GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2026 Spring 2018

Lab #3: Synthesis of Sinusoidal Signals: Music and DTMF Synthesis

Date: 5 Feb.- 8 Feb. 2018

You should read the Pre-Lab section of the lab and do all the exercises in the Pre-Lab section before your assigned lab time.

Verification: The In-Lab Exercise section of each lab must be completed **during your assigned Lab time** and the steps marked *Instructor Verification* must also be signed off **during the lab time**. When you have completed a step that requires verification, simply raise your hand and demonstrate the step to the TA or instructor. After completing the in-lab exercise section, turn in the verification sheet to your TA *before leaving the lab.*

It is only necessary to turn in Section 4 as the lab homework for this lab. More information on the lab report format can be found on `t-square` under the **INFO** link. Please **label** the axes of your plots and include a title for every plot. In order to reduce "orphan" plots, include each plot as a figure *embedded* within your report. For more information on how to include figures and plots from MATLAB in your report file, consult the **INFO** link on `t-square`, or ask your TA for details.

Forgeries and plagiarism are a violation of the honor code and will be referred to the Dean of Students for disciplinary action. You are allowed to discuss lab exercises with other students, but you cannot give or receive any written material or electronic files. In addition, you are not allowed to use or copy material from old lab reports from previous semesters. Your submitted work must be your own original work.

---

1 Introduction

The objective of this lab is to introduce more complicated signals that are related to the basic sinusoid. These signals which implement frequency modulation (FM), to be discussed in the next lab, and amplitude modulation (AM), to be studied in this and next labs, are widely used in communication systems such as cellular telephones, radios and television broadcasts. In addition, they can be used to create interesting sounds that mimic musical instruments and useful tones. There are a number of demonstrations on the CD-ROM that provide examples of these signals for many different conditions.

CD-ROM

FM Synthesis

2 Pre-Lab

We have spent a lot of time learning about the properties of sinusoidal waveforms of the form:

$$x(t) = A\cos(2\pi f_0 t + \varphi) = \Re\left\{\left(Ae^{j\varphi}\right)e^{j2\pi f_0 t}\right\} \tag{1}$$

Now, we will extend our treatment of sinusoids to more complicated signals composed of sums of sinusoidal signals. The objective of this lab is to learn to how short-duration sinusoids can be concatenated to make longer signals that "play" musical notes and dial telephone numbers. The resulting signal can be analyzed to show its time-frequency behavior by using the *spectrogram.*

2.1 Summation of Sinusoidal Signals

If we add several sinusoids, each with a different frequency (f_k), we cannot use the phasor addition theorem, but we can still express the result as a summation of terms with complex amplitudes via:

$$x(t) = \sum_{k=1}^{N} A_k \cos(2\pi f_k t + \varphi_k) = \Re\left\{ \sum_{k=1}^{N} \left(A_k e^{j\varphi_k} \right) e^{j2\pi f_k t} \right\} \tag{2}$$

where $A_k e^{j\varphi_k}$ is the complex amplitude of the k^{th} complex exponential term. The choice of f_k will determine the nature of the signal—for amplitude modulation or beat signals we would pick two or three frequencies f_k that are very close together, see Chapter 3.

Therefore, it will be necessary to establish the connection between musical notes, their frequencies, and sinusoids. A secondary objective of the lab is to learn the relationship between the synthesized signal, its spectrogram and the musical notes. There are several specific steps that will be considered in this lab:

1. Synthesizing a single short-duration sinusoid with a MATLAB M-file, and adding it to an existing long signal vector.
2. Mapping piano keys to explicit frequencies using the "equally-tempered" definition of twelve notes within each octave.
3. Concatenating many short-duration sinusoids with different frequencies and durations.
4. *Spectrogram:* Analyzing the long (concatenated) signal to display its time-frequency spectral content.

2.2 Beat Signals: Summing Two Sinusoids with A Small Frequency Difference

In the section on beat notes in Chapter 3 of the text, we discussed signals formed as the product of two sinusoidal signals of slightly different frequencies; i.e.,

$$x(t) = B\cos(2\pi f_\Delta t + \varphi_\Delta)\cos(2\pi f_c t + \varphi_c) \tag{3}$$

where f_c is the (high) center frequency, and f_Δ is the (low) frequency that modulates the envelope of the signal. An equivalent representation for the beat signal is obtained by rewriting the product as a sum:

$$x(t) = A_1\cos(2\pi f_1 t + \varphi_1) + A_2\cos(2\pi f_2 t + \varphi_2) \tag{4}$$

It is relatively easy to derive the relationship between the frequencies $\{\, f_1,\ f_2 \,\}$ and $\{\, f_c,\ f_\Delta \,\}$.

