EE102 Linear Systems
Spring 2010
MatLab Project #07

Assignment Date: 4-14-10 Due Date: 5-3-10

Two to three students may work on this assignment together and hand in one project write-up with both their names on it and share the grade.

Project Objective: To understand how touchtone telephone systems use signals of multiple frequencies to determine which key on the keypad has been selected. To detect and decode a touchtone telephone number encoded with dual-tone multi-frequency signaling using Fourier analysis. Modern telecommunications is based upon coding two frequencies for each tone associated with a telephone keypad character. The dual tone nature of the signal facilitates its detection throughout the network and for systems that utilize the tones for command and switching input required by users.

Background:

Dual-Tone Multi-Frequency (DTMF)

Dual-tone multi-frequency (DTMF) signaling is used for telecommunication signaling over analog telephone lines in the voice-frequency band between telephone handsets and other communications devices via a carrier's switching center. A carrier is the company that provides the communication lines for transmitting telephone calls from one telephone to another. The switching center or telephone switch, as it is colloquially referred to, is the computational processing center for dial service. Dial service is the most popular method and the telephone switch is the most popular device for routing calls from one telephone to another, generally as part of the public switched telephone network

Touch-Tone was the registered trademark of the Bell Telephone system in the United States to designate DTMF signaling until it was standardized by the International Telecommunication Union (ITU). The ITU is the eldest organization of the United Nations (UN) that remains in existence. Founded as the *International Telegraph Union* in Paris on May 17, 1865, it became part of the UN in 1947. It remains the UN's agency for information and communication technology in developing networks and services for governments and private companies. In 1956, its role expanded and it was renamed the International Telegraph and Telephone Consultative Committee (CCITT) (from the French *Comité Consultatif International Téléphonique et Télégraphique*). In 1993, it became known as the ITU-T. DTMF became ITU-T Recommendation Q.23, the multi-frequency system used for internal signaling within the telephone network.

DTMF signaling is used to instruct the telephone switch to route the requested telephone number, to issue commands to the telephone switch or to downstream switching systems, or related telephony equipment, for example, answering systems used by companies to route their calls within their own switched network. DTMF uses eight different frequency signals transmitted in pairs to represent a maximum of sixteen different numbers (0 – 9), symbols (# and *) and letters (A, B, C, and D). The letters were dropped for commercial telephones, but were used for a short time by the US military. The two symbols are referred to as the *octothorpe* or *pound* key, and the *asterisk* or *star* key, respectively. These keys are referred to as vertical service codes, e.g. *72, in the United States. They are used for a variety of functions dependent upon the carrier in some cases. Their more general usage is for caller ID, call forwarding, and call blocking.

The DTMF keypad is a 4×4 matrix where rows represent a *low* frequency, and columns represent a *high* frequency tone. See Table 1. The 1633 Hz frequencies are shown in Table 1 for completeness. Modern landline telephone sets and mobile phones use only the three lower high frequencies. When a single key is pressed, a sinusoidal tone, one for each of the two frequencies is transmitted. For example, when the number 4 key is pressed, the 770Hz low frequency and 1209 Hz high frequency are engaged.

Table 1
DTMF Telephone Keypad Frequencies and Layout

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

Additional tones are used to indicate the status of lines, equipment, and the outcome of calls. These special tones are standardized in each country. In the United States, a dual frequency system is used for busy signal, ringback tone, and dial tone. These are generated by the telephone switch, not by the handset.

Table 2
Special Tone Frequencies

Special Tone	Low Frequency	High Frequency
Busy	480 Hz	620 Hz
Ringback	440 Hz	480 Hz
Dial tone	350 Hz	440 Hz

According to the telephone standard, the technical specification is that a tone may not vary more than ±1.8% from its nominal frequency. If the tolerance is violated the switching center ignores the signal. High frequencies may be the same volume or louder as the low frequencies. If a loudness differential exists, then it cannot exceed 3 decibels.

Tones are required to have a minimum duration of 70 ms in the United States and the spacing between tones cannot generally exceed 5 seconds or the call is aborted.

The initial caller encodes the desired telephone number by using the keypad to produce the DTMF tones sending the signal to the carrier's telephone switch.

The telephone switch must properly detect and decode the signal to route the call onto its destination. The decoding operation was originally performed using tuned filter banks. Today the decoding is achieved by computers running a digital signal processing algorithm known as the Goertzel algorithm.

The Goertzel algorithm is a computational procedure, which is a faster method then computing the Discrete Fourier Transform directly. The algorithm achieves faster computational times by realizing the periodicity of the term $e^{j2\pi n}$. The impulse response for the algorithm is: $h[n]-(e^{2\pi\omega}+e^{-2\pi\omega})h[n-1]+h[n-2]=\delta[n]$. The Goertzel algorithm is beyond the scope of this MatLab project and EE102, but presented to the student to realize there are other more efficient computational techniques to detect and decode sinusoidal signals.

Filters

One purpose of a filter is frequency selection. We design a filter to pass a certain frequency or a band of frequency necessary for our purposes. This section explains two approaches, which are straightforward as compared to the Goertzel algorithm. However, we are not required to have the efficiency, nor the accuracy of a telephone system. Our simplified model will suit our purposes for selecting a particular frequency to identify the information we are required to obtain from the signal.

