****

数字信号处理实验报告

**实验一、数字信号处理基础**

|  |  |
| --- | --- |
| **学院**  **班级**  **学生姓名**  **学生学号**  **指导教师**  **提交日期** | **电子与信息学院** |
| **2017级信工一班** |
| **苏佳棋** |
| **201730256051** |
| **余华** |
| **2019年 10 月19 日** |

**实验一：数字信号处理基础**

**一、实验目的**

1、熟悉并掌握离散系统的差分方程表示法；

2、加深对冲激响应和卷积分析方法的理解；

3、加深对离散信号的DFT的理解；

4、熟悉离散系统的频率响应分析方法、加深对零、极点分布的概念理解；

5、掌握Matlab常用函数的使用方法。

**二、实验原理**

**1. LTI系统的表示**

在时域中，离散时间系统对输入信号或者延迟信号进行运算处理，生成具有所需特性的输出信号，具体框图如下:

x[n] y[n]

Discrete-time

system

其输入、输出关系可用以下差分方程描述

输入信号分解为冲激信号,

记系统单位冲激响应 ,则系统响应为如下的卷积计算式：

当0,k=1,2,……N时，h[n]是有限长度的(n=1:M)，称系统为FIR 系统；反之，称系统为IIR系统。

**2. 离散傅里叶变换(DFT)的定义**

N点序列的DFT和IDFT变换定义式如下

利用旋转因子具有周期性，可以得到快速算法（FFT）。

**3. 离散时间系统的变换域分析方法**

离散系统的时域方程为

其变换域分析方法如下：

频域:

系统的频率响应为:

Z域:

系统的转移函数为:

分解因式:

其中和称为零、极点。



**三、预习要求**

1、在MATLAB 中，熟悉利用函数y = filter(b,a,x)实现差分方程的使用方法；

2、在MATLAB 中，熟悉用函数y=conv(x,h)计算卷积的使用方法；

3、在MATLAB 中，熟悉用函数y=impz(b,a)求系统冲激响应的使用方法。

4、在MATLAB 中，熟悉用函数y=stepz(b,a)求系统冲激响应的使用方法。

5、在MATLAB中，函数和计算N点序列的DFT正、反变换的使用方法。



6、在MATLAB中，熟悉函数tf2zp、zplane、freqz、residuez、zp2sos的使用。

1、**y = filter(b,a,x)**

是filter函数是一维的数字滤波器，b是差分方程中输入x的系数向量，a是差分方程中输出y的系数向量，则y是输入信号x通过系统的结果。

2、**y=conv(x,h)**

是用来计算卷积的一个函数，其中x是输入信号，h是系统函数，y是x和h卷积的结果

3、**y=impz(b,a,N)**

用来求解单位冲激响应，b是差分方程中输入x的系数向量，a是差分方程中输出y的系数向量，N是采样点数，返回系统的单位冲激响应h的0到N-1对应的值。若用impz(b,a,N)，则起到单位冲激响应可视化的效果，类似于stem(0:N-1,h)的效果

4、**y=stepz(b,a,N)**

用来求解单位阶跃响应，b是差分方程中输入x的系数向量，a是差分方程中输出y的系数向量，N是采样点数，返回系统的单位阶跃响应s的0到N-1对应的值。若用stepz(b,a,N)，则起到单位阶跃响应可视化的效果，类似于stem(0:N-1,s)的效果

5、**X=fft(x,N)**

x是输入信号，N是采样点的个数，求出的x的快速傅里叶变换。若要对快速傅里叶变换的结果进行观察，可以通过plot(n,abs(X))观察傅里叶变换的模的情况，通过plot(n,angle(X))观察傅里叶变换的相位的情况

**x=ifft(X,N)**

使用快速傅里叶变换算法计算 X 的逆离散傅里叶变换。x与 X 的大小相同。如果 Y 是向量，则 ifft(Y) 返回该向量的逆变换。如果 Y 是矩阵，则 ifft(Y) 返回该矩阵每一列的逆变换。如果 Y 是多维数组，则 ifft(Y) 将大小不等于 1 的第一个维度上的值视为向量，并返回每个向量的逆变换。

6、**tf2zp**

是将有理分式形式的传递函数转换为零极点形式的一个转换函数,用来求 num/den传函的极零点及增益

[Z,P,K] = tf2zp(NUM,DEN)这样的格式就可以求出零点(Z)、极点(P)和增益(K)

7、**zplane**

zplane(z, p)　绘制出列向量z中的零点（以符号“○”表示）和列向量p中的极点（以符号“×”表示）同时画出参考单位圆，并在多阶零点和极点的右上角标出其阶数。

zplane(B, A)　绘制出系统函数H(z)的零极点图。其中B和A为系统函数H(z) = B(z)/A(z)的分子和分母多项式系数向量。

8、**freqz**

常见用法为[H,w]=freqz(B,A,N)，B和A分别为离散系统的系统函数分子、分母多项式的系数向量，返回量H则包含了离散系统频响在 0~pi范围内N个频率等分点的值（其中N为正整数），w则包含了范围内N个频率等分点。调用默认的N时，其值是512。

9、**residuez**

常见用法为[r,p,c]=residuez(b,a),把b(z)/a(z)展开成部分分式的r、p、c数组，返回有理多项式

10、**zp2sos**把

系统零极点形式变为二阶分割形式 [SOS,G]=zp2sos(Z,P,K)

SOS是矩阵形式，G是增益

………………………………………………………………………………………………..

