DSP Program Homework #1

電通所 Q36114221 蘇沛錦

(1) Please write a complex 32-point FFT program (in Matlab or C-program), called **X = FFT32(x)**, where **x** is a complex 32-point vector.

(a) Theoretical derivations

Decimation-in-Frequency FFT Algorithm

N – point DFT : $N = 2^m$

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{nk}=\sum_{n=0}^{N/2-1}x[n]W_N^{nk}+\sum_{n=N/2}^{N-1}x[n]W_N^{nk}$$

Even Frequency:

$$X[2r]=\sum_{n=0}^{N-1}x[n]W_N^{2nr}=\sum_{n=0}^{N/2-1}x[n]W_N^{2nr}+\sum_{n=N/2}^{N-1}x[n]W_N^{2nr}$$

$$=\sum_{n=0}^{N/2-1}x[n]W_N^{2nr}+\sum_{n=0}^{N/2-1}x[n+(N/2)]W_N^{2r[n+(N/2)]}$$

$$=\sum_{n=0}^{N/2-1}(x[n]+x[n+(N/2)])W_{N/2}^{nr} \qquad \boxed{N/2-\text{point DFT}}$$

$$W_N^{2nr}=W_{N/2}^{nr}$$

$$W_N^{2r[n+(N/2)]}=W_N^{rN}W_N^{2rn}=W_{N/2}^{nr}$$

Odd Frequency:

$$X[2r+1]=\sum_{n=0}^{N-1}x[n]W_N^{n(2r+1)}=\sum_{n=0}^{N/2-1}x[n]W_N^{n(2r+1)}+\sum_{n=N/2}^{N-1}x[n]W_N^{n(2r+1)}$$

$$=\sum_{n=0}^{N/2-1}(x[n]W_N^{n})W_N^{2nr}+\sum_{n=0}^{N/2-1}x[n+(N/2)]W_N^{n+(N/2)}W_N^{2nr}$$

$$=\sum_{n=0}^{N/2-1}\{(x[n]-x[n+(N/2)])W_N^{n}\}W_{N/2}^{nr} \qquad \boxed{N/2-\text{point DFT}}$$

