专业：信息工程
姓名：张青铭
学号：3200105426
日期：2022.11.13

浙江大学实验报告

课程名称：数字信号处理 指导老师：_____ 成绩：_____

实验名称：基 4-FFT 算法编程 实验类型：设计 同组学生姓名：——

一、实验目的和要求

FFT 是快速计算 DFT 的一类算法的总称。通过序列分解，用短序列的 DFT 代替长序列的 DFT，使得计算量大大下降。基 4-FFT 是混合基 FFT 的一个特例。

通过编写基 4-FFT 算法程序，加深对 FFT 思路、算法结构的理解。

二、实验内容和步骤

编写 16 点基 4-FFT 算法的 MATLAB 程序（studentname.m 文件）。

产生 16 点输入序列 x，用自己的学号作为前 10 点的抽样值，后面补 6 个零值抽样。算出 16 点频谱序列 X，用 stem(X)显示频谱图形。并进行 FFT 结果与 DFT 结构对比。

三、主要仪器设备

用 MATLAB。

四、操作方法和实验步骤

（参见“二、实验内容和步骤”）

五、实验数据记录和处理

5.1 基 4-FFT 算法思路、流图结构简述如下

5.1.1 算法思路

1.首先判断 n 是否为 4 的整数次幂，如不是则对序列进行补零；

2.对序列进行位序调整，第一组：0，2，4，…，第二组：1，3，5，…，进一步对第一组做相同分组处理，第一组：0，4，8，…，第二组：2，6，10，…，直到每组 4 个点排序和分组完成；

3.对每组做四点 DFT，再将相邻两组组合，乘以旋转因子，进行蝶形运算，不断重复，最终求得结果；

5.1.2 理论推导如下

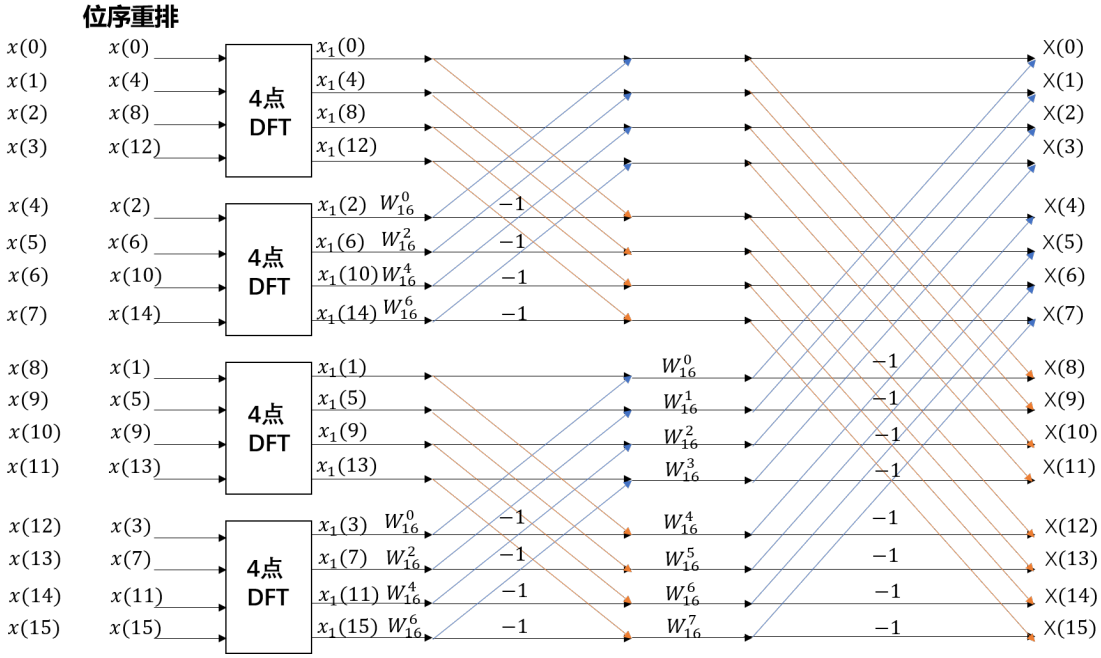
令序列$x(n)$的N点 DFT 结果为$X(k)$，且有$N=4^m$，按$((n))_4$的结果对序列x(n)的分组如下：

$$x^{(0)}(n)=x(4n)\quad X^{(0)}(k)=DFT_{4^{m-1}}\left\{x^{(0)}(n)\right\}$$
$$x^{(1)}(n)=x(4n+1)\quad X^{(1)}(k)=DFT_{4^{m-1}}\left\{x^{(1)}(n)\right\}$$
$$x^{(2)}(n)=x(4n+2)\quad X^{(2)}(k)=DFT_{4^{m-1}}\left\{x^{(2)}(n)\right\}$$
$$x^{(3)}(n)=x(4n+3)\quad X^{(3)}(k)=DFT_{4^{m-1}\left\{x^{(3)}(n)\right\}}$$
$$0\le n\le \frac{N}{4}-1\qquad 0\le k\le N-1=4^m-1$$

