Digital Signal Processing

Homework 2. Exercise for FFT & Convolutions

R06942106 Kuan-Chun Chen

1. Objective

In this assignment, I will write functions for "real subroutine FFT", and go through some data to verify my functions are correct.

2. Approach

I implement the following matlab functions for FFT,

[X, Y] = FFTDIT(X, Y, N, M, KIND);

[X, Y] = FFTDIF(X, Y, N, M, KIND);

[X, Y] = FFTDITnbr(X, Y, N, M, KIND);

[X, Y] = FFTDIFnbr(X, Y, N, M, KIND);

X = FFTreorder(X);

where
FFTDIT is fft with the structure of decimation in time;
FFTDIF is fft with the structure of decimation in frequency;
FFTDITnbr is FFTDIT no bit reversal;
FFTDIFnbr is FFTDIF no bit reversal;
FFTreorder is to do decimation(bit reversal) in fft.

Algorithm. FFTDIT

Input：

X, real part of the data; Y, imaginary part of the data;
N, total length of the data; M, $2^M = N$;
KIND, 1 for forward FFT, and -1 for inverse FFT.

Output：

X, real part of the result after fft; Y, imaginary part of the result after fft.

% ---------- for IFFT ----------
if $(KIND == -1)$ ***do***
$Y = Y^*$;
end if
% ----------

% ---------- decimation in time ----------
X = FFTreorder(X);
Y = FFTreorder(Y);
% ----------

$g_n = X_{2n} + iY_{2n}$ $(even - number\ \ samples)$
$h_n = X_{2n+1} + iY_{2n+1}(odd - number\ \ samples)$ $n = 0,1,2,\dots,\frac{N}{2} - 1.$

N-point DFT to $\frac{N}{2}$-point DFT

$$ {}_NF_k = {}_{\frac{N}{2}}G_k + W_N^k\left({}_{\frac{N}{2}}H_k\right); \qquad 0 \leqq k \leqq \frac{N}{2} - 1. $$
$$ {}_NF_k = {}_{\frac{N}{2}}G_{k-\frac{N}{2}} + W_N^k\left({}_{\frac{N}{2}}H_{k-\frac{N}{2}}\right); \qquad \frac{N}{2} \leqq k \leqq N - 1. $$

We already reduce N-point DFT to $\frac{N}{2}$-DFT.
This process can be repeated until we reduce to one-point DFT.

% ---------- for IFFT ----------
if $(KIND == -1)$ ***do***
$X = X/N$;
$Y = Y^*/N$;
end if
% ----------

Algorithm. FFTDIF

Input：

X, real part of the data; Y, imaginary part of the data;

N, total length of the data; M, $2^M = N$;

KIND, 1 for forward FFT, and -1 for inverse FFT.

Output：

X, real part of the result after fft; Y, imaginary part of the result after fft.

% ---------- for IFFT ----------

if (KIND == −1) ***do***

$Y = Y^*$;

end if

% ----------

$g_n = X_{2n} + iY_{2n}$ (even − number samples)

$h_n = X_{2n+1} + iY_{2n+1}$(odd − number samples) $n = 0,1,2,\dots,\frac{N}{2} - 1.$

N-point DFT to $\frac{N}{2}$-point DFT

$$ {}_NF_{2k} = {}_{\frac{N}{2}}G_k + {}_{\frac{N}{2}}H_k; \qquad 0 \leqq k \leqq \frac{N}{2} - 1. $$

$$ {}_NF_{2k+1} = \left({}_{\frac{N}{2}}G_k - {}_{\frac{N}{2}}H_k \right) W_N^k; \qquad 0 \leqq k \leqq \frac{N}{2} - 1. $$

We already reduce N-point DFT to $\frac{N}{2}$-DFT.

This process can be repeated until we reduce to one-point DFT.

% ---------- for IFFT ----------

if (KIND == −1) ***do***

$X = X/N$;

$Y = Y^*/N$;

end if

% ----------

% ---------- decimation in frequency ----------

X = FFTreorder(X);

Y = FFTreorder(Y);

% ----------

Algorithm. FFTreorder

Input：
x, the data.
Output：
x, the data after reorder.

