北京理工大學
Beijing Institute of Technology

本科实验报告

实验名称：信号与信息处理实验Ⅱ

课程名称：	数字信号处理	实 验 时 间：	2018-04-19 8-10 节 2018-05-14 3-7 节
任课教师：	周治国	实 验 地 点：	理 B 404
实验教师：	何冰松	实 验 类 型：	■ 原理验证 □ 综合设计 □ 自主创新
学生姓名：	施念		
学号/班级：	1120161302/05011609	组　　号：	
学　　院：	信息与电子学院	同 组 搭 档：	
专　　业：	电子信息工程	成　　绩：	

信息与电子学院
SCHOOL OF INFORMATION AND ELECTRONICS

目录

实验 1 利用 DFT 分析信号频谱

一、实验目的

1. 加深对 DFT 原理的理解。
2. 应用 DFT 分析信号频谱。
3. 深刻理解利用 DFT 分析信号频谱的原理，分析现实过程现象及解决办法。

二、实验原理

1、DFT 和 DTFT 的关系

有限长序列 $x(n)$ 的离散时间傅里叶变换 $X(e^{j\omega})$ 在频率区间 $(0 \le \omega \le 2\pi)$ 的 N 个等分点 $\{x(0), x(1), \cdots x(k), \cdots x(N-1)\}$ 上的 N 个取样值可以由下式表示：

$$X(e^{j\omega})|_{\omega=2\pi k} = \sum_{k=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} = X(k) \quad 0 \le k \le N-1 \qquad (2-1)$$

由上式可知，序列 $x(n)$ 的 N 点 DFT $X(k)$，实际上就是 $x(n)$ 序列的 DTFT 在 N 个等间隔频率点 $\{X(0), X(1), \cdots X(k), \cdots X(N-1)\}$ 上样本 $X(k)$。

2、利用 DFT 求 DTFT

方法 1：由 $X(k)$ 恢复出 $X(e^{j\omega})$ 的方法如图 2.1 所示：

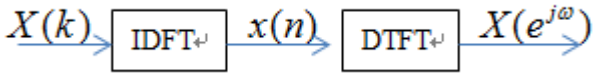


图 2.1. 由 N 点 DFT 恢复频谱 DTFT 的流程

由图 2.1 所示流程图可知：

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} = \frac{1}{N} \sum_{n=-\infty}^{\infty} \left[\sum_{k=0}^{\infty} X(k) W_N^{-kn} \right] e^{-j\omega n} \quad (2-2)$$

由式 2-2 可以得到

$$x(e^{j\omega}) = \sum_{k=1}^{N} X(k) \phi(\omega - \frac{2\pi k}{N}) \qquad (2-3)$$

其中 $\phi(x)$ 为内插函数

$$\phi(\omega)=\frac{\sin(N\omega/2)}{N\sin(\omega/2)}\bullet e^{-j\omega\frac{N-1}{2}} \qquad (2-4)$$

方法 2：然而在实际 MATLAB 计算中，上诉插值公式不见得是最好的方法。由于 DFT 是 DTFT 的取样值，其相邻的两个频率样本点的间距为$2\pi/N$，所以如果我们增加数据的长度 N，使得得到的 DFT 谱线就更加精细，其包络就越接近 DTFT 的结果，这样可以利用 DFT 来近似计算 DTFT。如果没有更多的数据，可以通过补零来增加数据长度。

3、利用 DFT 分析连续时间信号的频谱

采用计算机分析连续时间信号的频谱，第一步就是把连续时间信号离散化，这里需要进行连个操作：一是采样，二是截断。

对于连续非周期信号$x_\alpha(t)$，按采样间隔 T 进行采样，截取长度为 M，那么

$$X_\alpha(j\Omega)=\int_{-\infty}^{+\infty}x_\alpha(t)e^{-j\Omega t}dt=T\sum_{n=0}^{M-1}x_\alpha(nT)e^{-j\Omega nT} \qquad (2-5)$$

对$X_\alpha(j\Omega)$进行 N 点的频率采样，得到

$$X_\alpha(j\Omega)\big|_{\Omega=k\frac{2\pi}{NT}}=T\sum_{n=0}^{M-1}x_\alpha(nT)e^{-j\frac{2\pi}{N}kn}=TX_M(k) \qquad (2-6)$$

因此，可以将利用 DFT 分析连续非周期信号频谱的步骤归纳如下：

（1）确定时域采样间隔 T，得到离散序列$x(n)$；

（2）确定截取长度 M，得到 M 点离散序列$x_M(n)=x(n)w(n)$，这里的$w(n)$为窗函数。

（3）确定频域采样点数 N，要求$N\geq M$。

（4）利用 FFT 计算离散序列的 N 点 DFT，得到$X_M(k)$。

（5）根据式（2-6）由$X_M(k)$计算$X_\alpha(j\Omega)$采样点的近似值。

采用上诉方法计算的频谱，需要注意如下三点问题：

（1）频谱混叠。如果不满足采样定理的条件，频谱会很出现混叠误差。对于频谱无限宽的信号，应考虑覆盖大部分主要频率的范围。

（2）栅栏效应和频谱分辨率。使用 DFT 计算频谱，得到的结果只是 N 个频谱样本值，样本值之间的频谱是未知的，就像通过一个栅栏观察频谱，称为“栅栏效应”。频谱分辨率与记录长度成正比，提高频谱分辨率，就要增加记录时间。

（3）频谱泄露。对于信号截断会把窗函数的频谱会引入到信号频谱中，造成频谱泄露。解决这问题的主要办法是采用旁瓣小的窗函数，频谱泄露和窗函数均会引起误差。

因此，要合理选取采样间隔和截取长度，必要时还需考虑适当的窗。

对于连续周期信号，我们在采用计算机进行计算时，也总是要进行截断，序列总是有限长

的，仍然可以采用上诉方法近似计算。

4、可能用到 MATLAB 函数与代码

实验中的 DFT 运算可以采用 MATLAB 中提供的 FFT 来实现。
DTFT 可以利用 MATLAB 矩阵运算的方法进行计算。

三、实验内容

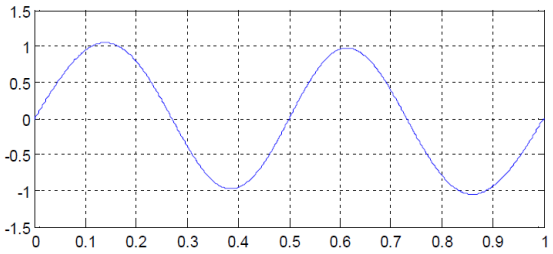
1. $x(n)=\{2,-1,1,1\}$，完成如下要求：
（1）计算其 DTFT，并画出$[-\pi,\pi]$区间的波形。
（2）计算 4 点 DFT，并把结果显示在（1）所画的图形中。
（3）对 $x(n)$ 补零，计算 64 点 DFT，并显示结果。
（4）根据实验结果，分析是否可以由 DFT 计算 DTFT，如果可以，请编程实现。

2. 考察序列

$$x(n)=\cos(0.48\pi n)+\cos(0.52\pi n)$$

（1）$0\le n\le 10$ 时，用 DFT 估计 $x(n)$ 的频谱；将 $x(n)$ 补零加长到长度为 100 点序列用 DFT 估计 $x(n)$ 的频谱。要求画出相应波形。
（2）$0\le n\le 100$ 时，用 DFT 估计 x(n)的频谱，并画出波形。
（3）根据实验结果，分析怎样提高频谱分辨率。

3. 已知信号 $x(t)=0.15\sin(2\pi f_1 t)+\sin(2\pi f_2 t)-0.1\sin(2\pi f_3 t)$ ，其中 $f_1=1Hz$，$f_2=2Hz$，$f_3=3Hz$。从 $x(t)$ 的表达式可以看出，它包含三个频率的正弦波，但是，从其时域波形（下图）来看，似乎是一个正弦信号，利用 DFT 做频谱分析，确定适合的参数，使得到的频谱的频率分辨率符合需要。



4. 利用 DFT 近似分析连续时间信号$x(t)=e^{-0.1t}u(t)$的频谱（幅度谱）。分析采用不同的采样间隔和截取长度进行计算的结果，并最终确定合适的参数。

四、实验结果和分析

1. 代码和分析

1.1 DTFT 及波形

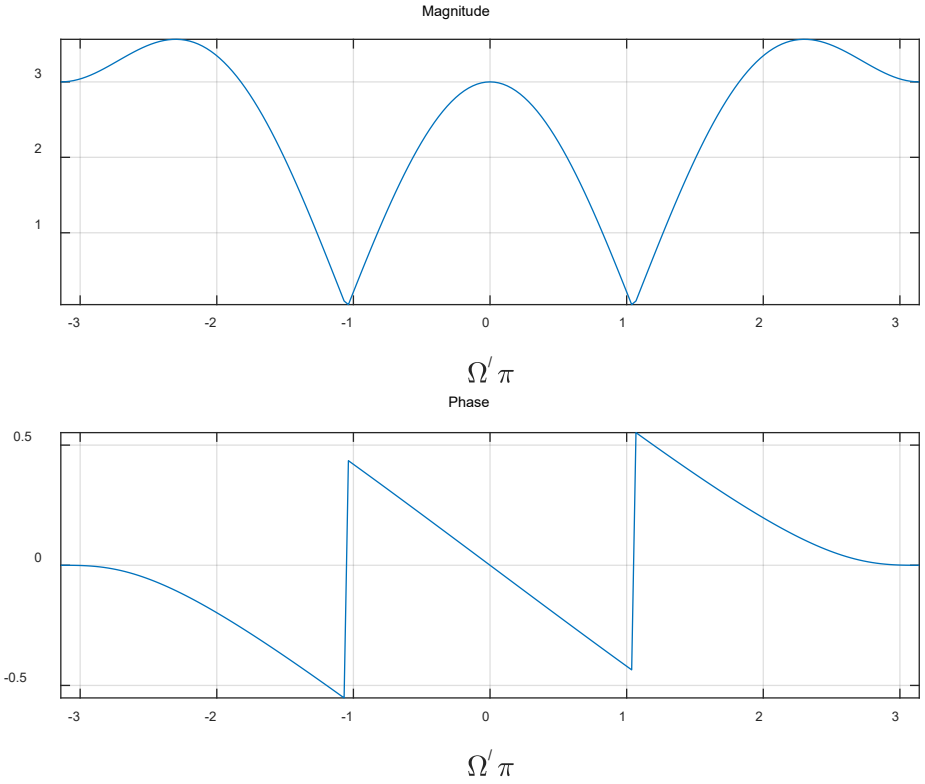
代码

```
x = [2 -1 1 1];
n = 0:3;
```

```
n1 = 0:63;
z = zeros(1,60);
```

```
x1 = [x,z];
y = fft(x);
y1 = fft(x1);
w = -pi : 0.01*pi: pi;
X = x* exp(-j*n'*w);
subplot(211);
plot(w,abs(X));
xlabel('\Omega/\pi');
title('Magnitude');
axis tight;
subplot(212);
plot(w,angle(X)/pi);
xlabel('\Omega/\pi');
title('Phase');
axis tight;
```

