**Fourier Series: The Math Behind the Music**

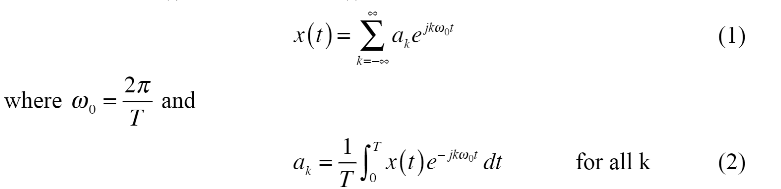
**Introduction:**

You have been working at an Electronics Team Gurus (ETG) for several years, but they just transferred you to the Audio Engineering Department. It has been almost a decade since you took EE 224, so it might be time to review a little bit of math behind the music. To start, you dig through your closet to find your dusty Signals and Systems book. Upon skimming through the pages, you find some old notes about Fourier Series equations.

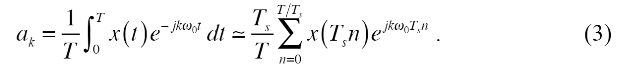
**Prelab:**

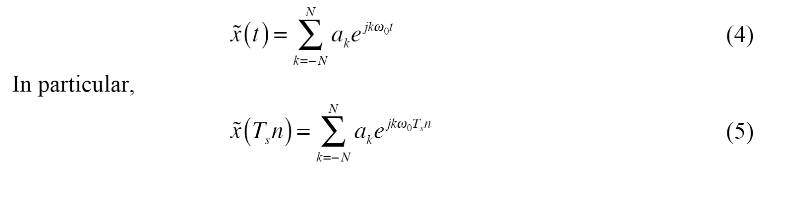
Before coming to work, or in this case lab, you need to make sure that you are familiar with the math behind the music.

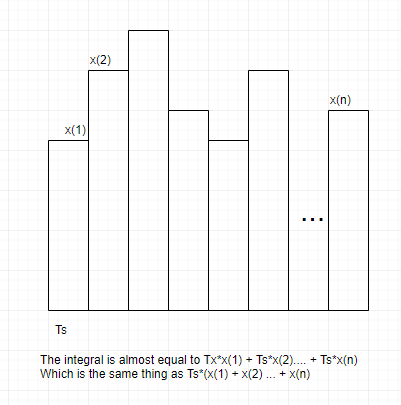
Fourier Analysis is the process in which a signal can be approximated using a sum of sinusoids. To make the equations easier to write, polar notation is used for the sinusoids. Equation 1 is for the reconstruction of a signal using the sinusoids. To find the coefficients for a\_k, equation 2 is used. In equation 2, T is the period, a\_k is the coefficient to be found, and k corresponds the coefficient number. x(t) is the corresponding function for the signal that is being analyzed. Use equation 2 to calculate a\_k -3 to a\_k 3 for a square wave that is 1 from 0 seconds to 1 second an then 0 from 1 second to 2 seconds. Assume the period is 2. Also, w0 = 2\*pi\*f and f = ½. So, w0 = pi. (hint: a\_0 is the DC offset of a signal. It should just be the average value of that particular signal).



Equation 2 is fine to do once or twice by hand, but several values of a\_k are needed to appropriately approximate a periodic waveform. So, it is convenient to do it in MatLab. However, MATLAB does things discretely. It is possible to use a Reiman’s Sum instead. For a function x[n], just take the value of x[n] and then multiply it by the distance to the away from the previous value. In this case, the distance between values the sample time Ts. So, the some would be Ts\*x[1] + Ts\*x[2] …. +Ts\*x[n]. Since every value is multiplied by Ts, it can be pulled to the outside of the sum as seen in equation 3. The figure below the equations gives a visual representation of what is happening. **This is important as it will be used in the first part of lab. Make sure you understand this because it will make it easier to program in MatLab.**







Ts = 1/fs, in that the sample time is 1 divided by the sample rate.

T is the number of samples multiplied by Ts. So, the number of samples is T/Ts.

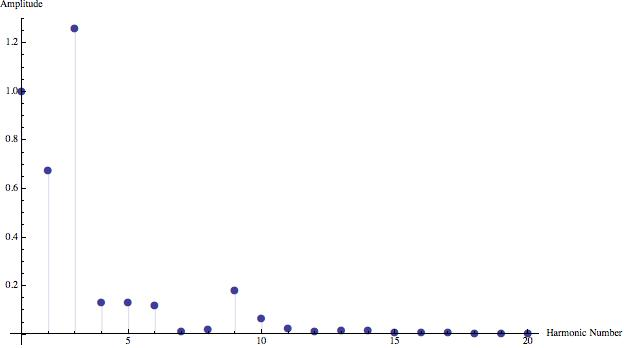
If the fundamental frequency that an instrument is playing is 440 hz, corresponding to a1, then the next harmonic is k\*440 hz. What is the second, third and fourth harmonics? (hint: these correspond to a2, a3, and a4 where k = 2, 3 and 4 respectively.

