DSP Program Homework #2

電通所 Q36114221 蘇沛錦

(1) Convolution Computation:

A real linear convolution program, which can compute the convolution of two 32 real-data, which are $x[n]$=[3, 6, 9,⋯, 96] and $h[n]$=32-2n, for n=1, 2,⋯, 32.

(a) Theoretical derivations

Linear Convolution

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

Recall: Convolution Property of DFT

Circular Convolution

$$x_1[n] \overset{DFT}{\longleftrightarrow} X_1[k]$$

$$x_2[n] \overset{DFT}{\longleftrightarrow} X_2[k]$$

$$x_3[n] = ? \overset{DFT}{\longleftrightarrow} X_1[k]X_2[k]$$

$$x_3[n] = \sum_{m=0}^{N-1} x_1[((m))_N]x_2[((n-m))_N] = x_1[n] \circledast x_2[n]$$

Periodic Convolution ⇨ **Circular Convolution**

Linear Convolution Using FFT

$x_1[n] \longrightarrow L$ points

$x_2[n] \longrightarrow P$ points

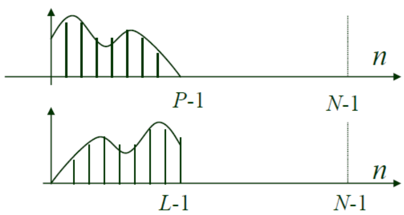
Take N-point DFT for $N > L+P-1$

Linear Convolution $x_3[n] = \sum_{m=-\infty}^{\infty} x_1[m]x_2[n-m] = (L+P-1)$points

$x_1[n] \xrightarrow{N\text{-point FFT}} X_1[k]$ ⇨ Computation needs $N\log_2 N$ multiplications

$x_1[n] \longrightarrow N$ points

$x_2[n] \longrightarrow N$ points



Circular Convolution=Linear Convolution (if $N \geq L+P-1$)

$$x_3[n] = \sum_{k=-\infty}^{\infty} x_1[k]x_2[n-k]$$

L-point $\quad x_1[n] \xrightarrow{DFT} X_1[k]$

P-point $\quad x_2[n] \xrightarrow{DFT} X_2[k]$

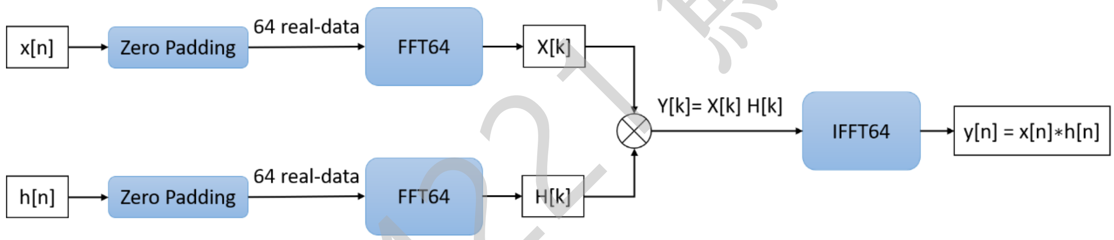
$$x_3[n] = x_1[n] \circledN x_2[n] \xleftarrow{IDFT} X_1[k]X_2[k]$$

Procedures:

1. Call N-point FFT for $x[n]$ to obtain $X[k]$
2. Call N-point FFT for $h[n]$ to obtain $H[k]$
3. Compute $Y[k] = X[k]\ H[k]$
4. Call N-point IFFT for $Y[k]$ to obtain $y[n]$

(b) Flow diagram

因為 $x[n]$和 $h[n]$都是 32 real-data，所以要在後面補上 32 個 0 形成 64 real-data，才能夠進行 64-point FFT 計算。



(c) Algorithms in Matlab

A. a direct real convolution program

```
function [output] = direct_convolution(x,h)
% direct_convolution (A Direct Linear Convolution Program)
% direct_convolution(x,h) computes the convolution of two 32 real-data,
% which are x[n] = [3, 6, 9, ..., 96] and h[n] = 32 - 2n,
% for n = 1, 2, ..., 32.

m = length(x);  % Length of x[n]
n = length(h);  % Length of h[n]
output = zeros(1,m+n-1);
for i = 1:m+n-1
    for j = 1:m
        if(i-j+1 >= 1 && i-j+1 <= n)
            output(i) = output(i) + x(j)*h(i-j+1);
```

```
        end
    end
end

end
```

