LAB3 第三章 周期信号的傅里叶级数表示

应逸雯12210159 陈薇羽12210460

通过本实验的练习，掌握了以下技能：

1.通过y./x辨别特征函数

2.使用for循环计算傅里叶级数

3.使用fft、ifft、fftshift等快速傅里叶级数计算工具

作业内容

3.5a a0=1,a2=a2*=exp(1i*pi/4),a4=a-4*=2*exp(1i*pi/3)。判断x[n]是复数/实数/纯虚数。

由于ak=a-k*，傅里叶级数共轭相等，故x[n]是实数。

3.5b 创建向量a。

条件隐含a1=a-4，a3=a-2。注意元素索引需+1

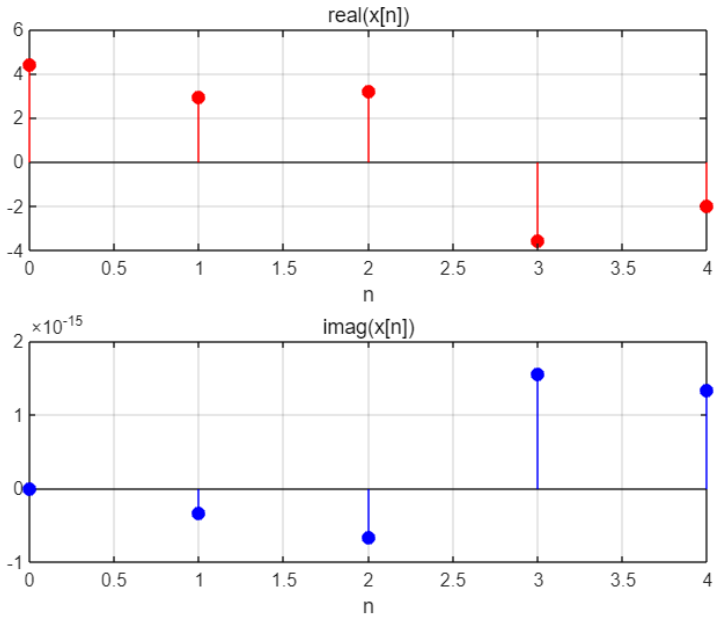
```
clc,clear,close all;
a=zeros(1,5);
a(1)=1;
a(2)=2*exp(-1i*pi/3);
a(3)=exp(1i*pi/4);
a(4)=exp(-1i*pi/4);
a(5)=2*exp(1i*pi/3);
a
```

```
a = 1×5 complex
   1.0000 + 0.0000i   1.0000 - 1.7321i   0.7071 + 0.7071i   0.7071 - 0.7071i   1.0000 + 1.7321i
```

3.5c 计算出x[n]，并画图，并于3.5a的结论相比较。

用for循环带入合成公式计算，若仅有舍入误差，则为纯实数信号。

```
x=[];
nx=0:4;
N=5;
for u=0:4%u为x的索引值
    temp=0;
    for v=0:4%用于求和
        temp=temp+a(v+1)*exp(1i*v*2*pi/N*u);
    end
    x=[x temp];
end
%得到了x[n]
figure;
subplot(2,1,1),stem(nx,real(x),'r','filled'),title('real(x[n])'),xlabel('n'),grid on;
subplot(2,1,2),stem(nx,imag(x),'b','filled'),title('imag(x[n])'),xlabel('n'),grid on;
```

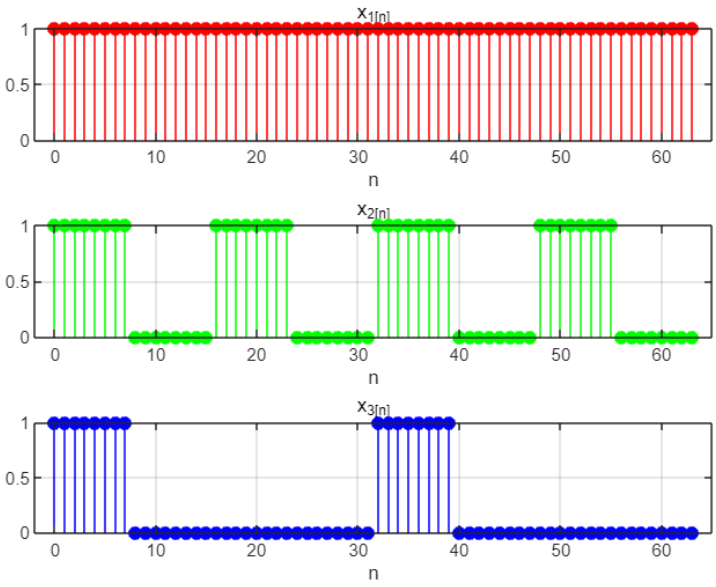


根据x[n]的实部和虚部的图像，虚部误差量级为10^-15，属于舍入误差，在可接受范围内，可得知，x[n]是纯实数信号。

3.5d 定义x1=ones(1,8)(0<=n<=7)，x2=[ones(1,8) zeros(1,8)](0<=n<=15)，x3=[ones(1,8) zeros(1,24)](0<=n=31)，并且绘制[0,63]区间的图像。

先给x1，x2，x3赋值，绘图时拼接多个周期。

```
clc,clear,close all;
nx1=0:7;
x1=ones(1,8);
nx2=0:15;
x2=[ones(1,8) zeros(1,8)];
nx3=0:31;
x3=[ones(1,8) zeros(1,24)];
nx=0:63;
%创建好了x1,x2,x3
figure;
subplot(3,1,1),stem(nx,[x1 x1 x1 x1 x1 x1 x1 x1],'r','filled'),title('x_1[n]'),xlabel('n'),grid on,xlim([-2 65]);
subplot(3,1,2),stem(nx,[x2 x2 x2 x2],'g','filled'),title('x_2[n]'),xlabel('n'),grid on,xlim([-2 65]);
subplot(3,1,3),stem(nx,[x3 x3],'b','filled'),title('x_3[n]'),xlabel('n'),grid on,xlim([-2 65]);
```



