DSP First

Laboratory Exercise #7

Everyday Sinusoidal Signals

This lab introduces two practical applications where sinusoidal signals are used to transmit information: a touch-tone dialer and amplitude modulation (AM) for radio. In both cases, FIR filters can be used to extract the information encoded in the waveforms.

1 Background

This lab has two parts: Part A investigates the generation and detection of the signals used to dial the telephone. In Part B you will modulate and demodulate AM (amplitude modulation) waveforms such as those used in AM radio.

1.1 Background A: Telephone Touch Tone[1] Dialing

Telephone touch pads generate *dual tone multi frequency* (DTMF) signals to dial a telephone. When any key is pressed, the tones of the corresponding column and row (in Fig. 1) are generated, hence dual tone. As an example, pressing the **5** button generates the tones 770 Hz and 1336 Hz summed together.

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**	#

Figure 1: DTMF encoding table for Touch Tone dialing. When any key is pressed the tones of the corresponding column and row are generated.

The frequencies in Fig. 1 were chosen to avoid harmonics. No frequency is a 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.[2] This makes it easier to detect exactly which tones are present in the dial signal in the presence of line distortions.

1.2 DTMF Decoding

There are several steps to decoding a DTMF signal:

1. Divide the signal into shorter time segments representing individual key presses.
2. Determine which two frequency components are present in each time segment.
3. Determine which button was pressed, **0**–**9**, *, or #.

It is possible to decode DTMF signals using a simple FIR filter bank. The filter bank in Fig. 2 consists of filters which each pass only one of the DTMF frequencies and whose inputs are the same DTMF signal.

[1] Touch Tone is a registered trademark

[2] More information can be found at: `http://arrow.cso.uiuc.edu/telecom/dtmf/dtmf.html`

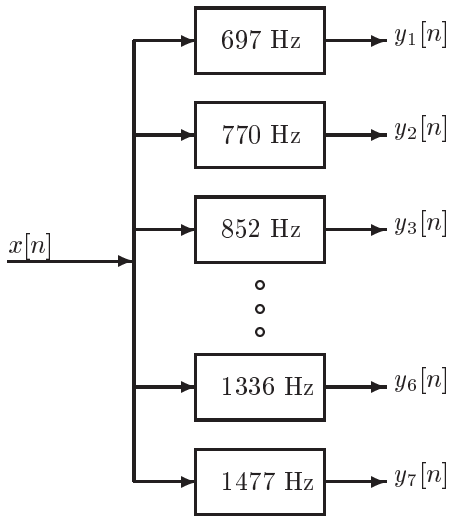

Figure 2: Filter bank consisting of bandpass filters which pass frequencies corresponding to the seven DTMF component frequencies listed in Fig. 1.

When the input to the filter bank is a DTMF signal the outputs of two filters should be larger than the rest. The two corresponding frequencies must be detected in order to determine the DTMF code. A good measure of the output levels is the average power at the filter outputs. This is calculated by squaring the filter outputs and averaging over a short time interval. More discussion of the detection problem can be found in Section 4.

1.3 Background B: Amplitude Modulation (AM)

Amplitude modulation is often used to transmit a signal with low-frequency content using a high-frequency transmission channel. A common example is AM radio. In AM radio a relatively low-frequency signal such as a speech signal (which has frequencies between 50 Hz and 4 kHz) is transmitted by radio waves at frequencies around 1 MHz.

Amplitude modulation is performed by multiplying a high frequency signal (called the *carrier*) by a the low-frequency *message* signal $m(t)$:

$$x(t) = (1 + m(t)) \cos(2\pi f_c t + \phi). \tag{1}$$

where the carrier signal corresponds to the $\cos(2\pi f_c t + \phi)$ term. Although the message signal $m(t)$ may be very complicated, a good understanding of AM can be obtained by analyzing AM signals of the form:

$$x(t) = (1 + A\cos(2\pi f_m t)) \cos(2\pi f_c t) \tag{2}$$

i.e., the message signal is a cosine, $m(t) = A\cos(2\pi f_m t)$. A straightforward expansion of (2) shows that:

$$x(t) = \cos(2\pi f_c t) + \frac{A}{2}\cos(2\pi(f_c + f_m)t)) + \frac{A}{2}\cos(2\pi(f_c - f_m)t). \tag{3}$$

In communications jargon, the signal components at $f = f_c \pm f_m$ are called the *sidebands* and the signal component with a frequency of f_c is called the carrier. When $f_c \gg f_m$ the spectrum of the AM signal looks like that shown in Fig. 3. Note that an AM signal in (3) is very similar to the beat signals studied earlier in Lab 4 and Chapter 3, except for the addition of the carrier term.

