

情報工学科 講義 ディジタル信号処理A (2025/05/15)

第6回 高速フーリエ変換と短時間フーリエ変換

情報工学科 准教授 高道 慎之介

ディジタル信号処理Aの授業予定 (仮)

第XX回	日付	内容 (順次変わっていくので予想)	応用数学の復習
第01回	2025/04/10	イントロダクション, デジタル信号処理	
第02回	2025/04/17	フーリエ級数展開・フーリエ変換から離散フーリエ変換へ	
第03回	2025/04/24	ラプラス変換から z 変換へ	
第04回	2025/05/01	インパルス応答と伝達関数, 安定性	
第05回	2025/05/08	デジタルフィルタ	昨年のBの途中まで
第06回	2025/05/15	高速フーリエ変換と短時間フーリエ変換	
第07回	2025/05/22	総合演習. 期末試験の練習としての立ち位置.	
期末試験	2025/06/??	(日程は後日アナウンス)	

前回の課題の解答

以下の課題についてレポートを作成し, LMS 上で提出せよ。

演習 (残りは課題)

$$H(z) = 1 + 2z^{-1} + 4z^{-2} + 2z^{-3} + z^{-4}$$

- $z = \exp(j\omega T_s)$ に置き換える
- 実部と虚部を求める
- 絶対値と偏角を求める

バターワースフィルタ (一旦, 連続時間のフィルタを考える)

- 次式のパワースペクトル (振幅スペクトルの二乗) を持つフィルタ

$$|H(\omega)|^2 = \frac{1}{1 + (\omega/\omega_c)^{2N}}$$

ω_c, N : カットオフ周波数, フィルタ次数 (係数数)

- [演習] カットオフ周波数と次数を変えたら $|H(\omega)|^2$ はどう変わる?
 - プログラムで図示して考察せよ.



ざっくり解説

本日の内容

今日の話



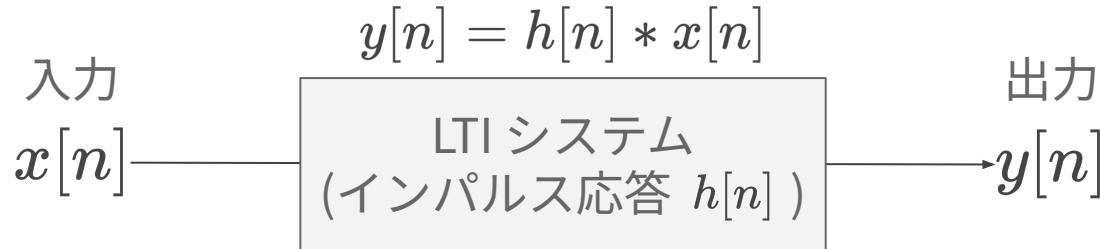
本日の内容

- フィルタの続き
- 高速フーリエ変換
- 短時間フーリエ変換

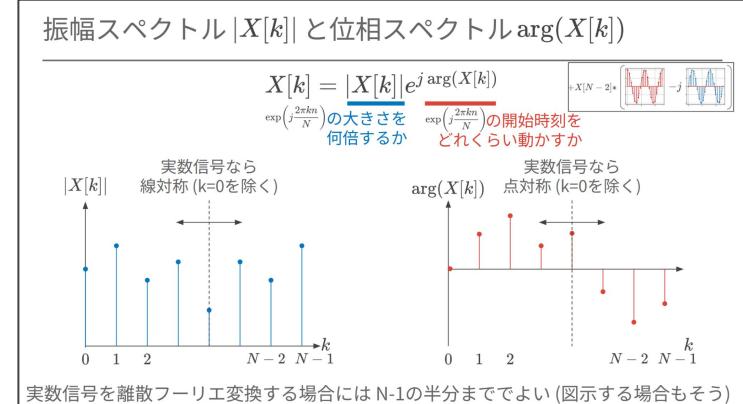
離散フーリエ変換の拡張を理解しよう！

復習：フィルタ

周波数特性を求める



- このシステムの周波数特性(周波数応答)を求めてみよう。具体的には
 - 振幅特性(振幅スペクトル)
 - 位相特性(位相スペクトル)
- を求める。これにより、(実)周波数に対応するシステムの応答を調べられる



例題①：以下の伝達関数の振幅特性と位相特性を求めよ

$$H(z) = 1 + 0.5z^{-1}$$

- $z = \exp(j\omega T_s)$ に置き換える

$$H(\omega) = 1 + 0.5e^{-j\omega T_s}$$

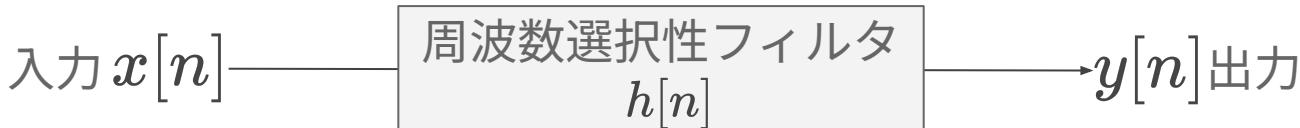
- 実部と虚部を求める

$$\mathcal{R}e[H(\omega)] = 1 + 0.5 \cos(\omega T_s), \quad \mathcal{I}m[H(\omega)] = -0.5 \sin(\omega T_s)$$

- 絶対値と偏角を求める

$$|H(\omega)| = \sqrt{(1.25 + \cos \omega)} \quad \arg H(\omega) = -\tan^{-1} \frac{0.5 \sin \omega}{1 + 0.5 \cos \omega}$$

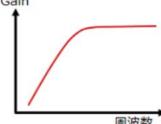
どの周波数を強めるか／弱めるかを決めるフィルタを
周波数選択性フィルタと呼ぶ。



周波数選択性フィルタは 5 種類に大別される。Gain は $|H(\omega)|$ のこと



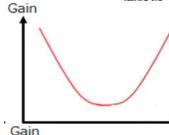
低域通過フィルタ (low pass filter, LPF)：低周波数を通過させ、高周波数を遮断するフィルタ。高周波数を遮断することを強調してハイカットフィルタとも言う