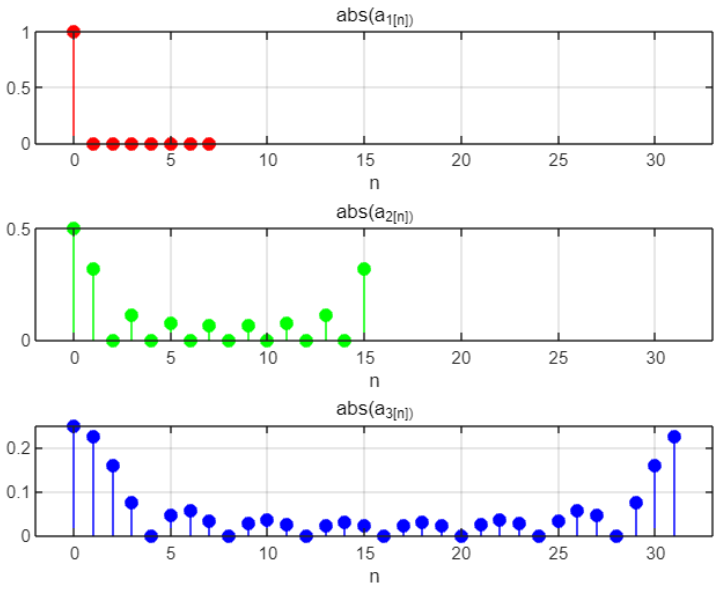
结果如图。

3.5e 用fft计算a1，a2，a3，并画出绝对值图像。预测a1(1)，a2(1)，a3(1)，并与图像相比较。

计算a1，a2，a3使用a=fft(x)/N公式。

a(1)=sum(x[n])/N，故a1(1)=1，a2(1)=0.5，a3(1)=0.25。

```
N1=8;
N2=16;
N3=32;
a1=fft(x1)/N1;
a2=fft(x2)/N2;
a3=fft(x3)/N3;
%计算出了a1,a2,a3
figure;
subplot(3,1,1),stem(nx1,abs(a1),'r','filled'),title('abs(a_1[n])'),xlabel('n'),grid on,xlim([-2 33]);
subplot(3,1,2),stem(nx2,abs(a2),'g','filled'),title('abs(a_2[n])'),xlabel('n'),grid on,xlim([-2 33]);
subplot(3,1,3),stem(nx3,abs(a3),'b','filled'),title('abs(a_3[n])'),xlabel('n'),grid on,xlim([-2 33]);
```



根据图像，a1(1)，a2(1)，a3(1)与理论值相同。

3.5f 观察仅使用部分a，计算出x3-2，x3-8，x3-12，x3-all

用for循环计算x，需调整v部分参数，并且分开求正数索引和负数索引，最后画图。

```
nx=0:31;
N=32;
x3_2=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:2%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-2:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_2=[x3_2 temp];
end
x3_2
```

```
x3_2 = 1×32 complex
   0.6610 + 0.0000i   0.8256 + 0.0000i   0.9478 - 0.0000i   1.0129 + 0.0000i   1.0129 - 0.0000i   0.9478 + 0.0000i   0.8256 +
```

```
%得到了x3_2[n]
x3_8=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:8%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-8:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_8=[x3_8 temp];
end
x3_8
```

```
x3_8 = 1×32 complex
   0.7347 + 0.0000i   1.0900 + 0.0000i   1.0900 - 0.0000i   0.9285 + 0.0000i   0.9285 - 0.0000i   1.0900 + 0.0000i   1.0900 +
```

```
%得到了x3_8[n]
x3_12=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:12%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-12:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
```

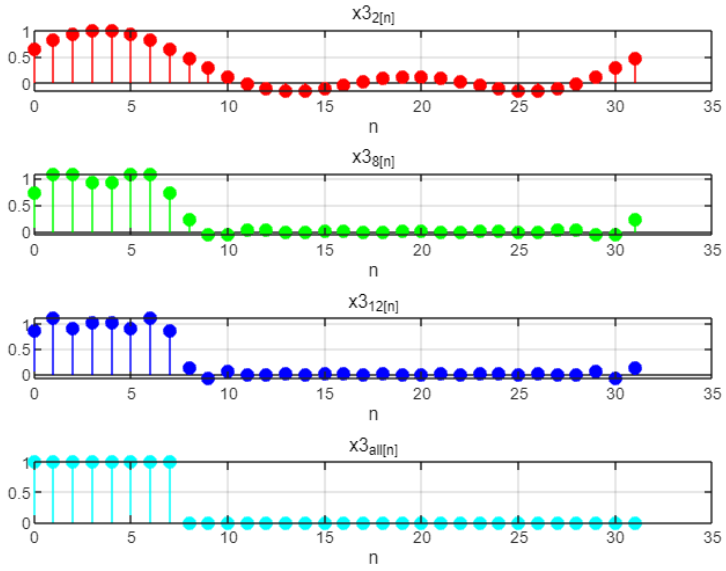
```
    x3_12=[x3_12 temp];
end
x3_12
```

```
x3_12 = 1×32 complex
   0.8686 + 0.0000i   1.1199 + 0.0000i   0.9160 - 0.0000i   1.0302 - 0.0000i   1.0302 - 0.0000i   0.9160 + 0.0000i   1.1199 +
```

```
%得到了x3_12[n]
x3_all=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:16%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-15:-1
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_all=[x3_all temp];
end
x3_all
```

```
x3_all = 1×32 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 - 0.0000i   1.0000 - 0.0000i   1.0000 - 0.0000i   1.0000 + 0.0000i   1.0000 +
```

```
%得到了x3_all[n]
figure;
subplot(4,1,1),stem(nx,real(x3_2),'r','filled'),title('x3_2[n]'),xlabel('n'),grid on;
subplot(4,1,2),stem(nx,real(x3_8),'g','filled'),title('x3_8[n]'),xlabel('n'),grid on;
subplot(4,1,3),stem(nx,real(x3_12),'b','filled'),title('x3_12[n]'),xlabel('n'),grid on;
subplot(4,1,4),stem(nx,real(x3_all),'c','filled'),title('x3_all[n]'),xlabel('n'),grid on;
```

