EEE 391
Basics of Signals and Systems
Spring 2019–2020
MATLAB Assignment 1
Due: 3 March 2020, Tuesday by 23:55 on Moodle

In this assignment, you are going to generate various sinusoidal signals and hear what sinusoids of different audible frequencies sound like, and experiment with some musical sounds. We will extend our treatment of sinusoidal waveforms to more complicated signals composed of sums and products of sinusoidal signals or sinusoids with changing amplitude, frequency, or phase that are related to the basic sinusoid.

The Musical Scale

The frequencies 440 Hz and 880 Hz both correspond to the musical note A, but one octave apart. The next higher A in the musical scale would have the frequency 1760 Hz, twice 880 Hz. Thus, the following frequencies all correspond to the note A: 440 Hz, $440 \times 2 = 880$ Hz, $880 \times 2 = 1760$ Hz, $1760 \times 2 = 3520$ Hz.

In the western musical scale, there are 12 notes in every octave. These notes are evenly distributed (geometrically), so the next note above A, which is B flat, has frequency $440 \times \beta$ where β is the 12th root of two, or approximately 1.0595. The next note above B flat, which is B, has frequency $440 \times \beta^2$, etc.

Below is a table of the complete musical scale between middle A and A-880. Each frequency is β times the frequency that comes before it.

musical note	frequency (Hz)
A	440
B flat (B♭♭)	466
B	494
C	523
C sharp (C#)	554
D	587
D sharp (D#)	622
E	659
F	698
F sharp (F#)	740
G	784
A flat (A♭♭)	831
A	880

Table 1: Musical notes and the corresponding frequencies over one octave.

Part 1: Fundamental Frequency and Harmonics
Consider a sinusoidal signal of the form

$$x_1(t) = \sin(2\pi f_o t).$$

Take $f_o = 440$ Hz and evaluate the signal's defining formula at discrete instants of time. These are called *samples* of the signal. In other words, discretize your signal at uniform sampling intervals of $T_s = 0.0001$ s using $x_1[n] = x_1(nT_s)$ over a duration of 3.0 s. Store the values of $x_1[n]$ in an array called **x1**. Plot **x1** versus **t** over a duration of 0.01 s.

Type `help plot` in the MATLAB command window to see what you can do with the `plot` command which is one of the vital commands of MATLAB. Also study the MATLAB commands `xlabel, ylabel, title, xlim, ylim`, and `grid`. These commands are essential for producing professional looking plots in MATLAB. Make sure that you can properly use these commands.

Turn on the speakers of your computer. Examine the `sound(.)` and `soundsc(.)` commands of MATLAB by typing `help sound` and `help soundsc` in the MATLAB command window. Notice that `soundsc(.)` requires the sampling rate (frequency) at which the signal samples were created. Then, type `sound(x1)` or `soundsc(x1)`. Listen to the sound. Repeat the same for $2f_o = 880$ Hz and $4f_o = 1760$ Hz. What happens to the pitch of the sound as the frequency increases?

The psychoacoustic properties of the musical scale are fascinating. The musical scale is based on our perception of frequency and harmonic relationships between frequencies. Frequencies that are harmonically related tend to sound good together. Among all the harmonic relationships in the scale, A, C sharp, and E have among the simplest. This possibly accounts for the predominance of the **major triad** in western music.

What about $440 \times 3 = 1320$ Hz? Notice that $1320/2 = 660$ Hz, which is almost exactly the frequency of note E. Thus, 440×3 is the note E, one octave above the E immediately above A-440. E and A are harmonically related, and to most people, they sound good together. It is because $440 \times 3 \approx 659 \times 2$.

Where does the C sharp come from in the major triad? Notice that $440 \times 5 \approx 554 \times 4$.

Repeat the above exercise for the sinusoids corresponding to the notes C sharp and E and hear what these notes sound like.

Next, try combining A, C sharp, and E additively to create a major triad which can be expressed as a sum of sinusoids:

$$s(t) = \sin(2\pi 440\, t) + \sin(2\pi 554\, t) + \sin(2\pi 659\, t)$$

Writing a single line of code, compute $s(t)$ for 3.0 s and store it in an array named **s**. Make a plot of the array **s** versus **t** and listen to **s** by the `soundsc(.)` command in the same way as you did above for **x1**. If you have musical background, you will recognize this sound as a major triad. What is special about this frequency combination?

For somewhat more arcane reasons, the interval between A and E, which is a frequency rise of 3/2, is called a fifth. The note 3/2 above E has the frequency 988 Hz, which is an octave above B-494. Another 3/2 above that is approximately F sharp (740 Hz). Continuing in this fashion, multiplying frequencies by 3/2, and then possibly dividing by two, you can approximately trace the 12 notes of the scale. This progression based on the choice of 12 evenly spaced notes corresponds to the so-called **circle of fifths**. The notion of key in music and a scale are based on this circle of fifths.

