Optimization-Based Data Analysis

Homework 3

Your submission should consist of 2 files: a PDF containing your solutions, and a .zip file containing your source code.

1. (Properties of Fourier Coefficients)

   (a) Suppose $f : [-1/2, 1/2] \to \mathbb{C}$ is real-valued, even (i.e., that $f(t) = f(-t)$ for all $t \in [-1/2, 1/2]$), and $f \in L_2[-1/2, 1/2]$.

      i. Prove that the Fourier coefficients are all real.

      ii. Show that the Fourier series of f can be written as
      $$S\{f\}(t) = \sum_{k=0}^{\infty} a_k \cos(2\pi k t),$$
      for some $a_0, \ldots \in \mathbb{R}$.

   (b) Suppose $f : [-1/2, 1/2] \to \mathbb{R}$ is defined by $f(t) = \cos(2\pi(t + \varphi))$ for some fixed $\varphi \in \mathbb{R}$. What are the Fourier coefficients of f?

   (c) Define the Dirichlet kernel $D_n(t)$ by
   $$D_n(t) := \sum_{k=-n}^{n} e^{2\pi i k t}.$$
   Prove that
   $$\lim_{n\to\infty} \int_{-1/2}^{1/2} D_n(t) f(t)\, dt = \int_{-1/2}^{1/2} f(t)\delta(t)\, dt,$$
   where δ is the Dirac (generalized) function centered at 0, and $f : [-1/2, 1/2] \to \mathbb{C}$ is continuously differentiable and satisfies $f(-1/2) = f(1/2)$. [Hint: You may use, without proof, the result from the notes that the Fourier series of f converges to f for every $t \in [-1/2, 1/2]$.]

2. (Sampling Theorem and Aliasing) Suppose $f : \mathbb{R} \to \mathbb{C}$ takes the form
   $$f(t) = \sum_{k=-k_c}^{k_c} a_k e^{2\pi i k t},$$
   for some finite $k_c > 0$ with $a_k \in \mathbb{C}$.

   In the timedata folder of hw3.zip you will find data.py. The load_data function will give you 3 numpy arrays. Each has the form
   $$[f(0/N), f(1/N), \ldots, f((N-1)/N)]$$

where $N = 2049, 4097, 8193$ for the three arrays, respectively. Each of the arrays are sampled from the same function f.

(a) Give plots of the magnitudes of the discrete Fourier coefficients computed from each of the arrays (3 plots in total). Use the standard DFT (not orthonormalized). Make sure to order your plot so that the 0 frequency is in the center, with negatives to the left and positives to the right. What do you notice about the magnitudes of the plots? [Hint: Use fftfreq in numpy with $d = 1.0/N$.]

(b) Assuming $k_c \leq 4096$ give the three values of a_k with the largest magnitudes, along with their corresponding frequencies (i.e., k-values). [Hint: The largest value isn't 307237.5.]

(c) Let $F^{(2049)}$ and $F^{(8193)}$ denote the discrete Fourier coefficients computed from the small and large arrays, respectively. Suppose you have only computed $F^{(8193)}$ and assume $k_c \leq 4096$. Show how you could use these Fourier coefficients to infer the entries of $F^{(2049)}$ without computing a DFT or an inverse DFT. [Hint: This is a bit tricky, but think about aliasing.]

(d) True or False: Since none of the plots computed above are non-zero for any frequency k with $|k| > 2048$ it follows that $k_c \leq 2048$. No justification needed.

(e) True or False: For any integer k with $|k| \leq k_c$, and any integer $M \geq 2k_c + 1$ we must have

$$\int_0^1 f(t)e^{-2\pi ikt}\,dt = \frac{1}{M}\sum_{j=0}^{M-1} f(j/M)e^{-2\pi ijk/M}.$$

Either prove the statement is true or give a counterexample. [Hint: If you use results taken from the notes, this is very short.]

3. (Justification of the FFT) Define the matrix $W^{[m]} \in \mathbb{C}^m \times \mathbb{C}^m$ by $W^{[m]}_{j,k} = e^{-2\pi ijk/m}$ where $0 \leq j, k < m$ (we index from 0 since it is convenient for Fourier matrices). Fixing $m = 2^n$ for some $n \geq 1$, answer the following questions. Below $a : b$ denotes the inclusive range $a, a+1, \ldots, b$ in our indexing.

(a) For any $k \geq 0$ with $2k < m$ prove a formula showing how to compute $W^{[m]}_{:,2k+1}$ assuming you are given $W^{[m]}_{:,2k}$.

(b) For any $k \geq 0$ with $2k < m$ prove that $W^{[m]}_{0:m/2-1,2k} = W^{[m]}_{m/2:,2k}$.

(c) Prove that $W^{[m]}_{0:m/2-1,2k} = W^{[m/2]}_{:,k}$.

4. (Discrete Convolutions) Let $\vec{x}, \vec{y} \in \mathbb{C}^n$, and suppose we use 0-based indexing (i.e., $\vec{x}[0], \ldots, \vec{x}[n-1]$) to denote the coordinates. Define the circular convolution of $\vec{x}, \vec{y}$ by

$$(\vec{x} * \vec{y})[k] = \sum_{j=0}^{k} \vec{x}[j]\vec{y}[k-j] + \sum_{j=k+1}^{n-1} \vec{x}[j]\vec{y}[k-j+n],$$

for $k = 0, \ldots, n-1$. This is the same as $\sum_{j=0}^{n-1} \vec{x}[j]\vec{y}[k-j]$ where we treat negative indices as wrapping around (modulo n).