高域通過フィルタ (high pass filter, HPF)：高周波数を通過させ、低周波数を遮断するフィルタ。低周波数を遮断することを強調してローカットフィルタとも言う



帯域通過フィルタ (band pass filter, BPF)：中間帯域を通過させるフィルタ



帯域遮断フィルタ (band elimination filter, BEF)：中間帯域を遮断するフィルタ



全域通過フィルタ (all pass filter, APF)：全帯域を同一ゲインで通過させるフィルタ
[Q. これは何に使うのだろうか？]

フィルタの形状の種類

FIR (finite impulse response) filter

差分方程式

入力にのみ依存

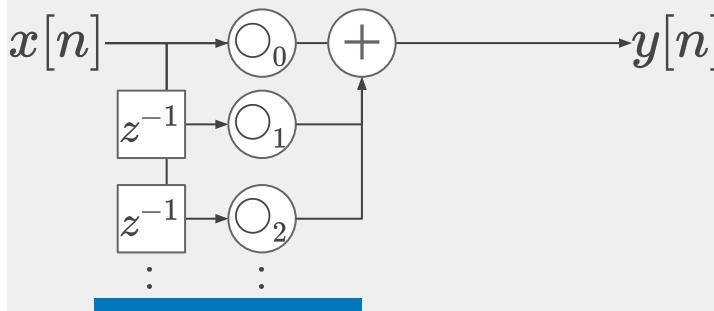
$$y[n] = \sum_m \bigcirc_m x[n - m]$$

伝達関数

分子のみ

$$H(z) = \sum_m \bigcirc_m z^{-m}$$

ブロック図



フィードフォワードのみ

IIR (infinite impulse response) filter

差分方程式

過去の出力にも依存

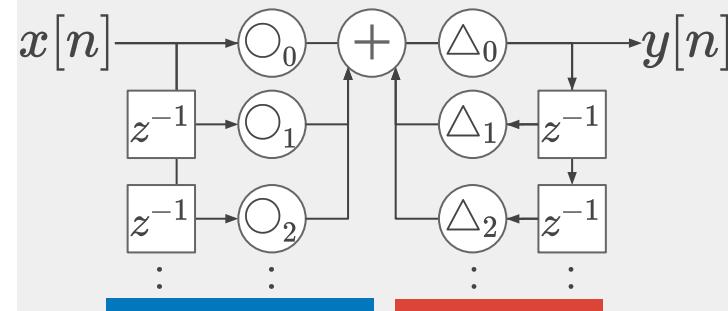
$$y[n] = \sum_m \bigcirc_m x[n - m] + \sum_l \triangle_l y[n - l]$$

伝達関数

$$H(z) = \frac{\sum_m \bigcirc_m z^{-m}}{\sum_l \triangle_l z^{-l}}$$

ブロック図

分母もある



フィードバック構造あり

フィルタ設計には種類がある

- 周波数選択フィルタの満たすべき条件を全て満たすことは難しい
 - → いくつかの条件について良好なフィルタを設計しよう
- バターワースフィルタ (Butterworth filter)
 - 過渡域の落ち方は緩やかだが、通過帯域が可能な限り平坦。Stephen Butterworth による作。食べるバターではない。
- チェビシェフフィルタ (Chebyshev filter)
 - 通過帯域にリップルがあるが、過渡域が急さに落ちる。Pafnuty Lvovich Chebyshev による作。
- 他にも色々

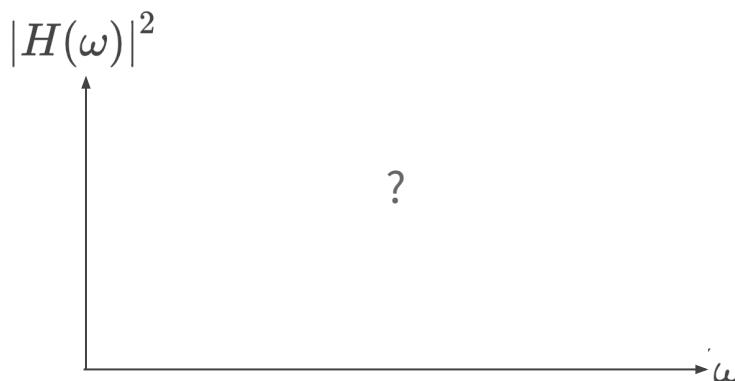
バターワースフィルタ (一旦、連續時間のフィルタを考える)

- 次式のパワースペクトル(振幅スペクトルの二乗)を持つフィルタ

$$|H(\omega)|^2 = \frac{1}{1 + (\omega/\omega_c)^{2N}}$$

ω_c, N : カットオフ周波数, フィルタ次数(係数数)

- [演習] カットオフ周波数と次数を変えたら $|H(\omega)|^2$ はどう変わる?
 - プログラムで図示して考察せよ.



ラプラス変換を利用して極を求め伝達関数を導出しよう

$$\begin{aligned}|H(\omega)|^2 &= \frac{1}{1 + (\omega/\omega_c)^{2N}} \\|H(s)|^2 &= \frac{1}{1 + (-j)^{2N}(s/\omega_c)^{2N}} \\&= \frac{1}{1 + (-1)^N(s/\omega_c)^{2N}}\end{aligned}$$

$|H(s)|^2 = \infty$ となる s が極. k 番目の極 p_k は

$$p_k = \omega_c e^{j(\pi+2\pi k)/(2N)} \quad (N \text{ is even}), \quad \omega_c e^{j2\pi k/N} \quad (N \text{ is odd})$$

そのうち, 安定な極は s 平面の左側にある. 安定な極だけを使ってフィルタを形成する. $N=2$ の場合は

$$H(s) = \frac{1}{(s - \omega_c e^{j3\pi/4})(s - \omega_c e^{j5\pi/4})} = \frac{1}{s^2 + \sqrt{2}\omega_c s + \omega_c^2}$$

アナログフィルタからデジタルフィルタを作る
(converting analogue filter to digital filter)

