Digital Signal Processing - Report

Q36081088 曾紹銘

Program Homework #1

(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 Derivation

$$x[n]=\frac{1}{N}\sum_{k=0}^{N-1}X[k]\,e^{j2\pi kn/N}\quad ,\ N = 32$$

(b) Flow Diagram

$$X[k] \longrightarrow \boxed{FFT_N} \longrightarrow x[n]$$

(c) Algorithms in Matlab

```matlab
function FX = fft32(x)
if length(x) == 32
    FX = fft_dit(x);
elseif length(x) < 32
    x = [x zeros(1, 32 - length(x))];
    FX = fft_dit(x);
end
```

```matlab
function x = fft_dit(x)
N = length(x);
if N > 1
    k   = [0:N-1];
    WNk = exp(-2j*pi/N.*k);
    f   = x(1:2:end);
    g   = x(2:2:end);
    F   = fft_dit(f);
    G   = fft_dit(g);
    x   = [F,F] + (WNk).*[G,G];
end
```

(d) Complexity Analysis

Since we use the N-point FFT program once, it needs N log2 N complex multiplications and additions. The computational complexity is O(nlog2n).

(e) Verification by Programs

```matlab
%(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.
clear;

x  = [3+1i 6-2i 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];

X_my = fft32(x);
X_original = fft(x);
error = X_my - X_original;
error = norm(error.*error);
```

Errors about 2.5716e-25 thanks to the finite precision accuracy of computer.

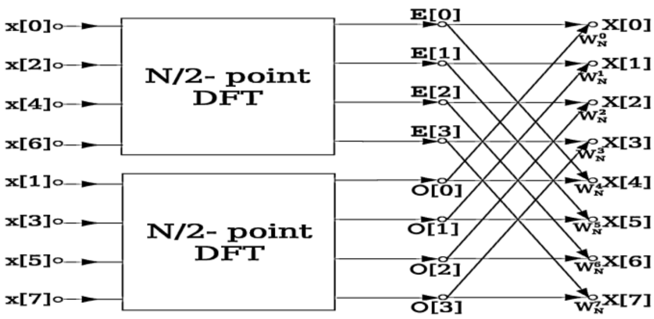
(2) Please use FFT32 to write a complex 64-point FFT program, called FFT64(x), where x is a complex 64-point vector.

(a) Theoretical Derivation

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{nk}=\sum_{r=0}^{N/2-1}x[2r]W_N^{(2r)k}+\sum_{r=0}^{N/2-1}x[2r+1]W_N^{(2r+1)k}$$
$$=\underbrace{\sum_{r=0}^{N/2-1}x[2r]W_{N/2}^{rk}}_{\boxed{N/2-\text{point DFT}}}+W_N^k\underbrace{\sum_{r=0}^{N/2-1}[x[2r+1]W_{N/2}^{rk}}_{\boxed{N/2-\text{point DFT}}}=G[k]+W_N^kH[k]$$

, N=64

(b) Flow Diagram



,N=64

(c) Algorithms in Matlab

```
function FX = fft64(x)
N = 64;
k = [0:N-1];
WNk = exp(-2j*pi/N.*k);
f   = x(1:2:end);
g   = x(2:2:end);
F   = fft32(f);
G   = fft32(g);
FX   = [F,F] + (WNk).*[G,G];
```

(d) Complexity Analysis

Since we use the N-point FFT program once, it needs N log2 N complex multiplications and additions. The computational complexity is O(nlog2n).

(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.
clear;

x  = [3+1i 6-2i 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 3+1i 6-2i 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 96-10i];

