**实验7 离散信号的DFT&FFT频谱MATLAB仿真**

# 计算机20-1 刘宇诺

# 【实验目的】

1. 掌握序列傅氏变换的计算机实验方法，利用序列的傅氏变换对离散信号、系统及系统响应进行频域分析。
2. 加深对离散信号的DTFT、DFT及其相互关系的理解。
3. 在理论学习的基础上，加深对快速傅里叶变换FFT的理解，熟悉FFT算法
4. 熟悉应用FFT对典型信号进行频谱分析的方法。
5. 加深对离散信号FFT算法的运用，以便在实际中正确应用FFT。

# 【实验内容】

1、验证所有例程，理解原理，观察分析结果，总结实验所得结论。

**【例7-1】**

设x(n)是4点序列{1 1 1 1 },调用dft、idft、dft函数实现以下要求：

1)求解序列DTFT，并画出幅频曲线，

2)分别求解序列4点、8点、16点DFT，并绘制幅频曲线，

代码：

clear;

xn=[1 1 1 1];

k=0:511; %由wk=2πk/512可求得模拟频率f

Xw=dtft(xn,2\*pi\*k/512); % 近似模拟信号频谱

subplot(221),plot(2\*pi\*k/512,abs(Xw));title('DTFT幅频'),xlabel('w')

Xk4=dft(xn,4);

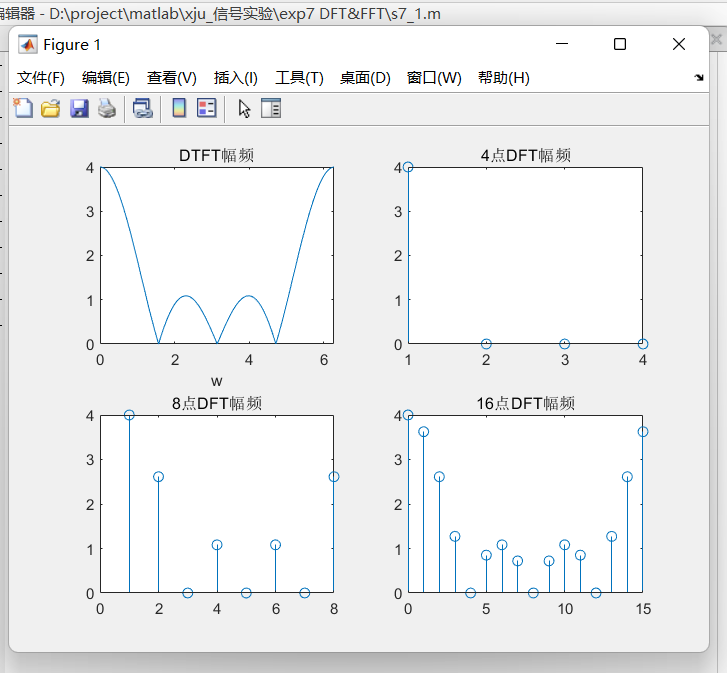
subplot(222),stem(abs(Xk4));title('4点DFT幅频')

Xk8=dft(xn,8);

subplot(223),stem(abs(Xk8));title('8点DFT幅频')

Xk16=dft(xn,16);

subplot(224),stem(0:15,abs(Xk16));title('16点DFT幅频')



分析：随着DFT点数的增加（4点到16点），DFT的结果越来越接近DTFT的结果。因为随着DFT点数的增加，在频域中采样得越密，更接近连续的DTFT。

**【例7-2】**

已知x(n)sin(n/8)sin(n/4)，用 MATLAB 求解N=8,16,32时 DFT 的结果，并绘制幅频曲线，比较结果。

代码：

clear;

N=16;

n=0:1:N-1; %时域采样

xn=sin(n\*pi/8)+sin(n\*pi/4); %以下DFT求解也可以调用自编函数dft实现

k=0:1:N-1; %频域采样

WN=exp(-j\*2\*pi/N);

nk=n'\*k;