**四、实验内容**

**1. Matlab函数conv和filter的使用**

**1.1 实验要求**

**以下程序中分别使用conv和filter函数计算h和x的卷积y和y1，运行程序，并分析y和y1是否有差别，为什么要使用x[n]补零后的x1来产生y1；具体分析当h[n]有M个值，x[n]有M个值，使用filter完成卷积功能，需要如何补零？**

使用两个函数计算卷积得到的结果没有差别，使用y = filter(h,1,x)来计算卷积时，必须满足结果长度和第三个参数x的长度相同，把x的长度记作lx，h的长度记作lh，卷积结果的长度为lx+lh-1，所以要通过补零操作来使第三个参数x的长度和卷积结果的长度相等。当h[n]有M个值，x[n]有M个值时，卷积结果长度为2M-1，若要使用filter完成卷积功能，需要在第三个参数x后面补M-1个零

**实验代码**

clf;

h = [3 2 1 -2 1 0 -4 0 3]; %impulse response

x = [1 -2 3 -4 3 2 1]; %input sequence

y = conv(h,x);

n = 0:14;

subplot(2,1,1);

stem(n,y);

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

title('Output Obtained by Convolution'); grid;

x1 = [x zeros(1,8)];

y1 = filter(h,1,x1);

subplot(2,1,2);

stem(n,y1);

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

title('Output Generated by Filtering'); grid;

**1.2 实验结果与分析**



实验结果分析：从上面两图可以看出，当输入为x，系统函数为h时，使用y=conv(x,h) 和y = filter(h,1,x)都可以实现对x和h求卷积，但要注意运用y = filter(h,1,x)求卷积时必须满足结果长度和第三个参数x的长度相同。

**2.Matlab函数impz和stepz的使用**

**2.1 实验要求**

编制程序用impz和stepz分别求解下列两个系统的单位冲激响应和阶跃响应：

y[n] + 0.75 y[n-1] + 0.125 y[n-2] = x[n] – x[n-1]

y[n] = 0.25 \*(x[n-1] + x[n-2] + x[n-3] + x[n-4])

给出**理论计算结果**和**程序计算结果**并讨论。

对于第一个式子

单位冲激响应理论计算结果是：

单位阶跃响应理论计算结果是：

对于第二个式子

单位冲激响应理论计算结果是：

单位阶跃响应理论计算结果是：

**2.2 程序代码**

clf;

N=16;

n=0:15;

y1= [1,0.75,0.125];

x1= [1,-1];

subplot(4,2,1);

impz(x1,y1,N);

title('式1单位冲击响应程序计算结果')

grid;

h1=6\*(-0.5).^n.\*(n>=0)-5\*(-0.25).^n.\*(n>=0);

subplot(4,2,2);

stem(n,h1);

title('式1单位冲击响应理论计算结果')

grid;

y2= [1];

x2= [0,0.25,0.25,0.25,0.25];

subplot(4,2,3);

impz(x2,y2,N);

title('式2单位冲击响应程序计算结果')

grid;

h2=[0,0.25,0.25,0.25,0.25,zeros(1,11)];

subplot(4,2,4);

stem(n,h2);

title('式2单位冲击响应程序计算结果')

grid;

subplot(4,2,5);

stepz(x1,y1,N);

title('式1阶跃响应程序计算结果')

grid;

s1=2\*(-0.5).^n.\*(n>=0)-(-0.25).^n.\*(n>=0);

subplot(4,2,6);

stem(n,s1);

title('式1阶跃响应理论计算结果')

grid;

subplot(4,2,7);

stepz(x2,y2,N);

title('式2阶跃响应程序计算结果')

grid;

s2=0.25\*((n>=1)+(n>=2)+(n>=3)+(n>=4));

subplot(4,2,8);

stem(n,s2);

title('式2阶跃响应理论计算结果')

grid;