アナログフィルタからデジタルフィルタを作る

- 例えばバターワースフィルタのように、所望の特性を持つアナログフィルタ(連続時間信号のためのフィルタ)から、同じ特性を持つデジタルフィルタを形成しよう。
- インパルス不変法 (**impulse invariant method**)
 - アナログフィルタのインパルス応答を周期 T_s でサンプリングする
 - 長所：**実装が簡単**、元のフィルタが安定なら変換後も安定
 - 短所：**サンプリング定理を満たさない場合がある**
- 双一次変換 (**Bilinear transform**)
 - ラプラス変換と z 変換の変換式を 線形で近似
 - 長所：**サンプリング定理を必ず満たす**。元フィルタが安定なら変換後も安定
 - 短所：**周波数対応が線形でない**

インパルス不变法：インパルス信号を入力したときの出力が変わらないよう(不变になるよう)に設計する方法

- アナログフィルタのインパルス応答を周期 T_s でサンプリングすればよい。
- 例：あるアナログフィルタの伝達関数を考える

$$H_A(s) = \frac{a_1}{s - s_1} + \frac{a_2}{s - s_2} + \dots + \frac{a_N}{s - s_N}$$

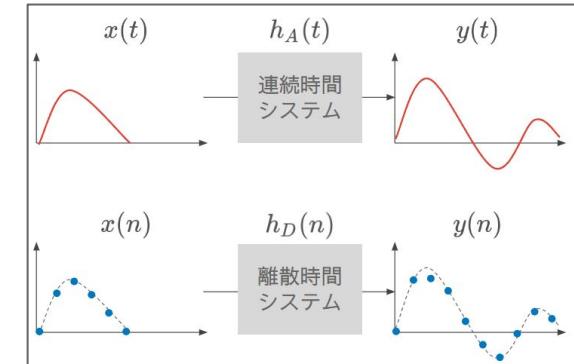
逆ラプラス変換

$$h_A(t) = a_1 e^{s_1 t} + a_2 e^{s_2 t} + \dots + a_N e^{s_N t}$$

$$h_D[n] = h_A(nT_s) = a_1 e^{s_1 nT_s} + a_2 e^{s_2 nT_s} + \dots + a_N e^{s_N nT_s}$$

$$H_D(z) = \frac{a_1}{1 - e^{s_1 T_s} z^{-1}} + \frac{a_2}{1 - e^{s_2 T_s} z^{-1}} + \dots + \frac{a_N}{1 - e^{s_N T_s} z^{-1}}$$

極. $s = \sigma + j\omega$ としたら $|e^{sT_s}| = |e^{\sigma T_s}| |e^{j\omega T_s}| = |e^{\sigma T_s}|$



サンプリング定理を満たす必要がある
→ LPF, BPF しか変換できない

インパルス不变法でサンプリング

z 変換 (各項が等比級数)

元フィルタが安定 (左半面)
→ 変換後も安定 (単位円内)

アナログフィルタの伝達関数が別の形でも、上記の形に持ってこれれば同様に変換

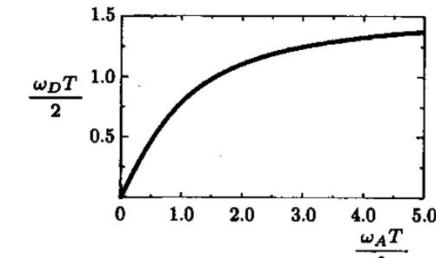
双一次変換：ラプラス変換→z変換の変換式を (一次式) / (一次式) で近似して変換

ラプラス変換→z変換の変換式を近似。具体的は分母と分子をテイラー展開。

$$z = e^{sT_s} = \frac{e^{sT_s/2}}{e^{-sT_s/2}} \simeq \frac{1 + sT_s/2}{1 - sT_s/2} \xrightarrow{s = \frac{2}{T_s} \frac{1 - z^{-1}}{1 + z^{-1}}} \text{双一次変換の式}$$

変換式の意味を考えよう。{連続, 離散}時間信号の角周波数を ω_A, ω_D とし、
{ラプラス, z}変換をそれぞれ対応するフーリエ変換に置き換えると

$$j\omega_A = \frac{2}{T} \frac{1 - e^{-j\omega_D T}}{1 + e^{j\omega_D T}} \xrightarrow{\omega_A = \frac{2}{T} \tan(\omega_D T/2)}$$



連続時間信号の角周波数は、離散時間信号のそれに非線形に対応（インパルス不变法は線形に対応）。全ての角周波数は一定範囲に変換され、サンプリング定理を自動で満たす。

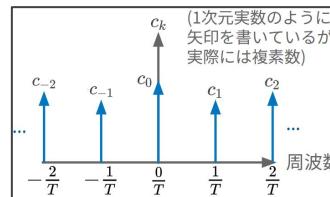
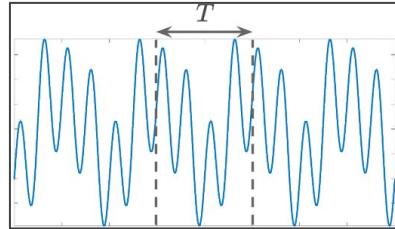
復習：離散フーリエ変換

手法の比較

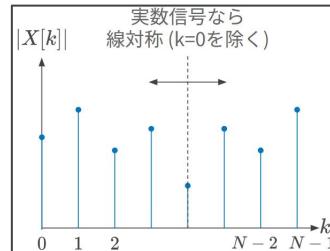
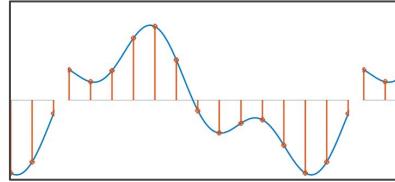
連続時間信号

フーリエ級数展開

周期連続時間信号 → 離散周波数



離散時間信号



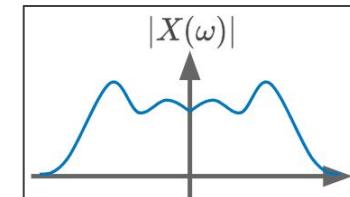
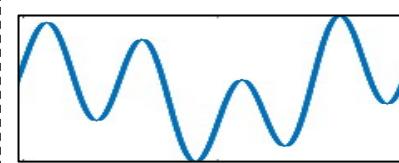
離散フーリエ変換