If the trumpet was sampled at 44.1 kHz and there are 170 samples in a period, what is the fundamental frequency the trumpet’s note? How long is the period? This concept will be used later in the lab, so remember how it is done.

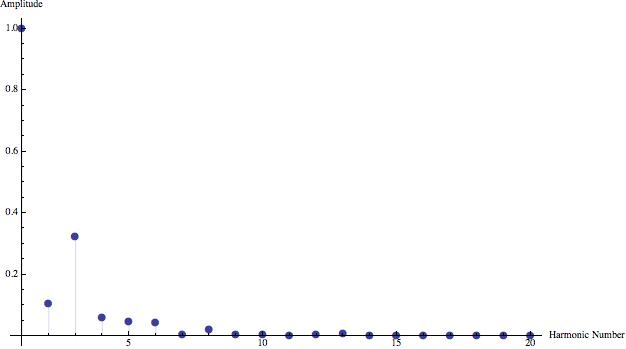
The note that is heard when an instrument is played is the fundamental frequency. However, a piano and a trumpet do not sound the same even if they are playing they are playing a note at the same frequency. This is because each instrument has a different “timbre,” or sound to it. This is because an instrument playing a C at 440 hz will have different magnitudes for the harmonics. In other words, the instruments will have different values for a2, a3, a4 ect. Changing these harmonics will affect how the instrument sounds.

Pianos are known for having a “pure” sound. If listening to the fundamental frequency is most important when tuning an instrument, why is a piano preferred for tuning instead of a guitar? (hint: the fundamental frequency is when the harmonic number k = 1). The uniqueness of the harmonics make instruments sound more pure, distorted, warm, or even bright. This is referred to as the “timbre” of an instrument. As an electrical engineer, it is possible to use Fourier Analysis to synthesize the “timbre” of an instrument or even someone’s voice.

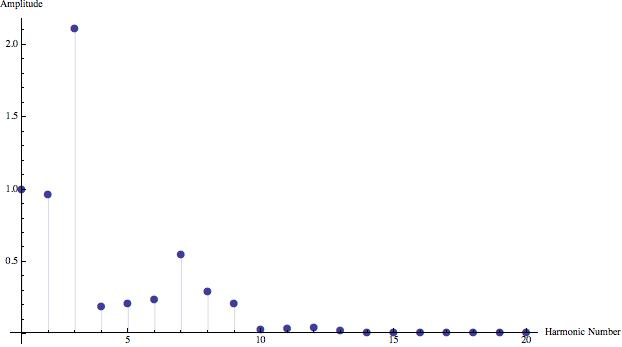
**Guitar:**



**Piano:**



**Oboe:**

****

Optional: bring an instrument or other device that can create a constant tone.

**Experiment 1:**

Doing the calculations for a single coefficient is simple enough, but doing so for thousands of them can be a bit tedious. As an audio engineer, you will be tasked with sampling and recreating many different signals. So, you decide to be proactive and write a function in MatLab that will do the calculations for you.

Create a function that can compute a coefficient of a Fourier Series by taking the sampling rate, the desired coefficient and the sample itself.

The function should be like so: find\_ak(x, k, fs)

Where x is the samples

K is the coefficient to be found

And fs is the sampling rate

Use **gen\_square** to create a square wave. (I would like to use the CyDAQ to sample a wave when it is completed).

Test find\_ak.m on the square wave to find a-2- a1. Check with a TA before moving on because this one function is essential for the rest of the lab. (Assume the sample rate it 44100 samples/second).

Use the function **find\_aks.m** and then compare the results of a0 to those in the prelab.

Create a function that synthesizes an approximate waveform by taking in a\_k coefficients.

% C is the vector of a\_k coeficients

% f is the desired frequency to recreate

% fs is the desired new sample rate

% It can be used to control the

% amount of sample time between functions

% ss is the starting point of a\_k in C

% ff is the ending point of a\_k in C

% but ff may or may not be necessary

% depending on the implementation

function synth = fsynt(C, f, fs)

Use this function to synthesize the square wave using a\_ks -10 to 10. Using plots in MatLab, compare this to a synthesized waveform that has a\_ks from -1 to 1 and then one that has a\_ks from -100 to 100.

What happens as the a\_ks approach -infinity to infinity?