B. a real convolution program by calling DRFFT64(x, y) once

```
function [output] = convolution_DRFFT64(x,h)
% convolution_DRFFT64 (A Linear Convolution Program by Calling DRFFT64)
% convolution_DRFFT64(x,h) computes the convolution of two 32 real-data,
% which are x[n] = [3, 6, 9, ..., 96] and h[n] = 32 - 2n,
% for n = 1, 2, ..., 32.

N = 32;
x_64 = [x zeros(1,N)];  % Zero data extended padding
h_64 = [h zeros(1,N)];  % Zero data extended padding

[X,H] = DRFFT64(x_64,h_64);     % Calling DRFFT64 once
IFFT64_ouput = IFFT64(X.*H);    % Calling IFFT64 IFFT program once
output = real(IFFT64_ouput(1:2*N-1));

end
```

(d) Complexity analyses

A. Direct Convolution

$$
\begin{aligned}
&1 + 2 + \cdots + (N-1) + N + (N-1) + \cdots + 2 + 1 \\
&= 2\left[\frac{(N-1)+1}{2}(N-1)\right] \\
&= N^2
\end{aligned}
$$

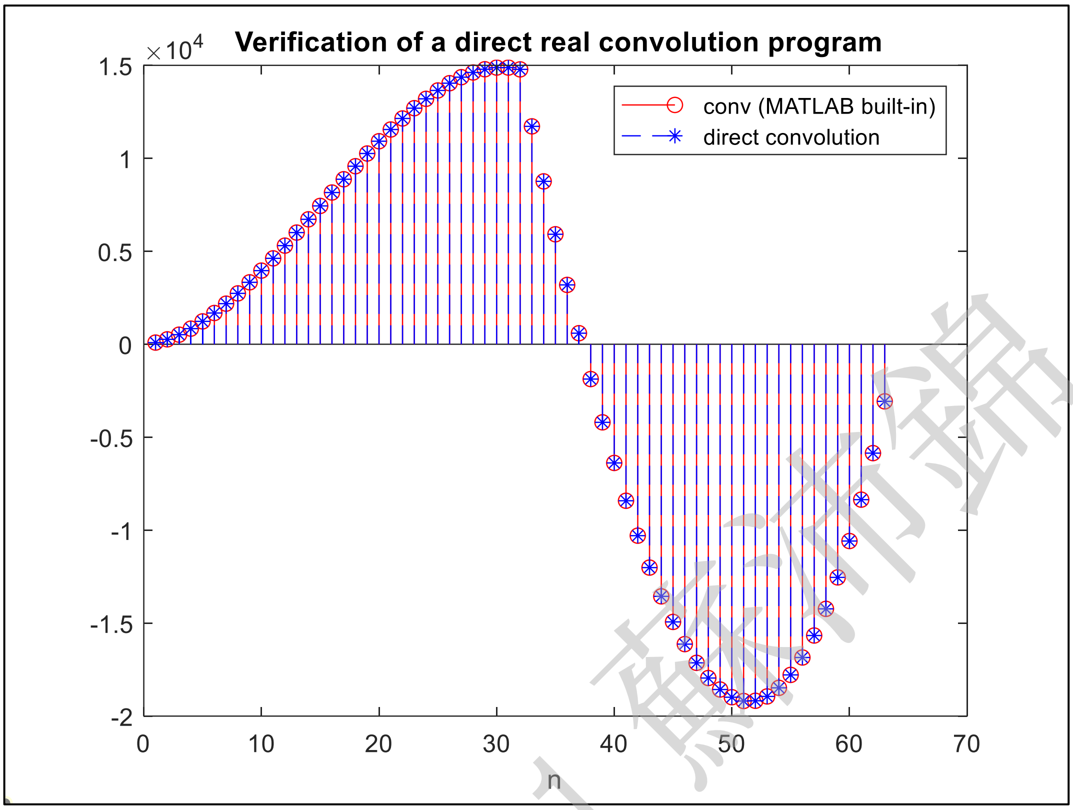
B. FFT Approach

1.	$2N$-point DFT	$x[n]$	$\rightarrow$	$X[k]$		FFT = $2N\log_2 N$
2.	$2N$-point DFT	$h[n]$	$\rightarrow$	$H[k]$		FFT = $2N\log_2 N$
3.	Multiplication	$x[n]*h[n]$	$\rightarrow$	$X[k]\ H[k]$		$2N$
4.	$2N$-point IDFT	$Y[k]$	$\rightarrow$	$y[n]$		FFT = $2N\log_2 N$

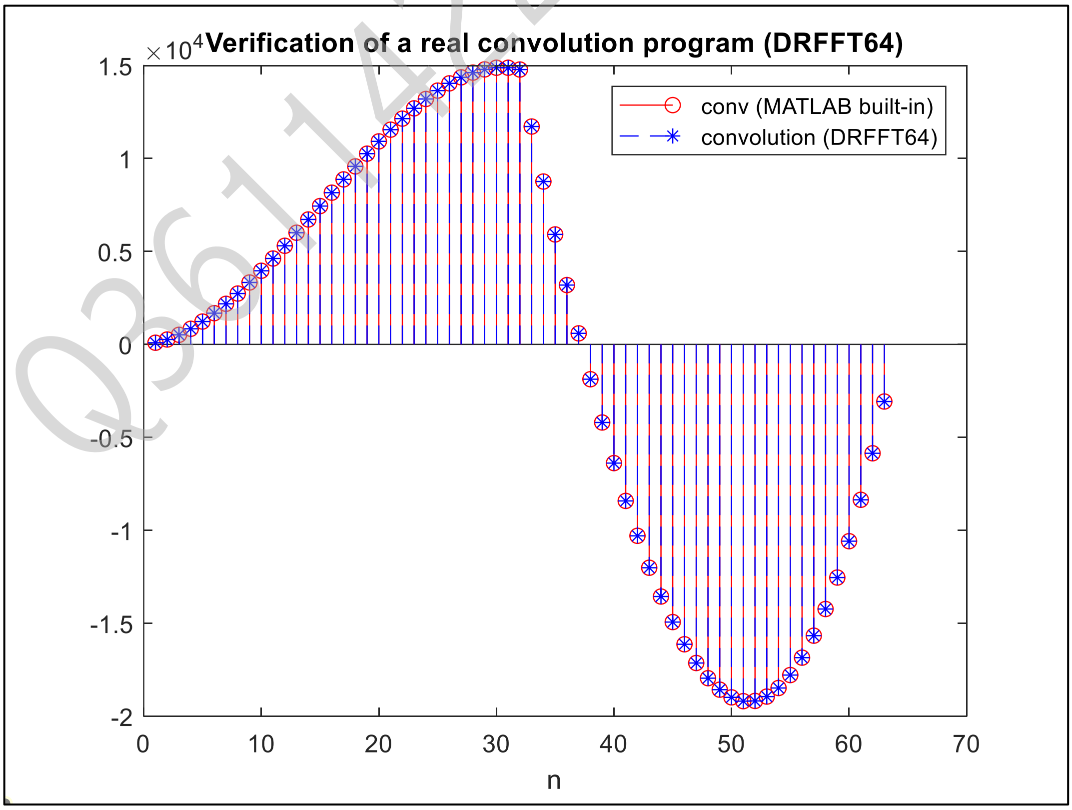
Total Computation: 2 DFT + $2N$ multiplications + 1 IDFT = $\boxed{6N\log_2 N + 2N}$