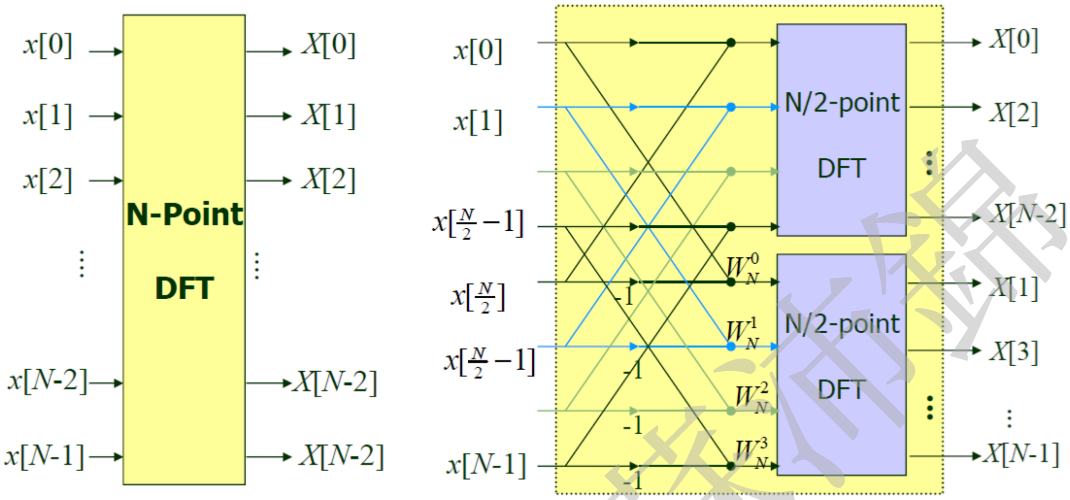
$$W_N^{n(2r+1)}=W_N^{n}W_{N/2}^{nr}$$

$$W_N^{(2r+1)[n+(N/2)]}=W_N^{rN}W_N^{N/2}W_N^{n}W_N^{2rn}=-W_N^{n}W_{N/2}^{nr}$$

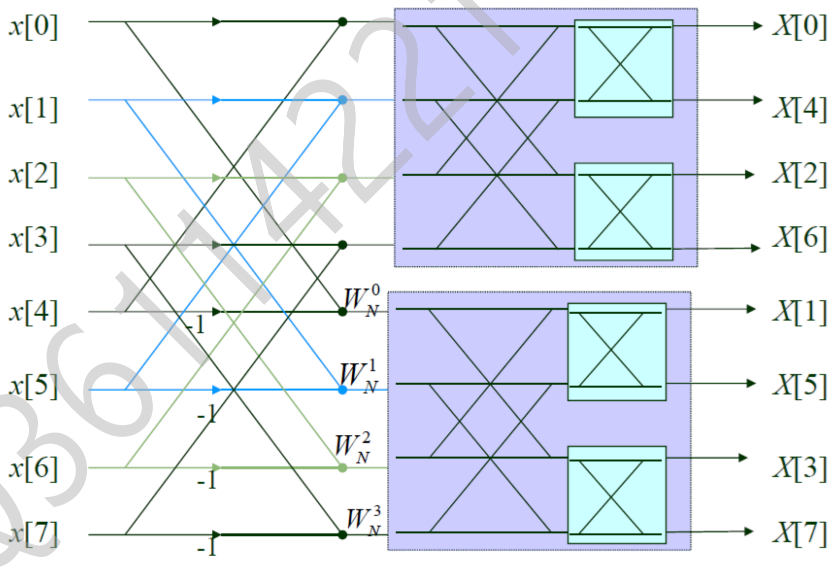
(b) Flow diagram

2/N Point DFT

$$\begin{cases} X[2r] = \sum_{n=0}^{N/2-1} (x[n] + x[n+(N/2)]) W_{N/2}^{nr} \\ X[2r+1] = \sum_{n=0}^{N/2-1} \{(x[n] - x[n+(N/2)]) W_N^n\} W_{N/2}^{nr} \end{cases}$$



以 8-Point DIF FFT Algorithm 為例:



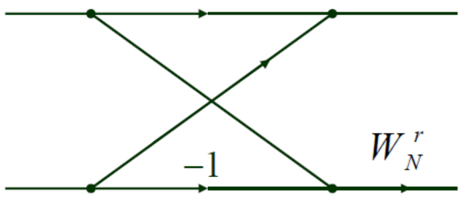
(c) Algorithms in Matlab

```
function [X] = FFT32(x)
% FFT32 (Decimation-in-frequency FFT Algorithm)
% FFT32(x) computes 32-point FFT of x and returns the result,
% where x is a complex 32-point vector.
```

```matlab
N = 32;

W_N = exp(-1i*2*pi./2.^(log2(N):-1:1));  % Complex sinusoidal components

for m = 1:log2(N)   % m = log2(N) stages (N-point FFT)
    for k = 1:2^(m-1)   % Each stage with k = 2^(m-1) parts
        for btf_Num = 1:N/(2^m)     % Each part needs N/(2^m) butterflies
            btf_Idx_1 = (k-1)*N/(2^(m-1)) + btf_Num;
            btf_Idx_2 = (k-1)*N/(2^(m-1)) + btf_Num + N/(2^m);
            X1 = x(btf_Idx_1);
            X2 = x(btf_Idx_2);
            x(btf_Idx_1) = X1 + X2;
            x(btf_Idx_2) = (X1 - X2)*(W_N(m)^(btf_Num-1));
        end
    end
end

X = x(bitrevorder((1:N)-1)+1);  % Permute data into bit-reversed order

end
```

