CSE 3313 – Homework #4

Assigned:	Monday, February 18, 2019
Due:	Wednesday, February 27, 2019 at the end of class

Note the following about the homework:

1. You must show your work to receive credit.
2. For the hand-written problems, submit a hard-copy.

Assignment:

1. (20 points) An LTI system can be described by the difference equation

$$y[n] = 2x[n] - 3x[n-1] + 2x[n-2]$$

   (a) Find $y[n]$ over the range $0 \le n \le 10$ when the input is

$$x[n] = \begin{cases} 0 & \text{for } n < 0 \\ n+1 & \text{for } n = 0, 1, 2 \\ 5-n & \text{for } n = 3, 4 \\ 1 & \text{for } n \ge 5 \end{cases}$$

   (b) Plot $x[n]$ as a stem plot and the values of $y[n]$ as a stem plot that you just found. These should be separate plots.

   (c) Find the unit impulse response $h[n]$ for this system when the input is a unit impulse sequence, $\delta[n]$.

Applications Note the changes to the submission procedure.

2. (40 points) **Application Area: Design of Lowpass and Highpass FIR Filters**
   Purpose: Learn how to produce the filter coefficients needed to construction lowpass and highpass filters.

   A common signal processing task is to remove the components of a signal whose frequency is outside a particular range. A lowpass filter removes the components whose frequency is higher than a certain cut-off frequency while a highpass filter removes the components whose frequency is below a certain cut-off frequency.

   Note the following about your submission:

   - You will create two different filters, one lowpass and one highpass, and apply them to a digitized signal.
   - The process we are using is based on the tutorial at [Gre15].

- On the course website is a file, `data-filtering.csv`, that is the data you will use. The sampling rate for the data is $f_s = 2000$. Both filters will use this data, but you should not hard-code your logic to this particular data beyond knowing the filename and the sampling rate.
- Your source file will be named `firDesign.py` and will include both the lowpass and highpass implementations.

(a) Design a lowpass filter with the following specification:

- a cut-off frequency of $f_c = 50$ Hz
- the filter length $L = 21$ and $M =$ filter length - 1.
- $0 \leq n < L$

The filter weights $w[n]$ are found from

$$w[n] = \begin{cases} \frac{\sin[2\pi f_t(n-\frac{M}{2})]}{\pi(n-\frac{M}{2})} & n \neq \frac{M}{2} \\ 2f_t & n = \frac{M}{2} \end{cases}$$

where

- $f_t = f_c/f_s$ is the normalized cut-off frequency.

Then use Numpy's `convolve()` function to apply the filter.

(b) Demonstrate the application of the filter by reproducing the plot in Figure 1.