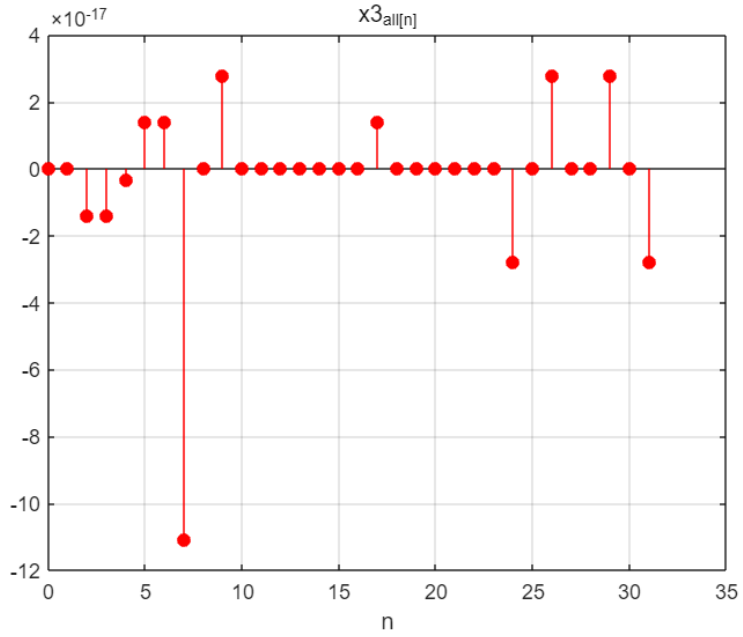


可观察到，当使用的谐波数量越多，越能够还原信号。

3.5g 证明x3_all[n]一定为纯实数信号。

画出x3_all虚部，如果仅有舍入误差，则为纯实数信号。

```
figure;
stem(nx,imag(x3_all),'r','filled'),title('x3_all[n]'),xlabel('n'),grid on;
```



误差量级为10^-17，考虑到舍入误差，在可接受范围内，故x3_all[n]为纯实数信号。

x3_all是从a3变换坐标计算而来，而间隔一个周期变换坐标不影响还原信号，a3又是由x3原信号计算得出，故x3_all理应与x3相同。

3.5h 观察更多的谐波数量是否能够更加还原信号。并且观察是否有吉布斯现象。

```
nx=0:31;
N=32;

x3_1=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:1%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    w=-1;%求负数索引部分
    temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    x3_1=[x3_1 temp];
end
x3_1
%得到了x3_1[n]

x3_2=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:2%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-2:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_2=[x3_2 temp];
end
x3_2
```

```
x3_2 = 1×32 complex
   0.6610 + 0.0000i   0.8256 + 0.0000i   0.9478 - 0.0000i   1.0129 + 0.0000i   1.0129 - 0.0000i   0.9478 + 0.0000i   0.8256 +
```

```
%得到了x3_2[n]

x3_3=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:3%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-3:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_3=[x3_3 temp];
```

```
end
x3_3
```

```
x3_3 = 1×32 complex
   0.5893 + 0.0000i   0.8405 + 0.0000i   1.0444 - 0.0000i   1.1586 + 0.0000i   1.1586 - 0.0000i   1.0444 + 0.0000i   0.8405 +
```

```
%得到了x3_3[n]

x3_4=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:4%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-4:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_4=[x3_4 temp];
end
x3_4
```

```
x3_4 = 1×32 complex
   0.5893 + 0.0000i   0.8405 + 0.0000i   1.0444 - 0.0000i   1.1586 + 0.0000i   1.1586 - 0.0000i   1.0444 + 0.0000i   0.8405 +
```

```
%得到了x3_4[n]

x3_5=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:5%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-5:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_5=[x3_5 temp];
end
x3_5
```

```
x3_5 = 1×32 complex
   0.6790 + 0.0000i   0.9130 + 0.0000i   1.0352 - 0.0000i   1.0759 + 0.0000i   1.0759 - 0.0000i   1.0352 + 0.0000i   0.9130 +
```

```
%得到了x3_5[n]

x3_6=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:6%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-6:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_6=[x3_6 temp];
end
x3_6
```

```
x3_6 = 1×32 complex
   0.7415 + 0.0000i   1.0234 + 0.0000i   1.0572 - 0.0000i   0.9824 + 0.0000i   0.9824 - 0.0000i   1.0572 + 0.0000i   1.0234 +
```

```
%得到了x3_6[n]

x3_7=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:7%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-7:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_7=[x3_7 temp];
end
x3_7
```

```
x3_7 = 1×32 complex
   0.7347 + 0.0000i   1.0900 + 0.0000i   1.0900 - 0.0000i   0.9285 + 0.0000i   0.9285 - 0.0000i   1.0900 + 0.0000i   1.0900 +
```

```
%得到了x3_7[n]

x3_8=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:8%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-8:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_8=[x3_8 temp];
end
x3_8
```

```
x3_8 = 1×32 complex
   0.7347 + 0.0000i   1.0900 + 0.0000i   1.0900 - 0.0000i   0.9285 + 0.0000i   0.9285 - 0.0000i   1.0900 + 0.0000i   1.0900 +
```

```
%得到了x3_8[n]

x3_9=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:9%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-9:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_9=[x3_9 temp];
end
x3_9
```

```
x3_9 = 1×32 complex
   0.7916 + 0.0000i   1.0734 + 0.0000i   1.0396 - 0.0000i   0.9648 + 0.0000i   0.9648 - 0.0000i   1.0396 + 0.0000i   1.0734 +
```

```
%得到了x3_9[n]

x3_10=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:10%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-10:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_10=[x3_10 temp];
end
x3_10
```

```
x3_10 = 1×32 complex
   0.8541 + 0.0000i   1.0881 + 0.0000i   0.9659 - 0.0000i   1.0066 - 0.0000i   1.0066 - 0.0000i   0.9659 + 0.0000i   1.0881 +
```

```
%得到了x3_10[n]

x3_11=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:11%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-11:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_11=[x3_11 temp];
end
x3_11
```

```
x3_11 = 1×32 complex
   0.8686 + 0.0000i   1.1199 + 0.0000i   0.9160 - 0.0000i   1.0302 - 0.0000i   1.0302 - 0.0000i   0.9160 + 0.0000i   1.1199 +
```

```
%得到了x3_11[n]

x3_12=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:12%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-12:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_12=[x3_12 temp];
end
x3_12
```

```
x3_12 = 1×32 complex
   0.8686 + 0.0000i   1.1199 + 0.0000i   0.9160 - 0.0000i   1.0302 - 0.0000i   1.0302 - 0.0000i   0.9160 + 0.0000i   1.1199 +
```

```
%得到了x3_12[n]

x3_13=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:13%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-13:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_13=[x3_13 temp];
end
x3_13
```

```
x3_13 = 1×32 complex
   0.9093 + 0.0000i   1.0739 + 0.0000i   0.9517 - 0.0000i   1.0168 - 0.0000i   1.0168 - 0.0000i   0.9517 + 0.0000i   1.0739 +
```

```
%得到了x3_13[n]

x3_14=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:14%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-14:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_14=[x3_14 temp];
end
x3_14
```

```
x3_14 = 1×32 complex
   0.9718 + 0.0000i   1.0209 + 0.0000i   0.9871 - 0.0000i   1.0044 - 0.0000i   1.0044 - 0.0000i   0.9871 + 0.0000i   1.0209 +
```

```
%得到了x3_14[n]

x3_15=[];
for u=0:31%u为x的索引值
    temp=0;
    for v=0:15%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-15:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_15=[x3_15 temp];
end
x3_15
```