周期離散時間信号 → 周期離散周波数

離散周波数 (信号の周期性を仮定)

フーリエ変換

非周期連続時間信号 → 連続周波数



本講義ではやりません。
興味があれば調べてください。

離散時間フーリエ変換

非周期離散信号 → 連続周波数

連続周波数 (信号は非周期で良い)

離散フーリエ変換とその逆変換

- 離散 Fourier 変換：離散時間信号 → 離散周波数

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp\left(-j\frac{2\pi kn}{N}\right)$$

$$(k \in \{0, 1, \dots, N-1\})$$

DFT $[x[n]] = X[k]$

- 逆離散 Fourier 変換：離散周波数 → 離散時間信号

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \exp\left(j\frac{2\pi kn}{N}\right)$$

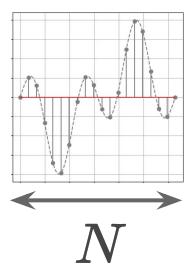
$$(n \in \{0, 1, \dots, N-1\})$$

DFT⁻¹ $[X[k]] = x[n]$

その意味

複素数

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \exp\left(j \frac{2\pi kn}{N}\right)$$



$$\begin{aligned} &= \dots + X[N-2]* \left(\begin{array}{c} \text{Red plot} \\ -j \end{array} \right) + X[N-1]* \left(\begin{array}{c} \text{Red plot} \\ -j \end{array} \right) \\ &\quad + X[0]* \left(\begin{array}{c} \text{Red plot} \\ 1 \end{array} \right) \\ &\quad \dots + X[2]* \left(\begin{array}{c} \text{Red plot} \\ +j \end{array} \right) + X[1]* \left(\begin{array}{c} \text{Red plot} \\ +j \end{array} \right) \end{aligned}$$

The diagram illustrates the inverse Fourier transform process. It shows how the original signal is reconstructed by summing the scaled products of the DFT coefficients ($X[k]$) and their corresponding complex exponential basis functions. The basis functions are shown as red plots, and the scaling factor is indicated by the value of k (e.g., $-j$, 1 , $+j$). The final result is a signal over N samples, represented by a red line and a blue dashed line.

$\exp(j2\pi(N-1)\frac{n}{N}) = \exp(-j2\pi\frac{n}{N}) = \cos(2\pi\frac{n}{N}) - j \sin(2\pi\frac{n}{N})$

$\exp(j2\pi\frac{n}{N}) = \cos(2\pi\frac{n}{N}) + j \sin(2\pi\frac{n}{N})$

DFT の行列表現

$$\mathbf{X} = \mathbf{W}\mathbf{x}$$

$$\begin{bmatrix} X[0] \\ \vdots \\ X[k] \\ \vdots \\ X[N-1] \end{bmatrix} = \text{N-by-N matrix W} \begin{bmatrix} x[0] \\ \vdots \\ x[n] \\ \vdots \\ x[N-1] \end{bmatrix}$$

```
import numpy as np

def DFT_matrix(N):
    W = np.zeros((N,N), dtype=complex) # N-by-N zero matrix
    """
    Generate DFT matrix of size N.
    """
    return W

xn = np.array([1, 2, 3, 4, 5, 6, 7, 8]) # input signal
W = DFT_matrix(len(xn))

# check
print("your answer:\n", W @ xn) # Xk = W xn
print("numpy:\n", np.fft.fft(xn)) # reference. OK if your answer matches to this.
```

高速フーリエ変換 (fast Fourier transform: FFT)

行列計算は $O(N^2)$ の計算量 → とても重い

$$X = Wx$$
$$\begin{bmatrix} X[0] \\ \vdots \\ X[k] \\ \vdots \\ X[N-1] \end{bmatrix} = \text{N-by-N matrix } W \begin{bmatrix} x[0] \\ \vdots \\ x[n] \\ \vdots \\ x[N-1] \end{bmatrix}$$

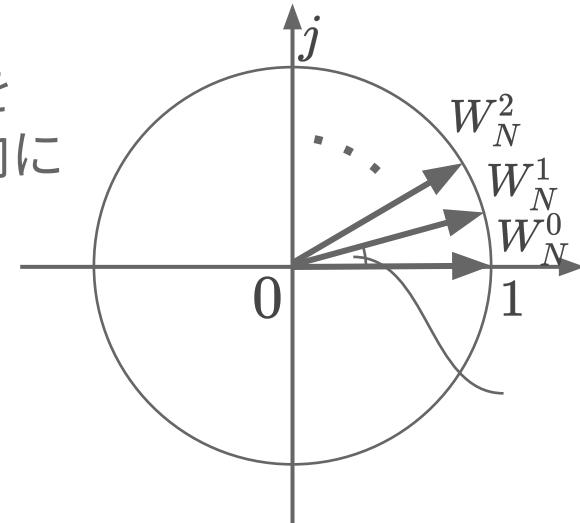
- N^2 のオーダーで計算しなければならない
 - 例えば、3分の音声は $180 \text{ [sec]} * 48000 \text{ [Hz]} = 8640000 \text{ [sample]}$ なので,**7.4e13 回の乗算が必要.**
- **計算量を抑えるアルゴリズムが高速フーリエ変換**
 - 厳密解が求まる。DFT と FFT の結果は同じ。計算量が違う。

表記を簡単にするために回転子を定義

$$\mathbf{X} = \mathbf{W}\mathbf{x}$$

$W_N = \exp(j\frac{2\pi}{N})$: 回転子. 単位円を順回転方向に N 分割したもの

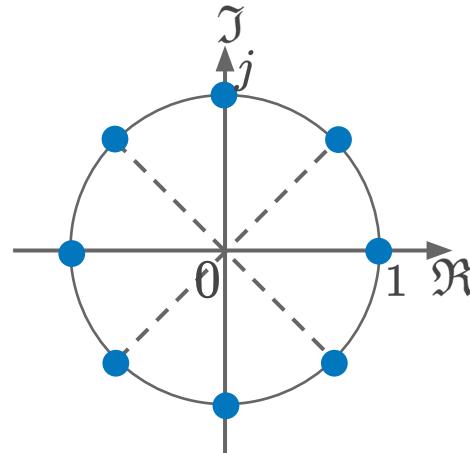
$(\mathbf{W}_N)_{nk}$ = $W_N^{-kn} = \exp\left(-j\frac{2\pi kn}{N}\right)$: N 分割したものを nk だけ逆回転方向に進んだもの.
行列 \mathbf{W}_N の
(n, k)番目の要素



