ECE503 Homework Assignment Number 3 Solution

1. 4 points. Mitra 5.12.
   Solution: Starting from the definition, we can write

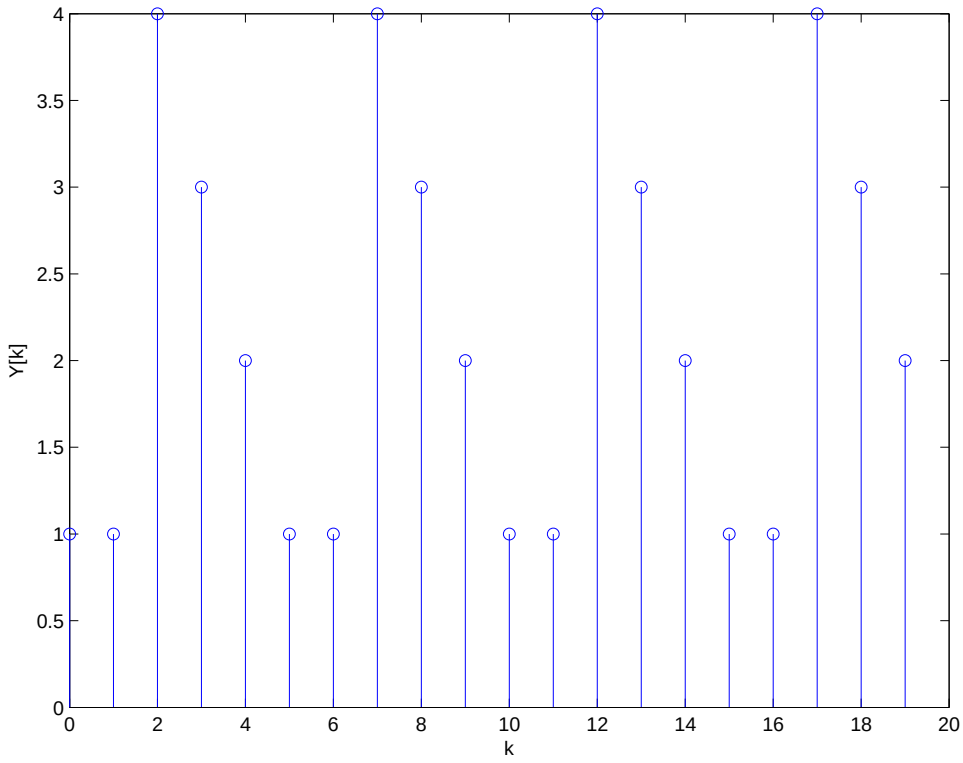
$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{nk} \qquad k = 0, 1, \ldots, N-1 \\
&= \sum_{n \text{ even}} x[n]W_N^{nk} + \sum_{n \text{ odd}} x[n]W_N^{nk} \\
&= \sum_{m=0}^{N/2-1} x[2m]W_N^{2mk} + \sum_{m=0}^{N/2-1} x[2m+1]W_N^{(2m+1)k} \\
&= \underbrace{\sum_{m=0}^{N/2-1} x[2m]W_{N/2}^{mk}}_{N/2\text{-point DFT of even sample indices}} + W_N^k \cdot \underbrace{\sum_{m=0}^{N/2-1} x[2m+1]W_{N/2}^{mk}}_{N/2\text{-point DFT of odd sample indices}} \\
&= \sum_{m=0}^{N/2-1} g[m]W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} h[m]W_{N/2}^{mk} \\
&= \begin{cases} G[k] + W_N^k H[k] & k = 0, \ldots, N/2-1 \\ G[k-N/2] + W_N^k H[k-N/2] & k = N/2, \ldots, N-1 \end{cases}
\end{aligned}$$

   where the last equality is from the periodicity property of the DFT. Splitting an N-point DFT into two $N/2$-point DFTs is one of the main ideas behind the FFT, which we will cover when we get to Chapter 11.

2. 3 points. Mitra 5.20.
   Solution to part (a):

$$\begin{aligned}
Y[k] &= \sum_{n=0}^{NL-1} y[n]W_{NL}^{nk} \qquad k = 0, 1, \ldots, NL-1 \\
&= \sum_{n=0,L,2L,\ldots,(N-1)L} y[n]W_{NL}^{nk} \\
&= \sum_{m=0}^{N-1} x[m]W_{NL}^{mLk} \\
&= \sum_{m=0}^{N-1} x[m]W_N^{mk} \\
&= X[\langle k \rangle_N] \qquad k = 0, 1, \ldots, NL-1
\end{aligned}$$

   Solution to part (b):


