ECE 280L Fall 2024

Laboratory 4:
Touch Tone Phone

Contents

1 Introduction

This project introduces a practical application where sinusoids are used to transmit information and correlation is used to recover this information: a dual-tone multi-frequency (DTMF) encoder/decoder. DTMF is commonly used with touch-tone phones. Using MATLAB, Simulink, and the Audio System Toolbox, you will design and implement a real-time system that simulates the standard touch-tone phone system.

2 Objectives

The objectives of this project are to:

- Gain an understanding of the standard DTMF phone dialing system
- Design and implement a software-based DTMF tone generator (encoder)
- Design and implement a software-based DTMF receiver that runs in real-time (decoder)

These objectives will be achieved by designing and implementing two major subsystems: a DTMF encoder and a DTMF decoder. For the encoder, you will write MATLAB code capable of generating the tones representing an arbitrary phone number. For the decoder, you will use a Simulink block diagram to design a system that translates the tones back into numbers (which will be displayed using a scope).

3 Background

The standard touch-tone telephone system operates using dual-tone multi-frequency (DTMF) signals to transmit information. When any key on the telephone key pad is pressed, a pair of sinusoids (corresponding to the row and column of the number in Table 1) are generated and added together, producing a "dual tone." For example, when you press the '6' key, a signal containing the sum of 770 Hz and 1477 Hz is generated.

Freqs	**1209 Hz**	**1336 Hz**	**1477 Hz**
697 Hz	1	2	3
770 Hz	4	5	6
852 Hz	7	8	9
941 Hz	*	0	#

Table 1. DTMF frequencies

The frequencies in Table 1 were chosen by engineers such that no two are harmonically-related (i.e., no integer multiples) and none of the frequencies is equal to either the sum or difference of two frequencies. Having different numbers represented by frequencies that are not integer multiples of each other makes the decoding/number identification process simpler and more reliable. Sum and difference frequencies can appear in a system where there are non-linear distortions, so this design makes the system more robust to line distortions/noise. Finally, two tones per number were used so that the voice could never accidentally be interpreted as the press of a button.

NOTE: There is actually one more frequency used in a DTMF system (1633 Hz). This frequency is required for the representation of the symbols 'A', 'B', 'C', and 'D'. As our system will only need to represent the digits $0 - 9$, this frequency will not be included in our model.

4 Equipment Needed

- PC with MATLAB, Simulink with the Audio System Toolbox
- Speakers

5 Pre-Laboratory Assignment

Based on the Background information provided, answer the following questions on Gradescope:

1. When you press the '2' key, a signal containing which frequencies will be generated?
2. When you press the '0' key, a signal containing which frequencies will be generated?
3. A signal containing the frequencies 770 Hz and 1336 Hz would be generated by pressing which key?, and
4. A signal containing the frequencies 852 Hz and 1477 Hz would be generated by pressing which key?

6 Instructions

6.1 Exercise #1: DTMF Encoder (Dialer)

The encoding process is relatively simple. As you can see in Table 1, you simply need to generate a signal of the form:

$$y[n] = \sin(2\pi f_1 n) + \sin(2\pi f_2 n)$$

where f_1 and f_2 correspond to the row and column frequencies associated with the number being dialed. For example, for the digit '1', $f_1 = 697$ Hz and $f_2 = 1209$ Hz.

Your first task is to write a MATLAB function, `dtmfdial.m`, which will generate the appropriate DTMF tones for an arbitrary phone number. A skeleton of the function is provided below (and on Canvas):

```
function PhoneNum = dtmfdial(KeyNames, fs)
% DTMFDIAL  Create a signal vector of tones which will dial a
%           DTMF telephone system.
%
% usage:    PhoneNum = dtmfdial(KeyNames, fs)
%           KeyNames = a vector of characters containing valid
%           key names
%           fs = sampling frequency
%           PhoneNum = signal vector that is the concatenation
%           of DTMF tones
%
dtmf.Keys = ['1', '2', '3';
             '4', '5', '6';
             '7', '8', '9';
             '*', '0', '#'];
dtmf.colTones = ones(4,1)*[1209, 1336, 1477];
dtmf.rowTones = [697; 770; 852; 941]*ones(1,3);
```

In order to complete the function, we will first implement an encoder which operates on a single key at a time, after which we will generalize to accept a vector of characters as input.