2.3 Piano Keyboard

The exercise in Section 3 of this lab will consist of synthesizing a simple musical passage.[1] Since these "music" signals use sinusoidal tones to represent piano notes, a quick introduction to the layout of the piano keyboard is needed. A piano keyboard is divided into octaves—the notes in one octave being twice the frequency of the notes in the next lower octave. The white keys in each octave are named A through G. In order to define the frequencies of all the keys, one key must be designated as the reference. Usually, the reference note is the A above middle-C, called A-440 (or A_4) because its frequency is exactly 440 Hz.[2] Each

[1]If you have little or no experience reading music, don't be intimidated. Only a little music knowledge is needed. On the other hand, the experience of working in an application area where you must quickly acquire new knowledge is a valuable one. Many real-world engineering problems have this flavor, especially in signal processing which has such a broad applicability in diverse areas such as geophysics, medicine, radar, speech, etc.

[2]In this lab, we are using the number 40 to represent middle C. This is somewhat arbitrary; for instance, the Musical Instrument Digital Interface (MIDI) standard represents middle C with the number 60.

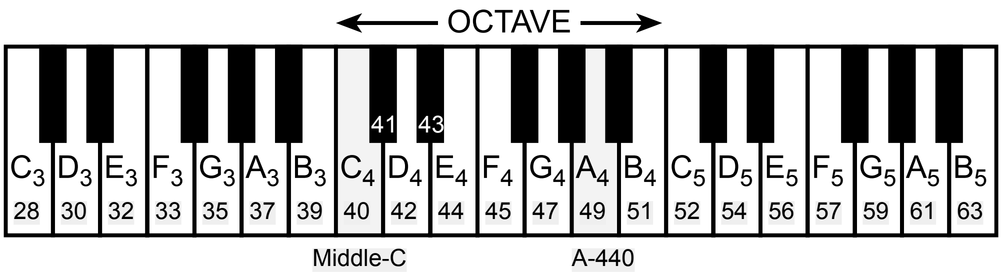

Figure 1: Layout of a piano keyboard. Key numbers are shaded. The notation C_4 means the C-key in the fourth octave, which is middle-C.

octave contains 12 notes (5 black keys and 7 white) and the ratio between the frequencies of neighboring notes is constant. As a result, this ratio must be $2^{1/12}$. Since middle C is 9 keys below A-440, its frequency is approximately 261 Hz, i.e., $261 \approx 440 \times 2^{-9/12}$. Consult the SP-First text for even more details.

Musical notation shows which notes are to be played and their relative timing (half, quarter, or eighth notes). Figure 2 shows how the keys on the piano correspond to notes drawn in musical notation. The subscript denotes the octave where the note lies (octaves start with C). The white keys are labeled as A, B, C, D, E, F, and G; but the black keys are denoted with "sharps" or "flats." A sharp such as $A^{\#}$ is one key number larger than A; a flat is one key lower, e.g., $A_4^{\flat}$ (A-flat) is key number 48.

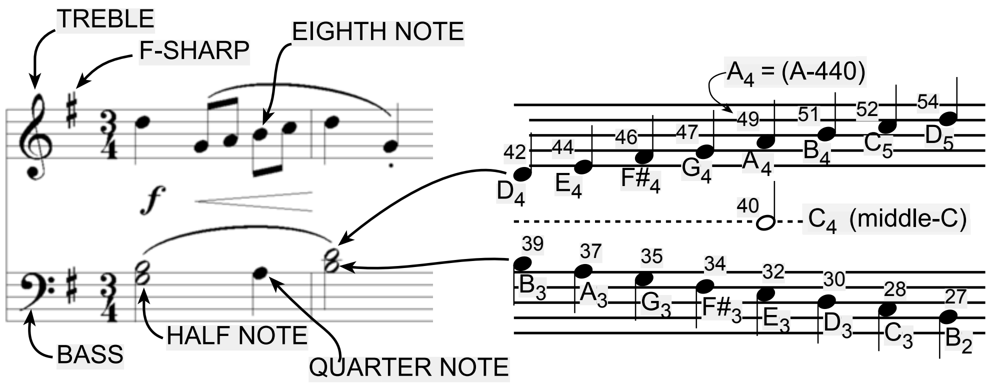

Figure 2: Musical notation is a time-frequency diagram where vertical position indicates which note is to be played. Notice that the shape of the note defines it as a half, quarter or eighth note, which in turn defines the duration of the sound.

