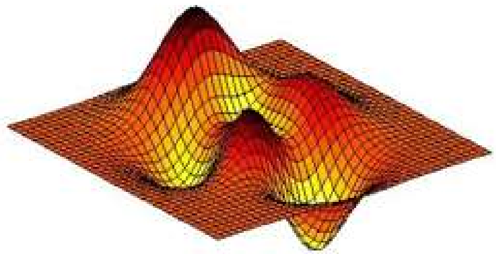
数字信号处理实验

授课老师：何　美霖 (Meilin He)
单　　位：通信工程学院
邮　　箱：meilinhe@hdu.edu.cn

第6讲 快速傅里叶变换（FFT）

- DIT-FFT
- IDFT快速算法



离散傅里叶变换

- DFT：

$$X[k] = \mathrm{DFT}[x(n)]$$

$$= \sum_{n=0}^{N-1} x(n)\,\mathrm{e}^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{N-1} x(n)\,W_N^{nk}$$

- IDFT：

$$x(n) = \mathrm{IDFT}[X(k)]$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} X(k)\mathrm{e}^{j\frac{2\pi}{N}nk} = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-nk}$$

DIT-FFT算法的基本原理

- 设序列x(n)的长度为N，且满足$N=2^M$，M是自然数。将序列x(n)按n的奇偶数分为x1(n), x2(n)两组长度为N/2的子序列。

$$\begin{cases} x_1(r) = x(2r) \\ x_2(r) = x(2r+1) \end{cases}, \quad r = 0,1,\cdots,N/2-1$$

DIT-FFT算法的基本原理

- **将N点DFT分解为两个长度为N/2的DFT。**

$$X(k)=DFT[x(n)]=\sum_{n=0}^{N-1}x(n)W_N^{kn}$$

$$=\underbrace{\sum_{r=0}^{N/2-1}x(2r)W_N^{2rk}}_{n为偶}+\underbrace{\sum_{r=0}^{N/2-1}x(2r+1)W_N^{(2r+1)k}}_{n为奇}$$

$$=\underbrace{\sum_{r=0}^{N/2-1}x_1(r)W_{N/2}^{rk}}_{X_1(k)}+W_N^k\underbrace{\sum_{r=0}^{N/2-1}x_2(r)W_{N/2}^{rk}}_{X_2(k)}$$

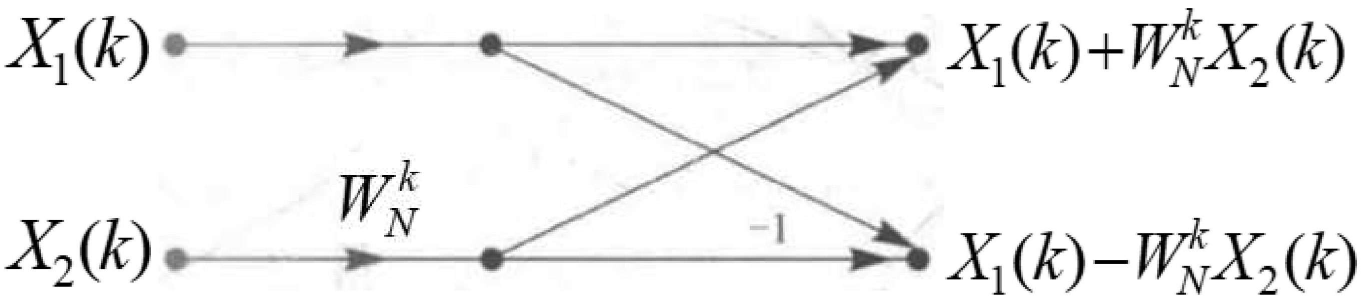
（这一步利用：$W_N^{2rk)}=W_{N/2}^{rk}$ $\quad r,k=0,1,\ldots N/2-1$