结果：

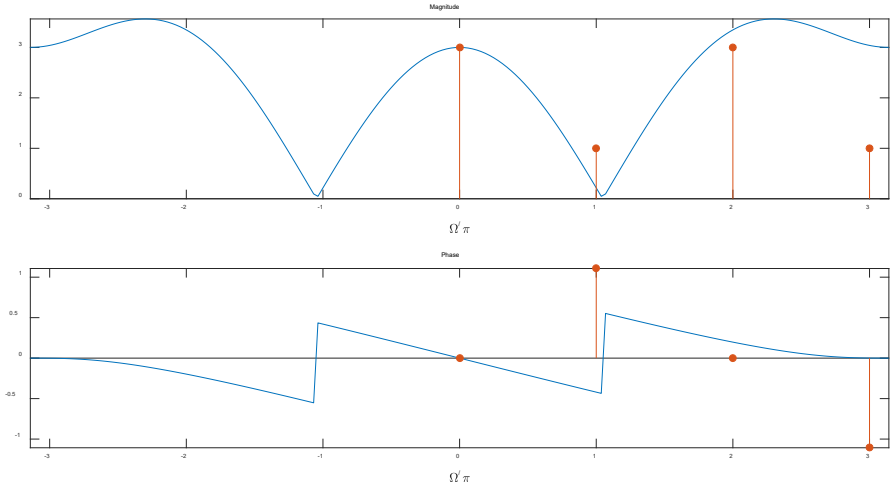


1.2 4 点 DFT 及波形

代码：

```
x = [2 -1 1 1];
n = 0:3;
n1 = 0:63;
z = zeros(1,60);
x1 = [x,z];
y = fft(x);
y1 = fft(x1);
w = -pi : 0.01*pi: pi;
X = x* exp(-j*n'*w);
subplot(211);
plot(w,abs(X));
xlabel('\Omega/\pi');
title('Magnitude');
hold on;
stem(n,y,'filled');
axis tight;
subplot(212);
plot(w,angle(X)/pi);
xlabel('\Omega/\pi');
title('Phase');
hold on;
stem(n,angle(y),'filled');
axis tight;
```

结果：

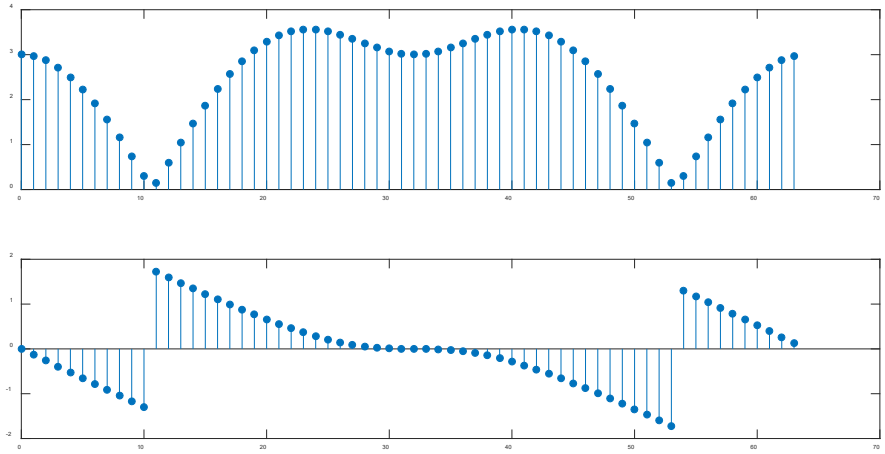


1.3 **补零、64 点 DFT 及波形**

代码：

```
x = [2 -1 1 1];
n = 0:3;
n1 = 0:63;
z = zeros(1,60);
x1 = [x,z];
y = fft(x);
y1 = fft(x1);
w = -pi : 0.01*pi: pi;
X = x* exp(-j*n'*w);
subplot(211);
stem(n1,abs(y1),'filled');
subplot(212);
stem(n1,angle(y1),'filled');
```

结果：



1.4

答： 由实验结果波形看出，序列补零后，长度越长，DFT 点数越多，其 DFT 越逼近其 DTFT 的连续波形。所以，可以推断出，当序列补零至无穷长时，可由其 DFT 无限逼近其 DTFT。

2. 代码和分析

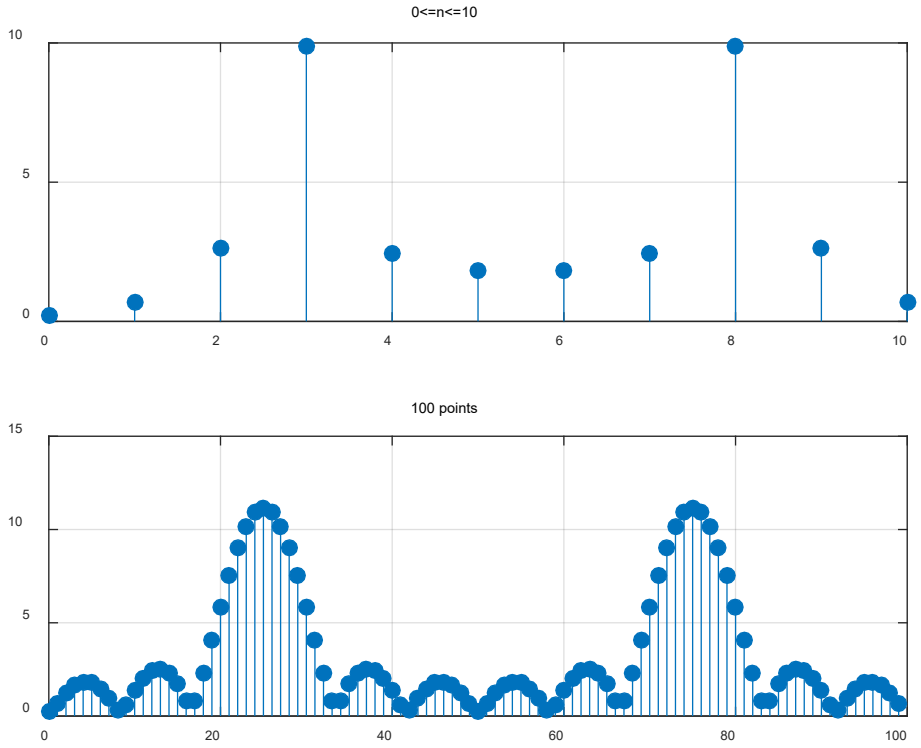
2.1 0≤n≤10

代码：

```
n = 0:10;
x = cos(0.48*pi*n)+cos(0.52*pi*n);
n1 = 0:99;
n2 = 0:100;
z = zeros(1,89);
x1 = [x,z];
x2 = cos(0.48*pi*n2)+cos(0.52*pi*n2);
y = fft(x);
y1 = fft(x1);
```

```
y2 = fft(x2);
subplot(211);
stem(n,abs(y),'filled');
title('0<=n<=10');
grid on;
subplot(212);
stem(n1,abs(y1),'filled');
title('100 points');
grid on;
```

结果：



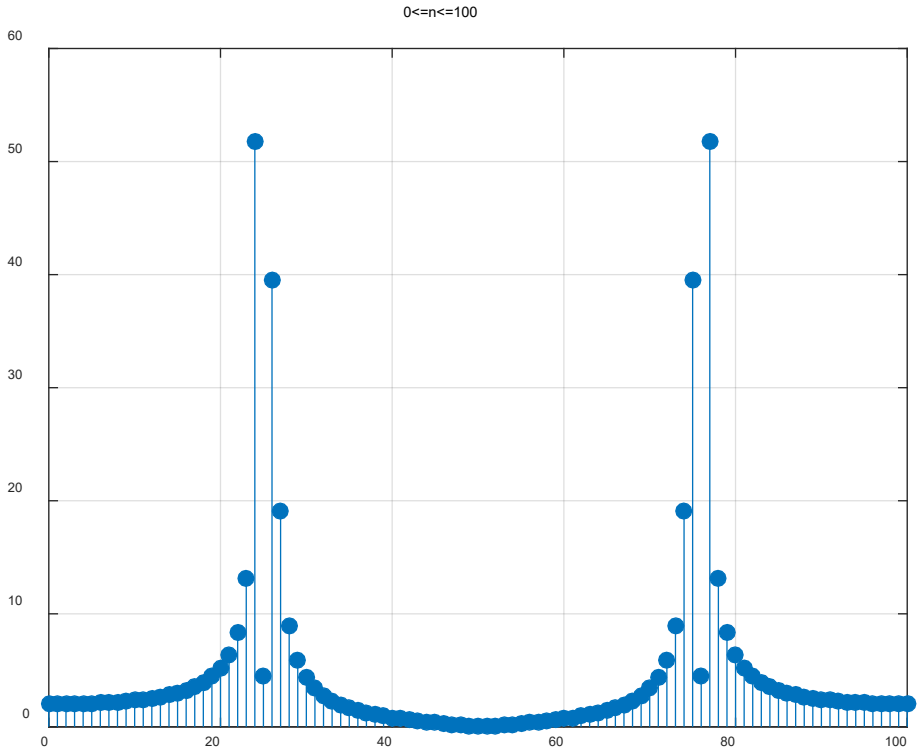
2.2 0≤n≤100

代码：

```
n = 0:10;
n2 = 0:100;
x2 = cos(0.48*pi*n2)+cos(0.52*pi*n2);
y2 = fft(x2);
```

```
stem(n2,abs(y2),'filled');
title('0<=n<=100');
grid on;
```

结果：



2.3

答：频谱分辨率，它与信号采集时间成反比，它的定义是在使用 DFT 时，在频 DFT 率轴上的所能得到的最小频率间隔，也就是与采样间隔成反比。所以我们可以通过增加时域内信号采样点数来增加分辨率。

