Applications of DSP

Project 4

UNIVERSITY OF CENTRAL FLORIDA
DEPARTMENT OF ELECTRICAL ENGINEERING

April 18, 2017

Authored by: **Rajib Dey**

Applications of DSP

Project 4

Question 1:

Here we need to compute 16 point DFT and IDFT (magnitude and phase) of the two given functions:

$$x_1(n) = \cos\left(\frac{2\pi n}{16}\right) + 2\cos\left(\frac{4\pi n}{16}\right)$$

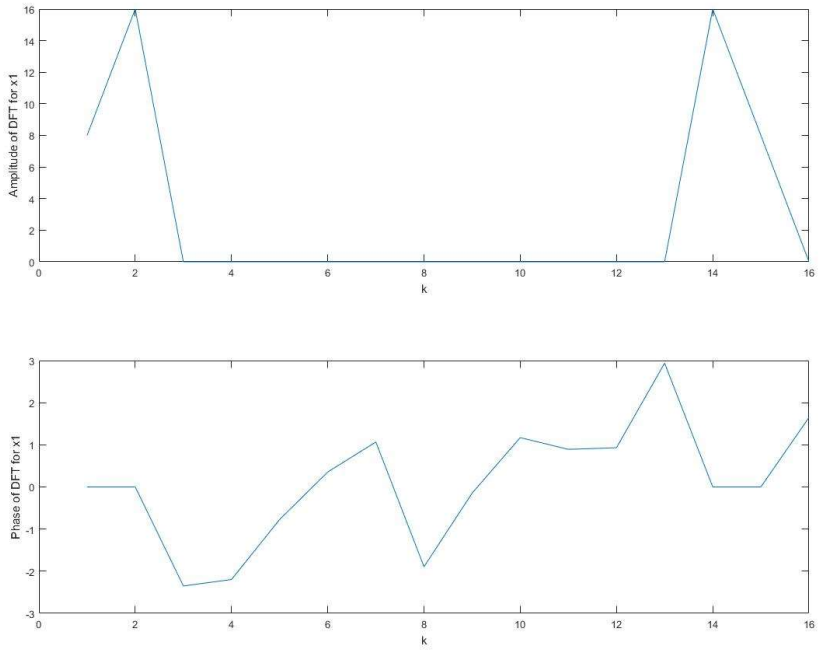
$$x_2(n) = \sin\left(\frac{2\pi n}{16}\right) + 2\sin\left(\frac{4\pi n}{16}\right)$$

Given, $X(k)=DFT\{x(n)\}\&x(n)=IDFT\{X(k)\}$.

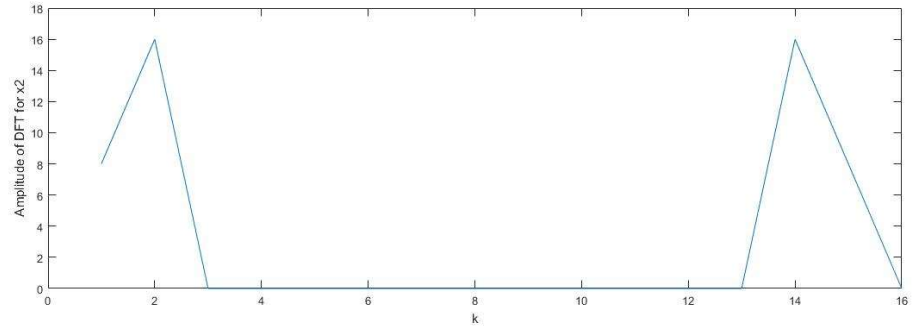
Discrete Fourier transform (DFT) converts a finite sequence of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids. Which is ordered by their frequencies, that has those same sample values. To convert the sampled function from its original domain (often time or position along a line) to the frequency domain.

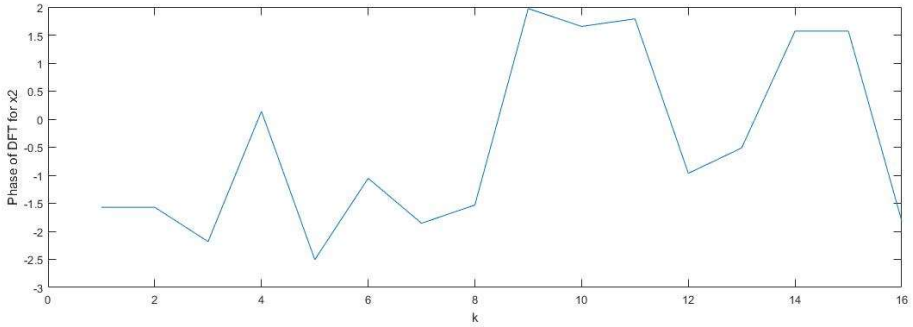
The resulting plots are shown below:

DFT of $x1(n)$:

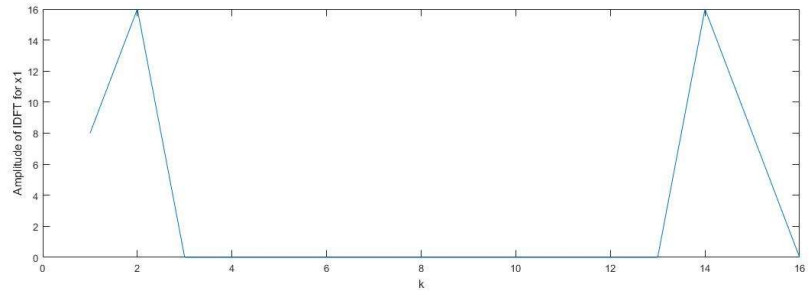


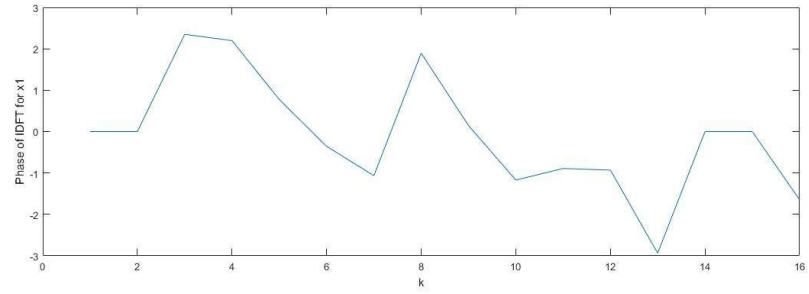
DFT of $x2(n)$:



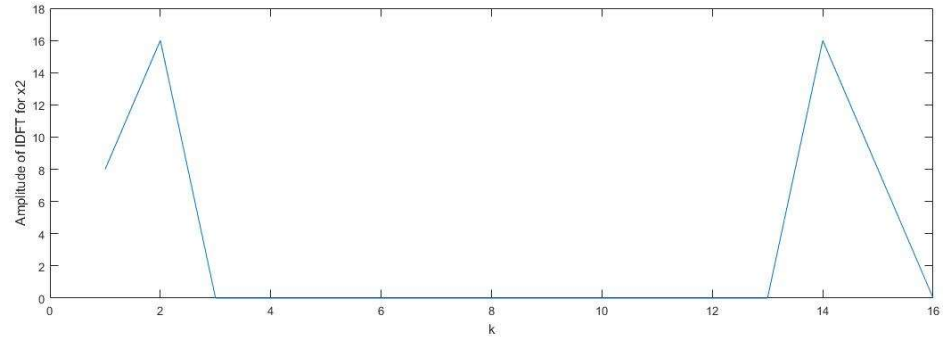


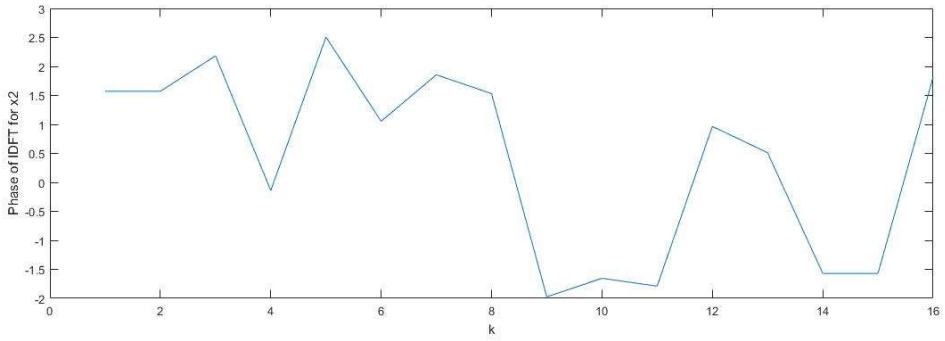
IDFT of $X1(k)$:





IDFT of $X2(k)$:





Question 2:

Given,

x(nT) = 4 sin $(\frac{2\pi n}{8})$ and $\alpha = 0.5$

For n = 0 to 15, compute y(n) using:

a. Difference equation
b. Convolution summation
c. FFT (overlap and add, 16-point - your own FFT)

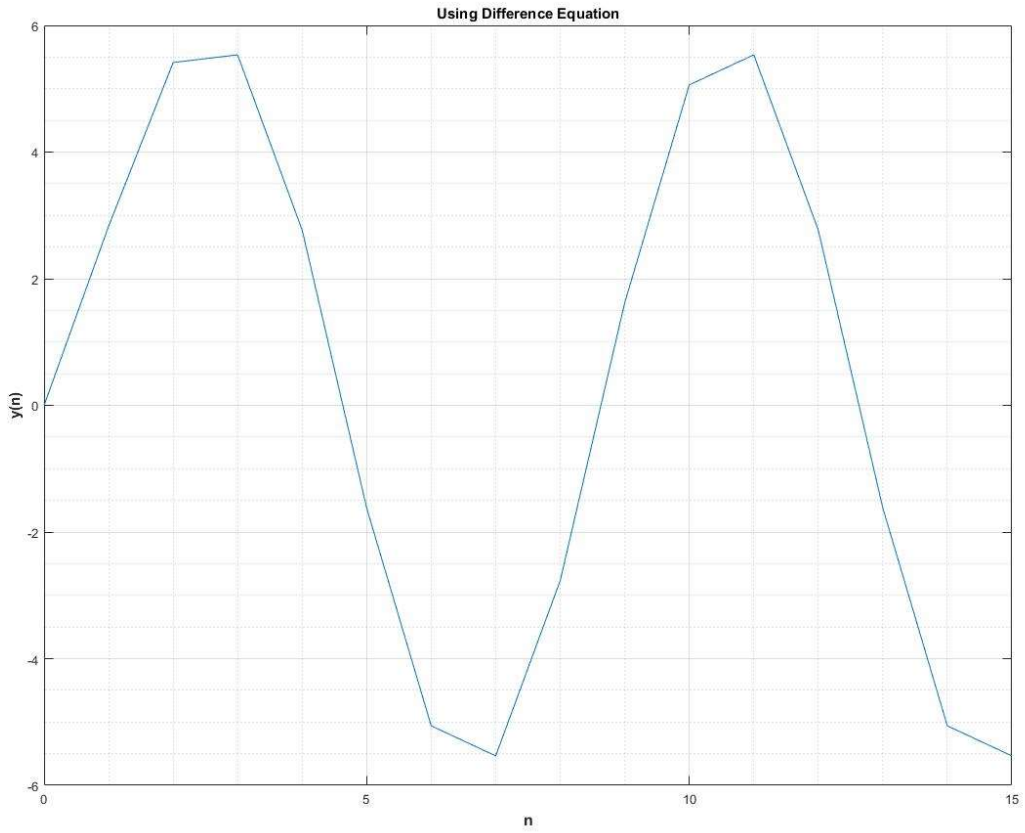
We know from the lectures,

$$y(nT)=X(nT)+\frac{1}{2}X(nT-T)+\frac{1}{4}X(nT-2T)+\frac{1}{8}X(nT-3T)$$

a. Difference equation:

From the MATLAB Console, we get the values of y(n)

