Fast Fourier Transform

Zhuoqi Cheng
zch@mmmi.sdu.dk
SDU Robotics

Topics to be covered in this course

→ Sampling and reconstruction
→ Aliasing
→ Quantization and dynamic range
→ Implementation
→ Conversion time-frequency domain
→ Z transform
→ Linear Time Invariant system (LTI)
→ System analysis
→ Window functions
→ Filter design
→ Impulse response (FIR and IIR)

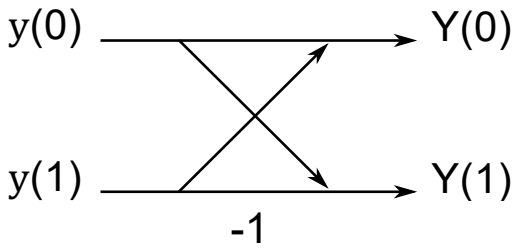
Computation complexity

We want to compute the N-Points DFT of $y(n)$, resulting $Y(m)$, where $m = 0, 1, \ldots, N-1$

$$Y(m) = \sum_{n=0}^{N-1} y(n)\, e^{-j\frac{2\pi}{N}mn}$$

N Points of DFT for the simplest model
N=2,

$$Y(m) = y(0) + y(1)e^{-j\pi m}, \qquad m = 0, 1$$
$$Y(0) = y(0) + y(1)$$
$$Y(1) = y(0) - y(1)e^{-j\pi}$$



Butterfly operation

The complexity for calculating the DFT equals to 2 times Addition

Suppose $N = 2^r$, $W_N = e^{-j\frac{2\pi}{N}}$

$$
\begin{aligned}
Y(m) &= \sum_{n=0}^{N-1} y(n)\, e^{-j\frac{2\pi}{N}mn} \\
&= y(0) + y(1)W_N^m + y(2)W_N^{2m} + y(3)W_N^{3m} + \ \ldots + y(N-1)W_N^{(N-1)m} \\
&= y(0) + y(2)W_N^{2m} + y(4)W_N^{4m} + \ \ldots + \mathrm{y(N-2)}W_N^{(N-2)m} + \\
&\qquad y(1)W_N^m + y(3)W_N^{3m} + y(5)W_N^{5m} + \cdots + y(N-1)W_N^{(N-1)m} \\
&= y(0) + y(2)W_N^{2m} + y(4)W_N^{4m} + \ \ldots + \mathrm{y(N-2)}W_N^{(N-2)m} + \\
&\qquad W_N^m \left(y(1) + y(3)W_N^{2m} + y(5)W_N^{4m} + \ \ldots + y(N-1)W_N^{(N-2)m} \right) \\
&= \left(y(0) + y(2)W_M^m + y(4)W_M^{2m} + \ \ldots + \mathrm{y(N-2)}W_M^{(M-1)m} \right) + \\
&\qquad W_N^m \left(y(1) + y(3)W_M^m + y(5)W_M^{2m} + \ \ldots + y(N-1)W_M^{(M-1)m} \right)
\end{aligned}
$$

$$Y(m) = Y_{even}(m) + W_N^m Y_{odd}(m)$$

Let $M = N/2$

$W_N^2 = e^{-j\frac{2\pi}{N}*2}$
$= e^{-j\frac{2\pi}{N/2}} = W_{N/2}$
$= W_M$

Why FFT is faster?

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

$$Y(m) = Y_{even}(m) + W_N^m Y_{odd}(m)$$ m from 0 to N-1

The complexity for calculating DFT_N equals to 2 times $DFT_{N/2}$ + N times Multiplication + N times Addition

Calculating $DFT_{N/2}$ requires 2 times $DFT_{N/4}$ + $\frac{N}{2}$ times Multiplication + $\frac{N}{2}$ times Addition
Therefore, overall, we need to calculate
$N/2$ times $DFT_{N/(N/2)}$ + $N * (\log_2 N - 1)$ times Multiplication + $N * (\log_2 N - 1)$ times Addition

It is about
$N * (\log_2 N - 1)$ times Multiplication + $N * \log_2 N$ times Addition

