Fourier Series and Gibbs Phenomenon

Version 1.21: Oct 24, 2008 11:52 pm GMT-5

University Of Washington Dept. of Electrical Engineering

Abstract

Fourier series, sums of cosines.

This development of these labs was supported by the National Science Foundation under Grant No. DUE-0511635. Any opinions, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

In this lab, we will look at the Fourier series representation of periodic signals using MATLAB. In particular, we will study the truncated Fourier series reconstruction of a periodic function.

2 Some Useful MATLAB Commands

- `abs`, compute the complex magnitude.
- `angle`, compute the phase angle.
- `clear`, clears all variables.
- `help <command>`, online help.
- `whos`, list all variables and their sizes.

3 Signal Synthesis

We will see in exercise 3 that we can approximate a square wave with the Fourier series, but first let us approximate something more interesting, say a musical instrument? Many instruments produce very periodic waveforms.

Exercise 1 Synthesizer

1. Create a script file called `sigsynth.m` to put your code in for this problem.
2. Download the trumpet sound sample `trumpet.mat` from the Sound Resources[1] page. The sample rate, `Fs`, of the trumpet is 11,025 Hz. Play this sound with the `sound` command (remember to include the correct sample rate).

*http://creativecommons.org/licenses/by/2.0/

[1]"Sound resources" <http://cnx.org/content/m13854/latest/>

3. Plot only a small section of the trumpet sound to show three or so periods (try 100 samples or so). Does it looks the same at any time in the sound?
4. View the frequency spectrum of this sound by entering the following commands,

```
    Fs = 11025;             % our sample rate is 11025 Hz
Y = fft(trumpet, 512);  % take the fft of trumpet
Ymag = abs(Y);          % take the mag of Y
f = Fs * (0:256)/512;   % get a meaningful axis
plot(f, Ymag(1:257));   % plot Ymag (only half the points are needed)
xlabel('Frequency (Hz)')
ylabel('Magnitude')
```

   You should now see a series of peaks (these are the harmonics of the instrument).
5. We will synthesize the instrument using only the peak information. You can use the "data cursor" tool in MATLAB's figure window to easily read graph data. Write down the frequency and its strength (magnitude) for five to ten of the strongest peaks.
6. Create a function called `addcosines.m` that takes in three vectors: time vector `t`, frequency vector `freq`, and magnitude vector `mag`. Have your new function use a for-loop to add together cosines, one for each frequency/magnitude pair in the `freq` and `mag` vectors. Remember to normalize your output vector after you add up all the cosines (the output should be between -1 and 1), like in the Functions in MATLAB and the Groove Station[2] lab. Use the data you collected from the frequency plot of the trumpet sound with your new function to sum cosines at the noted frequencies.
7. Here are some hints for the above. Use a for-loop to create a cosine at each frequency in the freq vector. Your cosine function should look something like this, `mag(i)*cos(2*pi*freq(i)*t);`. Remember your time vector will have the form 0:1/Fs:time_in_seconds.

   NOTE: The command `soundsc` will normalize the input before it plays the sound.

   For example, if you had two harmonics, one at 100 Hz with magnitude 1 and another at 150 Hz with magnitude 2, then your vectors will be,

```
    t = 0:1/Fs:1;  % one second time vector at 11025 Hz
freq = [100 150];
mag = [1 2];
```

8. Play trumpet and your new synthesized sound. Do they sound the same? Use subplot to plot a small section of your new synthesized sound along with the trumpet sound, does it look the same? Save your plot as `synthwaves.tif`.
9. Try synthesizing the sound with fewer frequencies, then try more frequencies. How does this affect the sound of our synthesized trumpet?
10. You will need to show the TA the following files:

```
sigsynth.m
addcosines.m
synthwaves.tif
```

[2]"Functions in MATLAB and the Groove Station" <http://cnx.org/content/m13555/latest/>

Exercise 2

That funny phase You probably noticed in the last problem that even though the wave forms looked fairly different, the sound was similar. Let's look into this a bit deeper with a simpler sound.

1. Create a script file called `phasefun.m` to put your code in for this problem.
2. Pick two harmonic frequencies and generate a signal from two cosines at these frequencies added together and call it `sig1`. Use Fs = 8000 (remember that you can reproduce only frequencies that are less than `Fs/2`).
3. Now generate a second signal called `sig2` exactly the same as the first one, except time delay the second cosine by a half cycle (half of its period).
4. Use subplot to show a few periods of both signals, do they look different? Save the plot as `phasesigs.tif`. What did the time delay do to the phase?
5. Play each signal with `soundsc`, do they sound different?
6. Redo `sig2` with a few different delays and compare the sound to the first signal.
7. Create a `sig3` that is one cosine at some frequency. Now add `sig3` with a timed delayed version of itself and call it `sig4`. Use a quarter cycle delay.
8. Use subplot and plot a few periods of `sig3` and `sig4`. Play them with `soundsc`, do they sound different to you?
9. What is suggested about our hearing capabilities from this experiment?
10. You will need to show the TA the following files:

```
phasefun.m
phasesigs.tif
```

4 Truncated Fourier Series

In this section, we'll reconstruct the periodic function `x(t)`, shown in Figure 1, by synthesizing a periodic signal from a variable number of Fourier Series coefficients, and observe similarities and differences in the synthesized signal.