Y[k]
k
0
0.5
1
1.5
2
2.5
3
3.5
4
0
2
4
6
8
10
12
14
16
18
20

3. 3 points. Mitra 5.67

Solution:

Given: $y[n] = x[5n], 0 \le n \le \frac{N}{5} - 1.$

Therefore: $Y[k] = \sum_{n=0}^{\frac{N}{5}-1} y[n] W_{N/5}^{nk} = \sum_{n=0}^{\frac{N}{5}-1} x[5n] W_{N/5}^{nk}.$

We can write: $x[n] = \frac{1}{N} \sum_{m=0}^{N-1} X[m] W_N^{-5mn} = \frac{1}{N} \sum_{m=0}^{N-1} X[m] W_{N/5}^{-mn}.$

Hence:

$$Y[k] = \frac{1}{N} \sum_{n=0}^{\frac{N}{5}-1} \sum_{m=0}^{N-1} X[m] W_{N/5}^{-mn} W_{N/5}^{nk} = \frac{1}{N} \sum_{m=0}^{N-1} X[m] \sum_{n=0}^{\frac{N}{5}-1} W_{N/5}^{(k-m)n}.$$

Since:

$$\sum_{n=0}^{\frac{N}{5}-1} W_{N/5}^{(k-m)n} = \begin{cases} \frac{N}{5}, & m = k, k + \frac{N}{5}, k + \frac{2N}{5}, k + \frac{3N}{5}, k + \frac{4N}{5}, k + N, \\ 0, & \text{elsewhere.} \end{cases}$$

Therefore:

$$Y[k] = \frac{1}{5}\left(X[k] + X[k + \frac{N}{5}] + X[k + \frac{2N}{5}] + X[k + \frac{3N}{5}] + X[k + \frac{4N}{5}] + X[k + N] \right).$$

4. 4 points. Mitra 5.76
Solution:

(a) Standard linear convolution yields $y_L[n] = \{-12, 17, 11, -1, -3, -20\}$ for $n = 0, \ldots, 5$.

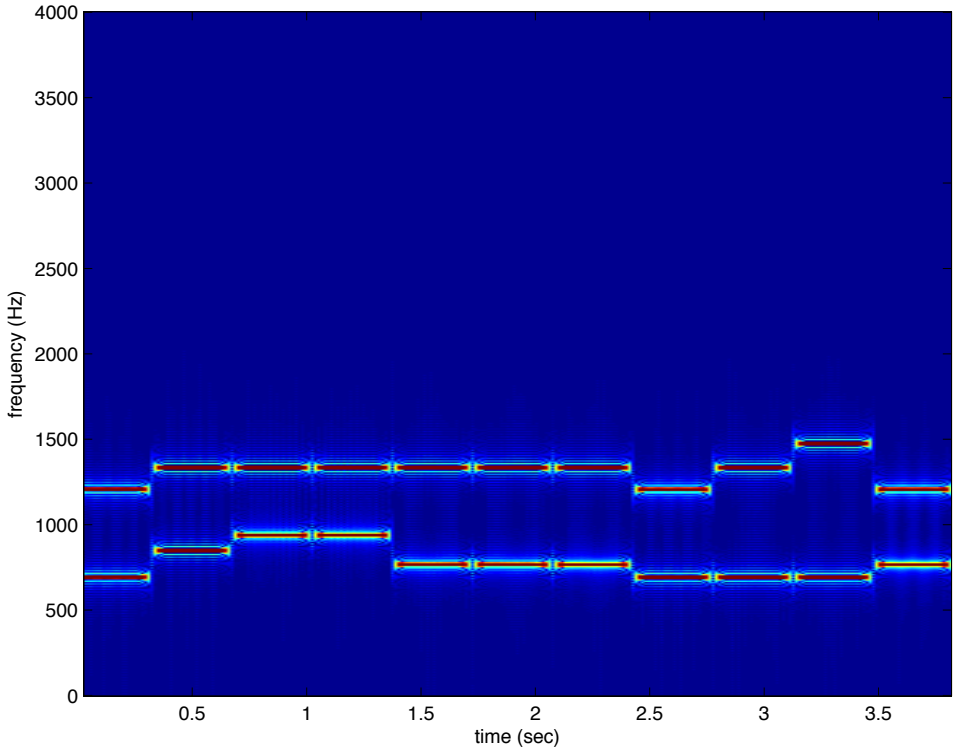
(b) The circular convolution can be computed using the method shown in lecture (or the methods developed in the textbook). The correct answer is $y_C[n] = \{-15, -3, 11, -1\}$.

