ECE 130A Computer Assignment #5

Due Date: March 18, 2021

Submit your homework as a zipped folder with the .m function files.

For this assignment, you will need create the function files with the right filenames and parameters/outputs.

Problem 1

Function Name: hw51

Inputs:

- *(double, 1 × M)* x - signal

Outputs:

- *(double, 1 × M)* X - DFT of the the given signal.

Function Description:
The Fourier Transform is a powerful analysis tool and is an integral part to many real-world algorithms. However, since computers can only deal with discrete data, the discrete version of the Fourier Transform - *Discrete Fourier Transform* (DFT) - is instead implemented. Given an M-length vector x, the DFT produces

$$X[k] = \sum_{m=0}^{M-1} x[m] e^{-\frac{2\pi j}{M} mk} \quad \text{for all} \;\; 0 \leq k \leq M-1.$$

Here each component of $X[k]$ describes the proportion of the sinusoid $e^{\omega_k n}$ associated with x for the frequency $\omega_k = \frac{2\pi jk}{M}$ with $0 \leq k \leq M-1$.

You would like to code your implementation of the DFT, based on the *Cooley-Tuckey FFT*, a popular algorithm. Surprisingly, this algorithm can be easily described recursively as

$$\text{FFT}(x)[k] \equiv X[k] = \begin{cases} X_{\text{even}}[k] + e^{-\frac{2\pi j}{M} k} X_{\text{odd}}[k] & \text{if } 0 \leq k < \frac{M}{2} \\ X_{\text{even}}[k - \frac{M}{2}] + e^{-\frac{2\pi j}{M} k} X_{\text{odd}}[k - \frac{M}{2}] & \text{if } \frac{M}{2} \leq k < M-1 \end{cases}$$

where $X_{\text{even}} \equiv \text{FFT}(x_{\text{even}})$ is the algorithm with input $x_{\text{even}} = \{x[0], x[2], \ldots, x[2n]\}$ with only the even indices of x. X_{odd} is similarly defined with $x_{\text{odd}} = \{x[1], x[3], \ldots, x[2n+1]\}$ instead. Notice that we can also calculate X_{even} through the above equation with $X_{\text{even even}}$ and $X_{\text{even odd}}$ and continue the process recursively.

Notes:

- for-loops and MATLAB DFT functions are not allowed for this problem.
- Recursion is needed to implement this algorithm.
- Round the resulting X to the nearest decimal place using `round(X, 1)`.
- What is the DFT(x) if the length of x is 1?
- You can assume that the length of x is always a power of 2.

Problem 2

Function Name: hw52

Inputs:

- *(double, 1 × 2^M)* x - input signal to the wavelet filter
- *(double)* thres - threshold for wavelet coefficients between 0 and 1

Outputs:

- *(double, 1 × 2^M)* y - output of the signal after the wavelet filter is applied

Function Description:
Signal compression and denoising is vitally important when transferring signals across communication channels. Wavelet filters are useful tools to achieve both of these goals. In this vein, you would like to implement a wavelet compression algorithm, using *Haar wavelets*. To do this, we first define the Haar wavelet vectors. The wavelet $\psi^{0,1}$ is defined as a 2^M-length vector with the ith component $\psi^{0,1}(i) = 1$ for all $1 \leq i \leq 2^M$. Similarly, for all $1 \leq j \leq M$ and $1 \leq k \leq 2^{j-1}$, we define the the ith component of the 2^M-length vector $\psi^{j,k}$ as

$$\psi^{j,k}(i) = \begin{cases} 1 & \text{if } (2k-2)2^{M-j}+1 \leq i \leq (2k-1)2^{M-j} \\ -1 & \text{if } (2k-1)2^{M-j}+1 \leq i \leq (2k)2^{M-j} \\ 0 & \text{else,} \end{cases} \tag{1}$$

where j determines the scale of the wavelet and k determines the shift of the wavelet. To define a wavelet compression algorithm, we can derive the respective coefficients of the wavelets as $\lambda^{j,k} = x^T\psi^{j,k}/\|\psi^{j,k}\|^2 \in \mathbb{R}$. Next, we remove the small coefficients $\lambda^{j,k}$, or set the coefficients as

$$\bar{\lambda}^{j,k} = \begin{cases} \lambda^{j,k} & \text{if } |\lambda^{j,k}| > \text{thres} \cdot \max_{j,k} |\lambda^{j,k}| \\ 0 & \text{else,} \end{cases} \tag{2}$$

where we zero out the coefficients with magnitude less than some fraction of the greatest coefficient magnitude. Thus, the reconstructed signal with the small coefficients zeroed out is defined as

$$y = \sum_{j=0}^{M} \sum_{k=1}^{\max\{2^{j-1},1\}} \bar{\lambda}^{j,k} \cdot \psi^{j,k}. \tag{3}$$

With this we have constructed a filtered signal y that is similar to x, but with the unnecessary coefficients removed.

Notes:

- Round the output signal to the nearest decimal place
- Try to examine the signals before and after applying the filter using `imshow(uint8(reshape((SIGNAL, 128, 128))))` to see the effects