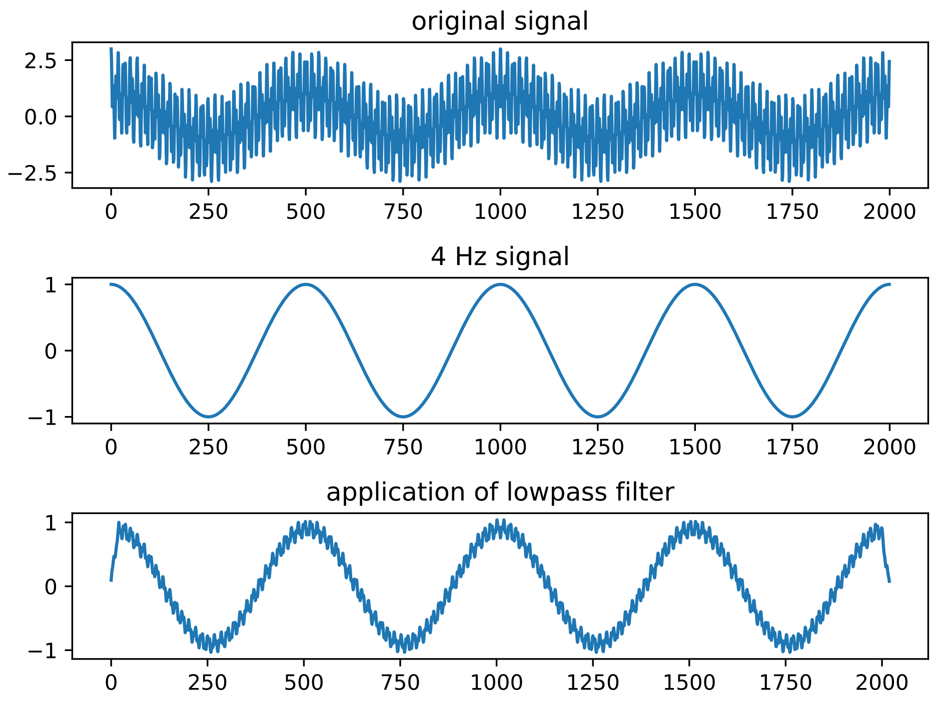

Figure 1: Lowpass filter

Note that the original signal and the 4 Hz signal that you generate will have 2000 values, all of which you should plot. However, the filtered signal will have more values (ask yourself why). Plot all of the values of the filtered signal.

(c) Design a highpass filter with the following specification:

- a cut-off frequency of $f_c = 280$ Hz
- the filter length is 21 and M = filter length - 1.

The filter weights $w[n]$ are found from

$$w[n] = \begin{cases} -\frac{\sin[2\pi f_t(n-\frac{M}{2})]}{\pi(n-\frac{M}{2})} & n \neq \frac{M}{2} \\ 1 - 2f_t & n = \frac{M}{2} \end{cases}$$

where

- $f_t = f_c/f_s$ is the normalized cut-off frequency.

Then use Numpy's `convolve()` function to apply the filter to the ORIGINAL signal.

(d) Demonstrate the application of the filter by reproducing the plot in Figure 2.

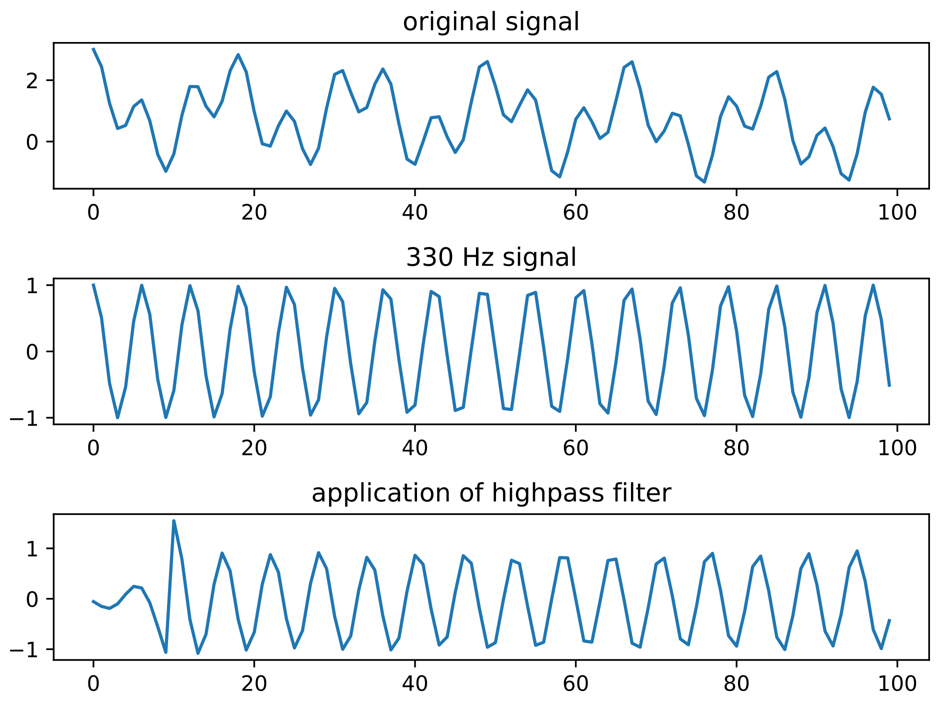

Figure 2: Highpass filter

Because of the high frequency components in the signal, it's hard to see what effect our filter had if we look at all 2000 values. Therefore, you should only plot the first 100 values of the original signal, the 330 Hz signal that you generate, and the filtered signal.