WNnk=WN.^nk;41

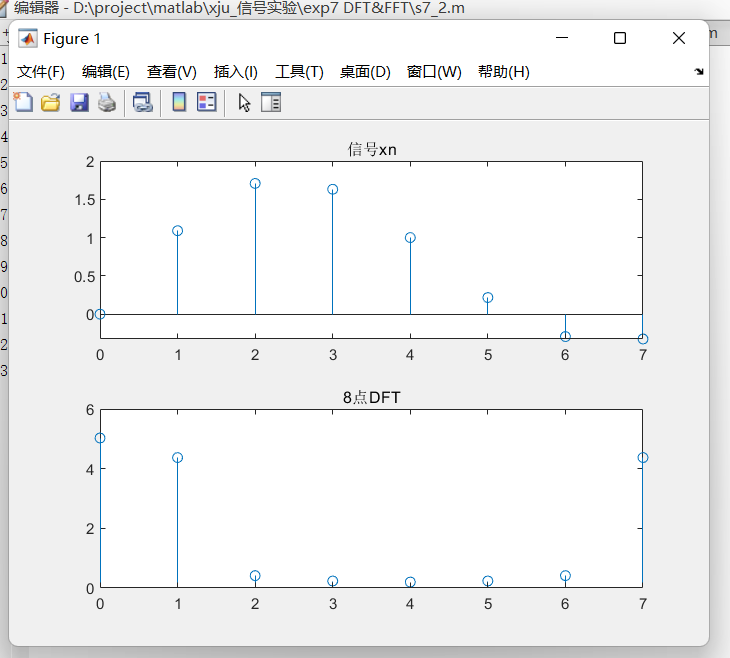
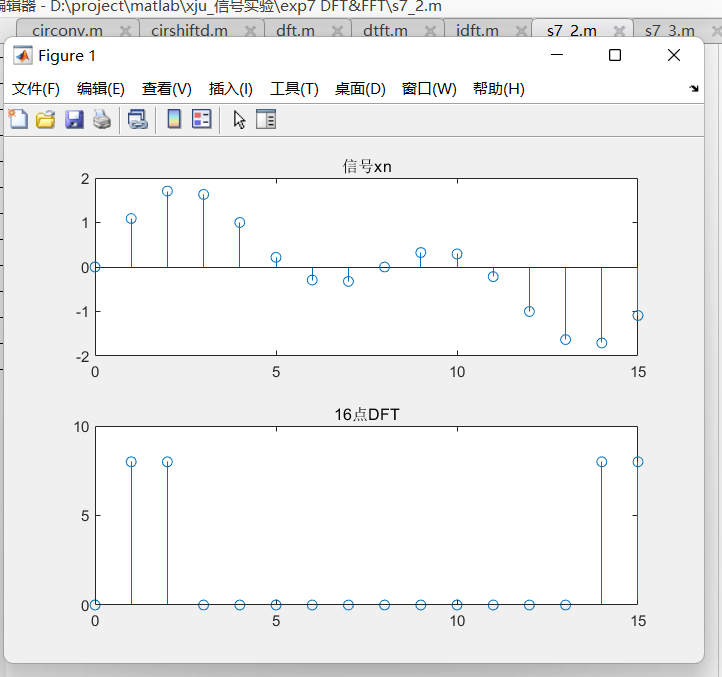
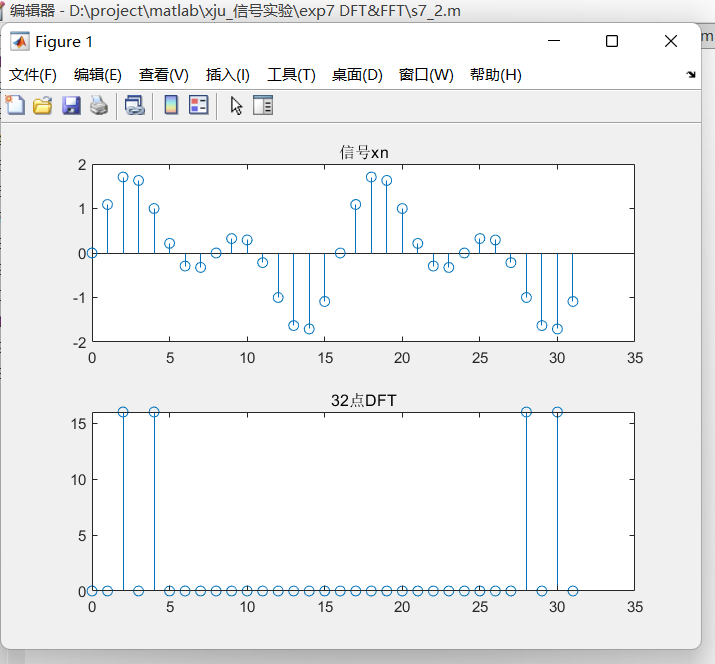
Xk=xn\*WNnk;

subplot(2,1,1)

stem(n,xn); title('信号xn')

subplot(2,1,2)

stem(k,abs(Xk)); title('16点DFT')



分析：第一个子图显示时域信号，第二个子图显示频域信号的幅度。

N取不同的值(8,16,32），程序会对信号进行不同数量的采样，然后计算不同大小的DFT。

N=8时，采样点较少，无法准确表示原始信号，特别是对于较高频率的正弦波分量。

当N=16时，采样点增多，对原始信号的表示更准确，同时，DFT的频率分辨率提高，能够更好地区分接近的频率成分。

当N=32时，采样点进一步增多，对原始信号的表示将更准确。DFT的频率分辨率进一步提高。计算复杂度也会增加。

随着N的增加，对信号的表示越来越精确，DFT的频率分辨率也越来越高，计算复杂度也会随之增加。

**[例7-3]**

求有限长序列的圆周移位,并画出其结果图。

代码：

clear;

N=10;

m=4;

n=0:1:N-1;

x=8\*(0.4).^n;

n1=mod((n+m),N);

xm=x(n1+1);

subplot(2,1,1)

stem(n,x);

title('原始序列');

xlabel('n');

ylabel('x(n)');

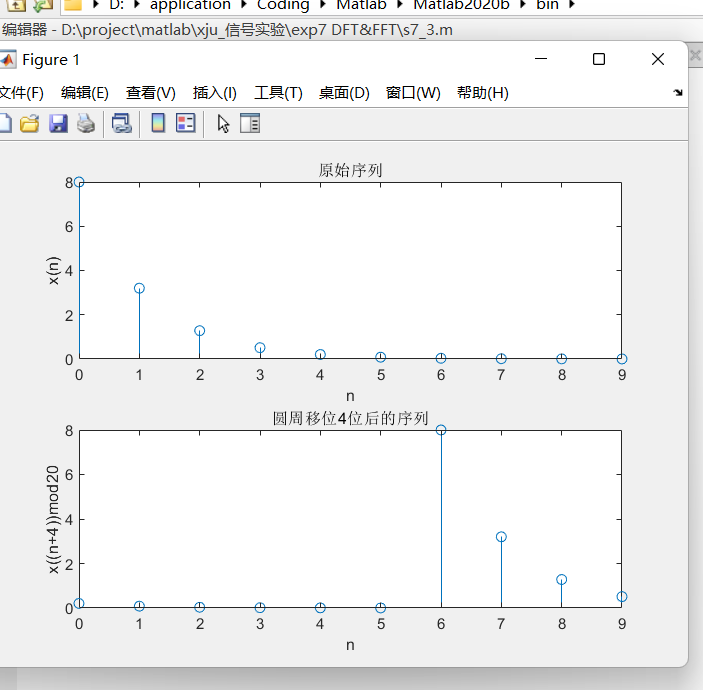
subplot(2,1,2)

stem(n,xm);

title('圆周移位4位后的序列');

xlabel('n');

ylabel('x((n+4))mod20');



分析：

观察图像可以看到整个序列进行圆周移位了4个单位。由于这是一个圆周移位，所以最初的4个元素现在出现在序列的末尾。从整体上看就像是整个序列向左“滚动”了4个单位。

**[例7-4]**

已知4点矩形脉冲R4(N)

