实现并优化快速傅里叶变换，并基于快速傅里叶变换实现两个离散信号的卷积

摘要：利用 Cooley-Tukey 算法，可以较好地实现对长度为 2^N 的离散序列的快速 FFT 计算。但当序列长度不是 2^N 时，需要在序列的末尾补 0 至长度满足要求，但此时采样点改变，需通过插值法寻找原来的采样点。本文将对仅可计算长度为 2^N 的 FFT 算法进行优化，使其能够尽可能精确、快速地计算任意长度序列的 FFT，并应用优化后的程序，计算两个离散序列的卷积。

关键词：FFT、插值法、卷积

一、理论基础与问题背景

1.1 离散序列的 DFT 与 FFT 算法

一个周期为 N 的离散序列$x[n]$的傅里叶级数计算方法为：

$$a_k = X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk}, \qquad k = 0,1,2,\dots N-1 \tag{1-1}$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} a_k W_N^{-nk}, \qquad n = 0,1,2,\dots N-1 \tag{1-2}$$

其中$W_N = e^{-j\frac{2\pi}{N}}$。易得$a_{k+N} = X[k+N] = \sum_{n=0}^{N-1} x[n]W_N^{n(k+N)} = \sum_{n=0}^{N-1} x[n]W_N^{nk} = a_k = X[k]$。而一个离散非周期信号的离散傅里叶变换和反变换为：

$$X\left(e^{j\omega}\right) = \sum_{n=-\infty}^{+\infty} x[n]e^{-j\omega n} \tag{1-3}$$

$$x[n] = \frac{1}{2\pi}\int_{2\pi} X\left(e^{j\omega}\right)e^{j\omega n}d\omega \tag{1-4}$$

可以看到，一个离散周期信号的傅里叶级数，是其一个周期内的非周期离散信号的傅里叶变换结果$X\left(e^{j\omega}\right)$在$[0,2\pi)$之间的均匀抽样，即：

$$a_k = X[k] = X\left(e^{j\omega}\right)\Big|\omega = \frac{2\pi}{N}k \tag{1-5}$$

若直接计算上述过程，其算法的复杂度为$O(N^2)$。

Cooley-Tukey 是按照奇偶抽选的 FFT（DIT-FFT）算法。当$N = 2^L$，L为正整数时，可将序列$x(n)$按照n的奇偶性分为以下两组：

$$\left.\begin{matrix} x[2r] = x_1[r] \\ x[2r+1] = x_2[r] \end{matrix}\right\}, \qquad r = 0,1,\dots,\frac{N}{2}-1$$

则其 N 点 DFT 为：

$$X[k]=\underbrace{\sum_{n=0}^{N-1}x[n]W_N^{nk}}_{n\text{为偶数}}+\underbrace{\sum_{n=0}^{N-1}x[n]W_N^{nk}}_{n\text{为奇数}}$$

$$=\sum_{r=0}^{\frac{N}{2}-1}x_1[r](W_N^2)^{rk}+W_N^k\sum_{r=0}^{\frac{N}{2}-1}x_2[r](W_N^2)^{rk}$$

利用$W_N^2=W_{N/2}$，上式可表示为：

$$X[k]=\sum_{r=0}^{\frac{N}{2}-1}x_1[r]W_{N/2}^{rk}+W_N^k\sum_{r=0}^{\frac{N}{2}-1}x_2[r]W_{N/2}^{rk}=X_1[k]+W_N^kX_2[k] \qquad (1-6)$$

式中$X_1(k)$和$X_2(k)$分别是$x_1(r)$和$x_2(r)$的$N/2$点 DFT。

$$X_1[k]=\sum_{r=0}^{\frac{N}{2}-1}x_1[r]W_{N/2}^{rk}=\sum_{r=0}^{\frac{N}{2}-1}x[2r]W_{N/2}^{rk} \qquad (1-7)$$

$$X_2[k]=\sum_{r=0}^{\frac{N}{2}-1}x_2[r]W_{N/2}^{rk}=\sum_{r=0}^{\frac{N}{2}-1}x[2r+1]W_{N/2}^{rk} \qquad (1-8)$$

注意到$X[k]$有N个点，而$X_1[k]$和$X_2[k]$各只有$N/2$个点。利用$W_{N/2}^{rk}=W_{N/2}^{r\left(k+\frac{N}{2}\right)}$，这样可以得到：

$$X_1[\frac{N}{2}+k]=\sum_{r=0}^{\frac{N}{2}-1}x_1[r]W_{N/2}^{r\left(k+\frac{N}{2}\right)}=\sum_{r=0}^{\frac{N}{2}-1}x_1[r]W_{N/2}^{rk}=X_1[k] \qquad (1-9)$$

同理可得：

$$X_2[\frac{N}{2}+k]=X_2[k] \qquad (1-10)$$

1-9、1-10 式说明了$\frac{N}{2}\le k\le N-1$所对应的$X_1[k]$、$X_2[k]$，分别等于$0\le k\le N/2-1$所对应的值。

又由于$W_N^{\frac{N}{2}+k}=-W_N^k$，这样可以将$X[k]$表达为前后两部分：

前半部分$X[k](k=0,1,\dots\frac{N}{2}-1)$可表示为：

$$X[k]=X_1[k]+W_N^kX_2[k] \qquad (1-11)$$

后半部分$X[k](k=\frac{N}{2},\dots,N-1)$可表示为：

$$X[k]=X_1[k]-W_N^kX_2[k] \qquad (1-12)$$

这样，只要求出$0\sim(N/2-1)$ 区间的所有$X_1[k]$和$X_2[k]$值，即可求出$0\sim(N-1)$ 区间内的所有$X[k]$ 值，大大节省了运算。可以算得这种算法的计算复杂度为$O(Nlog(N))$。

上述算法需要满足$N=2^L$，但是实际应用中，序列的长度大多数无法满足这个条件，因此需要对上述过程进行修改和完善。

1.2 离散序列的卷积计算

设序列$x[n]$的长度为N_1，$h[n]$的长度为N_2，则二者的卷积为：

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \qquad (1-13)$$

且$y[n]$的长度为$N_1 + N_2 - 1$。若直接按上式计算卷积，其复杂度为$O(N^2)$，当序列长度较大时费时过长。

二、原理分析与算法设计

对于一个长度不为2^L的序列$x[n]$（设其长度为N），考虑在其末尾补 0，获得一个长度$N' = 2^L$的新序列$x'[n]$：

$$x'[n] = \begin{cases} x[n], & 0 \le n \le N-1 \\ 0, & N < n \le N'-1 \end{cases}, \qquad n = 0,1,2,\dots,N'-1 \qquad (2-1)$$