計算量を抑えるためのヒント：対称性を利用する

1回転しても同じ

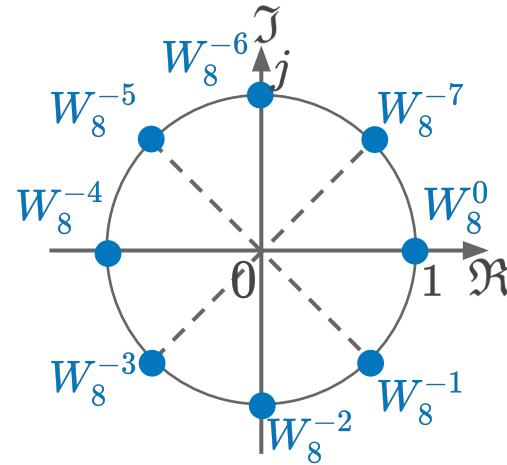
$$W_N^n = W_N^{n+N}$$



$W_N^n x$ の結果を知っていれば、
 $W_N^{n+N} x$ を計算する必要はない
(1倍すればよい)

単位円の反対側 $\rightarrow -1$ を乗算

$$W_N^n = -W_N^{n+N/2}$$



$W_N^n x$ の結果を知っていれば、
 $W_N^{n+N/2} x$ の結果はそれを -1 倍すればよい

DFT から FFT へ： DFT の行列表現

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0\cdot0} & W_8^{-1\cdot0} & W_8^{-2\cdot0} & W_8^{-3\cdot0} & W_8^{-4\cdot0} & W_8^{-5\cdot0} & W_8^{-6\cdot0} & W_8^{-7\cdot0} \\ W_8^{-0\cdot1} & W_8^{-1\cdot1} & W_8^{-2\cdot1} & W_8^{-3\cdot1} & W_8^{-4\cdot1} & W_8^{-5\cdot1} & W_8^{-6\cdot1} & W_8^{-7\cdot1} \\ W_8^{-0\cdot2} & W_8^{-1\cdot2} & W_8^{-2\cdot2} & W_8^{-3\cdot2} & W_8^{-4\cdot2} & W_8^{-5\cdot2} & W_8^{-6\cdot2} & W_8^{-7\cdot2} \\ W_8^{-0\cdot3} & W_8^{-1\cdot3} & W_8^{-2\cdot3} & W_8^{-3\cdot3} & W_8^{-4\cdot3} & W_8^{-5\cdot3} & W_8^{-6\cdot3} & W_8^{-7\cdot3} \\ W_8^{-0\cdot4} & W_8^{-1\cdot4} & W_8^{-2\cdot4} & W_8^{-3\cdot4} & W_8^{-4\cdot4} & W_8^{-5\cdot4} & W_8^{-6\cdot4} & W_8^{-7\cdot4} \\ W_8^{-0\cdot5} & W_8^{-1\cdot5} & W_8^{-2\cdot5} & W_8^{-3\cdot5} & W_8^{-4\cdot5} & W_8^{-5\cdot5} & W_8^{-6\cdot5} & W_8^{-7\cdot5} \\ W_8^{-0\cdot6} & W_8^{-1\cdot6} & W_8^{-2\cdot6} & W_8^{-3\cdot6} & W_8^{-4\cdot6} & W_8^{-5\cdot6} & W_8^{-6\cdot6} & W_8^{-7\cdot6} \\ W_8^{-0\cdot7} & W_8^{-1\cdot7} & W_8^{-2\cdot7} & W_8^{-3\cdot7} & W_8^{-4\cdot7} & W_8^{-5\cdot7} & W_8^{-6\cdot7} & W_8^{-7\cdot7} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

DFT 行列 ($N=8$ とする)

DFT から FFT へ： 対称性の適用

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0\cdot0} & W_8^{-1\cdot0} & W_8^{-2\cdot0} & W_8^{-3\cdot0} & W_8^{-4\cdot0} & W_8^{-5\cdot0} & W_8^{-6\cdot0} & W_8^{-7\cdot0} \\ W_8^{-0\cdot1} & W_8^{-1\cdot1} & W_8^{-2\cdot1} & W_8^{-3\cdot1} & W_8^{-4\cdot1} & W_8^{-5\cdot1} & W_8^{-6\cdot1} & W_8^{-7\cdot1} \\ W_8^{-0\cdot2} & W_8^{-1\cdot2} & W_8^{-2\cdot2} & W_8^{-3\cdot2} & W_8^{-4\cdot2} & W_8^{-5\cdot2} & W_8^{-6\cdot2} & W_8^{-7\cdot2} \\ W_8^{-0\cdot3} & W_8^{-1\cdot3} & W_8^{-2\cdot3} & W_8^{-3\cdot3} & W_8^{-4\cdot3} & W_8^{-5\cdot3} & W_8^{-6\cdot3} & W_8^{-7\cdot3} \\ W_8^{-0\cdot4} & W_8^{-1\cdot4} & W_8^{-2\cdot4} & W_8^{-3\cdot4} & W_8^{-4\cdot4} & W_8^{-5\cdot4} & W_8^{-6\cdot4} & W_8^{-7\cdot4} \\ W_8^{-0\cdot5} & W_8^{-1\cdot5} & W_8^{-2\cdot5} & W_8^{-3\cdot5} & W_8^{-4\cdot5} & W_8^{-5\cdot5} & W_8^{-6\cdot5} & W_8^{-7\cdot5} \\ W_8^{-0\cdot6} & W_8^{-1\cdot6} & W_8^{-2\cdot6} & W_8^{-3\cdot6} & W_8^{-4\cdot6} & W_8^{-5\cdot6} & W_8^{-6\cdot6} & W_8^{-7\cdot6} \\ W_8^{-0\cdot7} & W_8^{-1\cdot7} & W_8^{-2\cdot7} & W_8^{-3\cdot7} & W_8^{-4\cdot7} & W_8^{-5\cdot7} & W_8^{-6\cdot7} & W_8^{-7\cdot7} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