Another interesting relationship is the ratio of fifths and fourths as used in a chord. Strictly speaking the fifth note should be 1.5 times the frequency of the base note. For middle-C the fifth is G, but the (equally-tempered) frequency of G is 391.99 Hz which is not exactly 1.5 times 261.63. It is very close, but the slight detuning introduced by the ratio $2^{1/12}$ gives a better sound to the piano overall. This innovation in tuning is called "equally-tempered" or "well-tempered" and was introduced in Germany in the 1760's and made famous by J. S. Bach in the "Well Tempered Clavier."

Thus, you can use the ratio $2^{1/12}$ to calculate the frequency of notes anywhere on the piano keyboard. For example, the E-flat above middle-C (black key number 43) is 6 keys below A-440, so its frequency should be $f_{43} = 440 \times 2^{-6/12} = 440/\sqrt{2} \approx 311$ Hz.

2.4 Telephone Touch Tone Dialing

Telephone touch-tone keypads generate *dual tone multiple frequency* (DTMF) signals to represent digits in a phone number when dialing a telephone. When any key is pressed, the sinusoids of the corresponding row and column frequencies (see Fig. 3) are generated and summed, hence dual tone. As an example, pressing the **5** key generates a signal containing the sum of the two tones at 770 Hz and 1336 Hz together.

FREQS	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**

Figure 3: Extended DTMF encoding table for Touch Tone dialing. When any key is pressed the tones of the corresponding column and row are generated and summed. Keys A-D (in the fourth column) are not implemented on commercial and household telephone sets, but might be used in some special signaling applications, e.g., military communications.

The frequencies in Fig. 3 were chosen (by the design engineers) to avoid harmonics. No frequency is an integer multiple of another, the difference between any two frequencies does not equal any of the frequencies, and the sum of any two frequencies does not equal any of the frequencies.[3] This makes it easier to detect exactly which tones are present in the dialed signal in the presence of non-linear line distortions.[4]

2.5 Synthesizing Long Signals

Long signals can be created by joining together many sinusoids. When two signals are played one after the other, the composite signal could be created by the operation of *concatenation.* In MATLAB, this can be done by making each signal a row vector, and then using the matrix building notation as follows:

```
xx = [ xx, xxnew ];
```

where `xxnew` is the sub-signal being appended. The length of the new signal is equal to the sum of the lengths of the two signals `xx` and `xxnew`. A third signal could be added later on by concatenating it to `xx`.

2.6 Preliminary Topic: Spectrograms

It is often useful to think of a signal in terms of its spectrum. A signal's spectrum is a representation of the frequencies present in the signal. For a constant frequency sinusoid, the spectrum consists of two spikes, one at $\omega = 2\pi f_0$, the other at $\omega = -2\pi f_0$. For a more complicated signal the spectrum may be very interesting, e.g., the case of FM, where the spectrum components are time-varying. One way to represent the time-varying spectrum of a signal is the *spectrogram* (see Chapter 3 in the text). A spectrogram is produced by estimating the frequency content in short sections of the signal. The magnitude of the spectrum over individual sections is plotted as intensity or color over a two-dimensional domain of frequency and time.

When unsure about a command, use `help`.

There are a few important things to know about spectrograms:

[3]More information can be found at: `http://www.genave.com/dtmf.htm`, or search for "DTMF" on the internet.

[4]A recent paper on a DSP implementation of the DTMF decoder, "A low complexity ITU-compliant dual tone multiple frequency detector", by Dosthali, McCaslin and Evans, in *IEEE Trans. Signal Processing*, March, 2000, contains a short discussion of the DTMF signaling system. You can get this paper on-line from the GT library, and you can also get it at `http://www.ece.utexas.edu/~bevans/papers/2000/dtmf/index.html`.

1. In MATLAB the function `spectrogram` will compute the spectrogram. Type `help spectrogram` to learn more about this function and its arguments. The `spectrogram` function used to be called `specgram`, and had slightly different defaults—the argument list had a different order, and the output format always defaulted to frequency on the vertical axis and time on the horizontal axis.