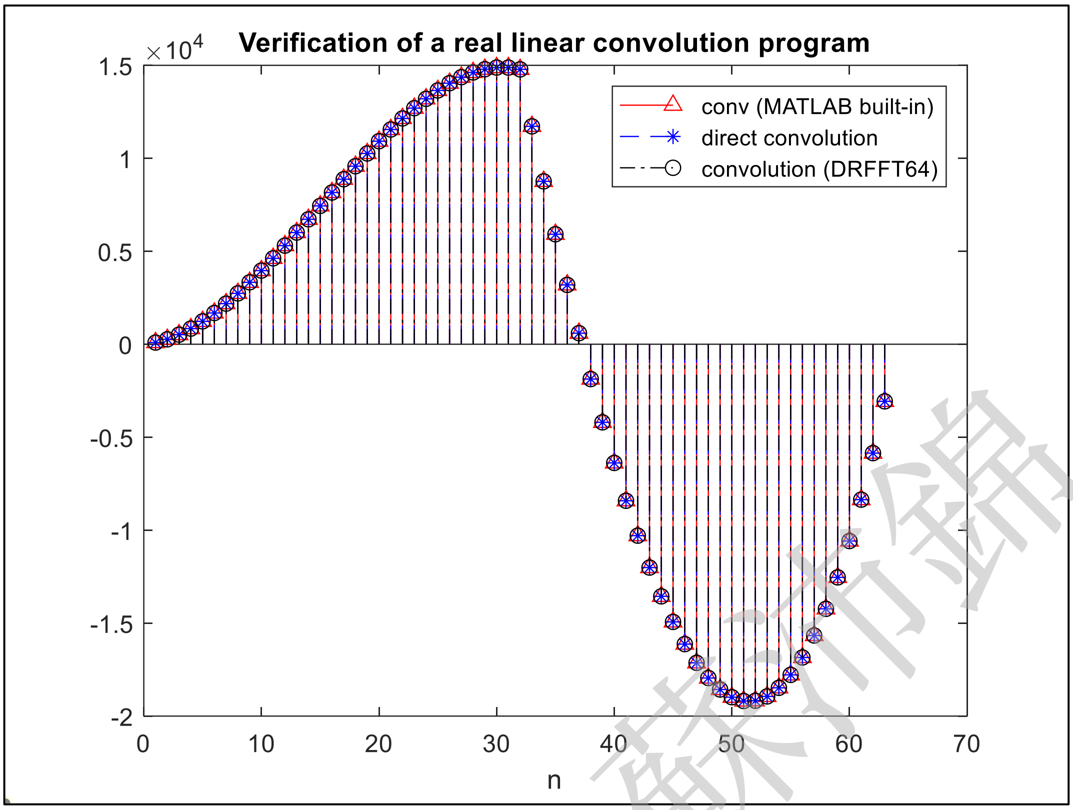
(e) Verification by programs

A. a direct real convolution program



B. a real convolution program by calling DRFFT64(x, y) once





(2) Autocorrelation Computation:

A real data autocorrelation program, which can compute the autocorrelation of $x[n] = n*(-1)^n$, for n=1, 2,…, 32 where autocorrelation is defined as:

$$R(k) = \sum_{n=-\infty}^{\infty} x[n]x[n+k] \quad \text{for } -(N-1) < k < (N-1)$$

(a) Theoretical derivations

$$x[n] \leftrightarrow X[k]$$
$$h[n] \leftrightarrow H[k]$$
$$y[n] = x[n] * h[n] \leftrightarrow X[k]H[k]$$

$$R(k) = \sum_{k=-\infty}^{\infty} x[k]x[n+k] = x[n] * x[-n] \quad \leftrightarrow \quad X[k]X^*[k]$$

Compute Autocorrelation by DFT

1. Form a N-point sequence by augmenting $x[n]$ (a 32 real-data) with 32 zero samples ($N = 64$)
2. Compute N-point DFT:

$$X[k]=\sum_{n=0}^{N-1}x[n]e^{-j(2\pi/N)kn} \quad \text{for} \quad k=0,1,\ldots,N-1$$

