数字信号处理第一次实验报告

Part A：example

2\_1 该程序实现的是整体均值滤波。思路是产生50次噪声，分别加到原函数中，在把50次产生的函数加起来取平均，完成整体均值滤波。为了更好地观察实验结果，我把原函数也一并显示出来



如图，绿色点为原函数，蓝色的为滤波结果，红色为噪声。可以看到总体均值滤波后的函数与原函数很接近，失真度较少，能比较好地滤去高斯噪声。可以看到，程序的产生的噪声幅值仅在[-0.5,0.5]区间内，也就是总体均值滤波并未对脉冲噪声进行验证，可以预想，对于脉冲噪声，其平均到每一个点必然使得输出函数幅值增加，产生较大失真。

2\_2 改程序实现的是两个序列的卷积，并给出结果图。实现卷积的函数是 conv() 。我对该函数进行改进是之能实现 输入序列时给定零点位置，最后卷积输出也能确定零点位置。

代码修改如下

a = input('Type in the first sequence = ');

a0 = input('the zero point is at ');

b = input('Type in the second sequence = ');

b0 = input('the zero point is at ');

c = conv(a, b);

M = length(c)-1;

n = 0-a0-b0+2:1:M-a0-b0+2;

disp('output sequence =');disp(c)

stem(n,c)

xlabel('Time index n'); ylabel('Amplitude');

结果如下：

Type in the first sequence = [3 -2 4]

the zero point is at 2

Type in the second sequence = [4 2 -1]

the zero point is at 1

output sequence =

12 -2 9 10 -4



可以看到，第二点为零点，结果正确

2\_3 该程序为产生复指信号。其产生方法是给定指数和系数，通过调用 exp 函数生成。在课本中，给出了另外一种表示复指数函数的通式，是直接有其实部和虚部给出

x[n] = |A|eσncos（ωn+Φ）+j|A|eσnsin（ωn+Φ）

在此处我们可以验证通过 exp 求到的虚部和实部是否等于上式的表达

修改代码如下

% Program 2\_3

% Generation of complex exponential sequence

%

clear;

clc;

a = input('Type in real exponent = ');

b = input('Type in imaginary exponent = ');

c = a + b\*i;

K = input('Type in the gain constant = ');

N = input ('Type in length of sequence = ');

n = 1:N;

x = K\*exp(c\*n);%Generate the sequence

**realpart = abs(K)\*exp(a\*n).\*cos(b\*n+angle(K));**

**imagpart = abs(K)\*exp(a\*n).\*sin(b\*n+angle(K));**

stem(n,real(x));%Plot the real part

hold on;

**stem(n,realpart,'r');**

xlabel('Time index n');ylabel('Amplitude');

title('Real part');

disp('PRESS RETURN for imaginary part');

pause

stem(n,imag(x));%Plot the imaginary part

hold on;

**stem(n,imagpart,'r');**

xlabel('Time index n');ylabel('Amplitude');

title('Imaginary part');

结果：





可以看到，两种形式完全重叠，也就是相等，从而验证这两种表达是一致的。

2\_4 该程序实现的是产生一个实指数函数。

2\_5该程序实现能量有限的相关序列的计算。这里计算相关是利用卷积函数 conv() 来进行计算，因为相关和卷积运算只是相差了一个时间反转，因此只要在卷积前把其中一个序列先进性时间反转，即可由卷积计算得到相关计算。另外MATLAB中也有直接计算相关的函数xcorr(x)/xcorr(x,y) 。下面利用此程序验证相关用于计算延迟：