对$x'[n]$进行 DFT：

$$X'[k] = \sum_{n=0}^{N'-1} x'[n]W_{N'}^{nk}, \qquad k = 0,1,2,\dots N'-1$$

由于$x'[n]$的末尾项均为 0，故上式可表示为：

$$X'[k] = \sum_{n=0}^{N'-1} x'[n]W_{N'}^{nk} = \sum_{n=0}^{N-1} x[n]W_{N'}^{nk}, \qquad k = 0,1,2,\dots N'-1 \qquad (2-2)$$

式 2-2 表明，增补的 0 值项不影响有效数据的个数，但会使时域点数增加，因而增加频域的抽样点个数。此时频谱$X(e^{j\omega})$不变，而$X[k]$是$X(e^{j\omega})$在区间$[0,2\pi)$内的均匀抽样，故抽样点位置改变。

如果想要确定原抽样点的值，一种方法是寻找补 0 后最靠近原抽样点的两个抽样点，用直线近似的方法估计原抽样点的值。在编写计算机程序时，考虑到 MATLAB 序列的下标是从 1 开始而不是从 0 开始的，因此需要做出细微调整。具体方法如下：

设原序列长度为N_1，原抽样点的下标为x，其对应的抽样值为y；补 0 后新的序列长度为N_2，原抽样点对应新序列的抽样点下标为x'，其对应的抽样值为仍为y。且不失一般性的，x'不为整数，而是夹在x_1与x_2两个相邻整数之间的值，其对应的抽样值分别为y_1、y_2，如图 1、图 2 所示。

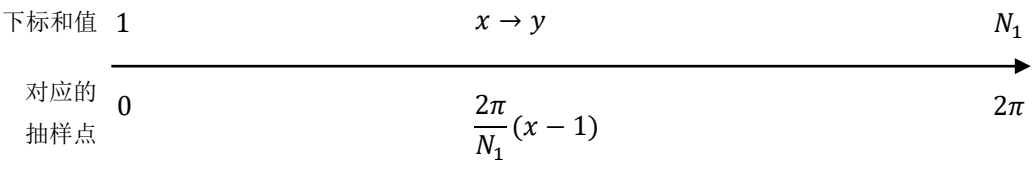

图 1 补 0 前下标和抽样点的对应关系（每一个抽样点对应的抽样值未知）

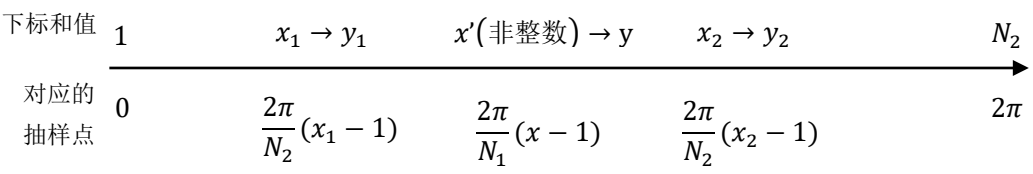

图 2 补 0 后下标和抽样点的对应关系（每一个抽样点对应的抽样值已知）

各变量之间的关系为：

$$\frac{x'-1}{N_2}=\frac{x-1}{N_1} \tag{2-3}$$

$$x_1=[x'] \tag{2-4}$$

$$x_2=x_1+1 \tag{2-5}$$

其中$[x']$表示不超过x'的最大整数。

由此我们可以得到原抽样点对应的抽样值y：

$$y-y_1=\frac{y_2-y_1}{x_2-x_1}(x'-x_1) \tag{2-6}$$

为了使得一阶线性近似尽可能准确，可以增加补 0 的个数。如果原序列的长度N_1较小，可以补 0 至第 3~6 个比N_1 大的 2 次幂数，但如果N_1本身较大，补 0 至相邻的 2 次幂数即可。

具体程序如下：

```
%fftNew.m
function Y = fftNew(X)
    if length(X) == 2                %仅有两位的序列最简单，也是递归的基础
        Y = zeros(1,2);
        Y(1) = X(1)+X(2);
        Y(2) = X(1)-X(2);
    else
        not2N = 0;
        Xlen = length(X);
        if bitand(length(X),(length(X)-1)) ~= 0    %判断序列的长度是不是 2^n
            not2N = 1;                             %如果不是，就补 0 至 2^n
            if length(X)<= 2^15              %补 0 的个数随序列长度增加而减少
                N = 2^floor(log2(length(X))+4);
            elseif length(X)<= 2^25
                N = 2^floor(log2(length(X))+2);
            else
                N = 2^floor(log2(length(X))+1);
            end
            X = [X,zeros(1,N-length(X))];
        else
            N = Xlen;
        end
        X1 = X(1:2:N);
        X2 = X(2:2:N);                 %按照奇数列和偶数列对原序列进行拆分
        Y1 = fftNew(X1);               %运用递归算法处理下一级的变换
        Y2 = fftNew(X2);
        Y = zeros(1,N);
        for k = 2:N/2                  %由式 1-6、1-11、1-12 计算 Y
            Y2(k) = Y2(k)*exp(-1j*2*pi*(k-1)/N);
        end
        for k = 1:N/2
            Y(k) = Y1(k) +Y2(k);
```

```
            Y(k+N/2) = Y1(k) - Y2(k);
        end
        if not2N ==1            %如果原序列的长度不是 2 的 n 次幂，
            Ytemp = Y;          %需要利用插值法估计原采样点的值
            Y1 = Y(1);
            Y = zeros(1,Xlen);
            Y(1) = Y1;
            for i = 2:1:Xlen
                x3 = length(Ytemp)*(i-1)/Xlen+1;
                x1 = floor(length(Ytemp)*(i-1)/Xlen+1);
                x2 = x1 + 1;
                Y(i) = Ytemp(x1) +(Ytemp(x1)-Ytemp(x2))/(x1-x2)*(x3-x1);
            end
        end
    end
end
```

由式 1-2 可知，FFT 反变换的程序和正变换基本相同，但要把所有W_N^{nk}变成W_N^{-nk}，同时除以序列的长度。设补 0 后序列的长度为$N'=2^L$，采用递归算法，共需递归L次，故每一次递归需要将所得的幅值除以 2。同时，由于在频域序列的末尾补 0，最后需要将幅值乘以一个放大倍数来恢复原来的幅值。其原因分析及计算过程如下：

设补 0 前序列$x[n]$，$X[n]$的长度为N，将式 1-2 重写如下：

$$x[n]=\frac{1}{N}\sum_{k=0}^{N-1}X[k]W_N^{-nk},\qquad n=0,1,2,\dots N-1 \tag{2-7}$$

补 0 后，正变换的结果$X'[n]$的长度变为$N'=2^L$，故反变换得到的$x[n]$结果为：

$$x[n]=\frac{1}{N'}\sum_{k=0}^{N'-1}X'[k]W_{N'}^{-nk},\qquad n=0,1,2,\dots N'-1 \tag{2-8}$$

由于

$$X'[n]=\begin{cases}X[n], & 0\le n\le N-1\\ 0, & N\le n<N'\end{cases} \tag{2-9}$$