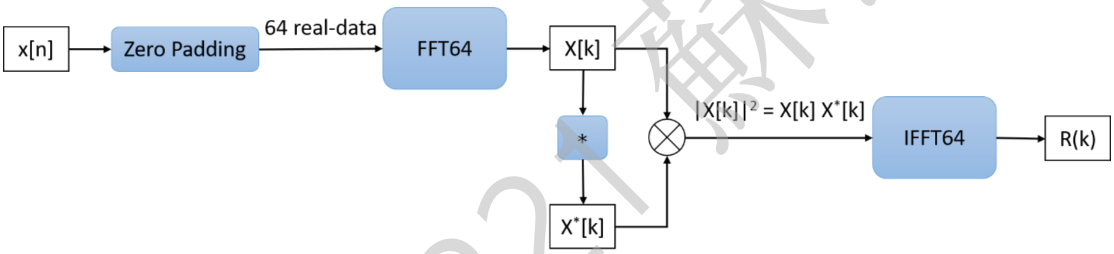
3. Compute $\left|X[k]\right|^2 = X[k]X^*[k]$ for $k=0,1,\ldots,N-1$

4. Compute the inverse DFT of $\left|X[k]\right|^2$ to obtain

$$R(m)=\frac{1}{N}\sum_{n=0}^{N-1}\left|X[k]\right|^2 e^{+j(2\pi/N)kn} \quad \text{for} \quad m=0,1,\ldots,N-1$$

(b) Flow diagram

因為 $x[n]$是 32 real-data，所以要在後面補上 32 個 0 形成 64 real-data，才能夠進行 64-point FFT 計算。



(c) Algorithms in Matlab

A. a direct real autocorrection computation program

```
function [output] = direct_autocorrelation(x)
% direct_autocorrelation (A Direct Real Autocorrelation Program)
% direct_autocorrelation(x) computes the autocorrelation of x[n]=n*(-1)^n,
% for n = 1, 2, ..., 32.

m = length(x);  % Length of x[n]
Rx = zeros(1,2*m-1);
for i = 1:m
    Rx(i)=sum(x(i:m).*x(1:m-i+1));
end
Rx_time_reverse = Rx(1+mod(0:-1:1-m,m));    % Time Reverse (Modulo 32)
output = [Rx_time_reverse(2:m), Rx(1:m)];


end
```

B. a computation program by calling DRFFT64 (x, y) once

```
function [output] = autocorrelation_DRFFT64(x)
% autocorrelation_DRFFT64 (Real Autocorrelation Program by Calling DRFFT64)
% autocorrelation_DRFFT64(x) computes the autocorrelation of x[n]=n*(-1)^n,
% for n = 1, 2, ..., 32.

N = 32;
x_64 = [x zeros(1,N)];      % Zero data extended padding

[X1,X2] = DRFFT64(x_64,x_64);     % Calling DRFFT64 once
Rx = IFFT64(X1.*conj(X2));        % Calling IFFT64 IFFT program once
Rx_time_reverse = Rx(1+mod(0:-1:1-N,N));    % Time Reverse (Modulo N)
output = [real(Rx_time_reverse(2:N)), real(Rx(1:N))];

end
```