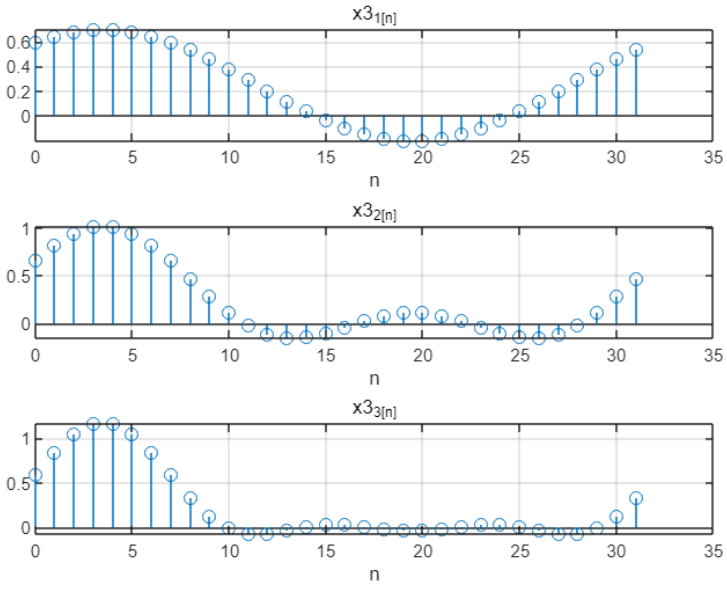
```
x3_15 = 1×32 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 - 0.0000i   1.0000 - 0.0000i   1.0000 - 0.0000i   1.0000 + 0.0000i   1.0000 +
```

```
%得到了x3_15[n]

x3_all=[];
```

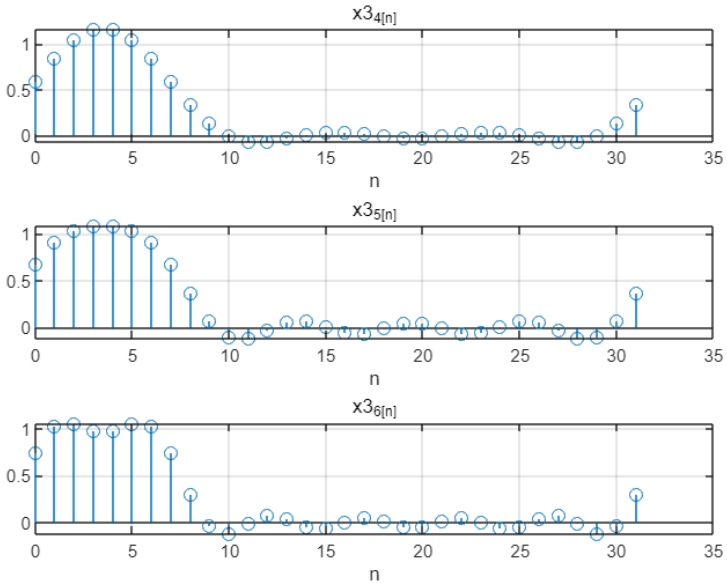
```
for u=0:31%u为x的索引值
    temp=0;
    for v=0:16%求正数索引部分
        temp=temp+a3(v+1)*exp(1i*v*2*pi/N*u);
    end
    for w=-15:-1%求负数索引部分
        temp=temp+conj(a3(1-w))*exp(1i*w*2*pi/N*u);
    end
    x3_all=[x3_all temp];
end
x3_all
```

```
x3_all = 1×32 complex
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 - 0.0000i   1.0000 - 0.0000i   1.0000 - 0.0000i   1.0000 + 0.0000i   1.0000 +
```

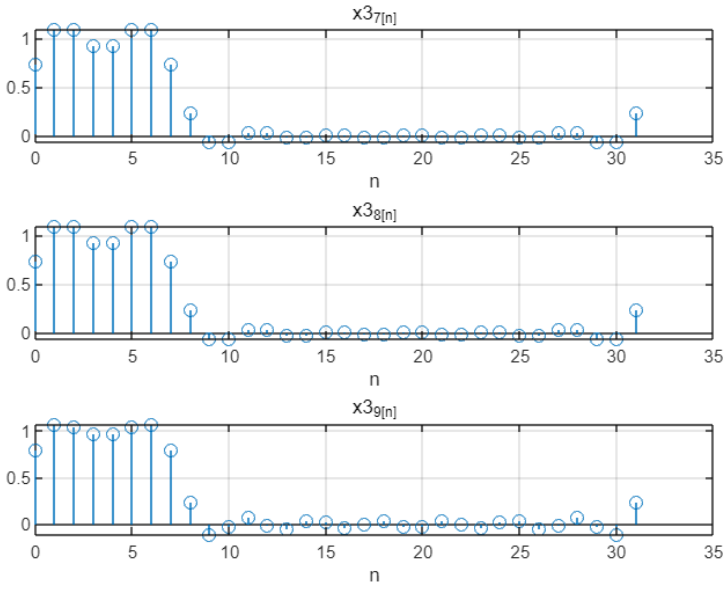
```
%得到了x3_all[n]

%输出所有的x3
figure;
subplot(3,1,1),stem(nx,real(x3_1)),title('x3_1[n]'),xlabel('n'),grid on;
subplot(3,1,2),stem(nx,real(x3_2)),title('x3_2[n]'),xlabel('n'),grid on;
subplot(3,1,3),stem(nx,real(x3_3)),title('x3_3[n]'),xlabel('n'),grid on;
```



