Digital Signal Processing Course Project

Digital Frequency Spectrum Analysis Based on DFT

Xie Kunyang, 2017200501038

May 15th, 2020

Abstract

I proudly present a cool project about the Digital Frequency Spectrum Analysis Based on DFT which is widely used in signal processing and analyzing.

To begin with, we need to acquire the maximum frequency of the input signal and send it to the anti-aliasing filter. Then, we can get input sequence by sampling, to avoid distortion, the sampling frequency must be set as two times of the maximum frequency. Because the computers can only process finite data, so we need to intercept a segment from the signal, then periodically extend the intercepted signal segment to obtain a virtual infinitely long signal, and finally carry out mathematical processing. If the input sequence is periodic, the main value interval is one period, otherwise, it may cause the leakage error after windowing. In this case, using a reasonable windowing function to reduce the error is very important.

In this experiment, we do all the procedures by software MatLab.

1. Introduction

In this experiment, we need to analyze the frequency spectrum of the signal:

$$x(t)=10\sin(2\pi\times 64t)+\sin(2\pi\times 250t)+20\sin(2\pi\times 256t)+3\sin(2\pi\times 260t)+10\sin(2\pi\times 512t)$$

To make it, the basic procedures are list as follows:

1. Set up sampling frequency, Ω_s.
2. Add an anti-aliasing filter.
3. Analog discrete converting, whose sampling interval is T_s.
4. Windowing.
5. Processing, convert from the time domain to the frequency domain.

2. Problem Formulation

2.1. Set up the sampling frequency

The input signal can be decomposed to five signals:

$$x_1(t)=10\sin(2\pi\times 64t),\ f_1=64$$

$$x_2(t)=\sin(2\pi\times 250t),\ f_2=250$$

$$x_3(t)=20\sin(2\pi\times 256t),\ f_3=256$$

$$x_4(t)=3\sin(2\pi\times 260t),\ f_4=260$$

$$x_5(t)=10\sin(2\pi\times 512t),\ f_5=512$$

We can notice that the maximum frequency is 512, according to Nyquist's law, the sampling frequency and period should be:

$$f_s\geq 2f_M=2f_5=1024Hz$$

$$T_s=\frac{1}{f_s}=\frac{1}{1024}s$$

In this experiment, we choose the sampling frequency equals to 1536Hz.

2.2. Add an anti-aliasing filter

The bandwidth of this filter is Ω_s, in this experiment, the maximum frequency of input is smaller than this frequency, so this step makes no difference.

2.3. ADC

Because the computer can only process discrete values, this step can transform the continuous signal to the discrete sequence by sampling. The basic idea is:

$$x[n]=x(nT_s)$$

If the sampling frequency is smaller than the Nyquist frequency, it may cause aliasing in the frequency spectrum and the distortion will occur in the time domain.

2.4. Windowing

The computers can only process finite data, so we need to intercept a segment from the signal, then periodically extend the intercepted signal segment to obtain a virtual infinitely long signal, and finally carry out mathematical processing. If the input sequence is periodic, the main value interval is one period, otherwise, it may cause the leakage error after windowing. In this case, using a reasonable windowing function to reduce the error is very important.

Common window functions are rectangular window, Bartlett window, Hann window, Hamming window, and Blackman window, they can all be generated by MabLab very quickly. In this experiment, we will study the effects of different window functions one by one.

After windowing, the sequence can be expressed as:

$$x_w[n] = x[n]w[n]$$

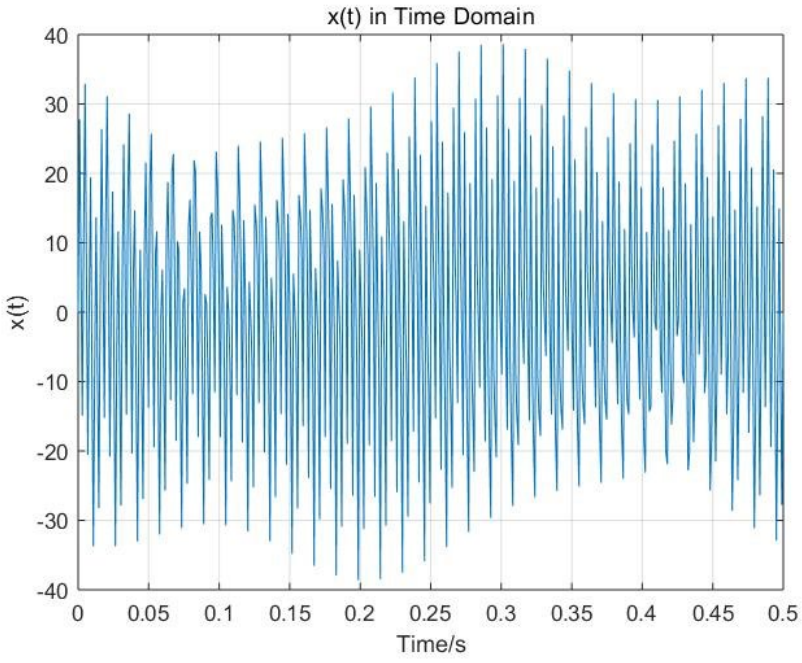
2.5. Processing

We can acquire the frequency spectrum in this step by transforming the signal from time-domain to frequency-domain, it can be processed in MatLab by the command `fft();`