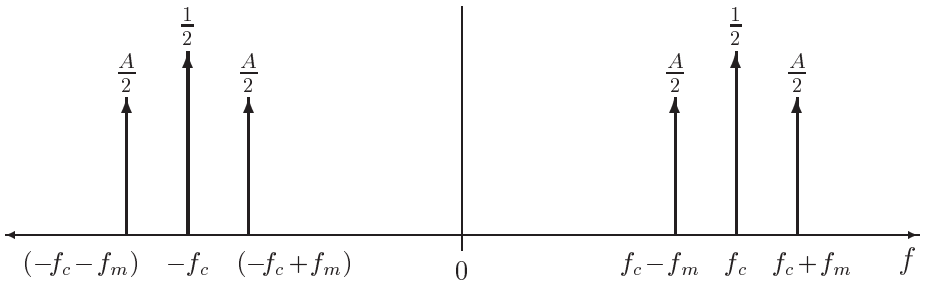

Figure 3: Spectrum of an amplitude modulated (AM) tone.

More complicated message signals may also be analyzed. If $m(t)$ in (2) is made up of multiple sinusoidal components, those components are each shifted in frequency as was the single sinusoid in (3).

$$\begin{aligned} x(t) &= \left(1 + \sum_k A_k \cos(2\pi f_k t)\right) \cos(2\pi f_c t) \qquad (4) \\ &= \cos(2\pi f_c t) + \sum_k \frac{A_k}{2} \cos(2\pi(f_c - f_k)t) + \sum_k \frac{A_k}{2} \cos(2\pi(f_c + f_k)t) \end{aligned}$$

Thus we would have many spectral lines in the sidebands.

1.4 AM Demodulation

Demodulation is the process of recovering the message waveform from a modulated signal such as AM. There are numerous methods of demodulating a signal but only two will be discussed here. This lab focuses on an LTI filtering approach, but a more common approach is presented first for comparison.

1.5 Envelope Detection (Peak Tracking)

A crystal radio (and other inexpensive AM radios) uses a capacitor, resistor, and diode to perform the AM demodulation (see Fig. 4). The idea is to get a waveform that approximately follows the

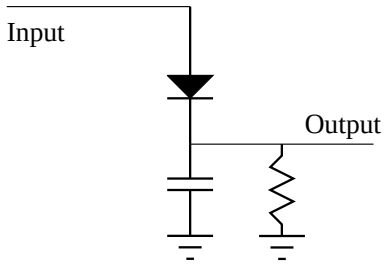

Figure 4: Simple capacitor, resistor, diode type AM demodulator as used in cheap AM radios.

peaks of the AM waveform. During each positive cycle of the AM signal, the capacitor is charged via the diode. Then during the negative part of the cycle, the resistor discharges the capacitor slowly so that the demodulated waveform can also follow the envelope of the modulated waveform

as the peaks decrease in amplitude. A MATLAB function to perform this type of demodulation is shown in Fig. 5.

CD-ROM
amdemod.m

```
function dd = amdemod(xx,fc,fs,tau)
%
% where
%      xx = the input AM waveform to be demodulated
%      fc = carrier frequency
%      fs = sampling frequency
%     tau = time constant of the RC circuit normalized by fs
%             (OPTIONAL: default value is tau=0.97)
%      dd = demodulated message waveform
```

Figure 5: Arguments for the MATLAB function `amdemod` which simulates an AM demodulator based on a simple capacitor, resistor, diode type circuit in Fig. 4.

A close-up of the output waveform from the simple demodulator of Fig. 4 is shown in Fig. 6. Also shown are the message waveform, $m(t) = 2\sin(2\pi(25)t)$ and the carrier at 371 Hz which make up the AM signal:

$$x(t) = (1 + 0.4m(t))\,\cos(2\pi(371)t)$$

The demodulated signal is not perfect, but it does approximate the sinusoidal shape of the message signal once you subtract the DC level of one. The jagged appearance is due to the exponential discharge of the RC circuit.

