专业：信息工程
姓名：黄嘉欣
学号：3190102060
日期：2021.10.25

浙江大学实验报告

课程名称：数字信号处理　　成绩：

实验名称：FFT 算法编程　　实验类型：设计性实验

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 及基 2-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。

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。

编写基 2DIT-FFT 或者基 2DIF-FFT 程序，验证基 4-FFT 的频谱序列计算结果。

撰写实验报告。

三、主要仪器设备

用 MATLAB。

四、实验数据记录和处理

4.1 基 4-FFT 算法思路、流图结构简述如下。

解：① 算法思路：与基 2-FFT 算法类似，基 4-FFT 算法在时域上按n的特点对$x(n)$进行不断的分组以及位序调整，进而通过逐级蝶形复合处理，以低点数 DFT 间接地完成高点数 DFT 的计算，达到降低运算量及节省存储空间的目的，其推导过程如下：

设序列$x(n)$的N点 DFT 结果为$X(k)$，且$N=4^m$，则按$((n))_4$的结果对序列$x(n)$进行分组，有：

$$x^{(0)}(n)=x(4n) \qquad X^{(0)}(k)=DFT_{4^{m-1}}\{x^{(0)}(n)\}$$

$$x^{(1)}(n) = x(4n+1) \qquad X^{(1)}(k) = DFT_{4^{m-1}}\{x^{(1)}(n)\}$$
$$x^{(2)}(n) = x(4n+2) \qquad X^{(2)}(k) = DFT_{4^{m-1}}\{x^{(2)}(n)\}$$
$$x^{(3)}(n) = x(4n+3) \qquad X^{(3)}(k) = DFT_{4^{m-1}}\{x^{(3)}(n)\}$$

其中$0 \le n \le \frac{N}{4} - 1$。

由 DFT：

$$X(k) = \sum_{n=0}^{N} x(n) W_N^{nk}$$
$$= \sum_{l=0}^{4^{m-1}-1} x(4l)\, W_N^{4lk} + \sum_{l=0}^{4^{m-1}-1} x(4l+1)\, W_N^{(4l+1)k} + \sum_{l=0}^{4^{m-1}-1} x(4l+2)\, W_N^{(4l+2)k}$$
$$+ \sum_{l=0}^{4^{m-1}-1} x(4l+3)\, W_N^{(4l+3)k}$$
$$= \sum_{l=0}^{4^{m-1}-1} x^{(0)}(l)\, W_{4^{m-1}}^{lk} + W_N^k \sum_{l=0}^{4^{m-1}-1} x^{(1)}(l)\, W_{4^{m-1}}^{lk} + W_N^{2k} \sum_{l=0}^{4^{m-1}-1} x^{(2)}(l)\, W_{4^{m-1}}^{lk}$$
$$+W_N^{3k} \sum_{l=0}^{4^{m-1}-1} x^{(3)}(l)\, W_{4^{m-1}}^{lk}$$
$$= X^{(0)}((k))_{4^{m-1}} + W_N^k X^{(1)}((k))_{4^{m-1}} + W_N^{2k} X^{(2)}((k))_{4^{m-1}} + W_N^{3k} X^{(3)}((k))_{4^{m-1}}$$

令$0 \le k \le 4^{m-1} - 1$，则有式 4.1.1：

$$\begin{cases} X(k) = X^{(0)}(k) + W_N^k X^{(1)}(k) + W_N^{2k} X^{(2)}(k) + W_N^{3k} X^{(3)}(k) \\ X(k+4^{m-1}) = X^{(0)}(k) - jW_N^k X^{(1)}(k) - W_N^{2k} X^{(2)}(k) + jW_N^{3k} X^{(3)}(k) \\ X(k+2\times 4^{m-1}) = X^{(0)}(k) - W_N^k X^{(1)}(k) + W_N^{2k} X^{(2)}(k) - W_N^{3k} X^{(3)}(k) \\ X(k+3\times 4^{m-1}) = X^{(0)}(k) + jW_N^k X^{(1)}(k) - W_N^{2k} X^{(2)}(k) - jW_N^{3k} X^{(3)}(k) \end{cases}$$

此时即可将一个N点的 DFT 运算简化为 4 个$\frac{N}{4}$点的 DFT 运算和一级蝶形复合。同理，将此$\frac{N}{4}$点的序列继续分组，可将 1 个N点的 DFT 运算简化为 16 个$\frac{N}{16}$点的 DFT 运算和两级蝶形复合。由于$N = 4^m$，所以可将序列$x(n)$连续进行$(m-1)$次分组，直到得到4^{m-1}个长度为 4 的序列为止。此时，一个4^m点 DFT 的计算转化为4^{m-1}个 4 点 DFT 的计算和$(m-1)$级蝶形复合，而 4 点 DFT 的计算为：

$$\begin{cases} Y(0) = y(0) + y(1) + y(2) + y(3) \\ Y(1) = y(0) - jy(1) - y(2) + jy(3) \\ Y(2) = y(0) - y(1) + y(2) - y(3) \\ Y(3) = y(0) + jy(1) - y(2) - jy(3) \end{cases}$$