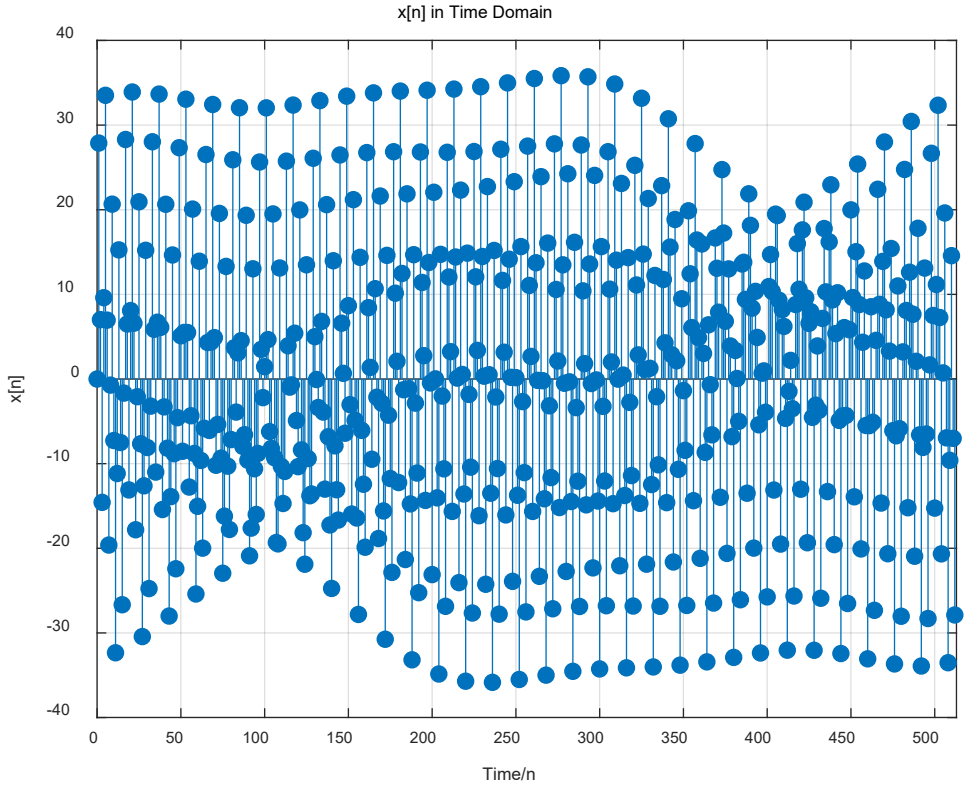
Finally, plot the spectrum to analyze.

3. Numerical Experiments

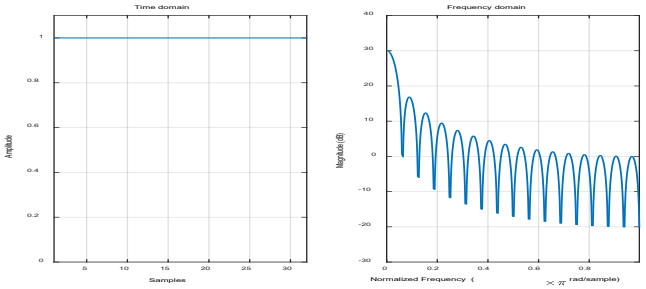
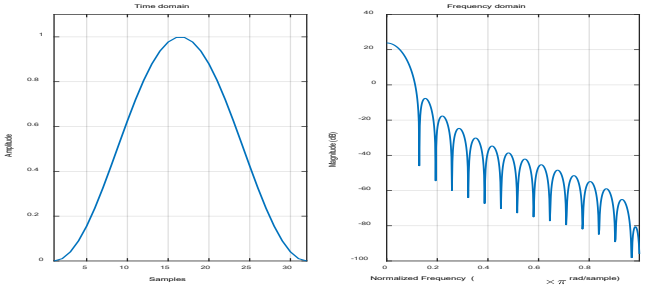
The input signal is a periodic function, we can find the period by finding the least common multiple of these 5 separated signals, which is, 0.5. So, we can get the continuous input signal (Generated by code 1):