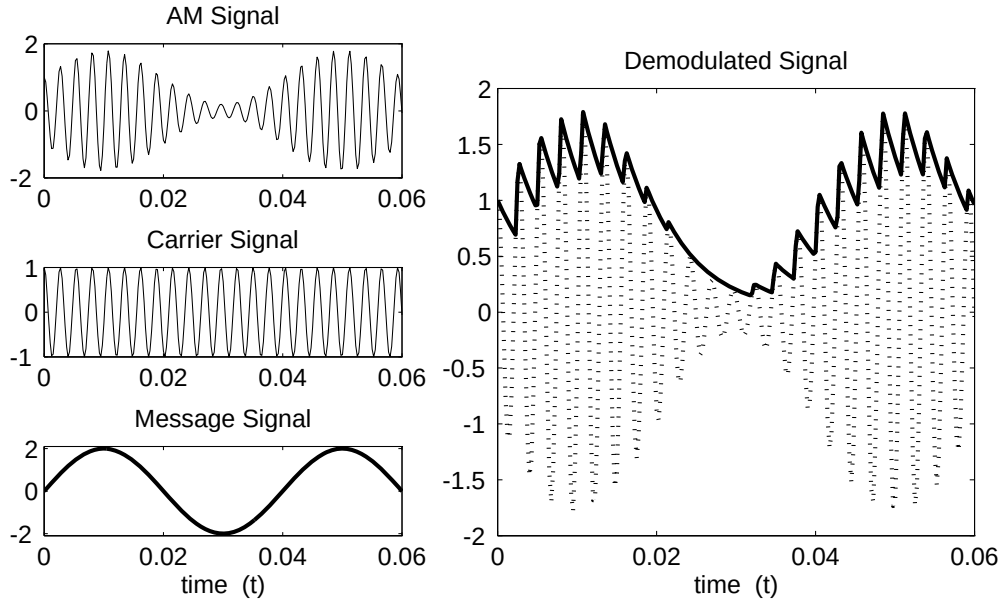

Figure 6: Close-up view of the waveform generated by the capacitor, resistor, diode type AM demodulator with $\tau = 0.96$. The dark jagged line is the demodulator output and the dotted line is a close-up of the cycles of the AM waveform.

1.6 LTI filter based demodulation

It is possible to recover the message signal by *modulating the modulated signal* and then filtering. This is the basic principle used in all commercial radios nowadays. Given an AM signal of the form (2), we can isolate the message signal by multiplying $x(t)$ by $\cos(2\pi f_c t)$.

$$\begin{aligned} x(t)\cos(2\pi f_c t) &= (1+m(t))\cos(2\pi f_c t)\ \cos(2\pi f_c t) && (5)\\ &= (1+m(t))\left(\tfrac{1}{2}+\tfrac{1}{2}\cos(2\pi(2f_c)t)\right) && (6)\\ &= \tfrac{1}{2}m(t)+\tfrac{1}{2}+\tfrac{1}{2}(1+m(t))\cos(2\pi(2f_c)t) && (7)\end{aligned}$$

Notice that the message signal is now available outside of the product term. There are still two terms in (7) which must be eliminated; the $\frac{1}{2}$ is a DC offset which can simply be subtracted out or ignored; the other term is a very high frequency term (at $f = 2f_c$) which we will eliminate by filtering. The scale factor of $\frac{1}{2}$ multiplying $m(t)$ can be compensated by doubling the final output.

1.7 Notch Filters for Demodulation

A notch filter will be used as part of the demodulation process. Notch filters are filters that completely eliminate some frequency other than $\hat{\omega} = 0$ or $\hat{\omega} = \pi$. It is possible to make a notch filter with as few as three coefficients. If $\hat{\omega}_{\text{not}}$ is the desired notch frequency, then the following length-3 FIR filter

$$y[n] = x[n] - 2\cos(\hat{\omega}_{\text{not}})x[n-1] + x[n-2] \quad (8)$$

will have a zero at $\hat{\omega} = \hat{\omega}_{\text{notch}}$. For example, a filter designed to completely eliminate signals of the form $Ae^{j0.5\pi n}$ would have coefficients

$$b_0 = 1, \quad b_1 = 2\cos(0.5\pi) = 0, \quad b_2 = 1.$$

However, our specifications will be given in terms of continuous-time frequency, e.g., eliminate the spectral component at f_{not}. We must convert to discrete-time frequency by using the frequency scaling due to sampling:

$$\hat{\omega}_{\text{not}} = 2\pi\frac{f_{\text{not}}}{f_s}$$

where f_s is the sampling frequency.