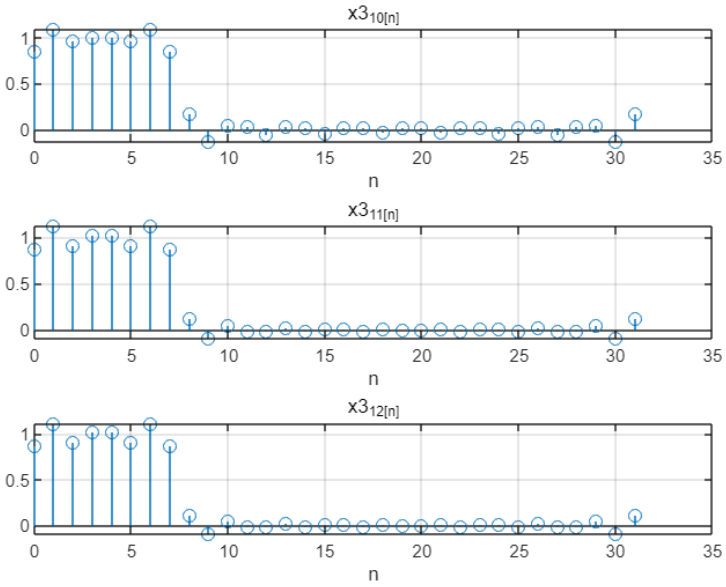

```
figure;
subplot(3,1,1),stem(nx,real(x3_4)),title('x3_4[n]'),xlabel('n'),grid on;
subplot(3,1,2),stem(nx,real(x3_5)),title('x3_5[n]'),xlabel('n'),grid on;
subplot(3,1,3),stem(nx,real(x3_6)),title('x3_6[n]'),xlabel('n'),grid on;
```



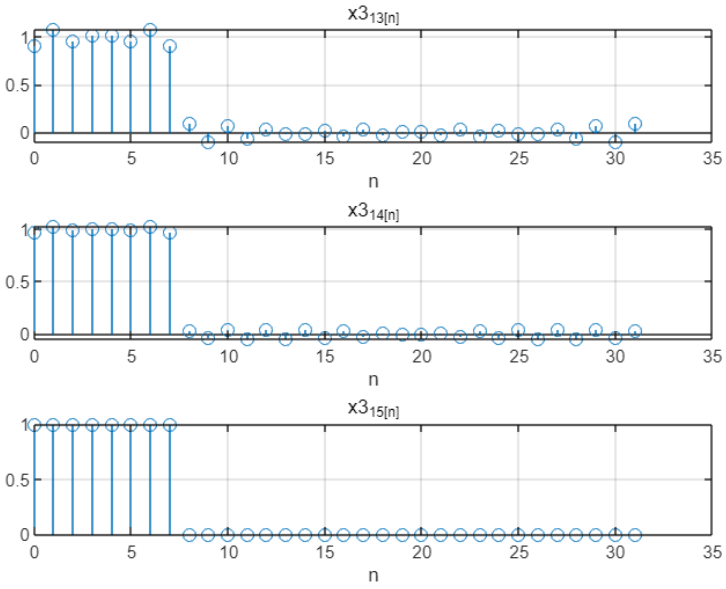

```
figure;
subplot(3,1,1),stem(nx,real(x3_7)),title('x3_7[n]'),xlabel('n'),grid on;
subplot(3,1,2),stem(nx,real(x3_8)),title('x3_8[n]'),xlabel('n'),grid on;
subplot(3,1,3),stem(nx,real(x3_9)),title('x3_9[n]'),xlabel('n'),grid on;
```



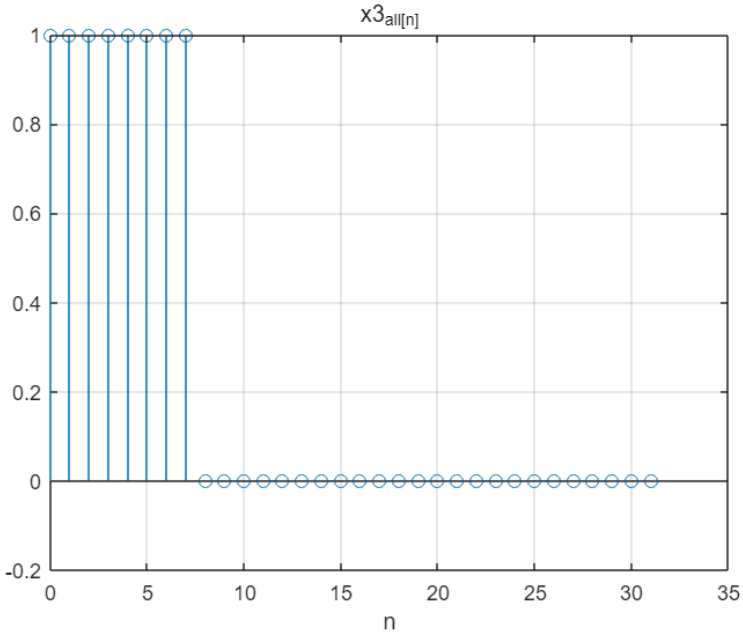

```
figure;
subplot(3,1,1),stem(nx,real(x3_10)),title('x3_10[n]'),xlabel('n'),grid on;
subplot(3,1,2),stem(nx,real(x3_11)),title('x3_11[n]'),xlabel('n'),grid on;
subplot(3,1,3),stem(nx,real(x3_12)),title('x3_12[n]'),xlabel('n'),grid on;
```




```
figure;
subplot(3,1,1),stem(nx,real(x3_13)),title('x3_13[n]'),xlabel('n'),grid on;
subplot(3,1,2),stem(nx,real(x3_14)),title('x3_14[n]'),xlabel('n'),grid on;
subplot(3,1,3),stem(nx,real(x3_15)),title('x3_15[n]'),xlabel('n'),grid on;
```




```
figure;
stem(nx,real(x3_all)),title('x3_all[n]'),xlabel('n'),grid on;
```



