EE 419 - Project 4
Discrete Fourier Transform and FFT Signal Processing 2



Learning Objectives:

Use Matlab to process discrete-time signals using the DFT/FFT, including:

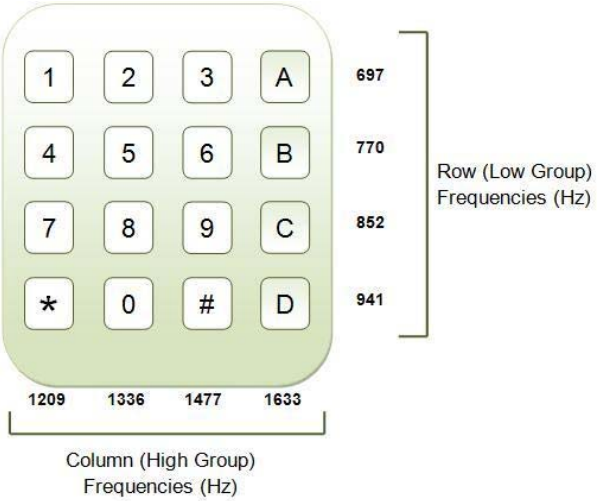
- **Spectral Peak Finding in the Presence of Noise**
- **FFT Spectrum Indexing**
- **Block Signal Processing of Non-Stationary Signals**

REMINDERS:
1) When you are finished, be sure to always copy your m-files to a place that you can retrieve them again in the future (copy them to a memory stick, email them to yourself, etc.).
2) Please delete your files from the Lab Computer (so that everyone else has to work as hard as you have!)

1) [Project] Telephone Touch Tone Decoder

DTMF (**Dual tone multi frequency**) is a multi-frequency tone encoding system used by the push button keypads in telephones and mobile devices to convey the number or key dialed by the caller. DTMF enables long distance signaling of dialed numbers over telephone lines and cellular networks within the limited bandwidth already established for voice communication.

DTMF, as the name suggests, uses a unique combination of two sine wave tones ("dual tones") to represent each key. The two tones for each key are at a unique combination of *row* and *column frequencies,* corresponding to the key's location in the standard layout of a telephone keypad. The row that a key occupies determines the DTMF *row frequency*, and the column position determines the DTMF *column frequency*.



Ref: DTMF (Dual Tone Multiple Frequency),
http://www.engineersgarage.com/tutorials/dtmf-dual-tone-multiple-frequency

A DTMF keypad (generator or encoder) generates a dual sinusoidal tone which is mixture of the row and column frequencies of the key pressed. The row frequencies occupy a lower range of frequencies. The column frequencies belong to a higher group of frequency values. The specific frequencies used for DTMF are chosen so that none have a harmonic relationship with the others, and so that mixing the frequencies does not produce sum or product frequencies that could mimic another valid tone. The high-group frequencies (the column tones) are slightly louder than the low-group to compensate for the high-frequency roll off of voice audio systems. The **allowed tolerance on the tone frequencies is ±1.5%** of their nominal values. (Ref: *ETSI ES 201 235-2 V1.1.1 (2000-09)* Specification of Dual Tone Multi-Frequency (DTMF) Transmitters and Receivers; Part 2: Transmitters)

YOUR JOB: To design a **DTMF signal decoder** that can take a **1/2 second** segment of audio (sound) samples and **correctly determine the key that was pressed** to produce the sound; even though the DTMF tones were produced by an inexpensive telephone handset (so the signals have harmonic-producing amplitude clipping) and have picked up a considerable amount of noise in the process of transmission. You are to use DFT/FFT signal processing techniques for your tone decoder, and some other logic to decide what key was pressed based on the tone frequencies present; all implemented in Matlab.

Your Matlab program (a function m-file) should accept an audio signal sample array name as an input parameter, and return a text variable with the key value [as in `return_value = '2';`]. Your program should display the spectrum of the signal you are decoding (perhaps using your plot_DFT_mag( ) function. Using the `text( )` Matlab function, your program should also display the decoded key character somewhere on the same DFT spectrum plot.

The audio signals for both uncorrupted and noisy signals are available in the *touchtones.mat* file posted on PolyLearn. These signals were all **sampled at 8 KHz** (the standard sampling rate for telephone voice signals). You should verify that your DTMF decoder works for the audio files for all of the individual keys; and that it is robust enough to operate properly even with the noisy signals.