Approach 1: One approach to reduce the number of calculations is to use a typical low pass filter with a spectrum shifted to the desired resonant frequency. While this may introduce some error depending upon how it is implemented, the FIR filter will be stable. The impulse response of a linear phase FIR filter of length $2N+1$ is defined as: $h[n]=h_0[n-N]$, where $h_0[n]$ is the impulse response of a low pass filter with zero phase and symmetric with respect to $n=0$. The z-transform of this function is: $H(z)=z^{-N}H_0(z)$. The frequency response is defined as:

$$H(e^{j\omega}) = H_0(e^{j(\omega-\omega_0)}) + H_0(e^{j(\omega+\omega_0)}),$$

which is a bandpass filter. The impulse response of this filter is:

$$h[n] = (e^{-j\omega_0 n} + e^{j\omega_0 n})h_0[n]$$

No generality is lost by making $h_0[n] = 1$ and this yields:

$$h[n] = (e^{-j\omega_0 n} + e^{j\omega_0 n})$$

However, the gain of the filter need not be 1, so it may be specified to be an arbitrary constant, thus:

$$h[n] = A(e^{-j\omega_0 n} + e^{j\omega_0 n}) \qquad \text{Equation (1)}$$

,

where ω_0 is the desired center frequency of the bandpass filter. Further, define an integer M, such that M > N to specify the length of the FIR filter defined in Equation (1). This makes the bandpass filter a weighted averaging filter over M-points.

Approach 2: Another approach is to utilize Parseval's Theorem, which states that the total energy in a discrete signal is the summation of the square of the magnitude of the discrete-time frequencies of the DFT over the length of the sequence from 0 to N. In mathematical notation, for a discrete signal $x[n]$, having DFT $X[k]$, then:

$$\sum_{n=0}^{N-1}|x[n]|^2 = \frac{1}{N}\sum_{k=0}^{N-1}|X[k]|^2 \qquad \text{Equation (1)}$$

This relationship allows us to look exclusively at the energy as a means to determine the frequency in a signal. If the tones are free on noise, then the highest energy occurs at the relevant frequencies in the signal. If the tones are not free of noise, then filtering would be required to isolate and determine the tones.

We will assume for our purposes the tones are free of noise since we generated them ourselves. This way we can evaluate the sampled frequency spectrum of the energy to determine the frequencies present in the tone. Additional, we will require the tones to have approximately 120 ms duration, rather than the standard 70 ns, to make the processing a bit more efficient. Also, we will minimize the spacing (silence) between tones to keep the total signal length short.

Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is an efficient tool doing Fourier analysis for finite-duration signals. If the Fourier transform of $x[n]$ is $X\left(e^{j\omega}\right)$, then $X\left(e^{j\omega}\right)$ is a function of continuous variable in frequency, namely ω. For implementing and performing Fourier analysis on digital computers, a continuous variable requires a discrete representation. If $x[n]$ is a finite-duration signal, then the DFT pair is defined as:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{jk\left(\frac{2\pi}{N}\right)n} \qquad \text{DFT Synthesis Equation}$$

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-jk\left(\frac{2\pi}{N}\right)n} \qquad \text{DFT Analysis Equation}$$

The advantage of the DFT is its use on digital computers. Efficiency is dictated by access to memory and to the speed of the processing. Much innovative research and development has been dedicated to designing algorithms that are able to compute the DFT and inverse DFT as quickly and accurately as possible. These efficient, and in some cases very compact, algorithms are responsible for a host of applications from cellular communication, video, and graphics processing. An extremely rapid computation is called the Fast Fourier Transform (FFT). MatLab utilizes an effective computation strategy to implement its `fft` command.

Circular Convolution

Consider two finite duration discrete sequences $s_1[n]$ and $s_2[n]$ each having a length of N and with DFTs $S_1[k]$ and $S_2[k]$, respectively. What is the sequence $s_3[n]$ which results from $S_3[k] = S_1[k] \cdot S_2[k]$, which is the DFT of $s_3[n]$?

To determine the answer to this question, we first consider two periodic sequences $\tilde{s}_1[n]$ and $\tilde{s}_2[n]$, with period N, having discrete Fourier series coefficients $\tilde{S}_1[k]$ and $\tilde{S}_2[k]$, such that $\tilde{S}_3[k] = \tilde{S}_1[k] \cdot \tilde{S}_2[k]$. The product is itself periodic and the periodic sequence $\tilde{s}_3[n]$ with Fourier series coefficients $\tilde{S}_3[k]$ is $\tilde{s}_3[n] = \sum_{m=0}^{N-1} \tilde{s}_1[m]\tilde{s}_2[n-m]$. This appears as a convolution sum, something we might expect since a multiplication in the frequency domain corresponds to a convolution in the sample domain as has been learned for linear transforms. However, this is not a linear (or aperiodic) convolution as the sequences are periodic with period N and the summation is over one period. Thus, the convolution is an example of *periodic convolution*. The differences with respect to linear convolution are: a) the sum is finite occurring over the range $0 \le m \le N$-1; b) the values of $[n - m]$ are in the interval $0 \le m \le N$-1 and are periodic in m outside the interval; and c) duality theorem implies that $\tilde{s}_3[n] = \tilde{s}_1[n] \cdot \tilde{s}_2[n]$ has discrete Fourier series coefficients

$$\widetilde{S}_3[n] = \frac{1}{N}\sum_{r=0}^{N-1}\widetilde{S}_1[r]\widetilde{S}_2[n-r].$$