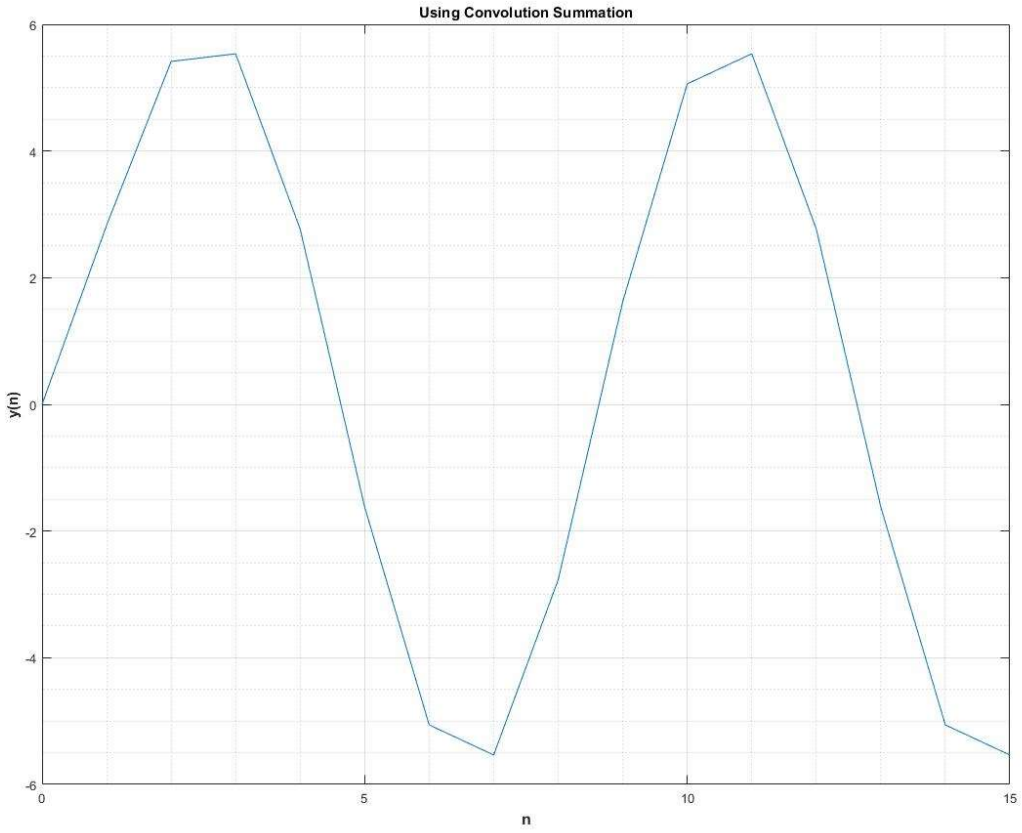
```
y_2a =

  Columns 1 through 15

         0    2.8284    5.4142    5.5355    2.7678   -1.6213   -5.0607   -5.5355   -2.7678    1.6213    5.0607    5.5355    2.7678   -1.6213   -5.0607

  Column 16

   -5.5355
```



b. Convolution summation:

From the MATLAB Console, we get the values of y(n)

```
y_2b =

  Columns 1 through 15

         0    2.8284    5.4142    5.5355    2.7678   -1.6213   -5.0607   -5.5355   -2.7678    1.6213    5.0607    5.5355    2.7678   -1.6213   -5.0607

  Column 16

   -5.5355
```