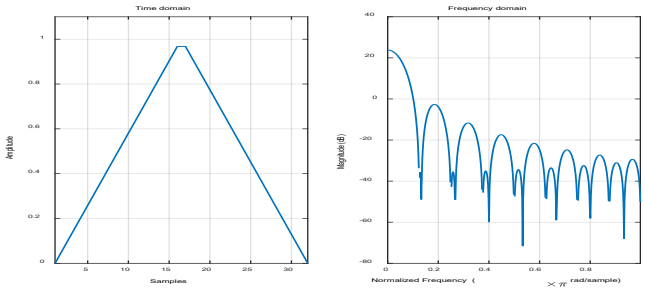
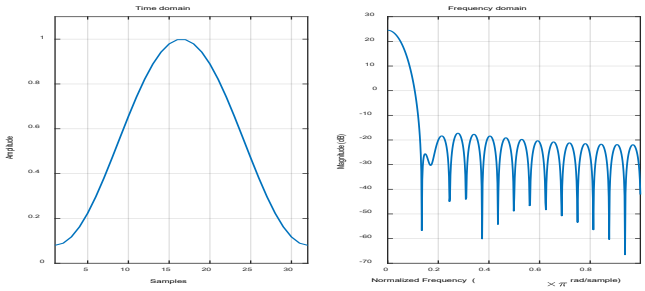
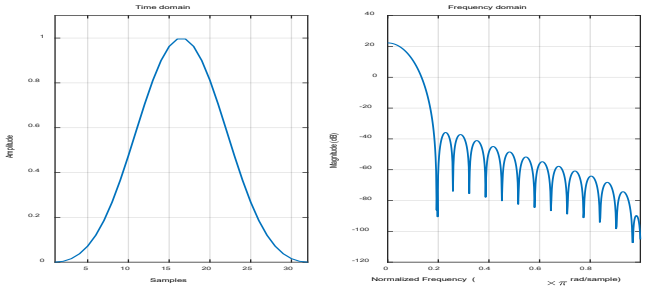


The sampling frequency should be 1024Hz, however, this frequency cannot sample the 512Hz frequency component, so we rise to 1026Hz, the sequence after sampling is (Generated by code 2):

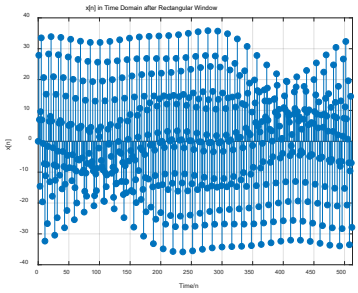
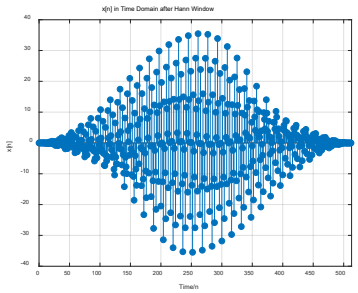


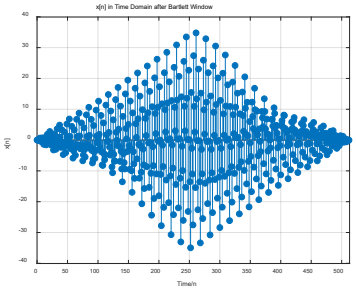
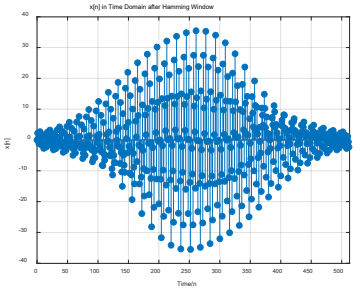
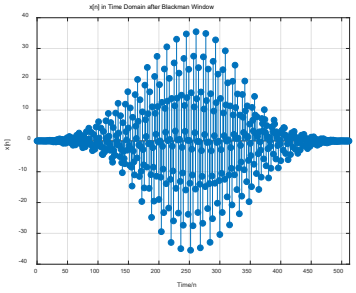
Now, let's look at the effect of the window function (Generated by code 3):