则有：

$$X(k)=X^{(0)}(k)+W_N^kX^{(1)}(k)+W_N^{2k}X^{(2)}(k)+W_N^{3k}X^{(3)}(k)$$
$$X\left(k+4^{m-1}\right)=X^{(0)}(k)-jW_N^kX^{(1)}(k)-W_N^{2k}X^{(2)}(k)+jW_N^{3k}X^{(3)}(k)$$
$$X\left(k+2\times4^{m-1}\right)=X^{(0)}(k)-W_N^kX^{(1)}(k)+W_N^{2k}X^{(2)}(k)-W_N^{3k}X^{(3)}(k)$$
$$X\left(k+3\times4^{m-1}\right)=X^{(0)}(k)+jW_N^kX^{(1)}(k)-W_N^{2k}X^{(2)}(k)-jW_N^{3k}X^{(3)}(k)$$

5.2 16 点基 4-FFT 算法的流图绘出如下



5.3 16 点基 4-FFT 算法的 MATLAB 程序（studentname.m）列出如下

5.3.1 四点 DFT 函数代码如下：

```
function y=DFT_4(x)
    n=0:3;
    y(1)=sum(x);
    y(2)=x*exp(-1j*pi/2*n).';
    y(3)=x*exp(-1j*pi*n).';
    y(4)=x*exp(-1j*pi*3/2*n).';
end
```

5.3.2 16 点 FFT 函数代码如下：

```
function y=FFT_16(x)
```

```
%位序重排和分组
x1=[x(1),x(5),x(9),x(13)];
x2=[x(3),x(7),x(11),x(15)];
x3=[x(2),x(6),x(10),x(14)];
x4=[x(4),x(8),x(12),x(16)];

%4点DFT结果
y1=DFT_4(x1);
y2_b=DFT_4(x2);
y3=DFT_4(x3);
y4_b=DFT_4(x4);

%乘以旋转因子
y2=[];
for i=1:4
    y2(i)=y2_b(i)*exp(-1j*pi/4*(i-1));
end
y4=[];
for i=1:4
    y4(i)=y4_b(i)*exp(-1j*pi/4*(i-1));
end

%蝶形运算
yy1=[];
for i=1:4
    yy1(i)=y1(i)+y2(i);
end
for j=5:8
    yy1(j)=y1(j-4)-y2(j-4);
end

yy2_b=[];
for i=1:4
    yy2_b(i)=y3(i)+y4(i);
end
for j=5:8
    yy2_b(j)=y3(j-4)-y4(j-4);
end

%乘以旋转因子
yy2=[];
for i=1:8
    yy2(i)=yy2_b(i)*exp(-1j*pi/8*(i-1));
end
```

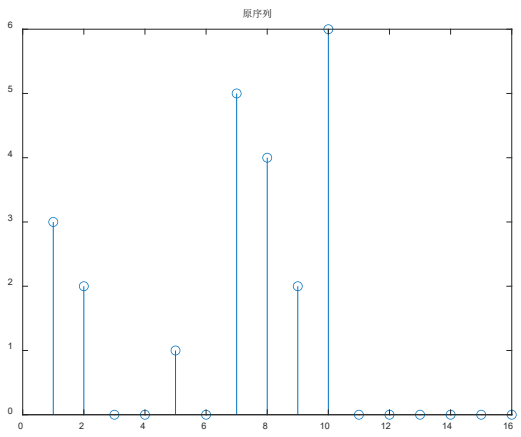
```
    %蝶形运算
    y=[];
    for i=1:8
        y(i)=yy1(i)+yy2(i);
    end
    for j=9:16
        y(j)=yy1(j-8)-yy2(j-8);
    end
end
```

5.3.3 主程序代码如下：

```
clear all;
close all;
n0=1:16;
n1=1:10;
x0=[3,2,0,0,1,0,5,4,2,6,0,0,0,0,0,0];
x1=[3,2,0,0,1,0,5,4,2,6];
y0=FFT_16(x0);%基4FFT
y1=fft(x1);%不填零DFT
y2=fft(x0);%16点DFT
%figure(1);
%stem(n,x);title('原序列');
figure(2);
subplot(3,2,1);stem(n0,real(y0));title(' fig.1 FFT变换实部序列');
subplot(3,2,2);stem(n0,imag(y0));title(' fig.2 FFT变换虚部序列');
subplot(3,2,3);stem(n1,real(y1));title(' fig.3 10点DFT变换实部序列');
subplot(3,2,4);stem(n1,imag(y1));title(' fig.4 10点DFT变换虚部序列');
subplot(3,2,5);stem(n0,real(y2));title(' fig.5 16点DFT变换实部序列');
subplot(3,2,6);stem(n0,imag(y2));title(' fig.6 16点DFT变换虚部序列');
```