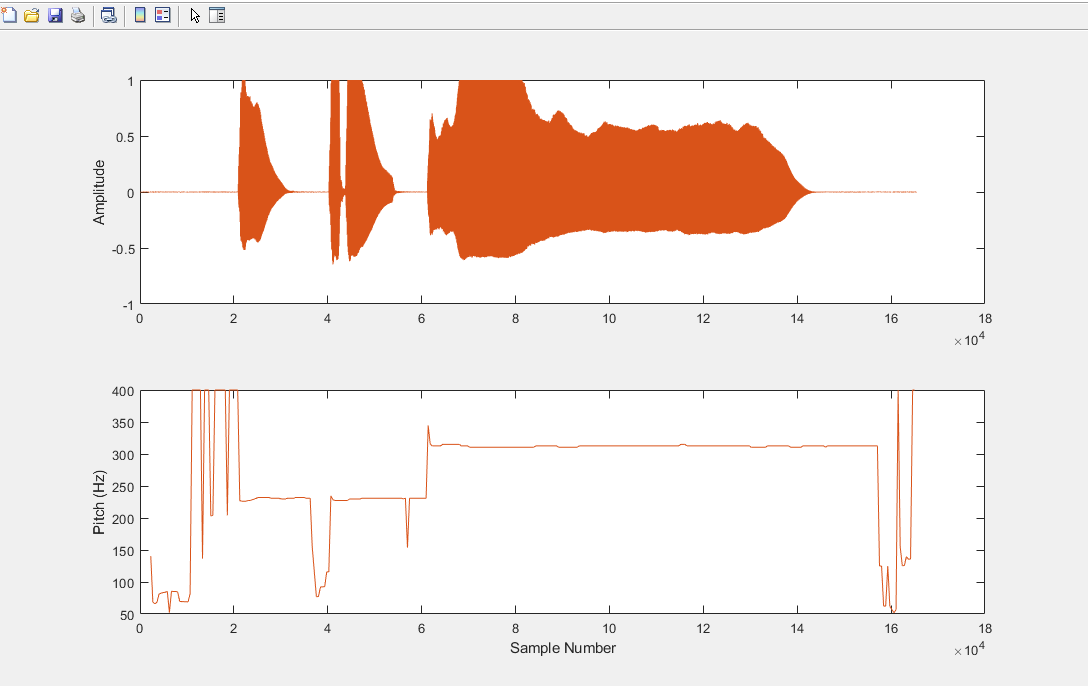
**Experiment 2:**

Since you have shown initiative, your boss has put you in charge of the synthesizer group. Your boss asked you to create sounds that will be played on the company's new line of digital keyboards. To save space in the memory, it was proposed to use fourier coefficients to store the data instead of storing the sampled waveforms.

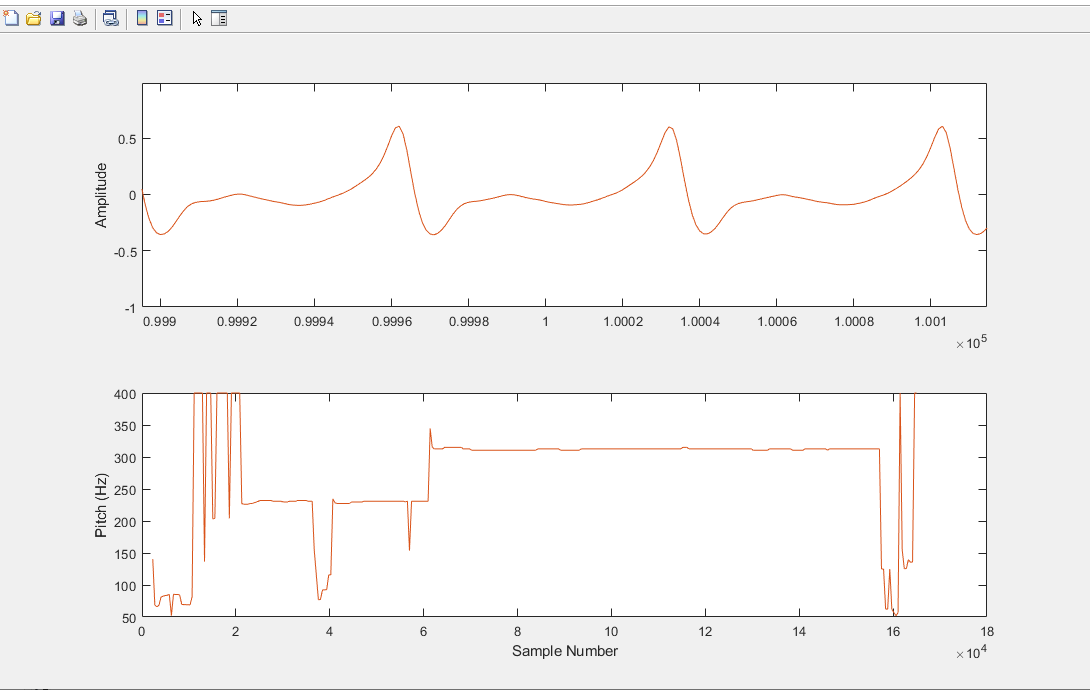
Use the CyDAQ to sample your voice, a tuning fork, musical instrument, or other device than can create a consistent tone. Since the CyDAQ is not up and going as of yet, use the trumpet.wav file for now. You could also record something using another mic. You only have to be able to get it into a .wav file format.

