CS168, Spring 2017

Mini-Project #8

Due by 11:59 PM on Tuesday, May 30th.

Instructions

- You can work individually or with one partner. If you work in a pair, both partners will receive the same grade.
- If you've written code to solve a certain part of a problem, or if the part explicitly asks you to implement an algorithm, you must also include the code in your pdf submission. See the problem parts below for instructions on where in your writeup to put the code.
- Make sure plots you submit are easy to read at a normal zoom level.
- Detailed submission instruction can be found on the course website (`http://cs168.stanford.edu`) under the "Coursework - Assignment" section. If you work in pairs, only one member should submit all of the relevant files.
- a) Use 12pt or higher font for your writeup. b) Code marked as "Deliverable" gets pasted into the relevant section, rather than into the appendix (though feel free to put it in both). Keep variable names consistent with those used in the problem statement, and with general conventions. No need to include import statements and other scaffolding, if it is clear from context. Also, please use the `verbatim` environment to paste code in LaTeX from now on, rather than the listings package:

```
def example():
print "Your code should be formatted like this."
```

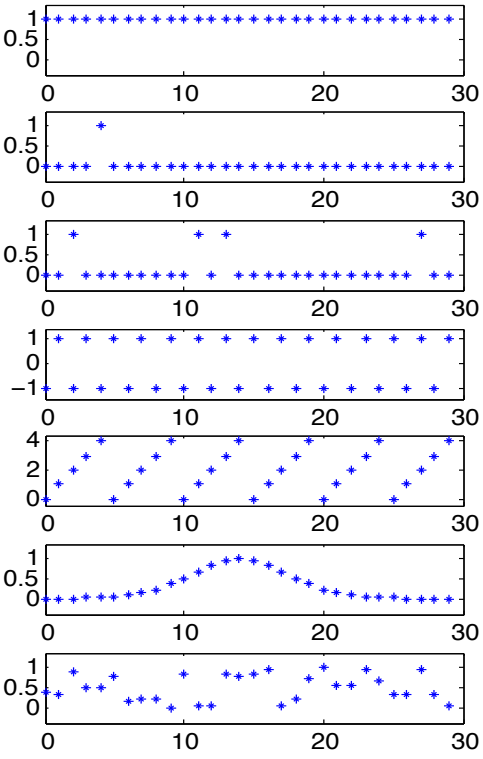
- **Reminder:** No late assignments will be accepted, but we will drop your lowest mini-project grade when calculating your final grade.

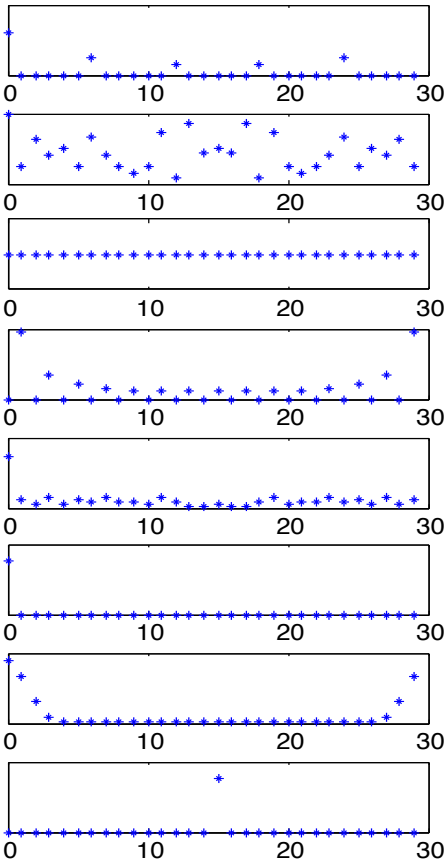
1. **Fourier Transforms.** Recall that the discrete Fourier transform of a length N vector/signal $\mathbf{f}$ is a length N complex-valued vector/signal $\mathbf{F}$ whose mth coordinate is defined to be

$$\mathbf{F}[m] = \sum_{j=0}^{N-1} \mathbf{f}[j] e^{-2\pi i m j/N}, \ m = 0, 1, \ldots, N-1$$

We denote the Fourier transform of a vector $\mathbf{f}$ by $\mathcal{F}\mathbf{f}$, and use $\mathcal{F}^{-1}$ to denote the inverse transformation. Hence for $\mathbf{F}$ as defined above, we have $\mathbf{F} = \mathcal{F}\mathbf{f}$ and $\mathbf{f} = \mathcal{F}^{-1}\mathbf{F}$.