显然，其也为蝶形运算。因此，一个4^m点 DFT 的计算实际上可由原序列$x(n)$出发，

经过分组及位序调整，由 4 点 DFT 开始，经过m级蝶形复合得到(最初的 4 点 DFT 也视为一级蝶形复合)，此即为基 4-FFT 算法的总体思路。若设第l级蝶形复合的输入为$x_{l-1}(k)$，输出为$x_l(k)$，则可给出蝶形运算的一般形式为：

$$\begin{cases} x_l(i)=x_{l-1}(i)+W_N^{4^{m-1}i}x_{l-1}(i+4^{l-1})+W_N^{4^{m-1}2i}x_{l-1}(i+2\times 4^{l-1}) \\ \qquad +W_N^{4^{m-1}3i}x_{l-1}(i+3\times 4^{l-1}) \\ x_l(i+4^{l-1})=x_{l-1}(i)-jW_N^{4^{m-1}i}x_{l-1}(i+4^{l-1})-W_N^{4^{m-1}2i}x_{l-1}(i+2\times 4^{l-1}) \\ \qquad +jW_N^{4^{m-1}3i}x_{l-1}(i+3\times 4^{l-1}) \\ x_l(i+2\times 4^{l-1})=x_{l-1}(i)-W_N^{4^{m-1}i}x_{l-1}(i+4^{l-1})+W_N^{4^{m-1}2i}x_{l-1}(i+2\times 4^{l-1}) \\ \qquad -W_N^{4^{m-1}3i}x_{l-1}(i+3\times 4^{l-1}) \\ x_l(i+3\times 4^{l-1})=x_{l-1}(i)+jW_N^{4^{m-1}i}x_{l-1}(i+4^{l-1})-W_N^{4^{m-1}2i}x_{l-1}(i+2\times 4^{l-1}) \\ \qquad -jW_N^{4^{m-1}3i}x_{l-1}(i+3\times 4^{l-1}) \end{cases}$$