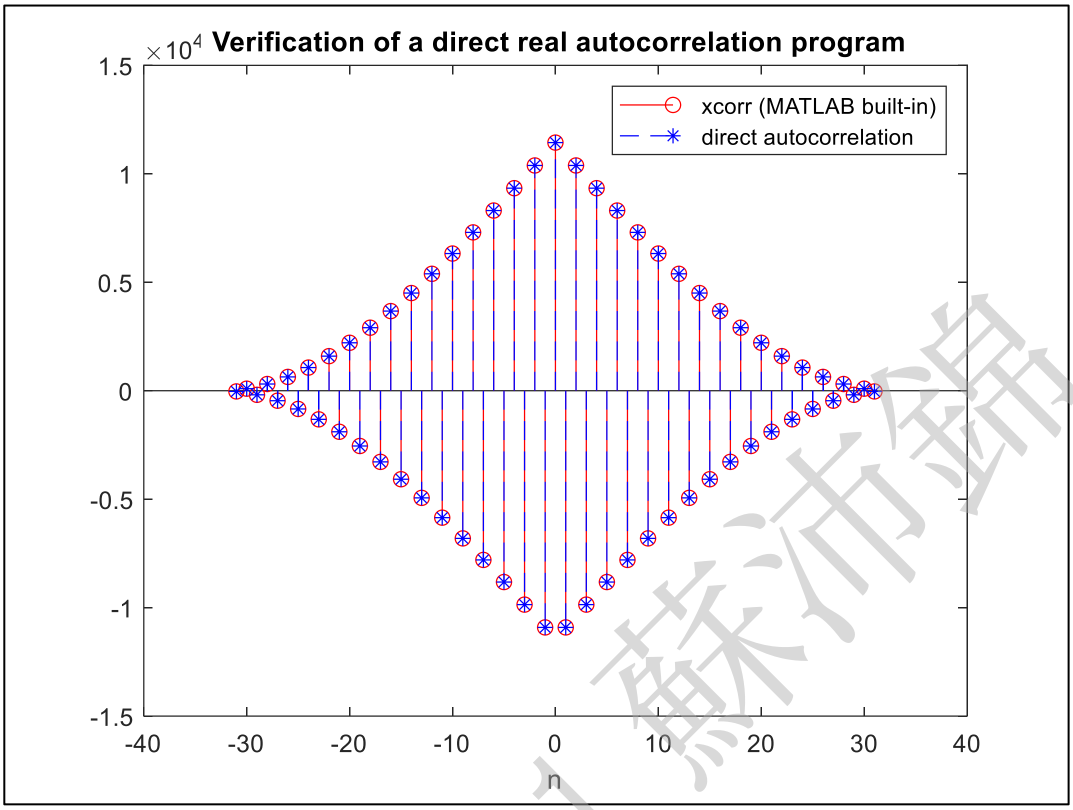
(d) Complexity analyses

Autocorrelation 和 Convolution 的差別在於 Autocorrelation 的輸入只有 $x[n]$，並且透過 FFT64 轉換成 $X[k]$後，還需要取共軛求出 $X^*[k]$。之後將 $X[k]$和 $X^*[k]$相乘(也就是$|X[k]|^2$)再經由 IFFT64 進行 Inverse FFT 轉換，最終輸出結果才是 $x[n]$的自相關 $R(k)$。所以 Autocorrelation 比 Convolution 多了一項取共軛的運算，但是**取共軛並不影響最後的複雜度**，因此 <u>Autocorrelation 和 Convolution 有相同的複雜度</u>。

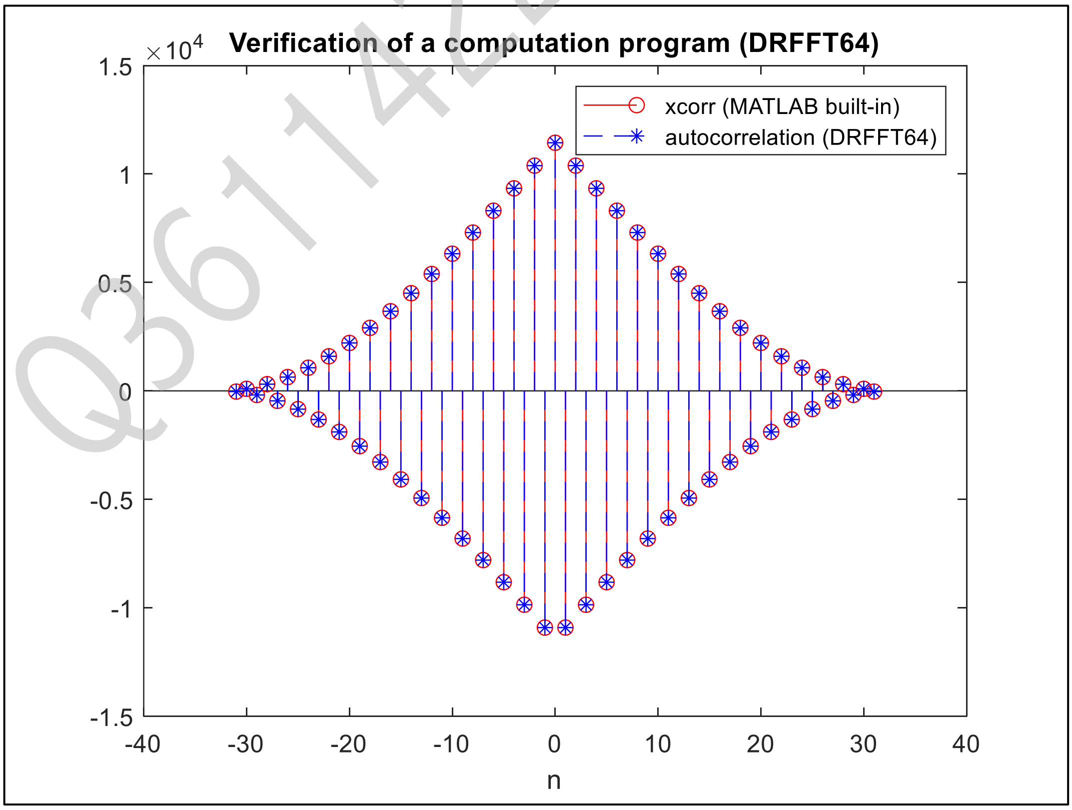
Total Computation: 2 DFT + $2N$ multiplications + 1 IDFT = $\boxed{6N\log_2 N + 2N}$