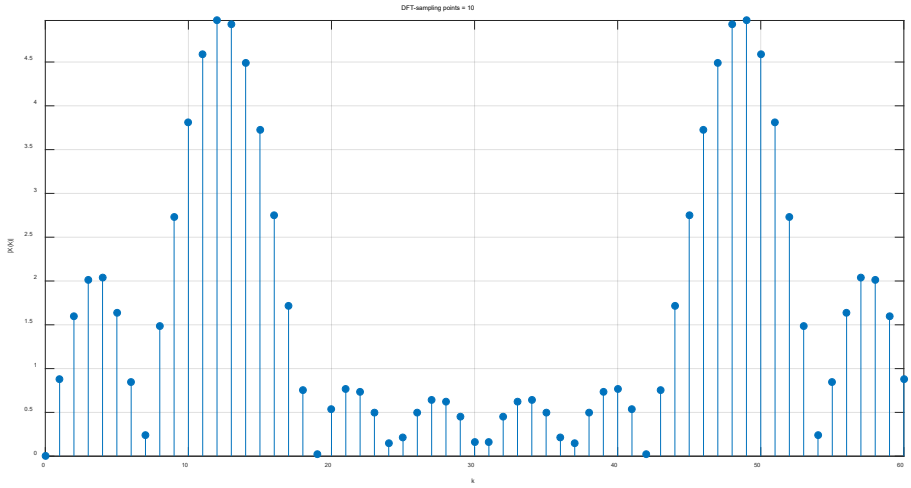
3. 代码和分析

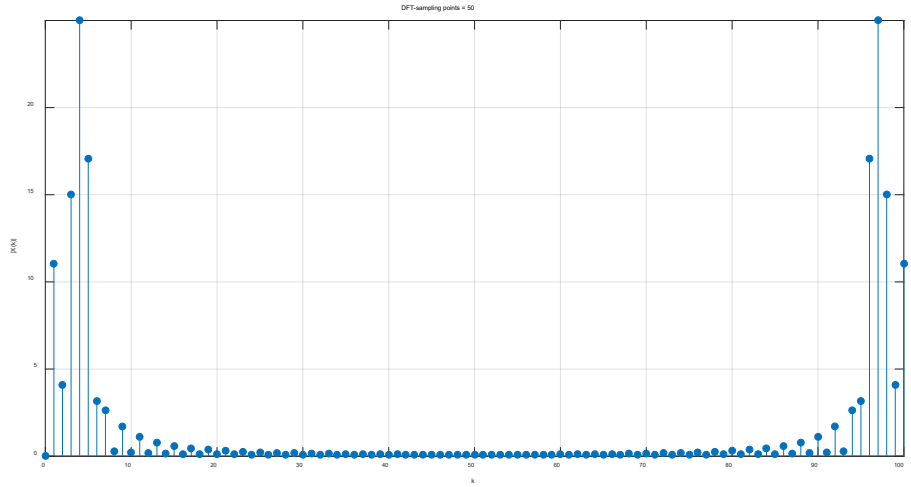
代码：

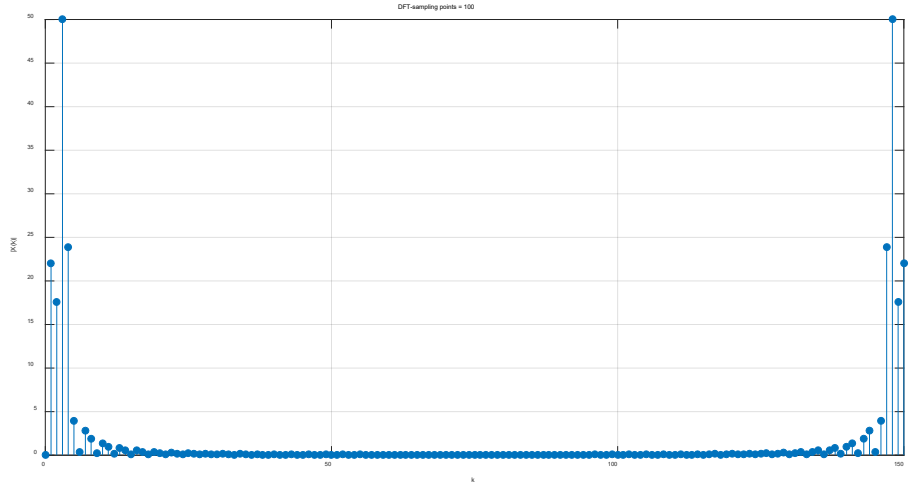
```
clear all;
N1 = input('sampling points=');
N2 = 50;
n = 0:1/N1:1;
x = 0.15 * sin(2*pi*n)+sin(4*pi*n)-
0.1*sin(6*pi*n);
z = zeros(1,N2);
x = [x,z];
```

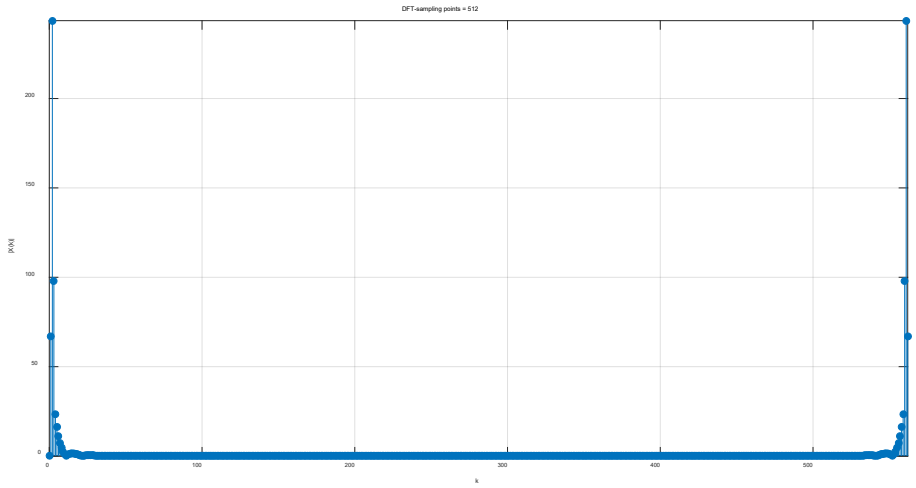
```
X = fft(x);
stem(0:(N1+N2),abs(X),'filled');
axis tight;
xlabel('k');
ylabel('|X(k)|');
title(['DFT-sampling points = '
num2str(N1)]);
grid on;
```

图形：


DFT-sampling points = 10
|X(k)|
k


DFT-sampling points = 50
|X(k)|
k


DFT-sampling points = 100
|X(k)|
k



分析：

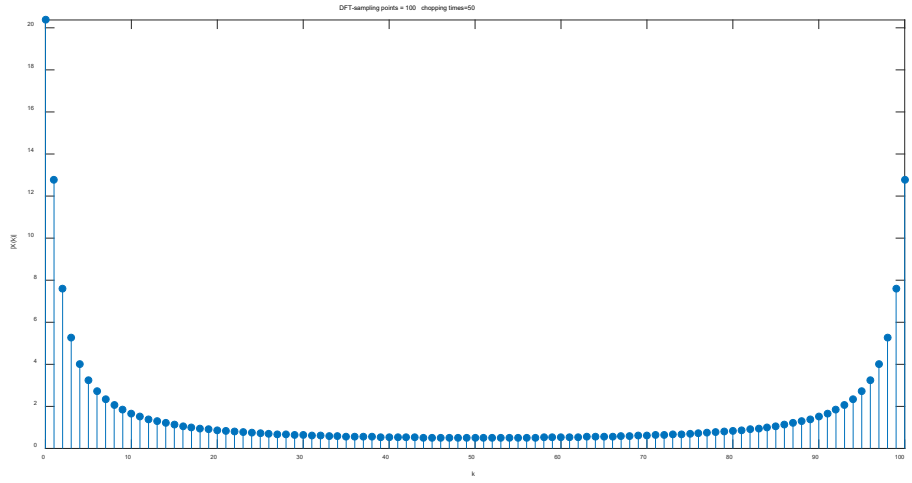
由上述几个不同采样点数来看，随着采样点数的增加，最后当采样点数为 512 的时候可以明显看出信号由三个频谱分量构成，其中一个频谱分量**幅值最大**，能量最高，其他两个频谱分量能量较小，这也是时域波形近似为一个正弦信号波形的原因。

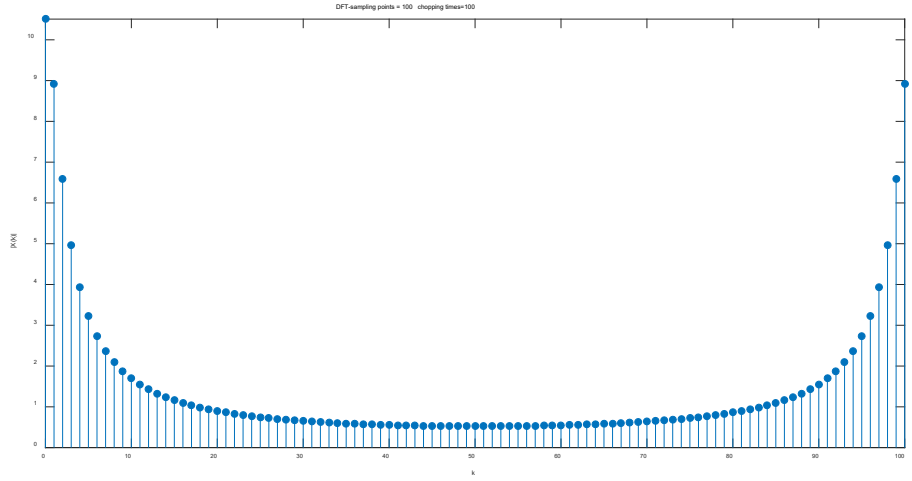
4. 代码和分析

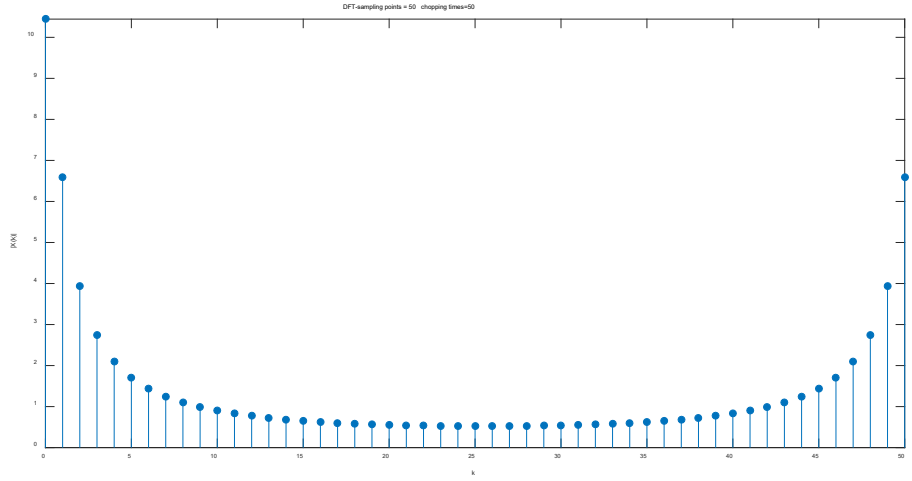
代码：

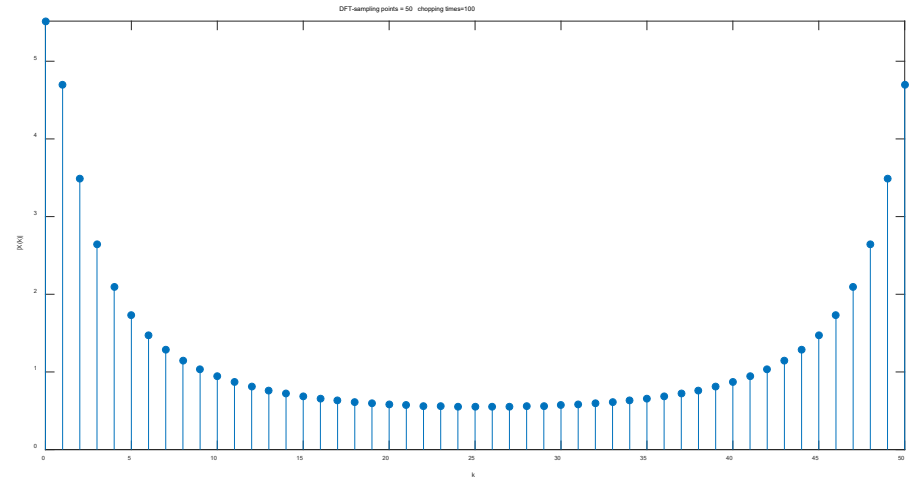
```
N1 = input('sampling points=');
T = input('chopping times=');
n = 0:T/N1:T;
x = exp(-0.1*n);
x = [x];
X = fft(x);
subplot(111)
stem(0:N1,abs(X),'filled');
axis tight;
xlabel('k');
ylabel('|X(k)|');
title(['DFT-sampling points = ' num2str(N1)
'     chopping times=' num2str(T)]);
```

结果：









分析：

由上面的结果不难看出，当取样点的间隔越小，即取样点的个数越多时，频谱越密集，也就是说用 DFT 得到的图形越接近频谱；当取样点间隔不变时，截取长度越大，频谱越密集，用 DFT 得到的图形越来越接近频谱。当取样点数 N=100 时可以得到比较精细的信号频谱波形，通过频谱可以看出，信号的能量主要集中在低频部分。