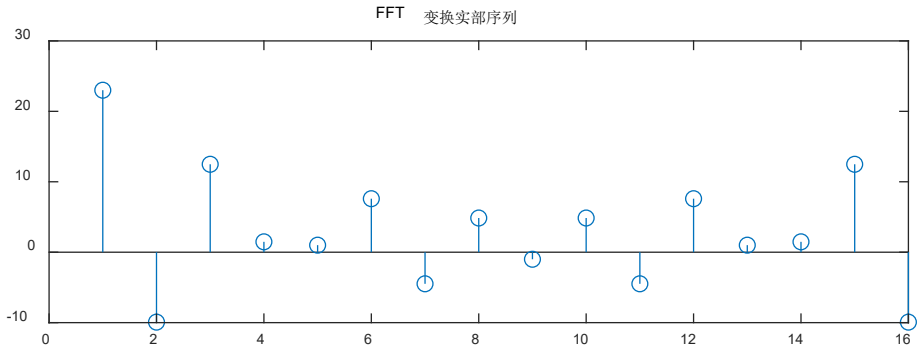
5.4 用自己的学号构成的输入序列为（列出数值，插入图形）

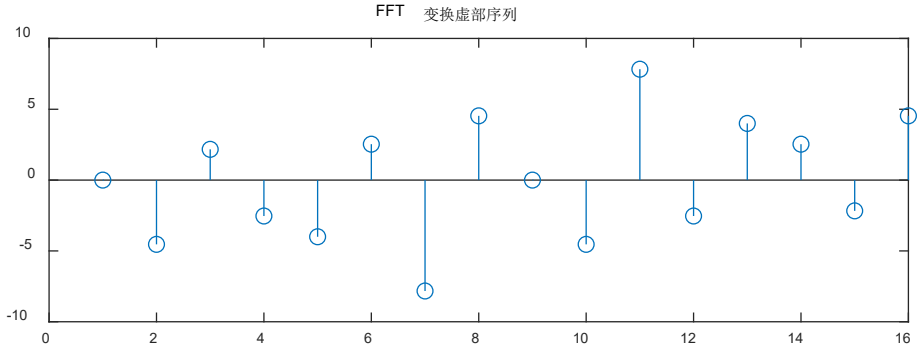
输入序列：3200105426000000



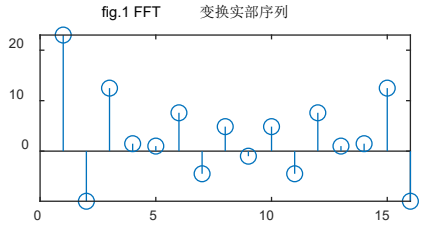
5.5 对应的输出频谱序列为（列出数值，插入图形）

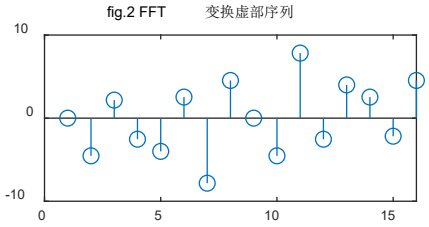
	1
1	23.0000 + 0.0000i
2	-9.9266 - 4.5355i
3	12.4853 + 2.1716i
4	1.4741 - 2.5355i
5	1.0000 - 4.0000i
6	7.5970 + 2.5355i
7	-4.4853 - 7.8284i
8	4.8555 + 4.5355i
9	-1.0000 + 0.0000i
10	4.8555 - 4.5355i
11	-4.4853 + 7.8284i
12	7.5970 - 2.5355i
13	1.0000 + 4.0000i
14	1.4741 + 2.5355i
15	12.4853 - 2.1716i
16	-9.9266 + 4.5355i

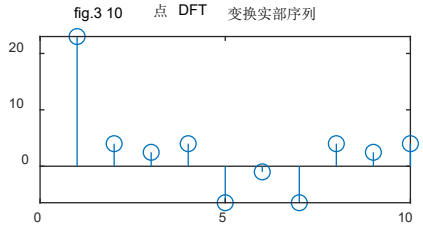


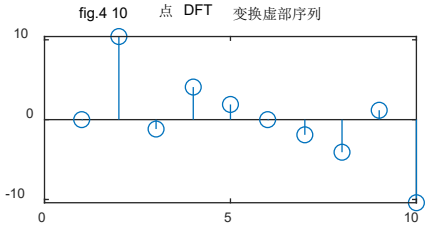


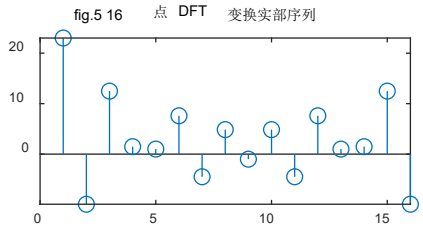
5.6 分别采用 10 点和 16 点（补 6 个零值后）DFT 计算频谱序列，与前面基 4-FFT 计算结果对比。