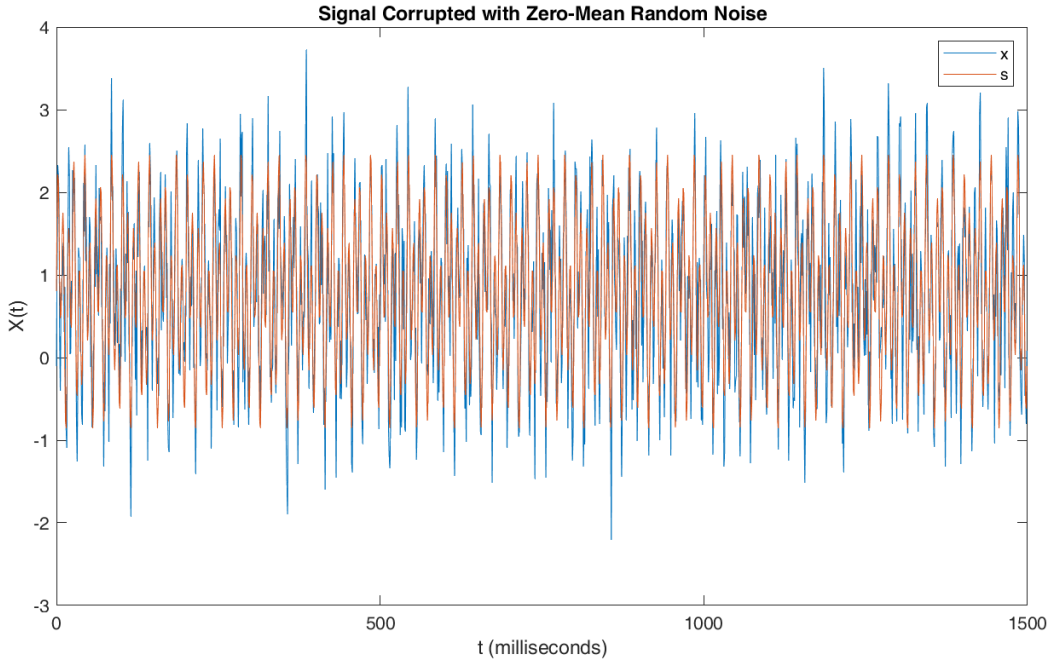
Calculate a DFT by direct use of the formulas, it requires N^2 times computation
When using the FFT algorithm, only $N * \log_2(N)$ times computation.

SDU

Matlab

Function:

- Fast Fourier Transform – fft()
- Shift zero-frequency to center – fftshift()



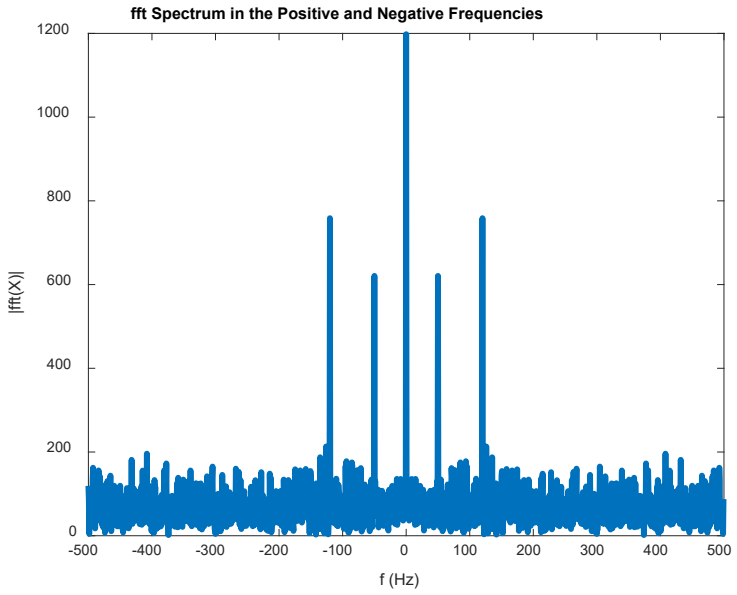

```
Fs = 1000;            % Sampling frequency
T = 1/Fs;             % Sampling period
N = 1500;             % Length of signal
t = (0:N-1)*T;        % Time vector
% construct a signal with a DC offset of amplitude 0.8, a 50 Hz sinusoid of amplitude 0.7,
and a 120 Hz sinusoid of amplitude 1.
S = 0.8 + 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
% add random noise to the signal
X = S + 0.5*randn(size(t));

%plot to see the signal
figure();
plot(1000*t,X); hold on; plot(1000*t,S);
xlabel("t (milliseconds)")
ylabel("X(t)")

%% FFT
Y = fft(X);
f = Fs/N*(0:N-1);
% plot the complex magnitude using abs()
figure();
plot(f,abs(Y)/N,"LineWidth",3)    % remember to divide 'N'
xlabel("f (Hz)")
ylabel("|fft(X)|")

%%
YY = fftshift(Y);
ff = Fs/N*(-N/2:N/2-1);
figure();
plot(ff,abs(YY)/N,"LineWidth",3)
xlabel("f (Hz)")
ylabel("|fft(X)|")
```

