Numerical Analysis 1 – Class 2

Thursday, January 28th, 2021

Subjects covered

- Sampled data and time series.
- Taking numeric derivatives.
- Complexity – big-O notation.
- Fourier series and Fourier transform.
- Filtering and processing of signals in the time and frequency domain.

Readings

- "Numerical Computing with Matlab", chapter 8, Fourier Analysis, by C. Moler (linked on Canvas).
- Fourier Transform Tutorial by S. Brorson (on Canvas).
- Kutz, Sections 4.1, 10.1, 13.1 – 13.3.

Problems

Most of the following problems require you to write a program. For each program you write, please make sure you also write a test which validates your program. You will be graded on both your program as well as on your test. Additionally, please place the answers to different questions into different directories and zip up your answers into a single zip file. E-mail your answers to our TA: Hiu Ying Man, man.h@northeastern.edu.

Problem 1

For this problem I have created a 10 second excerpt of Beethoven's famous "Ode to Joy", which I played on a piano. It is stored on Blackboard as the file "odetojoy.mat". Inside the file is a single vector, y, which is the sound recorded using a sample rate of 8000 samples/second. In Matlab, you can read in the sound file and listen to it using the following commands:

```
load('odetojoy');
Fs = 8000;  % samp freq
tune = audioplayer(y, Fs);
play(tune)
```

The piano is tuned so that the notes starting at "middle-C" have the following fundamental frequencies (in Hz):

C	C#	D	D#	E	F	F#	G	G#	A	A#	B
261.6	277.2	293.7	311.1	329.6	349.2	370.0	392.0	415.3	440.0	466.2	493.9

Your task is to determine in what key I am playing this tune using the Fourier Transform. (That is, am I playing in the key of C, or D, or E or....?) The easiest way to do this is to compute the spectrum of the of the input tune using the FFT, then determine the frequency axis, and finally plot the spectrum and figure out which of the spectral peaks correspond to the above notes. Keep in mind that there may be some measurement error (and mis-tuning of my piano) so that your measured frequencies might differ from the above values by a couple of Hz.

Please turn in your program which makes the plot. Also turn in the key in which the tune is played, and why you think that is the key – this is a paper and pencil exercise.

If you are not a musician, the key in which this tune is played is the note upon which it resolves musically (in this case, it is the lowest note you hear in the sound snippet).

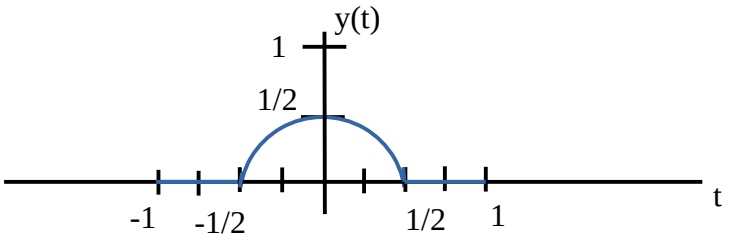
It's hard to think of a test for this program, but you can check your result using Matlab's "spectrogram" function to verify that the spectrum you compute matches that which Matlab computes.

Problem 2

Consider the "circular bump function" $y(t)$ shown below. The relevant part of the function is defined over the domain $-1 < t < 1$. Written as a mathematical expression it is

$$y(t) = \begin{cases} \sqrt{\frac{1}{4} - t^2} & \text{if } -1/2 < t < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

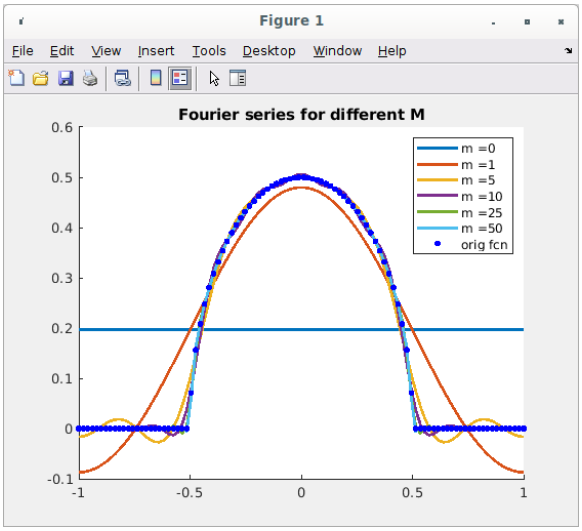
The peak of the circular bump occurs at $t = 0$. Assume the function $y(t)$ is zero for abs(y) > 1/2.



Please do the following:

- Using pencil/paper, derive the coefficients of its Fourier series expansion. (You can exploit the fact that the function is symmetric about the origin to deduce the basis functions for the expansion.) It is important to get the constants and limits of integration exactly right in order to get sane results. Working out the integral by hand is error-prone. Therefore, to perform the integrals I recommend you use the Wolfram Alpha integral calculator, available online at http://www.wolframalpha.com/calculators/integral-calculator/.