1）求解其和自己的线卷积与4点圆卷积。

2）在R4(N)后面添加3个零点，将它扩展成长度为7的序列后再计算它和自己的7点圆周卷积。

代码：

clear all;

xn=[1 1 1 1];

y1=conv(xn,xn); %矩形序列与其自身的线卷积

N=length(xn);

XK=dft(xn,N); %圆周卷积定理

YK=XK.\*XK;

yc4=idft(YK,N); %用圆周卷积定理实现的4点圆周卷积

xn1=[1 1 1 1 0 0 0];

N1=length(xn1);

XK1=dft(xn1,N1);

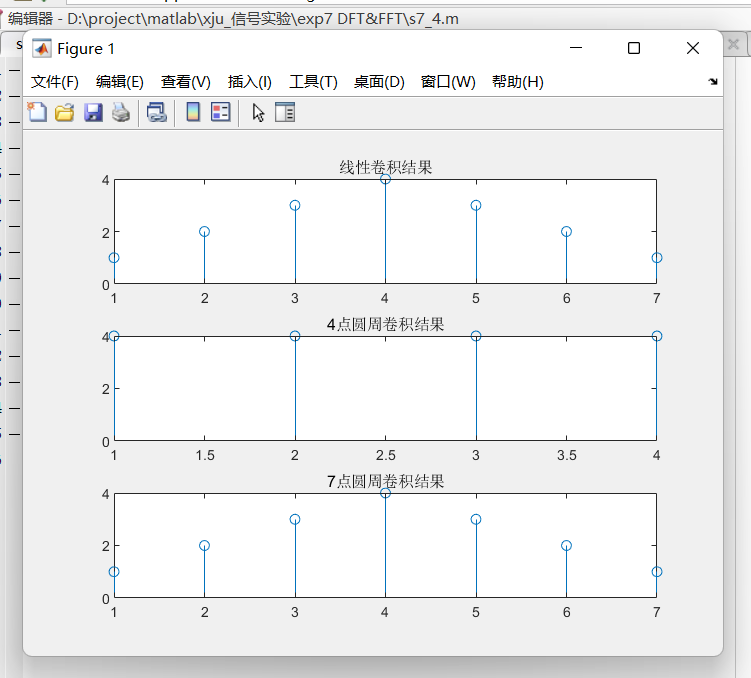
YK1=XK1.\*XK1;

yc7=idft(YK1,N1); %用圆周卷积定理实现的7点圆周卷积

subplot(311),stem(y1),title('线性卷积结果');

subplot(312),stem(yc4),title('4点圆周卷积结果');

subplot(313),stem(yc7),title('7点圆周卷积结果');



分析：

在线性卷积结果中，该序列自身的卷积结果是一个形状为三角形的序列，其峰值为4。

在4点圆周卷积结果中，观察到结果与线性卷积不一样，这是因为在计算圆周卷积时，序列的长度被限制为4，导致卷积的尾部重叠到了头部，形成了“环绕”的效果。

在7点圆周卷积结果中，看到结果与线性卷积结果相同。因为在序列的尾部填充了足够的零，使得卷积结果不会环绕到序列的头部。在这种情况下，圆周卷积等价于线性卷积。

**[实例7-5]**

已知序列

，



求它们的线卷积yl(n)=h(n)\*x(n)和不同N点的圆周卷积yN(n)=h(n)\*x(n)。

代码：

clear all

n=[0:1:11];

m=[0:1:5];

N1=length(n);

N2=length(m);

xn=n; %生成x(n)

hn=ones(1,N2); %生成h(n)

yln=conv(xn,hn); %直接用函数conv计算线性卷积

ycN1=circonv(xn,hn,N1); %用函数circonv计算N1点圆周卷积

ycn=circonv(xn,hn,N1+N2-1); %用函数circonv计算N1+N2-1点圆周卷积

nyl=[0:1:length(yln)-1];

ny1=[0:1:length(ycN1)-1];

ny2=[0:1:length(ycn)-1];

subplot(3,1,1); %画图

stem(nyl,yln,'.');

ylabel('线性卷积');

axis([0,18,0,60]);

subplot(3,1,2);

stem(ny1,ycN1,'.');

ylabel('圆周卷积N1')

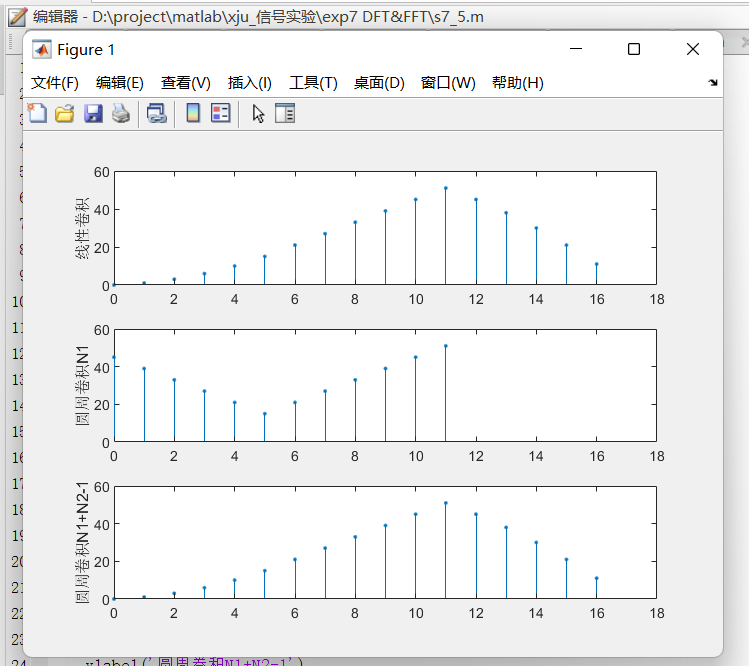
axis([0,18,0,60]);

subplot(3,1,3);

stem(ny2,ycn,'.');

ylabel('圆周卷积N1+N2-1')

axis([0,18,0,60]);



