Matlab Project 1: Fourier series refresher

EE-125

This project should serve as a refresher for Fourier series. Fourier series are the basis of the DTFT (discrete-time Fourier transforms) and other transforms we will use in this class.

The continuous-time Fourier series is discussed in section 4.1.1 of the textbook. Fourier's idea was that it is possible to write any signal as a sum of sinusoidal functions:

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi k F_0 t} \tag{1}$$

where c_k are the Fourier coefficients, F_0 is the fundamental frequency (in Hz), and t is a vector of times in seconds. Hopefully this looks familiar! The Fourier series is sometimes written as

$$x(t) = \sum_{k=-\infty}^{\infty} c_k \left[\cos\left(2\pi k F_0 t\right) + j \sin\left(2\pi k F_0 t\right)\right] \tag{2}$$

which is mathematically completely equivalent, using Euler's formula. However, in EE-125 we will use the form of Eq. 1. Note that kF_0 defines a new frequency which is a multiple of F_0.

In the equation above, time is assumed to be continuous. However, in DSP we work with digitized or 'digital' signals (this is the 'D' in 'DSP'). Almost always, analog-to-digital converters (ADCs) will sample data at evenly spaced time intervals. Thus, samples may be taken, for example, at times $0, T, 2T, 3T, 4T, ...$ seconds, where T is called the *sampling period* in seconds. We can also define T in terms of a *sampling rate* or *sampling frequency* $f_s = 1/T$. The units of f_s are (number of samples) / second, or Hz (since 'number of samples' is just a number and thus is unitless). Then the digitized samples are at times $0, 1/f_s, 2/f_s, 3/f_s, 4/f_s, ...$ and so on. In Matlab, we can set up this time sampling as shown below.

```
fs = 10; % example sampling frequecy
t = 0:(1/fs):5;  % set up times from 0 to 5 sec, every 0.1 sec
```

1 Exploring the Fourier terms

Do each of the following steps in term. When you turn in your report, please label your results by section (1.1, 1.2, etc) so it is easy to grade them.

1.1 Write function fourierTerm.m

Write a function called `fourierTerm` that calculates a single term of the Fourier series, i.e. $c_k e^{j2\pi k F_0 t}$. The two lines of the function should be:

```
function fterm = fourierTerm(ck,k,F0,t)
% function fterm = fourierTerm(ck,k,F0,t)
```

where the meaning of the terms should be clear from the equation. The time t should be passed in as a vector, which means the output `fterm` will also be a vector of the same length.

1.2 Results plotting, part 1: common problems

In EE-125, we will be working a lot with complex-valued variables. This section is designed to highlight a few common problems that students encounter in plotting complex vectors, so hopefully you can avoid them later during later projects. In your report, you don't need to include any of the plots you are asked to make, but you *do* need to include the discussion that is requested.

Set up a vector t that has times from 0 to 1 sec, sampled at f_s=100 Hz. Use t as input to `fourierTerm`, to create output vectors with the following parameters (you will want to save the outputs to different variable names):

- $ck = 1, k = 1, F_0 = 1/3$,
- $ck = 1, k = 2, F_0 = 1/3$,
- $ck = 1, k = 3, F_0 = 1/3$,

The resulting vectors should be vectors of complex numbers. Plot these vectors in two different ways. First, plot them vs. time (so if your vector is called `test`, do `plot(t,test)`). Be aware of any warning messages you see. Second, plot them with just a single argument (so if your vector is called `test`, do `plot(test)`). When you plot a complex vector in Matlab, you get the real part on the x-axis and the imaginary part on the y-axis.

Because k and F_0 are multiplied together, you should be able to replicate your $k = 3, F_0 = 1/3$ result above with $k = F_0 = 1$. Verify that is true.

You don't need to include these plots in your report. However, do include a brief discussion that describes what you see and how the plots depends on k and F_0. In interpreting the result, it will be helpful to think about Euler's formula (see Eqs. 1 vs. 2 above).

1.3 Results plotting, part 2: Fourier series interpretation

Now, write a second small function called `plotComplexData`, whose first two lines are:

```
function plotComplexData(t,fterm)
% function plotComplexData(t,fterm)
```

where you can assume that t is a real-valued vector of times and $fterm$ is a (possibly) complex vector of the same length. When you feed `t` and `fterm` into the function, it should create a plot with four labeled subfigures: real part of `fterm` vs. time, imaginary part vs. time, absolute value vs. time, and unwrapped phase vs. time[1]. All subplots should be labeled.

