Program Homework #2

Problem: With the double real 64-point FFT program DRFFT64(x, y) developed in Homework#7, please write

(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.

(1-a) a direct real convolution program

(a) Theoretical Derivation

For complex-valued functions f, g defined on the set Z of integers, the discrete convolution of f and g is given by

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m],$$

(b) Flow Diagram

None

(c) Algorithms in Matlab

```
function y_my = conv_direct(x,h)
y = zeros(1,32 + 32 -1);

for i = 1: 64
    y(i) = 0;
    for j = 1: 32
        if (i - j < 1 || i - j > 32)
            continue;
        end
        y(i) = y(i) +  x(i - j) * h(j); % convolve: multiply and accumulate
    end
end
y_my = [y(2:end)];
```

(d) Complexity Analysis

Requires N arithmetic operations per output value and N^2 operations for N outputs. $O(N^2)$

(e) Verification by Program

```
%(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.
%Please write:
%(a) a direct real convolution program;
clear;
x  = [3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72...
    75 78 81 84 87 90 93 96];
h = [30 28 26 24 22 20 18 16 14 12 10 8 6 4 2 0 -2 -4 -6 -8 -10 -12 -14 -16....
    -18 -20 -22 -24 -26 -28 -30 -32];
y_my = conv_direct(x,h);
y_original = conv(x,h);
error = y_my - y_original;
error = norm(error.*error);
```

Error is zero.

(1-b) a real convolution program by calling DRFFT64(x, y) once

(a) Theoretical Derivation

Assume $x_1[n]$ is of length L , and $x_2[n]$ is of length P . Since there are N-points after linear convolution, compute this via FFT method, we have to take N-point FFT of $x_1[n]$ and $x_2[n]$. Then the N-point circular convolution would become equal to the linear convolution.

$$\Rightarrow\ x_1[n] \circledN x_2[n] \xleftrightarrow{\mathrm{FFT_N}} X_1[k]X_2[k]$$

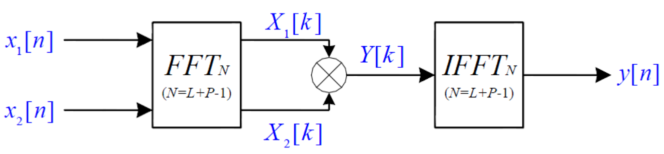
$$\Rightarrow\ \begin{cases} x_1[n] \xleftrightarrow{\mathrm{DFT_N}} X_1[k] \\ x_2[n] \xleftrightarrow{\mathrm{DFT_N}} X_2[k] \end{cases}$$

$$\Rightarrow\ Y[k] = X_1[k]X_2[k]$$

$$\Rightarrow\ Y[k] \xleftrightarrow{\mathrm{IFFT_N}} y[n]$$

The result sequence $y[n]$ can be obtained the fastest via the FFT method.

(b) Flow Diagram



(c) Algorithms in Matlab

```
function out =conv_drfft64(x,y)

N = 32 + 32;

x = [x zeros(1,N-length(x))];
y = [y zeros(1,N-length(y))];

[Fx,Fy] = drfft64(x,y);
Fc = Fx.*Fy;

out = ifft64(Fc);
out = [out(1:end-1)];
out = round(real(out)*10^7)/10^7;
```

(d) Complexity Analysis

Since we use the function **'conv_drfft64'** once, which can compute two N-point real FFT's by a single N-point complex FFT, it needs $2N \sim N\log_2 N$ complex multiplications and additions.

The multiplication of ${}_1X[k]$ and ${}_2X[k]$ also needs N complex multiplications to get $Y[k]$.

Then $Y[k]$ is transformed by N-point IFFT to get the result of linear convolution $y[n]$, this needs $N\log_2 N$ complex multiplications and additions again.

Hence the computational complexity here is $O(n\log_2 n)$.

(e) Verification by Programs

```
%(b) a real convolution program by calling DRFFT64(x, y) once.
clear;

x = [3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72....
    75 78 81 84 87 90 93 96];
h = [30 28 26 24 22 20 18 16 14 12 10 8 6 4 2 0 -2 -4 -6 -8 -10 -12 -14 -16....
    -18 -20 -22 -24 -26 -28 -30 -32];

y_my = conv_drfft64(x,h);
y_original = conv(x,h);

error = y_my - y_original;
error = norm(error.*error);
```

Error is zero.