五、心得与体会

通过这次的实验，我对书本上 DFT 的原理确实有进一步的理解，相对于最一开始停留在最基本的运算上，现在已经可以进行一定的分析。对 DFT 和 DTFT 之间的关系也有了更多的认识：序列 x(n)的 N 点 DFT X(K)，实际上就是 x(n)序列的 DTFT 在 N 个等间隔频率点上的样本 X(K)。所以，我们可以通过增加数据的长度 N 或通过补零来使 DFT 更加接近 DTFT 的结果。这样就可以利用 DFT 近似计算 DTFT。

实验 2 利用 FFT 计算线性卷积

一、实验目的

1.掌握利用 FFT 计算线性卷积的原理及具体实现方法。
2.加深理解重叠相加法和重叠保留法。
3.考察利用 FFT 计算线性卷积各种方法的适用范围。

二、实验原理

1.线性卷积与圆周卷积

设 x(n)为 L 点序列，h(n)为 M 点序列，x(n)和 h(n)的线性卷积为

$$y_1(n)=x(n)*h(n)=\sum_{m=-\infty}^{\infty}x(m)h(n-m) \qquad (3\text{-}1)$$

$y_1(n)$ 的长度为 L+M-1

x(n)和 h(n)的圆周卷积为

$$y(n)=x(n)\odot h(n)=N\sum_{m=0}^{N-1}x(m)h(n-m)_N R_N(n) \qquad (3\text{-}2)$$

圆周卷积与线性卷积相等而不产生交叠的必要条件为

$N \geq L+M+1$ (3-3)

圆周卷积定理：根据 DFT 性质，x(n)和 h(n)的 N 点圆周卷积的 DFT 等于它们的 DFT 的乘积：

$$\mathrm{DFT}[x(n)\odot h(n)]=X(k)H(k) \qquad (3\text{-}4)$$

2.快速卷积

快速卷积发运用圆周卷积实现线性卷积，根据圆周卷积定理利用 FFT 算法实现圆周卷积。可将快速卷积运算的步骤归纳如下：

(1)必须选择$N \geq L+M-1$；为了能使用基-2 算法，要求$N=2^{\gamma}$。采用补零的办法使得 x(n)和 h(n)的长度均为 N。

(2)计算 x(n)和 h(n)的 N 点 FFT。

$$x(n)\xrightarrow{FFT}X(k)$$

$$h(n)\xrightarrow{FFT}H(k)$$

(3)组成乘积

$$Y(k)=X(k)H(k)$$

(4)利用 IFFT 计算 Y(k)的 IDFT，得到线性卷积 y(n)

$$Y(k) \xrightarrow{IFFT} y(n)$$

3.分段卷积

我们考察单位取样响应为 h(n)的线性系统，输入为 x(n)，输出为 y(n)，则

$$y(n) = x(n) * h(n)$$

当输入序列 x(n)极长时，如果要等 x(n)全部集齐时再开始进行卷积，会使输出有较大延时；如果序列太长，需要大量存储单元。为此，我们把 x(n)分段，为别求出每段的卷积，合在一起得到最后的总输出。这称为分段卷积。分段卷积可以细分为重叠保留法和重叠相加法。

重叠保留法：设 x(n)的长度为N_x，h(n)的长度为 M。把序列 x(n)分成多段 N 点序列$x_i(n)$，每段雨前一段重写 M-1 个样本。并在第一个输入段前面补 M-1 个零。计算每一段与 h(n)的圆周卷积，其结果中前 M-1 个不等与线性卷积，应当舍去，只保留后面 N-M+1 个正确的输出样本，把它们合起来得到总的输出。

利用 FFT 实现重叠保留法的步骤如下：

(1)在 x(n)前面填充 M-1 个零，扩大以后的序列为

$$\bar{x}(n) = \{0,0,\cdots 0, x(n)\}$$

(2)将 x(n)分为若干段 N 点子段，设 L=N-M+1 为每一段的有效长度，则第 i 段的数据为：

$$x_i(n) = \bar{x}(m) \qquad iL \le m \le iL + N - 1,\ i \ge 0, 0 \le n \le N - 1$$

(3)计算每一段与 h(n)的 N 点圆周卷积，利用 FFT 计算圆周卷积

$$x_i(n) \xrightarrow{FFT} X_i(k)$$

$$h(n) \xrightarrow{FFT} H(k)$$

$$Y_i(k) \xrightarrow{FFT} X_i(k)H(k)$$

$$Y_i(k) \xrightarrow{IFFT} y(n)$$

(4)舍去每一段卷积结果的前 M-1 个样本，连接剩下的样本得到卷积结果 y(n)。

重叠相加法：设 h(n)长度为 M，将信号 x(n)分解成长为 L 的子段。以$x_i(n)$表示没断信号，则：

$$x(n) = \sum_{i=0}^{\infty} x_i(n)$$

$$x_i(n) = \begin{cases} x(n + iL), & x(n) 0 \le n \le L - 1 \\ 0 & \text{其他} \end{cases}$$

$$x(n) * h(n) = \sum_{i=0}^{\infty} x_i(n) * h(n)$$

每一段卷积$y_i(n)$的长度为 L+M-1，所以在做求和时，相邻两段序列由 M-1 个样本重叠，即

前一段的最后 M-1 个样本和下一段前 M-1 个样本序列重叠，这个重叠部分相加，再与不重叠的部分共同组成 y(n)。
利用 FFT 实现重叠保留法的步骤如下：

(1)将 x(n)分为若干 L 点子段$x_i(n)$。

(2)计算每一段与 h(n)的卷积，根据快速卷积法利用 FFT 计算卷积。
(3)将各段相加，得到输出 y(n)。

$$y(n)=\sum_{i=0}^{\infty} y_i(n-iL)$$

4、可能得到的 MATLAB 函数
实验中 FFT 运算可采用 MATLAB 中提供的函数 fft 来实现。

三、实验内容

假设要计算序列 x(n)=u(n)-u(n-L),0≤n≤L 和 h(n)=cos(0.2πn),0≤n≤M 的线性卷积完成以下实验内容。
1.设 L=M,根据线性卷积的表达式和快速卷积的原理分别编程实现计算两个序列线性卷积的方法,比较当序列长度分别为 8,16,32,64,256,512,1024 时两种方法计算线性卷积所需时间。
2．当 L=2048 且 M=256 时比较直接计算线性卷积和快速卷积所需的时间,进一步考察当 L=4096 且 M=256 时两种算法所需的时间。
3. 编程实现利用重叠相加法计算两个序列的线性卷积,考察 L=2048 且 M=256 时计算线性卷积的时间,与 2 题的结果进行比较。
4. 编程实现利用重叠保留法计算两个序列的线性卷积,考察 L=2048 且 M=256 时计算线性卷积的时间,与 2 题的结果进行比较。

四、实验结果和分析

1. 代码和分析

代码：（100 次取平均值，去除开始前五次不稳定的值）

```
toc_1 = ones(1,100);
toc_2 = ones(1,100);
for L = [8 16 32 64 128 256 512 1024 2048 4096]
   M = L;
   for q = 1:100
      xn = linspace(1, 1, L);
      hn = cos(0.2*pi*[0:M-1]);
      yn1 = zeros(1, L+M-1);
      tic;
      for k = 0:L+M-1
         for n = max([0 k-M+1]):min([k L-1])
            yn1(k+1) = yn1(k+1) + xn(n+1) * hn(k-n+1);
         end;
      end;
      toc_1(q) = toc;
      tic;
      Xk = fft(xn, L+M-1);
      Hk = fft(hn, L+M-1);
      Yk = Xk.*Hk;
      yn2 = ifft(Yk);
      toc_2(q) = toc;
   end;
```

```
    disp(['L = ' num2str(L) '      M = ' num2str(M)]);
    disp(['线性卷积时间为：' num2str(mean(toc_1(6:100)))]);
    disp(['快速卷积时间为：' num2str(mean(toc_2(6:100)))]);
    disp(['线性卷积时间/快速卷积时间 = ' num2str(mean(toc_1(6:100))/mean(toc_2(6:100)))]);
end;
```

结果：

内容 / 序号	L（x(n)点数）	M（h(n)点数）	线性卷积时间/s	快速卷积时间/s	线性卷积时间/快速卷积时间
1	8	8	2.8315e-05	5.4127e-06	5.2313
2	16	16	5.1955e-05	6.0968e-06	8.5217
3	32	32	0.00012825	1.0078e-05	12.7264
4	64	64	0.00026399	2.3833e-05	11.0763
5	128	128	0.00079629	2.6922e-05	29.5772
6	256	256	0.0021506	7.1458e-05	30.0958
7	512	512	0.0064981	9.6769e-05	67.1512
8	1024	1024	0.022803	0.0002921	78.0665
9	2048	2048	0.0848	0.00022546	376.1206
10	4096	4096	0.31869	0.00131	243.2721

分析：

由以上结果可以看出，随着序列长度的不断增加，快速卷积和线性卷积花费的时间都不断增加，但在最后一列中我们可以看出，线性卷积花费时间和快速卷积花费时间比值越来越大，这说明快速卷积的速度越来越快，即越来越有效。

需要注意的是，当点数增加到 4096 的时候，相对 2048，快速卷积的效率有所下降，思考后，我认为原因应该是快速卷积时间和线性卷积时间都较长，考虑到计算机线程和进程以及后台应用，认为这在个人 PC 中用 MATLAB 计算中是正常的现象

2. 代码和分析

代码：（100 次取平均值，去除开始前五次不稳定的值）

```
%1
L = input('L = ');
M = input('M = ');
toc_1 = ones(1,100);
toc_2 = ones(1,100);

for q = 1:100
    xn = linspace(1, 1, L);
    hn = cos(0.2*pi*[0:M-1]);
    yn1 = zeros(1, L+M-1);
    tic;
    for k = 0:L+M-1
        for n = max([0 k-M+1]):min([k L-1])
            yn1(k+1) = yn1(k+1) + xn(n+1) * hn(k-n+1);
        end;
    end;
    toc_1(q) = toc;
    tic;
    Xk = fft(xn, L+M-1);
```

```
    Hk = fft(hn, L+M-1);
    Yk = Xk.*Hk;
    yn2 = ifft(Yk);
    toc_2(q) = toc;
end;
disp(['L = ' num2str(L) '    M = '
num2str(M)]);
disp(['线性卷积时间为：'
num2str(mean(toc_1(6:100)))]);
disp(['快速卷积时间为：'
num2str(mean(toc_2(6:100)))]);
disp(['线性卷积时间/快速卷积时间 = '
num2str(mean(toc_1(6:100))/mean(toc_2(6:
100)))]);
```

结果：

L = 2048　M = 256
线性卷积时间为：0.013373
快速卷积时间为：0.00020682
线性卷积时间/快速卷积时间 = 64.6634

L = 4096　M = 256
线性卷积时间为：0.025924
快速卷积时间为：0.00081234
线性卷积时间/快速卷积时间 = 31.9126

分析：由上面计算时间可以看出快速卷积时间比线性卷积时间快。值随着 L 的变化而不同。

3. 代码和分析

代码：（100 次取平均值，去除开始前五次不稳定的值）

```
%3
L = input('L = ');
M = input('M = ');
toc_1 = ones(1,100);
toc_2 = ones(1,100);

for k = 1:100
    xn = linspace(1, 1, L);
    hn = cos(0.2*pi*[0:M-1]);
    yn = []

    tic;
    Xk = fft(xn, L+M-1);
    Hk = fft(hn, L+M-1);
    Yk = Xk.*Hk;
    yn2 = ifft(Yk);
    toc_1(k) = toc;

    tic;
    x1n = [linspace(0,0,M-1) xn];
    Hk = fft(hn, 2*M);
    iend = floor(length(x1n)/(M+1))+1;
    for i = 0:iend
        xin = x1n(i*(M+1)+1 :
min([length(x1n) i*(M+1)+2*M]));
        Xik = fft(xin, 2*M);
        Yik = Xik.*Hk;
        yin = ifft(Yik);
        yn = [yn yin(M:2*M)];
    end;
    yn = yn(1:L+M-1);
    toc_2(k) = toc;
end;

disp(['L = ' num2str(L) '    M = '
num2str(M)]);
```

```
disp(['快速卷积时间为：'
num2str(mean(toc_1(6:100)))]);
disp(['重叠相加法卷积时间为：'
num2str(mean(toc_2(6:100)))]);
```

```
disp(['重叠相加法卷积时间/快速卷积时间
= '
num2str(mean(toc_1(6:100))\mean(toc_2(6:
100)))]);
```