根据图像，可观察到，更多的谐波数量确实能够更加还原真实信号。

由于是离散信号，故不呈现吉布斯现象。

3.5i 写一个函数，给定x，n_init，最后检验

根据题目要求检验

```
clc,clear,close all;
dtfs([1 2 3 4],0)
```

```
ans = 1×4 complex
   2.5000 + 0.0000i  -0.5000 + 0.5000i  -0.5000 - 0.0000i  -0.5000 - 0.5000i
```

```
dtfs([1 2 3 4],1)
```

```
ans = 1×4 complex
   2.5000 + 0.0000i   0.5000 + 0.5000i   0.5000 + 0.0000i   0.5000 - 0.5000i
```

```
dtfs([1 2 3 4],-1)
```

```
ans = 1×4 complex
   2.5000 + 0.0000i  -0.5000 - 0.5000i   0.5000 + 0.0000i  -0.5000 + 0.5000i
```

```
dtfs([2 3 4 1],0)
```

```
ans = 1×4 complex
   2.5000 + 0.0000i  -0.5000 - 0.5000i   0.5000 + 0.0000i  -0.5000 + 0.5000i
```

```
dtfs([ones(1,4) zeros(1,4)],0)
```

```
ans = 1×8 complex
   0.5000 + 0.0000i   0.1250 - 0.3018i  -0.0000 - 0.0000i   0.1250 - 0.0518i   0.0000 - 0.0000i   0.1250 + 0.0518i   0.0000 -
```

```
dtfs([ones(1,4) zeros(1,4)],2)
```

```
ans = 1×8 complex
   0.5000 + 0.0000i  -0.3018 - 0.1250i   0.0000 + 0.0000i   0.0518 + 0.1250i   0.0000 - 0.0000i   0.0518 - 0.1250i  -0.0000 +
```

结果与示例相同。

3.10a fft运算次数。

ak=sum(x[n]*exp(-j*k*2pi/N*n))/N，做n次乘法，将x[n]和exp(-j*k*2pi/N*n)相乘；做n-1次加法，每个循环末尾将新的运算结果加进sum中（sum初值为0），循环外，再做一次除法，计算出结果。

故需要n+1次复数乘除法，n-1次复数加减法。

3.10b dtfs运算次数。

用etime记录运算时间。

```
clc,clear,close all;
N=[8 32 64 128 256];
```

```
dtfscomps=[];
for i=1:5
    x=(0.9).^[0:N(i)-1];
    f=@()dtfs(x,0);%fft运算
    dtfscomps=[dtfscomps timeit(f)];
end
dtfscomps
```

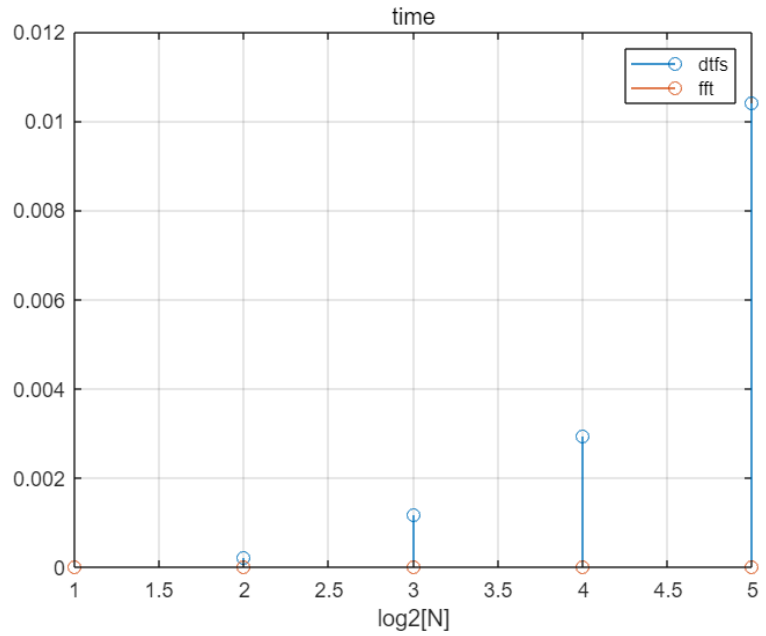
计算出不同的n下dtfs所需时间。

3.10c fft运算次数。

```
fftcomps=[];
for i=1:5
    x=(0.9).^[0:N(i)-1];
    f=@()fft(x,0);%fft运算
    fftcomps=[fftcomps timeit(f)];
end
```

警告：由于运行速度过快，F 的计时可能不准确。尝试对耗时更长的其他对象计时。

```
fftcomps
figure;
stem([1:5],dtfscomps),xlabel('log2[N]'),grid on;
hold on;
stem([1:5],fftcomps),xlabel('log2[N]'),grid on;
hold off;
legend('dtfs','fft'),title('time');
```