(2) Autocorrelation Computation: A real data autocorrelation program, which can compute the autocorrelation of x[n] = n*(-1)^n, for n=1, 2,…, 32.

(2-a) a direct real autocorrection computation program

(a) Theoretical Derivation

在訊號處理中，上面的定義通常不進行歸一化，即不減去均值並除以變異數。當自相關函數由均值和變異數歸一化時，有時會被稱作自相關係數。

$$R_{ff}(\tau) = (f * g_{-1}(\overline{f}))(\tau) = \int_{-\infty}^{\infty} f(u+\tau)\overline{f}(u)\,\mathrm{d}u = \int_{-\infty}^{\infty} f(u)\overline{f}(u-\tau)\,\mathrm{d}u$$

(b) Flow Diagram

None

(c) Algorithms in Matlab

```
function y_my = autocorr_direct(x,h)
N = 32;
L = 2*N-1;
x = [x zeros(1, L-length(x))];
y_my = zeros(1,L);

for k=1:L
    for i=1:L
        if((i - k + N)<1 || i > N)
            continue;
        end
        y_my(k) =y_my(k)+ x(i)*x(i - k + N);
    end
end
```

(d) Complexity Analysis

Requires N arithmetic operations per output value and N^2 operations for N outputs. $O(N^2)$

(e) Verification by Program

```
%(2) Autocorrelation Computation: A real data autocorrelation program,
%which can compute the autocorrelation of x[n] = n*(-1)^n, for n=1,2,...,32
%Please write:
% (a) a direct real autocorrelation program;
clear;

for i = 1:32
    x(i) = i * (-1)^i;
end
y_original = xcorr(x);
y_my = autocorr_direct(x);

error = y_my - y_original;
error = norm(error.*error);
```

Error is 3.5532e-24.

(2-b) a computation program by calling DRFFT64 (x, y) once

(a) Theoretical Derivation

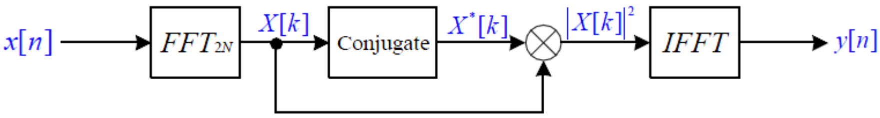
For real case,

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

$$= \sum_{m=0}^{2N-1} x\left[((-m))_{2N}\right] x\left[((k-m))_{2N}\right] \xleftrightarrow{\text{FFT}_{2N}} X[k]X^*[k] = \left|X[k]\right|^2$$

After taking inverse IFFT of $|X(k)|^2$, we move negative lags before positive lags.

(b) Flow Diagram



(c) Algorithms in Matlab

```
function out = autocorr_drfft64(x)

L = length(x);
N = 2^nextpow2(2*L-1);
x = [x zeros(1, N-L)];

FX = fft64(x);

Ty = ifft64(abs(FX).^2);
Ty = real(Ty);

%Move negative lags before positive lags
out = [zeros(1, 2*L-1)];
out = [Ty(L+length(x)-2*L+2 : end),Ty(1 : L)];
```

```
%delete errors produzed from finite precision accuracy of computer
for i = 1:N
    if abs(imag(x(i))) < 1e-11
        x(i) = real(x(i));
    end
    if abs(real(x(i))) < 1e-11
        x(i) = x(i)-real(x(i));
    end
end
```

(d) Complexity Analysis

We take the $2N$-point FFT of $x[n]$ to get $X[k]$, and multiply it with its conjugate version to get $|X[k]|^2$, and take the $2N$-point IFFT to get result. Total complex multiplications are $2N$ $4N\log_2 2N$, and complex additions are $4N\log_2 2N$.

The computational complexity is $O(n\log_2 n)$.

(e) Verification by Program

```
%(b) a computation program by calling DRFFT64 (x, y) once.
clear;
N = 32;
L = 2*N-1;

for i = 1:N
    x(i) = i * (-1)^i;
end
x = [x zeros(1,N-length(x))];
y_original = xcorr(x);
y_my = autocorr_drfft64(x);

error = y_my - y_original;
error = norm(error.*error);
```

Error is 2.2417e-23.