3. (40 points) **Application Area: Image Processing**
Purpose: Learn to apply simple filters to images.

A digitized image is a matrix of numbers. We can think of it as a 2D signal and apply some of the same signal processing techniques that we would apply to traditional 1D signals.

Note the following about your submission:

- Your source file will be named `imageProcessing.py`.
- The images you will use are in the compressed file, `images.zip`, which is on the course website. Unzip it wherever you do your development. The .tiff files came from `http://sipi.usc.edu/database/database.php?volume=misc`.
- Your program should read these images from the same directory as your .py file and use the names of the files as given.
- Do the work yourself as specified below; don't use Numpy, SciPy, OpenCV or other libraries to do all of the image processing. The purpose of this assignment is to see how things actually work.
- When submitting, only include your source code. Do not include the images.

For each of the tiff images (boat, clock, man, tank) do the following:

(a) Display the original image in its own figure window with a meaningful title. This means that all images should be on the screen at the same time, but don't use the subplot feature.

(b) Blur the image with a lowpass filter.
   i. The lowpass filter will be a 10-point moving average, that is, `h[n]` will consist of 10 values of 0.1 each.
   ii. Apply this filter to each row of the image using Numpy's `convolve()` function. When constructing the processed image, you might find the discussion at `http://akuederle.com/create-numpy-array-with-for-loop` helpful.
   iii. Display the processed image in its own figure window with a meaningful title. `matplotlib` can be used to display the image.

(c) Perform edge detection with a highpass filter.
   i. For this filter, `h[n] = {1, -1}`.
   ii. Apply this filter to each row of the ORIGINAL image using Numpy's `convolve()` function.
   iii. Display the processed image in its own figure window with a meaningful title.

For the image `darinGrayNoise.jpg` do the following:

(d) remove the noise
   i. Display the original image in its own figure window with a meaningful title. This image contains what is called "salt and pepper" noise.

ii. One way to remove noise is to average it out with a lowpass filter. Apply your 10-point running average filter to each row. Display the processed image in its own figure window with a meaningful title.

iii. Apply a median filter (which is a nonlinear filter) to the original noisy image using

```
outputImage = ndimage.median_filter(inputImage, 5)
```

where `inputImage` and `outputImage` are the names you choose. The library containing this function needs to be imported with something like

```
from scipy import ndimage
```

Display the processed image in its own figure window with a meaningful title.

Note that using this function this way actually applies the filter to the entire input image, producing the output image. Therefore, you do not need to write the code to try to apply it line by line as you did with the other filters.
More details about median filters can be found in [McA04].

General requirements about the Python problems:

a) As a comment in your source code, include your name.

b) The Python program should do the work. Don't perform the calculations and then hard-code the values in the code or look at the data and hard-code to this data unless instructed to do so.

c) The program should not prompt the user for values, read from files unless instructed to do so, or print things not specified to be printed in the requirements.

To submit the Python portion, do the following:

a) Create a directory using your net ID in lowercase characters plus the specific homework. For example, if your net ID is `abc1234` and the homework is `hw04`, then the directory should be named `abc1234-hw04`.

b) Place your .py files in this directory.

c) Zip the directory, not just the files within the directory. You must use the zip format and the name of the file (using the example above) will be `abc1234-hw04.zip`.

d) Upload the zip'd file to Blackboard.

References

[Gre15] Andrew Greensted. FIR Filters by Windowing. `http://www.labbookpages.co.uk/audio/firWindowing.html`, accessed March 15, 2015.

[McA04] Alasdair McAndrew. *Introduction to Digital Image Processing with MATLAB*. Thomson Learning, New York, 2004.