1. Use MATLAB's `find` function to determine the indeces of `dtmf.colTones` and `dtmf.rowTones` which correspond to the two frequencies associated with the input key.
   (a) The desired frequencies (See Table 1) are already encoded in the matrices `dtmf.colTones` and `dtmf.rowTones`, with the corresponding frequency located at the same position as the key itself in the matrix `dtmf.Keys`.
   (b) For the key '6', for example, we can search for its position in `dtmf.Keys` with the code:

```
[R,C] = find(dtmf.Keys == '6')
```

which returns `R = 2` and `C = 3`. Since the matrices with row and column tones have the same shape as `dtmf.Keys`, `R` and `C` also give us the indeces we need to search within `dtmf.colTones` and `dtmf.rowTones`.

2. Create a time vector, `tNote`, for the tone pair lasting exactly 0.5 seconds, sampling at a rate of $f_s = 8000$ Hz.
   Note: 0.5 seconds is a bit long for an actual application, but using 0.5 seconds will make your real-time results in Exercise #2 easier to evaluate.

3. Create the note corresponding to the key input by sampling from the equation $y[n] = \sin(2\pi f_1 n) + \sin(2\pi f_2 n)$ at times defined by `tNote`.

   (a) The values f_1 and f_2 can be found by indexing `dtmf.colTones` and `dtmf.rowTones` with `R` and `C`.

4. For multiple key inputs, create a loop that iterates over your KeyNames vector and apply the key encoding explained in steps 1-3 to each one of them.

5. To play your sound, add all your encoded key input sounds to a vector (similarly to what you did for your song in lab 1).
   Note: You will probably want to create a pause note and add it between each key sound so that they are distinguishable.

Your function should create the appropriate tone sequence to dial an arbitrary phone number. When played through a telephone handset, the output of your function will actually be able to dial a touch-tone phone.

Checkpoint (1/6): Show your TA your DTMF Encoder
Deliverable (1/8): Your completed dtmfdial.m file should be included in your lab report.

6.2 Exercise #2: DTMF Decoder (Real-Time)

DTMF decoding can be done in a number of ways. For this lab you will be using a correlation scheme. The correlation scheme operates as follows.

An input signal, $p(t) = \sin(2\pi(697)t) + \sin(2\pi(1209)t)$, corresponds to the tone generated when the '1' key is pressed. Since the input signal includes $\sin(2\pi(697)t)$, the correlation of the input signal with $\sin(2\pi(697)t)$ must be higher than the correlation of the input signal with $\sin(2\pi(770)t)$, $\sin(2\pi(852)t)$, or $\sin(2\pi(941)t)$. Similarly, the second frequency component of the input signal will have its greatest correlation with 1209 Hz.

1. Using the `dtmfdial.m` function you wrote for Experiment #1, generate and plot the DTMF signals corresponding to the keys 1, 5, and 9.
   Note: You might want to zoom in on the x-axis to get a good representation of the waveform. Use `subplot` to get all plots on one page.

   Deliverable (2/8): Include a print out of these plots in your lab report.

2. Now, plot the correlation of each signal with the seven individual tones (697 Hz, 770 Hz, etc.).
   Note: Make sure you adjust the x-axis so that it corresponds to the shift between the two signals (i.e., make sure that you get a peak at the correct x-axis value).

   Discussion (1/2): Discuss with your TA: For which tones do you expect a large correlation for each key (1, 5, and 9)? Are your results what you expect?
   Deliverable (3/8): Include a print out of these plots in your lab report.

3. Zoom in on one (or more) of the correlation plots.

   Discussion (2/2): What do you observe? Why does this happen?
   Deliverable (4/8): Include a print out of these plots in your lab report.

4. Load the file `UnknownKey.mat` (available on Canvas). Using correlation, determine which key this signal corresponds to.

   Checkpoint (2/6): Report to your TA which key the signal corresponds to.
   Deliverable (5/8): Include your answer, supported by plots, in your report.

Note: Another way to implement DTMF decoding is through a series of bandpass filters, each of which is tuned to one of the DTMF frequencies. To detect which key has been pressed, the decoder would look at the output of each filter and choose the two frequencies whose filters have the greatest output (since each DTMF tone consists of two summed sinusoids).