2 Warm-up A: DTMF Synthesis

The instructor verification sheet is included at the end of this lab.

2.1 DTMF Dial Function

Write a function, `dtmfdial`, to implement a DTMF dialer defined in Fig. 1. A skeleton of `dtmfdial.m` including the help comments is given in Fig. 7; you must complete the code so that it implements the following:

1. The input to the function is a vector of numbers which may range between 1 and 12, with 1 – 10 corresponding to the digits (10 corresponds to **0**), 11 is the * key, and 12 is the # key.

2. The output should be a vector containing the DTMF tones, sampled at 8 kHz. The duration of the tones should be about 0.5 sec., and a silence, about 0.1 sec. long, should separate each tone pair.

```
function tones = dtmfdial(nums)
%DTMFDIAL  Create a vector of tones which will dial
%           a DTMF (Touch Tone) telephone system.
%
% usage:  tones = dtmfdial(nums)
%     nums  = vector of numbers ranging from 1 to 12
%     tones = vector containing the corresponding tones.
%
if (nargin < 1)
    error('DTMFDIAL requires one input');
end
fs = 8000;   %-- This MUST be 8000, so dtmfdeco( ) will work.
   .
   .
```

Figure 7: Skeleton of `dtmfdial.m`. A DTMF phone dialer.

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 be able to dial the phone. You may use `specgram` to check your work.[3]

Instructor Verification (separate page)

3 Warm-up B: Tone Amplitude Modulation

(a) Derive (3) from (2). Hint: express the cosine terms as sums of complex exponentials using Euler's identity.

(b) Create a test AM signal with the following characteristics:

  (a) The carrier tone, `cc`, must have a frequency of 1200 Hz, a duration of 1 second, a sample rate of 8000 Hz, and a phase of zero.[4]

  (b) The message signal, `mm`, should be a 100 Hz tone with an amplitude of 0.8.

(c) Plot the first 200 points of the modulated signal and the message signal on the same plot. Use different colors or line types for the two signals (see `help plot`).

(d) Compare the spectra of the message, the carrier, and the modulated signals using the following command for each:

```
showspec(_, 8000);
```

Instructor Verification (separate page)

[3] In MATLAB the demo called `phone` also shows the waveforms and spectra generated in a DTMF system.

[4] Although we will not cover it in this lab, the phase of the carrier is important when using the demodulation described above—specifically, the demodulation tone must have the same phase (and frequency) as the carrier. There are some sophisticated ways of ensuring that the demodulation remains in phase with the carrier tone but we will not cover them here. Instead we will use a cosine function with zero phase for both the carrier and demodulation tone.

4 Lab A: DTMF Decoding

A DTMF decoding system needs two pieces: a bandpass filter to isolate individual frequency components, and a detector to determine whether or not a given component is present. The detector must "score" each possibility and determine which frequencies are most likely present. In a practical system where noise and interference are present, this scoring process is a crucial part of the system design, but we will only work with noise-free signals to understand the basic functionality in the decoding system.

4.1 Filter Design

The filters that will be used in the filter bank (Fig. 2) are a simple type constructed with sinusoidal impulse responses. In the section on useful filters in Chapter 7, a simple bandpass filter design method was presented in which the impulse response of the filter is simply a finite-length cosine of the form:

$$h[n] = \frac{2}{L}\cos\left(\frac{2\pi f_b n}{f_s}\right), \qquad 0 \le n < L$$

where L is the filter length, and f_s is the sample frequency. The parameter f_b defines the frequency location of the passband, e.g., we pick $f_b = 697$ if we want to isolate the 697 Hz component. The bandwidth of the bandpass filter is controlled by L; the larger the value of L, the narrower the bandwidth.