给出x=[0 0 1 2 3 4 5]和y=x[n+2]=[1 2 3 4 5 0 0]，结果如下，可以看到峰值在x=2处，准确地反映了延迟



下面验证加高斯噪声后x+d是否仍是在0点出取得最大值，结果如下，说明加噪后仍在0处有最大值



2\_6 改程序验证加噪的周期函数通过计算相关序列，可以看出周期函数的周期N。从输出的结果来看，确实验证了这个性质。峰值每隔一个周期出现一次，但是峰值的值却逐渐下降。开始我以为周期函数的相关序列应是周期出现，应该不会出现这种峰值下降的情况，仔细想想是因为此处只是截取了周期函数的某一段并且该段长度M>>周期T，也就是这样计算的周期函数的相关系列只是一个有限长函数的相关序列，其周期峰值的出现是因为每移动一个周期，两个序列就有x个周期重叠，并且当两者完全重叠，重叠的周期数就最大，因而有一个高峰，越往两边重叠的周期数越少，峰值就越来越小。

Part B：exercise

2\_1 该程序只是把一个有限长序列的共轭对称部分和共轭反对称部分求出来，并且显示的时候只是把其实部显示出来，并没有验证到由式2.28产生的是否满足共轭对称或共轭反对称，应该把虚部也显示出来。对于共轭对称部分，其实部应该是偶函数，虚部应为奇函数；对于共轭反对称部分，实部应为奇函数，虚部应为偶函数，修改后运行结果如下，容易验证例例2.8的结果

2\_2 该程序显示了几种常见的函数，代码值得我学习的是构造函数的参数事先定义好，然后在表达式中直接用这些参数这样以后修改起来就可以更方便。此外对于参数名的命名也做得很仔细。

2\_3

2\_4 在该题中要验证观察得到的周期是否如理论值。离散正弦函数的周期公式为2π\*r/ω，其中r为整数，可以在代码中加入求解理论周期的语句，并输出，与实际图像比对

修改代码为：

A=input('Type in amplitude of sequence=');

L=input('Type in length of sequence=');

omega=input('Type in angular frequency of sequence=');

phi=input('Type in phase of sequence=');

n=0:L;

x=A\*cos(omega\*n+phi);

stem(n,x);

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

title('sequence x[n]');

**for r = 1:100**

**if(2\*pi\*r/omega == round(2\*pi\*r/omega))**

**T = 2\*pi\*r/omega;**

**break;**

**end;**

**end;**

**display('the period in theory is');display(T);**

运行结果：由图可以看出周期为25，理论给出的结果也是25，说明理论与实际相符



the period in theory is

T =

25

2\_5 该程序中要对三个连续的余弦函数进行采样。采样间隔为0.1s 。由于不可能显示一个连续的图像，但又采样定理可知，对于这三个函数，采样间隔去0.01s足以确定这三个连续信号。为了更好地验证例2.15的结果，可以把3个图放在一个图中进行比较，这样代码就更加简洁

修改代码如下：

t=0:0.001:1;

g1=cos(6\*pi\*t);

g2=cos(14\*pi\*t);

g3=cos(26\*pi\*t);

figure;

**plot(t,g1,'--r',t,g2,'--g',t,g3,'--b');**

hold on;

%sampling

Ts=0.1;%sampling frequency

t1=0:Ts:1;

**t1 = t1';**

g1\_Ts=cos(6\*pi\*t1);

g2\_Ts=cos(14\*pi\*t1);

g3\_Ts=cos(26\*pi\*t1);

**G = [g1\_Ts,g2\_Ts,g3\_Ts];**

**stem(t1,G,'k');**

hold off;

结果如下，由图就可以验证例2.15：

2\_6 该程序通过输入一组数据然后利用循环来一次产生一组的正弦函数及其抽样，非常地方便，但由于是由循环统一产生，所有正弦函数的线型和颜色就一样，会导致不好观察结果。

2\_7&2\_8 两个程序都完成同一个题目，但2\_8更值得学习，它没有把输入写死在程序里面，而是通过外部输入。但是两者的横坐标的取值范围都是定死的，这部分也可以由外部输入。

2\_9 该程序基本实现了题目的要求，但是有一个细节，就是产生的高斯白噪声，应该改为**d=rand(N) – 0.5**

这样才是规范的产生高斯噪声的表达

2 \_ 10 这个程序有一个缺点，就是α定死在程序里面，这里应该从外部输入比较好，增加程序的灵活性；由于该题要求的是因果的系统，那么就要用到了阶跃信号，因此查看了如何产生阶跃信号：