结果：

L = 2048　　M = 256
快速卷积时间为：0.00025137
重叠相加法卷积时间为：0.00040132
重叠相加法卷积时间/快速卷积时间 = 1.5965

分析：

内容 / 序号	L（x(n)点数）	M（h(n)点数）	线性卷积时间	重叠相加法卷积时间	比较
1	2048	256	0.013373	0.00040132	重叠相加法更快
重叠相加法卷积时间/快速卷积时间 = 1.5965 线性卷积时间/快速卷积时间 = 64.6634 **重叠相加法速度是直接进行线性卷积速度的几十倍！**					

4. 代码和分析（100 次取平均值，去除开始前五次不稳定的值）

代码：

```
L = input('L = ');
M = input('M = ');
toc_1 = ones(1,100);
toc_2 = ones(1,100);


for k = 1:100
    xn = linspace(1, 1, L);
    hn = cos(0.2*pi*[0:M-1]);
    yn = [];

    tic;
    Xk = fft(xn, L+M-1);
    Hk = fft(hn, L+M-1);
    Yk = Xk.*Hk;
    yn2 = ifft(Yk);
    toc_1(k) = toc;

    tic;
    Hk = fft(hn, 2*M);
    iend = floor(length(xn)/M);
    i = 0;
    xin = xn(i*(M+1)+1 : min([length(xn)
i*(M+1)+2*M]));
    Xik = fft(xin, 2*M);
    Yik = Xik.*Hk;
    yin = ifft(Yik);
    yn = yin;
    for i = 1:iend
        xin = xn(i*(M+1)+1 :
min([length(xn) i*(M+1)+2*M]));
        Xik = fft(xin, 2*M);
        Yik = Xik.*Hk;
        yin = ifft(Yik);
        yn = [yn(1:(length(yn)-M+1))
yn((length(yn)-M+2):length(yn))+yin(1:M-1)
yin(M:2*M-1)];
    end;
    yn = yn(1:L+M-1);
    toc_2(k) = toc;
end;
```

```
disp(['L = ' num2str(L) '      M = '
num2str(M)]);
disp(['快速卷积时间为：'
num2str(mean(toc_1(6:100)))]);
disp(['重叠保留法卷积时间为：'
num2str(mean(toc_2(6:100)))]);
```

```
disp(['重叠保留法卷积时间/快速卷积时间
= '
num2str(mean(toc_1(6:100))\mean(toc_2(6:
100)))]);
```

结果：

L = 2048　　M = 256

快速卷积时间为：0.0001757

重叠保留法卷积时间为：0.00026916

重叠保留法卷积时间/快速卷积时间 = 1.5319

分析：

内容 / 序号	L（x(n)点数）	M（h(n)点数）	线性卷积时间	重叠保留法卷积时间	比较
1	2048	256	0.013373	0.00026916	重叠保留法更快
重叠相加法卷积时间/快速卷积时间 = 1.5965 重叠保留法卷积时间/快速卷积时间 = 1.5319 线性卷积时间/快速卷积时间 = 64.6634 **重叠相加法和重叠保留法速度是直接进行线性卷积速度的几十倍！**					

五、心得与体会

本次实验要求我们掌握利用 FFT 计算线性卷积的原理及具体实现方法，通过实验加深理解重叠相加法和重叠保留法并考察利用 FFT 计算线性卷积各种方法的适用范围。

本次实验让我切实看到了 FFT 算法的高效性及对于庞大的数据的处理实时性，同时对重叠保留法、重叠相加法也有了更深的认识，对以后的实验有了很好的理论基础，受益颇多。

实验 3　IIR 数字滤波器设计

一、实验目的

1. 掌握利用脉冲响应不变法和双线性变换法设计 IIR 数字滤波器的原理及具体方法。
2. 加深理解数字滤波器和模拟滤波器之间的技术指标转化。
3. 掌握脉冲响应不变法和双线性变换法设计 IIR 数字滤波器的优缺点及适用范围。

三、实验内容

1. 设采样频率为 fs =4kHz，采用脉冲响应不变法设计一个三阶巴特沃斯数字低通滤波器，其 3dB 截止频率为 fc=1kHz。
2. 设采样频率为 fs=10kHz，设计数字低通滤波器，满足如下指标

通带截止频率：fp=1kHz，通带波动：Rp=1dB

阻带截止频率：fst=1.5kHz，阻带衰减：As=15dB

要求分别采用巴特沃斯、切比雪夫 I 型、切比雪夫 II 型和椭圆模拟原型滤波器及脉冲响应不变法进行设计。结合实验结果，分别讨论采用上述设计的数字滤波器是否都能满足给定指标要求，分析脉冲响应不变法设计 IIR 数字滤波器的优缺点及适用范围。

四、实验结果和分析

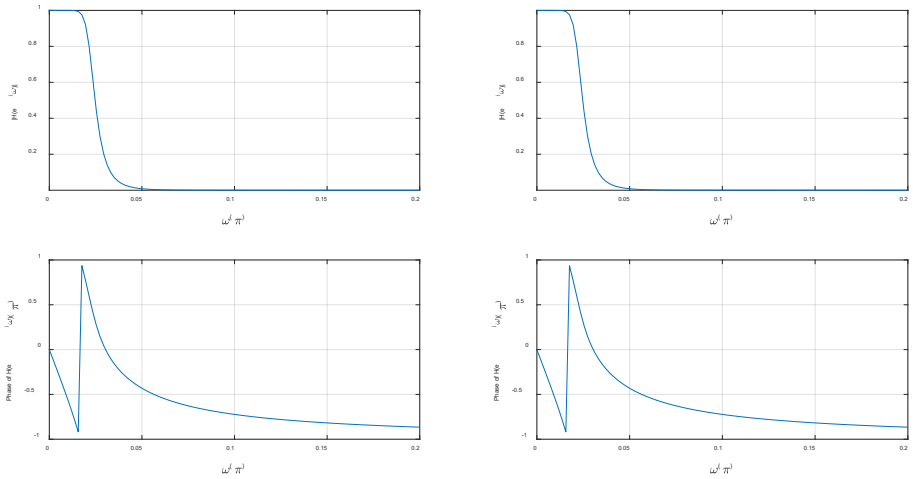
1. 代码和分析

代码：

```
Fs = 10e4;
fp = 1000;
fs = 1500;
Rp = 1;
As = 15;

Wp = fp*2*pi;
Ws = fs*2*pi;
[N,Wc] = buttord(Wp,Ws,Rp,As,'s');
[b,a]=butter(N,Wc,'s');

[b1,a1]=impinvar(b,a,Fs);
[b2,a2]=bilinear(b,a,Fs);
w=0:0.0001*pi:pi;
[H1,w]=freqz(b1,a1);
[H2,w]=freqz(b2,a2);
subplot(221);
plot(w/pi,abs(H1));
grid
on;xlabel('\omega(\pi)');ylabel('|H(e^j^\omega)|');
axis tight;
xlim([0 0.2]);
title('巴特沃斯脉冲响应不变');

subplot(223);
plot(w/pi,angle(H2)/pi);
grid on;xlabel('\omega(\pi)');ylabel('Phase of
H(e^j^\omega)(\pi)');
xlim([0 0.2]);

subplot(222);
plot(w/pi,abs(H2));
grid
on;xlabel('\omega(\pi)');ylabel('|H(e^j^\omega)|');
axis tight;
xlim([0 0.2]);
title('巴特沃斯双线性变换');

subplot(224);
plot(w/pi,angle(H2)/pi);
```

```
grid on;xlabel('\omega(\pi)');ylabel('Phase of H(e^j^\omega)(\pi)');
xlim([0 0.2]);
```

结果：



分析：

上述结果即为所要求的滤波器。

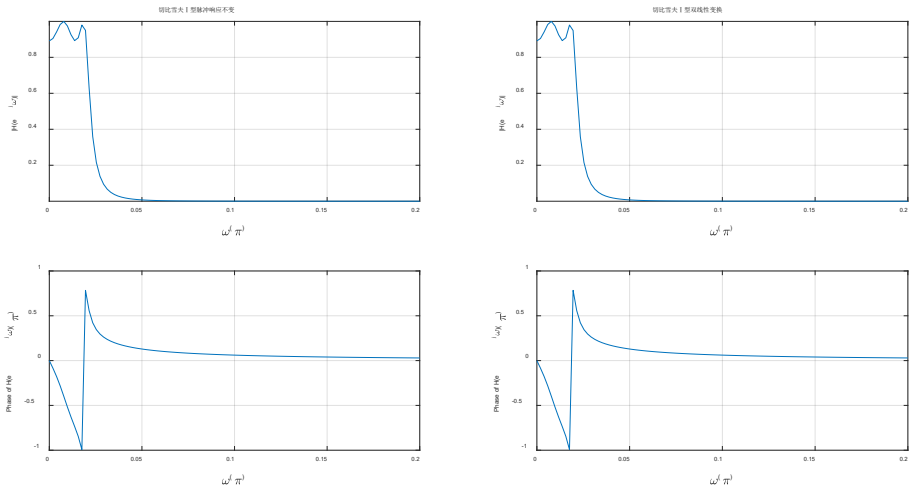
2. 代码和分析：

代码：

```
Fs = 10e4;
fp = 1000;
fs = 1500;
Rp = 1;
As = 15;

Wp = fp*2*pi;
Ws = fs*2*pi;
[N,Wc] = cheb1ord(Wp,Ws,Rp,As,'s');
[b,a]=cheby1(N,Rp,Wc,'s');

[b1,a1]=impinvar(b,a,Fs);
[b2,a2]=bilinear(b,a,Fs);
w=0:0.0001*pi:pi;
[H1,w]=freqz(b1,a1);
[H2,w]=freqz(b2,a2);
subplot(221);
plot(w/pi,abs(H1));
grid
on;xlabel('\omega(\pi)');ylabel('|H(e^j^\omega)|');
axis tight;
xlim([0 0.2]);
title('切比雪夫Ⅰ型脉冲响应不变');

subplot(223);
plot(w/pi,angle(H2)/pi);
grid on;xlabel('\omega(\pi)');ylabel('Phase of H(e^j^\omega)(\pi)');
xlim([0 0.2]);

subplot(222);
plot(w/pi,abs(H2));
grid
on;xlabel('\omega(\pi)');ylabel('|H(e^j^\omega)|');
axis tight;
xlim([0 0.2]);
title('切比雪夫Ⅰ型双线性变换');

subplot(224);
plot(w/pi,angle(H2)/pi);
grid on;xlabel('\omega(\pi)');ylabel('Phase of H(e^j^\omega)(\pi)');
```

xlim([0 0.2]);