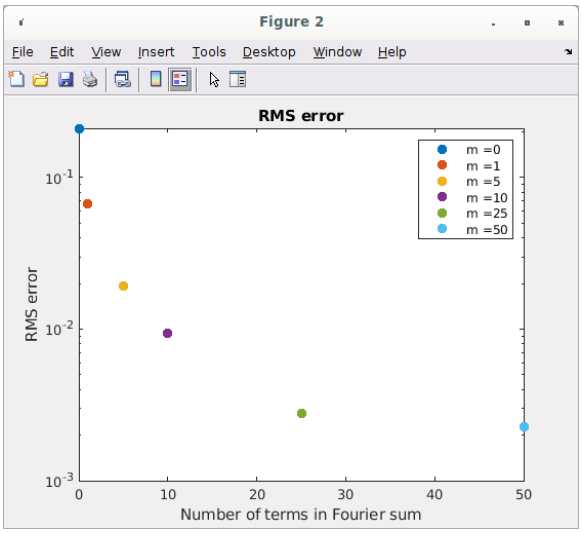
- Take M to be the highest order term in your series expansion. Write a program which loops over increasing M, computes the partial sums of the Fourier series, and makes a plot similar to that below showing how the series converges to the circular bump with increasing M. (My result is shown below.)



- Next, make a plot of the RMS error between your Fourier series expansion and the original function y(t) as a function of highest order term summed M. The RMS error is defined as

$$err = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\left(y_{true}(x_i) - y_{app}(x_i)\right)^2}$$

(For more information, please consult the note "On Computing RMS Error" available under "General information and handouts" on Canvas.) My RMS error plot is shown below.



Hand in your derivation of the Fourier coefficients, as well as your code which makes the two plots.

Problem 3

In class we discussed filtering a noisy time series by taking the Fourier transform of the input signal, multiplying it by a filter function in the frequency domain, then transforming it back to the time domain. Assuming you have a good filter function, this method's advantage is the O(N log N) complexity – it is quite fast.

A popular frequency-domain filter function is the Gaussian, whose frequency response may be written as

$$W(f)=e^{-\frac{1}{2}\frac{f^2}{f_c^2}}$$

where f is frequency (in Hz) and f_c is the cut-off frequency. For very low frequencies, $w(f)\approx 1$ whereas for high frequencies $w(f\to\infty)\to 0$. Although the filtering action is not "hard-wall", meaning the cut-off is not abrupt, the parameter f_c is a useful way to characterize the frequency response of this filter, so we shall treat it as the cut-off frequency of this filter.

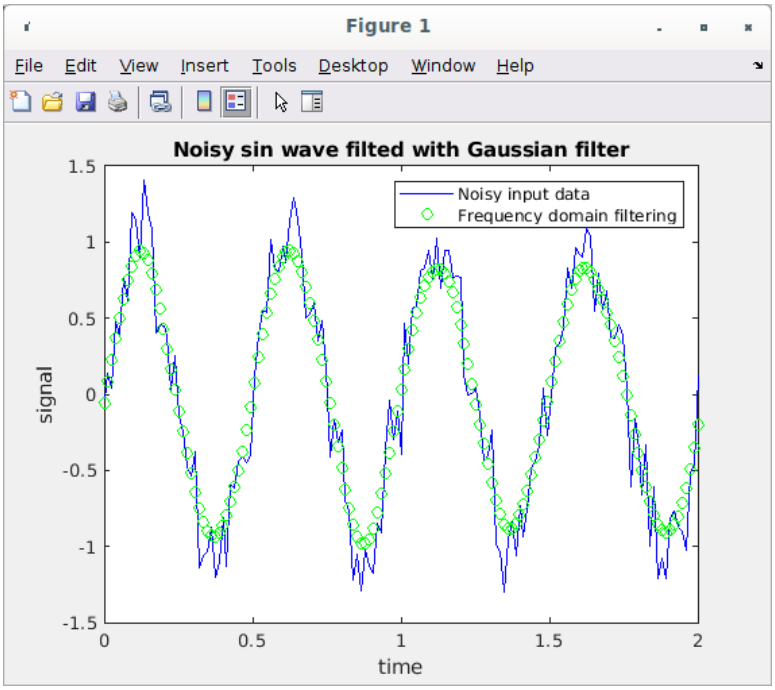
This frequency-domain filter is related to applying a Gaussian smoothing operator to the signal in time. I placed a short paper discussing the relationship between Gaussian filtering in time and in frequency on Canvas if you want to see how they are related.

Please do the following:

- Please write a program with calling signature "gauss_filter_freq(t, y, B)" which implements a frequency-domain filter of noisy input data. The inputs are the time axis t, the input signal y, and the cut-off frequency $B=f_c$. Implement the filter by Fourier transforming the input data y (using Matlab's fft()), multiplying by a Gaussian filter function $W(f)$, then inverse Fourier transforming the result. Make sure you use fftshift and ifftshift to place the transformed data into the correct form for multiplication with the filter function $W(f)$.

  Hint: After doing the ifft to take your result back to the time domain, use the Matlab function real() to extract the real part of the signal prior to plotting. You can discard the imaginary part, if the imaginary part is non-zero then it is only small noise produced by round-off error.

- Test your function by filtering a noisy sine wave. My test is shown at right. My sine wave frequency is 2Hz and my cut-off frequency is 4Hz. My sine wave has amplitude 1, and the noise is white Gaussian noise (randn()) of magnitude 0.2. I suggest you also verify your filter by filtering a clean sine wave having the same frequency and amplitude. How much do you expect to see the input sine wave diminished (attenuated) by the filter?



Feel free to use the filter programs I have placed on Canvas as a starting point for your programs.