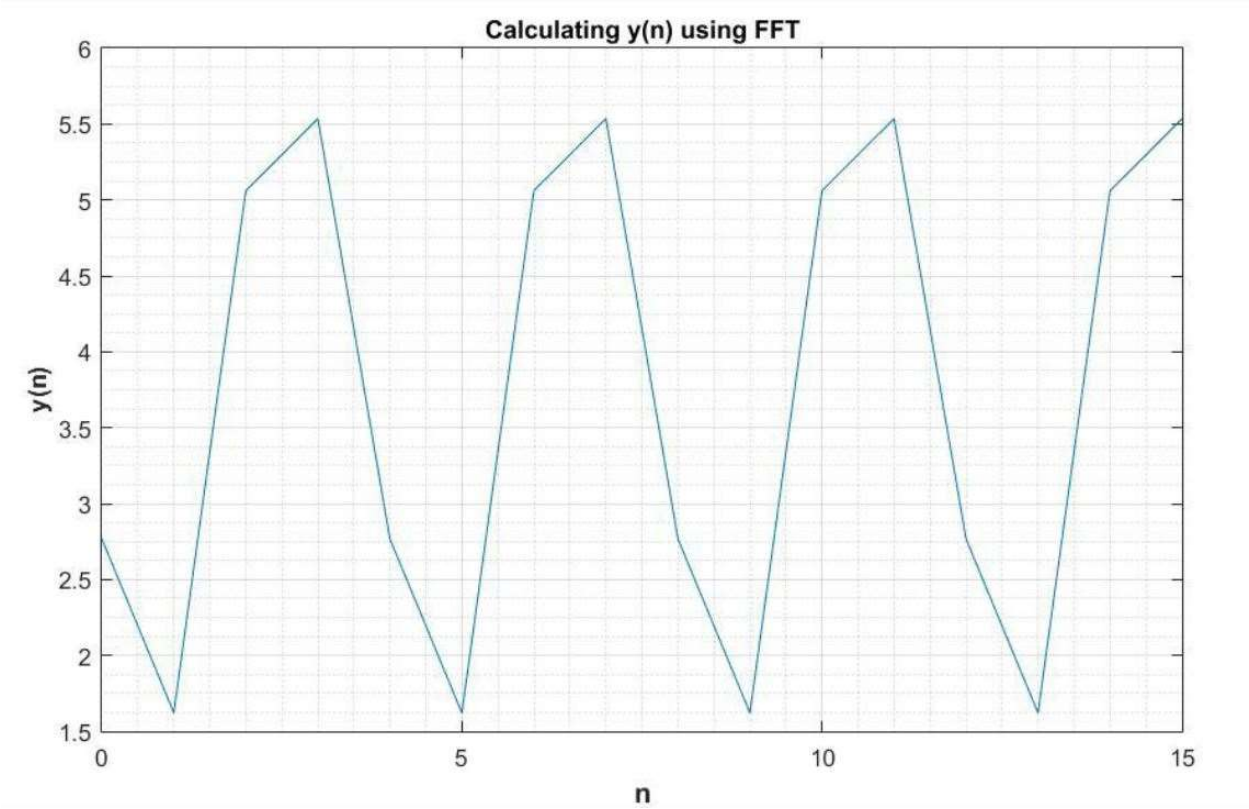


---

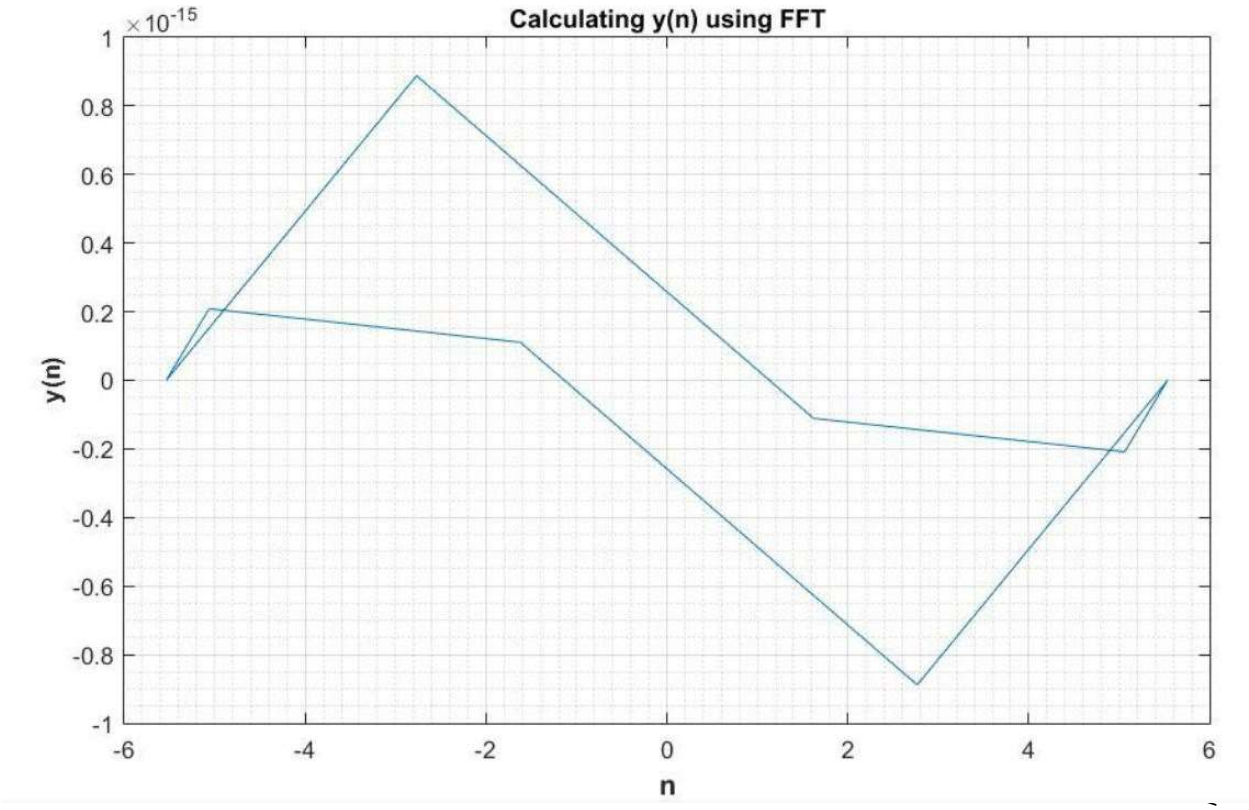
c. FFT (overlap and add):

From the MATLAB Console, we get the values of y(n)

```
y_2c =

  Columns 1 through 9

  -2.7678 + 0.0000i   1.6213 - 0.0000i   5.0607 - 0.0000i   5.5355 - 0.0000i   2.7678 - 0.0000i  -1.6213 + 0.0000i  -5.0607 + 0.0000i  -5.5355 - 0.0000i  -2.7678 + 0.0000i

  Columns 10 through 16

   1.6213 - 0.0000i   5.0607 - 0.0000i   5.5355 + 0.0000i   2.7678 - 0.0000i  -1.6213 + 0.0000i  -5.0607 + 0.0000i  -5.5355 + 0.0000i
```



After overlapping and adding,



Question 3:

Given,

$x(n)=3\sin\left(\frac{6\pi n}{16}\right)$

We need to compute and compare the energy in the signal from its 8-point discrete time domain and DFT representation.

Parseval's theorem usually refers to the result that the Fourier transform is unitary; loosely, that the sum (or integral) of the square of a function is equal to the sum (or integral) of the square of its transform.

It is often written as:

$$\int_{-\infty}^{\infty} |x(t)|^2\, dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2\, d\omega = \int_{-\infty}^{\infty} |X(2\pi f)|^2\, df$$

The interpretation of this form of the theorem is that the total energy of a signal can be calculated by summing power-per-sample across time or spectral power across frequency.

For discrete time signals, the theorem becomes:

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |X(e^{i\phi})|^2 d\phi$$

Alternatively, for the discrete Fourier transform (DFT), the relation becomes:

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |X[k]|^2$$

As discussed in the class lecture, The energy can be calculated using the following equations

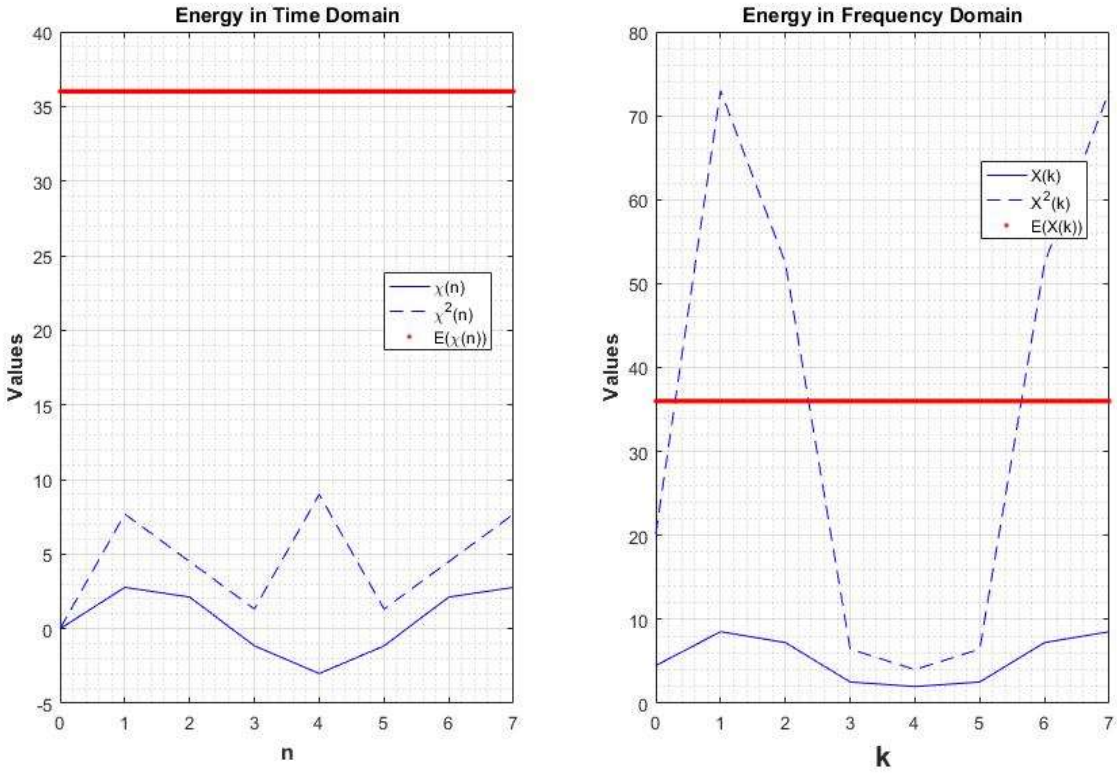
$$\mathrm{E}_1 = [X*(k)]^T[\mathrm{X(k)}] = \sum_{k=0}^{N-1} |x(n)|^2$$

$$\mathrm{E}_2 = [X*(k)]^T[\mathrm{X(k)}] = \sum_{k=0}^{N-1} |x(n)|^2$$

From the MATLAB console window, we get:

```
E_t =

    36


E_f =

   36.0000
```