(14 points) For each vector/signal depicted in the left column, 1) draw a line connecting it with the plot on the right of the magnitude of its (discrete) Fourier transform, and 2) write one sentence justifying your answer in a high-level intuitive way (e.g. "The Fourier transform of random noise is usual BLAH, and the plot at left looks like random noise and the plot at right looks like BLAH."). So that you need to look at the shapes of the plots, the scale on the y-axis of the plots in the right column have been removed.
Deliverables: For each plot on the left: an index of the corresponding plot on the right, and one sentence explanation.


1
0.5
0
0 10 20 30
1
0.5
0
0 10 20 30
1
0.5
0
0 10 20 30
1
0
−1
0 10 20 30
4
2
0
0 10 20 30
1
0.5
0
0 10 20 30
1
0.5
0
0 10 20 30


0 10 20 30
0 10 20 30
0 10 20 30
0 10 20 30
0 10 20 30
0 10 20 30
0 10 20 30
0 10 20 30

2. **Convolution.**

A concept that is closely related to the discrete Fourier transform is *(linear) convolution*. Given a N-tuple $\mathbf{f}$ and a M-tuple $\mathbf{g}$, define their convolution $\mathbf{f} * \mathbf{g}$ to be a $(M + N - 1)$ tuple: [1]

$$(\mathbf{f} * \mathbf{g})[k] = \sum_{j=0}^{N-1} \mathbf{f}[j]\mathbf{g}[k-j]$$

(a) (3 points) Given a six-sided die, let p be a 6-tuple where $p[i]$ denotes probability that $i+1$ appears. Let $q = \underbrace{p * p * \cdots * p}_{100 \text{ times}}$. What event occurs with probability $q[150]$? [Assume zero-index vectors.]

(b) (3 points) Given an N-tuple $\mathbf{f}$, define $\mathbf{f}^+$ to be the $(2N-1)$-tuple obtained by padding $\mathbf{f}$ with zeros:

$$\mathbf{f}^+[i] = \begin{cases} \mathbf{f}[i] & \text{if } 0 \le i \le N-1 \\ 0 & \text{if } N \le i \le 2N-2 \end{cases}$$

Verify that convolution is multiplication under the Fourier transform: for any two N-tuples $\mathbf{f}$ and $\mathbf{g}$, show that

$$\mathcal{F}(\mathbf{f} * \mathbf{g}) = \mathcal{F}\mathbf{f}^+ \cdot \mathcal{F}\mathbf{g}^+$$

where $\cdot$ denotes element-wise multiplication. [You should not just quote the lecture notes—you must write out the calculation explicitly.] Suggest an implementation of convolution using the Fast Fourier Transform and its inverse transform. What is the analogous conclusion you can make when the two tuples have different lengths?

(c) (4 points) Using the Fast Fourier Transform (and its inverse transform), write a method $multiply(x, y)$ that takes in two arrays of digits, each representing an integer (lower indices represent lower digits), and return an array that represents the product of the two integers. For example, the output of $[0, 1, 2, 3]$ and $[4, 5, 6]$ should be $[0, 4, 3, 9, 9, 0, 2]$, representing the fact that $3,210 \times 654 = 2,099,340$. Using your code, what is the product of the following two numbers:

$$x = 12345678901234567890, \text{ and } y = 98765432109876543210987654321 0?$$

[For this part, please embed your *actual* code in the solution, instead of including it in the appendix. Do not directly use convolution functions in your code.]

(d) (4 points) Compare the asymptotic run-time of this multiplication algorithm to that of the naive grade-school integer multiplication algorithm. Feel free to refer to the runtime of the Fast Fourier Transform algorithm that we discussed in class.