$N = \text{length}(x)$;
$M = \log_2 N$;

if $M == 1$ ***do***
return x;
else
$x_1 = \text{the first half of } x$;
$x_2 = \text{the second half of } x$;
$x_1 = \text{FFTreorder}(x_1)$;
$x_2 = \text{FFTreorder}(x_2)$;
$x = [x_1\ x_2]$;
end if

Note that $W_N^k = -W_N^{k-\frac{N}{2}}$ for $\frac{N}{2} \leqq k \leqq N-1$.

3. Results

3.1. Part 1

x	1+i 1+2i	1-i 4i	2+i 5i	2-i 8i
myFFT	7.00+19.00i 1.00-0.00i	-14.60-2.29i 6.60 - 3.70i	-4.00-2.00i 4.00-4.00i	-5.19-3.94i 13.19 + 5.94i
matlabFFT	7.00+19.00i 1.00-0.00i	-14.60-2.29i 6.60 - 3.70i	-4.00-2.00i 4.00-4.00i	-5.19-3.94i 13.19 + 5.94i
$\hat{x}$	1+i 1+2i	1-i 4i	2+i 5i	2-i 8i

$x \xrightarrow{FFT} X \xrightarrow{IFFT} \hat{x}$. $\hat{x} = x$, so my fft function is correct.

3.2. Part 2

3.2.1. 8-Points Circular Convolution

x_n	1 5	2 6	3 7	4 8
X_k	36.00 -4.00+0.00i	-4.00+9.65i -4.00-1.65i	-4.00+4.00i -4.00-4.00i	-4.00+1.65i -4.00-9.65i

y_n	2	3	5	6
	1	9	8	7
Y_k	41.00+0.00i	-2.53+7.94i	-10.00+1.00i	4.53+1.94i
	-9.00+0.00i	4.53-1.94i	-10.00-1.00i	-2.53-7.94i

$Z_k = X_k * Y_k$	1476	-66.62-56.28i	36-44i	-21.37-0.28i
	36	-21.37+0.28i	36+44i	-66.62+56.28i

z_n	176	193	194	187
	220	189	166	151

First we transform x_n and y_n to X_k and Y_k. Then multiply X_k and Y_k to get Z_k. At last, transform Z_k back to time domain z_n.
This is my method to do circular convolution to x_n and y_n.

Now, let's apply matlab circular convolution function, cconv, to x_n and y_n.

Matlab ans	176	193	194	187
	220	189	166	151

We can see two answers are the same. So the method that multiply in frequency domain can attain the same result.

3.2.2. 16-Points Normal Convolution

Pad zeros after x_n and y_n,

$$x_n = \{1,2,3,4,5,6,7,8,0,0,0,0,0,0,0,0\}$$
$$y_n = \{2,3,5,6,1,9,8,7,0,0,0,0,0,0,0,0\}$$

Apply the method that multiply in frequency domain,

z_n	2	7	11	31
	41	54	65	82
	90	81	68	41
	40	23	14	0

$$x_n = \{1,2,3,4,5,6,7,8\}$$
$$y_n = \{2,3,5,6,1,9,8,7\}$$

Apply matlab normal convolution function, conv, to x_n and y_n,

Matlab ans	2	7	11	31
	41	54	65	82
	90	81	68	41
	40	23	14	0

When we pad zeros with length same as data, and do the circular convolution, the result will same as the normal convolution with data no padding zeros.

3.3. Save Bit-Reversal Process

$$x_n \xrightarrow{FFA+BitReversal} X_k \xrightarrow{BitReversal+FFA} \widehat{x_n}$$

Two continuous bit reversal process can be canceled.
I can save two bit reversal process(no bit reversal at whole process), but need 2 different subroutine for FFT & IFFT.

$$x_n \xrightarrow{FFTDIFnbr} X_k \xrightarrow{FFTDITnbr} \widehat{x_n}$$

Note that "nbr" means no bit reversal.

x_n	1	2	3	4
	5	6	7	8

$\widehat{x_n}$	1.00+0.00i	2.00+0.00i	3.00+0.00i	4.00+0.00i
	5.00+0.00i	6.00+0.00i	7.00+0.00i	8.00+0.00i