故补 0 后$X'[n]$的有效数据不变，$x[n]$的长度仍为N：

$$\begin{aligned}x[n]&=\frac{1}{N'}\sum_{k=0}^{N-1}X'[k]W_{N'}^{-nk}\\ &=\frac{1}{N'}\sum_{k=0}^{N-1}X[k]W_N'^{-nk},\qquad n=0,1,\dots,N-1\end{aligned} \tag{2-10}$$

比较式 2-10 与式 2-7，发现在幅值上相差的比值，即放大倍数为：

$$Amp=\frac{N'}{N} \tag{2-11}$$

具体程序如下：

```
%ifftNew.m
function Y = ifftNew(X)
```

```
not2N = 0;
Xlen = length(X);
if length(X) == 2
    Y = zeros(1,2);
    Y(1) = X(1)+X(2);
    Y(2) = X(1)-X(2);
else
     if bitand(length(X),(length(X)-1)) ~= 0
        not2N = 1;
        if length(X)<= 2^15
            N = 2^floor(log2(length(X))+4);
        elseif length(X)<= 2^25
            N = 2^floor(log2(length(X))+2);
        else
            N = 2^floor(log2(length(X))+1);
        end
        X = [X,zeros(1,N-length(X))];
    else
        N = Xlen;
    end
    X1 = X(1:2:N);
    X2 = X(2:2:N);
    Y1 = ifftNew(X1);
    Y2 = ifftNew(X2);

    Y = zeros(1,N);
    for k = 2:N/2
        Y2(k) = Y2(k)*exp(1j*2*pi*(k-1)/N);    %与正变换程序相比，
    end                                        %这里需要将-j 变为就+j
    for k = 1:N/2
        Y(k) = Y1(k) +Y2(k);
        Y(k+N/2) = Y1(k) - Y2(k);
    end
end
Y= Y/2;                 %序列长度为 2^N，共递归 N 次，故每一次递归幅值需要/2
if not2N == 1
    Amp =  N/Xlen;        %乘上一个放大倍数补足损失
    Ytemp = Y;
    Y1 = Y(1);
    Y = zeros(1,Xlen);
    Y(1) = Y1*Amp;
    for i = 2:1:Xlen
        x3 = length(Ytemp)*(i-1)/Xlen+1;
        x1 = floor(length(Ytemp)*(i-1)/Xlen+1);
```

```
            x2 = x1 + 1;
            Y(i) = (Ytemp(x1) +(Ytemp(x1)-Ytemp(x2))/(x1-x2)*(x3-x1))*Amp;
        end
    end
end
```

由上述过程，我们可以基本实现任意长度序列的 FFT 与 IFFT。

在此基础上，我们可以用 DFT 来计算两个时域信号的卷积。首先证明圆卷积定理：设有限长序列$x_1[n]$的长度为N_1，有限长序列$x_2[n]$的长度为N_2，取$L \geq \max[N_1, N_2]$。将$x_1[n]$和$[n]$都补 0 至L点长序列，它们的L点 DFT 分别为$X_1[k]$和$X_2[k]$，且用$x[n-m]_L$表示L长度序列的循环位移。若

$$y[n] = x_1[n] \circledast x_2[n] \tag{2-12}$$

则

$$Y[k] = DFT(y[n]) = X_1[k]X_2[k] \tag{2-18}$$

证明：

$$Y[k] = \sum_{n=0}^{L-1}\left(\sum_{m=0}^{L-1} x_1[m]x_2[n-m]_L\right)W_L^{kn}$$

$$= \sum_{m=0}^{L-1} x_1[m]W_L^{km}\sum_{n=0}^{L-1} x_2[n-m]_L W_L^{k(n-m)}$$

$$= X_1[k]\left(\sum_{n=0}^{m-1} x_2[n-m]_L W_L^{k(n-m)} + \sum_{n=m}^{L-1} x_2[n-m]_L W_L^{k(n-m)}\right)$$

$$= X_1[k]\left(\sum_{n=0}^{m-1} x_2[n-m+L]W_L^{k(n-m+L)} + \sum_{n=0}^{L-1-m} x_2[n]W_L^{kn}\right)$$

$$= X_1[k]\left(\sum_{n=L-m}^{L-1} x_2[n]W_L^{kn} + \sum_{n=0}^{L-1-m} x_2[n]W_L^{kn}\right)$$

$$= X_1[k]X_2[k] \tag{2-19}$$

将圆卷积的两个序列的周期进行补 0 延拓，使得延拓后满足周期$L \geq N_1 + N_2 - 1$，这时各个延拓周期不会重叠，则延拓后的两个序列的圆卷积结果在主值区间内就是延拓前两个序列的线性卷积结果，即：

$$x_1[n] \circledast x_2[n] = x_1[n] * x_2[n], \quad \begin{cases} L \geq N_1 + N_2 - 1 \\ 0 \leq n \leq N_1 + N_2 - 2 \end{cases} \tag{2-20}$$

由此，可利用圆卷积和 FFT 来计算线性卷积。设输入序列为$x[n], 0 \leq n \leq N_1 - 1$，系统的单位脉冲响应为$h[n], 0 \leq n \leq N_2 - 1$，用计算圆卷积的方法求系统输出$y_l[n] = x[n] * h[n]$的过程为：

① 令$L = 2^m \geq N_1 + N_2 - 1$

② 取

$$x[n] = \begin{cases} x[n], & 0 \leq n \leq N_1 - 1 \\ 0, & N_1 \leq n < L - 1 \end{cases}$$

$$h[n]=\begin{cases}h[n], & 0\le n\le N_2-1\\ 0, & N_2\le n<L-1\end{cases}$$

③ $X[k]=DFT(x[n])$，L点；$H[k]=DFT(h[n])$，L点

④ $Y[k]=X[k]\cdot H[k]$

⑤ $y[n]=IDFT(Y[k])$，L点

⑥ $y_l[n]=y[n],0\le n\le N_1+N_2-2$

具体程序如下：

```
%DiscreteConvolutionUsingFFT.m
function y = DiscreteConvolutionUsingFFT(x,h)
    N1 = length(x);
    N2 = length(h);
    m = floor(log2(N1+N2-1))+1;
    L = 2^m;
    x = [x,zeros(1,L-N1)];
    h = [h,zeros(1,L-N2)];
    X = fftNew(x);
    H = fftNew(h);
    Y = X.*H;
    y = ifftNew(Y);
    y = y(1:N1+N2-1);
end
```