To find info about the sample waveform, use **info = audioinfo(filename.wav)**

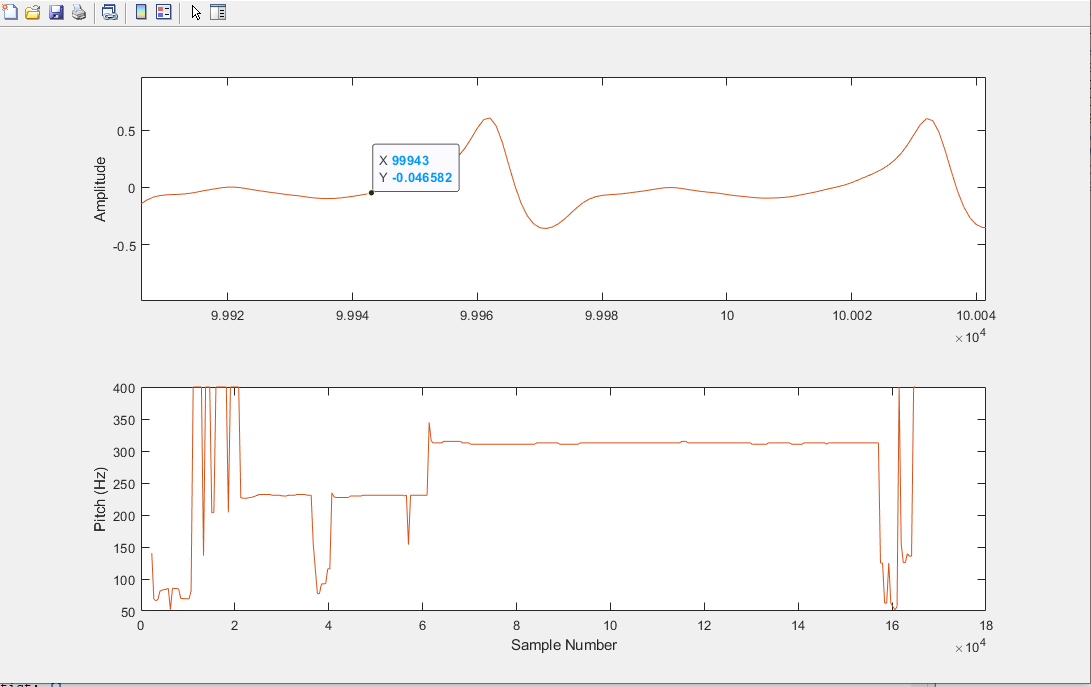
Use the provided function **PlotPitch.m** to plot the pitch. Find several areas that the pitch is consistent. Take the sample rate of the .wav file and the frequency found from PlotPitch.m to figure out how many samples long the fundamental frequency is. Your plot should look like the one below. (Hint: The fundamental frequency is in called “Pitch) on the bottom graph).



Knowing how many samples long a single Period is, zoom in to a section that has a constant tone. Go to Tools, Zoom in, and then click and draw until you can identify a periodic waveform.



Use audioinfo(‘filename.wav’) to find the sampling rate if it is unknown. Use the sample rate and the frequency in the Pitch plot to find the number of samples in one period. Use the Data Cursor found under tools to find a starting point in the plot.



Once a starting point and the number of samples in a period have been found, CutSample.m to create a single set of samples to work with. Use the functions that were created in Experiment 1 to find the a\_ks from -10 to 10. Plot their **magnitude.** How “pure” is the sound of the instrument that was sampled? (hint: compare the fundamental frequency to the other harmonics).

Use fsynt.m to recreate the sound file. Plot the original waveform alongside the reconstructed one. (hint: the exactanes of the approximated waveform will depend on the number of a\_ks used). Then, use AddPeriods to add the amount of periods required to play the reconstructed sound for 5 seconds (find out how long the period is and how many periods are needed to get close to 5 seconds). Then, play the sound to a TA. (Hint: Use soundsc(sample, sampleRate)).

Bonus points: Play a C major chord (hint: a C, E and G playing at the same time. 261.63 hz, 329.63hz and 392 hz.

Bonus points: Use the CyDAQ to make a keyboard with the notes C, E and G.

<https://www.projectrhea.org/rhea/index.php/Fourier_analysis_in_Music>