3.4. Use FFT Calculate Continuous Fourier Transform

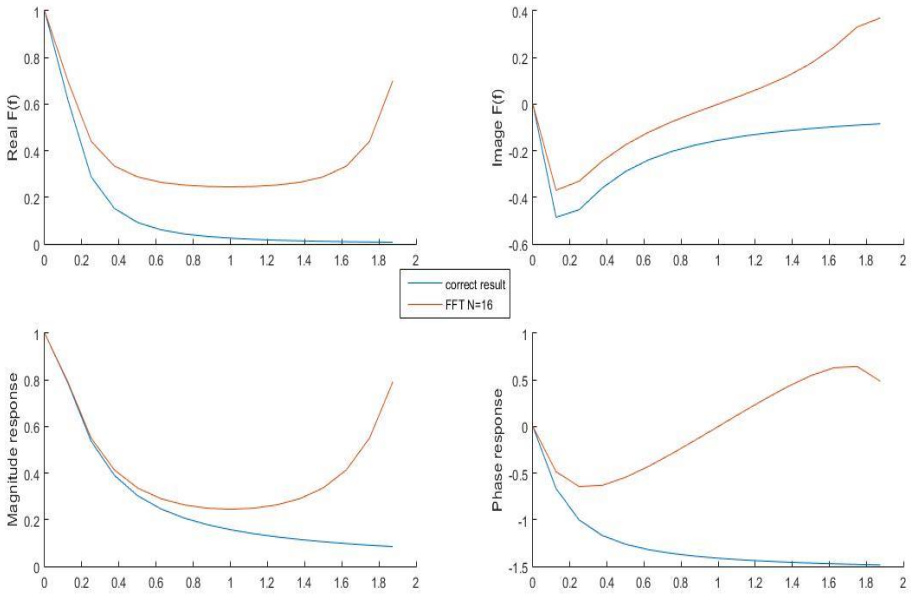
$$f(t) = \begin{cases} e^{-t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

The correct result is

$$F(f) = \frac{1}{1 + j2\pi f}$$

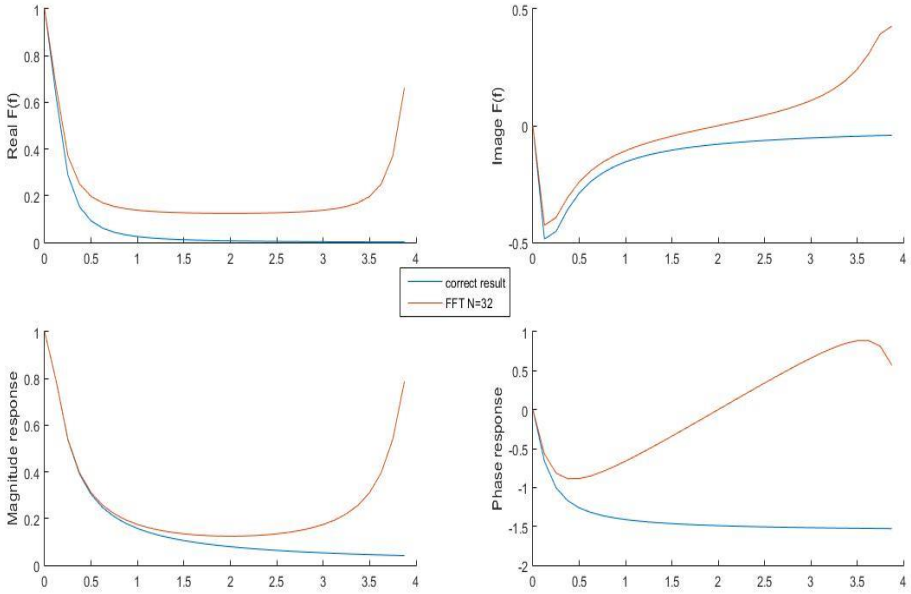
3.4.1.