(d) Complexity analyses

$N = 2^m$ ⟹ $m = \log_2 N$ stages: each stage with $N/2$ butterflies



Each butterfly needs: 1 complex multiplications
2 complex additions

Each stage needs: $N/2$ complex multiplications
N complex additions

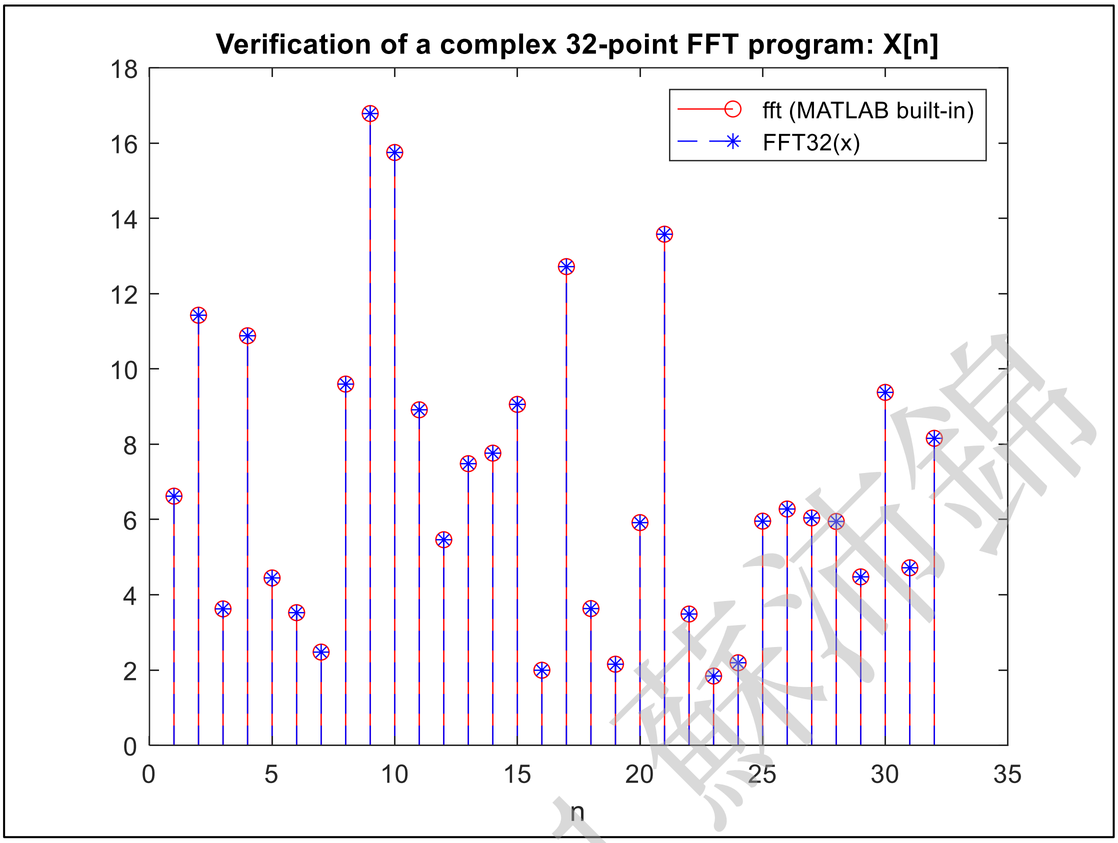
Total computation needs:

$0.5N\log_2 N$ complex multiplications

$N\log_2 N$ complex additions

$$\boxed{N = 32 \Longrightarrow \text{複雜度為}\ \ 2N\log_2 2N}$$

(e) Verification by programs



(2) Please use **FFT32** to write a complex 64-point FFT program, called **FFT64(x)**, where **x** is a complex 64-point vector.

(a) Theoretical derivations

Decimation-in-Time FFT Algorithm

$$\boxed{N-\text{point DFT}: N = 2^m}$$

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} = \sum_{r=0}^{N/2-1} x[2r] W_N^{(2r)k} + \sum_{r=0}^{N/2-1} x[2r+1] W_N^{(2r+1)k}$$

$$= \underbrace{\sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{rk}}_{N/2-\text{point DFT}} + W_N^k \underbrace{\sum_{r=0}^{N/2-1} [x[2r+1] W_{N/2}^{rk}}_{N/2-\text{point DFT}} = G[k] + W_N^k H[k]$$

$$\boxed{X[k] = G[k] + W_N^k H[k]}$$

$$X[0] = G[0] + W_N^0 H[0]$$

$$\vdots$$

$$X[3] = G[3] + W_N^3 H[3]$$

$$X[4] = G[4] + W_N^4 H[4]$$

$$\vdots$$

$$W_N^{2rk} = W_{N/2}^{rk}$$

$$e^{\frac{-j2\pi(2rk)}{N}} = e^{\frac{-2j\pi k}{N/2}} = W_{N/2}^{rk}$$

$$G[k]=\sum_{r=0}^{N/2-1} x[2r]W_{N/2}^{rk} \qquad H[k]=\sum_{r=0}^{N/2-1} x[2r+1]W_{N/2}^{rk}$$

