Numerical Analysis 1 – Class 2

Friday, January 20th, 2023

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).
- "An FFT 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 12 second excerpt of the children's tune "Twinkle twinkle little star", which I played on a piano. It is stored on Canvas as the file "twinkle.wav". In Matlab, you can read in the sound file and listen to it using the following commands:

```
[y,Fs] = audioread('twinkle.wav')
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 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).

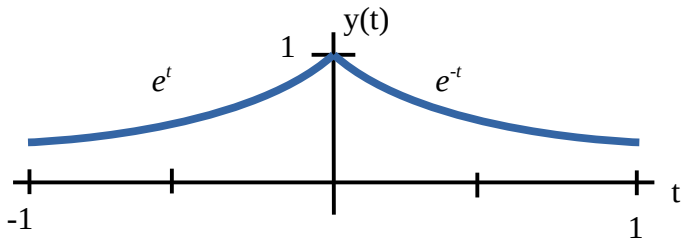
It's hard to think of a test for this program, but you can check your result using Matlab's "spectrogram" function to verify that the spectrum you compute matches that which Matlab computes.

Problem 2

Consider the "circus tent 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

$$y(t) = \begin{array}{ll} e^{t} & \text{if } t<0 \\ e^{-t} & \text{if } t>0 \end{array}$$

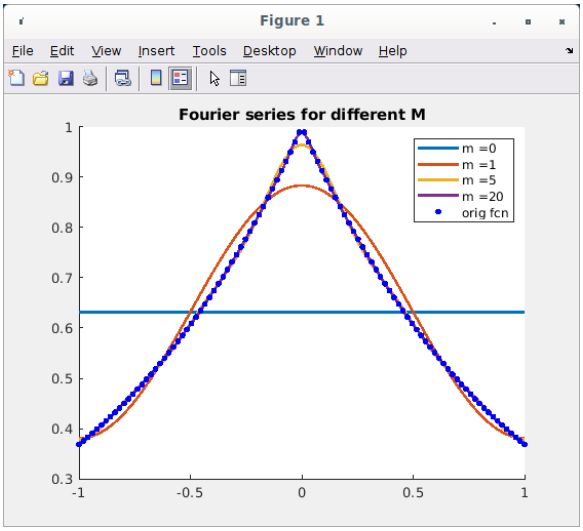
The peak of the tent occurs at $t = 0$.



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/.

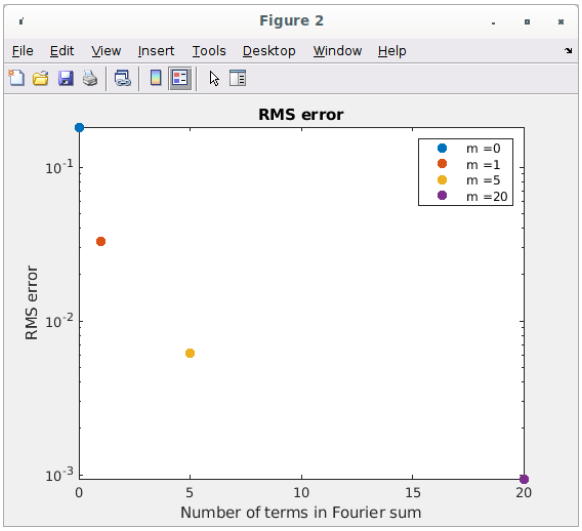
- Take M to be the highest order term in your series expansion. Write a program which loops / usr/local/include/gsl/gsl_math.hover increasing M, computes the partial sums of the Fourier series, and makes a plot similar to that below showing how the series converges to the circus tent with increasing M. (My result is shown below.) Please take a minute to marvel at the fact that you can approximate this circus tent shape – including the sharp peak – using a sum of simple trig functions.



- 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.



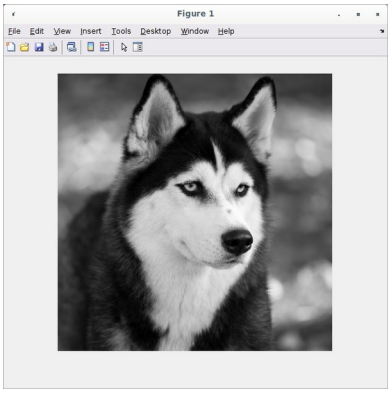
Hand in your derivation of the Fourier coefficients, as well as your code which makes the two plots.

Problem 3

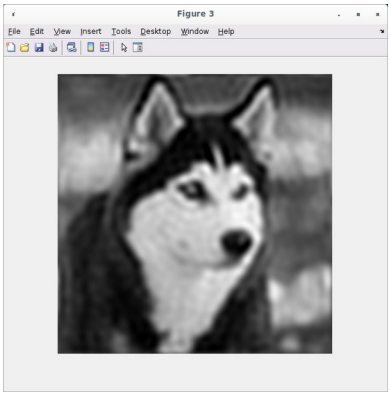
The Fourier transform is easily generalized to operate in multiple dimensions. In 2 dimensions, the transform is frequently used in image processing. Matlab implements the 2 dimensional Fourier transform using the functions fft2 and ifft2.

Similar to the 1D filtering application presented in class, once can apply the FFT to filtering of images

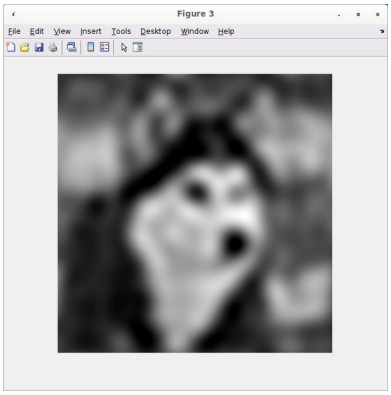
in 2 dimensions. For example, a low-pass filter applied to an image will tend to smooth out any detail in the original image, preserving only the areas where the image doesn't change (i.e. keeping the “low frequency information”). The effect is to blur the image. Refer to the figures below to see the effect.



Original image



After low-pass filter, C = 25



After low-pass filter, C = 10

Your assignment: On Canvas you will find a file called “HuskyBW.png”. Write a program which reads in the image, computes its FFT, cuts off the high frequency components, then computes the inverse FFT and displays the resulting, filtered image. Please make plots of the image before and after filtering. Don't forget that you need to use fftshift() to place the image in standard format prior to zeroing out the high frequency components. Your result should look like the above images, where I zeroed out components with index $[x, y]$ s.t. $x^2+y^2>C^2$, with $C = 25, 10$.

For additional information about this image filter, I suggest looking at the following web page: http://paulbourke.net/miscellaneous/imagefilter/. (His example 3 shows the result of using a high-pass filter on an image. This is the opposite of the problem here, which is a low-pass filter, but his discussion is very clear and easy to understand.)