三、实验过程与数据结果

3.1 用改进的程序计算 FFT 和 IFFT，测试准确性和响应时间

先取

$$x[n]=[1,1,4,5,1,4]$$

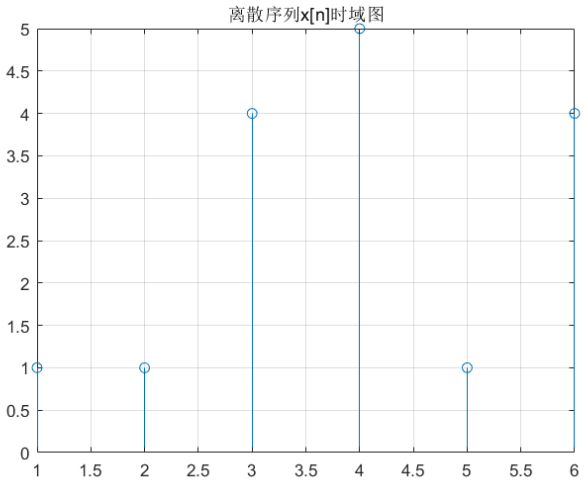

图 3

用 fftNew 计算，得到的结果如图 4 所示：

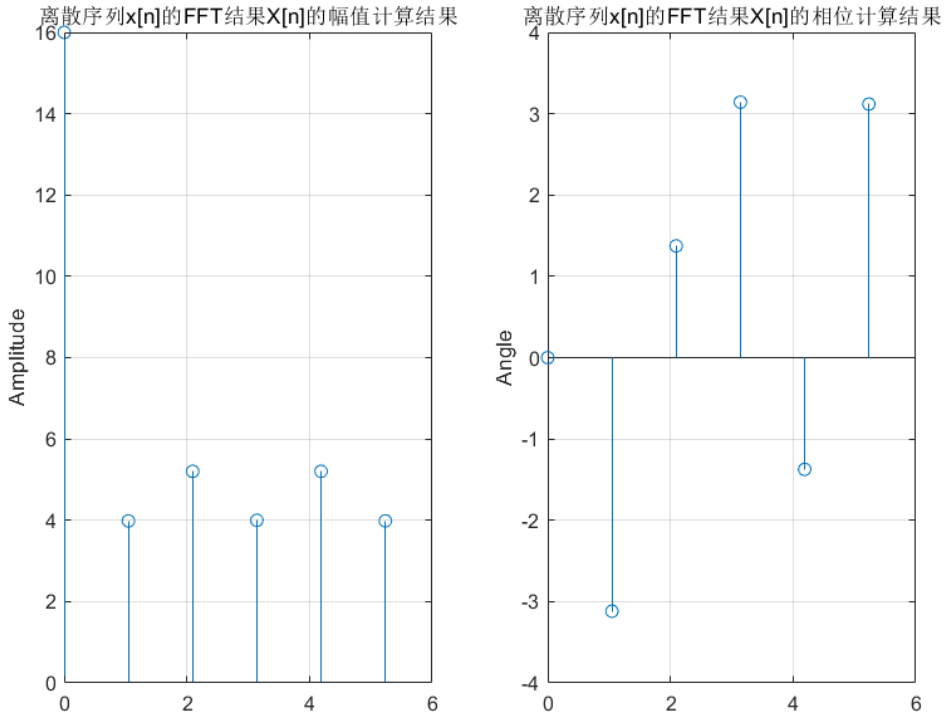

图 4 实际结果的幅度和相位

用 MATLAB 自带的 fft 验证，结果如图 5 所示：

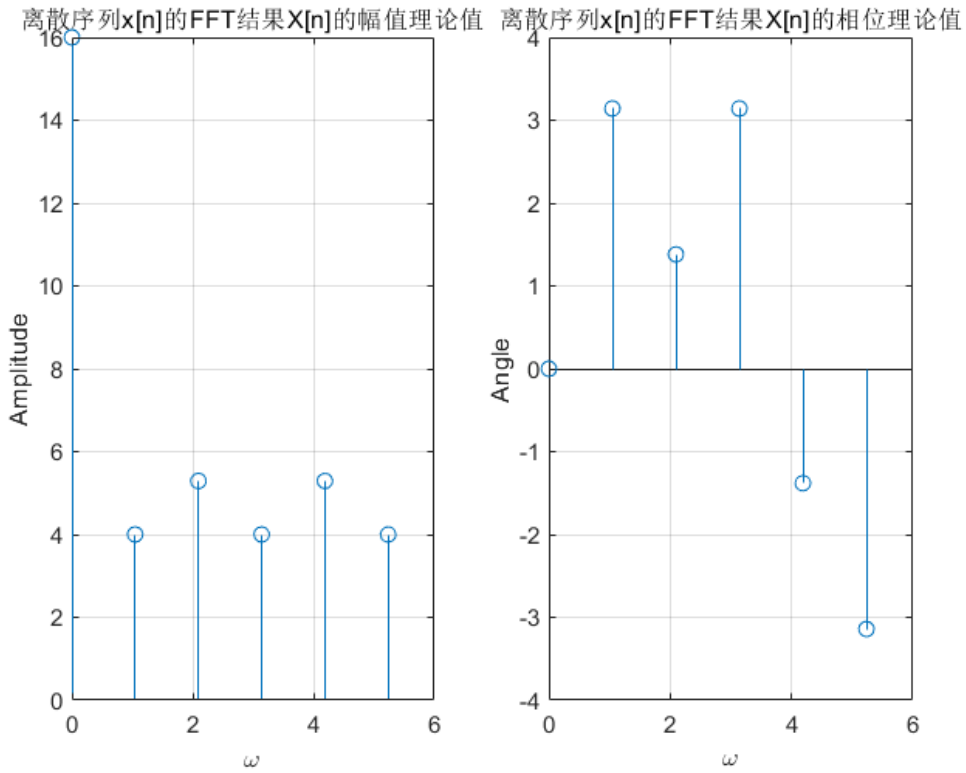

图 5 理论值的幅度和相位

总体上看，fftNew 子程序可以较为精准地恢复原抽样点频域信号的幅值和相位。（虽然图中第一个点和第五个点的相位误差很大，但稍加观察即可得知这是$-\pi$和$+\pi$之间的不同，没有本质差别）现考量幅度谱误差，如图 6 所示：