Now reconsider the two *finite-duration* sequences $s_1[n]$ and $s_2[n]$ both of length N having DFTs $S_1[k]$ and $S_2[k]$, respectively. If we consider the finite-duration sequence $s_3[n]$ as one period of a periodic sequence, we are able to use the result from *periodic convolution* just derived. Thus,

$$S[k] = \begin{cases} \widetilde{S}[k], & 0 \le k \le N-1 \\ 0, & otherwise \end{cases}$$

and likewise:

$$s_3[n] = \frac{1}{N}\sum_{m=0}^{N-1}\widetilde{s}_1[m]\widetilde{s}_2[n-m], \quad 0 \le n \le N-1 \qquad \text{Equation (2).}$$

Since $s[n]$ has finite length N and there is no overlap between samples $s[n]$ and $s[n-rN]$ for different values of r, we can associate $s[n]$ with a periodic sequence $\widetilde{s}[n]$ defined as:

$$\widetilde{s}[n] = \sum_{m=-\infty}^{\infty} s[n-mN]$$

or:

$$s[n] = \begin{cases} \widetilde{s}[n], & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

where the periodic sequence $\widetilde{s}[n]$ may be written as $\widetilde{s}[n] = s[(n \text{ modulo } N)]$ or more compact as $\widetilde{s}[n] = s[((n))_N]$.

If we substitute this into Equation (2) we have:

$$s_3[n] = \frac{1}{N}\sum_{m=0}^{N-1} s_1[((m))_N]s_2[((n-m))_N], \quad 0 \le n \le N-1$$

Since $((m))_N = m$ for $0 \le n \le N-1$, the above equation may be rewritten as:

$$s_3[n] = \frac{1}{N}\sum_{m=0}^{N-1} s_1[m]s_2[((n-m))_N], \quad 0 \le n \le N-1. \qquad \text{Equation (3)}$$

This equation differs from discrete linear convolution. In linear convolution, the computation of a sequence $s[n]$ is the summation involving the multiplication of one

sequence by a time-reversed and linearly shifted version of another sequence. This equation differs from discrete periodic convolution as the two sequences are not periodic. Therefore, Equation (3) is the definition of *circular* convolution. Actually, it is defined as the N-point circular convolution. This is an important distinction as it defines the convolution for two sequences each having a finite-length which does not exceed *N*, where the sequences are shifted modulo *N*. The convolution described by Equation (3) has the second sequence circularly time reversed with respect to the first sequence. [Note: in MatLab when the finite-length sequences are of different length, it eases calculation to add zeros on to the end of the shorter sequences to have the two sequences have equal length. This is called zero-padding. It does not change the computation.] Circular convolution, like linear and periodic convolution, is commutative. This convolution is a multiplication in the frequency domain.

As an example of circular convolution, consider two identical discrete rectangular pulses of length M.

$$x_1[n] = x_2[n] = \begin{cases} 1, & 0 \le n \le M-1 \\ 0, & otherwise \end{cases}$$

Let $M = 6$. If we let $N = 6$ for the DFT length (although it need not be as it may be selected for the best resolution of the frequency spectrum), then N = L and the N-point DFTs are:

$$X_1[k] = X_2[k] = \sum_{n=0}^{N-1} W_N^{kn} = \begin{cases} N, & k = 0 \\ 0, & otherwise \end{cases}$$

where $$W_N^{kn} = \left(e^{-j\left(\frac{2\pi}{N}\right)} \right)^{kn}$$

Multiplying $X_1[k]$ and $X_2[k]$ we obtain: $X_3[k] = X_1[k]X_2[k] = \begin{cases} N^2, & k = 0 \\ 0, & otherwise \end{cases}$.

Taking the inverse transform yields the sample domain signal:

$$x_3[n] = N, \quad 0 \le n \le N-1$$

If you sketch out the two 6 impulse long sequences, it is clearly seen that as the sequence $x_2[((n-m))_N]$ is rotated with respect to the first sequence $x_1[n]$, the sum of the products will always be equal to N.

[Note: The result of the circular convolution of two N-point DFTs may be the same as linear convolution of the corresponding finite-length sequences depending upon the length of the DFT with respect to the length of the finite-length sequences. This may be explained in terms of aliasing. In the example of circular convolution above if the two

finite-length sequences were $2M$ in length, where they were padded with M zeros, a linear convolution would result.

Sampling

Analog systems signals are processed utilizing the entire signal. However, in digital computations, only samples of the signals are required to perform the desired calculations. We restrict ourselves to periodic, or uniform, sampling. This is typical of engineering practice. To obtain a discrete sequence $x[n]$ we take samples of an analog signal $x_{ana\log}(t)$ every T seconds:

$$x[n] = x_{ana\log}(nT), \qquad -\infty \le n \le \infty .$$

It is not important for the purposes of our discussion here the method of taking samples, rather we are more interested in establishing the relationship between the analog signal and the discrete sampled signal. The interval T between successive samples is called the sampling period. We define:

$$F_s = \frac{1}{T}$$

as the sampling rate (in samples per second) or the sampling frequency (in Hertz). Since $t = nT$, then:

$$t = nT = \frac{1}{F_s}$$

and this demonstrates that there is a relationship between the analog signal frequency F_a (or ω_a) and the discrete signal frequency f (or ω). Whether the choice is F_a or ω_a for analog or f or ω for discrete is determined by the choice of units.