分析：

线性卷积结果中，绘出了序列xn和hn的线性卷积结果。因为序列hn是一个常数序列，所以它的卷积等于在序列xn上应用了一个滑动窗口，窗口内的元素被加起来。结果是一个逐渐上升然后下降的序列，峰值出现在窗口完全包含序列xn的时候。

在N1点圆周卷积结果中，得到序列xn和hn的N1（12）点圆周卷积结果。在这种情况下，由于圆周卷积的环绕效应，观察到到结果在峰值后突然下降。是因为卷积操作在序列的尾部和头部“环绕”导致的。

在N1+N2-1点圆周卷积结果中，得到到序列xn和hn的N1+N2-1（17）点圆周卷积结果。由于圆周长度足够大，没有发生环绕效应，因此圆周卷积的结果与线性卷积的结果相同。

**【例7-6】**

研究当 1≤N≤64 时，FFT 函数的执行时间。最后画出执行时间相对于N 的图。

代码：

clear;

Nmax=10;

Fft\_time=zeros(1,Nmax);

for n=1:1:Nmax;

x=rand(1:n);

t=clock;

fft(x);

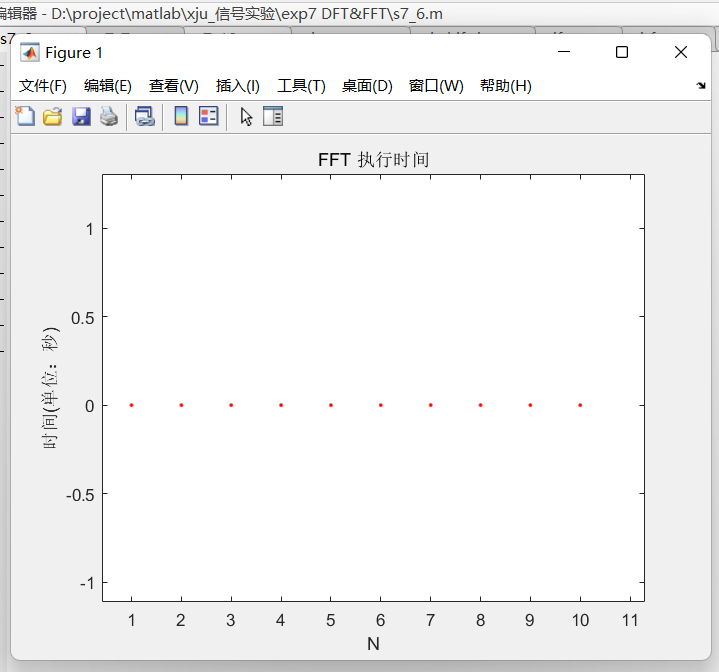
fft\_time(n)=etime(clock,t);

end

n=[1:1:Nmax];

plot(n,Fft\_time,'r.');

xlabel('N');ylabel('时间(单位：秒)');title('FFT 执行时间')



分析：

每次迭代生成一个长度为n的随机序列，然后对这个序列执行FFT，最后记录这个操作花费的时间。生成的图形应该展示FFT执行时间随着输入序列长度N的增加而变化的趋势。

观察图像时间趋势是一条近似直线，可能是这个大小范围的输入，FFT操作的执行时间主要由常数时间的开销决定，而这个开销与输入的大小无关。

**【例7-7】** 已知信号x(t)=0.15sin(2πf1t)+sin(2πf2t)-0.1sin(2πf3t)，f1=1Hz，f2=2Hz，f3=3Hz。取fs=32Hz作频谱分析。

代码：

clear all

fs=32;

N=fs;

n=0:N-1;

f1=1;f2=2;f3=3;

xn=0.15\*sin(2\*pi\*f1\*n/fs)+sin(2\*pi\*f2\*n/fs)-0.1\*sin(2\*pi\*f3\*n/fs);XK=fft(xn,N);

magXK=abs(XK);

phaXK=angle(XK);

subplot(1,2,1)

stem(n,xn,'.');

xlabel('n');ylabel('x(n)');

axis([0,32,-1.2,1.2]);

grid;

subplot(1,2,2)

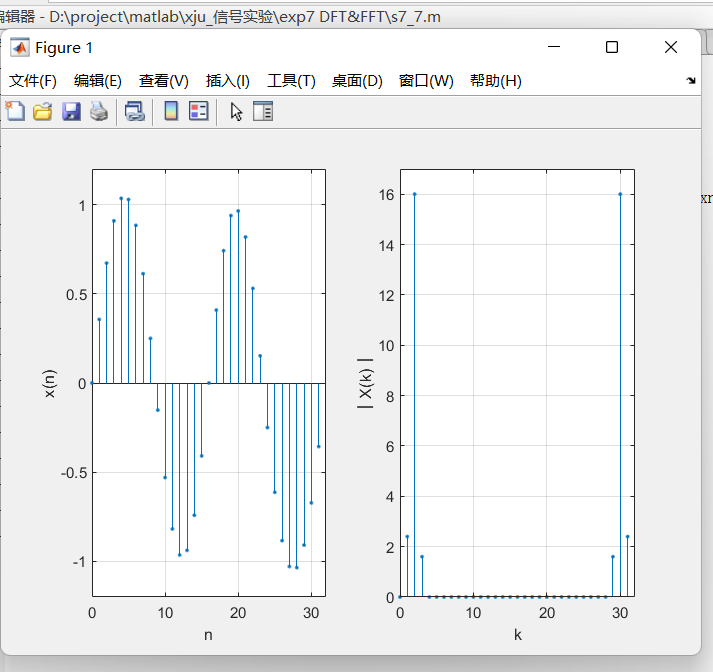
k=0:length(magXK)-1;

stem(k,magXK,'.');

xlabel('k');ylabel('│X(k)│');

axis([0,32,0,17]);

grid



分析：生成了一个由三个不同频率（1Hz、2Hz和3Hz）正弦波相加形成的信号的时间序列，并使用FFT将这个时间序列信号转换为频域。

在第一个子图中观察到时间序列的波形。这个波形是由三个不同频率（1Hz, 2Hz和3Hz）和不同幅度（0.15, 1和-0.1）的正弦波叠加而成。

在第二个子图中，观察到到的是信号的频谱。频谱图显示了各种频率成分的强度。在频率为1Hz、2Hz和3Hz处有显著的峰值，对应于在时间序列信号中加入的三个正弦波。同时，由于给这三个正弦波赋予了不同的振幅，所以这三个峰值的高度也不同。