② 流图结构：

如图，为式 4.1.1 所对应的信号流图：

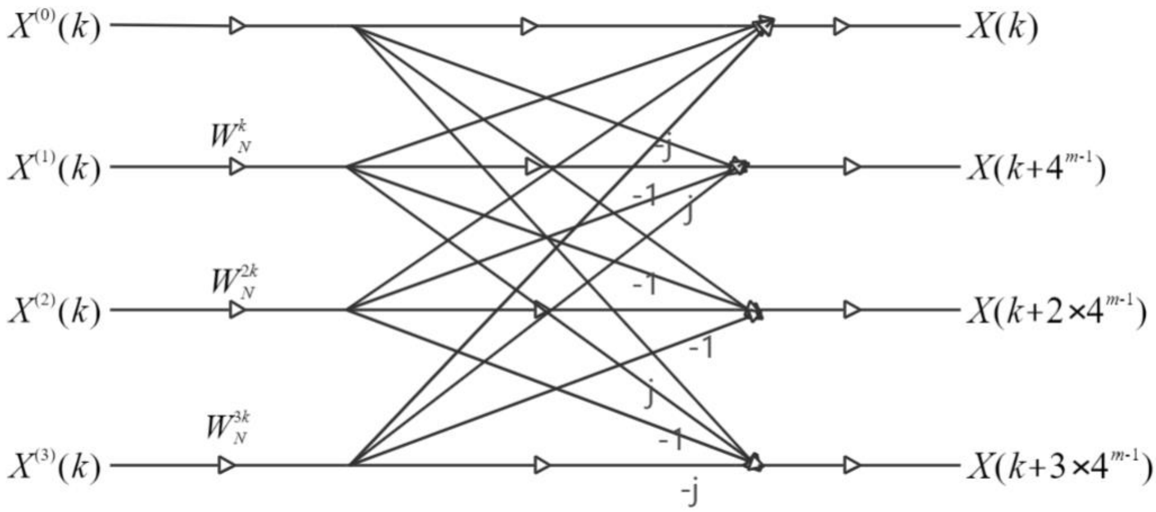

图 4.1.1 基 4-FFT 中的蝶形运算

显然，N点 DFT 转化为了 4 个$\frac{N}{4}$点 DFT 和 3 次复乘、12 次复加，此即为单次蝶形运算的流图结构；

由①中思路，基 4-FFT 的总体计算流程为：

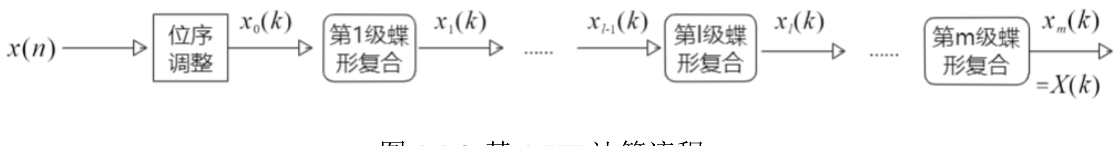

图 4.1.2 基 4-FFT 计算流程

即将N点序列进行$(m-1)$次分组，由 4 点 DFT 开始，逐级进行蝶形复合(共m级)，直到求得最终结果$X(k)$。

4.2 16 点基 4-FFT 算法的流图绘出如下。

解：由 4.1 中所述原理思路，16 点基 4-FFT 算法的流图如图所示：

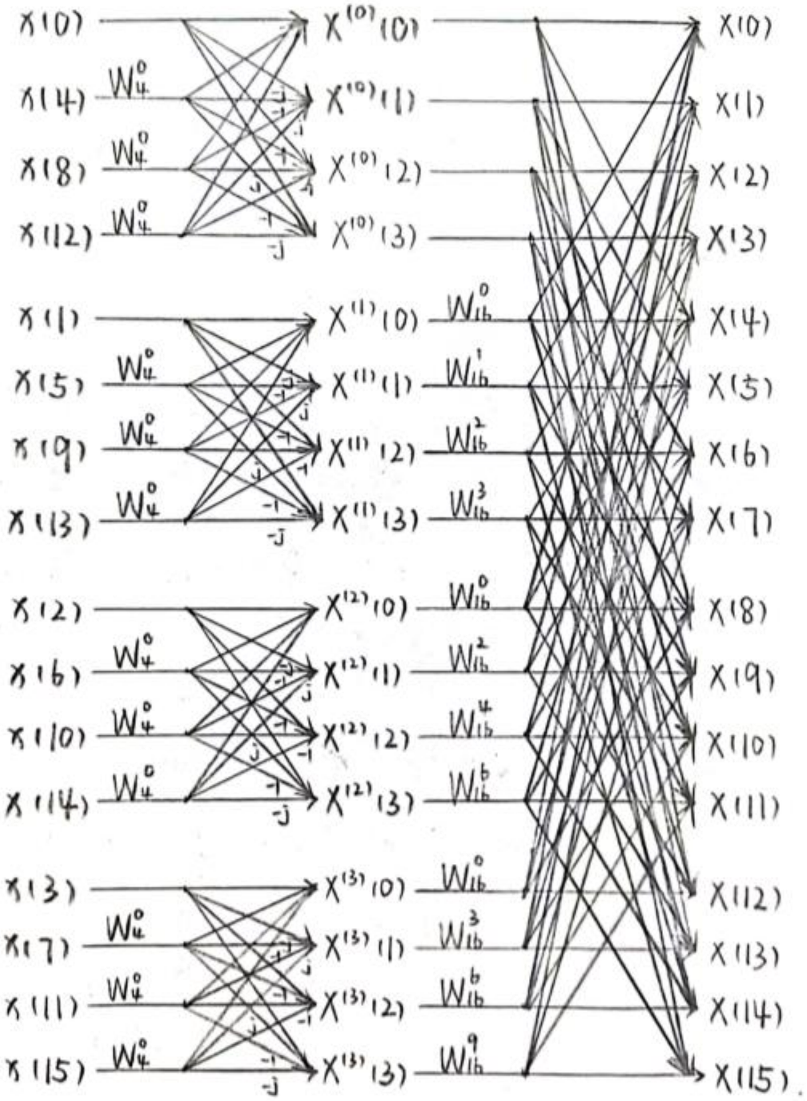

图 4.2 16 点基 4-FFT 流图

由于图形较为稠密，在图中第二级蝶形复合中，箭头上的系数$-j$, -1, j均已省略，每个箭头上具体系数可见图 4.1.1。

4.3 16 点基 4-FFT 算法的 MATLAB 程序（studentnameB4.m）如下（列出）：

解：

```
% studentnameB4.m
clc;
clear;

% 输入序列，前10点为学号，后补6个0
x = [3 1 9 0 1 0 2 0 6 0 0 0 0 0 0 0];
N = 16; % 点数
n = 0:N-1;
figure; stem(n,x,'filled'); % 输入序列图形
xlabel('n'); ylabel('x(n)'); title('输入序列');

XDFT = dftmtx(N)*x'; % 输入序列的DFT
X = zeros(16,1); % 初始化输出的频谱序列
W = exp(-1j*2*pi/N); % W_16^1
W4 = dftmtx(4); % 4点DFT系数矩阵
x0 = [x(1);x(5);x(9);x(13)]; % 分组
x1 = [x(2);x(6);x(10);x(14)];
x2 = [x(3);x(7);x(11);x(15)];
x3 = [x(4);x(8);x(12);x(16)];

% 第一级蝶形复合
X0 = W4*x0; X1 = W4*x1; X2 = W4*x2; X3 = W4*x3;
% 第二级蝶形复合
for k = 0:3 % 每次循环求出四点
    t = W4*[X0(k+1);W^k*X1(k+1);W^(2*k)*X2(k+1);W^(3*k)*X3(k+1)];
    X(k+1) = t(1); X(k+5) = t(2); X(k+9) = t(3); X(k+13) = t(4);
end

figure;
subplot(2,2,1); stem(n,real(XDFT),'filled');
xlabel('k'); ylabel('Re\{X\}'); title('DFT所得频谱实部');
subplot(2,2,2); stem(n,imag(XDFT),'filled');
xlabel('k'); ylabel('Im\{X\}'); title('DFT所得频谱虚部');
subplot(2,2,3); stem(n,real(X),'filled');
xlabel('k'); ylabel('Re\{X\}'); title('基4-FFT频谱实部');
subplot(2,2,4); stem(n,imag(X),'filled');
xlabel('k'); ylabel('Im\{X\}'); title('基4-FFT频谱虚部');


% 将数据写入文件
f = fopen('studentnameB4.txt','w');
for i = 1:16
    if imag(X(i))<0
        fprintf(f,'X(%d) = %f-j%f\n',i-1,real(X(i)),-imag(X(i)));
    else
```

```
        fprintf(f,'X(%d)  = %f+j%f\n',i-1,real(X(i)),imag(X(i)));
    end
end
```