A straightforward method to establish the relationship is to utilize an example. Consider an analog signal

$$x_a(t) = A\cos(2\pi F_a t + \theta) .$$

Sampling this signal at a uniform rate of $F_s = \frac{1}{T}$ produces then signal

$$x_a(nT) \equiv x[n] = A\cos(2\pi F_a nT + \theta)$$

or after substituting for T

$$x_a(nT) \equiv x[n] = A\cos\left[\frac{2\pi F_a n}{F_s} + \theta\right].$$

Since a discrete time signal results from the uniform sampling we define the relationship between the analog signal frequency and the discrete frequency as:

$$f = \frac{F_a}{F_s} \text{ or } \omega = \omega_a T \, .$$

When the variable f is used to describe the frequency relationship it is sometimes referred to as the *normalized* frequency. Note that the ranges on the frequency variables are: $-\infty \le F_a \le \infty$ and $-\pi \le \omega \le \pi$. Looking at the range for the discrete frequency $-\pi \le \omega \le \pi$, and substituting for $\omega = 2\pi f$ and then $f = \frac{F_a}{F_s}$ yields

$$-\frac{F_s}{2} \le F_a \le \frac{F_s}{2}$$

which implies:

$$F_s \ge \left|2F_a\right| \qquad \text{Equation (4)}$$

The sampling rate F_s at which an analog signal must be sampled depends upon the frequencies that make up the signal. Fourier analysis is the tool that allows determination of the frequencies that compose a signal. Equation (4) is called the Nyquist Criterion after Harry Nyquist an early contributor to Information Theory who discovered the relationship. If a signal contains a frequency that is no higher than F_a, then the signal can (in theory) be exactly reconstructed from its samples, provided that the sampling frequency F_s is selected to be more than twice the value of F_a.

Project

Part A – Understanding the impact of the N-point DFT and Periodic, Circular, and Linear Convolution

Please be as exact in your explanation as these questions will have significant weight in the grading of the MatLab project.

1. The m-file `DFT_Impact.m` is shown in the box labeled DFT Impact m-File. Run the m-file to view the output. The output is two figures. Figure 1 is three subplots comparing the N-point DFTs of three signals, while Figure 2 is four subplots comparing as many signals each generated as either a periodic, circular, or linear

2. A finite-duration signal is used to represent either a periodic signal where one period is equal in length to the finite-duration signal or a finite-duration signal itself. Please explain the relationship to and the impact on the output by changing the value of N of the N-point DFT as a result of running the commands that are displayed in the Figure 1 output of the file DFT_Impact.m. A review of the Help files, which describe the `fft` command in the on-line MatLab documentation, may help to provide some insight in preparation of your answer.

3. The output displayed in Figure 2 from the computation of the file DFT_Impact.m shows the results of several MatLab calculations involving the `fft` and `conv` commands. What happens if the N-point version of the `fft` command is used rather than the version of the `fft` command used in the m-file? How do you explain the results of using the Matlab `conv` command and using the DFT for convolution? Remember that the DFT is a sampled version of the frequency spectrum generated by the Discrete-time Fourier transform or from a duality property perspective it is the Discrete Time Fourier series.

DFT Impact m-file

```
% DFT_Impact.m
% N-point DFT Computation
s1 = [1 1 1 1 1 1];
S1 = fft(s1, 6);
S2 = fft(s1, 12);
S3 = fft(s1, 120);
figure;
subplot(3, 1, 1);
stem(1:length(S1), S1);
title('S1');
xlabel('k (where \omega = 2\pik/N)');
subplot(3, 1, 2);
stem(1:length(S2), S2);
title('S2');
xlabel('k (where \omega = 2\pik/N)');
subplot(3, 1, 3);
stem(1:length(S3), S3);
title('S3');
xlabel('k (where \omega = 2\pik/N)');
% Periodic, Circular, and Linear convolution comparison
s4 = conv(s1, s1);
s5 = [s1 zeros(1, 3)];
S5 = fft(s5);
s6 = ifft(S5.*S5);
s7 = [s1 zeros(1, 5)];
S7 = fft(s7);
s8 = ifft(S7.*S7);
s9 = [s1 zeros(1, 12)];
S9 = fft(s9);
s10 = ifft(S9.*S9);
figure;
subplot(4, 1, 1);
stem(1:length(s4), s4);
title('s4');
```

```
xlabel('n');
subplot(4, 1, 2);
stem(1:length(s6), s6);
title('s6');
xlabel('n');
subplot(4, 1, 3);
stem(1:length(s8), s8);
title('s8');
xlabel('n)');
subplot(4, 1, 4);
stem(1:length(s10), s10);
title('s10');
xlabel('n)');
```

Part B – Generate a touchtone telephone number

1. We will restrict our efforts to considering only the tones associated with the numeric values on the telephone keypad for this and the next part of the MatLab project. Create an m-file function to `generate_DTMF` that generates a DMTF tone for any of the ten telephone numeric characters in Table 1. Each tone should have a length of 1000 and be sampled at a frequency of 8192. When the tone is an input to the MatLab command `sound(tone, Fs)`, where Fs = 8192, the output should sound like a tone you hear when you push a button on a telephone dial. Verify that the tones generated by your function sound like actual telephone tones. The best way to represent the tone is tone$[n] = \sin[\omega_{low} n] + \sin[\omega_{high} n]$ and in MatLab as:

   `tone = sin(omegaLow*n) + sin(omegaHigh*n)`, where `omegaLow` and `omegaHigh` are the particular low and high frequencies DTMF pair associated with the telephone numeric keypad value. Specify the range of n to be $1 \le n \le 1000$ to generate a tone that is approximately 120 ms in length. The structure of your function is to be

```
function tone = generate_DTMF (key);
% key = is a character representing the key, 'c'
% tone =  the output tone generated
```

2. Create an m-file function called `silence_DTMF`, which generates an interval of silence. The length of interval of silence is 100 which will approximate a period of silence of approximately 9 ms and have the same sampling considerations of Fs = 8192 as the tone in step 1. Note while this is below the actual telephones spec it will simplify the computation.