**【例7-8】**

构造一个信号频率f = 1Hz，抽样频率fs = 32Hz的正弦序列，分别在整周期抽样间隔和非整周期抽样间隔两种情况下作谱分析。

代码：

clear all;fs=32;N=32;n=0:N-1;f=1;

d=(1/fs)\*0.05; %抽样间隔误差

Ts=1/fs; %整周期抽样间隔，Ts\*N=T

Ts1=1/fs-d; %非整周期抽样间隔，Ts\*N≠T

xn=sin(2\*pi\*f\*n\*Ts); %整周期抽样得到的序列xn

xn1=sin(2\*pi\*f\*n\*Ts1); %非整周期抽样得到的序列xn1

XK=fft(xn,N); %由整周期抽样序列xn的DFT得到的谱

magXK=abs(XK);phaXK=angle(XK);

XK1=fft(xn1,N); %由非整周期抽样序列xn1的DFT得到的谱

magXK1=abs(XK1);phaXK1=angle(XK1);

subplot(2,2,1);stem(n,xn,'.');

xlabel('n');ylabel('整周期抽样得到的序列xn');

axis([0,N,-1.2,1.2]);grid;

subplot(2,2,2)

k=0:length(magXK)-1;

stem(k,magXK,'.');

xlabel('k');ylabel('整周期抽样序列xn的幅值谱│X(k)│');

axis([0,N,0,(fs/2)+1]);grid

subplot(2,2,3);stem(n,xn1,'.');

xlabel('n');ylabel('非整周期抽样得到的序列xn1');

axis([0,N,-1.2,1.2]);grid;

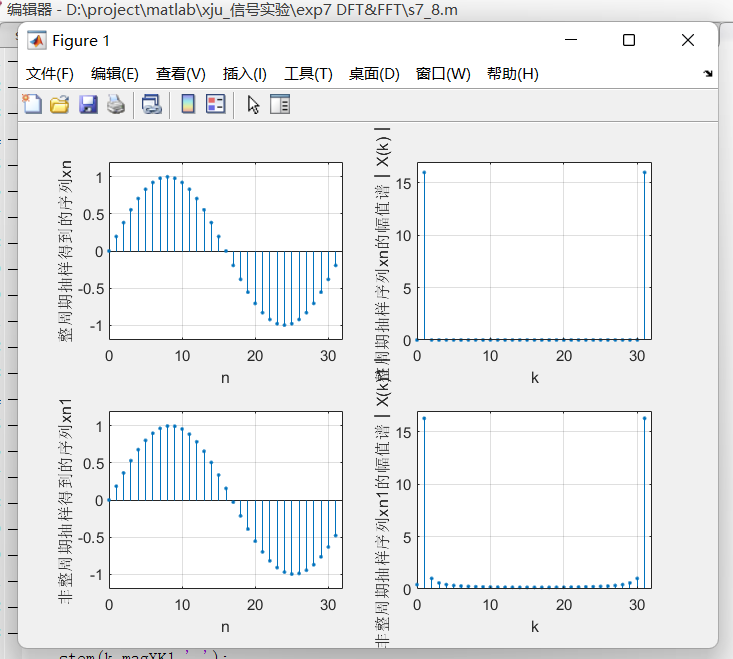
subplot(2,2,4)

k=0:length(magXK1)-1;

stem(k,magXK1,'.');

xlabel('k');ylabel('非整周期抽样序列xn1的幅值谱│X(k)1│');

axis([0,N,0,(fs/2)+1]);grid



分析：

一组是基于整周期抽样的信号，另一组是基于非整周期抽样的信号。然后，利用快速傅里叶变换将这两组信号转换到频域，并生成了这两组信号的时间序列和频谱图。

通过对比体现了抽样过程中周期选择的重要性。如果采样周期不是信号周期的整数倍，即使误差微小，也可能导致频谱分析的结果出现明显的偏差。

**【例7-9】**

用FFT（即快速卷积法）实现

,两序列的线卷积。

代码：

clear all;

n=[0:11];m=[0:5];

N1=length(n);N2=length(m);

xn=n;hn=ones(1,N2);

N=N1+N2-1;

XK=fft(xn,N);

HK=fft(hn,N);

YK=XK.\*HK;

yn=ifft(YK,N);

if all (imag(xn)==0)&(all(imag(hn)==0))

yn=real(yn);

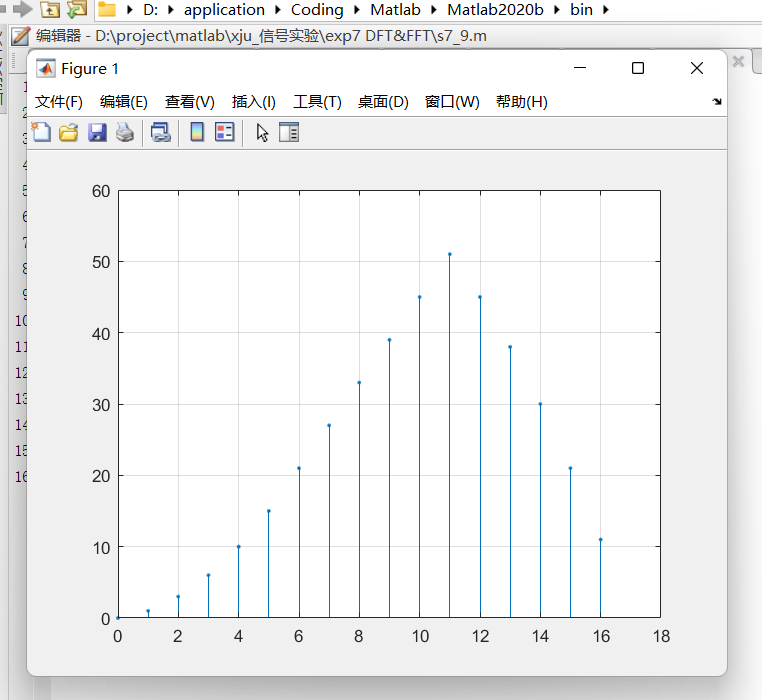
end

x=0:N-1;

stem(x,yn,'.');

axis([0,18,0,60]);

grid on;



