

Signal Processing and Linear Systems II

Lab 4: Touch-Tone Dialing (DTMF)

April 25, 2019

Today's Topics

- How does touch-tone (DTMF) dialing work
- Task 1: Generating the DTMF signals
- Task 2: Analyzing a single tone with the FFT
- Task 3: Processing the DTMF signal with a filter bank
- Task 4: Decoding the DTMF signal

Touch-Tone Dialing

- Touch tone signals are a combination of two sinusoids at different frequencies (*Dual Tone Multi-Frequency* or DTMF for short)
 - Lower frequency encodes the row of the key
 - Higher frequency encodes the column

		f_c			
		1209	1336	1477	1633
$f_r,$	697	1	2	3	A
	770	4	5	6	B
	852	7	8	9	C
	941	*	0	#	D

- Example: Key 6 produces the signal

$$y[nT] = \cos(2\pi 770nT) + \cos(2\pi 1477nT).$$

- Time between keys is at least 40 ms
- Key must be pressed for at least 40 ms
 - Note that the frequency separation is as small as 73 Hz!
 - This is only three cycles at 40 ms
- The duration and starting time of each key and space is unknown

Task 1: Generate a DTMF signal for given phone number

- Assume
 - Each key is pressed for 0.5 s (a long time)
 - Keys are spaced by 0.125 s
 - The sampling rate is 8192 Hz

- Write a matlab m-file that takes a number

```
>> phone_number = [ 1 6 5 0 5 5 5 1 2 3 4];
```

and returns the signal

```
>> ttsignal = ttdial(phone_number)
```

- Encode your phone number, and play it back with `sound()`. Compare it to the sound the phone makes when you dial. If you play it into the mouthpiece of a phone, it should make the call! You can now make your own robo-caller.

Task 2: Check the spectrum for a Signal Key

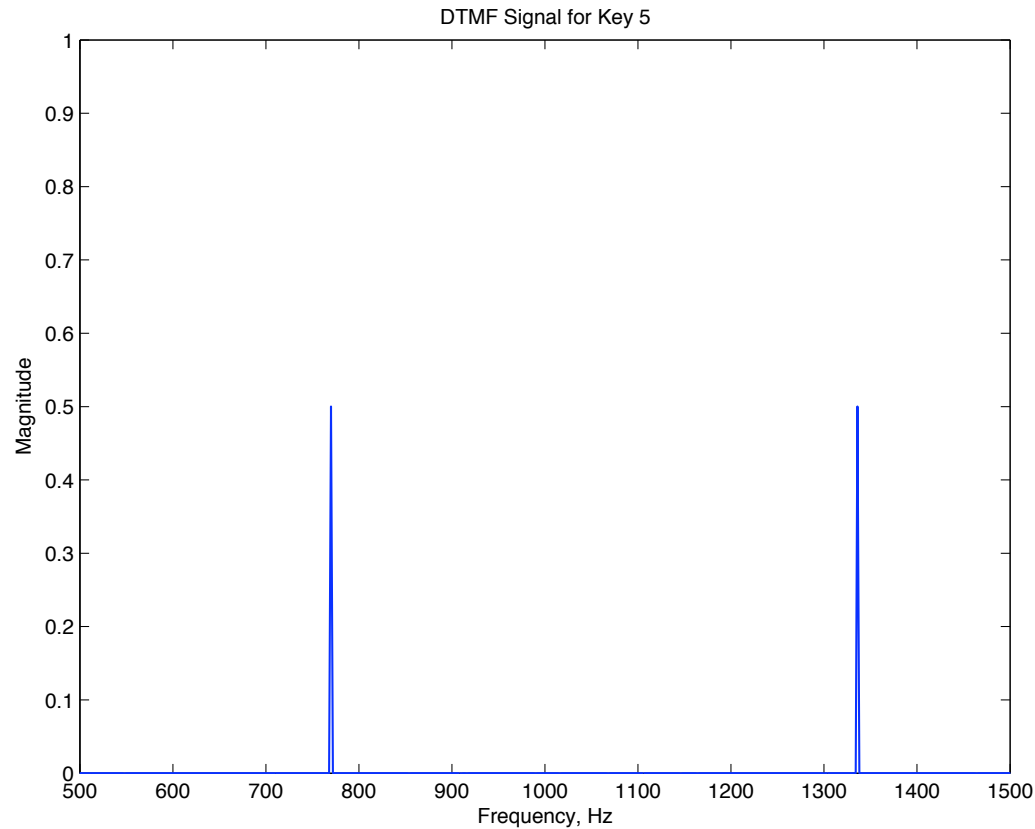
To check whether we are generating the right signals, we'll use the FFT to compute the spectrum of the signal from one key.

