ENGIN 100: Music Signal Processing

LAB #2: Frequencies of Musical Tones

Professor Andrew E. Yagle

Dept. of EECS, The University of Michigan, Ann Arbor, MI 48109-2122

I. ABSTRACT

This lab begins the study of music by determining the frequencies of musical notes. The goals of this lab are: (1) To be able to use a simple signal processing algorithm for computing frequencies of sampled sinusoids; (2) To use this algorithm to compute the frequencies of musical tones in a pure tonal version of "The Victors"; (3) To use simple data visualization to determine the relations between these frequencies. The result will be the 12-tone chromatic scale used for most Western music. You will need to know these frequencies to build simple music synthesizers and transcribers in Projects #1 and #3.

II. BACKGROUND

We now make a first stab at analyzing musical signals. It makes sense to start with the simplest musical signals: pure tones. Even an untrained ear can sense that these are the simplest signals.

We will analyze a basic tonal version of "The Victors" (if you don't know what "The Victors" is, your admission to the University of Michigan may be rescinded!). We will quickly discover that these are sinusoids of different frequencies. We then need a way of computing the frequencies of sinusoids from their samples. Then we need a way of interpreting these frequencies: why them? What are the relations between them? How can we compute them from the tonal signals?

III. FREQUENCY COMPUTATION

We will make extensive use of the cosine addition formulae

$$\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y), \tag{1}$$

$$\cos(x-y) = \cos(x)\cos(y) + \sin(x)\sin(y). \tag{2}$$

Adding and subtracting these gives

$$2\cos(x)\cos(y) = \cos(x+y) + \cos(x-y). \tag{3}$$

This formula will yield two different methods for determining frequency.

A. Tuning Fork Method

This is best illustrated using an example. Let $x = 2\pi 441t$ and $y = 2\pi t$ in (3). This gives

$$\cos(2\pi 440t) + \cos(2\pi 442t) = 2\cos(2\pi t)\cos(2\pi 441t) \tag{4}$$

Generalizing, this shows that the sum of two sinusoids with the same amplitude and with approximately equal frequencies f and f+Δ is equal to a sinusoid having frequency midway between these frequencies with amplitude that **varies sinusoidally with frequency Δ**. You heard an example of this in Lab #1.

Hence one way of measuring the frequency of a sinusoid is as follows:

1. Listen to the sum of the sinusoid and another sinusoid at a known and similar frequency;
2. This sum will sound like this: "loud-soft-loud-soft," etc. The period of this variation is $\frac{1}{\Delta}$;
3. Vary the known frequency until this "beat" disappears. Then the two sinusoids have the same frequency.

Musicians will recognize this as the way pianos are tuned using a tuning fork. You don't even have to know the frequencies, as long as the tuning fork is at the correct frequency.

"You Can Tune a Piano, But You Can't Tuna Fish"–REO Speedwagon's first album.

B. Signal Processing

We prefer an automatic procedure that operates directly on samples of the sinusoid and uses a computer to compute its frequency. We can do this as follows. As motivation, note that

$$x(t) = A\cos(2\pi ft + \theta) \rightarrow \frac{d^2x}{dt^2} = -(2\pi f)^2 x(t) \rightarrow (2\pi f)^2 = -\frac{1}{x(t)}\frac{d^2x}{dt^2}. \tag{5}$$

If you don't know what a derivative is yet, don't worry–you don't need to know. But this suggests that a discretization of this formula could be used to compute the frequency of a sinusoid from its samples.

Indeed, let $x[n]$ be samples of the sinusoid every Δ seconds (recall this from Lab #1):

$$x[n] = x(t = n\Delta) = A\cos(2\pi ft + \theta)|_{t=n\Delta} = A\cos(2\pi fn\Delta + \theta). \tag{6}$$

Setting $x = 2\pi fn\Delta + \theta$ and $y = 2\pi f\Delta$ in (3) and multiplying by A gives

$$A\cos(2\pi f(n+1)\Delta + \theta) + A\cos(2\pi f(n-1)\Delta + \theta) = 2A\cos(2\pi fn\Delta + \theta)\cos(2\pi f\Delta). \tag{7}$$

Using (6), this can be recognized as (compare to the derivative formula)

$$x[n+1] + x[n-1] = 2\cos(2\pi f\Delta)x[n] \rightarrow f = \frac{1}{2\pi\Delta}\cos^{-1}[(x[n+1] + x[n-1])/(2x[n])]. \tag{8}$$

This is a simple formula for computing the frequency f from samples $x[n]$ of a sinusoidal signal. It also has the advantage of detecting *changes* in f quickly (in a couple of time samples)–frequency *tracking*.

IV. DATA VISUALIZATION

Recall the steeply-dropping curve that depicts the radioactive decay of a radioisotope. Then recall the steeply-dropping curve that depicts the cable holding up a suspension bridge, from a tower to the midpoint of the bridge. They look similar, but are they?