SDU

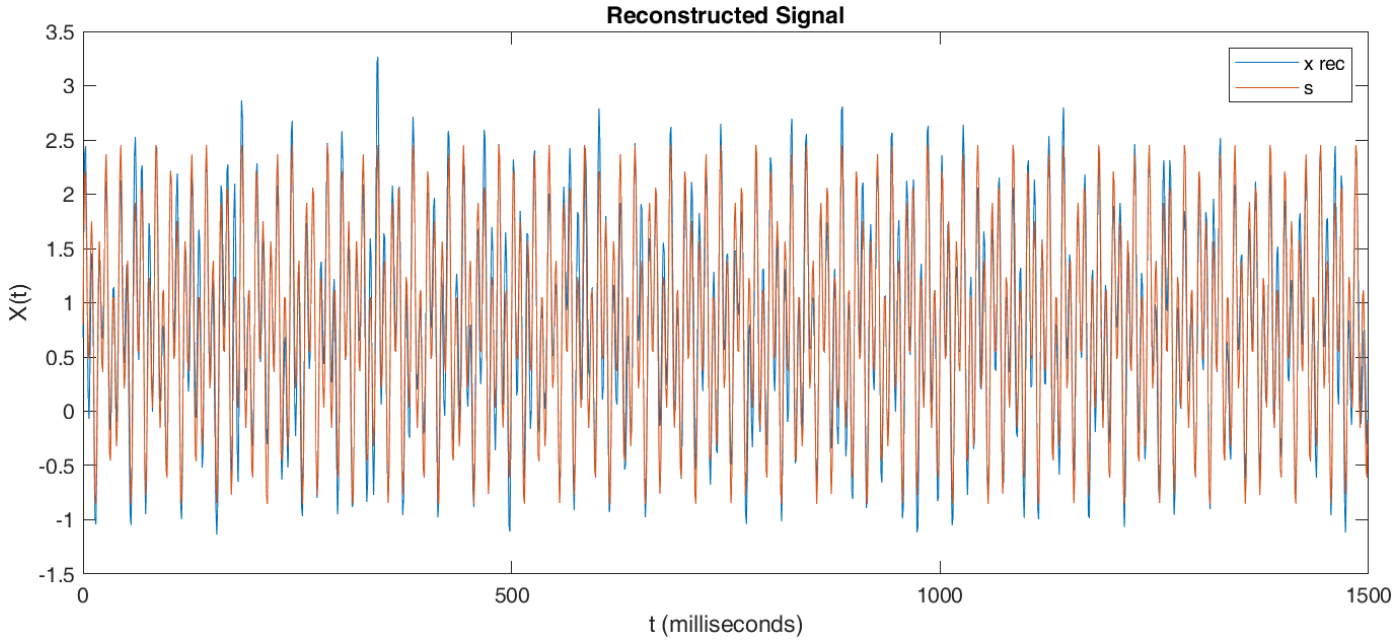
Matlab

Function:

- Inverse zero-frequency shift – iffshift()
- Inverse Fast Fourier Transform – ifft()




```
% assume that I know frequency components above 150Hz are noise.
% remove noise by setting that range to 0
YY(ff>150)=0;
YY(ff<-150)=0;
plot(ff,abs(YY),"LineWidth",3)

%%
Y_rec = ifftshift(YY);
X_rec = ifft(Y_rec);
figure();
plot(1000*t,X_rec)
hold on
plot(1000*t,S)
title("Reconstructed Signal")
xlabel("t (milliseconds)")
ylabel("X(t)")
```



SDU

Exercise