- The FFT of an N -point input computes the N frequencies from 0 Hz to $(N - 1)f_s/N$ Hz in steps of f_s/N Hz.
- For example, if we encode a single key

```
>> d5 = ttdial([5])
```

this will give 0.5 s, or 4096 samples of the signal for key 5. We would plot the spectrum with

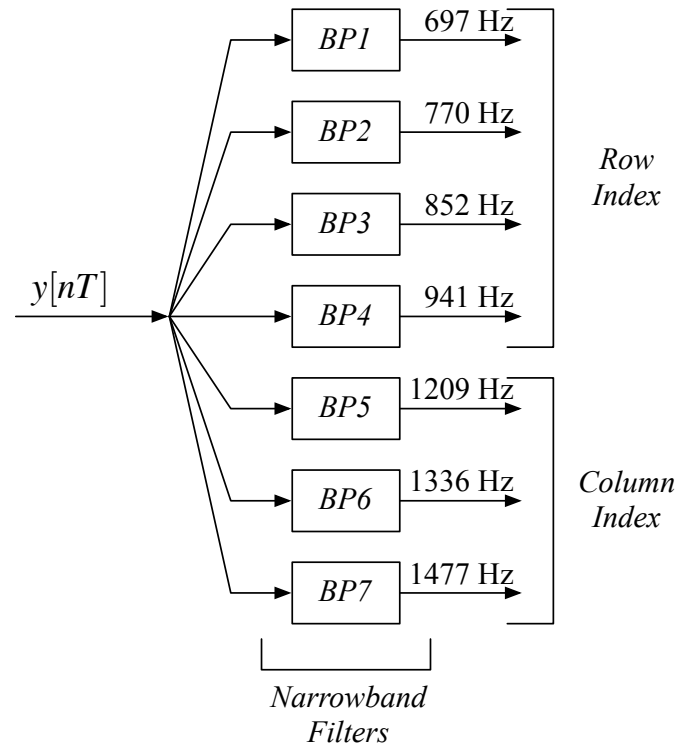
```
>> f = [0:4095]*fs/4096;  
>> plot(f,abs(fft(d5)/length(d5)))  
>> axis([500 1500 0 1]);
```



- Check digits 0, 3, 5 and 7 (one per row and column) to make sure your are generating the right frequencies.

Task 3: Narrowband Filter Bank to Decode DTMF

- The FFT allows us to identify any frequency in the input.
- Since there are only seven frequencies, an easier solution is to use a bank of narrowband filters, one per frequency.



- Since a key press is as short as 40 ms, the filter should be less than $0.04 * 8192 = 328$ samples long.
- Subtask 3.1: Show that a 256 point Hamming window is sufficiently selective. First note that the direct approach

```
>> h = hamming(256)';  
>> f = [0:255]*8192/256;  
>> plot(f, abs(fft(h))/256);  
>> axis([ 0 200 0 1]);
```