2. If you are working at home, you might not have a `spectrogram` function because it is part of the *Signal Processing Toolbox*. In that case, use the function `plotspec(xx,fs,...)` which is part of the *SP-First Toolbox* which can be downloaded from

   `http://dspfirst.gatech.edu/matlab/SPFirstMATLAB.html`

   • Note: The argument list for `plotspec()` has a different order from `spectrogram` and `specgram`. In `plotspec()` the third argument is optional—it is the *section length* (default value is 256) which is often called the *window length*. In addition, `plotspec()` does not use color for the spectrogram; instead, darker shades of gray indicate larger values with black being the largest.

3. A common call to the MATLAB function is `spectrogram(xx,1024,[],[],fs,'yaxis')`. The second argument[5] is the *section length* (or window length) which could be varied to get different looking spectrograms. The spectrogram is able to "see" very closely spaced separate spectrum lines with a longer (window) section length,[6] e.g., 1024 or 2048.

In order to see a typical spectrogram, run the following code:

```
fs=8000; xx = cos(2000*pi*(0:1/fs:0.5)); spectrogram(xx,1024,[],[],fs,'yaxis'); colorbar
```

or, if you are using `plotspec(xx,fs)`:

```
fs=8000; xx = cos(2000*pi*(0:1/fs:0.5)); plotspec(xx,fs,1024); colorbar
```

Notice that the spectrogram image contains one horizontal line at the correct frequency of the sinusoid.

3 In-Lab Exercise: Beat, Piano and DTMF Signals

The instructor verification sheet may be found at the end of this lab.

3.1 MATLAB Structure for Beat Signals

A beat signal is defined by five parameters $\{ B, f_c, f_\Delta, \varphi_c, \varphi_\Delta \}$ as shown in Section 2.2 so we can represent it with a MATLAB structure that has seven fields (by including start and end times), as shown in the following template:

[5]If the second argument of `spectrogram` is made equal to the "empty matrix" then the default value used, which is the maximum of 256 and the signal length divided by 8.

[6]Usually the window (section) length is chosen to be a power of two, because a special algorithm called the FFT is used in the computation. The fastest FFT programs are those where the FFT length is a power of 2.

```
sigBeat.Amp = 10;     %-- B in Equation (3)
sigBeat.fc = 480;     %-- center frequency in Eq. (3)
sigBeat.phic = 0;     %-- phase of 2nd sinusoid in Eq. (3)
sigBeat.fDelt = 20;  %-- modulating frequency in Eq. (3)
sigBeat.phiDelt = -2*pi/3;  %-- phase of 1st sinusoid Eq.~(\ref{Labeq:beatSigSum})
sigBeat.t1 = 1.1;     %-- starting time
sigBeat.t2 = 5.2;     %-- ending time
%
%----- extra fields for the parameters in Equation (4)
%
sigBeat.f1      %-- frequencies in Equation (4)
sigBeat.f2      %--
sigBeat.X1  %-- complex amps for sinusoids in Equation (4)
sigBeat.X2  %--     derived from A's and phi's
%
sigBeat.values %-- vector of signal values
sigBeat.times  %-- vector of corresponding times
```

(a) Write a MATLAB function that will add fields to a `sigBeatIn` structure. Follow the template below:

```
function sigBeatSum = sum2BeatStruct( sigBeatIn )
%
%--- Assume the five basic fields are present, plus the starting and ending times
%--- Add the four fields for the parameters in Equation (4)
%
%  sigBeatSum.f1, sigBeatSum.f2, sigBeatSum.X1, sigBeatSum.X2
```

(b) Produce a beat signal with two frequency components: one at 570 Hz and the other at 610 Hz. Use a longer duration than the default to hear the *beat frequency* sound. Use the feature discussed in Section 2.6 to generate a spectrogram plot of the beat signal here. Demonstrate the plot and sound to your lab instructor or TA.

Instructor Verification (separate page)

3.2 Note Frequency Function

Complete the following M-file to produce a desired note for a given duration. Your M-file should be in the form of a function called `key2sinus.m`. Your function should have the following form:

```
function [xx,tt] = key2sinus(keynum, amp, phase, fsamp, dur )
% KEY2SINUS  Produce a sinusoidal waveform corresponding to a
%       given piano key number
%
%  usage:  xx = key2sinus(keynum, amp, phase, fsamp, dur )
%
%        xx = the output sinusoidal waveform
%        tt = vector of sampling times
%    keynum = the piano keyboard number of the desired note
%  amp, phase = sinusoid params
%     fsamp = sampling frequency, e.g., 8000, 11025 or 22050 Hz
%       dur = the duration (in seconds) of the output note
%
tt = 0:(1/fsamp):dur;
freqKey =  ????       %<=============== fill in this line
Xphasor = ????        %<=============== fill in this line
xx = real( Xphasor*exp(j*2*pi*freqKey*tt) );
```

For the "`freqKey = `" line, use the formulas given in Sect. 2.3 to determine the frequency for a sinusoid in terms of its key number. You should start from a reference note (A-440 is recommended, or middle-C) and solve for the frequency based on this reference. Notice that the "`xx = real( )`" line generates the actual sinusoid as the real part of a complex exponential at the proper frequency.