$G[k]$ and $H[k]$ are periodic sequences with period $N/2$

For $k < N/2$: $X[k]=G[k]+W_N^k H[k]$

For $k \geq N/2$: $(k = k'+N/2)$ $(k' < N/2)$

$$X[k]=G[k]+W_N^k H[k]$$

$$X[k'+\tfrac{N}{2}]=G[k'+\tfrac{N}{2}]+W_N^{k'+N/2} H[k'+\tfrac{N}{2}]$$

$$=G[k']+W_N^{k'+N/2} H[k']$$

$$g[r]=x[2r] \qquad h[r]=x[2r+1]$$

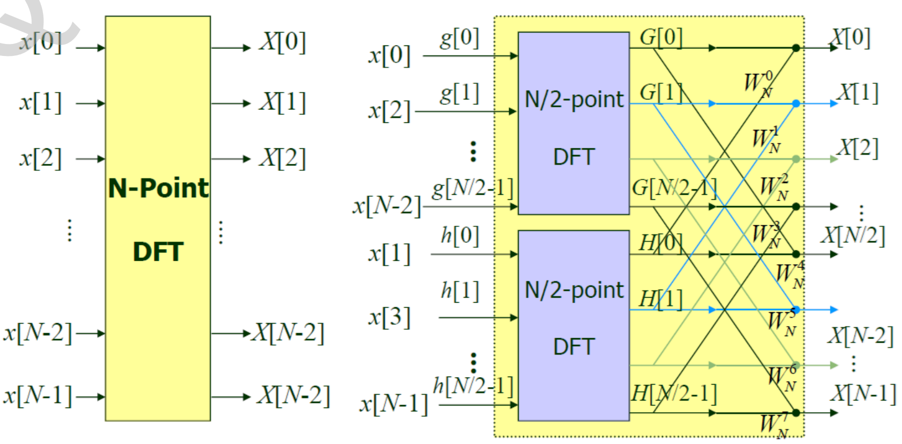
$$G[k]=\sum_{r=0}^{N/2-1} g[r]W_{N/2}^{rk}=\sum_{\ell=0}^{N/4-1} g[2\ell]W_{N/2}^{(2\ell)k}+\sum_{\ell=0}^{N/4-1} g[2\ell+1]W_{N/2}^{(2\ell+1)k}$$

$$=\sum_{\ell=0}^{N/4-1} g[2\ell]W_{N/4}^{\ell k}+W_{N/2}^{k}\sum_{\ell=0}^{N/4-1} [g(2\ell+1)]W_{N/4}^{\ell k}$$

$$H[k]=\sum_{r=0}^{N/2-1} h[r]W_{N/2}^{rk}=\sum_{\ell=0}^{N/4-1} h[2\ell]W_{N/2}^{(2\ell)k}+\sum_{\ell=0}^{N/4-1} h[2\ell+1]W_{N/2}^{(2\ell+1)k}$$

$$=\sum_{\ell=0}^{N/4-1} h[2\ell]W_{N/4}^{\ell k}+W_{N/2}^{k}\sum_{\ell=0}^{N/4-1} [h(2\ell+1)]W_{N/4}^{\ell k}$$