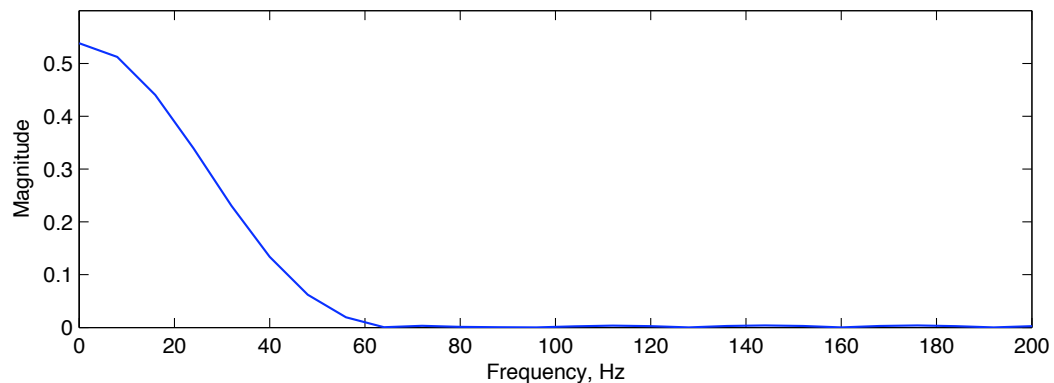
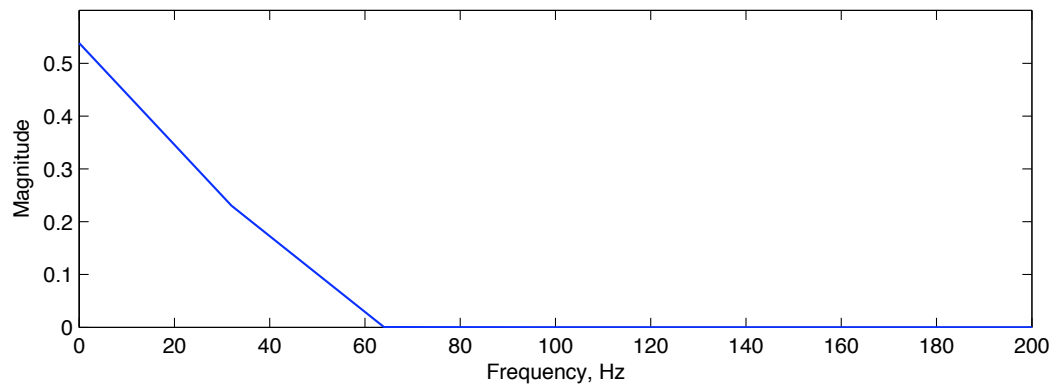
is hard to interpret. We need more spectral resolution.

- The spectral resolution is f_s/N . Since f_s is fixed, we need to increase N . We do this by adding zeros

```
>> hp = zeros(1,1024);  
>> hp(1:256) = h;  
>> fp = [0:1023]*8192/1024;  
>> plot(fp, abs(fft(hp))/1024);
```

```
>> axis([ 0 200 0 1]);
```

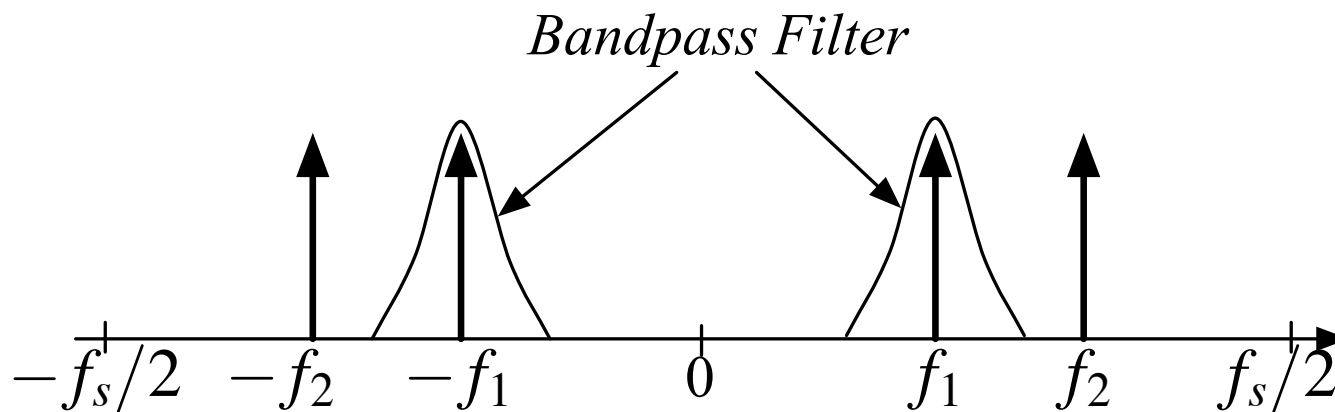
This simply interpolates, and doesn't add any new information. However, it does give a much more interpretable representation of the frequency response.