方法一：（因果）

x=ones(1,N); N>0

Stem(n,x);

方法二：

n=N1:N2; N1<0<N2

x=[(n>=0)];

Stem(n,x)

带延时的阶跃信号：

n=N1:N2;

X=[(n-k)>=0];

PartC：homework

1代码：

% Homework 1\_1

% Genration of exponential signal

%

clc;

clear;

R = 50;

n = 0:R-1;

% z for the exponent of sequence 1/12+pi\*i/6 -1/12+pi\*i/6 1/12 2+pi\*i/6 pi\*i/6

z = input('Type in the exponent sequence = ');

for i = 1:length(z)

y = exp(z(i)\*n);

figure;

subplot(1,2,1);

stem(n,real(y));

xlabel('Time index n');ylabel('Amplitude');title('the real part of x[n]');

subplot(1,2,2);

stem(n,imag(y));

xlabel('Time index n');ylabel('Amplitude');title(['the imaginary part of x[n] z =' num2str(z(i))]);

end;

结果：











当z=j\*pi/6 时，周期为12

2 （1）代码：

% Homework 1\_2\_1

% Sampling of

%

clc;

clear;

R = 1;

Ts = 0.0125;%sampling frequency

n = 0:Ts:R;

A = 1.5;

omega = input('Type in the argular frequcy = ');

y = A\*sin(omega\*n);

stem(n,y);

xlabel('Time index n');ylabel('Amplitude'); title('Sampling of x[n]');

结果：



采样后，频率是20Hz，周期是4.

（2）代码：

% Homework 1\_2\_2

% Genration of sine function and frequency is 0.9

%

clc;

clear;

R = 40;

n = 0:R-1;

omega = input('Type in the argular frequcy = ');

y = sin(omega\*n);

stem(n,y);

xlabel('Time index n');ylabel('Amplitude'); title('x[n]');

结果:

其周期为10

3 代码：

% Homework 1\_3

% Generation of several common function

%

clc;

clear;

L = input('Type in length of sequelnce =');

n = 0:L-1;

%Generation of impulse function

figure;

Im = [(n==0)];

stem(n,Im);

xlabel('n');ylabel('amplitude');title('impulse funciton');

axis([-L+1,L-1,0,1]);

%Generation of step function

figure;

St = ones(1,L);

stem(n,St);

xlabel('n');ylabel('amplitude');title('step funciton');

axis([-L+1,L-1,0,1]);

%Generation of real exponent function

A\_r\_exp = input('Type in amplitude of real exponent =');

alpha\_r = input('Type in exponent of real exponent =');

Re = A\_r\_exp\*exp(alpha\_r\*n);

figure;

stem(n,Re);

xlabel('n');ylabel('amplitude');title('real exponent funciton');

%Generation of complex exponent function

A\_c\_exp = input('Type in amplitude of complex exponent =');

alpha\_c\_r = input('Type in real part exponent =');

alpha\_c\_i = input('Type in imaginary part exponent =');

alpha\_c = alpha\_c\_r+1i\*alpha\_c\_i;

Re = A\_r\_exp\*exp(alpha\_c\*n);

figure;

subplot(2,1,1);

stem(n,real(Re));

xlabel('n');ylabel('amplitude');title('real part of exponent funciton');

subplot(2,1,2);

stem(n,imag(Re));

xlabel('n');ylabel('amplitude');title('imaginary part of exponent funciton');

%%Generation of sine function

A\_sine=input('Type in amplitude of sine function=');

phi=input('Type in phase =');

omega=input('Type in angular frequency =');

Sin = A\_sine\*sin(omega\*n+phi);

figure;

stem(n,Sin);

xlabel('n');ylabel('amplitude');title('sine funciton');

结果：