结果：

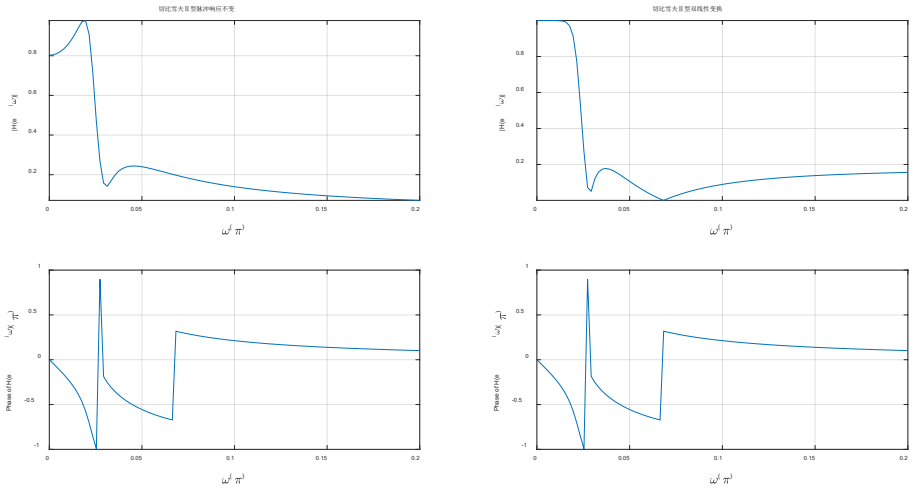


3. 代码和分析

代码：

```
Fs = 10e4;
fp = 1000;
fs = 1500;
Rp = 1;
As = 15;

Wp = fp*2*pi;
Ws = fs*2*pi;
[N,Wc] = cheb2ord(Wp,Ws,Rp,As,'s');
[b,a]=cheby2(N,As,Wc,'s');

[b1,a1]=impinvar(b,a,Fs);
[b2,a2]=bilinear(b,a,Fs);
w=0:0.0001*pi:pi;
[H1,w]=freqz(b1,a1);
[H2,w]=freqz(b2,a2);
subplot(221);
plot(w/pi,abs(H1));
grid
on;xlabel('\omega(\pi)');ylabel('|H(e^j^\om
ega)|');
axis tight;
xlim([0 0.2]);
title('切比雪夫 II 型脉冲响应不变');

subplot(223);
plot(w/pi,angle(H2)/pi);
grid on;xlabel('\omega(\pi)');ylabel('Phase
of H(e^j^\omega)(\pi)');
xlim([0 0.2]);

subplot(222);
plot(w/pi,abs(H2));
grid
on;xlabel('\omega(\pi)');ylabel('|H(e^j^\om
ega)|');
axis tight;
xlim([0 0.2]);
title('切比雪夫 II 型双线性变换');

subplot(224);
plot(w/pi,angle(H2)/pi);
grid on;xlabel('\omega(\pi)');ylabel('Phase
of H(e^j^\omega)(\pi)');
xlim([0 0.2]);
```

结果：

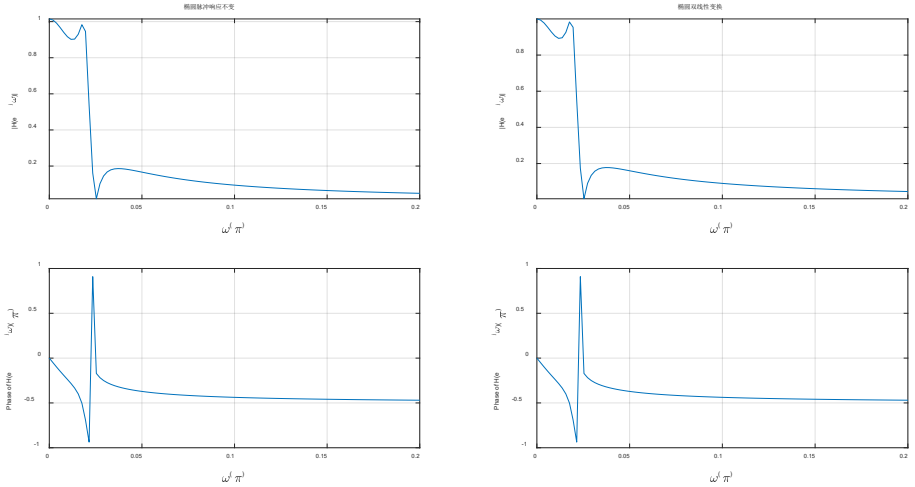


4. 代码和分析

代码：

```
Fs = 10e4;
fp = 1000;
fs = 1500;
Rp = 1;
As = 15;

Wp = fp*2*pi;
Ws = fs*2*pi;
[N,Wc] = ellipord(Wp,Ws,Rp,As,'s');
[b,a]=ellip(N,Rp,As,Wc,'s');

[b1,a1]=impinvar(b,a,Fs);
[b2,a2]=bilinear(b,a,Fs);
w=0:0.0001*pi:pi;
[H1,w]=freqz(b1,a1);
[H2,w]=freqz(b2,a2);
subplot(221);
plot(w/pi,abs(H1));
grid
on;xlabel('\omega(\pi)');ylabel('|H(e^j^\om
ega)|');
axis tight;
xlim([0 0.2]);
title('椭圆脉冲响应不变');

subplot(223);
plot(w/pi,angle(H2)/pi);
grid on;xlabel('\omega(\pi)');ylabel('Phase
of H(e^j^\omega)(\pi)');
xlim([0 0.2]);

subplot(222);
plot(w/pi,abs(H2));
grid
on;xlabel('\omega(\pi)');ylabel('|H(e^j^\om
ega)|');
axis tight;
xlim([0 0.2]);
title('椭圆双线性变换');

subplot(224);
plot(w/pi,angle(H2)/pi);
grid on;xlabel('\omega(\pi)');ylabel('Phase
of H(e^j^\omega)(\pi)');
xlim([0 0.2]);
```

结果：



分析：

脉冲响应不变法的优点是频率坐标的变换是线性的，因此如果模拟滤波器的频响是限带于折叠频率以内的话，通过变换后的数字滤波器的频响可以不失真地反映原响应与频率之间的关系。但是，其最大的缺点是有频谱的周期延拓效应，会产生混叠失真。故，脉冲响应不变法只能用于限带的频响特性，如衰减特性较好的低通，或带通，而且高频衰减越大，频响的混叠效应就越小。双线性变换法的优点是消除了脉冲响应不变法的固有的混叠失真，但缺点是频率变换之间的非常严重的非线性，会使变换后时域上的图像产生非常严重的失真。

五、心得与体会

通过实验，我学习了利用脉冲响应不变法设计 IIR 数字滤波器的基本原理为：从时域响应出发，使数字滤波器的单位脉冲响应$h(n)$模仿模拟滤波器的单位冲激响应$h_a(t)$，$h(n)$等于$h_a(t)$的取样值。所以设计中，应先根据数字指标转换来的模拟指标，设计模拟滤波器，再利用 impinvar 函数实现脉冲响应不变法模拟滤波器到数字滤波器的变换。

脉冲响应不变法和双线性法设计 IIR 数字滤波器是非常实用的技能，也会时我对 DSP 这门课程有更深刻的理解和认识。

本次实验最大的收获是了解了多种模拟滤波器到数字滤波器的转换方法。通过实验，确实巩固了课本知识。完成实验报告的过程中，查阅了相关资料，不仅开拓了知识面，更加深了对本部分知识的理解。

实验 4 频率取样法设计 FIR 数字滤波器

一、实验目的

掌握频率取样法设计 FIR 数字滤波器的原理及具体方法

二、实验原理

1、基本原理

频率取样法从频域出发，把理想的滤波器 $H_d(e^{j\omega})$ 等间隔采样得到 $H_d(k)$，将 $H_d(k)$ 作为实际设计滤波器的 $H(k)$。

$$H(k)=H_d(k)=H(e^{j\omega})|_{\omega=\frac{2\pi}{N}k}\quad k=0,1,\cdots,N-1 \qquad (8\text{-}1)$$

得到 $H(k)$ 以后可以由 $H(k)$ 来确定唯一确定滤波器的单位脉冲响应 $h(n)$，$H(e^{j\omega})$ 可以由 $H(k)$ 求得：

$$h(n)=IDFT[H(k)]$$

$$H(e^{j\omega})=\sum_{k=0}^{N-1}H(k)\phi(\omega-\frac{2\pi}{N}k) \qquad (8\text{-}2、3)$$

其中 $\phi(x)$ 为内插函数

$$\phi(\omega)=\frac{\sin(N\omega/2)}{N\sin(\omega/2)}\cdot e^{-j\omega\frac{N-1}{2}} \qquad (8\text{-}4)$$

由 $H(k)$ 求得的频率响应 $H(e^{j\omega})$ 将逼近 $H_d(e^{j\omega})$

如果我们设计的是线性相位 FIR 滤波器，则 $H(k)$ 的幅度和相位满足线性相位滤波器的约束条件。

我们将 $H(k)$ 表示为如下形式：

$$H(k)=\left|H(k)\right|e^{j\theta(k)}=H_r(k)e^{j\theta(k)} \qquad (8\text{-}5)$$

当 $h(n)$ 为实数，则 $H(k)=H^*(N-k)$

由此得到

$$H_r(k)=H_r(N-k) \qquad (8\text{-}6)$$

即 $H_r(k)$ 以 $k=N/2$ 为中心偶对称。在利用线性相位条件可知，对于 1 型和 2 型线性相位滤波器

$$\theta(k)=\begin{cases} -(\frac{N-1}{2})\frac{2\pi k}{N} & k=0,\cdots,\left\lfloor\frac{N-1}{2}\right\rfloor \\ (\frac{N-1}{2})\frac{2\pi}{N}(N-k) & k=\left\lfloor\frac{N-1}{2}\right\rfloor+1,\cdots N-1 \end{cases} \quad (8\text{-}7)$$

对于 3 型和 4 型线性相位滤波器

$$\theta(k)=\begin{cases} \pm\frac{\pi}{2}-(\frac{N-1}{2})\frac{2\pi k}{N} & k=0,\cdots,\left\lfloor\frac{N-1}{2}\right\rfloor \\ \mp\frac{\pi}{2}(\frac{N-1}{2})\frac{2\pi}{N}(N-k) & k=\left\lfloor\frac{N-1}{2}\right\rfloor+1,\cdots N-1 \end{cases} \quad (8\text{-}8)$$