Instructor Verification (separate page)

3.3 Generating a C-Major Scale

Use the feature discussed in Section 2.5 to m file that calls the key2sinus function studied in Section 3.2 to synthesize a C-Major scale shown in Figure 1, starting with 0.1 second of silence and 0.4 second of the C_4 note, and so on, finally ending with 0.1 second of silence, 0.4 second of the C_5 note and 0.1 second of silence. Make sure you have the time index correct. How long in seconds in total is the generated C-Major scale signal? Demonstrate the signal plot, sound and spectrogram of the C-Major scale signal to your lab instructor or TA. Note the blurry effect showing at the boundaries between notes and silence segments.

Instructor Verification (separate page)

3.4 Dual Tone Signals

For the DTMF synthesis each key-press generates a signal that is the sum of two sinusoids. For example, when the key **3** is pressed, the two frequencies are 697 Hz and 1477 Hz, so the generated signal is the sum of two sinusoids which could be created in MATLAB via

```
Ts = 0.3e-3;      %- Sampling period = 3 msec
fsamp = 1/Ts;     %- Sampling rate
tt = 0:1/fsamp:0.3;
DTMFsig = cos(2*pi*???*tt+rand(1)) + cos(2*pi*???*tt+rand(1));  %- Use random phases
xx = zeros(1,round(2/Ts));  %- pre-allocate vector to hold DTMF signals
n1 = round(0.6/Ts);   n2 = n1+length(DTMFsig)-1;
xx(n1:n2) = xx(n1:n2) + DTMFsig;
%-- soundsc(xx,fsamp);  %- Optional: Listen to a single DTMF signal
plotspec(xx,fsamp); grid on %- View its spectrogram
```

Demonstrate the signal plot, sound and spectrogram of the signal, when pressing DTMF key **8**, to your lab instructor or TA. Explain what is similar to or different from the beat signal you have created earlier?

Instructor Verification (separate page)

4 Lab Report: Beats, Piano Keys and DTMF

For the lab exercise and lab report, you will synthesize some AM and DTMF signals. In order to verify that these signals have the correct frequency content, you will use the spectrogram. Your lab report should discuss the connection between the *time-domain* definition of the signal and its *frequency-domain* content.

4.1 Beat Note Spectrograms

Beat notes have a simple time-frequency characteristic in a spectrogram. Even though a beat note signal may be viewed as a single frequency signal whose amplitude envelope varies with time, *the spectrum or spectrogram requires an* ***additive combination*** which turns out to be the sum of two sinusoids with different constant frequencies.

(a) Use the MATLAB function(s) written in Section 3.1 to create a beat signal defined via: $b(t) = 50\cos(2\pi(30)t + \pi/4)\cos(2\pi(800)t)$, starting at $t = 0$ with a duration of 4.04 s. Use a sampling rate of $f_s = 8000$ samples/sec to produce the signal in MATLAB. Use `testingBeat` as the name of the MATLAB structure for the signal. Plot a very short time section to show the amplitude modulation.

(b) Derive (mathematically) the spectrum of the signal defined in part (a). Make a sketch (by hand) of the spectrum with the correct frequencies and complex amplitudes.

(c) Plot the *two-sided spectrogram* of using a (window) section length of 512 using the commands[7]:

```
plotspec(testingBeat.values+j*1e-12,fs,512); grid on, shg
```

Comment on what you see. Can you see two spectral lines, i.e., horizontal lines at the correct frequencies in the spectrum found in the previous part? If necessary, use the zoom tool (in the MATLAB figure window).