(b) Flow diagram

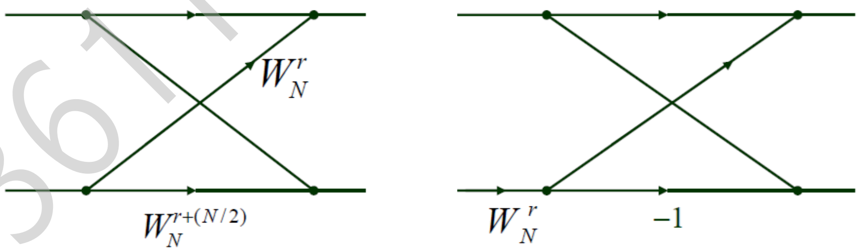


(c) Algorithms in Matlab

```
function [X] = FFT64(x)
% FFT64 (Decimation-in-time FFT Algorithm)
% FFT64(x) computes 64-point FFT of x and returns the result,
% where x is a complex 64-point vector.

N = 64;

% Use FFT32
G = FFT32(x(1:2:N));    % 32-point FFT (Even part of x)
H = FFT32(x(2:2:N));    % 32-point FFT (Odd part of x)

W_N = exp(-1i*2*pi/N.*(0:N/2-1));  % Complex sinusoidal components

% Butterfly
X(1:N/2) = G + H.*W_N;
X(N/2+1:N) = G - H.*W_N;

end
```

(d) Complexity analyses

$N = 2^m \Rightarrow m = \log_2 N$ stages: each stage with $N/2$ butterflies



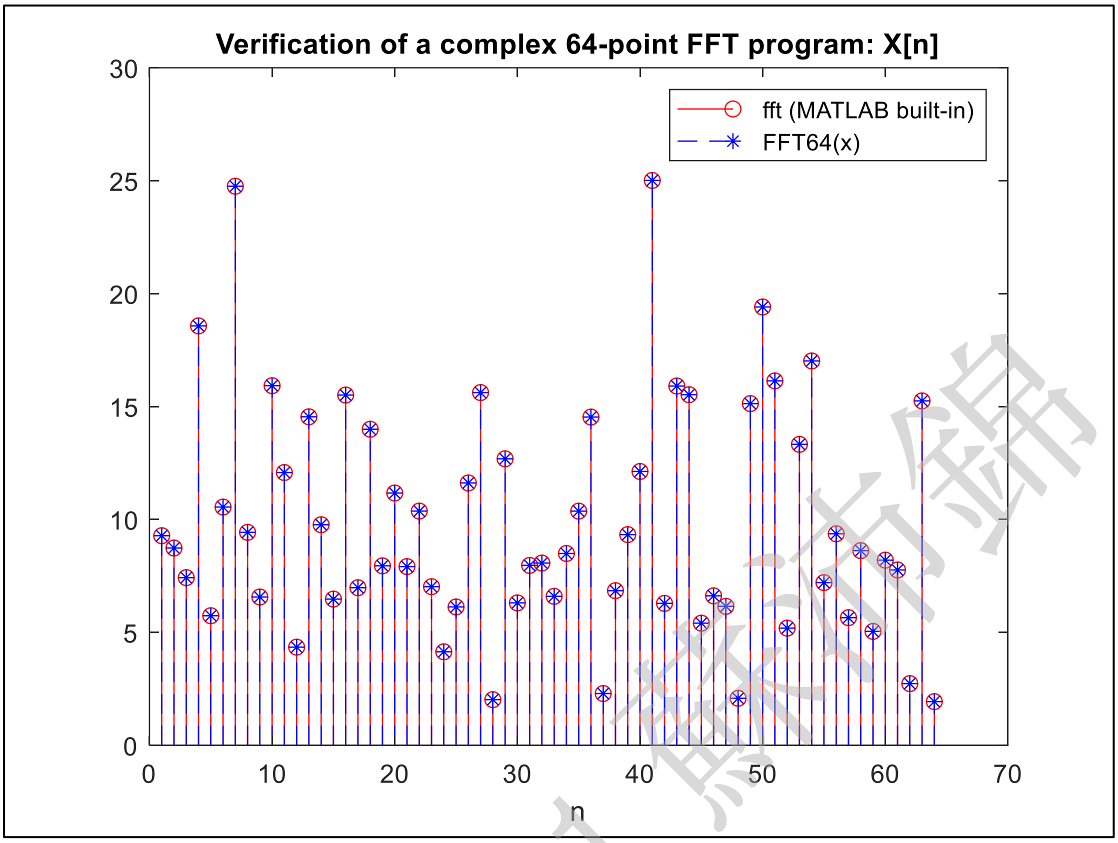
Each butterfly needs: 1 complex multiplication
2 complex additions