- Subtask 3.2: Generate narrowband bandpass filters by modulating the Hamming window

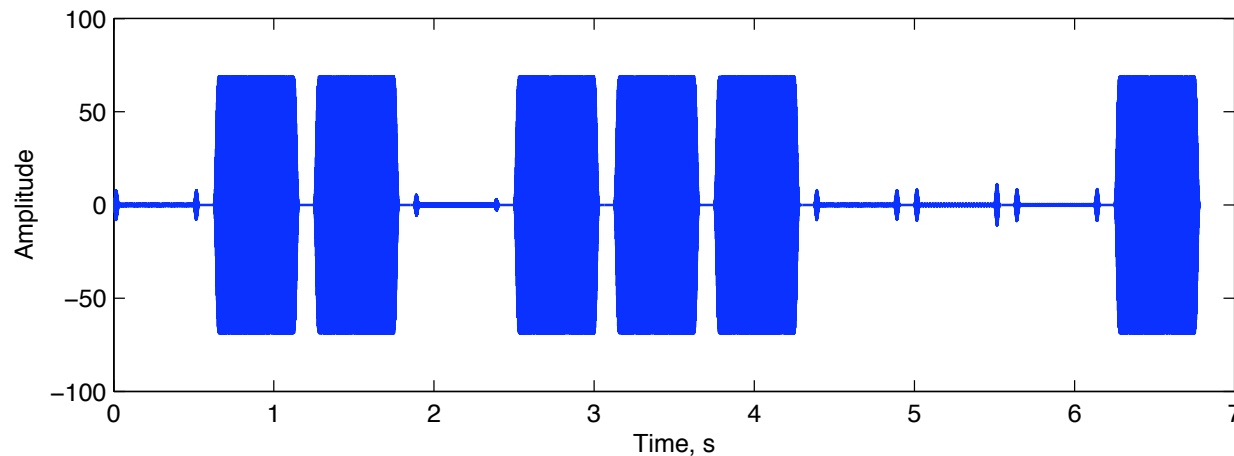
```
>> tf = [0:255]/8192;  
>> h1 = h.*cos(2*pi*f1*tf);
```

- This passes both sidebands of a sinusoid,



A cosine input gives a cosine out (plus a phase shift).

- Filter the signal for your phone number. For example, if we filter the signal for 1-650-555-1234 for the frequency 770 Hz, we get



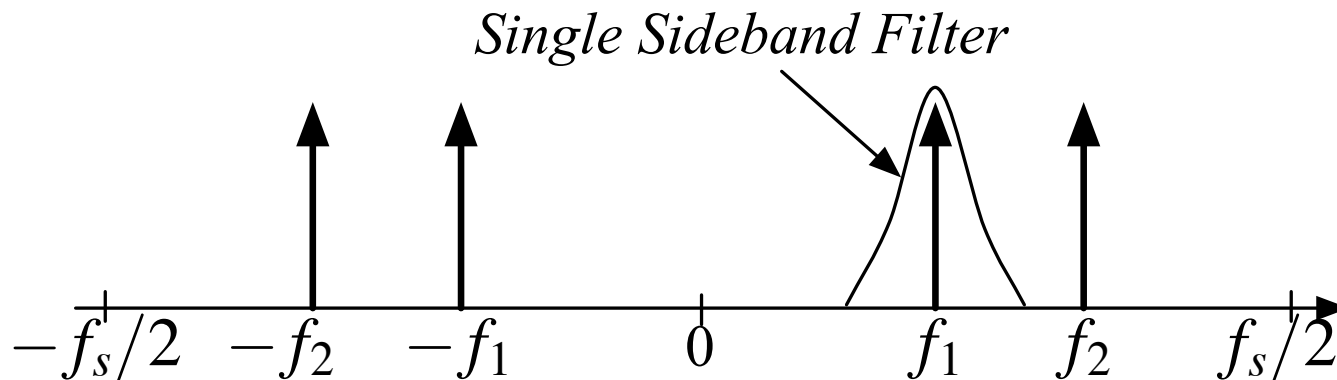
since 770 Hz is the frequency for key row 2, which includes keys 4, 5, and 6.