4.4 16 点基 2-FFT 算法的 MATLAB 程序（studentnameB2.m）如下（列出）：

解：① 基 2DIT-FFT：

```
% studentnameB2_DIT.m
clc;
clear;

% 输入序列，前10点为学号，后补6个0
x = [3 1 9 0 1 0 2 0 6 0 0 0 0 0 0 0];
N = 16; % 点数
n = 0:N-1;
figure; stem(n,x,'filled'); % 输入序列图形
xlabel('n'); ylabel('x(n)'); title('输入序列');

XDFT = dftmtx(N)*x'; % 输入序列的DFT
X1 = zeros(2,8); % 初始化第一级蝶形运算结果
X2 = zeros(4,4); % 初始化第二级蝶形运算结果
X3 = zeros(8,2); % 初始化第三级蝶形运算结果

X = zeros(16,1); % 初始化输出的频谱序列
W = exp(-1j*2*pi/N); % W_16^1
W2 = dftmtx(2); % 2点DFT系数矩阵
x0 = [x(1);x(9)]; x1 = [x(5);x(13)]; % 分组
x2 = [x(3);x(11)]; x3 = [x(7);x(15)];
x4 = [x(2);x(10)]; x5 = [x(6);x(14)];
x6 = [x(4);x(12)]; x7 = [x(8);x(16)];

% 第一级蝶形复合
X1(:,1) = W2*x0; X1(:,2) = W2*x1; X1(:,3) = W2*x2; X1(:,4) = W2*x3;
X1(:,5) = W2*x4; X1(:,6) = W2*x5; X1(:,7) = W2*x6; X1(:,8) = W2*x7;
% 第二级蝶形复合
for k = 0:1
    for m = 0:3 % 每次循环求出两点
        t1 = W2*[X1(k+1,2*m+1);W^(4*k)*X1(k+1,2*m+2)];
        X2(k+1,m+1)  = t1(1); X2(k+3,m+1)  = t1(2);
    end
end
% 第三级蝶形复合
for k = 0:3
    for m = 0:1 % 每次循环求出两点
```

```
            t2 = W2*[X2(k+1,2*m+1);W^(2*k)*X2(k+1,2*m+2)];
            X3(k+1,m+1)  = t2(1); X3(k+5,m+1)  = t2(2);
        end
end
% 第四级蝶形复合
for k = 0:7
        t3 = W2*[X3(k+1,1);W^k*X3(k+1,2)];
        X(k+1) = t3(1); X(k+9)  = t3(2);
end

figure;
subplot(2,2,1); stem(n,real(XDFT),'filled');
xlabel('k'); ylabel('Re\{X\}'); title('DFT所得频谱实部');
subplot(2,2,2); stem(n,imag(XDFT),'filled');
xlabel('k'); ylabel('Im\{X\}'); title('DFT所得频谱虚部');
subplot(2,2,3); stem(n,real(X),'filled');
xlabel('k'); ylabel('Re\{X\}'); title('基2DIT-FFT频谱实部');
subplot(2,2,4); stem(n,imag(X),'filled');
xlabel('k'); ylabel('Im\{X\}'); title('基2DIT-FFT频谱虚部');

% 将数据写入文件
f = fopen('studentnameB2_DIT.txt','w');
for i = 1:16
        if imag(X(i))<0
            fprintf(f,'X(%d)  = %f-j%f\n',i-1,real(X(i)),-imag(X(i)));
        else
            fprintf(f,'X(%d)  = %f+j%f\n',i-1,real(X(i)),imag(X(i)));
        end
end
```