3. Generate a complete encoded 10-digit phone number with tones and intervals of silence from the m-files in steps 1 and 2. One approach is to create a single row vector called `phoneNumber` as an input to the sound command. For example:

Sample Program

```
% Sample program to generate DTMF tones for a 10-digit telephone number
% Program simulates the tone's of Professor Lasser's
% office telephone number
Fs = 8192;
```

```
phoneNumber = [generate_DTMF('6') ...
               silence_DTMF ...
               generate_DTMF('1') ...
               silence_DTMF ...
               generate_DTMF('7') ...
               silence_DTMF ...
               generate_DTMF('6') ...
               silence_DTMF ...
               generate_DTMF('2') ...
               silence_DTMF ...
               generate_DTMF('7') ...
               silence_DTMF ...
               generate_DTMF('4') ...
               silence_DTMF ...
               generate_DTMF('9') ...
               silence_DTMF ...
               generate_DTMF('7') ...
               silence_DTMF ...
               generate_DTMF('7') ...
               ];
sound(phoneNumber, Fs);
```

Use your own telephone number to generate the sequence of tones or an arbitrary telephone number such as (617) 555-1234.

Part C – Detect a touchtone telephone number

1. The objective of this step is to create an m-file function named `detect_tone` which determines whether or not an input tone is a valid DMTF tone and outputs the correct character if the tone is valid. The MatLab `fft` command is very helpful to do this. Use the `fft(x, N)`, where `N` = 2048, to compute a 2048 evenly spaced frequency sample of tone x. The samples are space at interval of $\omega_k = \frac{2\pi k}{N}$, where k varies from $0 \le k \le N-1$, and again $N = 2048$. Therefore you can determine which value of ω_k maps to k and is the closest frequency to the particular touchtone frequency pair. You can use either a band pass filter or energy methods to detect the frequencies present in the signal. A quick overview of the approach that could be used is:
    a. Detect and eliminate the silence part of the signal and to use it as a means to isolate the individual tones composing the entire signal. Generate a signal of two tones and properly determine the amount of silence between the two tones and the frequency-pair associated with each tone. A tone which is approximately 120 ms has a finite duration of 1000 samples and silence is approximately 9 ms has 100 samples.
    b. Attempt to detect a single tone first, and then generalize for the remaining nine tones. Select *either* a band pass approach *or* an energy approach to implement to do the frequency detection. There is no extra credit for doing both approaches, one will suffice.
        i. Using a band pass approach requires you to:

1. Eliminate the silence portions of the signal so that you can analyze the individual tones composing the complete signal
2. Convert the impulse response to the frequency domain
3. Multiply it with a finite interval sequence which is the one tone under consideration and is a subset of the entire signal
4. Do this for each frequency to determine the frequency-pair of the tone
5. Do this for all the tones in the signal

ii. Using an energy approach requires calculation of the value of the energy at index *k* which maps to frequency ω_k, namely $\left|X\left(e^{j\omega_k}\right)\right|^2$

1. Eliminate the silence portions of the signal so that you can analyze the individual tones composing the complete signal
2. Convert the tone under consideration, which is a finite interval and a subset of the entire signal to the frequency domain
3. Either: a) map the energy in the tone to the known energy in a known tone, or b) determine the frequencies (locations in the spectrum at *k*) of maximum energy for the tone under consideration. [Note: option b) may be the easiest of all, the most straightforward, and fewest lines of MatLab code.]
4. Do this for each frequency to determine the frequency-pair of the tone
5. Do this for all the tones in the signal

c. Compete the `detect_tone` function which has a function prototype of:

```
function key = detect_tone (signal);
  % signal = the input signal
  % key = the touchtone key associated with tone
```

2. Using the `detect_tone` function from step 1 to write an m-file which has a signal representing a ten digit telephone number as its input and outputs the character string of the numeric characters for the associated telephone number. For example, the m-file function might be instantiated as:

`phoneDigits = detect_number(toneSignal);`,

where `phoneDigits` is a 1 by 10 vector of numeric digits of the telephone number which was encoded by the digital signal `toneSignal`. On the BlackBoard website in the MatLab Project section you will find a MatLab mat file associated with the first 8 letters of your last name. If your last name is less then 8 letters, the balance of the 8 letters filename has been completed with 0's.The mat-file contains two ten digit encoded telephone numbers appropriately named `phoneNumber1` and `phoneNumber2`. The ten digit telephone numbers have been encoded using the same protocol which you used to encode a telephone number in Part B, Questions 1 and 2, namely a sequence of length 1000 to represent a tone and a sequence between each tone to represent silence. Each student has a unique pair of telephone numbers. Each signal has been encoded using DTMF tones and silence to represent a telephone number. The individual digits of the first telephone number, `phoneNumber1`, add