DIT-FFT算法的基本原理

- **将N点DFT分解为两个长度为N/2的DFT。**

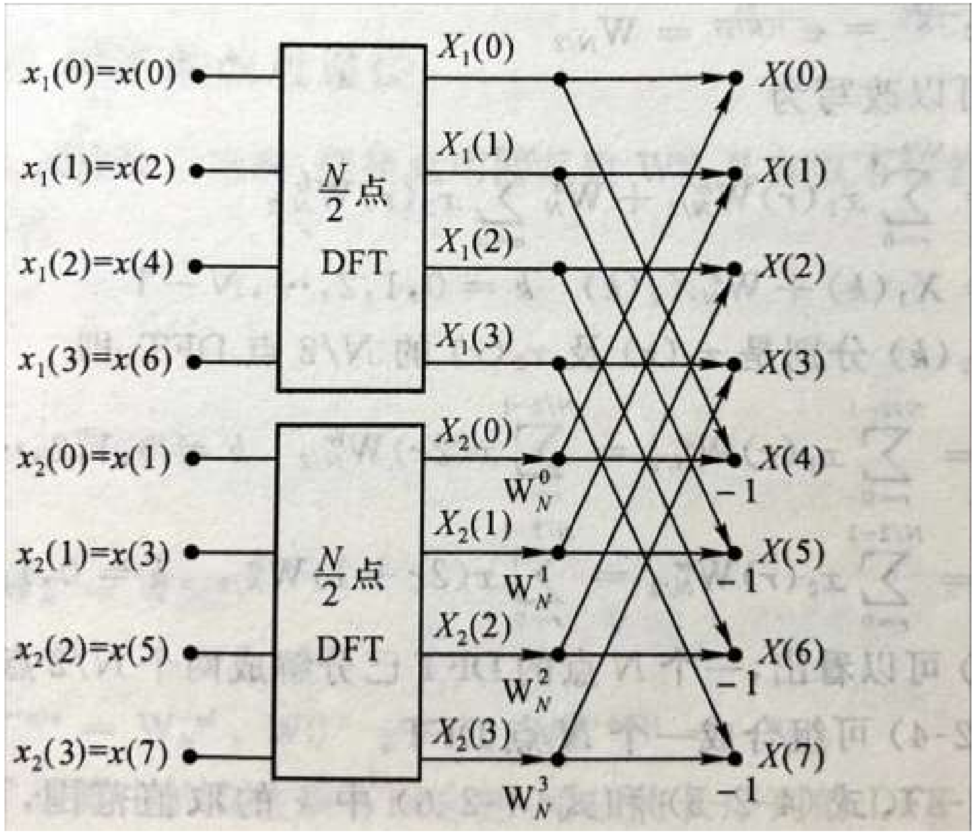
$$\therefore \begin{cases} X(k) = X_1(k) + W_N^k X_2(k) \\ X(k+\dfrac{N}{2}) = X_1(k) - W_N^k X_2(k) \end{cases} \quad k = 0,1,\ldots,N/2-1$$

