ECE210 Honors Final Assignment

(Choose 3 out of 5)

Congratulations on surviving all the previous assignments! Now you are only one step away from getting the honors credit :). Every student must complete and demo the final assignment to receive the honors credit. You can use either MATLAB or Python as you like, and you only need to complete any three (out of the total five) problems for demo. Please make sure you understand what you write in your code, as I will also read through your code and ask questions during the demo. Student who cannot answer basic questions on their own code will be suspected to have plagiarized other students' code. This may result in the student not being able to receive the honors credit or even being reported to the department. Again, ANY FORM OF PLAGIARISM OR CHEATING WILL NOT BE TOLERATED.

As always, discussion and collaboration between students are encouraged; but every student must write up his own code and understand how the code works. **Deadline for the final assignment will be announced on Piazza.** Please plan your time accordingly and start early. In the last session, I will review material from the previous session and use the rest of time as office hours and demo time for the Python Session 3. Good Luck!

Question 1

For this question we will cover something called a spectrogram. A spectrogram is visual tool used to illustrate the Short Time Fourier Transform which is, as advertised, a Fourier Transform over a small period of time. We can use it to see how the frequency spectrum changes over time for a signal. For this question it is recommended you use the built in spectrogram function from scipy.signal in python.

1. Create a sinesweep from 0 to 10kHz, we recommend using the chirp function

   In python:

   ```
   import numpy as np
   import scipy.signal as sig
   import matplotlib.pyplot as plt
   t=np.linspace(0,1,num=20000)
   y=sig.chirp(t,0,1,10000)
   ```

2. Use the spectrogram function and plot. Please label axes and put a title.

   In python use:

   ```
   f, t, Sxx = sig.spectrogram(y, len(y))
   plt.pcolormesh(t, f, Sxx)
   ```

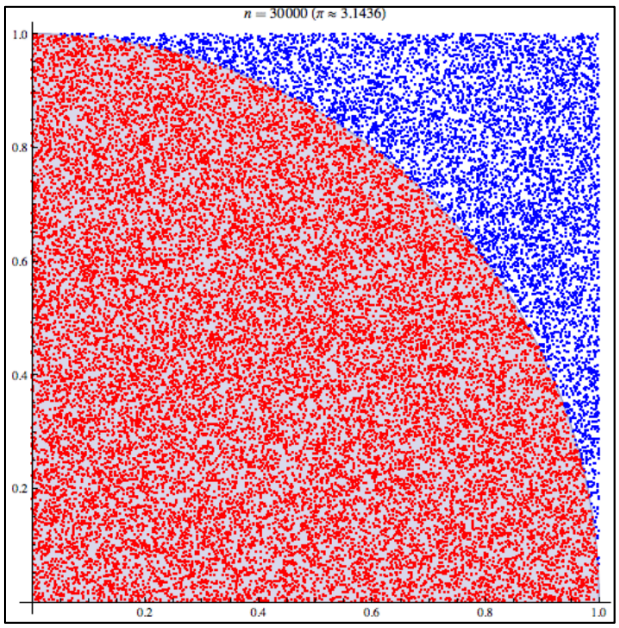
3. Now we will do something called downsampling. Take your array and simply discard every other sample. Once you have done this plot the spectrogram of your new downsampled signal. What is different? Do you know why?
4. We wish to avoid the above affect so take the original signal and pass it through a low pass filter which keeps only frequencies below and including 5kHz to keep this from happening. Once you have done this, downsample the signal and plot the spectrogram. You should see a difference from the previous step. Think about why the three spectrograms are different, i.e. why would this happen?

   Reference: ECE 311 Spring 2017

Question 2

This problem is about simulating the π value using the famous Monte-Carlo method.

1. Suppose we have a unit square, and we inscribe a quarter unit circle inside the unit square.
2. Using 2 randomly generated numbers to find x, y coordinates in the unit square
   a. Python: np.random.rand()
   http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand.html
   b. Matlab: rand()
3. Determine whether the coordinates lie within the unit circle
4. Repeat Steps 2 & 3 for large numbers
5. Compute the approximation of PI for N=100, 200, 500, 1000, 2000, 5000, 10000, 100000. The approximate value of PI is given by 4 * (m/n) where m is the number of points that fall within the circle and n is the total number of points considered.
6. And plot the Error of Approximation against N on a Log-Log Plot
   a. Error = abs(Actual value - Approximate Value)
   b. Use pi (Matlab) or np.pi (Python) to get "Actual Value''
7. Compare against an iterative method of finding pi (example done in lab) and plot the error of approximation on the same plot generated in (6)



Taken from http://en.wikipedia.org/wiki/Approximations_of_%CF%80#/media/File:Pi_30K.gif

Question 3

Amplitude Modulation and using FFT

1. Read http://en.wikipedia.org/wiki/Amplitude_modulation
2. Set a sampling frequency: Fs >= 32kHz
3. Generate a Carrier Wave with frequency of 8 kHz and amplitude of 1 and sufficient time length (t) to cover 10 periods
4. Use 1024 point FFT (Fast Fourier Transform, N=1024) to observe the spectrum of the carrier wave. Hint:
    - Matlab Code:
      ```
      spectrum=fftshift(abs(fft(,N)))
      f_k=linspace(-fs/2,fs/2,length(spectrum))
      plot(f_k,spectrum)
      xlabel('Frequency (Hz)')
      ylabel('X(\Omega)')
      ```
    - Python Code:
      ```
      spectrum=np.fftshift(np.abs(np.fft(,N)))
      f_k=linspace(-fs/2, fs/2,length(spectrum)
      plt.plot(f_k,spectrum)
      ```
    - Sanity Check:
      What frequency does the peaks occur at?
5. Generate a Modulating Signal waveform (another sine wave) with frequency of 1 kHz of the same length (same time and sampling frequency) and amplitude as the carrier wave
6. Perform Amplitude Modulation with Modulation index = 50% and plot both carrier wave and modulated waveform on the same figure on 2 different subplots
7. Plot the FFT of the modulated signal and the carrier signal side by side. Does it match what you expected? Provide an explanation as to why the modulated signal looks the way it does.