Each stage needs: $N/2$ complex multiplications
N complex additions

Total computation needs:
$0.5N\log_2 N$ complex multiplications
$N\log_2 N$ complex additions

$\boxed{N = 64 \Longrightarrow \text{複雜度為 } 2N\log_2 2N}$

(e) Verification by programs



(3) Please write a double-real 64-point FFT program, called **DRFFT64(x, y)**, where **x**, **y** are two real 64-point data vector, by only calling **FFT64** FFT program once.

(a) Theoretical derivations

Symmetry Property of DFT

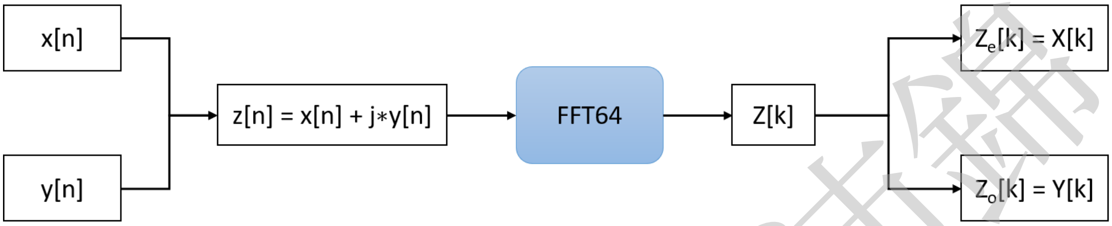
$x[n] \leftrightarrow X[k]$
$x^*[n] \leftrightarrow X^*[((-k))_N]$
$\mathrm{Re}\{x[n]\} = \dfrac{x[n]+x^*[n]}{2} \leftrightarrow X_e[k] = \dfrac{X[k]+X^*[((-k))_N]}{2}$
$\mathrm{Im}\{x[n]\} = \dfrac{x[n]-x^*[n]}{2j} \leftrightarrow X_o[k] = \dfrac{X[k]-X^*[((-k))_N]}{2}$

1. 把 2 個實數訊號$x[n]$和$y[n]$合併成一個複數訊號$z[n]$，其中$z[n]$的實數部分為$x[n]$；而虛數部分為$y[n]$，表示成

$$z[n] = x[n] + j * y[n]$$

2. 然後將$z[n]$透過 FFT64 進行 FFT 轉換得到$Z[k]$。
3. 最後再藉由 Symmetry Property，求出$Z[k]$的 Even function 和 Odd function，分別代表$x[n]$和$y[n]$的 FFT 轉換。