② 基 2DIF-FFT：

```
% studentnameB2_DIF.m
clc;
clear;

% 输入序列，前10点为学号，后补6个0
x = [3 1 9 0 1 0 2 0 6 0 0 0 0 0 0 0];
N = 16; % 点数
n = 0:N-1;
figure; stem(n,x,'filled'); % 输入序列图形
xlabel('n'); ylabel('x(n)'); title('输入序列');

XDFT = dftmtx(N)*x'; % 输入序列的DFT
```

```
X1 = zeros(8,2); % 初始化第一级蝶形运算结果
X2 = zeros(4,4); % 初始化第二级蝶形运算结果
X3 = zeros(2,8); % 初始化第三级蝶形运算结果
X4 = zeros(2,8); % 初始化第三级蝶形运算结果
X = zeros(1,16); % 初始化输出的频谱序列
W = exp(-1j*2*pi/N); % W_16^1
W2 = dftmtx(2); % 2点DFT系数矩阵

% 第一级蝶形复合
for k = 0:7
    t1 = W2*[x(k+1);x(k+9)];
    X1(k+1,1) = t1(1); X1(k+1,2) = t1(2);
end
% 第二级蝶形复合
for k = 0:3
    t2 = W2*[X1(k+1,1);X1(k+5,1)];
    X2(k+1,1) = t2(1); X2(k+1,2) = t2(2);
    t2 = W2*[W^k*X1(k+1,2);W^(k+4)*X1(k+5,2)];
    X2(k+1,3) = t2(1); X2(k+1,4) = t2(2);
end
% 第三级蝶形复合
for k = 0:1
    t3 = W2*[X2(k+1,1);X2(k+3,1)];
    X3(k+1,1) = t3(1); X3(k+1,2) = t3(2);
    t3 = W2*[W^(2*k)*X2(k+1,2);W^(2*k+4)*X2(k+3,2)];
    X3(k+1,3) = t3(1); X3(k+1,4) = t3(2);
    t3 = W2*[X2(k+1,3);X2(k+3,3)];
    X3(k+1,5) = t3(1); X3(k+1,6) = t3(2);
    t3 = W2*[W^(2*k)*X2(k+1,4);W^(2*k+4)*X2(k+3,4)];
    X3(k+1,7) = t3(1); X3(k+1,8) = t3(2);
end
% 第四级蝶形复合
X4(:,1) = W2*X3(:,1); X4(:,2) = W2*[X3(1,2);W^4*X3(2,2)];
X4(:,3) = W2*X3(:,3); X4(:,4) = W2*[X3(1,4);W^4*X3(2,4)];
X4(:,5) = W2*X3(:,5); X4(:,6) = W2*[X3(1,6);W^4*X3(2,6)];
X4(:,7) = W2*X3(:,7); X4(:,8) = W2*[X3(1,8);W^4*X3(2,8)];

% 位序调整
X(1) = X4(1,1); X(2) = X4(1,5); X(3) = X4(1,3); X(4) = X4(1,7);
X(5) = X4(1,2); X(6) = X4(1,6); X(7) = X4(1,4); X(8) = X4(1,8);
X(9) = X4(2,1); X(10) = X4(2,5); X(11) = X4(2,3); X(12) = X4(2,7);
X(13) = X4(2,2); X(14) = X4(2,6); X(15) = X4(2,4); X(16) = X4(2,8);

figure;
```

```
subplot(2,2,1); stem(n,real(XDFT),'filled');
xlabel('k'); ylabel('Re\{X\}'); title('DFT所得频谱实部');
subplot(2,2,2); stem(n,imag(XDFT),'filled');
xlabel('k'); ylabel('Im\{X\}'); title('DFT所得频谱虚部');
subplot(2,2,3); stem(n,real(X),'filled');
xlabel('k'); ylabel('Re\{X\}'); title('基2DIF-FFT频谱实部');
subplot(2,2,4); stem(n,imag(X),'filled');
xlabel('k'); ylabel('Im\{X\}'); title('基2DIF-FFT频谱虚部');

% 将数据写入文件
f = fopen('studentnameB2_DIF.txt','w');
for i = 1:16
    if imag(X(i))<0
        fprintf(f,'X(%d)  = %f-j%f\n',i-1,real(X(i)),-imag(X(i)));
    else
        fprintf(f,'X(%d)  = %f+j%f\n',i-1,real(X(i)),imag(X(i)));
    end
end
```