(e) (Bonus: 2 points) For your implementation for part (c), roughly how large can the inputs be for the method to return accurate results? Justify your response via experiments and a discussion of the Fourier transform and the known limitations of your program environment (i.e. number of significant bits stored, etc.).

Deliverables: answer for (a); calculation, suggestion, and analogous conclusion for (b); code and answers for (c); discussion for (d) and (e).

3. **The Sound of Fourier Transform.** In this part, we explore how Fourier transform can help us understand human sounds. You'll need to record yourself, and transform the audio signal into an array which you can analyze. You'll also need to know the sample rate of the recordings. In Matlab, you can do this by:

```
recObj = audiorecorder; % recorder configuration
recordblocking(recObj, 3); % records 3 seconds of audio
data = getaudiodata(recObj); % the vector representing the audio
sampleRate = 8000; % default sample rate
```

[1] We define $\mathbf{g}[k-j] = 0$ when $k - j < 0$.

If you're working with Python, first save your recording as a `.wav` file, and then load it into Python: [2]

```
import scipy.io.wavfile as wavfile
import numpy as np

with open("audio.wav", "r") as f:
  sampleRate, data = wavfile.read(f)
  if len(data.shape) == 2 and data.shape[1] == 2:
      data = data[:,1] itshape itshape'''itshaperemoveitshape itshapetheitshape itshapesecondits
```

In addition to recording yourself, you can also listen to the sound represented by an array. The Matlab command `soundsc(data)` allows you to do that; if you're using Python, you'll need to save the signal array as a `.wav` file and then listen to the file:

```
data = data * 1.0 / np.max(np.abs(data))
with open("output.wav", "w") as f:
  wavfile.write(f, sampleRate, data)
```

Exercises:

(a) (1 point) Record yourself making the following sounds for at least three seconds each:[3]

- The Dentist Check: open your mouth wide and make an "Ahhhhhh" sound.
- The E: make the sound of the letter "Eeeeeee."
- The Skeptic: close you mouth and make a "Mmmmmm" sound.

For each sound, plot the signal against time, and zoom in to the point where you can see the waveforms.

(b) (3 points) Take the Fourier Transform of each signal. Plot the **magnitude** of the elements in the Fourier Transform against the tuple index. You should get three (almost) symmetric plots. What is the dominant frequency (the frequency of the lowest-frequency spike—note that this might not be the frequency with the largest magnitude) of each sound? [Hint: you'll need to use the sample rate to calculate the dominant frequencies.]

(c) (5 points) Making references to the Fourier transforms of the signals, what are the main differences between these sounds? Why do you think the transforms of the "Ahhhh" sound and the "Mmmmm" sound differ in this way? (For example, from a physics standpoint, how do you think the physical differences (mouth open, mouth closed) contribute to the differences in the frequencies present, number of harmonics, etc?)

(d) (5 points) Record yourself saying something. Now we will add an echo-effect: what filter (i.e. vector) $\mathbf{f_1}$ has the property that $\mathbf{f} * \mathbf{f_1}$ consists of the original recording, over-layed with an echo of delay 0.2 seconds at half the volume? Make sure you listen to the signal—it should sound like the original recording but with a boomy/echo effect. Plot the original signal $\mathbf{f}$ and $\mathbf{f} * \mathbf{f_1}$, zoomed in to a resolution where you can see the wave-form.

(e) (ungraded bonus) Experiment with other sorts of filters, and manipulations of the Fourier transform of your signal. How would you lower the pitch of your recording an octave? How would you dampen the high-frequency portion? How would you "brighten" the sound by slightly amplifying the high-frequency?

Deliverables: Plots for (a); plots and calculation for (b); discussion for (c), answer and plots for (d).

[2]There seems to be a bug in the way the **scipy.io.wavfile** package processes some PPC **.wav** files. If **read()** throws an exception in your code, try to save the file in a different architecture, or use the Python default **wave** package.

[3]With your best effort, try to make it consistent in volume and pitch, but don't worry about it too much.