Suppose we believe two sets of data x and y are related by $y = ax^n$ for two constants a and n (n need not be an integer). How can we test this hypothesis and determine a and n? Taking logarithms gives

$$y = ax^n \rightarrow \log(y) = n\log(x) + \log(a) \tag{9}$$

So a plot of $\log(y)$ vs. $\log(x)$ is a straight line with slope=n and y-intercept=$\log a$. Making such a plot is a quick way of testing this hypothesis and determining n and a.

If we think x and y are related by $y = ab^x$ for some constants a and b, we can plot $\log(y)$ vs. x since

$$y = ab^x \rightarrow \log(y) = x\log(b) + \log(a) \tag{10}$$

which is a straight line with slope $\log(b)$ and y-intercept $\log(a)$.

If you are too lazy to compute logarithms of your data, you can plot your data directly on log-log or semi-log paper, which have non-uniform scaling that takes the logarithms for you. This has the advantage that the data values themselves, not their logarithms, are on the axes. Matlab's `semilogy` does this.

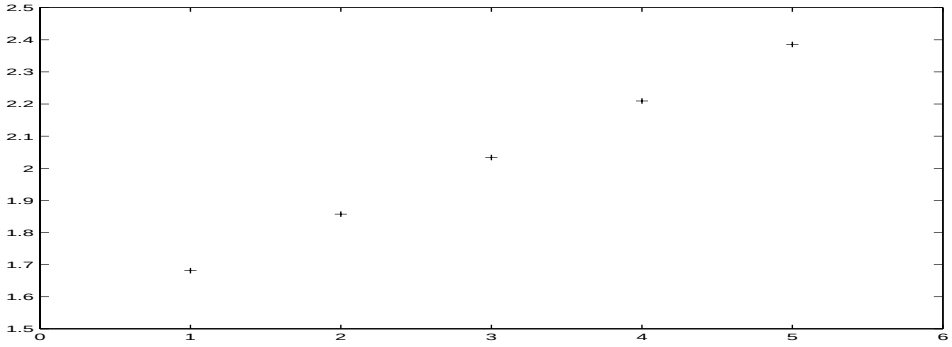
Replotting the two curves mentioned above as semi-log and log-log plots reveals that radioactive decay follows $y = ab^x$ where $-1/\log_2(b)$ is the half-life, and the suspension bridge cable curve is a parabola $y = ax^2$. (This should not be confused with a freely-hanging cable curve, which is a catenary.)

A. Example #1: Semi-Log Plots

We observe the data in the table below left. We wish to determine the function $x(t)$.

A log-log plot of the data gives a curve, but a semi-log plot of the data gives a straight line (below right). Its slope is (2.4-1.7)/(5-1)=0.175=$\log_{10}$1.5 and its y-intercept is 1.505=$\log_{10}$32. So $\mathbf{x(t)} = \mathbf{32(1.5)^t}$.

t	x(t)
1	48
2	72
3	108
4	162
5	243



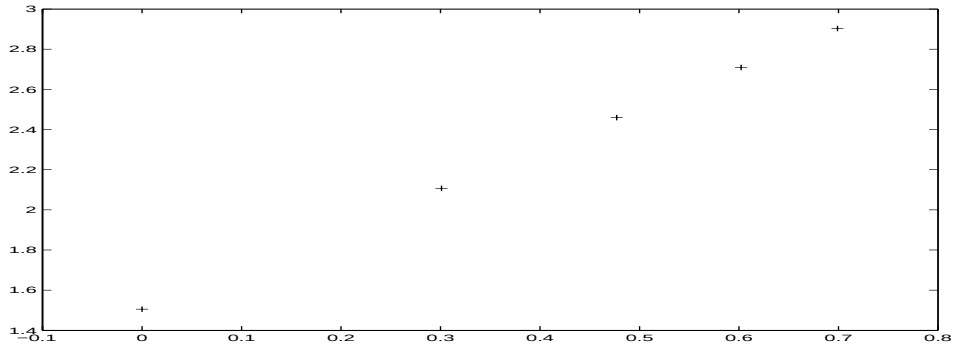
B. Example #2: Log-Log Plots

We observe the data in the table below left. We wish to determine the function $x(t)$.

A semi-log plot of the data gives a curve, but a log-log plot of the data gives a straight line (below right).

Its slope is (2.9-1.5)/(0.7-0)=2=exponent and its y-intercept is 1.5=$\log_{10}32$. So $\mathbf{x(t) = 32t^2}$.

t	x(t)
1	32
2	128
3	288
4	512
5	800



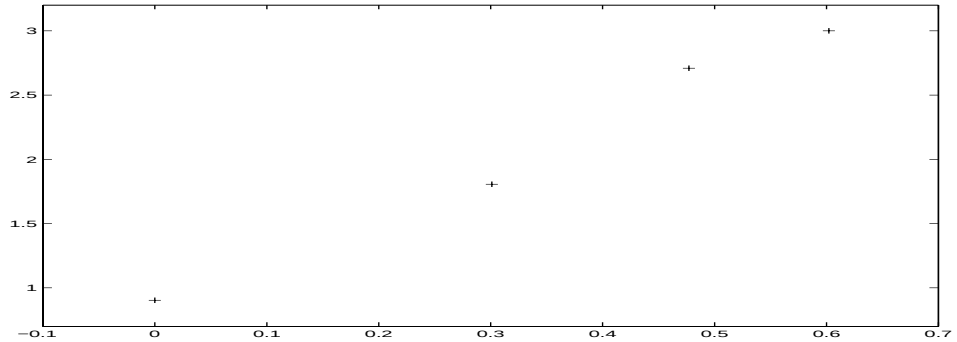
C. Example #3: Determination of Missing Data

Now we know $x(t) = \{8, 64, 512, 1000\}$ for *some subset* of $t = \{1, 2, 3, 4, 5\}$. We can *still* find $x(t)$!