将上式运算用一个专用“蝶形”信号流图表示：



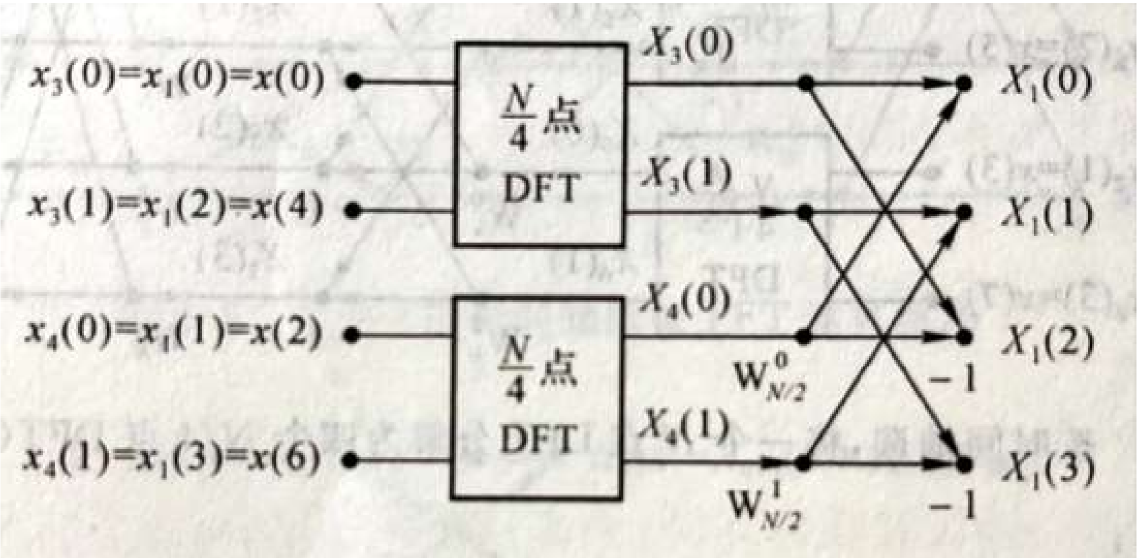
DIT-FFT算法的基本原理

- **将N点DFT分解为两个长度为N/2的DFT。**



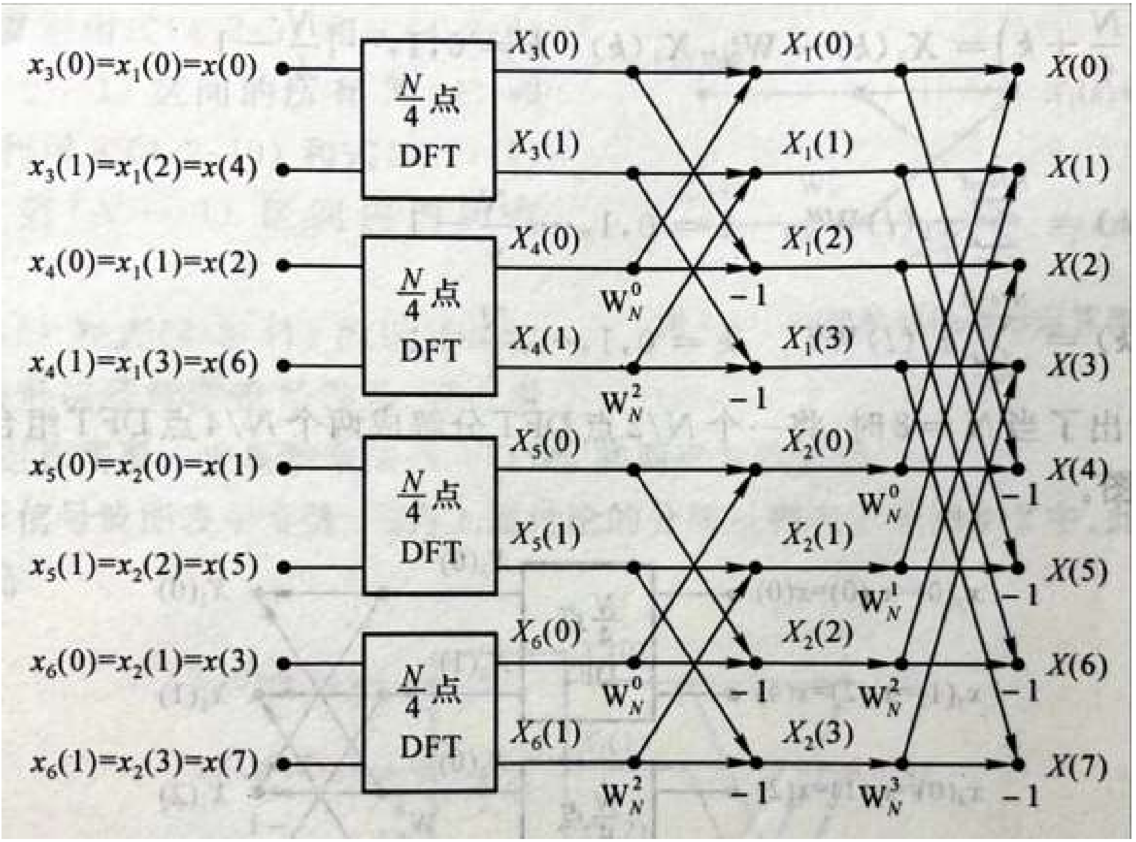
DIT-FFT算法的基本原理

- **将N/2点DFT分解为两个长度为N/4的DFT。**



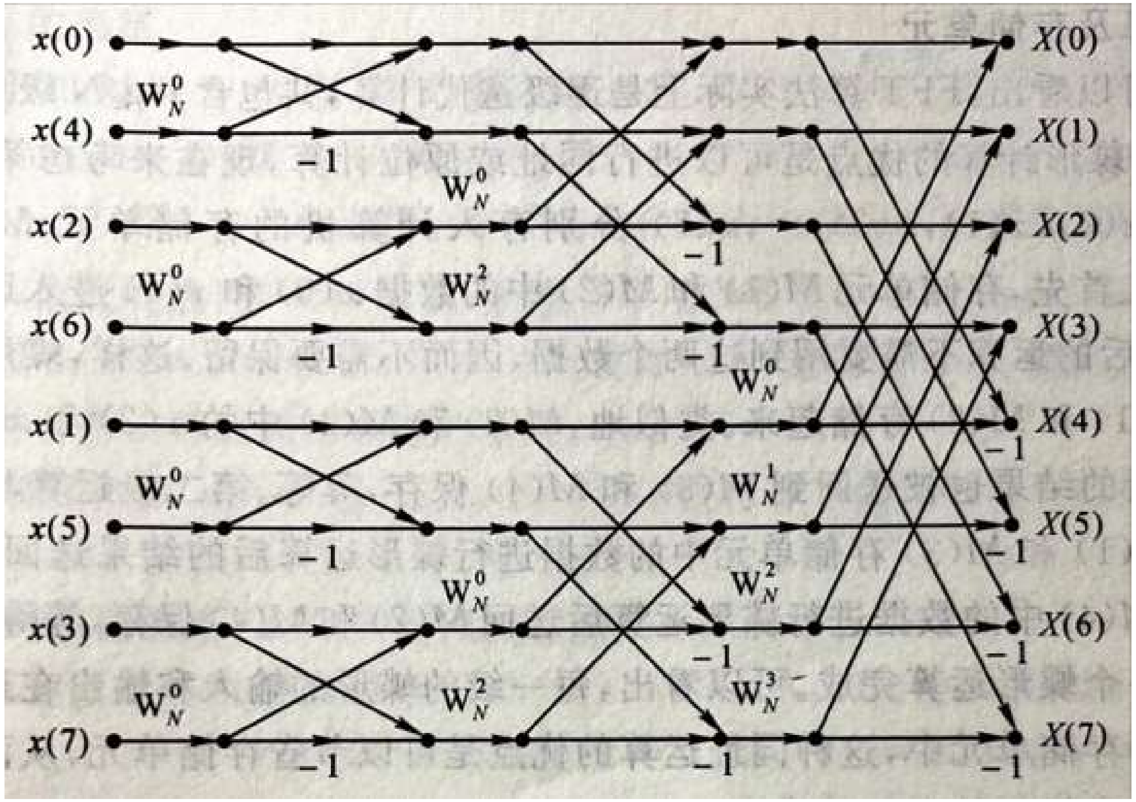
DIT-FFT算法的基本原理

- **将N点DFT分解为4个长度为N/4的DFT。**



DIT-FFT算法的基本原理

- **N = 8的DIT-FFT运算流图**



DIT-FFT算法的特点

- 蝶形结构，运算量小。
- 原位运算
- 输入或输出倒位序

倒序算法

- 雷德（Rader）算法-实现倒位序
- 自然序排列的二进制数，其下面一个数总比上面的数大1。
- 而倒序二进制数的下面一个数是上面一个数在最高位加1，并由高位向低位进位而得到的。

N=8			
顺序排列	二进制码	倒码(从0开始)	倒码对应的数
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