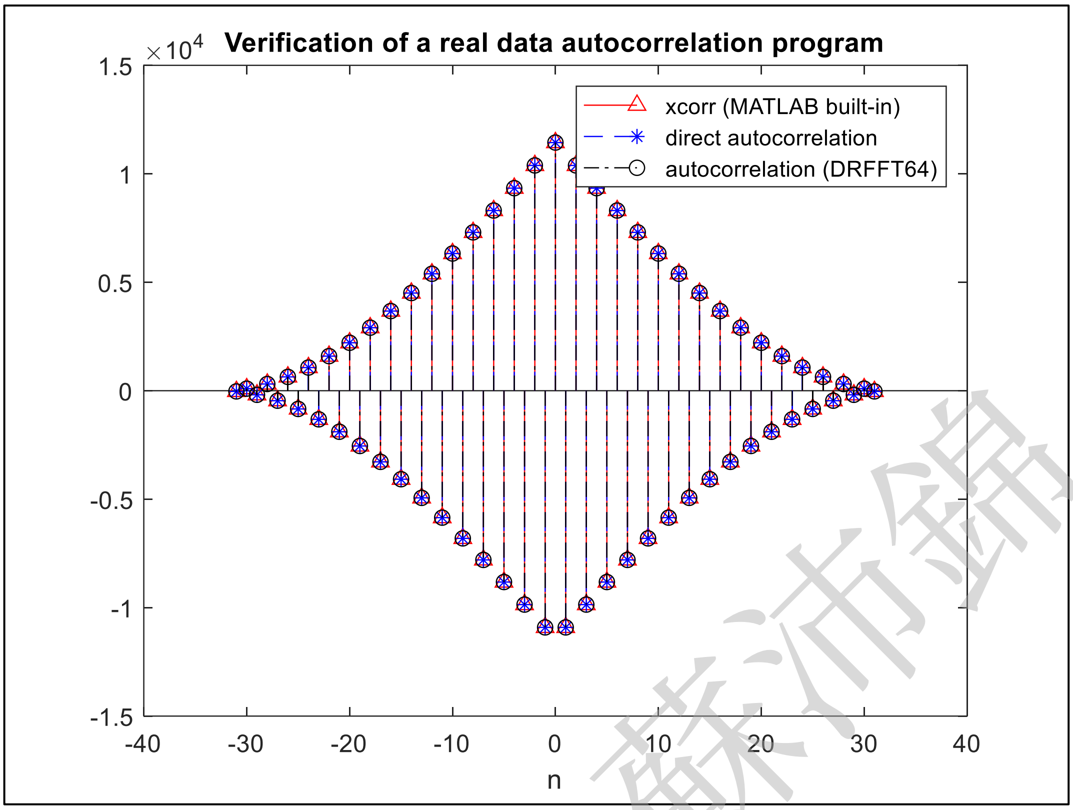
(e) Verification by programs

A. a direct real autocorrection computation program



B. a computation program by calling DRFFT64 (x, y) once




Verification of a real data autocorrelation program
$\times 10^4$
1.5
1
0.5
0
-0.5
-1
-1.5
-40
-30
-20
-10
0
10
20
30
40
n
xcorr (MATLAB built-in)
direct autocorrelation
autocorrelation (DRFFT64)