Define the (linear) convolution between $\vec{x}, \vec{y}$ by

$$(\vec{x} *_L \vec{y})[k] = \sum_{j=0}^{k} \vec{x}[j]\vec{y}[k-j],$$

for $k = 0, \ldots, n-1$. This can also be written as

$$(\vec{x} *_L \vec{y})[k] = \sum_{j=-\infty}^{\infty} \vec{x}[j]\vec{y}[k-j],$$

where we think of $\vec{x}[j] = 0$ and $\vec{y}[j] = 0$ when $j < 0$ or $j > n-1$.

(a) Recall from the notes that $\vec{x}_{[m]}[j] = \vec{x}[j-m]$ is the circular time shift of $\vec{x}$ by m. Prove that

$$(\vec{x} * \vec{y})_{[m]}[j] = (\vec{x}_{[m]} * \vec{y})[j] = (\vec{x} * \vec{y}_{[m]})[j],$$

for $k = 0, \ldots, n-1$. [Hint: DFT.]

(b) Given $\vec{x}, \vec{y} \in \mathbb{C}^n$ show how to compute $\vec{w} = \vec{x} *_L \vec{y}$ in $O(n \lg n)$ time (here the convolution is not circular). Justify that your method is correct.

(c) For x, y both 2-dimensional define the 2-dimensional (linear) convolution by

$$(x * y)[a, b] = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[j, k]y[a-j, b-k],$$

where we consider $x[j, k]$ and $y[a-j, b-k]$ to be zero whenever the indices are outside of the ranges in which x, y are defined. Define x by

$$x = \begin{array}{cccc} \text{-1} & 0 & +1 & \\ -1/8 & -1/8 & -1/8 & \text{-1} \\ -1/8 & 1 & -1/8 & 0 \\ -1/8 & -1/8 & -1/8 & +1 \end{array}$$

where $j, k \in \{-1, 0, +1\}$. The 2-dimensional y will be given as input to the edge_detect function in convolve.py found in the convolution folder of hw3.zip. Implement the edge_detect function by computing the 2-dimensional (linear) convolution of x and y as described above. Then threshold the result as described in the comments. [Hint: Do not try to use the FFT. Just implement the formula directly using loops or use scipy.ndimage.convolve.]

(d) Give some explanation as to why the kernel x used in the previous part can assist in edge detection.

5. (Wiener Deconvolution) In this problem we will use the files in the wiener folder of hw3.zip. You will implement the functions given in wiener_filter .py.

   (a) Implement the convolve function that takes an image and a filter and produces the 2-dimensional (circular) convolution. Both will be 2-dimensional numpy arrays of size 64×64, and your result should have the same dimension. For efficiency (and ease of coding) you should use the 2-dimensional FFT and inverse FFT methods in numpy. [Hint: The implementation is nearly identical to the 1d-version, but instead use fft2 and ifft2.]

   (b) Implement the deconvolve function that takes a noiseless convolved image and the filter used to produce the convolution and returns the deconvolved image. [Hint: Check if a division by zero will nearly occur, and replace with a small non-zero value.]

   (c) Next we add noise to the problem. More precisely, we look at data of the form $A*X+Z$ where A is the filter, X is the image, and Z is additive iid Gaussian noise. You will notice that your deconvolve method works fine on noiseless convolved data, but gives terrible results on noisy data.

   To address this shortcoming, we will employ the Wiener deconvolution algorithm. Implement the wiener_deconvolve function. It takes the image, the filter, the variance of the noise, and a collection of many other images you will use to learn the per-frequency variances. [Hint: Carefully follow the Wiener deconvolution formulas from the notes. Also, be mindful of what the variance of the 2-dimensional FFT of the noise is.]

   (d) Include the plots generated by the program in your submission document.

6. (Spectral Super-Resolution) Fix a finite set K with known size s given by

$$T = \{t_1, \ldots, t_s\} \subseteq [-1/2, 1/2),$$

and fix $c_1, \ldots, c_s \in \mathbb{C} \setminus \{0\}$. Define $f : \mathbb{R} \to \mathbb{C}$ by

$$f(x) = \sum_{i=1}^{s} c_i e^{2\pi i t_i x}.$$

Our goal is to estimate the unknown t_i and c_i values given a finite number of samples $n \geq 2s$ given by

$$f(0), \ldots, f(n-1) \in \mathbb{C}^n.$$

In this problem we will use the file music.py in the music folder of hw3.zip.

   (a) Define the periodogram $P(t)$ by

$$P(t) := \left| \frac{1}{n} \sum_{j=0}^{n-1} f(j) e^{-2\pi i j t} \right|^2,$$

for $t \in [-1/2, 1/2)$. Suppose T has only a single element so that

$$f(x) = c_1 e^{2\pi i t_1 x}.$$

Prove that $P(t) \leq |c_1|^2$ and that $P(t) = |c_1|^2$ if and only if $t = t_1$. [Hint: Show that $|1 + e^{2\pi i \omega}| < 2$ when $\omega \notin \mathbb{Z}$.]

(b) Suppose you have determined the elements of T. Describe a method for finding the coefficients $c_1, \ldots, c_s$.

(c) Implement the periodogram function in music.py that takes as input s and the $f(k)$-values and plots the periodogram defined in the previous part for $t \in [-1/2, 1/2)$. On the same plot the function should superimpose a stem plot of the points $(t_i, |c_i|^2)$. Include the plots in your document submission.

(d) Implement the music function in music.py that takes as input s and the $f(k)$-values and plots the (log) pseudospectrum as defined in the notes. Include the plots in your submission.