(b) Flow diagram



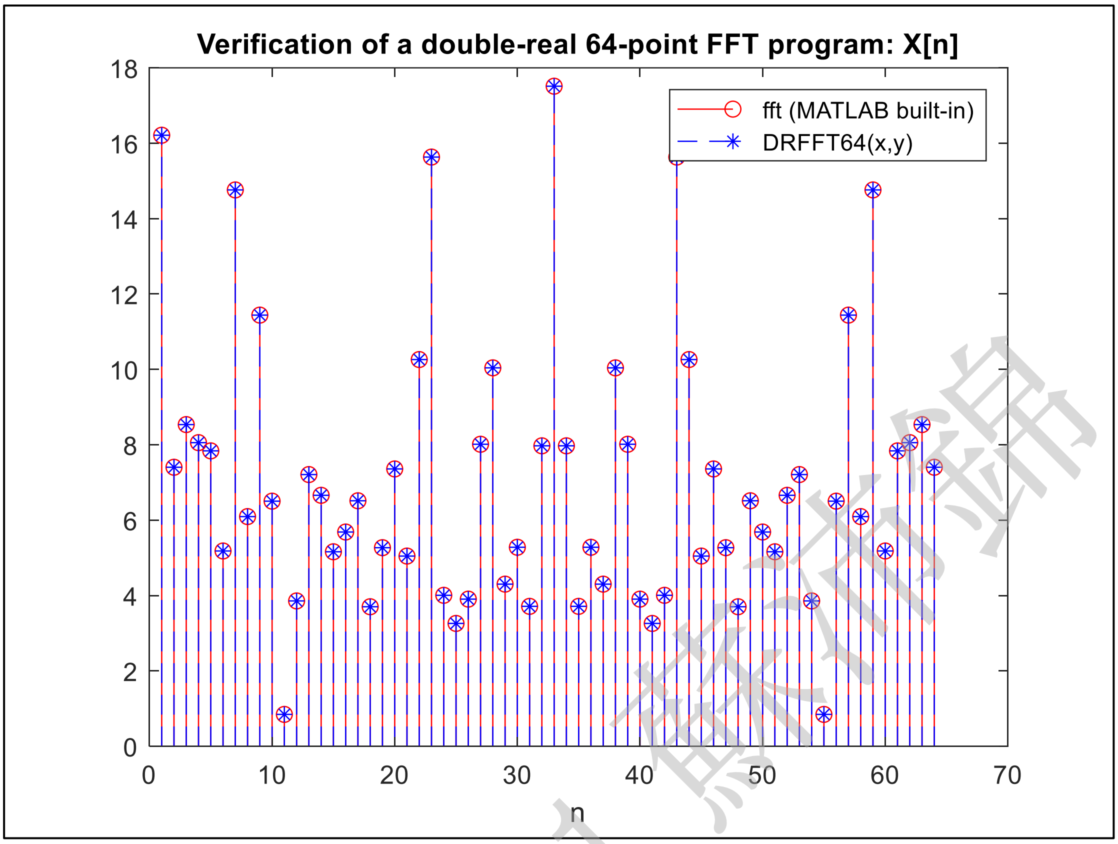
(c) Algorithms in Matlab

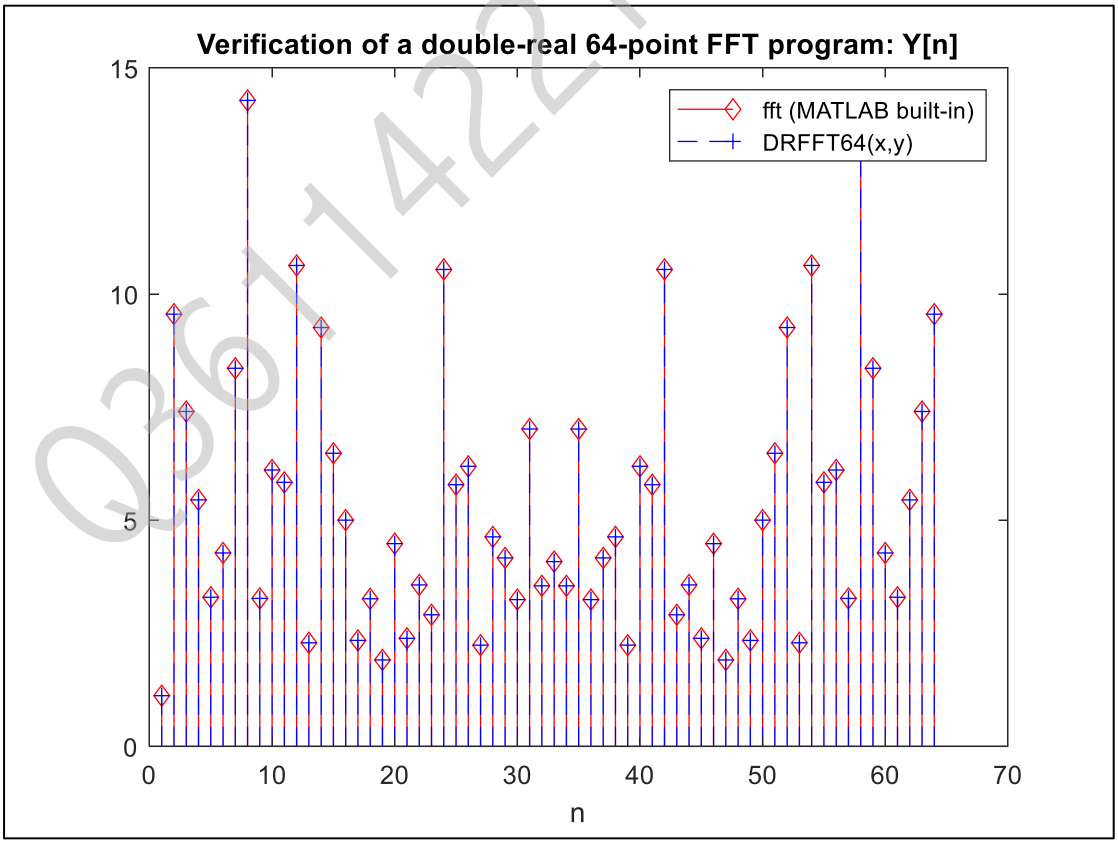
```
function [X,Y] = DRFFT64(x,y)
% DRFFT64 (Double-real 64-point FFT Program)
% DRFFT64(x) is a double-real 64 FFT program,
% where x and y are two real 64-point data vectors.

N = 64;

z = x + y*1i;   % z[n] = x[n] + j*y[n]

Z = FFT64(z);     % Calling FFT64 FFT program once

X = 0.5*(Z+conj(Z(1+mod(0:-1:1-N,N))));     % Symmetry Property: Re{z[n]}

Y = 0.5*(Z-conj(Z(1+mod(0:-1:1-N,N))))/1i;  % Symmetry Property: j*Im{z[n]}

end
```