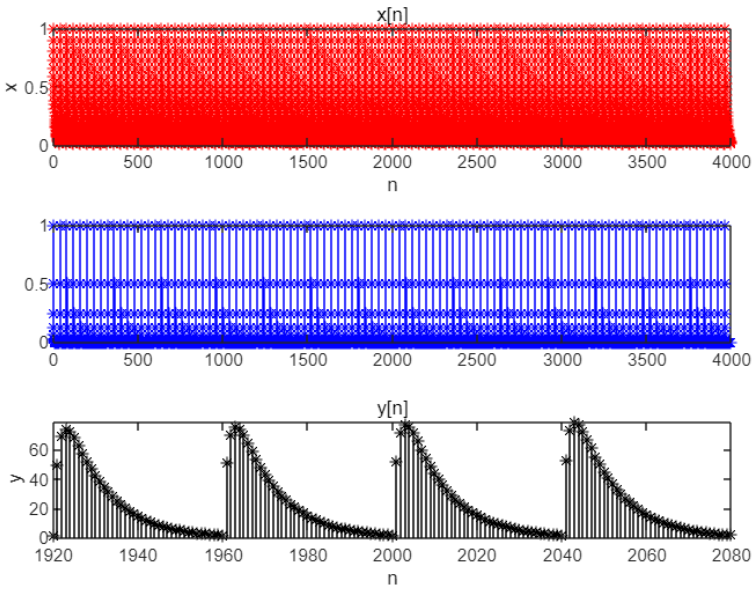
观察图像可知，fft方法较dtfs方法大幅度省时间。

3.10d x[n]与h[n]都是周期为N的DT信号，求y[n]=x[n]*h[n]的周期Ny，并指出卷积计算所需的运算次数。

分析：对于两个具有相同周期的信号，它们的卷积结果也将具有相同的周期。下面将举例说明这一点，并计算卷积所需的运算次数。

```
clc,clear,close all;
N=40;
%构造100个周期的x[n]与h[n]信号
x=zeros(1,100*N);
h=zeros(1,100*N);
for i=0:39
    x(i+1)=0.9.^i;
    h(i+1)=0.5.^i;
end
for i=40:3999
    x(i+1)=x(i-39);
    h(i+1)=h(i-39);
end
y=conv(x,h);
figure;
subplot(3,1,1);
stem(x,'r*');xlabel('n');ylabel('x');title('x[n]')
```

```
subplot(3,1,2);xlabel('n');ylabel('h');title('h[n]')
stem(h,'b*');
subplot(3,1,3);
stem(y,'k*');xlabel('n');ylabel('y');title('y[n]')
xlim([1920,2080]);
```



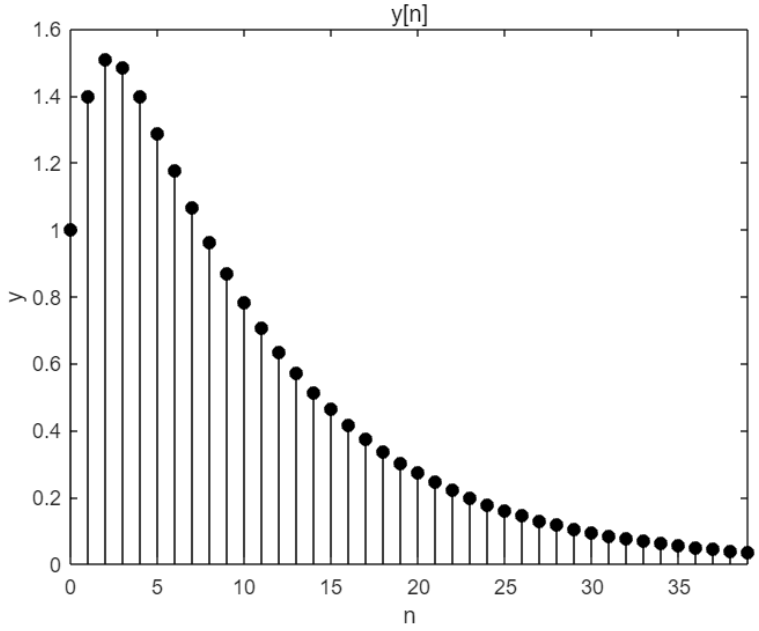
如图所示，y[n]的周期Ny=N=40（误差来源于x[n],h[n]并非无限长信号）。

由卷积公式可知，计算一个y[n]的值需要N次乘法运算与N-1次加法运算，所以计算一个周期内的y[n]共需要N*（2N-1）次运算，所以题目中需要O(N^2)数量级的运算次数的结论正确。

3.10e 记录N=40一个周期内卷积所需的运算次数。

分析：由于flops函数不可用，此处改为记录运算时间

```
x1=zeros(1,N);
h1=zeros(1,N);
for i=0:39
    x1(i+1)=0.9.^i;
    h1(i+1)=0.5.^i;
end
t0=clock;
for t=1:1000
    y1=conv([x1 x1],h1);
end
f40c = etime(clock,t0)/1000;
n=0:length(y1)-1;
figure;
stem(n,y1,'filled','k');title('y[n]');xlim([0,39]);xlabel('n');ylabel('y');title('y[n]');
```

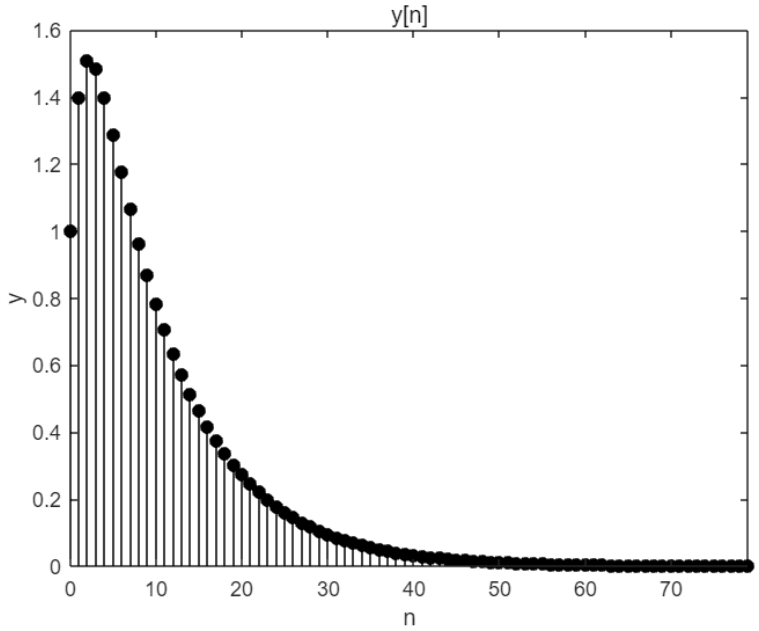



```
disp('运算时间为:'),disp(f40c);
```

```
运算时间为:
   6.0000e-06
```

3.10f 记录N=80一个周期内卷积所需的运算时间。

```
x2=zeros(1,2*N);
h2=zeros(1,2*N);
for i=0:79
    x2(i+1)=0.9.^i;
    h2(i+1)=0.5.^i;
end
t1=clock;
for t=1:1000
    y2=conv([x2 x2],h2);
end
f80c = etime(clock,t1)/1000;
n=0:length(y2)-1;
figure;
stem(n,y2,'filled','k');title('y[n]');xlim([0,79]);xlabel('n');ylabel('y');title('y[n]');
```