其中，$[x]$表示取小于该数的最大的整数。

设计步骤

（1）由给定的理想滤波器给出$H_r(k)$和$\theta(k)$。

（2）由$H(k)=\left|H(k)\right|e^{j\theta(k)}=H_r(k)e^{j\theta(k)}$求得$H(k)$

（3） 根据$H(k)$求得$h(n)$或$H(e^{j\omega})$

三、实验内容

1. 设计一个数字低通 FIR 滤波器，其技术指标如下：

ω_p=0.2π， R_p=0.25dB

ω_st=0.3π， A_s=50dB

分别采用矩形窗、汉宁窗、海明窗、布莱克曼窗、凯瑟窗设计该滤波器。结合实验结果分别讨论上述方法设计的数字滤波器是否符合指标。

2. 设计一个数字带通 FIR 滤波器，其技术指标如下：

下阻带边缘：$\omega_{st1}=0.2\pi, A_s=60dB$

下通带边缘：$\omega_{p1}=0.35\pi, R_p=1dB$

上通带边缘：$\omega_{p2}=0.65\pi, R_p=1dB$

上阻带边缘：$\omega_{st2}=0.8\pi, A_s=60dB$

3. 采用频率取样设计法设计 FIR 数字低通滤波器，满足以下指标

ω_p=0.2π， R_p=0.25dB

ω_st=0.3π， A_s=50dB

（1）取 N=20，过渡带没有样本。

（2）取 N=40，过渡带有一个样本，T=0.39。

（3）取 N=60，过渡带有两个样本，T1=0.5925， T2=0.1099。

（4）分别采用上述方法设计的数字滤波器是否都能满足给定的指标要求。

4. 采用频率取样技术设计下面的高通滤波器

ω_st=0.6π，A_s=50dB

ω_p=0.8π，R_p=1dB

对于高通滤波器，N 必须为奇数（或 1 型滤波器）。选择 N=33，过渡带有两个样本，过渡带样本最优值为 T1=0.1095，T2=0.598。

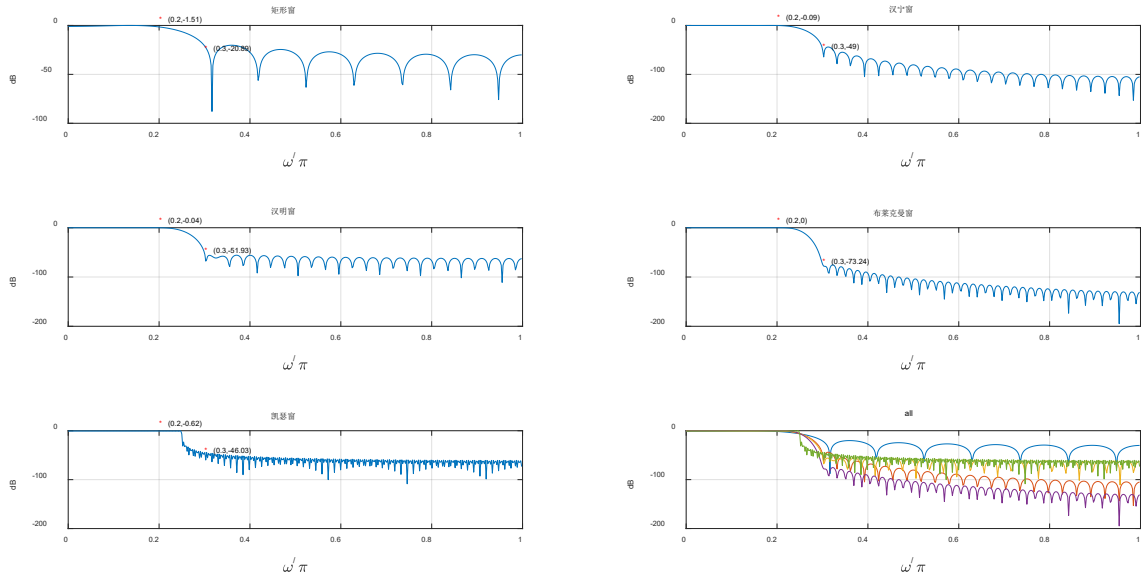
四、实验结果和分析

1. 代码和分析

代码：

```
clear;
wp = 0.2*pi;
wst = 0.3*pi;
As = 50;
tr_width = wst-wp;
Nn = ceil([1.8 6.2 6.6 11 (As-7.95)/2.285/tr_width]*pi/tr_width) + 1; %N 的选取
WW = {boxcar(Nn(1))'; hanning(Nn(2))'; hamming(Nn(3))'; blackman(Nn(4))'; kaiser(Nn(5))'}; % 各种窗
titles = {'矩形窗'; '汉宁窗'; '汉明窗'; '布莱克曼窗'; '凯瑟窗'};
N = 0; %初始化
for i = 1:5
    N = Nn(i);
    w = WW{i};
    n = 0:N-1;
    wc = (wp + wst)/2;
    alpha = (N-1)/2;
    hd = (wc/pi)*sinc((wc/pi)*(n-alpha));
    h = hd.*w;
    [H,w] = freqz(h,1);
    y = 20*log10(abs(H)/max(abs(H))); % 纵坐标
    subplot(3,2,i);
    plot(w/pi,y);
    title(titles{i});
    xlabel('\omega/\pi');ylabel('dB');
    digits(2);
    y_p = interp1(w/pi,y,wp/pi);
    y_st = interp1(w/pi,y,wst/pi);
    text(wp/pi,y_p,'*','color','r');
    text(wp/pi+0.02,y_p,['(',num2str(wp/pi),',',num2str(roundn(y_p,-2)),')']);
    text(wst/pi,y_st-10,'*','color','r');
    text(wst/pi+0.02,y_st-10,['(',num2str(wst/pi),',',num2str(roundn(y_st,-2)),')']);
    grid on;
    subplot(3,2,6);
    grid on;
    title('all')
    plot(w/pi,y);
    hold on;
    xlabel('\omega/\pi');ylabel('dB');
end
```

结果：



分析：

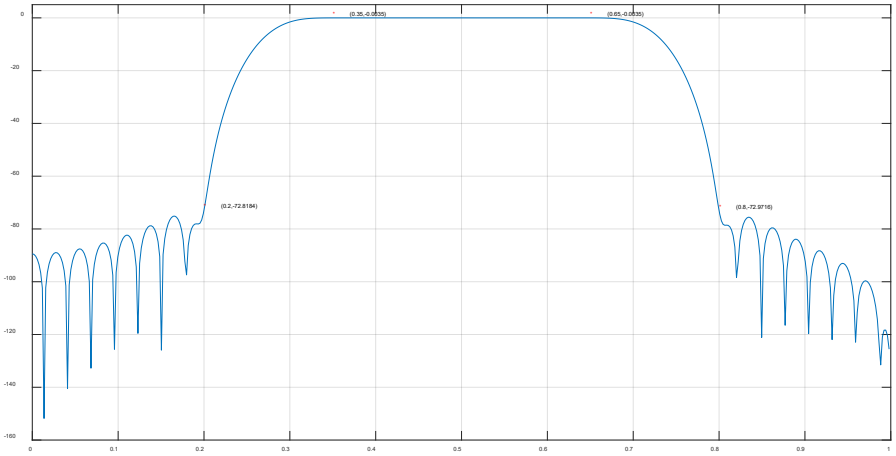
因为 $\begin{cases} R_p = 0.25dB \\ A_s = 50dB \end{cases}$ ，由图可知，符合标准的有：汉明窗，布莱克曼窗。

2. 代码和分析

代码：

```
clear;
w_st1 = 0.2*pi; w_p1 = 0.35*pi;
w_st2 = 0.8*pi; w_p2 = 0.65*pi;
As = 60; Rp = 1;
tr_width = w_p1-w_st1; % 相等，取第一个作为宽度
wc1 = (w_st1+w_p1)/2; wc2 = (w_st2+w_p2)/2;
%采取 blackman 窗
N = ceil(11*pi/tr_width);
hn = fir1(N-1, [wc1 wc2]/pi, blackman(N));
[H,w] = freqz(hn,1);
y = 20*log10(abs(H)/max(abs(H)));
plot(w/pi, y);
grid on;
ws = [w_st1 w_st2 w_p1 w_p2];
% for 循环标出特殊点便于观察
for i = 1:4
    wi = ws(i);
    yi = interp1(w/pi,y,wi/pi);
    text(wi/pi,yi,'*','color','r');
    text(wi/pi+0.02,yi,['(',num2str(wi/pi),',',num2str(roundn(yi,-4)),')']);
end
ylim([-160 5]);
```

结果：



分析：

由图中标出的点容易看出，运用 blackman 窗设计得到的滤波器满足指标。

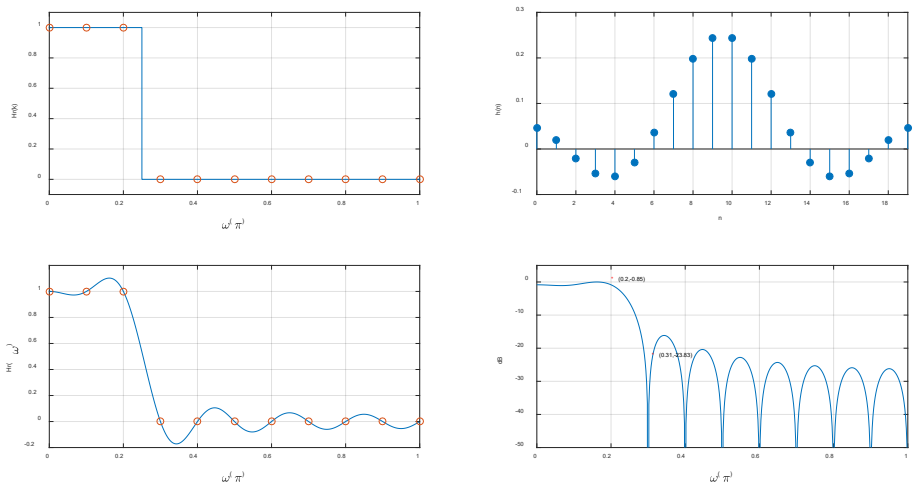
3. 代码和分析

3.1 N=20

代码：

```
N = 20; alpha = (N - 1) /2; l = 0 : N
- 1; wl = (2 * pi / N) * l;
Hrs = [1, 1, 1, zeros(1, 15), 1, 1];
Hdr = [1, 1, 0, 0]; wdl = [0, 0.25,
0.25, 1];
k1 = 0 : floor((N - 1) / 2); k2 =
floor((N - 1) / 2) + 1 : N - 1;
angH = [-alpha * (2 * pi) / N * k1,
alpha * (2 * pi) / N * (N - k2)];
H = Hrs .* exp(j * angH);
h = ifft(H, N);
w = [0 : 500] * pi / 500;
H = freqz(h, 1, w);
[Hr, wr] = zerophase(h);
subplot(221); plot(wdl, Hdr, wl(1 :
11) / pi, Hrs(1 : 11), 'o');
axis([0, 1, -0.1, 1.1]);
xlabel('\omega(\pi)');
ylabel('Hr(k)');
subplot(222);
stem(l, h, 'filled');
axis([0, N - 1, -0.1, 0.3]);
xlabel('n'); ylabel('h(n)');
subplot(223); plot(wr / pi, Hr,
wl(1 : 11) / pi, Hrs(1 : 11), 'o');
axis([0, 1, -0.2, 1.2]);
xlabel('\omega(\pi)');
ylabel('Hr(\omega)');
subplot(224);
y =  20 * log10((abs(H) /
max(abs(H))));
plot(w / pi,y);
wp = 0.2*pi;
wst = 0.31*pi;
y_p = interp1(w/pi,y,wp/pi);
y_st = interp1(w/pi,y,wst/pi);
text(wp/pi,y_p,'*','color','r');
text(wp/pi+0.02,y_p,['(',num2str(wp/p
i),',',num2str(roundn(y_p,-2)),')']);
text(wst/pi,y_st,'*','color','r');
text(wst/pi+0.02,y_st,['(',num2str(ws
t/pi),',',num2str(roundn(y_st,-
2)),')']);
axis([0, 1, -50, 5]);
grid on;
xlabel('\omega(\pi)'); ylabel('dB');
```