- After we filter the signal, we need to detect whether a key was pressed in a given interval.
- The key press we can consider a baseband signal, and the key frequency we can consider a carrier.
- We could use an approach like AM demodulation.

- A simpler approach is to use a filter that only passes one of the two sidebands of the cosine signal

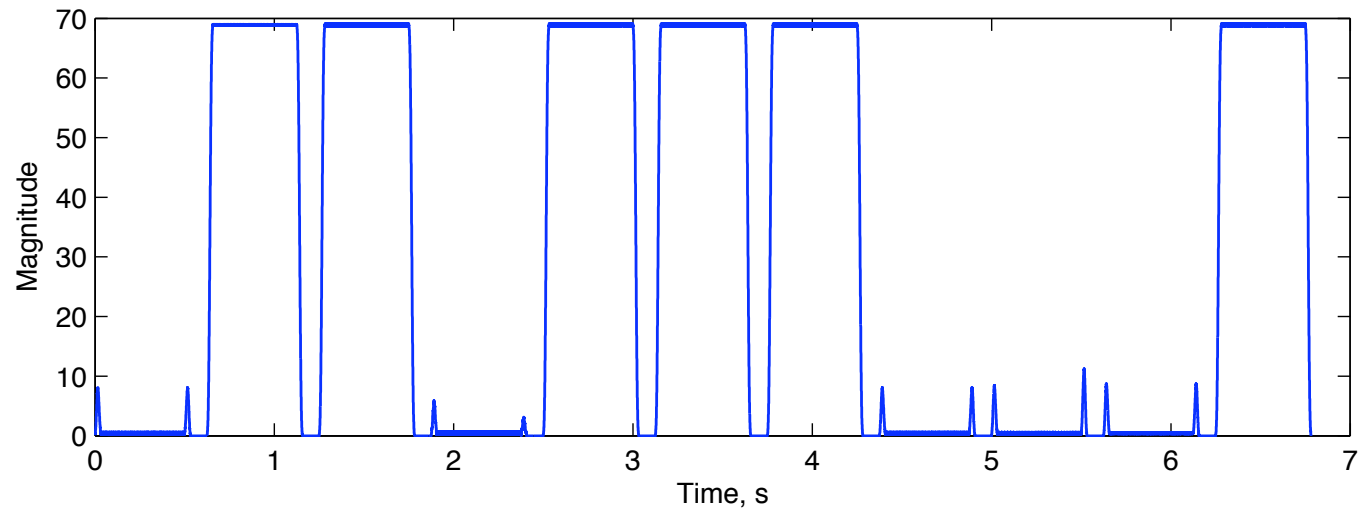
```
>> h1e = h.*exp(i*2*pi*f1*tf);
```

- This passes the upper sideband of a sinusoid,



A cosine input gives a complex exponential out.

The envelope is just the magnitude of the filter output. This looks much more like a logic signal.



Task 4: Decode the touch-tone waveform

Assume that the digits and the quiet period are at least 0.125 s, but could be longer. Write an m-file that takes the touch-tone signal as an input and

1. Finds the quiet separators,
2. Finds the intervals with key presses,
3. Identifies the two largest frequencies in each interval,
4. Looks up the digits these corresponds to, and
5. Returns the decoded phone number.

Several test cases are provided on the EE102B web site, in the `tt_test_cases.mat` file. Try your decoder on these waveforms, and report the results.

These aren't particularly interesting numbers, but if you are curious, you can look them up with Google.