By plotting them together to compare we get:



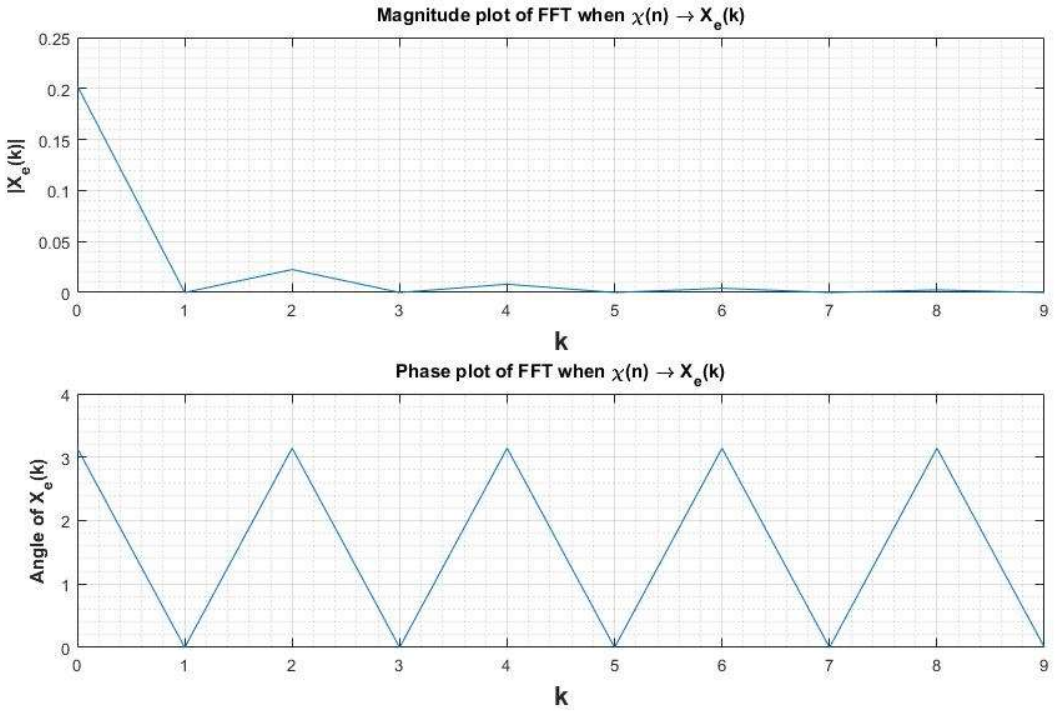
Question 4:

Obtain Fourier series coefficients (complex exponential) from d.c. to 10th harmonic

a. Fourier coefficients using exact

From the MATLAB console window, we get:

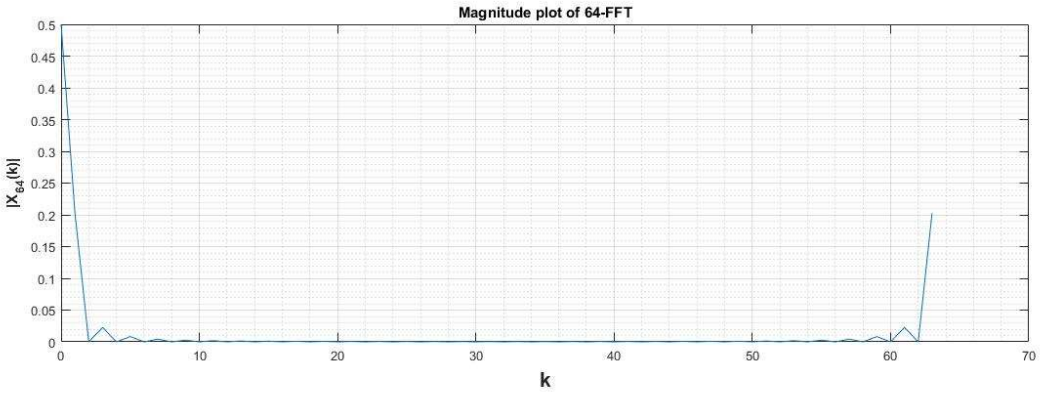
```
X_k_a =

  -0.2026         0   -0.0225         0   -0.0081         0   -0.0041         0   -0.0025         0
```