DIT-FFT算法的实现

- 例1：已知序列x(n) = n, 1 ≤ n ≤ 8，编程实现DIT-FFT，计算X(k)。

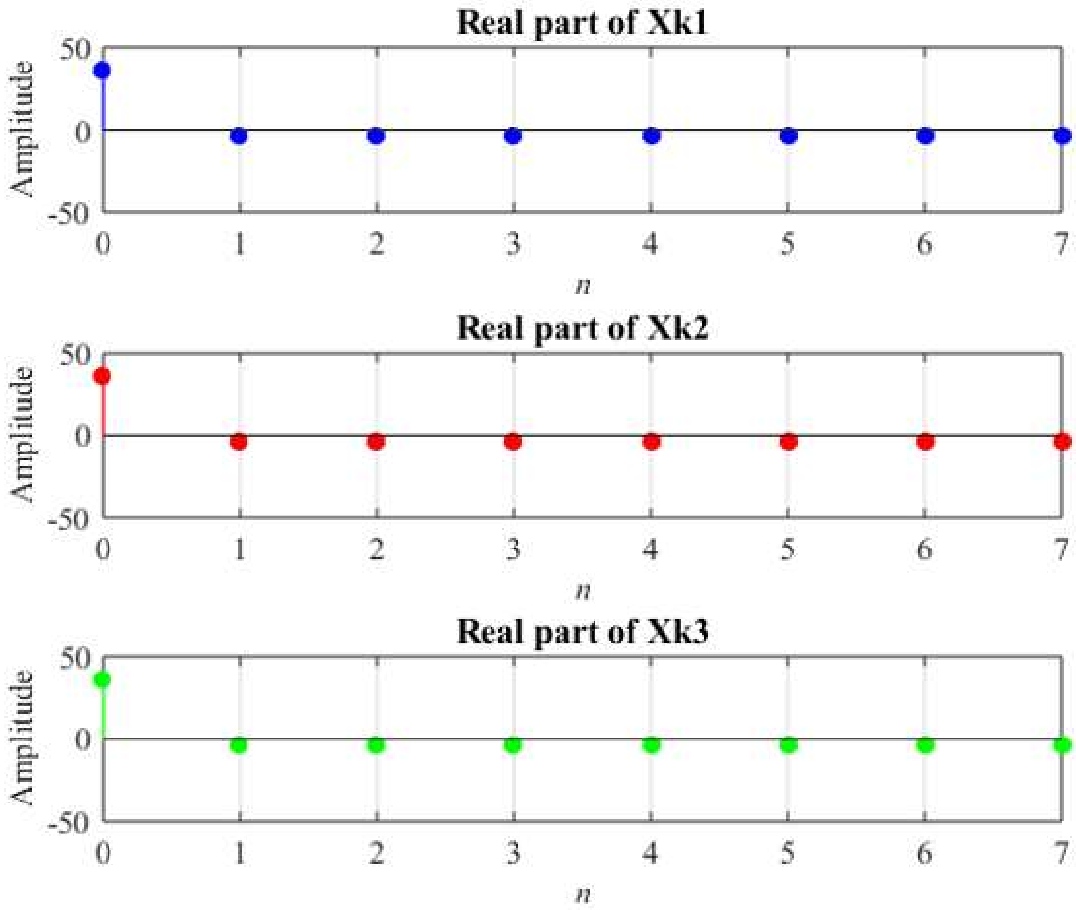
```
clc; clear; close all;
xn=1:8;
N = length(xn);
Xk1 = fft(xn);
Xk2 = ditfft(xn);
Xk3 = diffft(xn);

figure(1);
subplot(3,1,1); stem(0:N-1,real(Xk1),'fill','b','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Real part of Xk1');
subplot(3,1,2); stem(0:N-1,real(Xk2),'fill','r','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Real part of Xk2');
subplot(3,1,3); stem(0:N-1,real(Xk3),'fill','g','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Real part of Xk3');
```

```
function [Xk]=diffft(xn)
N=length(xn);
A=xn;
v=floor(log2(N));
WN=exp(-j*2*pi/N);
for m=1:v
  for k=0:2^(v-m+1):N-1
    for K=0:2^(v-m)-1
      p=k+K;
      q=p+2^(v-m);
      %基于DIT_FFT的修改;
      r=2^(m-1)*mod(p,2^(v-m+1));
      B(p+1)=A(p+1)+A(q+1);
      B(q+1)=(A(p+1)-
A(q+1))*WN^r;
    end
  end
  A=B;
  disp(A);
end
NI=N/2;
for I=1:N-1
  if I
    t=A(I+1);
    A(I+1)=A(NI+1);
    A(NI+1)=t;
  end
  T=N/2;
  while NI>=T
    NI=NI-T;
    T=T/2;
  end
  NI=NI+T;
end
Xk=A;
disp('X[k]:')
disp(Xk);
end
```