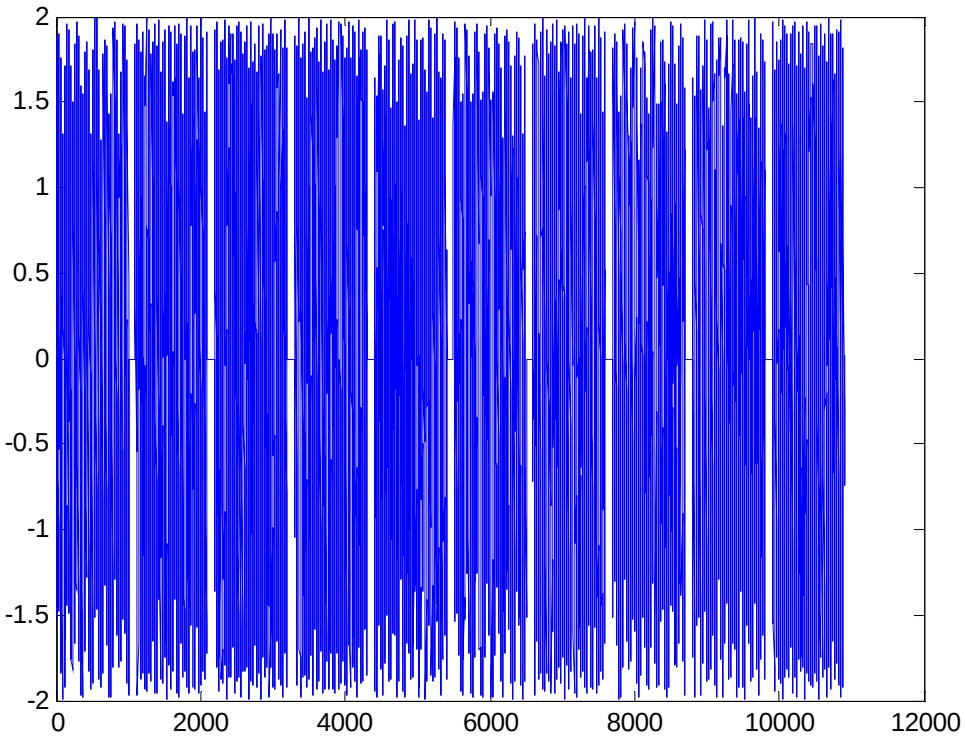
to a checksum of 40. This will help you determine if your DTMF detect_tone function is working correctly. The second signal is a 10-digit telephone number, but there is no checksum. Determine the 10 digit telephone numbers in your mat file. To import the data from a mat file into Matlab you must use the `load` function. The syntax of the `load` function is different depending on whether you use on a command line or in a function. If more than one student is working on this project, each student must determine the telephone numbers in the mat file associated with their own name. In other words, the group may submit one project write up that decodes both telephone numbers for each student in the group. Each student set of telephone number is unique.

References:

1. Buck, J.M., Daniel, M.M., Singer, A.C., *Computer Explorations in Signals and Systems using MatLab*, Prentice-Hall, 2002.
2. Moore, H., *MatLab for Engineers*, Prentice-Hall, 2007.
3. Lathi, B. *Linear Systems and Signals, 2nd Ed*., Oxford, 2005.
4. Etter, D.M., *Engineering Problem Solving with MatLab, 2nd Ed*., Prentice-Hall, 1997.
5. Kamen, E.W., Heck, B.S., *Fundamentals of Signals and Systems using the Web and MatLab, 3rd Ed*, Prentice-Hall, 2007.
6. Oppenheim, A.V., Willsky, A.S., Nawab, S.A., *Signals and Systems, 2nd Ed*., Prentice-Hall, 1997.
7. Soliman, S.R., Srinath, M.D., *Continuous and Discrete Signals and Systems, 2nd Ed*., Prentice-Hall, 1998.
8. Haykin, S., *Adaptive Filter Theory*, Prentice-Hall, 2003.
9. *Getting Started with MatLab*, The MathWorks, Inc., 2006.
10. Oppenheim, A. Schafer, R., *Discrete-Time Signal Processing*, Prentice-Hall, 1975.
11. Bergen, S.W.A., Antoniou, A., "An efficient closed-form design method for cosine modulated filter banks using window functions", Signal Processing, (87), p811-823, 2007.
12. Dutta Roy, S.C., Ahuja, S.S., "Frequency Transformations for Linear-Phase Variable-Cutoff Digital Filters", IEEE Trans. Circuits and Systems, 26 (1), p73-75, 1979.
13. Jarske, P, Neuvo, Y., Mitra, S.K., "A Simple Approach to the Design of Linear Phase FIR Digital Filters with Variable Characteristics", Signal Processing, (14), p313-326, 1988.
14. Rabiner, L.R., Crochiere, "A Novel Implementation for Narrow-Band FIR Digital Filters", IEEE Trans. Acoustics, Speech, and Signal Processing, (23) 5, p457-464, 1975.
15. Schenker, L., "Pushbutton Calling with a Two-Group Voice Frequency Code", The Bell System Technical Journal, (39) 1, 1990.