分析：使用快速傅里叶变换和逆快速傅里叶变换来高效地计算两个信号的卷积。先对输入信号xn和hn进行了傅里叶变换，并将结果相乘，然后对乘积进行逆傅里叶变换以得到最终的卷积结果yn

**[例7-10]**

fft在信号分析中的应用

使用频域分析方法从受噪声污染的信号x(t)中鉴别出有用的信号。

代码：

t=0:0.001:1;

%采样周期为 0.001s,即采样频率为 1000Hz;

%产生受噪声污染的正正弦波信号；

x=sin(2\*pi\*100\*t)+sin(2\*pi\*200\*t)+rand(size(t));

subplot(2,1,1)

plot(x(1:100)); title('含噪原始信号')

%画出时域内的信号；

Y=fft(x,512);

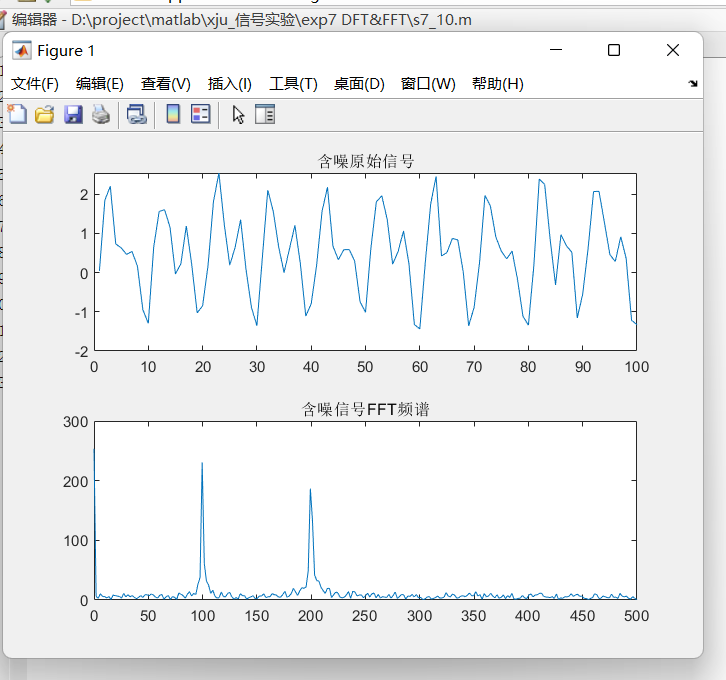
%对 X 进行 512 点的傅立叶变换；

f=1000\*(0:256)/512;

%设置频率轴（横轴）坐标，1000 为采样频率；

subplot(2,1,2)

plot(f,abs(Y(1:257))); title('含噪信号FFT频谱')



分析：在第一个子图中，观察原始信号在时域中的图形，可以看到信号波形由于噪声的存在而显得复杂且不规则。

然后，该代码对复合信号进行了512点的快速傅里叶变换（FFT）。这将信号从时域转换到频域，能够查看该信号的频率分量。

在第二个子图中，绘制的是FFT结果的幅度谱。可以看到在100Hz和200Hz处有两个显著的峰值，这正是原始信号中的两个正弦波成分。也可以看到在其他频率下存在一些较小的幅度，这是由随机噪声导致的。

2、分别计算16点序列的16点和32点fft,绘出幅度谱图形，并绘出该序列的DTFT幅频，观察分析结果。

代码：

clear;

% 计算16点序列

n1 = 0:15;

x1 = cos(5\*pi\*n1/16);

X1 = fft(x1, 16);

% 计算32点FFT

X2 = fft(x1, 32);

% DTFT

f = -0.5:0.001:0.5;

w = 2\*pi\*f;

n2 = 0:15;

X3 = x1 \* exp(-1j \* n2.' \* w);

figure;

subplot(3, 1, 1);

stem(abs(X1));

title('16点 FFT');

xlabel('频率 (Hz)');

subplot(3, 1, 2);

stem(abs(X2));

title('32点 FFT');

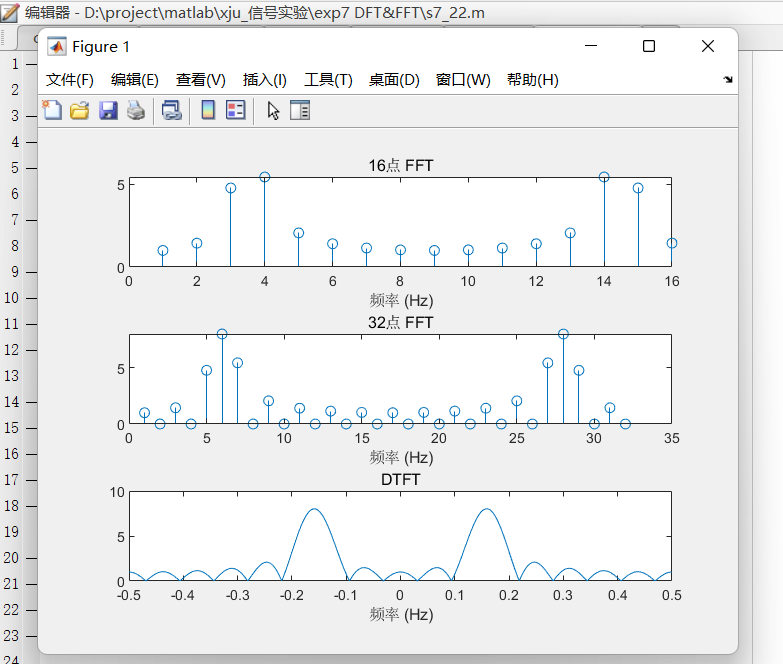
xlabel('频率 (Hz)');

subplot(3, 1, 3);

plot(f, abs(X3));

title('DTFT');

xlabel('频率 (Hz)');