DIT-FFT算法的实现

- 例1：



IDFT快速算法原理

$$X[k]=\mathrm{DFT}[x(n)]=\sum_{n=0}^{N-1}x(n)\,e^{-j\frac{2\pi}{N}nk}=\sum_{n=0}^{N-1}x(n)\,W_N^{nk}$$

$$x(n)=\mathrm{IDFT}[X(k)]=\frac{1}{N}\sum_{k=0}^{N-1}X(k)e^{j\frac{2\pi}{N}nk}=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N^{-nk}$$

- 首先，对IDFT式取共轭，可得：

$$x^*(n)=\frac{1}{N}\sum_{k=0}^{N-1}X^*(k)W_N^{nk}$$

整理可得：

$$x(n)=\frac{1}{N}\left[\sum_{k=0}^{N-1}X^*(k)W_N^{nk}\right]^*=\frac{1}{N}\left\{\mathrm{DFT}[X^*(k)]\right\}^*$$

过程：先将X(k)取共轭，就可以直接利用FFT子程序，再将运算结果取共轭，并乘以1/N，即得到x(n)。

IDFT快速算法实现

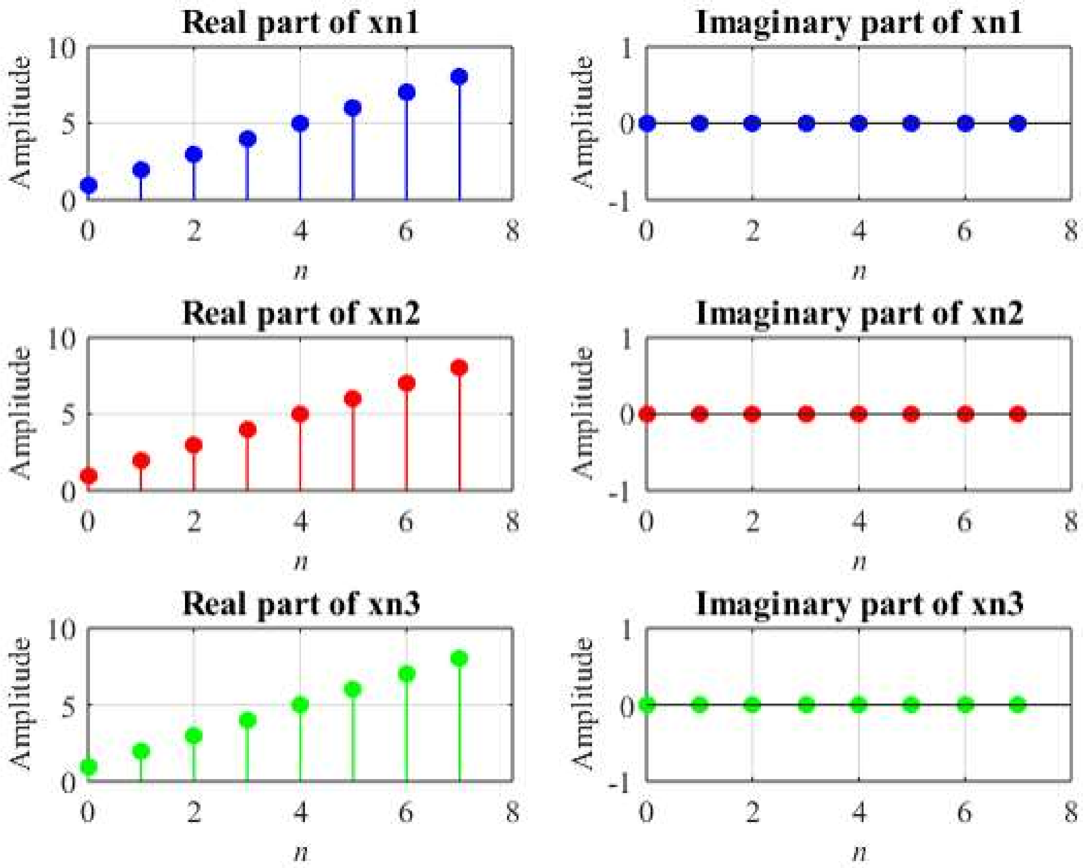
- 例2：已知X(k)={36.0000, -4.0000+9.6569i, -4.0000+4.0000i, -4.0000+1.6569i, -4.0000, -4.0000-1.6569i, -4.0000-4.0000i, -4.0000-9.6569i}，试求x(n)。

```
clc; clear; close all;
Xk=[36.0000, -4.0000+9.6569i, -4.0000+4.0000i, -4.0000+1.6569i, -4.0000, -4.0000-1.6569i, -4.0000-4.0000i, -4.0000-9.6569i];
N = length(Xk);
%% 直接调用ifft函数计算Xk的离散傅里叶反变换
xn1 = ifft(Xk);
%% 利用fft函数和ditfft函数计算Xk的离散傅里叶反变换
Xk_conj = conj(Xk);
xn2 = conj(fft(Xk_conj))/N;
xn3 = conj(ditfft(Xk_conj))/N;
```

```
figure(1);
subplot(3,2,1); stem(0:N-1,real(xn1),'fill','b','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Real part of xn1');
subplot(3,2,2); stem(0:N-1,imag(xn1),'fill','b','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Imaginary part of xn1');