(c) The DFT approach is to compute $G_e[k] = \mathsf{DFT}\{g_e[n]\}$ for $k = 0, 1, 2, 3$ and $H[k] = \mathsf{DFT}\{h[n]\}$ for $k = 0, 1, 2, 3$ and then compute $y[n] = \mathsf{IDFT}\{G_e[k] \cdot H[k]\}$ for $n = 0, 1, 2, 3$. This approach gives the same answer as part (b).

(d) Zero-padding the sequences to length six and doing the circular convolution either directly or using the DFT method gives the answer $y[n] = \{-12, 17, 11, -1, -3, -20\}$ for $n = 0, \ldots, 5$, which is the same as part (a). This is called "fast convolution" when the circular convolution is performed with FFTs.

5. 4 points. Use Matlab to plot a spectrogram of the .wav file on the course web page and determine the phone number entered by comparing the frequencies present in the signal to the frequencies of DTMF tones.
Solution:

```
[x,fs] = wavread('phonecall.wav');
[s,f,t] = spectrogram(x,ones(1,512),400,1024,fs);
image(t,f,abs(s))
axis xy
xlabel('time (sec)');
ylabel('frequency (Hz)');
```



Zooming in on the figure suggests the following approximate frequency pairs and corresponding DTMF digits:

index	approx low frequency	approx high frequency	digit
1	700	1210	1
2	850	1330	8
3	940	1330	0
4	940	1330	0
5	750	1330	5
6	750	1330	5
7	750	1330	5
8	700	1210	1
9	700	1330	2
10	700	1490	3
11	750	1210	4

6. 7 points. Use the Matlab functions `fft2` and `dct2` to do the following image compression experiments:

   - Load the image "post512.tif" from the course website using the `imread` command. This will result in a 512×512 matrix of values from 0 to 255. You can make Matlab plot this original uncompressed image with the `imshow` command.
   - Perform a two-dimensional DCT on this image, resulting in a 512×512 real-valued matrix.
   - Set all but the N largest magnitude values of this DCT matrix equal to zero.
   - Perform an inverse two-dimensional DCT and plot the resulting image using `imshow`.
   - Repeat the above with the two-dimensional FFT.
   - You should generate three plots: original uncompressed image, DCT compressed image, FFT compressed image.

   Experiment with different values of N to see if you can find cases where the DCT compression looks better to you than the FFT compression. For small values of N, the compressed images will probably not look very good. Note that the FFT will be complex, and after zeroing out some of the values, the inverse FFT may not be real anymore. You should come up with a good way of handling that. Also note when $N = 512 \times 512$, both techniques should produce an image identical to the original image (you aren't throwing away any information).
   Solution: After trying different values for the number of coefficients to save, I settled on $N = 10 * 512$ as the number that seemed to give the biggest perceptual difference between DCT compression and FFT compression. This corresponds to about 2% of the DCT/FFT coefficients. Here is my code:

```
% ECE503 Homework 3
% Spring 2012
% ================================================
% User parameters
% ================================================
N = 10*512;                    % number of values to save
% ================================================

% read in original uncompressed image and display it
X = imread('post512.tif');
figure(1)
imshow(X)

% DCT
J = dct2(X);
%imshow(log(abs(J)),[]), colormap(jet(64)), colorbar
Jsort = sort(abs(J(:)),'descend');
Jmaxval = Jsort(N);
I = abs(J)<Jmaxval;
J(I) = 0;
K = idct2(J);
figure(2)
imshow(K,[0 255])
s1 = sprintf('Percentage of DCT coefficients saved: %2.2f%%',100*(512^2-sum(sum(I)))/512^2);

% ------------------------------------------------------
% FFT
% The approach below ensures FFT coefficients are deleted symmetrically.
```

```
% We may end up deleting slightly fewer than N coefficients, but
% the symmetric deletion of coefficients ensures the IFFT will be real
% -------------------------------------------------------
J = fft2(X);
%imshow(log(abs(J)),[]), colormap(jet(64)), colorbar
[Jsort,index] = sort(abs(J(:)),'descend');
Jmaxval = Jsort(N);
I = abs(J)<Jmaxval;
J(I) = 0;
K = ifft2(J);
figure(3)
imshow(K,[0 255])
s2 = sprintf('Percentage of FFT coefficients saved: %2.2f%%',100*(512^2-sum(sum(I)))/512^2);

% print out information
disp(s1)
disp(s2)
if isreal(K) % confirm IFFT is real
    disp('IFFT2 is real');
else
    disp('IFFT is complex!');
end
```

The original (top), DCT-compressed (bottom left), and FFT-compressed (bottom right) images are shown below.

The DCT-compressed image seems to me to retain more detail in the post than the FFT-compressed image. The sky also looks slightly better in the DCT-compressed image.