第二个子图显示的是32点FFT的幅度谱。相比于16点FFT，32点FFT提供了更高的频率分辨率，可以在频率域中看到更多的细节。但是，原始信号的长度仍然是16点，所以多出来的点实际上是由零填充产生的，并没有增加原始信号的信息。

第三个子图显示的是DTFT的幅度谱。DTFT提供了连续的频率分辨率，可以看到更多的频谱信息。DTFT是对所有频率进行计算，所以可以看到整个频谱范围内的细节。

3、已知某序列在单位圆上的N=64等分样点的Z变换为

用N点IFFT程序计算，绘出。

代码：

clear;

N = 64;

k = 0:N-1;

Zk = exp(-1j\*2\*pi\*k/N);

Xk = 1 ./ (1 - 0.8 \* Zk);

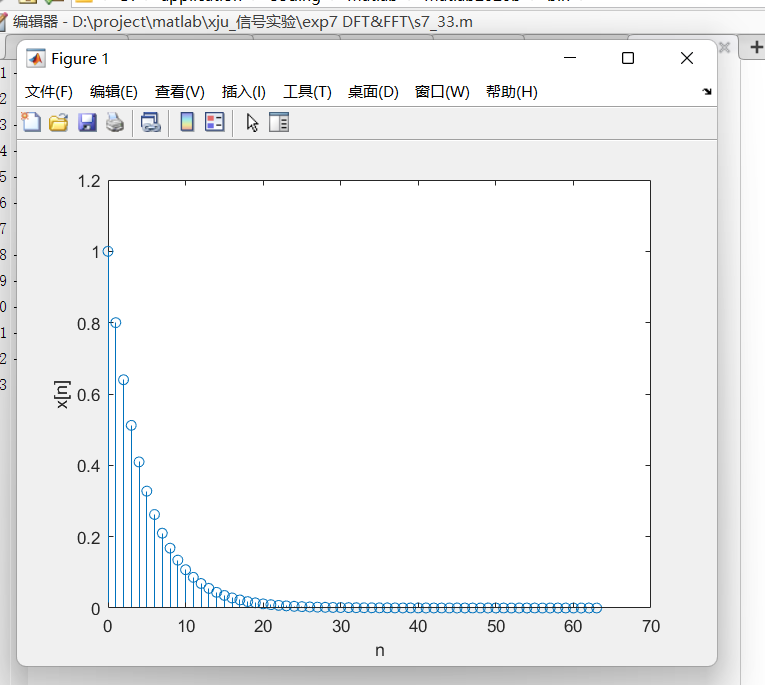
xn = ifft(Xk, N);

figure;

stem(0:N-1, real(xn));

xlabel('n');

ylabel('x[n]');



4、在实验四基础上继续完善（选做）

1）请查询资料，MATLAB实现：采集一段10秒电话拨号音(长按1或其他数字)，选择合适的采样频率抽样，转换为离散时间信号，存储在MATLAB中，并对其添加强度不同的随机噪声后播放出来，描述一下听见的效果如何？

2）选择合适的点数，对采样后的纯净信号和含噪信号分别求解FFT，绘制其幅频曲线，观察结果。

代码：

% 读取音频文件

[dialtone, fs] = audioread('bohaoyin.wav');

% 定义噪声强度

noise\_intensities = [0.02, 0.1, 0.5];

for i = 1:length(noise\_intensities)

% 添加噪声

noise = noise\_intensities(i) \* randn(size(dialtone));

dialtone\_noisy = dialtone + noise;

% 播放音频

soundsc(dialtone\_noisy, fs);

% 求解FFT并绘制幅频曲线

N = length(dialtone);

dialtone\_noisy\_fft = abs(fft(dialtone\_noisy, N));

f = (0:N-1)\*(fs/N); % 频率轴

figure;

plot(f, dialtone\_noisy\_fft);

title(['幅频曲线（噪声强度 = ' num2str(noise\_intensities(i)) '）']);

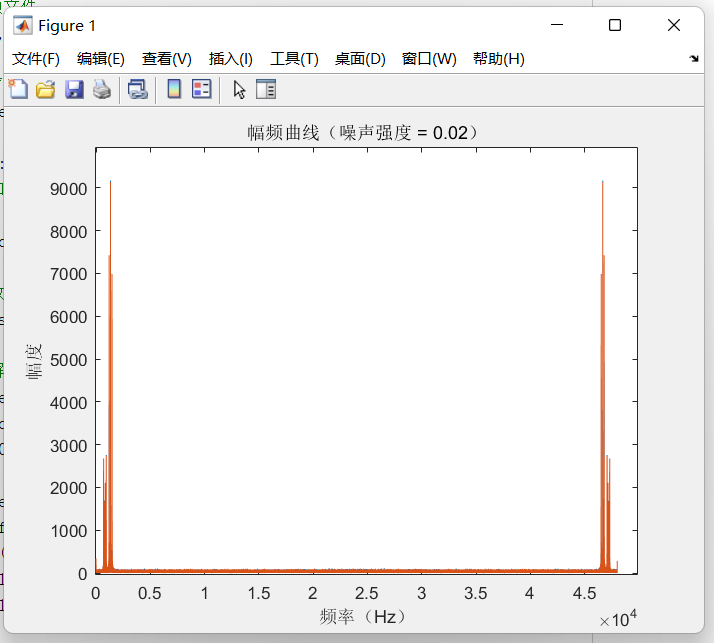
xlabel('频率（Hz）');