4.5 用自己的学号构成的输入序列为（列出数值，画出图形）：

解：输入序列为：$x(n) = \{3,1,9,0,1,0,2,0,6,0,0,0,0,0,0,0\}$

其图形为：

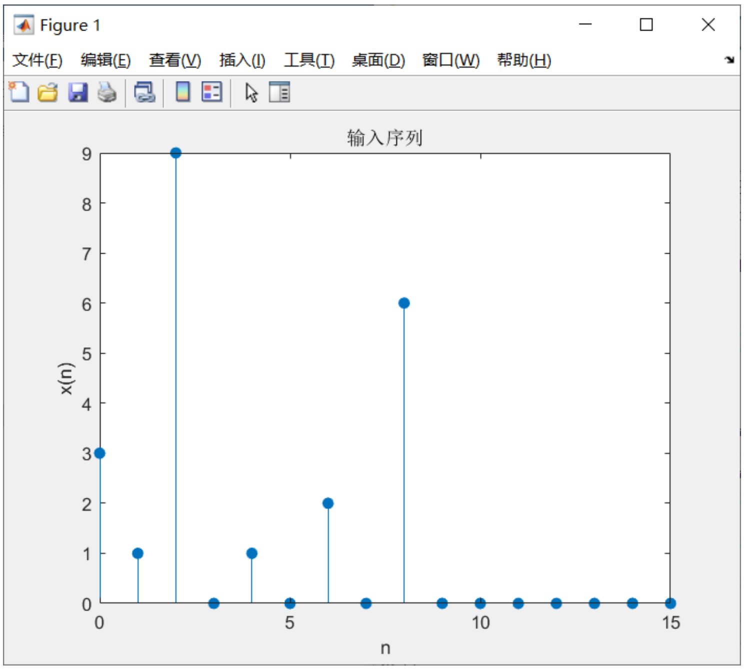

图 4.5 输入序列

4.6 对应的基 4-FFT 输出频谱序列为（列出数值，画出图形）：

解：基 4-FFT 输出频谱序列为：

X(0) = 22.000000+j0.000000
X(1) = 2.873627-j9.160858
X(2) = 8.707107-j7.707107
X(3) = -7.567064-j7.702054
X(4) = -1.000000-j1.000000
X(5) = -8.332431+j5.854295
X(6) = 7.292893+j6.292893
X(7) = 1.025868+j8.395491
X(8) = 20.000000+j0.000000
X(9) = 1.025868-j8.395491
X(10) = 7.292893-j6.292893
X(11) = -8.332431-j5.854295
X(12) = -1.000000+j1.000000
X(13) = -7.567064+j7.702054
X(14) = 8.707107+j7.707107
X(15) = 2.873627+j9.160858