(a) Generate a bandpass filter, `h770`, for the 770 Hz component with $L = 64$ and $f_s = 8000$. Plot the filter coefficients in the first panel of a two-panel subplot using the `stem()` function.

(b) Generate a bandpass filter, `h1336`, for the 1336 Hz component with $L = 64$ and $f_s = 8000$. Plot the filter coefficients in the second panel of a two-panel subplot using the `stem()` function.

(c) Use the following commands to plot the frequency response (magnitude) of `h770`

```
fs = 8000;
ww = 0:(pi/256):pi;   %-- only need positive freqs
ff = ww/(2*pi)*fs;
H = freqz(h770,1,ww);
plot(ff,abs(H)); grid on;
```

(d) Indicate the locations each of the DTMF frequencies (697, 770, 852, 941, 1209, 1336, and 1477 Hz) on the plot from part (c). Hint: use the `hold` and `stem()` commands.

(e) Comment on the selectivity of the bandpass filter `h770`, i.e., use the frequency response to explain how the filter passes one component while rejecting the others. Is the filter's passband narrow enough?

(f) Plot the magnitude response of the `h1336` filter and compare its passband to that of the `h770` filter.

4.2 A Scoring Function

The final objective is decoding—a process that requires a binary decision on the presence or absence of the individual tones. In order to make the signal detection an automated process, we need a *score* function that rates the different possibilities.

(a) Complete the `dtmfscor` function based on the skeleton given in Fig. 8. Assume that the input signal `xx` to the `dtmfscor` function is actually a short segment from the DTMF signal. The task of breaking up the signal so that each segment corresponds to one key will be done by another function prior to calling `dtmfscor`.

The implementation of the FIR bandpass filter is done with the `conv` function, but we could also use `firfilt`. The running time of the convolution function is proportional to the filter length L. Therefore, the filter length L must satisfy two competing constraints: L should be large so that the bandwidth of the BPF is narrow enough to isolate individual frequencies, but making it too large will cause the program to run slowly.

```
function  ss = dtmfscor(xx, freq, L, fs)
%DTMFSCOR
%          ss = dtmfscor(xx, freq, L, [fs])
%    returns 1 (TRUE) if freq is present in xx
%            0 (FALSE) if freq is not present in xx.
%     xx = input DTMF signal
%   freq = test frequency
%      L = length of FIR bandpass filter
%     fs = sampling freq (DEFAULT is 8000)
%
%   The signal detection is done by filtering xx with a length-L
%   BPF, hh, squaring the output, and comparing with an arbitrary
%   setpoint based on the average power of xx.
%
if (nargin < 4), fs = 8000; end;

hh =           %<======== define the bandpass filter coeffs here
ss = (mean(conv(xx,hh).^2) > mean(xx.^2)/5);
```

Figure 8: Skeleton of the `dtmfscor.m` function.

(b) Explain the last line in `dtmfscor.m`:

```
ss = (mean(conv(xx,hh).^2) > mean(xx.^2)/5);
```

4.3 DTMF Decode Function

The DTMF decode function, `dtmfdeco` will use `dtmfscor` to determine which key was pressed based on an input DTMF signal. The skeleton of this function in Fig. 9 includes the help comments and the table of tone pairs from Fig. 1. You must add the logic to decide which key is present.

Assume that the input signal `xx` to the `dtmfscor` function is actually a short segment from the DTMF signal. The task of breaking up the signal so that each segment corresponds to one key has already been done by the function that calls `dtmfdeco`.

There are several ways to write the `dtmfdeco` function, but you should avoid excessive use of "`if`" statements to test all 12 cases. Hint: use MATLAB's vector logicals (see `help relop`) to implement the tests in a few statements.

4.4 Telephone Numbers

There is a function `dtmfmain` supplied with the *DSP First* CD-ROM which will run the entire

```
function key = dtmfdeco(xx,fs)
   %DTMFDECO    key = dtmfdeco(xx,[fs])
   %    returns the key number corresponding to the DTMF waveform, xx.
   %     fs = sampling freq (DEFAULT = 8000 Hz if not specified.
   %
   if (nargin < 3), fs = 8000; end;
   tone_pairs = ...
   [ 697  697  697  770  770  770  852  852  852  941  941  941;
     1209 1336 1477 1209 1336 1477 1209 1336 1477 1336 1209 1477 ];
       .
       .
```

Figure 9: Skeleton of `dtmfdeco.m`.