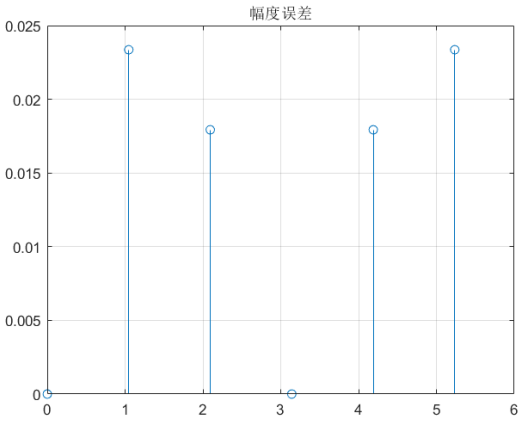

图 6 幅度误差

可见当补 0 数量足够多时，幅度谱的误差基本较小。

当$x[n] = [5,9,2,9,8,7,68,62,5,1,36,1,4,5,7,5,6]$，同样进行上述过程，结果如图 7~图 9。

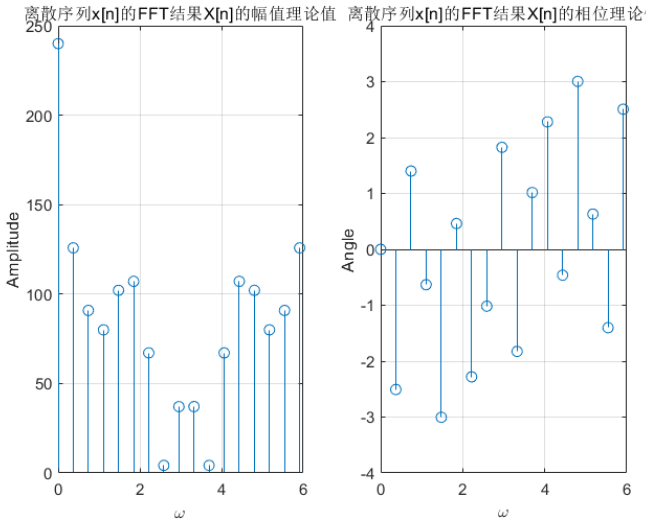

图 7 FFT 理论值幅度和相位

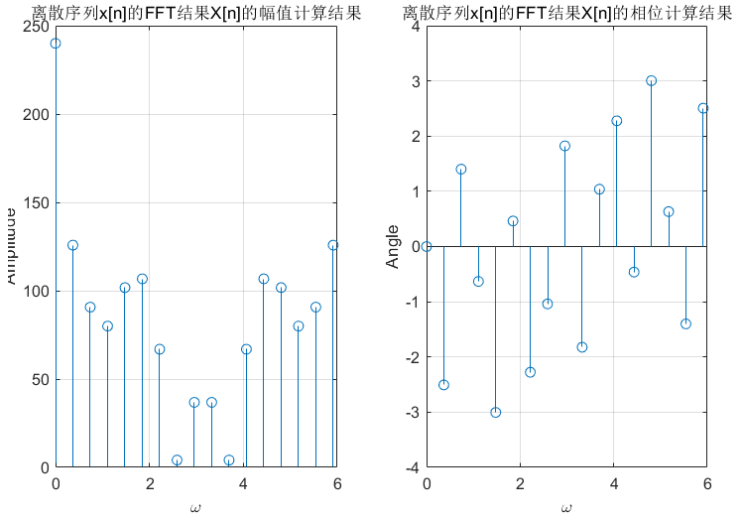

图 8 FFT 实际值幅度和相位

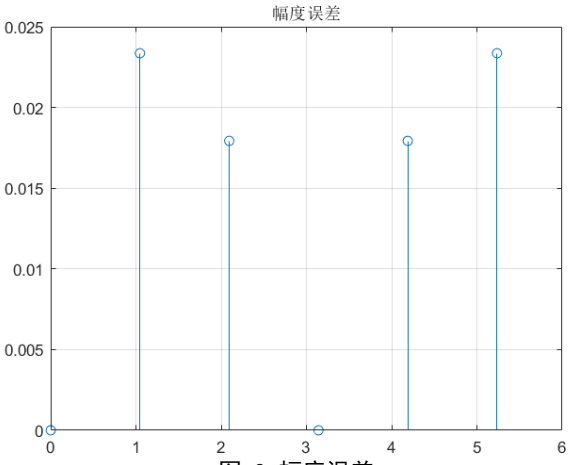

图 9 幅度误差

从新的变换结果可知，幅度误差和相位误差依旧很小并保持在稳定范围内。上述过程表明，fftNew 子程序能够基于插值法对任意长度序列的 FFT 变换。

考察 FFT 逆变换过程，取输入频域抽样序列为

$$X[n] = [2,9,5,3,7,12,14,2,6,35,1]$$

其 IFFT 过程如图 10~12 所示。

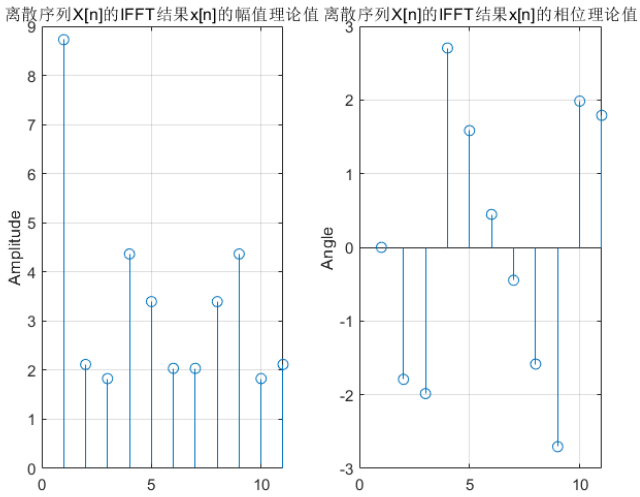

图 10 IFFT 的理论值幅度和相位

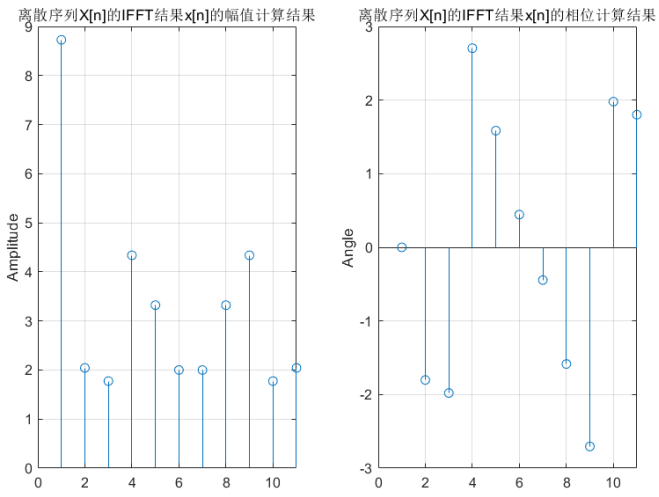

图 11 IFFT 的实际值幅度和相位

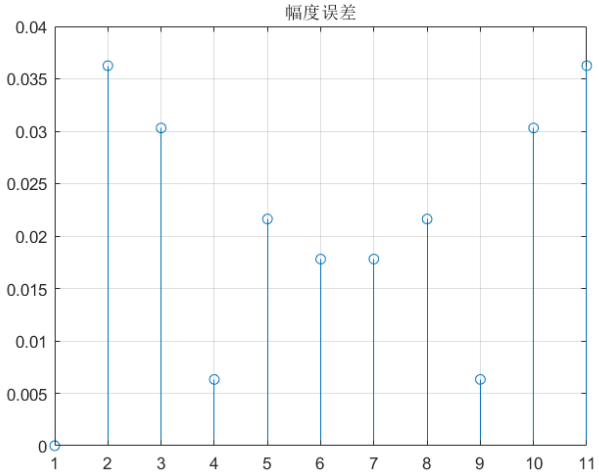

图 12 幅度误差

从图中观察到，IFFT 结果的幅度误差和相位误差均较小，故 ifftNew 子程序可以较准确地实现傅里叶逆变换

现计算自编子程序的计算速度。将输入序列$x1[n]$取成长度为2^{20}的序列，分别用标准 FFT 程序和 fftNew 程序对其进行 FFT，观察记录其计算所需时间，并绘制误差，结果如图 13 所示。

```
x1 = rand(1,2^20).*10;
X1 = fftNew(x1);                                    2.37 sec
X2 = fft(x1);
plot((abs(X1)-abs(X2))./abs(X2));
```