(d) Complexity analyses

原本 64-point FFT 的複雜度為$2N\log_2 2N$，但是 double-real 64-point FFT 可以一次直接算出 2 個 64-point FFT，所以複雜度為原來的一半。

(e) Verification by programs





(4) Please write an inverse complex 64-point FFT program by only calling **FFT64** FFT program once.

(a) Theoretical derivations

$$X[k]=\sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \qquad x[n]=\frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{+j2\pi kn/N}$$

Duality Property of DFT

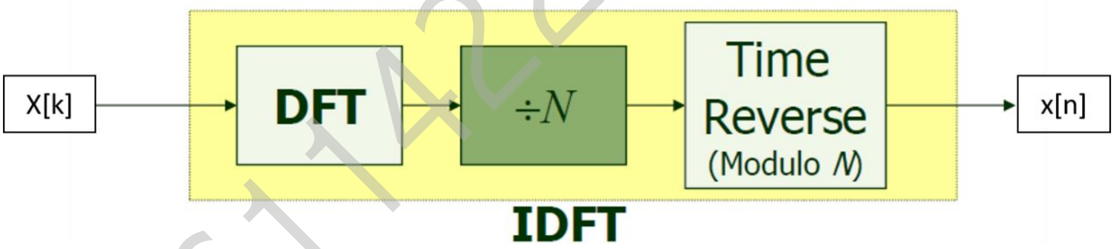
$$x[n]\overset{DFT}{\leftrightarrow} X[k] \qquad X[n]\overset{DFT}{\leftrightarrow} ?$$

Exchange n and k: $x[k]=\frac{1}{N}\sum_{n=0}^{N-1} X[n]e^{+j2\pi kn/N}$

Multiply N: $Nx[k]=\sum_{n=0}^{N-1} X[n]e^{+j2\pi kn/N}$

Change k by $-k$: $Nx[((-k))_N]=\sum_{n=0}^{N-1} X[n]e^{-j2\pi kn/N}$

(b) Flow diagram



(c) Algorithms in Matlab

```
function [x] = IFFT64(X)
% IFFT64 (Inverse complex 64-point FFT Program)
% IFFT64(X) is an inverse complex 64-point FFT program,
% where X is a complex 64-point vectors.

N = 64;

x_tilde = FFT64(X)/N;    % Calling FFT64 FFT program once
```

```
x = x_tilde(1+mod(0:-1:1-N,N));     % Time Reverse (Modulo N)

end
```

(d) Complexity analyses

Inverse 64-point FFT 多了除 N 和時間反轉的步驟，但是除法和反轉都可以透過二進制的 shift 運算完成，因此複雜度和 64-point FFT 一樣。

(e) Verification by programs