DTMF system consisting of the three M-files you have written: `dtmfdial.m`, `dtmfscor.m`, and `dtmfdeco.m`. If you are presenting this project in a lab report, the `dtmfmain` function can be used to demonstrate a working version of your programs.

An example of using `dtmfmain` is shown here:

```
>> dtmfmain( dtmfdial([1:12]) )
ans =
    1    2    3    4    5    6    7    8    9   10   11   12
```

For this function to work correctly, all three M-files must be on the MATLAB path. It is also essential to have short pauses in between the tone pairs so that `dtmfmain` can parse out the individual signal segments.

5 Lab B: AM Waveform Detection

As discussed in Section 1.3 on background, there are several ways to perform AM waveform detection or *demodulation*. Most AM radios use a method of peak tracking to recover the envelope of the modulated waveform. Better detectors use a combination of additional modulation and filtering.[5] This is the method that we will use in this lab.

(a) Demodulate the AM test signal created in the warmup, Section 1.3, using the filter based method as follows:

   (i) Multiply the AM test signal by the carrier as described in (7) and look at the spectrum using the `showspec` command. Identify each of the spectral peaks with terms in (7).

   (ii) Create a three-term notch filter designed to eliminate the high-frequency component at 2400 Hz. Plot the frequency response of this notch filter to verify that you have done this correctly.

   (iii) Filter the product signal created in step (i) with your notch filter. The result should look similar to the original message signal.

(b) Demodulate the AM test signal using the `amdemod` function which implements the peak following circuit of Figs. 4 and 5. Experiment with the decay rate (the fourth input parameter

[5] You may wish to try the MATLAB `demod` function on the signal used in the Warm-up section of the lab. This function is part of the MATLAB Signal Processing Toolbox.

to `amdemod`) to find the best setting. Usually values of τ near 0.9 will give good results, but look at the output for $\tau = 1.2$ and $\tau = 0.4$.

(c) Create a three-panel subplot with containing plots of the *middle 200 points* from the following signals:

(i) the original message signal,

(ii) the demodulated signal found in part (a),

(iii) and the demodulated signal found in part (b).

Compare the quality of the two demodulated signals and judge how well they follow the original message signal in the time domain.

(d) Compare the spectra of the message and the two demodulated signals using the following command for each:

```
showspec(_,8000);
```

(e) Explain how the filter method could be modified to produce a better quality signal. Hint: look at the spectrum of the demodulated signal to see if the high-frequency component at 2400 Hz is completely gone. If not, then filter the signal again with the same notch filter used in the demodulation of part (a). Is this equivalent to using a higher-order notch filter with more coefficients?

6 Optional: Amplitude Modulation with Speech

Load the speech signal and other MATLAB data with the command

```
load lab7dat
```

This MATLAB data file contains three variables:

`cc`: a carrier signal at 24 kHz with a sampling rate of 96 kHz.

`mm`: a speech signal at the same sampling rate as the carrier signal.

`ss`: the speech signal at its original sampling rate of 8 kHz.

(a) Create a modulated signal, `aa`, from `cc` and `mm` with the command:

$$\texttt{aa} = (1 + \texttt{mm}) .* \texttt{cc};$$

(b) In a three-panel subplot show the spectra of `cc`, `mm`, and `aa`. Comment briefly on the relationship between them.

(c) Demodulate the speech waveform using the function `amdemod`; call the demodulated waveform `dd`. The sampling rate of the demodulated waveform is very high, much higher than it needs to be. Since the frequency content of `dd` is now almost entirely below 4 kHz, we may sample it at 8 kHz for playback. This is done by only taking every 12^{th} sample from `dd`, and throwing away the rest:

$$\texttt{ds} = \texttt{dd}(1 : 12 : \texttt{length}(\texttt{dd}));$$

(d) Listen to `ds` and `ss` and comment on what you hear.

Lab 7
Instructor Verification Sheet

Staple this page to the end of your Lab Report.

Name: _____ Date: _____

Warm-up A

Part 2.1 Complete `dtmfdial.m`:

Verified: _____

Warm-up B

Part 3 Create and explain AM signal:

Verified: _____