Name	Graph in Time and Frequency
Rectangular	
Hann	

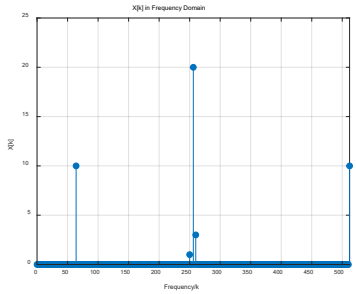
Bartlett	
Hamming	
Blackman	

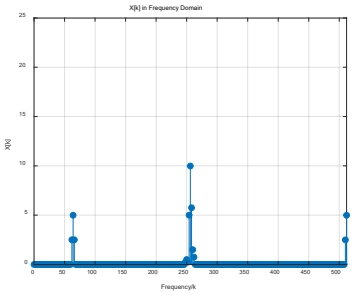
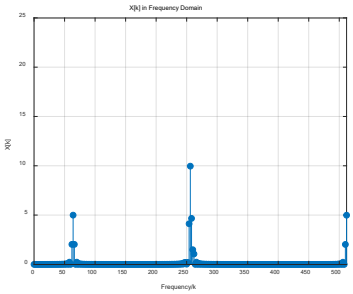
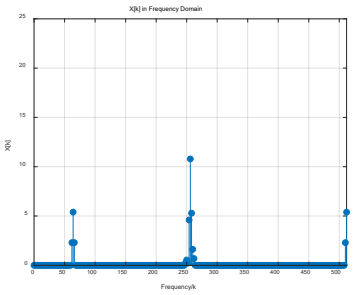
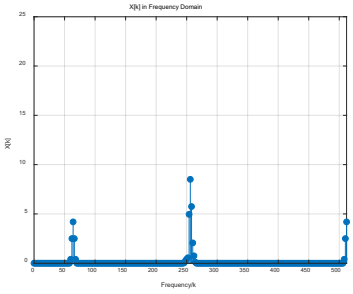
As for the windowed sequences (Generated by code 4):

Name	Windowed Sequence
Rectangular	
Hann	

Bartlett	
Hamming	
Blackman	

The time domain period is 0.5s, and the sampling rate is 1026Hz, therefore, it is a 513-point DFT because it samples 513 times in a main value interval. Finally, the frequency spectrums (Note that we have adjusted the amplitude and the frequency of the spectrums, the vertical axis means the amplitude while the horizontal means the frequency) are (Generated by code 5):

Name	Windowed Sequence
Rectangular	

Hann	
Bartlett	
Hamming	
Blackman	

4. Concluding Remarks

According to figures of frequency spectrum, we can notice that rectangular windows has the best result compare to other windows.

5. Codes

5.1. Code 1:

```
clear;
clc;
fmax = 512;
```

```
fs = 1026;  % Smallest
Ts = 1/fs;  % Sampling Period, Ts = 1/1024
%% Time
t = linspace(0,0.5,512);
x_t = 10*sin(2*pi*64.*t) + sin(2*pi*250.*t) + 20*sin(2*pi*256.*t) +
3*sin(2*pi*260.*t) + 10*sin(2*pi*512.*t);
T = 0.5;   % Period
figure
plot(t,x_t);
xlabel('Time/s');
ylabel('x(t)');
grid on;
title('x(t) in Time Domain');
axis([0 T -40 40]);
```

5.2. Code 2:

```
%% Sampling
n_max = T/Ts;   % Main interval
n = 0:n_max-1;
x_n = 10*sin(2*pi*64.*n*Ts) + sin(2*pi*250.*n*Ts) + 20*sin(2*pi*256.*n*Ts) +
3*sin(2*pi*260.*n*Ts) + 10*sin(2*pi*512.*n*Ts);
figure
stem(n,x_n,'filled');
xlabel('Time/n');
ylabel('x[n]');
grid on;
title('x[n] in Time Domain');
axis([0 n_max -40 40]);
```

5.3. Code 3:

```
n = 32;
w = boxcar(n);
% w = hann(n);
% w = bartlett(n);
% w = hamming(n);
% w = blackman(n);
wvtool(w);
```

5.4. Code 4:

```
%% Windowing
w = boxcar(n_max);
% w = hann(n_max);
% w = bartlett(n_max);
% w = hamming(n_max);
```

```
% w = blackman(n_max);
w = transpose(w);
x_w = x_n.*w;
figure
stem(n,x_w,'filled');
xlabel('Time/n');
ylabel('x[n]');
grid on;
title('x[n] in Time Domain after Rectangular Window');
axis([0 n_max -40 40]);
```

5.5. Code 5:

```
%% Frequency
X = fft(x_w);
X_abs = abs(X);
figure
stem(2*n,X_abs/1026*4,'filled');
xlabel('Frequency/k');
ylabel('X[k]');
grid on;
title('X[k] in Frequency Domain');
axis([0 512 0 25]);
```

References

[1] MathWorks Help Center, WVTool, Open Window Visualization Tool, https://ww2.mathworks.cn/help/signal/ref/wvtool.html, [Accessed on 18/May/2020]

[2] GitHub, abnormal0, DFT, https://github.com/abnormal0/DFT.git, [Accessed on 23/May/2020]