Your report should include:

- a description of the signal processing techniques and algorithms that you used to determine the two tone frequencies that were present.
- a summary table of test results showing the audio signals (array variable names) from the *touchtones.mat* file that you tested, and the decoded key determined by your program.
- Include two DFT plot examples with your report – one from an uncorrupted (no noise) tone, and one from a different key's noisy tone. These should include the decoded key character, shown on the plot.
- Of course, also include all of your Matlab code (well documented with headers and comments).

Some Matlab commands that may prove helpful: text( ), find( ), max( ), sum( ), if, char( )

2) Touchtone Sequence Decoder

Now that you have a working digital decoder for a single touchtone button sound, use this as the starting point for a Matlab program that is able to decode **a complete 7 or 10-digit touchtone sequence** (as if a complete telephone number, possibly including the area code, is being dialed).

Functional Requirements:

- Your program should properly handle either **7-digit** (no area code) **or 10-digit** (with area code) touchtone sequences. (No need to handle country codes or long distance prefixes).
- Once 10 digits are detected, your program should stop looking for tones. (**Extra tones should be ignored**.) Once finished, you should verify that you ended up with either 7 or 10 digits; and indicate an "Error – Invalid Sequence" if it did not. (Extra tones beyond 10 should NOT generate an error; but only the first 10 keys should be returned by your function.]
- The decoding should work properly even when there are **harmonics** of the touchtones from clipping or other distortions, or in the presence of **noise**.
- Each individual tone will be **separated by a pause** (no tone) that also can last from 0.2 – 10 seconds. (Note: There may still be noise during the pause.)
- Your design should be able to function with **different time durations** for each of the individual touch tones (with each varying in length from 0.2 – 3.0 seconds)

Output:

- Your Matlab program should display the final decoded 7 or 10 digit button sequence in the Matlab Command Window, or an error message if an invalid sequence was provided.
- Your program should also display a **`spectrogram`**( ) plot (using the built-in Matlab function to plot it) of nearly all of the original tone samples. (You may have to truncate some samples from the end to use a fixed spectrogram segment block size.) From this, you should be able to visually pick out where each different tone starts and ends, and where the frequency peaks are in each segment (to help you check your results).
- Since you are probably using your tone decoder from part 1 above, your program will also produce spectrum plots of each sub-segment processed. These should also be helpful for debugging and verifying your program. You do not need to include more examples of these in this part of your project report.

Testing:

Use the touch tone sequence test sound arrays from PolyLearn (*DTMFseqtest.mat*) to verify your design.

Include all the results for the provided standard test cases, to verify the following features:

1. An **invalid length** sequence (invalid # of key tones)
2. A sequence with **> 10 individual tones**.
3. A 10-tone sequence with different, valid lengths for all tones and pauses, with **no noise**.
   Include at least one example of each of the following in the sequence:
   a. A minimum time length tone.
   b. A maximum time length tone.
   c. A minimum time length pause.
   d. A maximum time length pause.
4. A 10-tone sequence with different, valid lengths for all tones and pauses, **with noise** in all tones and pauses.
   Include at least one example of each of the following in the sequence:
   a. A minimum time length tone.
   b. A minimum time length pause.
   c. Three redundant key tones in a row (like: …8-8-8…)
5. A **7-tone**, noisy sequence that **begins with a no-tone (noisy) pause**.

Turn in:

1. **Descriptions of the algorithms** you used for
   a. Separating the individual tones out from the multiple tone/pause sequence. (Finding the beginning and ends of different tones/pauses, or removing redundant results)
   b. Limiting the decoding to 10 keystrokes.
2. **Explain your selection for the FFT size** used in your FFT signal processing: What value did you choose? How did you arrive at this particular value as your choice?
3. **Matlab code listing** (with appropriate comments)
4. **Description and results from the 5 standard test cases you ran**. (Using the table format below, describe the tone and pause durations in the sequence; whether noisy or not; the decoded key results; and whether any errors occurred**.) Include a `spectrogram`** plot for each test case as well. (To convert the spectrogram from color to grayscale for black/white printing, use the command `colormap gray` after selecting the figure with the `figure( figure# )` command.)

Test Case #:		**Test Case Description:**							**Test Array Name:**			
Sequence:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**(11)**	**Noisy?**
Key:												Tones Noisy? ↓
Key Tone Duration:												
Pause Duration:												
Decoded Key/Error												Pauses ⇑ Noisy?