**2.3实验结果与分析**



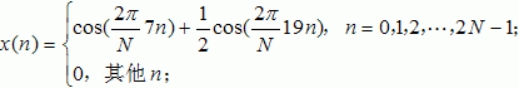
实验结果分析：从上图可以看出对于无论是单位冲激响应还是单位阶跃响应，

式子1和式子2的理论计算结果和程序计算结果一致，侧面反映了impz可以用来求解系统的单位冲激响应和stepz可以用来求解系统的阶跃响应

**3. 用N点DFT计算2N点实数的DFT**

**3.1 实验要求**

2N点实数序列



N=64。用一个64点的复数FFT程序，一次算出，并绘出 的图形。



**3.2 实现原理**

要实现用N点DFT计算2N点实数的DFT，令2n点的实序列为x[n], 0<=n<=2N-1;它的傅里叶变换为X[k],首先要将它分为两个长度为N的实序列，令x1[n]=x[2n],0<=n<=N-1; 它的傅里叶变换为X1[k], x2[n]=x[2n+1],0<=n<=N-1,它的傅里叶变换为X2[k]。

注意到上面最后一个表达式中第一个求和就是x1[n]的DFT X1[k], 第二个求和就是x2[n]的DFT X2[k],所以。从而实现用N点DFT计算2N点实数的DFT

**3.3 实现代码**

N=64;

n=0:N-1;

k=0:2\*N-1;

x=cos(14\*pi\*k/N)+0.5\*cos(38\*pi\*k/N);

x1=cos(14\*pi\*2\*n/N)+0.5\*cos(38\*pi\*2\*n/N);

x2=cos(14\*pi\*(2\*n+1)/N)+0.5\*cos(38\*pi\*(2\*n+1)/N);

X=fft(x,2\*N);

X1=fft(x1,N);

X2=fft(x2,N);

for m=1:N

H(m)=X1(m)+exp(-j\*2\*pi\*(m-1)/(2\*N))\*X2(m);

end

for m=(N+1):(2\*N)

H(m)=X1(m-N)+exp(-j\*2\*pi\*(m-1)/(2\*N))\*X2(m-N);

end

subplot(3,1,1);

stem(k,abs(X));

title('直接求的2N点DFT')

xlabel('k');

ylabel('|X|');

subplot(3,1,2);

stem(k,abs(H));

title('用N点求出的2N点DFT')

xlabel('k');

ylabel('|H|');

subplot(3,1,3);

stem(k,abs(H)-abs(X));

title('两种求法的差值')

xlabel('k');

ylabel('|H|-|X|');

**3.4实验结果与分析**



实验结果分析：从上图可以看出，直接求的2N点DFT和用N点求出的2N点DFT得到的图形基本一致，二者的差接近于零，可忽略不计。侧面说明可以通过N点DFT来计算一个实序列的2N点DFT，

**4. 用DFT验证Z变换**

**4.1 实验要求**

已知某序列x(n)的Z变换在单位圆上的N=64等分样点为：

。



用N点IFFT程序计算出并和理论求解的x(n)比较。



**4.2 实现原理**

理论计算得到的结果,0<=n<=64

要观察X[k]傅里叶逆变换得到的x[n],可以通过ifft函数，采用x=ifft(X,N);其中X是频域的函数，N是要计算傅里叶逆变换的点的个数，这里取64，x是傅里叶逆变换得到的x[n]

**4.3 实现代码**.

N=64;

k=0:N-1;

X=1./(1-0.8\*exp(-2\*j\*pi\*k/N));

x=ifft(X,N);

n=0:N-1;

x1=(0.8).^n.\*(n>=0);

subplot(3,1,1);

stem(n,x);

title('傅里叶逆变换得到的序列x[n]')

subplot(3,1,2);

stem(n,x1);

title('理论求的序列x[n]')

subplot(3,1,3);

stem(n,x1-x);

title('两种求法求出的序列x[n]的差')

**4.4实验结果与分析**



实验结果分析：从上图可以看出，理论计算得到的x[n]和对频域函数进行傅里叶逆变换得到的x[n]基本一致，他们的差值非常小，接近于零，可忽略不计。侧面说明我们可以通过对一个频域函数进行傅里叶逆变换得到它的时域的函数

**5 离散时间系统的变换域分析**

**5.1实验要求**

用matlab函数，求系统



的零、极点和幅度频率响应和相位响应。

**5.2 实现代码**

B=[0.0528,0.0797,0.1295,0.1295,0.797,0.0528];

A=[1.-1.8107,2.4947,-1.8801,0.9537,-0.2336];

subplot(3,1,1)

zplane(B,A)

title('系统的零极点图');

grid;

[h,w]=freqz(B,A,'whole');

subplot(3,1,2)

plot(w,abs(h));

title('幅度频率响应');

xlabel('w');

ylabel('|h|');

grid;

subplot(3,1,3)

plot(w,angle(h));

title('相位频率响应');

xlabel('w');

ylabel('p(h)');

grid;