Question 4

Digital Filtering using FFT and IFFT

1. Sampling Frequency=32Hz and time vector length of 1024
2. Generate the following sequences
   a. y1: Square wave with amplitude 10 and frequency 1 kHz
      i. Use the square function in Matlab
      ii. Use the scipy.signal.square function in python
   b. y2: cosine with amplitude 1 and frequency 1 kHz and +pi/2 phase
   c. y3: cosine with amplitude 1 and frequency 10 kHz and +pi/4 phase
   d. sum = y1+y2+y3;
   e. Plot all 3 signals and the sum in a single figure in 4 subplots, plot the first 100 points
3. Use 1024 point FFT (Fast Fourier Transform, N=1024) to observe the spectrum of all 4 signals in a single figure in 3 subplots
   a. Matlab Code:

```
spectrum=fftshift(abs(fft(,N)))
f_k=linspace(fs/2,fs/2,length(spectrum))
plot(f_k,spectrum)
xlabel('Frequency (Hz)')
ylabel('X(\Omega)')
```

   b. Python Code:

```
spectrum=np.fftshift(np.abs(np.fft(,N)))
f_k=linspace(fs/2,fs/2,length(spectrum))
plt.plot(f_k,spectrum)
```

4. Observe that Fourier Transform is a Linear Operation
5. Since we are in frequency domain, convolving a signal with a filter in time time domain is equivalent to multiplying in frequency domain. .
6. Generate an ideal low pass filter and ideal high pass filter with cutoff at fs/4, with same length as f_k;
   a. Determine index of f_k that correspond to fs/4;
   b. Generate the filter response
   c. Plot the frequency response of the filters against f_k;
7. Multiply both filters to the frequency response and plot the filtered response
8. Perform the IFFT on the filtered frequency responses
   a. Matlab Code:

```
spectrum_shifted=fftshift()
y_out=ifft(spectrum_shifted,N)
```

   b. Python Code:

```
spectrum_shifted=np.fftshift()
y_out=fft.ifft(spectrum_shifted,N)
```

9. Plot the time domain signals and the original signal on the same figure in 3 subplots, plot only the first 100 points

Question 5:

Signal Detection in MATLAB/Python

You are a member of a secret organization. Your ally from across the world needs to send you an encrypted message. However, with the superior decryption technology available nowadays, your ally fears that the information could be stolen. As a student from UIUC, you possess superior skills in signal processing and decide to decrypt his message by counting the number of times the signal 'sig' appears in 'x'.

You possess with yourself the signal 'sig' which serves as the key for decrypting the transmission. Your ally sends you the signal 'x' which you must interpret. You know that your ally is alarming you of the number of enemy battalions approaching. With most of your troops gone, your only hope of surviving is using MATLAB/Python to decrypt this message.

1. Load the signal in MATLAB or Python from the **'q1_signal.mat'** file.
    - In MATLAB you can load the signal using the following command:
      ```
      load q1_signal.mat
      ```
    - In Python you can load the signal using the following command:
      ```
      import scipy.io as scio
      import numpy as np
      mat_contents = scio.loadmat('q1_signal.mat')
      ```

      In Python, mat_contents is created as a dictionary. Extract 'x' and 'sig' using the basic dictionary concepts that you learned in class. Once you extract the signals use the following command to convert it to an array:

      ```
      x = mat_contents[___] #fill in the blank
      sig = mat_contents[___] #fill in the blank

      x = np.asarray(x).reshape(-1);
      //Do a similar thing for sig
      ```
2. Plot the magnitude and phase response of x.
3. Plot the magnitude and phase response of sig.
4. Here is the code to get the FFT and range of frequencies:
    - MATLAB
      ```
      % FFT of 'x'
      N = 2048;
      X = fftshift(fft(x,N));
      ```

```
% Create a range of frequencies from -pi to pi
w = fftshift((0:N-1)/N * 2 * pi);
w(1:N/2) = w(1:N/2) – 2*pi;
%plot the magnitude and phase response
```

- Python

```
import numpy.fft as fft
N = 2048
X = fft.fft(x,N)
X = fft.fftshift(X);

#Create a range of frequencies from -pi to pi
w = np.array(0, N) / N * 2 *np.pi
w = fft.fftshift(w)
w[0:N/2] = w[0:N/2] – 2 * np.pi

#plot the magnitude and phase response
```

5. Find the number of times ‘sig’ appears in the signal ‘x’ by performing cross correlation.
6. Only the positive spikes matter. Only count those spikes above a threshold of **3.4** (height of 3.4). Adjust the scale of your x-axis to get a nice plot. This is similar to the autocorrelation that was performed in Python Session 3. In MATLAB you can perform cross correlation using xcorr.
7. On a piece of paper explain the relationship between correlation and convolution in a few lines.

Congratulations! You can now save yourself and your remaining troops!

Reference: ECE 311 Final Spring 2017