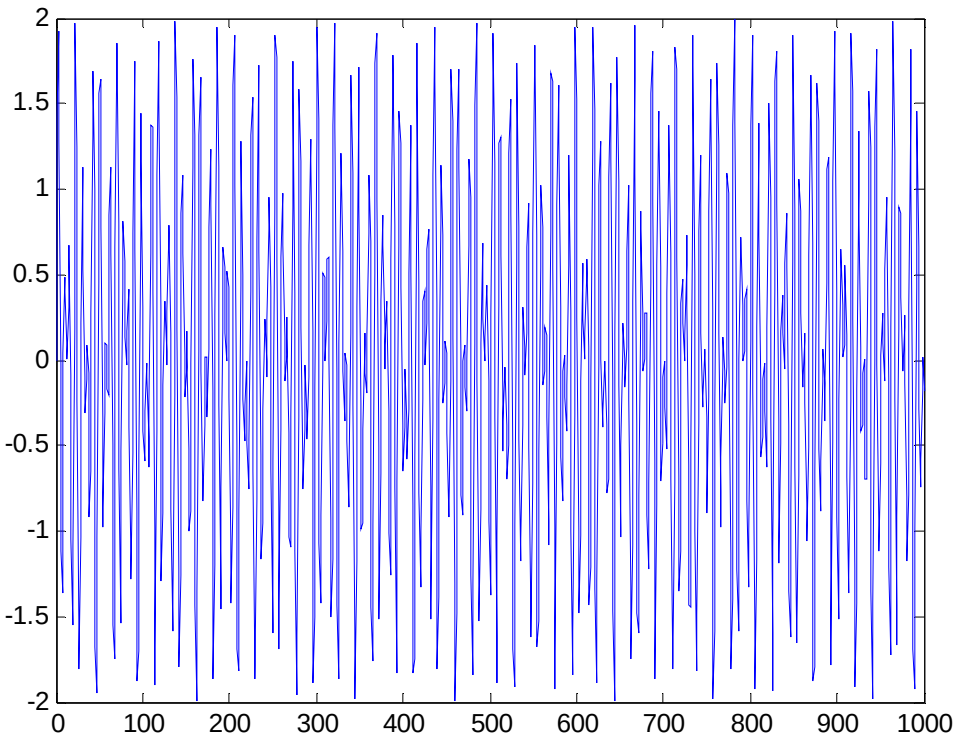
Suggestion Appendix

1. A telephone number signal encoded in the mat file for each student is shown in the Appendix Figure 1. Note that the dark colors are separated by short white spaces. The dark color areas in the plot are the encoded tones and the short white space is encoded silence. Note that silence only occurs between tones, not after the tenth digit. As such the over length of the signal is 10,900.



Appendix Figure 1 – Encoded 10 digit telephone signal

2. A single tone is shown in Appendix Figure 2. This figure shows a single encoded digit or tone. This subset of the overall signal is a combination of the two sinusoid frequencies that make up a single "dual tone". The length of the tone is 1000 samples



Appendix Figure 2 – Single encoded digit or tone

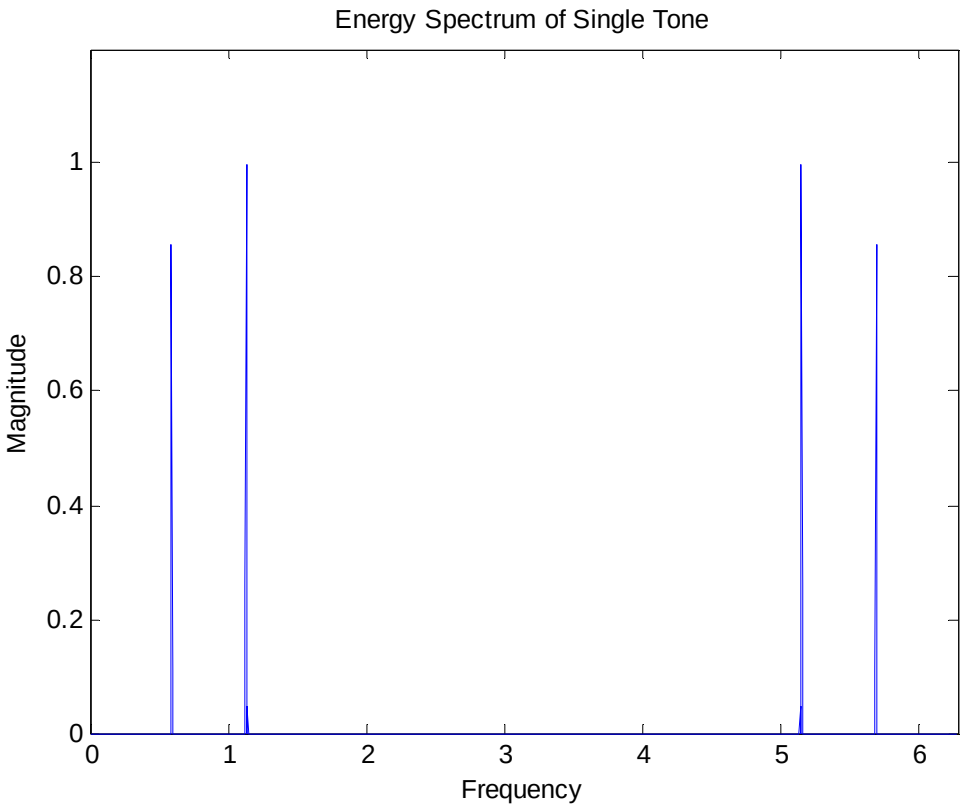
3. As a recommendation, the energy approach to detect the tones may be easier to implement than using a band pass filter. However, the choice is yours. In either case, after generating the spectrum, it might be easier to detect the tones by normalizing the spectrum to 1 by dividing the spectrum by it maximum value. This will allow a simple way to find the largest values. Note, that when MatLab computes the spectrum the frequencies range from 0 to $2\pi k / N$, where N is the length of the spectrum and k ranges from 0 to $(N - 1)$. As such, there will be four frequencies of significant magnitude, representing the positive and negative frequency components in the spectrum of the two frequencies used to generate a tone. However, when MatLab generates the spectrum, it produces the spectrum from 0 to 2π, not $-\pi$ to π.If you leave the spectrum in the frequency order as generated by MatLab, the location of the two lowest frequencies are the frequencies you are required to detect to determine and match the digit (or keypad) value. Why is this the case?