其图形为：

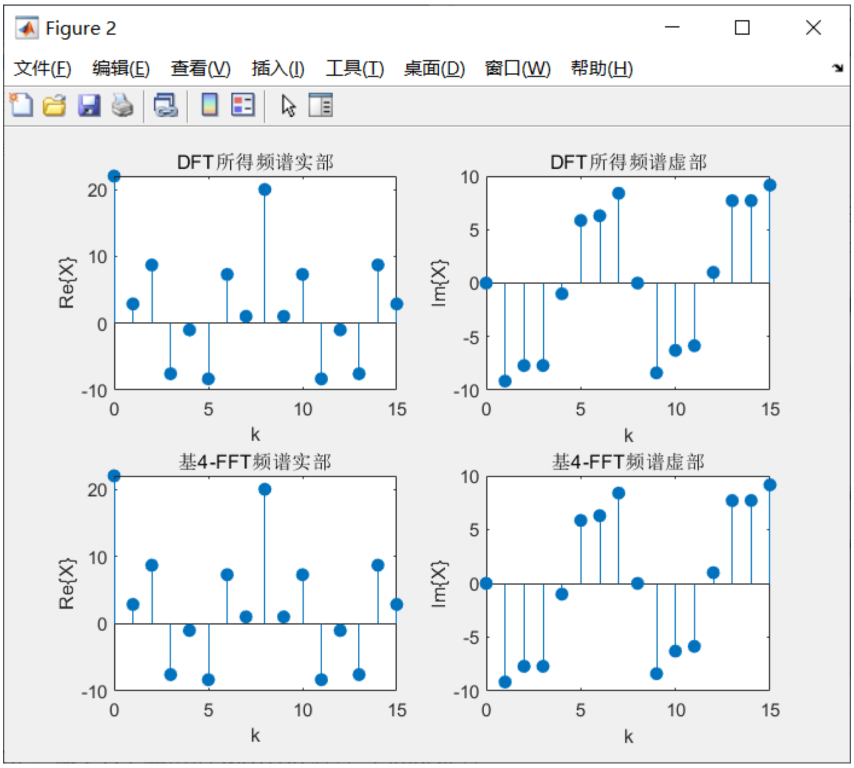

图 4.6 基 4-FFT 输出频谱序列

4.7 对应的基 2-FFT 输出频谱序列为（列出数值，画出图形）：

解：① 基 2DIT-FFT：

输出频谱序列为：

X(0) = 22.000000+j0.000000
X(1) = 2.873627-j9.160858
X(2) = 8.707107-j7.707107
X(3) = -7.567064-j7.702054
X(4) = -1.000000-j1.000000
X(5) = -8.332431+j5.854295
X(6) = 7.292893+j6.292893
X(7) = 1.025868+j8.395491
X(8) = 20.000000+j0.000000
X(9) = 1.025868-j8.395491
X(10) = 7.292893-j6.292893
X(11) = -8.332431-j5.854295
X(12) = -1.000000+j1.000000
X(13) = -7.567064+j7.702054
X(14) = 8.707107+j7.707107
X(15) = 2.873627+j9.160858

其图形为：

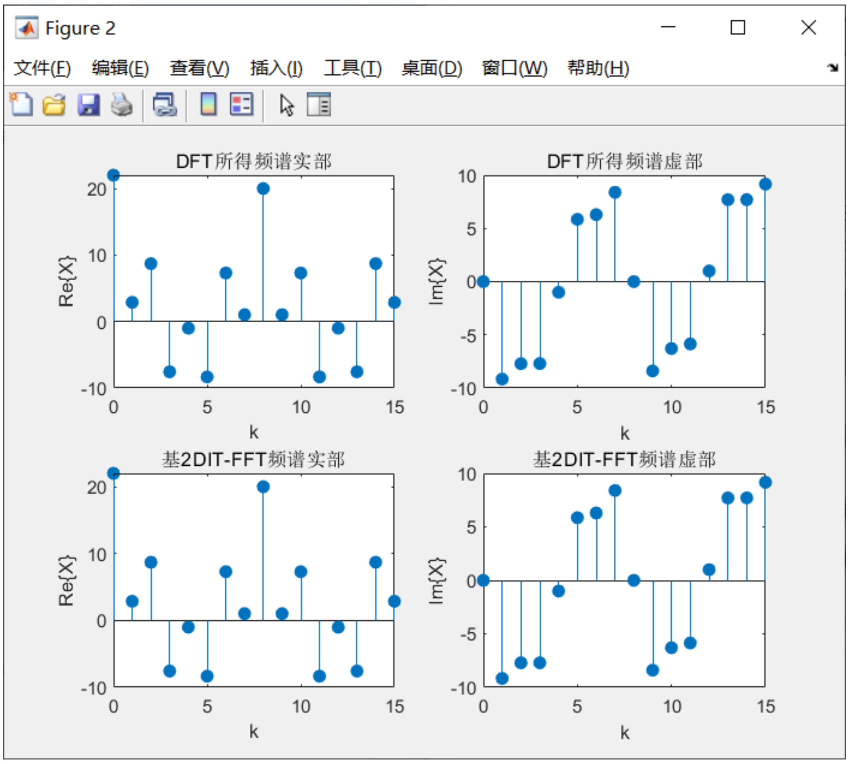

图 4.7.1 基 2DIT-FFT 输出频谱序列

② 基 2DIF-FFT：

输出频谱序列为：

X(0) = 22.000000+j0.000000
X(1) = 2.873627-j9.160858
X(2) = 8.707107-j7.707107
X(3) = -7.567064-j7.702054
X(4) = -1.000000-j1.000000
X(5) = -8.332431+j5.854295
X(6) = 7.292893+j6.292893
X(7) = 1.025868+j8.395491
X(8) = 20.000000+j0.000000
X(9) = 1.025868-j8.395491
X(10) = 7.292893-j6.292893
X(11) = -8.332431-j5.854295
X(12) = -1.000000+j1.000000
X(13) = -7.567064+j7.702054
X(14) = 8.707107+j7.707107
X(15) = 2.873627+j9.160858