图 13

该程序对较长序列的响应时间为 2.37 秒，而标准 FFT 的响应时间在 0.1~0.3 秒之间。关于响应时间的问题，将在本文第五部分“讨论与展望”中具体展开。

图 14 展示了此时序列 FFT 变换结果与理论值的误差，可以看到其数量级在10^{-13}。故当输入序列的长度非常大时，将其补 0 至相邻的二次幂数也可以较好地利用插值法恢复原抽样点的值。

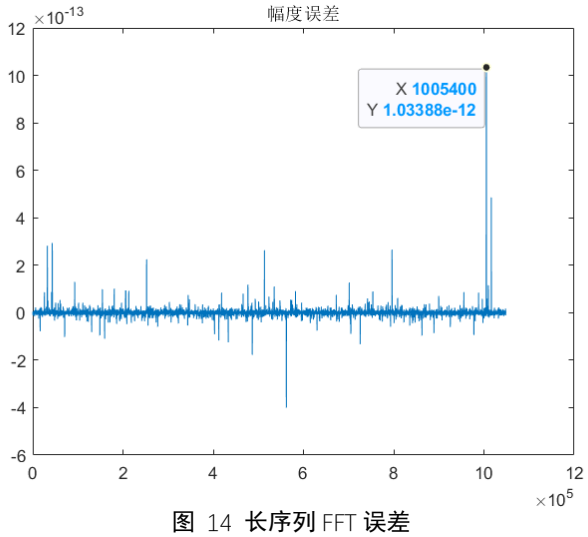

图 14 长序列 FFT 误差

由于 ifftNew 是利用W_N^{nk}和W_N^{-nk}的共轭关系，由 fftNew 程序改写而成，每一次运算仅在数值上相差一个负号或一个系数，不改变加法和乘法的数量级，故 ifftNew 的响应时间也为 2 秒左右。另由图 16 可见，其误差非常小。

```
X1 = rand(1,2^20).*10;
x1 = ifftNew(X1);                    2.79 sec
x2 = ifft(X1);
plot((abs(x1)-abs(x2))./abs(x...
```