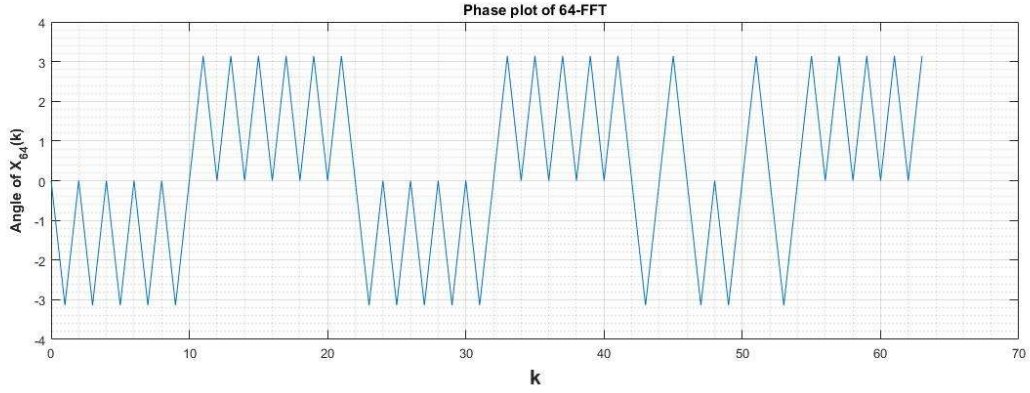


b. Using 64-point DFT or FFT

From the MATLAB console window, we get:

```
X_3_b =

  Columns 1 through 7

   0.5000 + 0.0000i  -0.2028 - 0.0000i   0.0000 + 0.0000i  -0.0227 - 0.0000i   0.0000 + 0.0000i  -0.0083 - 0.0000i   0.0000 + 0.0000i

  Columns 8 through 14

  -0.0043 - 0.0000i   0.0000 + 0.0000i  -0.0027 - 0.0000i   0.0000 + 0.0000i  -0.0018 + 0.0000i   0.0000 + 0.0000i  -0.0014 + 0.0000i

  Columns 15 through 21

   0.0000 + 0.0000i  -0.0011 + 0.0000i   0.0000 + 0.0000i  -0.0009 + 0.0000i   0.0000 + 0.0000i  -0.0008 + 0.0000i   0.0000 + 0.0000i

  Columns 22 through 28

  -0.0007 + 0.0000i   0.0000 + 0.0000i  -0.0006 - 0.0000i   0.0000 + 0.0000i  -0.0006 - 0.0000i   0.0000 + 0.0000i  -0.0005 - 0.0000i

  Columns 29 through 35

   0.0000 + 0.0000i  -0.0005 - 0.0000i   0.0000 + 0.0000i  -0.0005 - 0.0000i   0.0000 + 0.0000i  -0.0005 + 0.0000i   0.0000 + 0.0000i

  Columns 36 through 42

  -0.0005 + 0.0000i   0.0000 + 0.0000i  -0.0005 + 0.0000i   0.0000 + 0.0000i  -0.0006 + 0.0000i   0.0000 + 0.0000i  -0.0006 + 0.0000i

  Columns 43 through 49

   0.0000 + 0.0000i  -0.0007 - 0.0000i   0.0000 + 0.0000i  -0.0008 + 0.0000i   0.0000 + 0.0000i  -0.0009 - 0.0000i   0.0000 + 0.0000i

  Columns 50 through 56

  -0.0011 - 0.0000i   0.0000 + 0.0000i  -0.0014 + 0.0000i   0.0000 + 0.0000i  -0.0018 - 0.0000i   0.0000 + 0.0000i  -0.0027 + 0.0000i

  Columns 57 through 63

   0.0000 + 0.0000i  -0.0043 + 0.0000i   0.0000 + 0.0000i  -0.0083 + 0.0000i   0.0000 + 0.0000i  -0.0227 + 0.0000i   0.0000 + 0.0000i

  Column 64

  -0.2028 + 0.0000i
```