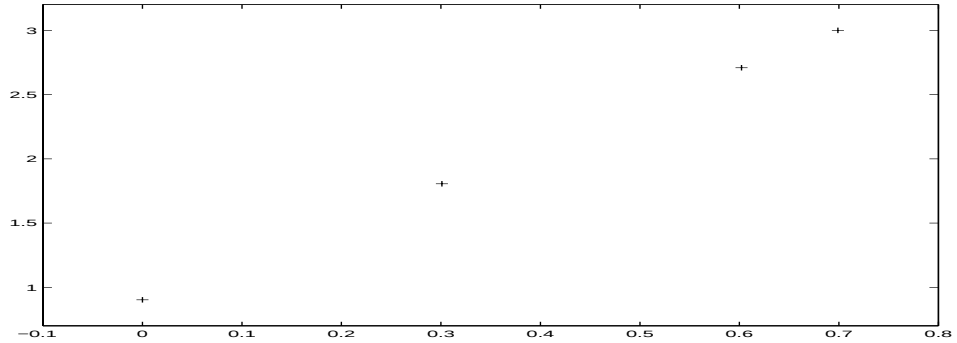
A semi-log plot gives a curve, but a log-log plot gives a straight line with a break$\rightarrow t = 3$ is missing.

The missing value $x(t)$ at $t = 3$ can be seen to be just under 300 (actually it is 288). We can fill it in!

The slope of the 2 segments is 3; the slope of the steeper segment is 6. The y-intercept is 8, so $\mathbf{x(t) = 8t^3}$.



Two log-log plots of data: vs. $\{1, 2, 3, 4\}$ above; vs. $\{1, 2, 4, 5\}$ below, "straightening out" the break.



V. LAB #2: WHAT YOU HAVE TO DO

A. Frequency Measurements

- Download "victorstone.mat" and type >> `load victorstone.mat;sound(X)`. Recognize this?
- Type >> `T(2:77999)=acos((X(3:78000)+X(1:77998))./X(2:77999)/2)/(2*pi/8192);plot(T)`

This implements the above frequency estimation formula (Δ=1/8192 seconds). (Don't print this plot.)

- Type >> `Y(2:77999)=(X(3:78000)+X(1:77998))./X(2:77999)/2;subplot(211),plot(Y)`
- Now explain what happened in the previous plot, and why this formula is much better that the first one.
- Explain why this could actually be helpful in *segmenting* the signal into separate tones.
- Type >> `Y(78000)=0; Z=reshape(Y,3000,26);F=acos(mean(Z(2:2999,:)))/(2*pi/8192);F=sort(F)`
- This eliminates the first and last numbers in each segment, which are *outliers*, takes the mean (average) of the numbers in each segment, and sorts the result. Are these reasonable estimates for frequency in Hertz?
- Type >> `wavwrite(X,8192,'victors');X=wavread('victors');`
- Type >> `Y(2:77999)=(X(3:78000)+X(1:77998))./X(2:77999)/2;subplot(212),plot(Y)`
- This writes the .wav file to X and implements the same formula as before. What changed?

B. Data Visualization

- Type >> `F1=[?  ?  ?  ?  ?  ?]` where each ? is a frequency (in F) you found, in ascending order.
- Type >> `subplot(211),plot(F1,'+')`. Within each of 2 segments are the frequencies linearly related?
- Type >> `subplot(212),plot(log(F1),'+')`. Compare slopes: 2 segments and the break in the line.
- Comparing to the example above, note that there are missing frequencies *except* for the break in the line. That is, the break in the line is the only range within which there *aren't* any missing frequencies!
- Fit an appropriate model to the six frequencies that also includes the frequencies that do not appear.
- Give numerical values for all 12 of the frequencies, whether they appear or not.

C. Lab Report

Summarize your results in a question-and-answer format. The audience will be your lab instructor, not your discussion instructor. Answer the questions posed in the lab in one sentence each. Include all plots, and list the frequencies you measured. and those you inferred. This assignment is due in lab next week.

D. Next Week

Now that you know what pure musical tones are, and their frequencies, you will write a Matlab program for a simple musical tone synthesizer with a Matlab graphical user interface (i.e., on-screen keyboard) that generates musical tones. Then you will write a Matlab program for a simple musical note analyzer that, when given a pure tonal musical signal, produces a Matlab stem plot resembling musical staff notation.