4.2 Touch-Tone Dial Function

Write a function, `DTMFdial.m`, to implement a Touch-Tone dialer based on the frequency table defined in Fig. 3. A skeleton of `DTMFdial.m` is given in Fig. 4.
In this exercise, you must complete the dialing code so that it implements the following:

1. One of the inputs to the function is the sampling frequency `fs` which should be high enough, e.g., at $T_s = 0.2$ ms, to avoid aliasing (more in Chapter 4). The other input to the function is a vector of characters, each one being equal to one of the key names on the telephone. The n-th character is `keyNames(n)`. The MATLAB array called `TTkeys` containing the key names is a 4×4 matrix that corresponds exactly to the keyboard layout in Fig. 3. To convert any key name to its corresponding row-column indices, consider the following example:

```
[jrow,jcol] = find('3'==TTkeys)
```

2. The output should be a vector of signal samples containing the DTMF sinusoids—each key being the sum of two sinusoids. Remember that each DTMF signal is the sum of two (equal amplitude) sinusoidal signals. The duration of each tone pair should be exactly 180 ms, and a gap of silence, exactly 48 ms long, should separate the DTMF tone pairs. These times can be declared as fixed variables in the code for `DTMFdial`, i.e., there is no need to pass the durations as input variables.

[7]Use `plotspec` instead of `specgram` in order to get a linear amplitude scale rather than logarithmic. Also, use the tiny imaginary part to get the negative frequency region.

```
function xx = DTMFdial(keyNames,fs)
%DTMFDIAL  Create a signal vector of tones that will dial
%            a DTMF (Touch Tone) telephone system.
%
% usage:  xx = DTMFdial(keyNames,fs)
%  keyNames = vector of CHARACTERS containing valid key names
%        fs = sampling frequency (1/Ts)
%   xx = signal vector that is the concatenation of DTMF tones.
%
TTkeys =  ['1','2','3','A';
           '4','5','6','B';
           '7','8','9','C';
           '*','0','#','D'];
TTcolTones = [1209,1336,1477,1633];  %-- in Hz
TTrowTones = [697,770,852,941];
numKeys = length(keyNames);
durDualTone = ?  %-- in seconds
LenDualTone = ?
durSilence = ?
LenSilence = ?
xx = ....  %- initialize xx to be long enough to hold the entire output
n1 = 1;
for kk=1:numKeys
   [jrow,jcol] = find(....  %- which key?
   ... more code to make the dual-tone signals
   ... precede each dual-tone signal with a short interval of silence
end
```

Figure 4: Skeleton of DTMFdial.m, a Touch-Tone phone dialer. Complete this function by adding more lines of code to generate the dual-tone sinusoids. The vector of characters needed for the input keyNames is actually a string, e.g., '9786501234ABCD'.

3. The frequency information is given as two 4-element vectors (TTcolTones and TTrowTones): one contains the column frequencies, the other has the row frequencies. You can translate a key such as the **6** key into the correct location in these vectors by using MATLAB's find function. For example, the **6** key is in row 2 and column 3, so we would generate sinusoids with frequencies equal to TTrowTones(2) and TTcolTones(3).

   Also, consult the MATLAB code in Section 3.4 for hints about writing DTMFdial.m.

4. You could implement error checking so that an illegitimate key name is rejected.

Your function should create the appropriate tone sequence to dial an arbitrary phone number. In fact, when played through a speaker into a conventional telephone handset, the output of your function will be able to dial the phone.[8] For verification, please use plotspec to show the time-frequency analysis of the generated signal for the key sequence '268*073A' when the sampling period is $T_s = 0.2\,\text{ms}$.

[8]In MATLAB the demo called phone also shows the waveforms and spectra generated in a Touch-Tone system.

Lab #3
ECE-2026 Spring-2018
INSTRUCTOR VERIFICATION SHEET

Turn this page in to your lab grading TA before the end of your scheduled Lab time.

Name: ______________________ gtLoginUserName: ____________ Date: ______

Part 3.1 Write the MATLAB code for generating the beat signal. Show the signal, spectrogram and sound.

Verified:________________ Date/Time:_________

Part 3.2 Write MATLAB code for calculating frequency from a given piano key number, and for setting the complex amplitude.

```
freqKey =

Xphasor =
```

Verified:________________ Date/Time:_________

Part 3.3 Synthesize the C-Major scale and make a spectrogram.
Explain features in the spectrogram at the segment boundaries.

Verified:________________ Date/Time:_________

Part 3.4 Synthesize the DTMF digit **3** and make a spectrogram.
Explain features in the spectrogram.

Verified:________________ Date/Time:_________

Lab #3
ECE-2026 Spring-2018
LAB REPORT QUESTION

Turn this page in to your lab grading TA at the very beginning of your next scheduled Lab time.

Name: ____________________________ gtLoginUserName: ______________ Date: _______

Part 4.1: Beat Note Spectrogram (Write your code and cut and paste the spectrogram below)

Part 4.2: DTMF Tone Synthesis (Write your code and cut and paste the spectrogram below)