対称性 ($W_N^n = W_N^{n+N}$) を適用すると…

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^0 & W_8^0 \\ W_8^0 & W_8^{-1} & W_8^{-2} & W_8^{-3} & W_8^{-4} & W_8^{-5} & W_8^{-6} & W_8^{-7} \\ W_8^0 & W_8^{-2} & W_8^{-4} & W_8^{-6} & W_8^0 & W_8^{-2} & W_8^{-4} & W_8^{-6} \\ W_8^0 & W_8^{-3} & W_8^{-6} & W_8^{-1} & W_8^{-4} & W_8^{-7} & W_8^{-2} & W_8^{-5} \\ W_8^0 & W_8^{-4} & W_8^0 & W_8^{-4} & W_8^0 & W_8^{-4} & W_8^0 & W_8^{-4} \\ W_8^0 & W_8^{-5} & W_8^{-2} & W_8^{-7} & W_8^{-4} & W_8^{-1} & W_8^{-6} & W_8^{-3} \\ W_8^0 & W_8^{-6} & W_8^{-4} & W_8^{-2} & W_8^0 & W_8^{-6} & W_8^{-4} & W_8^{-2} \\ W_8^0 & W_8^{-7} & W_8^{-6} & W_8^{-5} & W_8^{-4} & W_8^{-3} & W_8^{-2} & W_8^{-1} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

DFT から FFT へ： 対称性の適用 → 偶数行を見る

偶数行目の要素は 左半分 と 右半分 が同じ (W_8^{-0})

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-1} & W_8^{-2} & W_8^{-3} & W_8^{-4} & W_8^{-5} & W_8^{-6} & W_8^{-7} \\ W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} & W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} \\ W_8^{-0} & W_8^{-3} & W_8^{-6} & W_8^{-1} & W_8^{-4} & W_8^{-7} & W_8^{-2} & W_8^{-5} \\ W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} \\ W_8^{-0} & W_8^{-5} & W_8^{-2} & W_8^{-7} & W_8^{-4} & W_8^{-1} & W_8^{-6} & W_8^{-3} \\ W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} & W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} \\ W_8^{-0} & W_8^{-7} & W_8^{-6} & W_8^{-5} & W_8^{-4} & W_8^{-3} & W_8^{-2} & W_8^{-1} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

DFT から FFT へ： 対称性の適用 → 奇数行を見る

奇数行目の要素は **左半分** と **右半分** が半周期ずれ(W_8^{-4})

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-1} & W_8^{-2} & W_8^{-3} & W_8^{-4} & W_8^{-5} & W_8^{-6} & W_8^{-7} \\ W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} & W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} \\ W_8^{-0} & W_8^{-3} & W_8^{-6} & W_8^{-1} & W_8^{-4} & W_8^{-7} & W_8^{-2} & W_8^{-5} \\ W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} \\ W_8^{-0} & W_8^{-5} & W_8^{-2} & W_8^{-7} & W_8^{-4} & W_8^{-1} & W_8^{-6} & W_8^{-3} \\ W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} & W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} \\ W_8^{-0} & W_8^{-7} & W_8^{-6} & W_8^{-5} & W_8^{-4} & W_8^{-3} & W_8^{-2} & W_8^{-1} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

DFT から FFT へ： 行列を並び替えて分割する

並び替え

$$\begin{bmatrix} X[0] \\ X[2] \\ X[4] \\ X[6] \\ X[1] \\ X[3] \\ X[5] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-0} & W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} \\ W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} \\ W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} \\ W_8^{-0} & W_8^{-1} & W_8^{-2} & W_8^{-3} \\ W_8^{-0} & W_8^{-3} & W_8^{-6} & W_8^{-1} \\ W_8^{-0} & W_8^{-5} & W_8^{-2} & W_8^{-7} \\ W_8^{-0} & W_8^{-7} & W_8^{-6} & W_8^{-5} \end{bmatrix} \begin{bmatrix} W_8^{-0} & W_8^{-0} & W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} \\ W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} \\ W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} \\ W_8^{-4} & W_8^{-5} & W_8^{-6} & W_8^{-7} \\ W_8^{-4} & W_8^{-7} & W_8^{-2} & W_8^{-5} \\ W_8^{-4} & W_8^{-1} & W_8^{-6} & W_8^{-3} \\ W_8^{-4} & W_8^{-3} & W_8^{-2} & W_8^{-1} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

同じ

分割

$$\begin{bmatrix} X[0] \\ X[2] \\ X[4] \\ X[6] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-0} & W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} \\ W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} \\ W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] \\ x[1] + W_8^{-0}x[5] \\ x[2] + W_8^{-0}x[6] \\ x[3] + W_8^{-0}x[7] \end{bmatrix}$$

半周期ずれ

$$\begin{bmatrix} X[1] \\ X[3] \\ X[5] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-1} & W_8^{-2} & W_8^{-3} \\ W_8^{-0} & W_8^{-3} & W_8^{-6} & W_8^{-1} \\ W_8^{-0} & W_8^{-5} & W_8^{-2} & W_8^{-7} \\ W_8^{-0} & W_8^{-7} & W_8^{-6} & W_8^{-5} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-4}x[4] \\ x[1] + W_8^{-4}x[5] \\ x[2] + W_8^{-4}x[6] \\ x[3] + W_8^{-4}x[7] \end{bmatrix}$$

DFT から FFT へ： さらに分割 (前頁と色の対応が違うので注意)

左右の比較

$$\begin{bmatrix} X[0] \\ X[2] \\ X[4] \\ X[6] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-0} & W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} \\ W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} \\ W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] \\ x[1] + W_8^{-0}x[5] \\ x[2] + W_8^{-0}x[6] \\ x[3] + W_8^{-0}x[7] \end{bmatrix}$$

並び替え

$$= \begin{bmatrix} W_8^{-0} & W_8^{-0} & W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-4} & W_8^{-0} & W_8^{-4} \\ W_8^{-0} & W_8^{-2} & W_8^{-4} & W_8^{-6} \\ W_8^{-0} & W_8^{-6} & W_8^{-4} & W_8^{-2} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] \\ x[1] + W_8^{-0}x[5] \\ x[2] + W_8^{-0}x[6] \\ x[3] + W_8^{-0}x[7] \end{bmatrix}$$