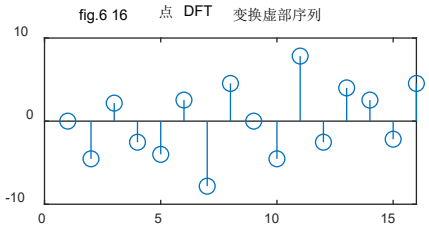












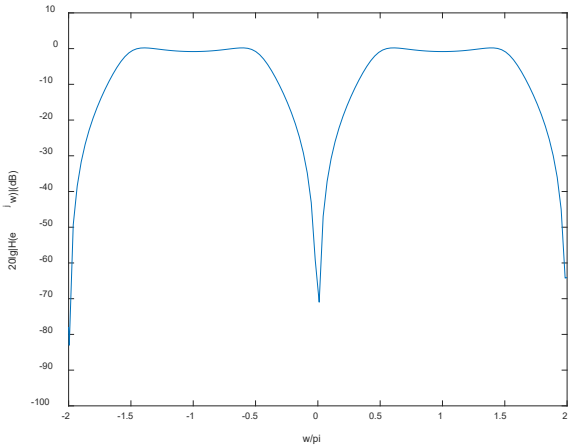
基 4 的 16 点 FFT，10 点 DFT，16 点 DFT 运算结果如下：

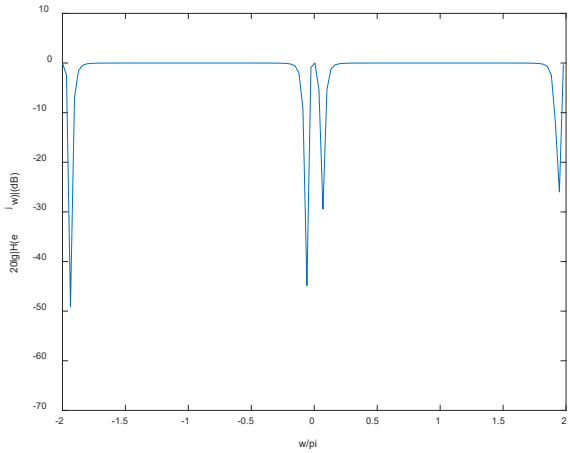
	1	2	3
1	23.0000 + 0.0000i	23.0000 + 0.0000i	23.0000 + 0.0000i
2	-9.9266 - 4.5355i	4.0000 + 10.4086i	-9.9266 - 4.5355i
3	12.4853 + 2.1716i	2.4721 - 1.1756i	12.4853 + 2.1716i
4	1.4741 - 2.5355i	4.0000 + 4.0817i	1.4741 - 2.5355i
5	1.0000 - 4.0000i	-6.4721 + 1.9021i	1.0000 - 4.0000i
6	7.5970 + 2.5355i	-1.0000 + 0.0000i	7.5970 + 2.5355i
7	-4.4853 - 7.8284i	-6.4721 - 1.9021i	-4.4853 - 7.8284i
8	4.8555 + 4.5355i	4.0000 - 4.0817i	4.8555 + 4.5355i
9	-1.0000 + 0.0000i	2.4721 + 1.1756i	-1.0000 + 0.0000i
10	4.8555 - 4.5355i	4.0000 - 10.4086i	4.8555 - 4.5355i
11	-4.4853 + 7.8284i	0.0000 + 0.0000i	-4.4853 + 7.8284i
12	7.5970 - 2.5355i	0.0000 + 0.0000i	7.5970 - 2.5355i
13	1.0000 + 4.0000i	0.0000 + 0.0000i	1.0000 + 4.0000i
14	1.4741 + 2.5355i	0.0000 + 0.0000i	1.4741 + 2.5355i
15	12.4853 - 2.1716i	0.0000 + 0.0000i	12.4853 - 2.1716i
16	-9.9266 + 4.5355i	0.0000 + 0.0000i	-9.9266 + 4.5355i

可以发现：基 4 的 16 点 FFT 运算结果与 16 点 DFT 运算结果一致，结果均与 10 点 DFT 不同运算结果不同

六、实验结果与分析

经过上述序列和图像对比，matlab 自带的 fft 函数与基 4-fft 算法结果一致。算法上，基 4 的最小运算单元是 4 点 DFT，但之后的蝶形运算和基 2 算法没有本质区别。


10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
w/pi
20log|H(e
|(dB)


10
0
-10
-20
-30
-40
-50
-60
-70
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
w/pi
20log|H(e
|(dB)