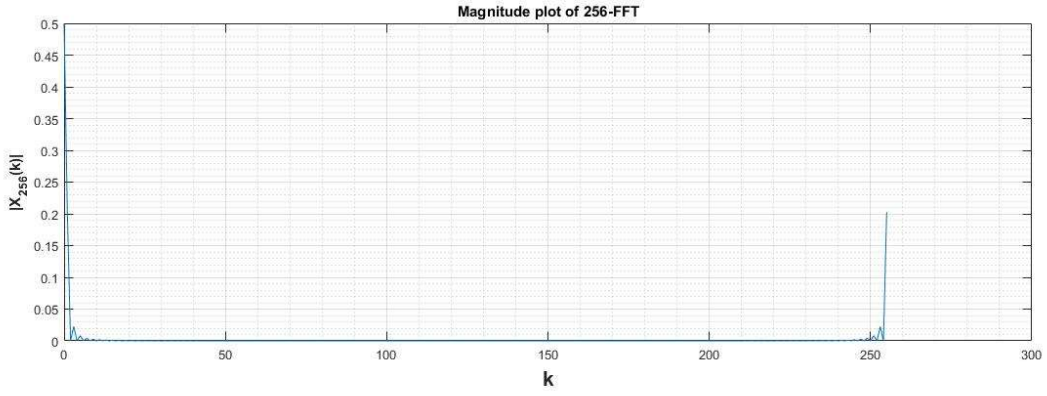
c. Using 256-point DFT or FFT

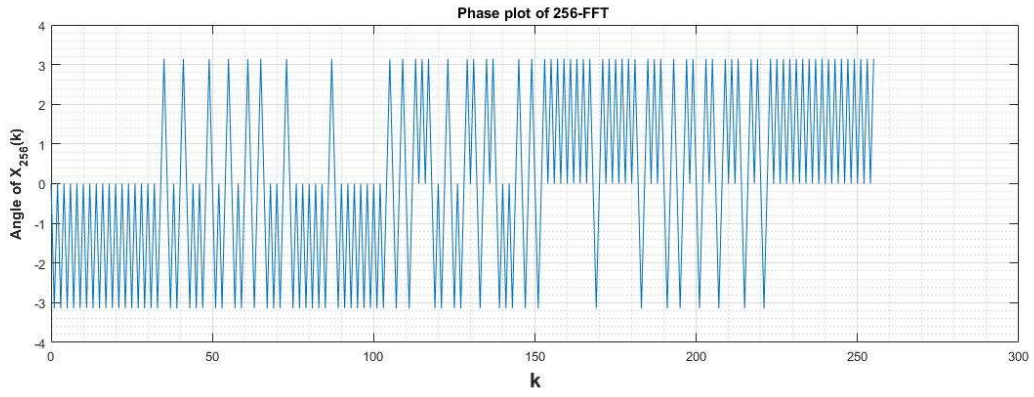
From the MATLAB console window, we get:

```
X_3_256b =

  Columns 1 through 7

   0.5000 + 0.0000i  -0.2027 - 0.0000i   0.0000 + 0.0000i  -0.0225 - 0.0000i   0.0000 + 0.0000i  -0.0081 - 0.0000i   0.0000 + 0.0000i

  Columns 8 through 14

  -0.0041 - 0.0000i   0.0000 + 0.0000i  -0.0025 - 0.0000i   0.0000 + 0.0000i  -0.0017 - 0.0000i   0.0000 + 0.0000i  -0.0012 - 0.0000i

  Columns 15 through 21

   0.0000 + 0.0000i  -0.0009 - 0.0000i   0.0000 + 0.0000i  -0.0007 - 0.0000i   0.0000 + 0.0000i  -0.0006 - 0.0000i   0.0000 + 0.0000i

  Columns 22 through 28

  -0.0005 - 0.0000i   0.0000 + 0.0000i  -0.0004 - 0.0000i   0.0000 + 0.0000i  -0.0003 - 0.0000i   0.0000 + 0.0000i  -0.0003 - 0.0000i

  Columns 29 through 35

   0.0000 + 0.0000i  -0.0003 - 0.0000i   0.0000 + 0.0000i  -0.0002 - 0.0000i   0.0000 + 0.0000i  -0.0002 - 0.0000i   0.0000 + 0.0000i

  Columns 36 through 42

  -0.0002 + 0.0000i   0.0000 + 0.0000i  -0.0002 - 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i

  Columns 43 through 49

   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i

  Columns 50 through 56

  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i

  Columns 57 through 63

   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i

  Columns 64 through 70

  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i
```

```
Columns 71 through 77

 0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i

Columns 78 through 84

-0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i

Columns 85 through 91

 0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i

Columns 92 through 98

-0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i

Columns 99 through 105

 0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i

Columns 106 through 112

-0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i

Columns 113 through 119

 0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i

Columns 120 through 126

-0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i

Columns 127 through 133

 0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i

Columns 134 through 140

-0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i

Columns 141 through 147

 0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i

Columns 148 through 154

-0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i

Columns 155 through 161

 0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i

Columns 162 through 168

-0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i

Columns 169 through 175

 0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i

Columns 176 through 182

-0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i

Columns 183 through 189

 0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i

Columns 190 through 196

-0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i

Columns 197 through 203

 0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i

Columns 204 through 210

-0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i
```

```
Columns 211 through 217

 0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0001 - 0.0000i   0.0000 + 0.0000i

Columns 218 through 224

-0.0001 + 0.0000i   0.0000 + 0.0000i  -0.0002 + 0.0000i   0.0000 + 0.0000i  -0.0002 - 0.0000i   0.0000 + 0.0000i  -0.0002 + 0.0000i

Columns 225 through 231

 0.0000 + 0.0000i  -0.0002 + 0.0000i   0.0000 + 0.0000i  -0.0003 + 0.0000i   0.0000 + 0.0000i  -0.0003 + 0.0000i   0.0000 + 0.0000i

Columns 232 through 238

-0.0003 + 0.0000i   0.0000 + 0.0000i  -0.0004 + 0.0000i   0.0000 + 0.0000i  -0.0005 + 0.0000i   0.0000 + 0.0000i  -0.0006 + 0.0000i

Columns 239 through 245

 0.0000 + 0.0000i  -0.0007 + 0.0000i   0.0000 + 0.0000i  -0.0009 + 0.0000i   0.0000 + 0.0000i  -0.0012 + 0.0000i   0.0000 + 0.0000i

Columns 246 through 252

-0.0017 + 0.0000i   0.0000 + 0.0000i  -0.0025 + 0.0000i   0.0000 + 0.0000i  -0.0041 + 0.0000i   0.0000 + 0.0000i  -0.0081 + 0.0000i

Columns 253 through 256

 0.0000 + 0.0000i  -0.0225 + 0.0000i   0.0000 + 0.0000i  -0.2027 + 0.0000i
```





d. Compare the accuracy in 2.b and 2.c and give justification:

FFT gives the accurate reconstruction of the signal when compared to the convolution. Convolution, for discrete-time sequences, is equivalent to polynomial multiplication which is not the same as the term-by-term multiplication. Convolution also requires a lot more calculation typically N^2multiplications for sequences of length N instead of the N multiplications of the term-by-term multiplication.

The key point of Fourier analysis is that term-by-term multiplication in one domain is the same as convolution in the other domain. So, to calculate the results of a convolution, you can either do it directly, using N^2 multiplications, or transform to the other domain, do a term-by-term multiplication, and transform back. This requires two transformations to go from one domain to the other, N multiplications in the other domain, and one inverse transformation to come back to the domain where the convolution result is needed. This more complicated process can, in fact, require less computation because the transformations can be done very efficiently via the Fast Fourier Transform (FFT) algorithm which requires about N log N multiplications. So, compare the computational effort in calculating the three transforms and doing the term-by-term multiplication to N^2 to see if the FFT gives you a more efficient method of computing a convolution.

Convolution is defined as $y[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k]$

$$\boldsymbol{x(k)} = \sum_{\boldsymbol{n=0}}^{\boldsymbol{n=N-1}} \boldsymbol{x(n)}\,.\,\boldsymbol{e}^{\left(-\frac{\boldsymbol{jwkn}}{\boldsymbol{N}}\right)} \qquad 0 < k < N-1$$

Where x[n] is input signal, h[n] is impulse response, and y[n] is output which denotes convolution. Notice that we multiply the terms of x[k] by the terms of a time-shifted h[n] and add them up. In general, a signal can be decomposed as a weighted sum of basis signals. For example, in Fourier series, any periodic signal (even rectangular pulse signal) can be represented by a sum of sine and cosine functions. But here, we use impulse (delta) functions for the basis signals, instead of sine and cosine. So, the period signal is converted into the impulse coefficients using convolution.

By comparing both graphs, we can say that here in this case also FFT has been more accurate than convolution-summation, because In the FFT graph we have more peaks than the graph we got from convolution-summation.

MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EEL 5513 Applications of DSP
% Project 4
% Submitted by Rajib Dey
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Code for Qu. no. 1-------------------------------------
N=16;
n=1:N;
x1=cos(2.*pi.*n./16)+2.*cos(4.*pi.*n./16);
x2=sin(2.*pi.*n./16)+2.*sin(4.*pi.*n./16);
DFT_x1 = []; DFT_x2 = [];
IDFT_X1 = []; IDFT_X2 = [];
for k=1:N
    X1(k)=sum(x1(n).*exp(-j.*(2.*pi./N).*k.*n));
    X2(k)=sum(x2(n).*exp(-j.*(2.*pi./N).*k.*n));
    x1_IDFT(k)=sum(x1(n).*exp(j.*(2.*pi./N).*k.*n));
    x2_IDFT(k)=sum(x2(n).*exp(j.*(2.*pi./N).*k.*n));
end

figure(1)
subplot(2,1,1);
plot(abs(X1));
xlabel('k');
ylabel('Amplitude of DFT for x1');
subplot(2,1,2);
plot(angle(X1));
xlabel('k');
ylabel('Phase of DFT for x1');

figure(2)
subplot(2,1,1);
plot(abs(X2));
xlabel('k');
ylabel('Amplitude of DFT for x2');
subplot(2,1,2);
plot(angle(X2));
xlabel('k');
ylabel('Phase of DFT for x2');

figure(3)
subplot(2,1,1);
plot(abs(x1_IDFT));
xlabel('k');
ylabel('Amplitude of IDFT for x1');
subplot(2,1,2);
plot(angle(x1_IDFT));
xlabel('k');
ylabel('Phase of IDFT for x1');

figure(4)
```

```
subplot(2,1,1);
plot(abs(x2_IDFT));
xlabel('k');
ylabel('Amplitude of IDFT for x2');
subplot(2,1,2);
plot(angle(x2_IDFT));
xlabel('k');
ylabel('Phase of IDFT for x2');

