Digit Key')
axis([600 1600 0 0.3])
```



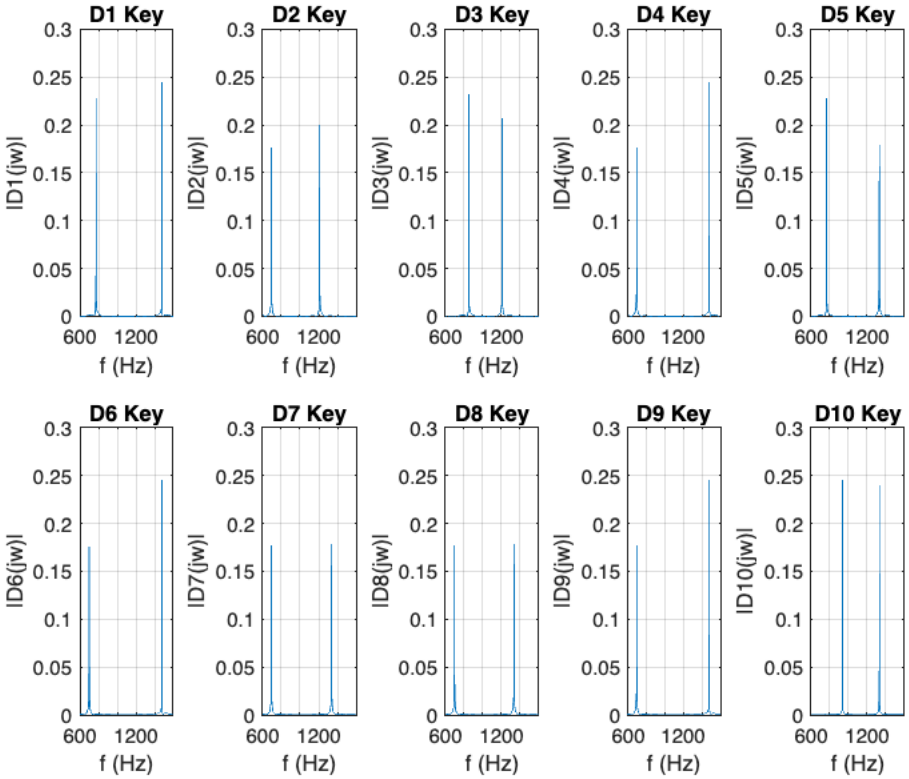

```
%%% using ctft, it seems like the digit represented is: * (star)
```

c) DTMF representation of 10-digit telephone number. correctly decode the phone nuumber, splitting overall signals into components corresponding to digit or space, prior to perforing Fourier analysis

```
[nstart, nstop] = dtmfcut(sNumber, Fs);     % Find key break points

% extracting 10 digits in overall number sequence
d1 = sNumber(nstart(1):nstop(1));
d2 = sNumber(nstart(2):nstop(2));
d3 = sNumber(nstart(3):nstop(3));
d4 = sNumber(nstart(4):nstop(4));
d5 = sNumber(nstart(5):nstop(5));
d6 = sNumber(nstart(6):nstop(6));
d7 = sNumber(nstart(7):nstop(7));
d8 = sNumber(nstart(8):nstop(8));
d9 = sNumber(nstart(9):nstop(9));
d10 = sNumber(nstart(10):nstop(10));

% decode all 10 digits
[D1, f] = ctft(d1, Fs);
figure;
subplot(2,5,1);
plot(f, abs(D1));
grid;
xlabel('f (Hz)')
```

```
ylabel('|D1(jw)|')
title('D1 Key')
axis([600 1600 0 0.3])

[D2, f] = ctft(d2, Fs);
subplot(2,5,2);
plot(f, abs(D2));
grid;
xlabel('f (Hz)')
ylabel('|D2(jw)|')
title('D2 Key')
axis([600 1600 0 0.3])

[D3, f] = ctft(d3, Fs);
subplot(2,5,3);
plot(f, abs(D3));
grid;
xlabel('f (Hz)')
ylabel('|D3(jw)|')
title('D3 Key')
axis([600 1600 0 0.3])

[D4, f] = ctft(d4, Fs);
subplot(2,5,4);
plot(f, abs(D4));
grid;
xlabel('f (Hz)')
ylabel('|D4(jw)|')
title('D4 Key')
axis([600 1600 0 0.3])

[D5, f] = ctft(d5, Fs);
subplot(2,5,5);
plot(f, abs(D5));
grid;
xlabel('f (Hz)')
ylabel('|D5(jw)|')
title('D5 Key')
axis([600 1600 0 0.3])

[D6, f] = ctft(d6, Fs);
subplot(2,5,6);
plot(f, abs(D6));
grid;
xlabel('f (Hz)')
ylabel('|D6(jw)|')
title('D6 Key')
axis([600 1600 0 0.3])

[D7, f] = ctft(d7, Fs);
```

```
subplot(2,5,7);
plot(f, abs(D7));
grid;
xlabel('f (Hz)')
ylabel('|D7(jw)|')
title('D7 Key')
axis([600 1600 0 0.3])

[D8, f] = ctft(d8, Fs);
subplot(2,5,8);
plot(f, abs(D8));
grid;
xlabel('f (Hz)')
ylabel('|D8(jw)|')
title('D8 Key')
axis([600 1600 0 0.3])

[D9, f] = ctft(d9, Fs);
subplot(2,5,9);
plot(f, abs(D9));
grid;
xlabel('f (Hz)')
ylabel('|D9(jw)|')
title('D9 Key')
axis([600 1600 0 0.3])

[D10, f] = ctft(d10, Fs);
subplot(2,5,10);
plot(f, abs(D10));
grid;
xlabel('f (Hz)')
ylabel('|D10(jw)|')
title('D10 Key')
axis([600 1600 0 0.3])
```



Decoded Number is:

617 353 2230

4c)

In order to create an automatic decoder of DTFM signals, I would need to implement 7 parallel bandpass filters that isolate the specific frequencies associated with each DTMF frequency-correspoonding pair for each dial tone. Because each key has a unique combination of two frequencies, we could be able to decode each number. To reduce noise as a form of preprocessing, we can incorporate a low-pass and high-pass filter --> a bandpass filter. The low-pass would have a cutoff frequency right after the highest tone (~1477 Hz, + 1.5%), and the high-pass would have a cutoff frequency right before the lowest tone (~697 Hz, - 1.5%). This means that the bandpass would fit these constraints, with the lower cutoff being right before the lowest tone and the higher cutoff being right after the highest tone. This can reduce noise around our DTMF signal frequency range. To individually decode, we can pass the signal through the 7 parallel bandpass filters, each with a cutoff range of +/- 1.5% for each frequency. This means that we would need the following bandpass filters, with cutoff frequencies of +/- 1.5% of the frequency given:

1. 697 Hz
2. 770 Hz
3. 852 Hz
4. 941 Hz
5. 1209 Hz

6. 1336 Hz
7. 1477 Hz

For the varying length of keypresses, we should analyze magnitude output from each filter over time, looking for a stable signal that lasts beyond the short threshold of 40ms, to verify that we have detected the signal for a number.

To make sure that the process is fully automated, I imagine that we would need a lookup table to decode the signals recieved to the number it corresponds to. This can be done with AND logic of verifying if two signals are detected for the row and column signals.

High-level block diagram

- assume each bandpass has ±1.5% threshold