**5.3实验结果与分析**



实验结果分析：zplane(B,A)画出来的零极点图经过验证与理论一致。

[h,w]=freqz(B,A,'whole')用来计算频率响应。'whole'是为了在整个单位圆上计算频率响应,对应的角频率范围是[0,2pi]。然后可以通过plot(w,abs(h))来画出幅频特性曲线，通过plot(w,angle(h))来画出相频特性曲线。

**6 附加应用题**

课前每人录制一段拼音字母“a、o、e”的声音 (采样率8kHz，单声道，16bit量化)，利用自相关法编程实现声音信号的基音周期估计，观察不同字母的基音变化，注意：语音信号需分帧处理，20ms一帧（160个样点），每一帧求一个周期。要求：掌握相关函数的计算原理，会通过matlab求互相关函数，并解读互相关函数峰值的物理意义。

**要实现这个功能，首先录制一段符合要求的音频。录制音频代码如下：**

clc;

clear all;

close all;

R=audiorecorder(8000,16,1)

disp('start speaking');

recordblocking(R,5);

disp('end of speak');

play(R)

myspeech=getaudiodata(R);

plot(myspeech);

audiowrite('C:\Users\26363\Desktop\sujiaqi.wav',myspeech,8000);

**思想：**然后利用自相关法编程实现声音信号的基音周期估计。先把刚刚录制的语音文件可视化进行观察，代码是stem(speech)，会发现它主要由三个部分组成，不难知道这三个部分分别对应’a’,’o’,’e’因为我们要处理的正是这三个部分，所以可以抛弃三部分的左边和右边，通过观察，我们可以知道只要截取speech(10000:34960)部分就可以覆盖’a’,’o’,’e’三个部分。因为题目要求语音信号需分帧处理，20ms一帧（160个样点）。所以将这24960个点分为156个小部分进行处理。对每个小部分进行一次求自相关，然后选择其中的自相关值最大点的位置，为ad1,接着选择最大点位置左边的自相关值第二大点的位置。两个位置的差理论上就是求的该部分语音的基波周期。

**自相关峰值的解读：**一个序列求自相关的表达式是，当l=0是自相关可以求得最大值，当一个周期序列无限长时，相差一个周期时，自相关同样会求得最大值，而实际中的序列都是有限长的，对于语音序列，当l=N或-N时（N为基波周期），自相关会取得第二大值。所以通过确定自相关最大值和第二大值的位置可以然后求差值可以确定语音的基波周期。

**代码**

[speech,fs]=audioread('C:\Users\26363\Desktop\sujiaqi.wav');

subplot(2,1,1);

stem(speech);

begin=10000;

n=156;

f1=160;

subplot(2,1,2);

for i=1:n

s=speech((begin+f1\*(i-1)):(begin +f1\*(i)));

x=xcorr(s);

stem(x);

[MAX,ad1]=max(x);

[MAX,ad2]=max(x(1:ad1-6));

d(i)=ad1-ad2;

end

subplot(3,1,3);

stem(d);

disp('a的基音周期');

T\_a=mode(d(10:35))

disp('a的基音频率');

F\_a=8000/T\_a

disp('o的基音周期');

T\_o=mode(d(75:95))

disp('o的基音频率');

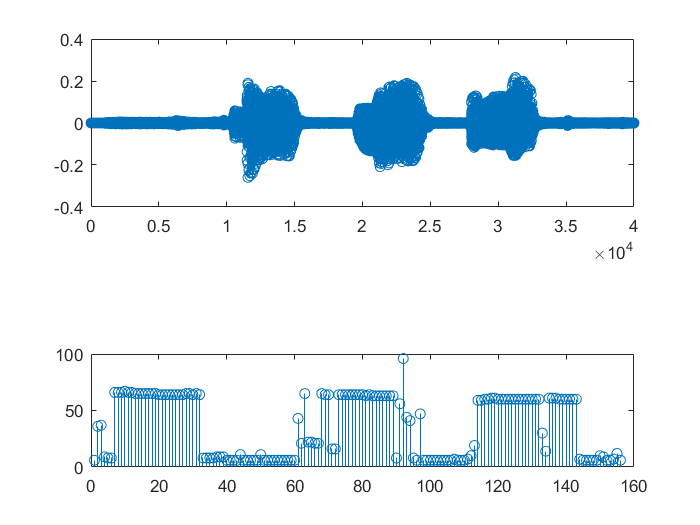
F\_o=8000/T\_o

disp('e的基音周期');

T\_e=mode(d(115:145))

disp('e的基音频率');

F\_e=8000/T\_e



**运行结果：**

a的基音周期

T\_a =

64

a的基音频率

F\_a =

125

o的基音周期

T\_o =

63

o的基音频率

F\_o =

126.9841

e的基音周期

T\_e =

60

e的基音频率

F\_e =

133.3333