X_my = fft64(x);
X_original = fft(x);
error = X_my - X_original;
error = norm(error.*error);
```

Errors about 1.0512e-24 thanks to the finite precision accuracy of computer.

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

(a) Theoretical Derivation

Let the real sequences vector be $x[n]$ and $y[n]$

Define a complex sequence $z[n]$ such that $z[n] = x[n] + jy[n]$, which can be reversed as

$$\Rightarrow \begin{cases} x[n] = \left(z[n] + z^*[n]\right)/2 \\ y[n] = \left(z[n] - z^*[n]\right)/2j \end{cases}$$

, after taking N-point FFT, we have the following relation,

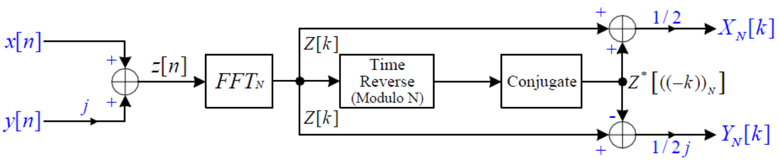
$$\Rightarrow \begin{cases} X_N[k] = \left(Z_N[k] + FFT_N\left\{z^*[n]\right\}\right)/2 \\ Y_N[k] = \left(Z_N[k] - FFT_N\left\{z^*[n]\right\}\right)/2j \end{cases} \quad \ldots\ldots \text{ [A]}$$

By means of the properties of symmetr$z^*[n] \xleftrightarrow{FFT} Z^*[((-k))_N]$, we can rewrite Eq. [A] as

$$\Rightarrow \begin{cases} X_N[k] = \left(Z_N[k] + Z^*\left[((-k))_N\right]\right)/2 \\ Y_N[k] = \left(Z_N[k] - Z^*\left[((-k))_N\right]\right)/2j \end{cases} \quad \ldots\ldots \text{ [B]}$$

Hence a simple pass after the FFT can be used to extract the transforms of $x[n]$ and $y[n]$.

(b) Flow Diagram



(c) Algorithms in Matlab

```
function [Fx,Fy] = drfft64(x,y)
N = 64;
% Form a complex sequence
z=x+1i*y;

F_kz=fft64(z);

%Time Reverse and Mod N
F_nkz(1)=conj(F_kz(1));
F_nkz(2:N)= conj(F_kz(N:-1:2));

%Get results of N-point FFT of x and y
Fx = (F_kz + F_nkz)/2;
Fy = (F_kz - F_nkz)/(2j);
```

(d) Complexity Analysis

Since we use the N-point FFT program only once, it needs $N\log_2 N$ complex multiplications and additions. The implementation of Eq. [B] in (a) also needs $2N$ complex multiplications and complex additions. Hence the computational complexity is $O(\ n\log_2 n\ )$.

(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 vectors, by only calling FFT64
% FFT program 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 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 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];
[X_my, H_my] = drfft64(x,h);
X_original = fft(x);
H_original = fft(h);
error = X_my - X_original;
error = norm(error.*error);
```

Errors about 3.9184e-25 thanks to the finite precision accuracy of computer.

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

(a) Theoretical Derivation

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

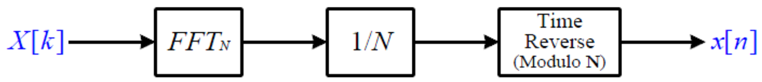
$$\Rightarrow\; x[k] = \frac{1}{N}\sum_{n=0}^{N-1} X[n]\, e^{j2\pi kn/N} \qquad \text{(Exchange index } n \text{ and } k\text{)}$$

$$\Rightarrow\; Nx[k] = \sum_{n=0}^{N-1} X[n]\, e^{j2\pi kn/N} \qquad \text{(Multiply } N\text{)}$$

$$\Rightarrow\; Nx\left[((-k))_N\right] = \sum_{n=0}^{N-1} X[n]\, e^{-j2\pi kn/N} \qquad \text{(Change } k \text{ by } -k\text{)}$$

Hence the inverse FFT can be obtained by first taking FFT of $X[k]$, then divide it by N, then be time reversed, modulo N at last.

(b) Flow Diagram



(c) Algorithms in Matlab

```matlab
function x = ifft64(X)
N = 64;
FX=fft_dit(X)/N;
% Faster implementation , instead of 'mod' function,of
% Time inverse and modula by N
x(1) = FX(1);
x(2:N) = FX(N:-1:2);

%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

Since we use the N-point FFT program once, it needs $N \log_2 N$ complex multiplications and additions. The computational complexity is $O(n\log_2 n)$.

(e) Verification by Programs

```
%(4) Please write an inverse complex 64-point FFT program by only calling
%FFT64 FFT program 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 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];

X_my = fft64(x);
X_original = fft(x);

x_my = ifft64(X_my);
x_original = ifft(X_original);

error = x_my - x_original;
error = norm(error.*error);
```

Errors about 3.9184e-25 thanks to the finite precision accuracy of computer.