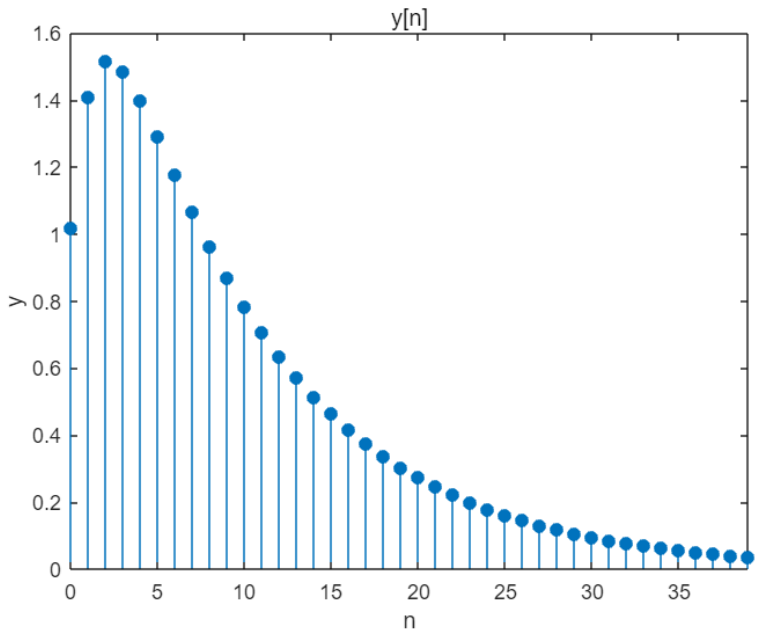



```
disp('运算时间为:'),disp(f80c);
```

```
运算时间为:
   6.0000e-06
```

3.10g 利用fft计算h[n]，x[n]的 DTFS，随后通过 DTFS 的卷积系数计算出 y[n]的DTFS，再通过ifft还原出 y[n]，作图与 3.10e 比较，并储存计算时间。

```
t2=clock;
for t=1:1000
    ax=fft(x1);
    ah=fft(h1);
    ay=ax.*ah;
    y3=ifft(ay);
end
f40f = etime(clock,t2)/1000;
figure;
n=0:39;
stem(n,y3,'filled');xlabel('n');xlim([0,39]);ylabel('y');title('y[n]');
```

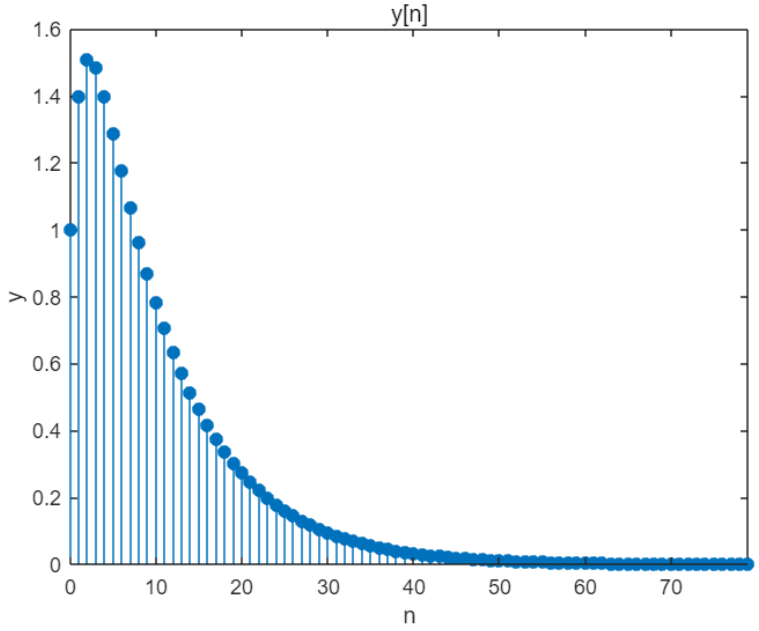



```
disp('运算时间为:'),disp(f40f);
```

```
运算时间为:
   9.0000e-06
```

3.10h 把周期变为N=80，进行3.10g中的操作后作图与3.10f作比较。

```
t3=clock;
for t=1:1000
    ax=fft(x2);
    ah=fft(h2);
    ay=ax.*ah;
    y4=ifft(ay);
end
f80f = etime(clock,t3)/1000;
figure;
n=0:79;
stem(n,y4,'filled');xlabel('n');xlim([0,79]);ylabel('y');title('y[n]');
```




```
disp('运算时间为:'),disp(f80f);
```

```
运算时间为:
   8.0000e-06
```

3.10i 计算 f40c/f40f 和 f80c/f80f，比较对于 N=40 和 N=80 ，哪种算法更高效。并判断当 N>80 时哪种算法更好。

```
disp('N=40时两种算法运算时间之比为：'),disp(f40c/f40f);
```

```
N=40时两种算法运算时间之比为：
    0.6667
```

```
disp('N=80时两种算法运算时间之比为：'),disp(f80c/f80f);
```

```
N=80时两种算法运算时间之比为：
    0.7500
```

f40c/f40f<1，f80c/f80f<1，预测N略大于80时直接卷积的算法更好。

可见对于小的N值，fft和ifft引入的额外开销可能会使其不如直接卷积快。但是N>>80时，由于fft与ifft函数运算次数更少，它的运算速度会快于直接卷积。

3.5i 函数部分

使用for循环遍历，带入公式，计算a

```
function a = dtfs(x,n_init)
    b=[];
    N=length(x);
    for i=n_init:N-1+n_init
        temp=0;
        for j=n_init:N-1+n_init
            h=x(j-n_init+1).*exp(-1i*2*pi*j*i/N);
            temp=temp+sum(h);
        end
        b=[b temp/N];
    end
    %得到了系数，但需调整至正确顺序
    if(n_init==0)
        a=b;
    end
    if(n_init>0)
        a=[b(end-n_init+1:end) b(1:end-n_init)];
    end
    if(n_init<0)
        a=[b(1-n_init:end) b(1:-n_init)];
    end
end
```

课堂参与证明：

自我评分：

应逸雯：10/10

陈薇羽：10/10