4. As a problem-solving approach consider the following set of tasks (albeit not a complete list) to help guide you toward your solution.

    a. Load the complete encoded telephone signal
    b. Generate an algorithm to separate the 10 tones from the 9 silence separations

c. Generate the energy spectrum of the tone
d. Generate a filter bank, that is, a set of functions where each one can detect a single set of the dual frequencies associated with each tone, and associate those frequencies with a digit
e. Process each tone through the filter bank to determine the associated digit
f. Collect the digit in a string to output as the complete ten digit telephone number

Appendix Figure 3 shows the magnitude of the energy spectrum of a single tone in the frequency range from 0 to 2π rad/sec.



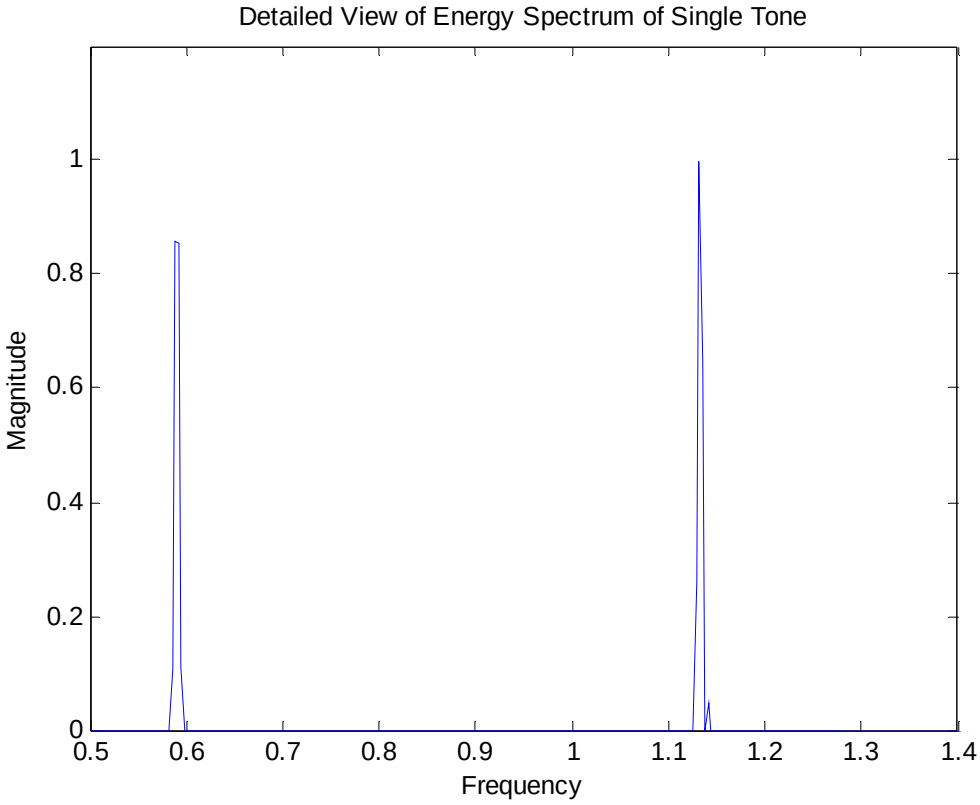
Appendix Figure 3 – Normalized Frequency Spectrum of Single Tone

Appendix Figure 4 shows the magnitude of the energy spectrum of a single tone in the frequency range from 0.5 to 1.4 rad/sec. Note the fine peaks of the spectrum and that they appear around 0.59 and 1.13 rad/sec which map to the digit 6. Recall the relationship between the analog signal frequency and the discrete frequency as:

$$f = \frac{F_a}{F_s} \text{ or } \omega = \omega_a T \, .$$

In Appendix Figure 5 a MatLab command window calculation is show to calculate the digital frequency ω using the above equation. Note that the analog frequencies F_a from

Table 1 are in Hertz and the sampling frequency F_s, also in Hertz, is 8192. This is consisted for the MatLab `sound` command.



Appendix Figure 4 – Detailed Normalized Frequency Spectrum of Single Tone

Designing an algorithm (or MatLab function) which can detect the magnitude of the energy spectrum at the particular frequencies and map this to the dual tone pairs for the associated digit as we have just done visually is a straightforward approach that might be utilized to decode a telephone number.

```
>> Fa = 770;
>> Fs = 8192;
>> w = 2*pi*Fa/Fs

w = 0.5906

>> Fa = 1477;
>> w = 2*pi*Fa/Fs

w = 1.1328
```

Appendix Figure 5 – MatLab Calculation of Digital Frequency