Musical sounds such as chords can be characterized as sums of pure tones. Of course, truly musical sounds are much more complex. For one thing, pure tones are not particularly appealing sounds. Musical instruments produce notes that are more complex than pure tones. The characteristic sound of an instrument is its **timbre** and some aspects of timbre can also be characterized as sums of sinusoids.

Part 2: The Effect of Phase

Let $x_2(t) = \cos(2\pi f_o t + \phi)$ where $f_o = 587$ Hz corresponding to the note D.

a) Let $\phi = 0$. Compute, plot, and listen to $x_2(t)$ in the same way as in Part 1.

b) Repeat for $\phi = \frac{\pi}{4}$, $\phi = \frac{\pi}{2}$, $\phi = \frac{3\pi}{4}$, $\phi = \pi$ rad.

How does the volume of the sound that you hear change with ϕ? How does the pitch of the sound that you hear change with ϕ?

Part 3: Sinusoid with Exponentially Decaying Envelope

Now consider the signal defined as

$$x_3(t) = e^{-(a^2+2)t} \cos(2\pi f_o t).$$

Take $a = 2$ and $f_o = 440$ Hz. Writing a single line of code, compute $x_3(t)$ and store it in an array named **x3**. In this code, make use of the element-wise multiplication facility of MATLAB while computing the product of $e^{-(a^2+2)t}$ and $\cos(2\pi f_o t)$. (Recall that in MATLAB, element-wise multiplication of arrays is achieved by placing a dot in front of the multiplication symbol. (You can examine the examples provided in the MATLAB exercises on the course web page.) Provide this code in your report. Make a plot of **x3** versus **t** and listen to **x3** by the `soundsc(.)` command in the same way as you did in Part 1. Compare your plot and what you hear to the results you obtained for $x_1(t)$ when f_o is 440 Hz. What is the effect of including the exponential term to the sound that you hear? Which one of $x_1(t)$ and $x_2(t)$ resembles the sound produced by a piano more? Which one resembles that of a flute more? Now take $a = 1$, and recompute **x3** (do not change f_o and t). Compare the sound you hear with that of the $a = 2$ case. Repeat for $a = 3$. How does the duration of the sound that you hear change as a increases? Can you relate this signal model to a physical system that we discussed in class that produces sinusoidal signals?

Part 4: Beat Notes and Amplitude Modulation

When we multiply two sinusoids having different frequencies, we can create an interesting audio effect called a **beat note**. The phenomenon, which may sound like a warble, is best heard by picking one of the frequencies very small (e.g., 10 Hz) and the other around 1 kHz in the multiplicative form of two sinusoids. Thus, consider the product of two cosine signals:

$$x_4(t) = \cos(2\pi f_1 t) \cos(2\pi f_2 t)$$

where $f_1 \ll f_2$. Take $f_1 = 10$ Hz and $f_2 = 1000$ Hz. Again using a single line command, compute the array **x4** (provide this code in your report), plot and listen to it in the same way as in Part 1. Compare your results with those of $x_1(t)$ in Part 1. What is the effect of the low-frequency cosine term $\cos(2\pi f_1 t)$ on the sound that you hear? Recompute **x4** for $f_1 = 5$ Hz and $f_1 = 15$ Hz. How does the sound that you hear change? By using the well-known trigonometric identity

$$\cos(\theta_1)\cos(\theta_2) = \frac{1}{2}\left[\cos(\theta_1 + \theta_2) + \cos(\theta_1 - \theta_2)\right],$$

express $x_4(t)$ as the sum of two different-frequency cosine signals and also interpret what you hear in this part using this equality. Picking two or three frequencies very close together in the additive form of sinusoids generates beat signals, which is the case here.

Some musical instruments naturally produce beating notes. Musicians use this beating phenomenon as an aid in tuning two instruments to the same pitch. When two notes are close but not identical in frequency, the beating phenomenon is heard. As one pitch is changed to become closer to the other, the effect disappears, and the two instruments are then "in tune."

Another use for multiplying sinusoids is **modulation** for radio broadcasting and communication systems. For example, AM radio stations use this method, which is called **amplitude (AM) modulation**. This is the process of multiplying a low-frequency signal $v(t)$ by a high-frequency sinusoid. Thus, an AM signal is a product of the form:

$$x(t) = v(t)\cos(2\pi f_c t)$$

where it is assumed that the frequency of the cosine term is much higher than any frequency contained in the spectrum of $v(t)$, which represents the voice or music signal to be transmitted. The cosine wave is called the **carrier signal** and its frequency is called the **carrier frequency**.

Part 5: Chirp Signals and Frequency Modulation

We will also consider signals whose frequency varies as a function of time. The argument of the constant-frequency sinusoid is $(2\pi f_o t + \phi)$. This angle function changes *linearly* versus time, and its time derivative is $\omega_o = 2\pi f_o$ which equals the constant frequency of the cosine in rad/s.