结果：

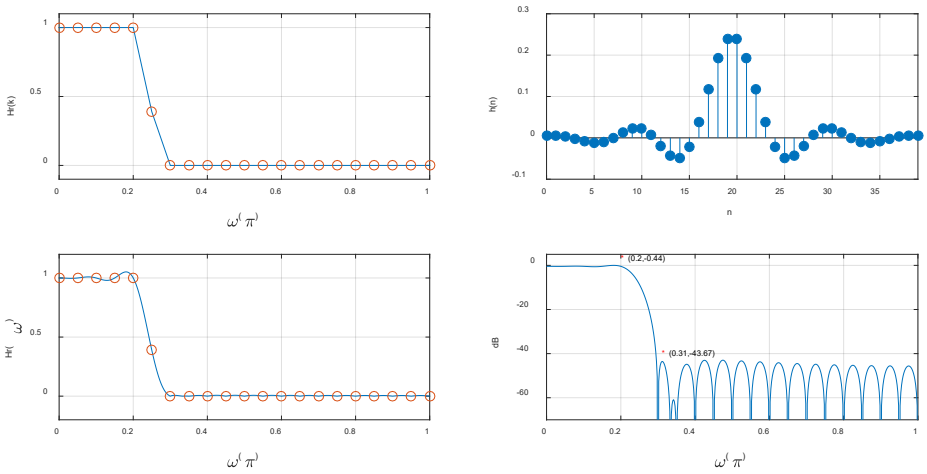


3.2 N=40

代码：

```
N = 40; alpha = (N - 1) /2; l = 0 : N
- 1; wl = (2 * pi / N) * l;
Hrs = [1, 1, 1, 1, 1, 0.39, zeros(1,
29), 0.39, 1, 1, 1, 1];
Hdr = [1, 1,0.39, 0, 0]; wdl = [0,
0.2, 0.25, 0.3, 1];
k1 = 0 : floor((N - 1) / 2); k2 =
floor((N - 1) / 2) + 1 : N - 1;
angH = [-alpha * (2 * pi) / N * k1,
alpha * (2 * pi) / N * (N - k2)];
H = Hrs .* exp(j * angH);
h = ifft(H, N);
w = [0 : 500] * pi / 500;
[H, w] = freqz(h, 1, w);
Hr = abs(H);

subplot(221); plot(wdl, Hdr, wl(1 :
21) / pi, Hrs(1 : 21), 'o');
axis([0, 1, -0.1, 1.1]);
xlabel('\omega(\pi)');
ylabel('Hr(k)');
grid on;
subplot(222);
stem(l, h, 'filled');
axis([0, N - 1, -0.1, 0.3]);
xlabel('n'); ylabel('h(n)');
grid on;
subplot(223); plot(w / pi, Hr, wl(1 :
21) / pi, Hrs(1 : 21), 'o');
axis([0, 1, -0.2, 1.2]);
xlabel('\omega(\pi)');
ylabel('Hr(\omega)');
grid on;
subplot(224); y =  20 * log10((abs(H)
/ max(abs(H))));
plot(w / pi,y);
wp = 0.2*pi;
wst = 0.31*pi;
y_p = interp1(w/pi,y,wp/pi);
y_st = interp1(w/pi,y,wst/pi);
text(wp/pi,y_p,'*','color','r');
text(wp/pi+0.02,y_p,['(',num2str(wp/p
i),',',num2str(roundn(y_p,-2)),')']);
text(wst/pi,y_st,'*','color','r');
text(wst/pi+0.02,y_st,['(',num2str(ws
t/pi),',',num2str(roundn(y_st,-
2)),')']);
axis([0, 1, -70, 5]);
grid on;
xlabel('\omega(\pi)'); ylabel('dB');
```

结果：

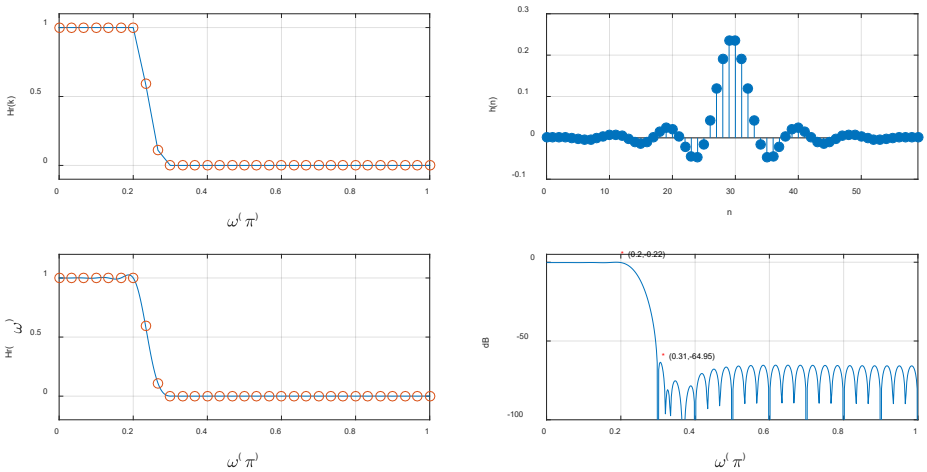


3.3 N=60

代码：

```
N = 60; alpha = (N - 1) /2; l = 0 : N
- 1; wl = (2 * pi / N) * l;
Hrs = [ones(1,7), 0.5925, 0.1099,
zeros(1, 43), 0.1099,
0.5925,ones(1,6)];
Hdr=[1,1,0.5925,0.1099,0,0];
wdl=[0,0.2,7/30,8/30,0.3,1];
k1 = 0 : floor((N - 1) / 2); k2 =
floor((N - 1) / 2) + 1 : N - 1;
angH = [-alpha * (2 * pi) / N * k1,
alpha * (2 * pi) / N * (N - k2)];
H = Hrs .* exp(j * angH);
h = ifft(H, N);
w = [0 : 500] * pi / 500;
[H, w] = freqz(h, 1, w);
Hr = abs(H);
subplot(221); plot(wdl, Hdr, wl(1 :
31) / pi, Hrs(1 : 31), 'o');
axis([0, 1, -0.1, 1.1]);
xlabel('\omega(\pi)');
ylabel('Hr(k)');
grid;
subplot(222);
stem(l, h, 'filled');
axis([0, N - 1, -0.1, 0.3]);
xlabel('n'); ylabel('h(n)');
grid;
subplot(223); plot(w / pi, Hr, wl(1 :
31) / pi, Hrs(1 : 31), 'o');
axis([0, 1, -0.2, 1.2]);
xlabel('\omega(\pi)');
ylabel('Hr(\omega)');
grid;
subplot(224);
y =  20 * log10((abs(H) /
max(abs(H))));
plot(w / pi,y);
wp = 0.2*pi;
wst = 0.31*pi;
y_p = interp1(w/pi,y,wp/pi);
y_st = interp1(w/pi,y,wst/pi);
text(wp/pi,y_p,'*','color','r');
text(wp/pi+0.02,y_p,['(',num2str(wp/p
i),',',num2str(roundn(y_p,-2)),')']);
text(wst/pi,y_st,'*','color','r');
text(wst/pi+0.02,y_st,['(',num2str(ws
t/pi),',',num2str(roundn(y_st,-
2)),')']);

axis([0, 1, -50, 5]);
grid;
xlabel('\omega(\pi)'); ylabel('dB');
```

结果：



分析：

由第四幅图中坐标点可以看出，该滤波器符合要求（可以通过增加取样点数来提高滤波器的 As 指标）。

3.4

答：从上图图中所标坐标可以看出，只有当 N=60 时此时的滤波器才满足设计指标，分析易知，这是因为在取样点数不断增加的时候，过渡带出现，通过增加取样点数来增加过渡带样本个数，这会使滤波器越来越贴近理想滤波器。

4. 代码和分析

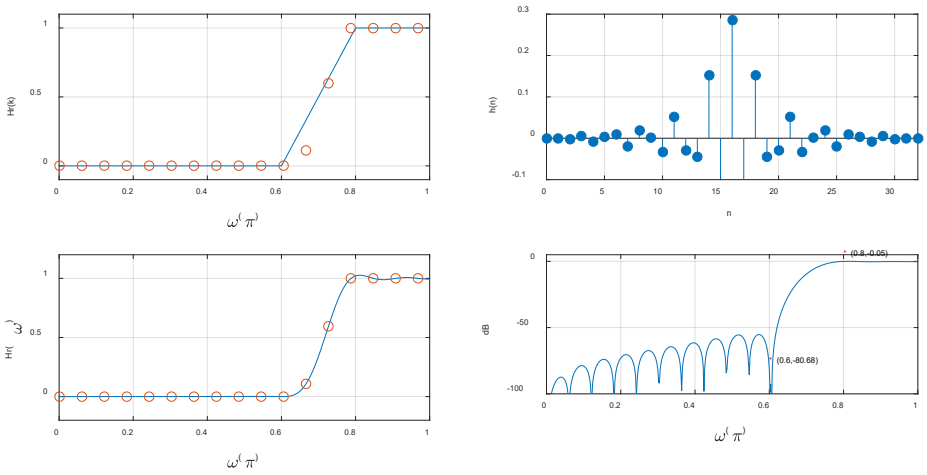
代码：

```
N = 33; alpha = (N - 1) /2; l = 0 : N - 1; wl = (2 * pi / N) * l;
Hrs = [zeros(1, 11), 0.1095, 0.598, ones(1, 8), 0.598, 0.1095, zeros(1, 10)];
Hdr = [0, 0, 1, 1]; wdl = [0, 0.6, 0.8, 1];
k1 = 0 : floor((N - 1) / 2); k2 = floor((N - 1) / 2) + 1 : N - 1;
angH = [-alpha * (2 * pi) / N * k1, alpha * (2 * pi) / N * (N - k2)];
H = Hrs .* exp(j * angH);
h = ifft(H, N);
w = [0 : 500] * pi / 500;
[H, w] = freqz(h, 1, w);
Hr = abs(H);
subplot(221); plot(wdl, Hdr, wl(1 : 17) / pi, Hrs(1 : 17), 'o');
axis([0, 1, -0.1, 1.1]);
xlabel('\omega(\pi)');
ylabel('Hr(k)');
subplot(222);
stem(l, h, 'filled');
axis([0, N - 1, -0.1, 0.3]);
xlabel('n'); ylabel('h(n)');
subplot(223); plot(w / pi, Hr, wl(1 : 17) / pi, Hrs(1 : 17), 'o');
axis([0, 1, -0.2, 1.2]);
xlabel('\omega(\pi)');
ylabel('Hr(\omega)');
subplot(224); y = 20 * log10((abs(H) / max(abs(H))));
plot(w / pi,y);
wp = 0.6*pi;
wst = 0.8*pi;
y_p = interp1(w/pi,y,wp/pi);
y_st = interp1(w/pi,y,wst/pi);
text(wp/pi,y_p,'*','color','r');
text(wp/pi+0.02,y_p,['(',num2str(wp/pi),',',num2str(roundn(y_p,-2)),')']);
text(wst/pi,y_st,'*','color','r');
text(wst/pi+0.02,y_st,['(',num2str(ws
```

```
t/pi),',',num2str(roundn(y_st,-
2)),')']);
axis([0, 1, -100, 5]);
```

```
grid on;
xlabel('\omega(\pi)'); ylabel('dB');
```

结果：



五、心得与体会

本次实验要求我们掌握频率取样法设计 FIR 数字滤波器的原理及具体方法。通过查阅相关资料，可以知道利用频率取样法设计 FIR 数字滤波器设计的基本原理是：从频域出发，把理想的滤波器 $H_d(e^{j\omega})$ 等间隔采样得到 $H_d(k)$，将 $H_d(k)$ 作为实际设计滤波器的 $H(k)$，然后通过内插公式恢复出实际的滤波器的频响。

通过此次实验，我充分了解了频率取样法在设计 FIR 滤波器时的原理和使用方法，对 FIR 滤波器的性能加深了理解，并且对设计滤波器的过程更加熟悉了，这对我以后的学习和将来的工作都大有裨益。