$T = 8\ \text{sec}, N = 16, T = N \cdot \Delta t.$
$\Delta t = \frac{T}{N}, \Delta f = \frac{1}{T}.$



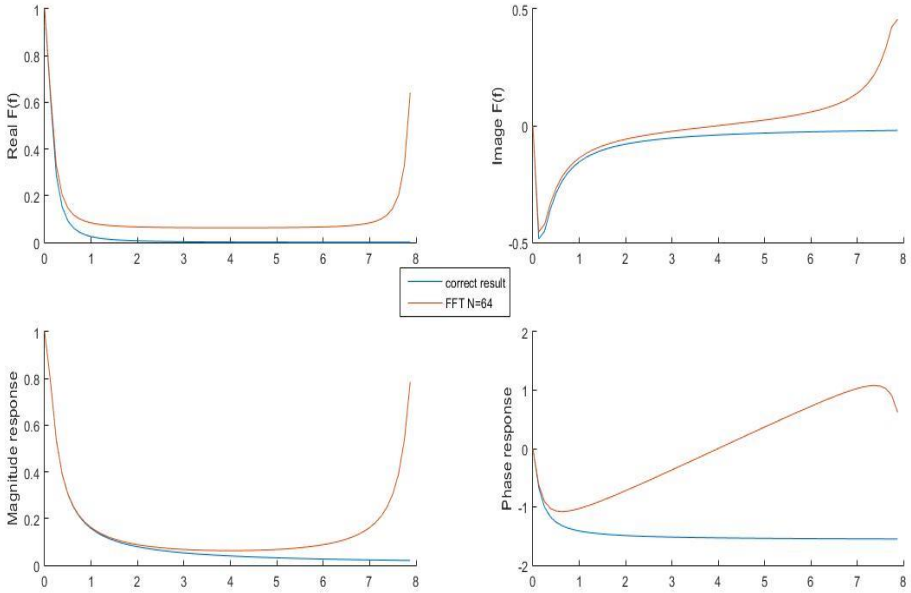
3.4.2.

$\mathrm{T} = 8\ \ \mathrm{sec}, \mathrm{N} = 32, \mathrm{T} = \mathrm{N} \cdot \Delta\mathrm{t}.$
$\Delta\mathrm{t} = \frac{\mathrm{T}}{\mathrm{N}}, \Delta f = \frac{1}{T}.$

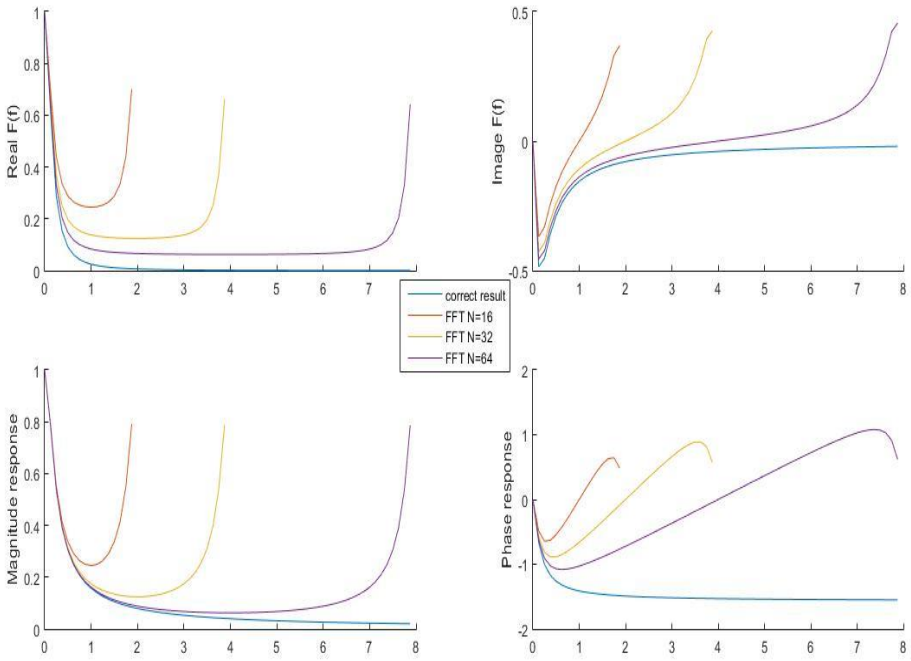


3.4.3.

$\mathrm{T} = 8\ \ \mathrm{sec}, \mathrm{N} = 64, \mathrm{T} = \mathrm{N} \cdot \Delta\mathrm{t}.$
$\Delta\mathrm{t} = \frac{\mathrm{T}}{\mathrm{N}}, \Delta f = \frac{1}{T}.$



3.4.4. Study the Aliasing Error



If we increase number of sample points, the result line will be closer to the correct result. Furthermore, increasing the number of sample points is equal to increasing the sampling frequency, so that it diminishes the aliasing effect.