6.3 Exercise #3: DTMF Correlator

You are going to begin by building a very simple correlator (using the Audio System Toolbox) that determines whether the input tone is 697 Hz or 1477 Hz. If the input is the lower frequency, the scope will show 1; if the input is the higher frequency, the scope will show 7. You have been provided with a template, `Simple_Corr.mdl`, which contains all of the logic and the Audio System Toolbox. Your task is to complete the model by adding the blocks necessary to compute the correlation between the incoming signal and a reference signal.

1. Load `Simple_Corr.mdl` now. The input will be a single tone (either 697 Hz or 1477 Hz) that you create in the MATLAB workspace and call `SinWave`.

   Checkpoint (3/6): Show your TA your SinWave output.

   The **Relational Operator** block compares two inputs. If the top input is less than the bottom input, the **Relational Operator** outputs a 1 which tells the **Switch** to pass the constant '1' through to the **Scope** block. If the top input is greater than the bottom input, the **Relational Operator** outputs a 0, which tells the Switch to pass through the constant '7'.

2. Build the intermediate steps to connect the **SinWave** (Signal from Workspace) block to the **Relational Operator** such that the scope shows 1 if the input is 697 Hz and the scope displays a 7 if the input is 852 Hz.

3. Replace the **Signal From Workspace** block with a **DSP Sine Wave** block and set the sample time to 1/8000 and samples per frame to 40. Set the frequency to the desired value

4. Be sure to set the simulation time to the duration of the sine wave. Each tone for the number is 0.5 seconds and a rest is 0.05 seconds. Run the model and view the results on the scope by double-clicking the scope.

   Checkpoint (4/6): Show your TA your new output.
   Deliverable (6/8): In your lab report, include a printout of your completed `Simple_Corr.mdl` and explain how it works.

5. Terminate the model and close `Simple_Corr.mdl` before proceeding.

6. Open the Simulink model, `DTMF.mdl`. This model implements a DTMF decoder (using correlation) in real-time. Examine the block diagram and make sure you can explain each step in the algorithm.
   Note: You may need to double-click on some blocks to see what they are and what parameters have been set.

Checkpoint (5/6): Discuss with your TA how the algorithm works.
Deliverable (7/8): In your lab report, explain how the algorithm works, and its details (e.g., explain parameter values).

7. Generate a phone number using your `dtmfdial.m` function and save the signal into the variable `PhoneNum`.
8. For a regular US phone number (10 digits), you can set the simulation parameter time to 5.5 seconds. Click the play button to run the `PhoneNumber` at real time and view it on the Scope.

Checkpoint (6/6): Show to your TA your Scope displaying the **PhoneNumber**.
Deliverable (8/8): In your Scope, you should have noticed erroneous values that did not correspond to the numbers you "dialed". Comment on why those values occur and how you could improve the system so that they are eliminated.

6.4 Wrap-Up

When you are done with laboratory work and ready to leave, please follow this sequence:

1. Log off of your computer account
2. Clean up laboratory space (put stool under table)

7 Lab Report

7.1 Objectives (5)

Summarize the stated objectives in your own words.

7.2 Background (8)

Briefly describe the design of a DTMF system (encoder and decoder) in your own words.

7.3 Results and Discussion (80)

- Include your `dtmfdial.m` file.
- Include plots of the signals corresponding to the keys 1, 5, and 9.
- Include plots of the correlation of the keys 1, 5, and 9 to the seven individual DTMF tones.
- Discuss what you observed about the correlation plots you generated for the three keys (1, 5, 9).
- How did you go about determining the value of the `UnknownKey`?
- Include a printout of your completed `Simple_Corr.mdl`. Explain how this model works.
- Explain, step-by-step, how the DTMF decoding algorithm presented in `DTMF.mdl` works.
- When you used the **Scope** to decode the phone number, you should have noticed erroneous values that did not correspond to the numbers you "dialed". Comment on why those values occur and how you could improve the system so that they are eliminated.

7.4 Conclusions (7)

Summarize what you have learned through this laboratory exercise.

Include the statement "I have adhered to the Duke Community Standard in completing this assignment", along with your signature, on the cover page and properly cite any sources you reference in your written report.

8 References

9 Updates

- October 2024: Added a prelab; made multiple adjustments based on recommendations of Jenny Green (Pratt '25), Eduardo Bortolomiol (Pratt '26), and Adam Davidson; eliminated extension and distributed points to required sections.
- June 2024: Converted to a standalone document
- September 2021: Converted to LaTeX.