% Code for Qu. no. 2a---------------------------------------
for n=0:15;
x(n+1)=4.*sin(2.*pi.*n./8);

if n-1<0
    t1=0;
else
    t1=x(n);
end

if n-2<0
    t2=0;
else
    t2=x(n-1);
end

if n-3<0
    t3=0;
else
    t3=x(n-2);
end

y(n+1)=x(n+1)+0.5*t1+0.5^2*t2+0.5^3*t3;
end

y

figure('Name','Using Difference Equation','NumberTitle','off','Position',[100
100 1000 600]);
plot(0:1:15,y);
grid on;
grid minor;
xlabel('n','FontSize',12,'FontWeight','bold');
ylabel('y(n)','FontWeight','bold');
title('Using Difference Equation');
% Code for Qu. no. 2b---------------------------------------
i=0:3;
h=0.5.^i;

for n=0:15;
x(n+1)=4.*sin(2.*pi.*n./8);

if n-1<0
    t1=0;
else
    t1=x(n);
```

```
end

if n-2<0
    t2=0;
else
    t2=x(n-1);
end

if n-3<0
    t3=0;
else
    t3=x(n-2);
end

y(n+1)=h*[x(n+1);t1;t2;t3];
end

y

figure('Name','Using Convolution
Summation','NumberTitle','off','Position',[100 100 1000 600]);
plot(0:1:15,y);
grid on;
grid minor;
xlabel('n','FontSize',12,'FontWeight','bold');
ylabel('y(n)','FontWeight','bold');
title('Using Convolution Summation');

% Code for Qu. no. 2c--------------------------------------
h=[h,zeros(1,12)];
X=fft16(x);
H=fft16(h);
y=ifft16(1/16.*X.*H);
y

figure('Name','Using FFT','NumberTitle','off','Position',[100 100 1000 600]);
plot(y);
grid on;
grid minor;
xlabel('n','FontSize',12,'FontWeight','bold');
ylabel('y(n)','FontWeight','bold');
title('Using FFT');

% Code for Qu. no. 3--------------------------------------
figure('Name','Compute and Compare Engery using 8 point DFT in Time & Freq.
Domain','NumberTitle','off','Position',[100 100 1000 600]);
n = 0:1:7;
subplot(1,2,1);
x_3 = 3*sin(6*pi*n/16);
plot(0:1:7,x_3,'b-');
hold on;
x_3_sq = x_3.^2;
plot(0:1:7,x_3_sq,'b--');
hold on;
E_t = sum(x_3_sq);
```

```
plot(0:0.001:7,E_t,'r.');
hold off;
grid on;
grid minor;
xlabel('n','FontSize',12,'FontWeight','bold');
ylabel('Values','FontWeight','bold');
title('Energy in Time Domain');
legend('\chi(n)','\chi^2(n)','E(\chi(n))','Location','Best');
E_t
subplot(1,2,2);
X_3 = fft(x_3,8);
plot(0:1:7,abs(X_3),'b-');
hold on;
X_3_sq = abs(X_3.^2);
plot(0:1:7,X_3_sq,'b--');
hold on;
E_f = 1/8*sum(X_3_sq);
plot(0:0.001:7,E_f,'r.');
hold off;
grid on;
grid minor;
xlabel('k','FontSize',16,'FontWeight','bold');
ylabel('Values','FontWeight','bold');
title('Energy in Frequency Domain');
legend('X(k)','X^2(k)','E(X(k))','Location','Best');

E_f

% Code for Qu. no. 4-------------------------------------
figure('Name','Fourier Coefficients using
Exact','NumberTitle','off','Position',[100 100 1000 600]);
k = 1:1:10;
X_k_a = -(1-cos(k.*pi))./(k.*pi).^2;
X_k_a
subplot(2,1,1);
plot(0:1:9,abs(X_k_a));
grid on;
grid minor;
xlabel('k','FontSize',16,'FontWeight','bold');
ylabel('|X_e(k)|','FontWeight','bold');
title('Magnitude plot of FFT when \chi(n) \rightarrow X_e(k)');
subplot(2,1,2);
plot(0:1:9,angle(X_k_a));
grid on;
grid minor;
xlabel('k','FontSize',16,'FontWeight','bold');
ylabel('Angle of X_e(k)','FontWeight','bold');
title('Phase plot of FFT when \chi(n) \rightarrow X_e(k)');
figure('Name','Fourier Coefficients using 64
FFT','NumberTitle','off','Position',[100 100 1000 600]);
n = 0:1:63;
x_3_b = 1-abs(1-n./32);
X_3_b = fft(x_3_b)/64;
X_3_b
subplot(2,1,1);
plot(0:1:63,abs(X_3_b));
```

```
grid on;
grid minor;
xlabel('k','FontSize',16,'FontWeight','bold');
ylabel('|X_6_4(k)|','FontWeight','bold');
title('Magnitude plot of 64-FFT');
subplot(2,1,2);
plot(0:1:63,angle(X_3_b));
grid on;
grid minor;
xlabel('k','FontSize',16,'FontWeight','bold');
ylabel('Angle of X_6_4(k)','FontWeight','bold');
title('Phase plot of 64-FFT');
figure('Name','Fourier Coefficients using 256
FFT','NumberTitle','off','Position',[100 100 1000 600]);
n = 0:1:255;
x_3_256b = 1-abs(1-n./128);
X_3_256b = fft(x_3_256b)/256;
X_3_256b
subplot(2,1,1);
plot(0:1:255,abs(X_3_256b));
grid on;
grid minor;
xlabel('k','FontSize',16,'FontWeight','bold');
ylabel('|X_2_5_6(k)|','FontWeight','bold');
title('Magnitude plot of 256-FFT');
subplot(2,1,2);
plot(0:1:255,angle(X_3_256b));
grid on;
grid minor;
xlabel('k','FontSize',16,'FontWeight','bold');
ylabel('Angle of X_2_5_6(k)','FontWeight','bold');
title('Phase plot of 256-FFT ');
```