分割

$$\begin{bmatrix} X[0] \\ X[4] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-4} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] + W_8^{-0}(x[2] + W_8^{-0}x[6]) \\ x[1] + W_8^{-0}x[5] + W_8^{-0}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$

$$\begin{bmatrix} X[2] \\ X[6] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-2} \\ W_8^{-0} & W_8^{-6} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] + W_8^{-4}(x[2] + W_8^{-0}x[6]) \\ x[1] + W_8^{-0}x[5] + W_8^{-4}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$

左右の比較

$$\begin{bmatrix} X[1] \\ X[3] \\ X[5] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-1} & W_8^{-2} & W_8^{-3} \\ W_8^{-0} & W_8^{-3} & W_8^{-6} & W_8^{-1} \\ W_8^{-0} & W_8^{-5} & W_8^{-2} & W_8^{-7} \\ W_8^{-0} & W_8^{-7} & W_8^{-6} & W_8^{-5} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-4}x[4] \\ x[1] + W_8^{-4}x[5] \\ x[2] + W_8^{-4}x[6] \\ x[3] + W_8^{-4}x[7] \end{bmatrix}$$

並び替え

$$= \begin{bmatrix} W_8^{-0} & W_8^{-1} & W_8^{-2} & W_8^{-3} \\ W_8^{-0} & W_8^{-5} & W_8^{-2} & W_8^{-7} \\ W_8^{-0} & W_8^{-3} & W_8^{-6} & W_8^{-1} \\ W_8^{-0} & W_8^{-7} & W_8^{-6} & W_8^{-5} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-4}x[4] \\ x[1] + W_8^{-4}x[5] \\ x[2] + W_8^{-4}x[6] \\ x[3] + W_8^{-4}x[7] \end{bmatrix}$$

分割

$$\begin{bmatrix} X[1] \\ X[5] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-1} \\ W_8^{-0} & W_8^{-5} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-4}x[4] + W_8^{-2}(x[2] + W_8^{-4}x[6]) \\ x[1] + W_8^{-4}x[5] + W_8^{-2}(x[3] + W_8^{-4}x[7]) \end{bmatrix}$$

$$\begin{bmatrix} X[3] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-3} \\ W_8^{-0} & W_8^{-7} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-4}x[4] + W_8^{-6}(x[2] + W_8^{-4}x[6]) \\ x[1] + W_8^{-4}x[5] + W_8^{-6}(x[3] + W_8^{-4}x[7]) \end{bmatrix} \quad 35$$

DFT から FFT へ： 対称性を適用する

$$W_N^n = -W_N^{n+N/2}$$

$$\begin{bmatrix} X[0] \\ X[4] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-0} \\ W_8^{-0} & W_8^{-4} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] + W_8^{-0}(x[2] + W_8^{-0}x[6]) \\ x[1] + W_8^{-0}x[5] + W_8^{-0}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$

$$\begin{bmatrix} W_8^{-0} & W_8^{-0} \\ W_8^{-0} & -W_8^{-0} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] + W_8^{-0}(x[2] + W_8^{-0}x[6]) \\ x[1] + W_8^{-0}x[5] + W_8^{-0}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$

$$\begin{bmatrix} X[2] \\ X[6] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-2} \\ W_8^{-0} & W_8^{-6} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] + W_8^{-4}(x[2] + W_8^{-0}x[6]) \\ x[1] + W_8^{-0}x[5] + W_8^{-4}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$

$$\begin{bmatrix} W_8^{-0} & W_8^{-2} \\ W_8^{-0} & -W_8^{-2} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] - W_8^{-0}(x[2] + W_8^{-0}x[6]) \\ x[1] + W_8^{-0}x[5] - W_8^{-0}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$

$$\begin{bmatrix} X[1] \\ X[5] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-1} \\ W_8^{-0} & W_8^{-5} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-4}x[4] + W_8^{-2}(x[2] + W_8^{-4}x[6]) \\ x[1] + W_8^{-4}x[5] + W_8^{-2}(x[3] + W_8^{-4}x[7]) \end{bmatrix}$$

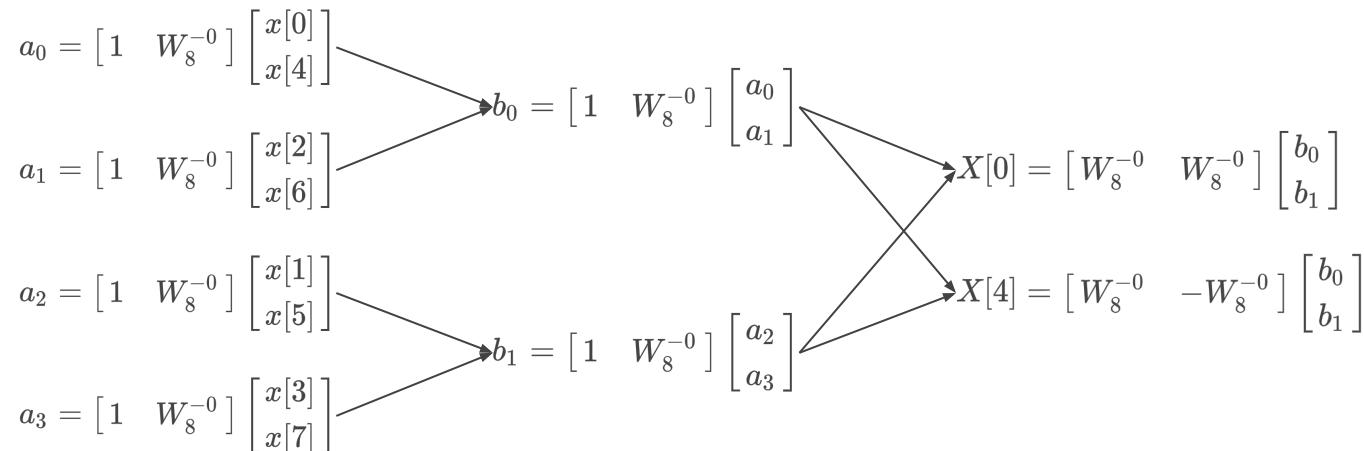
$$\begin{bmatrix} W_8^{-0} & W_8^{-1} \\ W_8^{-0} & -W_8^{-1} \end{bmatrix} \begin{bmatrix} x[0] - W_8^{-0}x[4] + W_8^{-2}(x[2] + W_8^{-0}x[6]) \\ x[1] - W_8^{-0}x[5] + W_8^{-2}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$