Set up a time vector t from 0 to 2 seconds at $f_s = 100$ Hz. For each of cases below, call `fourierTerm` and then `plotComplexData`. Include a plot of each result in your report, and include the discussion which is asked for:

1. Calculate a single Fourier term ($ck = 1, k = 1, F_0 = 2$). Explain the real and imaginary parts in terms of Euler's formula. Explain the absolute value and phase plots.
2. Consider the case of a complex c_k by computing a term for the case ($ck = e^{j\pi/4}, k = 1, F_0 = 2$). Explain how phase and time shifts are related (you can think about Euler's formula along with the math fact that $e^{ja}\, e^{jb} = e^{j(a+b)}$)
3. Compute and add two Fourier terms. For the first term, use $ck = 1, k = 1, F_0 = 2$. For the 2nd term, use $ck = 1, k = 1, F_0 = -2$. Explain the real and imaginary parts you get in terms of Euler's formula.
4. Compute and add two different Fourier terms. The first term should be computed using $ck = -j, k = 1, F_0 = 2$, while the 2nd term should use $ck = j, k = 1, F_0 = -2$. Explain the real and imaginary parts you get in terms of Euler's formula.

[1] see Matlab functions `real`,`imag`, `unwrap` and `angle`, as well as `subplot` and `figure`

5. Redo the last item, but add a third term $c_0 = 2$ for $k = 0$. How does this change the waveform? What is the interpretation of this term? Does it matter what F_0 you pick for this term? Why?

2 Synthesizing other waveforms

A very useful 'transform pair' that we'll be working with in the class is the 'sinc-boxcar' pair. Without worrying about the details too much at this point, let's do some plotting to build some intuition about Fourier synthesis.

In example 4.1.1 in the book, amplitudes of the Fourier series are computed for a train of rectangular pulses (the rectangular shapes are also called 'boxcars', because they look like boxcars on a freight train). The function `pulseTrainDFS` included with this assignment codes up Eq. 4.1.17-4.1.18, and will return a particular c_k for a given input k, fundamental frequency T_p, and pulse width τ. Take a minute to look at the code and compare it to the equations in the book.

We have the flexibility to pick T_p and τ. Just for fun, let's pick $T_p = 2$ sec and $\tau = 0.5$ sec.

Fourier's formula (Eq. 1) says that if we add up an *infinite* number of terms with c_k values as calculated in Example 4.1.1, we should get a periodic train of rectangular pulses. However, for computer implementation we are limited to a finite number of terms, i.e:

$$x(t) = \sum_{k=-K}^{K} c_k e^{j2\pi k F_0 t} \tag{3}$$

STEP 1: You are given a mostly-empty piece of code called `sumFourierSeries.m`. Fill out the code so it does the following:

- Sets up a new time vector t that goes from -4 to 4 seconds, with a sampling rate of 100 Hz.
- Loops over k, computes c_k and the Fourier term using `fourierTerm`, and carries out the summation shown in Eq. 3.

STEP 2: Then, use your code to compute x(t) for the cases $K = 1, K = 2, K = 3, K = 10, K = 50, K = 100$, and $K = 500$. For each case, plot the real part of $x(t)$ vs. t. Check that the imaginary part is very close to zero - if it isn't close to zero, you have a problem. You should see something that looks at least somewhat similar to Fig. 4.1.3, an ideal pulse train.

STEP 3: Look up 'Gibb's phenomena' and describe your plots in terms of that concept. You don't need to be very mathematically detailed.

3 For fun: vowel synthesis

This is not part of the actual assignment, but is fairly straightforward and you may want to try it...

While speech synthesizers rely on fairly complicated models to get general sounds, vowel sounds are simpler to model and are easily modeled using Fourier series. There is a simple reason for this: the vocal tract can be modeled as a tube that has a fundamental resonance frequency F_0 (when we make a vowel sounds, we open up the vocal tract so it resonanates as a tube). Then, various other resonances can occur at integer multiples of the fundamental frequency (these map to the k values in the Fourier series).

In any case, create a sound where you sum up Fourier series terms listed in the table below (where $\pm k$ means to add values for both positive and negative values of k, using the same c_k). Note that the fundamental frequency F_0 isn't specified, so you can pick it (it may be interesting to relate the frequency to notes on a piano keyboard; middle C on the piano is 262 Hz). Plot the resulting signal using `plotComplexData` and include it in your report. Any imaginary part should be tiny, for reasons we'll discuss later. Play the sound using the Matlab command `soundsc`) (make sure to feed in the sampling rate!). Does it sound like a vowel?

k	c_k
± 2	0.00772 + 0.122j
± 4	-0.0866+0.2805j
± 5	0.48-0.08996j
± 16	0.01656 - 0.1352 j
± 17	0.04724