其图形为：

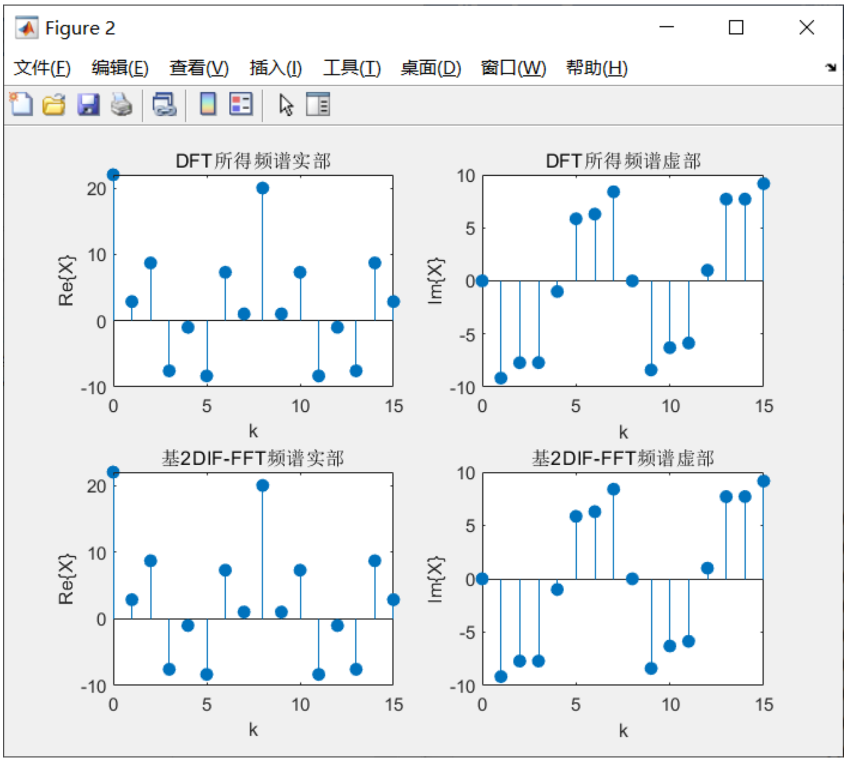

图 4.7.2 基 2DIF-FFT 输出频谱序列

五、实验结果与分析

由图 4.6、图 4.7.1、图 4.7.2，可以发现，通过基 4-FFT 算法、基 2DIT-FFT 算法及基 2DIF-FFT 算法得到的频谱序列，与利用 MATLAB 直接计算得到的 DFT 序列完全一致。将

如下所示计算值与前面各算法得到的频谱值进行对比，两者也完全相同。

```
XDFT =

  22.0000 + 0.0000i
   2.8736 - 9.1609i
   8.7071 - 7.7071i
  -7.5671 - 7.7021i
  -1.0000 - 1.0000i
  -8.3324 + 5.8543i
   7.2929 + 6.2929i
   1.0259 + 8.3955i
  20.0000 + 0.0000i
   1.0259 - 8.3955i
   7.2929 - 6.2929i
  -8.3324 - 5.8543i
  -1.0000 + 1.0000i
  -7.5671 + 7.7021i
   8.7071 + 7.7071i
   2.8736 + 9.1609i
```

一方面，上述结果表明此次编写的基 4-FFT 程序及其结果是正确的；另一方面，也说明基 4-FFT 与基 2-FFT 在本质上是相同的，即都是通过逐级的蝶形复合处理，间接完成高点数 DFT 的计算，由此达到降低运算量以及节省存储空间的目的。

六、讨论

此次实验，我们利用 MATLAB 编程，自行编写了基 4-FFT 和基 2-FFT 的算法程序，既加深了我们对快速傅里叶算法的理解和掌握，也对几者之间的关系进行了进一步的思考，对于我们掌握课堂所学，作用匪浅。

在编写基 4-FFT 算法的过程当中，由于逻辑上的粗心，首次得到的结果与标准 DFT 结果存在偏差。为了修正错误，我重新整理了 FFT 算法的流程，并对程序进行了校准，但并未如愿将错误找出。在此基础上，我通过手算每一级蝶形复合的结果，逐级比较、调试，发现自己在调整第三级蝶形运算位序时错误复制了数组内容。虽然整个调试过程花费了我不少的时间，但我也因此加深了对 FFT 的认识，对计算时的规律掌握得更加熟练，可谓“因祸得福”。

总的来说，虽然基 4-FFT 与基 2-FFT 在算法上存在差异，但两者的流程与思想都是一致的，即通过逐级蝶形复合，降低高点数 DFT 的计算量，提高运算速度。比较来看可以发现，基 4-FFT 的蝶形复合级数更少，运算量更小，因此可以算作基 2-FFT 的“进阶”版本。除此之外，对于基 4-FFT 与基 2-FFT 在点数上的限制，我们可以进一步采用混合基等运算方法，提高计算效率。因此，对于我们而言，FFT 算法能够有效降低 DFT 计算

的复杂度，对工程应用具有十分重要的作用，需要我们将其认真掌握。