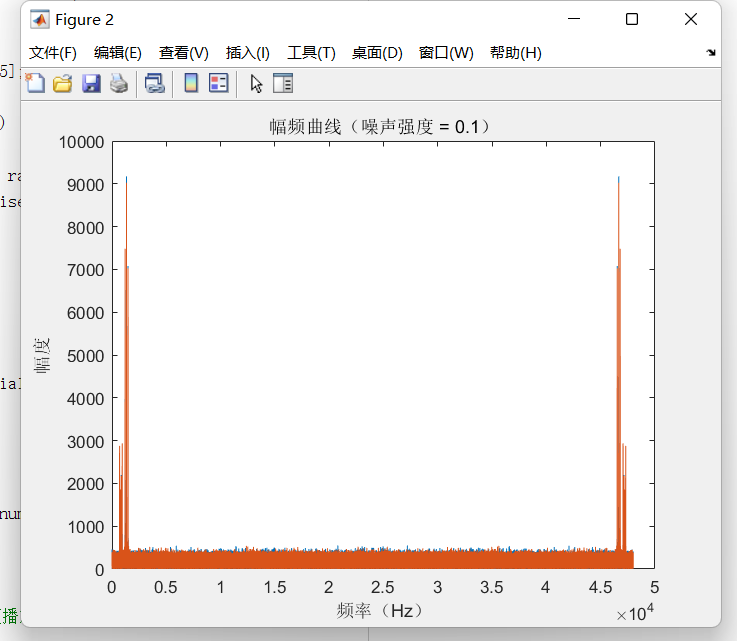
ylabel('幅度');

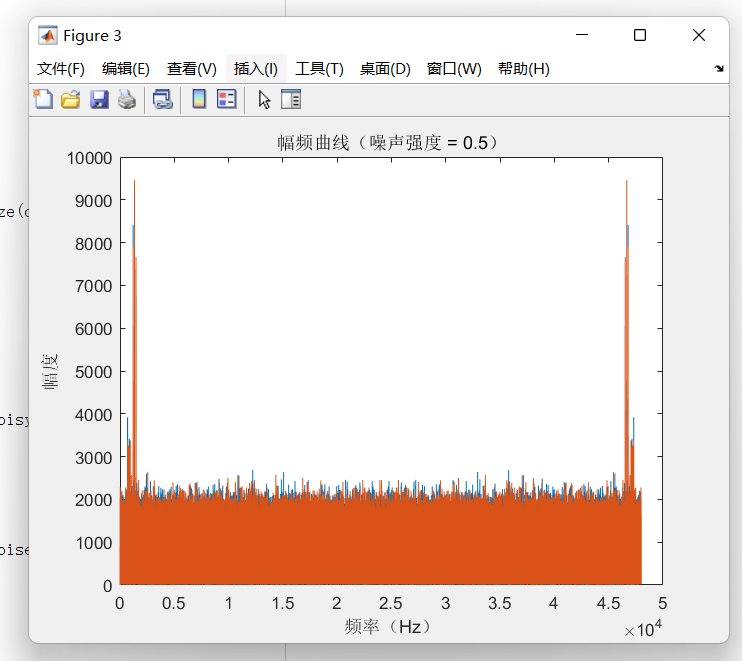
% 暂停，以便于我们听到每次的音频播放

pause(10);

end







噪声强度为0.02时可以听清拨号的声音，噪声的声音较小。到了0.1的强度时就可以明显听出噪声的声音，因为拨号声的频率较高还可以清晰听出拨号音。强度为0.5时，噪声过大，拨号声已经有部分已经被覆盖。

# 【思考题】

1. 分析ZT、DTFT和DFT之间的相互关系。

ZT是一种可对离散时间信号进行分析的方法，提供了在复平面的全局视图，可以用于分析离散时间系统的稳定性和频率响应等。

DTFT是将离散时间信号转换到频域的方法。与ZT不同，DTFT的频域是连续的，并且只在单位圆上定义。当ZT的Z在单位圆上时，ZT就退化为DTFT。

DFT则是DTFT的离散版本，它在时间和频率上都是离散的，只能提供信号在有限区间的频域信息。DFT可以看作是在DTFT的单位圆上采样的结果。

1. 分析比较N点DFT和FFT的复乘、复加运算量。

N点DFT的复杂度是O(N^2)，因为它需要进行N^2次复数乘法和复数加法。

FFT是一种高效的DFT算法，其复杂度是O(N logN)。FFT通过分治法将DFT分解为更小的DFT，大大减少了运算量。

1. 在什么条件下，两序列的圆周卷积和线性卷积相等？

当两个序列x(n)和h(n)的长度之和小于等于DFT的点数N时，它们的圆周卷积和线性卷积结果相等。

1. 利用FFT求解连续信号频谱需经过哪些环节/会遇到哪些问题？如何解决？

首先需要将连续信号离散化，涉及到抽样定理和抽样频率的选择。如果抽样频率不足，就可能导致混叠现象。

然后，由于FFT假设输入信号是周期的，如果输入信号不是完整的周期，可能会导致频谱泄漏。可以通过窗函数来减少泄漏。

最后，因为FFT的输入长度N必须是2的整数次幂，需要通过零填充来达到这个长度。

# 【实验总结】

在这次的实验中，研究了离散信号的傅立叶变换及其在信号处理中的重要应用。实验绍了ZT、DTFT和DFT之间的相互关系，这些理论知识是理解离散信号和系统的频域分析的基础。

深入学习了DTFT和DFT的概念及其计算方法，对离散时间信号和系统的频域分析有了更深入的理解。

了解了FFT是一种高效计算DFT的算法，其复杂度为O(N logN)，比直接计算DFT的O(N^2)复杂度要低得多。。

通过对合成信号和真实电话拨号音信号的FFT分析，我们实践了如何使用FFT对信号进行频域分析，并观察了不同强度噪声对信号频谱的影响。

这次实验提高了我对离散信号傅立叶变换理论的理解，提高了使用FFT进行信号处理的能力，为以后在实际工程问题中应用FFT打下了坚实的基础。