图 15

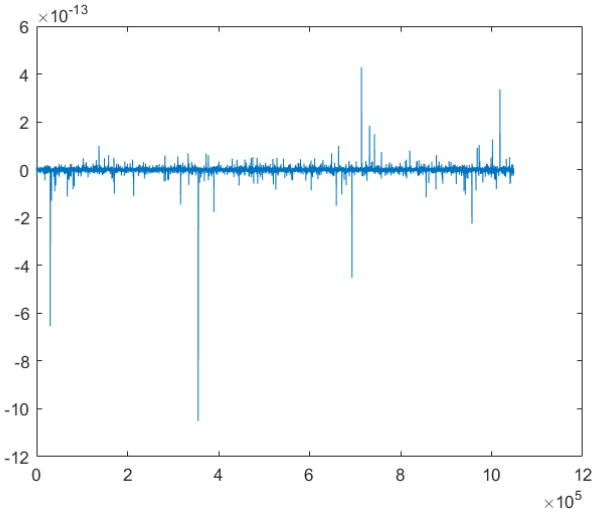

图 16 长序列 IFFT 误差

3.2 用改进的 FFT 和 IFFT 计算序列的线性卷积

取

$$x[n] = [1,1,4,5,1,4]$$
$$h[n] = [1,9,1,9,8,1,2,3,3,2,9,7]$$

利用自编的计算线性卷积的子函数，计算$y[n] = x[n] * h[n]$，同时用 MATLAB 自带的 conv 函数，计算其理论值。结果如图 17 所示

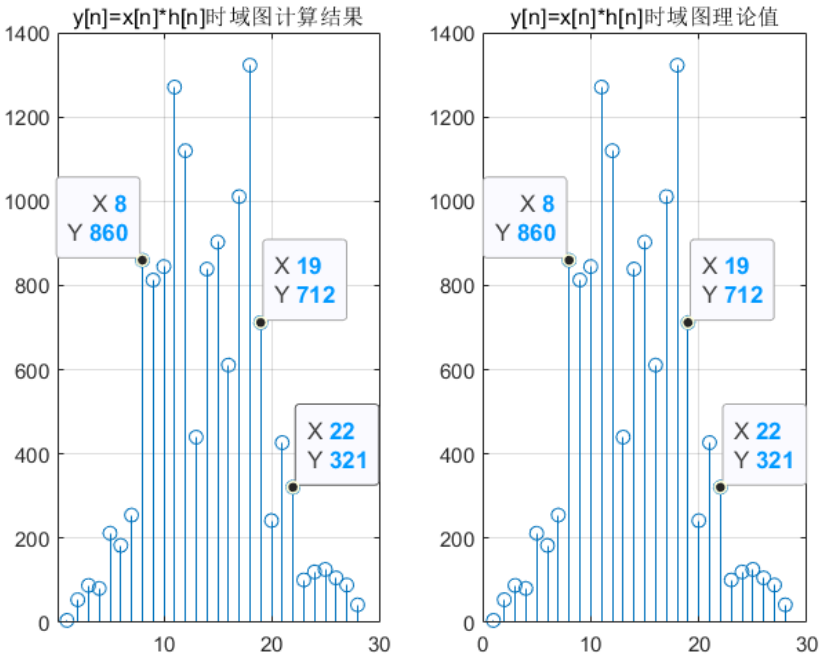

图 17 序列卷积结果

从结果看，理论值和实际值基本一致。计算其误差，结果如图 18 所示。

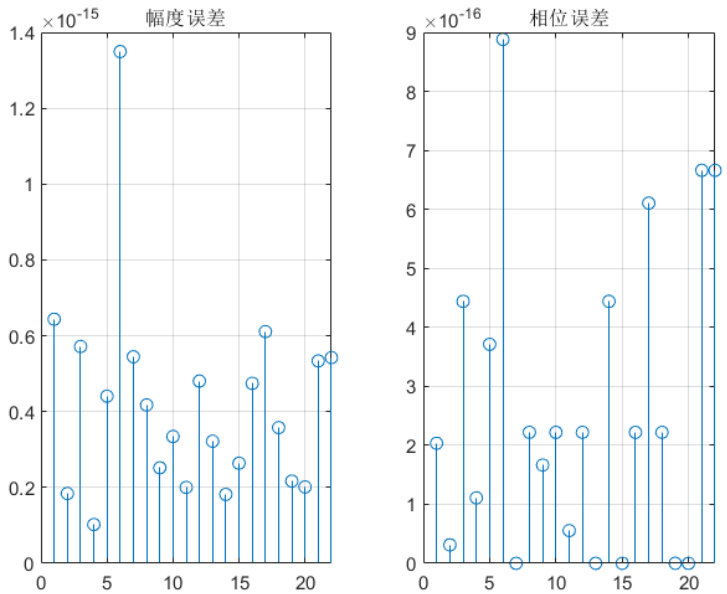

图 18 卷积结果误差

从结果看，无论是幅值误差还是相位误差，其结果都很小，可以认为是计算机表示双精度浮点数时产生的误差导致，其数量级在10^{-15}，完全可以忽略不计。理论上，卷积的结果应均为实数，实际结果中产生的虚部值是引起相位误差的原因，但它的数量级比幅度误差更小，也可以看作是由于浮点数在计算机中的表示引起的。

该子函数的误差明显小于单独使用自编的 fftNew 和 ifftNew 时造成的误差，这是 fftNew 和 ifftNew 程序上的相似性导致的结果（W_N^{nk}和W_N^{-nk}的共轭关系），即正变换和反变换过程中误差相互抵消，提高了最后的精度。

测试当序列长度较大、$x[n]$和$h[n]$长度相差较大的情况。取

$$x[n] = rand(1,6120).* 10;$$
$$h[n] = rand(1,206).* 10;$$

结果如图 19 所示。

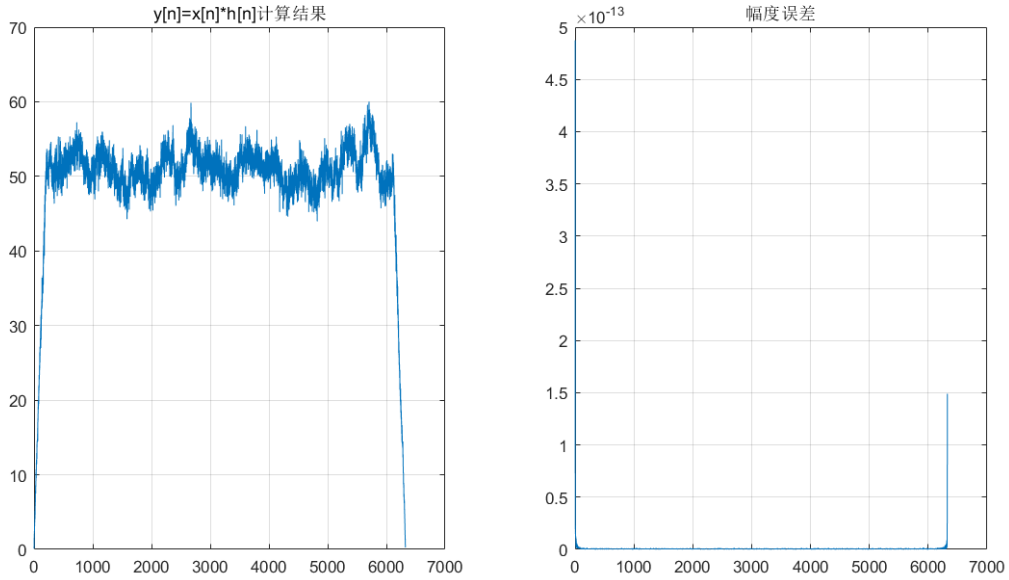

图 19

此时程序的时间在 0.9s 左右（包括绘图时间）。

上述过程表明，编写的离散序列卷积函数可以较精确、快速地计算离散序列的卷积。

四、讨论与展望

在本次实验中，当输入序列长度很大时，fftNew 和 ifftNew 程序的响应时间较长，这是因为当序列长度N落在两个相临二次幂数中间，如2^{20}和2^{21}中间，采用补 0 的方法，需要补的 0 的数量级在2^{19}~2^{20}之间，其数目之多必然导致响应速度变慢。事实上，有以下两种方法可以减少补 0 的数量

①混合基法

仿照以 2 为基本单元进行 FFT 的方法，可以设置以 3、5 等其他数为基的 FFT 算法。在此基础上，对输入序列以这些基为基础进行拆分、分块处理。

这种方法的好处是可以减少插 0 的数目，减少递归层数，但对于一个较大的数，其因式分解又会变得困难。此外，由于大数的因子可以有很多种组合方式，如 9990=1665×2×3=111×9×10，由于因子的顺序不影响运算量，如果要追求运算量尽可能小，应该选择第二种拆解方式，因为其各因子的和更小[1]，这就需要对不同的基设置算法。

②线性调频z变换（$chirp-z$变换或 CZT）算法

[1] 混合基 FFT 算法运算量分析;程志鹏,马琪,竺红卫; 太赫兹科学与电子信息学报;2016 第 14 卷第 6 期

CZT 算法利用卷积计算z平面单位圆内任意螺旋线上的抽样z变换，利用 DFT 的性质，线性卷积可以用圆卷积代替，而圆卷积又可以用 DFT 计算。

这种算法突破了普通 DFT 运算中抽样点均匀分布在$[0,2\pi)$之间、且输入输出序列长度必须一致的问题，可以在不补 0 的前提下以较高分辨率下计算某一频段内的频谱值。但该算法依赖其他 FFT 算法，且过程中对于卷积的计算与选题要求冲突。

考虑到递归层数的增多也会导致响应时间变长，可以利用数据倒序进行蝶形计算，程序如下：

```
clear;
xn = [1,1,4,5,1,4];
M = nextpow2(length(xn));
N = 2^M;
for m = 0:N/2-1
    WN(m+1) = exp(-1j*2*pi/N)^m; %计算旋转因子
end
A = [xn,zeros(1,N-length(xn))];
%数据倒序
J = 0;
for I = 0:N-1
    if I<J
        T = A(I+1);
        A(I+1) = A(J+1);
        A(J+1) = T;
    end
    %下一个倒序数
    K = N/2;
    while J>=K
        J = J-K;
        K = K/2;
    end
    J = J+K;
end
%分级按序进行蝶形运算
for L = 1:M     %分级
    B = 2^(L-1);
    for R = 0:B-1       %各级按序蝶形运算
        P = 2^(M-L)*R;
        for K = R:2^L:N-2
            T = A(K+1)+A(K+B+1)*WN(P+1);
            A(K+B+1) = A(K+1)-A(K+B+1)*WN(P+1);
            A(K+1) = T;
        end
    end
end
disp('输出各存储单元的数据：'),disp(A);
```