A generalization is available if we adopt the following notation for the class of signals represented by a cosine function with a time-varying angle:

$$x(t) = A\cos[\psi(t)]$$

The time derivative of the angle in the above equation gives a radian frequency changing with time:

$$\omega_i(t) = \frac{d}{dt}\psi(t) \qquad \text{(rad/s)}$$

but we prefer the cyclic frequency, therefore, we divide by 2π to define the *instantaneous frequency*:

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\psi(t) \qquad \text{(Hz)}$$

A **chirp signal** is a sinusoid whose frequency is linearly swept so that it changes linearly from a starting value to an ending one. The formula for such a signal can be obtained by creating a quadratic angle by defining $\psi(t)$ as:

$$\psi(t) = 2\pi\mu t^2 + 2\pi f_o t + \phi.$$

The time derivative of $\psi(t)$ yields an instantaneous frequency that changes *linearly* versus time

$$f_i(t) = 2\mu t + f_o.$$

The slope of $f_i(t)$ is equal to 2μ and its intercept is equal to f_o. If the signal starts at time $t = 0$ s, then f_o is also the starting frequency. The frequency variation produced by such a time-varying angle is called **frequency (FM) modulation**. This kind of signal is an example of a frequency modulated signal. More generally, we often consider them to be part of a larger class called **angle modulated** signals. Finally, since the linear variation of the frequency can produce an audible sound similar to a siren or a chirp, the linear-FM signals are also called "chirps." In nature, animals such as bats and dolphins produce chirp signals in the ultrasonic frequency range for echolocation and communication.

To get a feeling about the physical implication of the linearly changing instantaneous frequency, let us compute $x_5(t) = \cos(2\pi\mu t^2 + 2\pi f_o t + \phi)$ and listen to it. As a test case, generate a chirp signal whose frequency starts at 2500 Hz and ends at 500 Hz over a time duration of 2.0 s. Use a sampling interval of $T_s = 0.01$ s. Calculate the value of μ. Listen to the chirp using the `soundsc(.)` function. Then modify it so that the starting frequency is 500 Hz and ending frequency is 2500 Hz for the same duration. Listen to the chirp using the `soundsc(.)` function again. How is it different from the first chirp? What happens if you double the μ coefficient? What happens if you halve it? Comment on the changes you observe in your report.

A Chirp Puzzle

Synthesize another chirp signal with the following parameters:

A total time duration of 3.0 s, with a sampling interval of $T_s = 0.01$ s.

The instantaneous frequency starts at 3000 Hz and ends at –2000 Hz (negative frequency).

Listen to the signal. Does it chirp down, chirp up, or both? Use the theory of the frequency spectrum (with its positive and negative frequency components) to help explain what you hear. The changing instantaneous frequency implies that the frequency components in the spectrum are moving.

Part 6: Composing Music

Write the following function:

```
function [note] = notecreate(frq_no, dur)
  note = sin(2*pi*[1:dur]/8192*(440*2.^((frq_no-1)/12)));
end
```

Then, write the following script:

```
notename = [``A'', ``A#'', ``B'', ``C'', ``C#'', ``D'', ``D#'', ``E'',
``F'', ``F#'', ``G'', ``G#''];
song = [``A'', ``A'', ``E'', ``E'', ``F#'', ``F#'', ``E'', ``E'', ``D'',
``D'', ``C#'', ``C#'', ``B'', ``B'', ``A'', ``A''];
for k1 = 1:length(song)
  idx = strcmp(song(k1), notename);
  songidx(k1) = find(idx);
end
dur = 0.3*8192;
songnote = [ ];
for k1 = 1:length(songidx)
  songnote = [songnote; [notecreate(songidx(k1),dur) zeros(1,75)]'];
end
soundsc(songnote, 8192)
```

Try to understand the code and try to compose a simple song. Try to add some different sound effects on it. Have fun!

Note: In this assignment, you are *not* allowed to use symbolic operations in MATLAB. To modify the styles of the plots, to add labels, and to scale the plots, use only MATLAB commands; do *not* use the GUI of the figure windows. When your program is executed, the figures must appear exactly the same as you provide in your solution. You need to write your MATLAB codes not only correctly but efficiently as well.

Submit the results of your own work in the form of a well-documented report on Moodle. Borrowing full or partial code from your peers or elsewhere is not allowed and will be punished. Please include all evidence (plots, screen dumps, MATLAB codes, MATLAB command window print-outs, etc.) as needed in your report. Append your MATLAB code at the end of your assignment, do not upload it separately. The axes of all plots should be scaled and labeled. Typing your report instead of handwriting some parts will be better. Please do not upload any photos/images of your report. Your complete report should be uploaded on Moodle as a single good-quality pdf file by the given deadline. Please try to upload several hours before the deadline to avoid last minute problems that may cause you to miss the deadline. Please DO NOT submit any hardcopies or files by e-mail or on memory stick/CD.