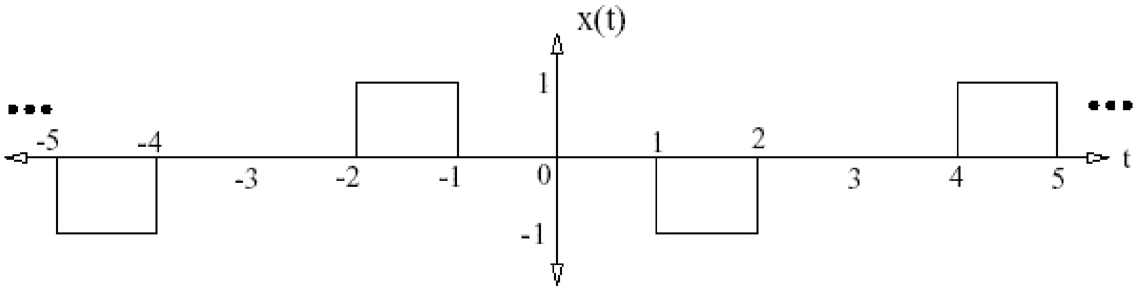

Figure 1: Periodic Signal

Exercise 3 Gibbs phenomena

1. Create a script file called `gibbs.m` to put your code in for this problem.
2. Click here[3] to download the MATLAB function `Ck.m`. Take a look at the contents of the function. This function takes one argument k , and creates the k th Fourier series coefficient for the squarewave above:

$$C_k = \{ \begin{array}{ll} 0 & \text{if } k = 0,\ k \text{ even} \\ \frac{1}{jk\pi}\left[\cos\left(\frac{2k\pi}{3}\right) - \cos\left(\frac{k\pi}{3}\right)\right] & \text{if } k \text{ odd} \end{array} \quad (1)$$

   $C_k(1) = \frac{-1}{j\pi} = 0 + j0.3183$. Plot the magnitude and phase of the coefficients C_k for $k \in \{-10, -9, \dots, 9, 10\}$. The magnitude and phase should be plotted separately using the subplot command, with the magnitude plotted in the top half of the window and the phase in the bottom half. Place frequency w on the x axis. Use the MATLAB command `stem` instead of `plot` to emphasize that the coefficients are a function of integer-valued (not continuous) k . Label your plots.
3. Save the graph as `Coeff.tif`.
4. Write whatever script/function files you need to implement the calculation of the signal $x(t)$ with a truncated Fourier series:

$$\begin{array}{rcl} x(t) & = & \sum_{k=-K_{\max}}^{K_{\max}} \left(C_k e^{\mathrm{j}\mathrm{k}\omega_0 t}\right) \\ & = & \sum_{k=0}^{K_{\max}} \left(2|C_k| cos\left(k\omega_0 t + \angle C_k\right)\right) \end{array} \quad (2)$$

   for a given K_{max}

   NOTE: You can avoid numerical problems and ensure a real answer if you use the cosine form. For this example, $w_0 = 1$.
5. Produce plots of $x(t)$ for $t \in [-5, 5]$ for each of the following cases: $K_{max} = 5$; 15; and 30. For all the plots, use as your time values the MATLAB vector `t=-5:.01:5`. Stack the three plots in a single figure using the `subplot` command and include your name in the title of the figure. Save the figure as `FourTrunc.tif`
6. Add clear comments describing what the files do. You will need to show the TA the following files:

```
gibbs.m
Coeff.tif
FourTrunc.tif
```

As you add more cosines you'll note that you do get closer to the square wave (in terms of squared error), but that at the edges there is some undershoot and overshoot that becomes shorter in time, but the magnitude of the undershoot and overshoot stay large. This persistent undershoot and overshoot at edges is called Gibbs Phenomenon.

In general, this kind of "ringing" occurs at discontinuities if you try to synthesize a sharp edge out of too few low frequencies. Or, if you start with a real signal and filter out its higher frequencies, it is "as if" you had synthesized the signal from low frequencies. Thus, low-pass filtering (a filter that only passes low-frequencies) will also cause this kind of ringing.

For example, when compressing an audio signal, higher frequencies are usually removed (that is, the audio signal is low-pass filtered). Then, if there is an impulse edge or "attack" in the music, ringing will occur. However, the ringing (called "pre-echo" in audio) can be heard only before the attack, because the attack masks the ringing that comes after it (this masking effect happens in your head). High-quality MP3 systems put a lot of effort into detecting attacks and processing the signals to avoid pre-echo.

[3]http://cnx.org/content/m13599/latest/Ck.m

5 What to Show the TA

Show the TA ALL m-files that you created or edited and the files below.

```
gibbs.m
Coeff.tif
FourTrunc.tif
sigsynth.m
addcosines.m
synthwaves.tif
phasefun.m
phasesigs.tif
any wav files created
```

6 Fun Links

An applet here[4] provides a great interface for listening to sinusoids and their harmonics. There are some well-known auditory illusions associated with the perception of pitch here[5] .

[4]http://www.phy.ntnu.edu.tw/ntnujava/viewtopic.php?t=33
[5]http://physics.mtsu.edu/∼wmr/julianna.html