该算法的好处是减少递归，增加长序列的响应速度，但仍需要补 0，且实测表明，该算法对长度非2^L的序列补 0 后的还原能力较差，精度较低。

五、总结

本文从最基本的 FFT 原理出发，对基-2FFT 算法进行了改造，使其能够计算任意长度序列的 FFT 和 IFFT。算法的结果基本做到了保持幅度和相位的误差在较小的范围内，且对于较长序列的输入，能够有较快的响应时间。在此基础上，本文利用了改进后的算法计算离散序列的线性卷积，实现了精度和响应速度在可接受范围内的要求。总体来看，本文涉及的算法虽然在精度和响应时间上无法与 MATLAB 自带的函数相比，但是对于处理学习生活中较为一般的序列仍具有较强的实用性。

参考文献：

[1]数字信号处理教程（第五版）;程佩青;清华大学出版社.
[2]混合基 FFT 算法运算量分析;程志鹏,马琪,竺红卫;太赫兹科学与电子信息学报;2016 第 14 卷第 6 期.
[3]按时间抽取的基 2-FFT 算法分析及 MATLAB 实现；张登奇,李宏民,李丹; 2011 年第 38 卷.

附录：主程序

```
clear;
x = [1,1,4,5,1,4];
h = [1,9,1,9,8,1,2,3,3,2,9,7];

figure(1);
stem(x);grid('on');title('离散序列 x[n]时域图');
%fft
%实际值
X = fftNew(x);
figure(2);
subplot(1,2,1);stem(0:2*pi/length(X):2*pi*(length(X)-
1)/length(X),abs(X));grid('on');
xlabel('\omega');ylabel('Amplitude');title('离散序列 x[n]的 FFT 结果 X[n]的
幅值计算结果');
subplot(1,2,2);stem(0:2*pi/length(X):2*pi*(length(X)-
1)/length(X),angle(X));grid('on');
xlabel('\omega');ylabel('Angle');title('离散序列 x[n]的 FFT 结果 X[n]的相位
计算结果');
%理论值
Xt = fft(x);
figure(3);
subplot(1,2,1);stem(0:2*pi/length(Xt):2*pi*(length(Xt)-
1)/length(Xt),abs(Xt));grid('on');
xlabel('\omega');ylabel('Amplitude');title('离散序列 x[n]的 FFT 结果 X[n]的
幅值理论值');
subplot(1,2,2);stem(0:2*pi/length(Xt):2*pi*(length(Xt)-
1)/length(Xt),angle(Xt));grid('on');
xlabel('\omega');ylabel('Angle');title('离散序列 x[n]的 FFT 结果 X[n]的相位
理论值');
%误差
err = abs(Xt - X)./abs(Xt);
figure(4);
stem(0:2*pi/length(err):2*pi*(length(err)-
1)/length(err),err);grid('on');title('幅度误差');

%新序列 fft
x = [5,9,2,9,8,7,68,62,5,1,36,1,4,5,7,5,6];

figure(5);
stem(x);grid('on');title('离散序列 x[n]时域图');
%实际值
X = fftNew(x);
figure(6);
```

```
subplot(1,2,1);stem(0:2*pi/length(X):2*pi*(length(X)-
1)/length(X),abs(X));grid('on');
xlabel('\omega');ylabel('Amplitude');title('离散序列 x[n]的 FFT 结果 X[n]的
幅值计算结果');
subplot(1,2,2);stem(0:2*pi/length(X):2*pi*(length(X)-
1)/length(X),angle(X));grid('on');
xlabel('\omega');ylabel('Angle');title('离散序列 x[n]的 FFT 结果 X[n]的相位
计算结果');
%理论值
Xt = fft(x);
figure(7);
subplot(1,2,1);stem(0:2*pi/length(Xt):2*pi*(length(Xt)-
1)/length(Xt),abs(Xt));grid('on');
xlabel('\omega');ylabel('Amplitude');title('离散序列 x[n]的 FFT 结果 X[n]的
幅值理论值');
subplot(1,2,2);stem(0:2*pi/length(Xt):2*pi*(length(Xt)-
1)/length(Xt),angle(Xt));grid('on');
xlabel('\omega');ylabel('Angle');title('离散序列 x[n]的 FFT 结果 X[n]的相位
理论值');
%误差
err = abs(Xt - X)./abs(Xt);
figure(8);
stem(0:2*pi/length(err):2*pi*(length(err)-
1)/length(err),err);grid('on');title('幅度误差');

%ifft
X = [2,9,5,3,7,12,14,2,6,35,1];
figure(9);
stem(X);grid('on');title('离散序列 X[n]频域图');
%实际值
x = ifftNew(X);
figure(10);
subplot(1,2,1);stem(abs(x));grid('on');
xlabel('n');ylabel('Amplitude');title('离散序列 X[n]的 IFFT 结果 x[n]的幅值
计算结果');
subplot(1,2,2);stem(angle(x));grid('on');
xlabel('n');ylabel('Angle');title('离散序列 X[n]的 IFFT 结果 x[n]的相位计算
结果');
%理论值
xt = ifft(X);
figure(11);
subplot(1,2,1);stem(abs(xt));grid('on');
xlabel('n');ylabel('Amplitude');title('离散序列 X[n]的 IFFT 结果 x[n]的幅值
理论值');
```

```
subplot(1,2,2);stem(angle(xt));grid('on');
xlabel('n');ylabel('Angle');title('离散序列 X[n]的 IFFT 结果 x[n]的相位理论值');
%误差
err = abs(xt - x)./abs(xt);
figure(12);
stem(err);grid('on');title('幅度误差');

%线性卷积
y = DiscreteConvolutionUsingFFT(x,h);
figure(13);
subplot(1,2,1);
stem(abs(y));grid('on');title('y[n]=x[n]*h[n]幅值计算结果');
yt = conv(x,h);
subplot(1,2,2);stem(abs(yt));grid('on');title('y[n]=x[n]*h[n]幅值理论值');
figure(14);
subplot(1,2,1);stem(angle(y));grid('on');title('y[n]=x[n]*h[n]相位计算结果');
subplot(1,2,2);stem(angle(yt));grid('on');title('y[n]=x[n]*h[n]相位理论值');
%误差
err = abs(yt - y)./abs(yt);
perr = angle(yt) - angle(y);
figure(15);
subplot(1,2,1);stem(abs(err));grid('on');title('幅度误差');
subplot(1,2,2);stem(abs(perr));grid('on');title('相位误差');
%长序列线性卷积
x = rand(1,6120).*10;
h = rand(1,206).*10;
y = DiscreteConvolutionUsingFFT(x,h);
figure(16)
subplot(1,2,1);plot(0:length(y)-1,abs(y));title('y[n]=x[n]*h[n]计算结果');grid('on');
yt = conv(x,h);
err = abs(yt - y)./abs(yt);
subplot(1,2,2);plot(abs(err));grid('on');title('幅度误差');
```