subplot(3,2,3); stem(0:N-1,real(xn2),'fill','r','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Real part of xn2');
subplot(3,2,4); stem(0:N-1,imag(xn2),'fill','r','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Imaginary part of xn2');
subplot(3,2,5); stem(0:N-1,real(xn3),'fill','g','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Real part of xn3');
subplot(3,2,6); stem(0:N-1,imag(xn3),'fill','g','linewidth',1.0);
xlabel('\itn'); ylabel('Amplitude'); title('Imaginary part of xn3');
```

IDFT快速算法实现

- 例2：



总结

- DIT-FFT
  - ditfft()
- IDFT快速算法
  - conj()
  - fft() or ditfft()

操作验收习题

6.1 已知序列x(n)={2,1,3,9,0,5,7,8}, 模仿例题编程实现DIT-FFT功能计算X(k), 并与直接调用函数fft()命令对比计算结果是否正确。

6.2 已知序列x(n)的DFT为X(k)={36.0000,-4.0000+9.6569i, -4.0000+4.0000i,-4.0000+1.6569i,-4.0000,-4.0000-0.6569i,-4.0000-4.0000i,-4.0000-9.6569i }, 模仿例题编程实现IFFT功能计算x(n), 并与直接调用函数 ifft()命令对比计算结果是否正确。

实验报告作业题和思考题

◆ 实验报告作业题:

6.1 已知序列x(n)={2,1,3,9,0,5,7,8}, 模仿例题编程实现DIT-FFT功能计算X(k), 并与直接调用函数fft()命令对比计算结果是否正确。

6.2 已知序列x(n)的DFT为X(k)={36.0000,-4.0000+9.6569i, -4.0000+4.0000i,-4.0000+1.6569i,-4.0000,-4.0000-0.6569i,-4.0000-4.0000i,-4.0000-9.6569i }, 模仿例题编程实现IFFT功能计算x(n), 并与直接调用函数 ifft()命令对比计算结果是否正确。

◆ **思考题：DIF-FFT**

感谢聆听!