$$\begin{bmatrix} X[3] \\ X[7] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-3} \\ W_8^{-0} & W_8^{-7} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-4}x[4] + W_8^{-6}(x[2] + W_8^{-4}x[6]) \\ x[1] + W_8^{-4}x[5] + W_8^{-6}(x[3] + W_8^{-4}x[7]) \end{bmatrix}$$

$$\begin{bmatrix} W_8^{-0} & W_8^{-3} \\ W_8^{-0} & -W_8^{-3} \end{bmatrix} \begin{bmatrix} x[0] - W_8^{-0}x[4] - W_8^{-2}(x[2] + W_8^{-0}x[6]) \\ x[1] - W_8^{-0}x[5] - W_8^{-2}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$

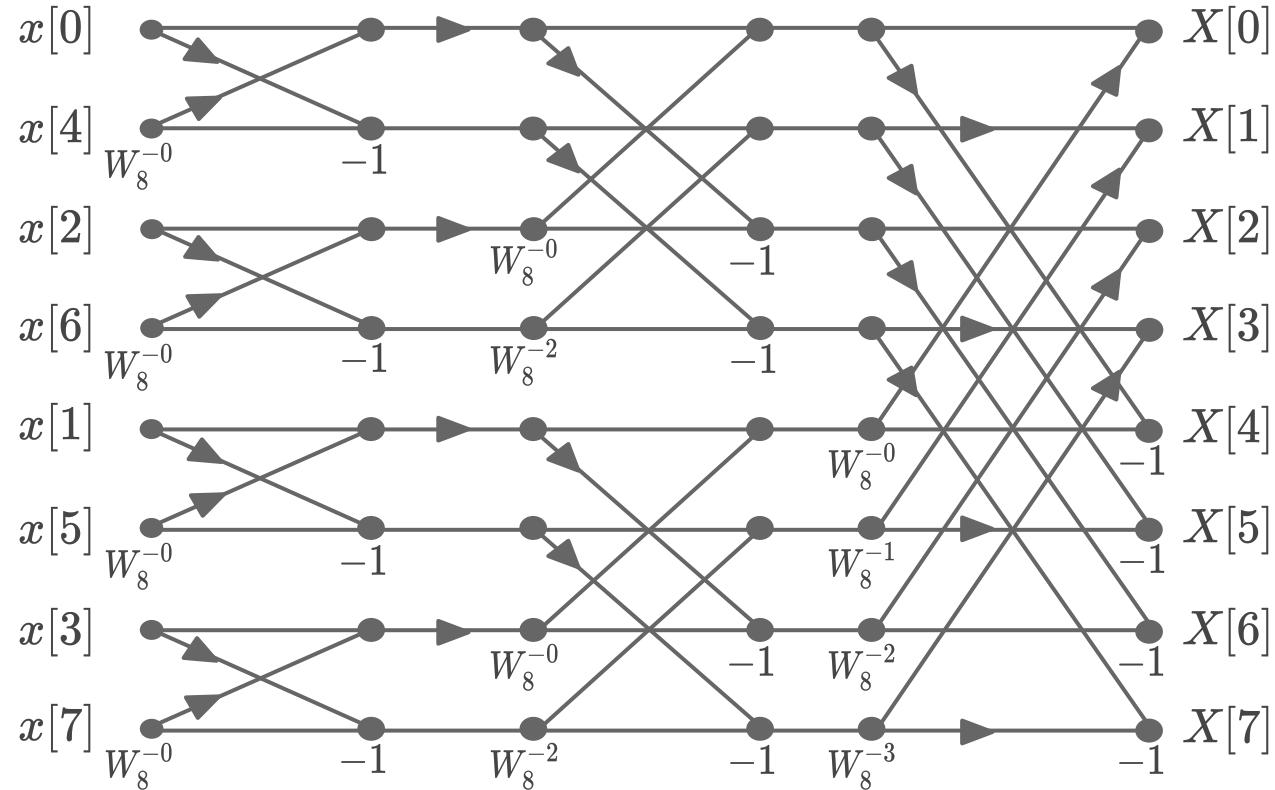
DFT から FFT へ： 分割したものを解釈する

$$\begin{bmatrix} X[0] \\ X[4] \end{bmatrix} = \begin{bmatrix} W_8^{-0} & W_8^{-0} \\ W_8^{-0} & -W_8^{-0} \end{bmatrix} \begin{bmatrix} x[0] + W_8^{-0}x[4] + W_8^{-0}(x[2] + W_8^{-0}x[6]) \\ x[1] + W_8^{-0}x[5] + W_8^{-0}(x[3] + W_8^{-0}x[7]) \end{bmatrix}$$



内積の計算の繰り返しによって最終的な結果が得られる

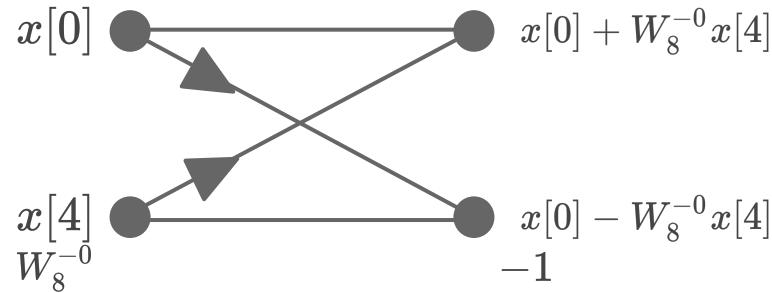
FFT 全体の演算を図で表す (黒丸で数値を乗算後に加算)



似たような演算手順 (次に示すバタフライ演算) の組み合わせ

バタフライ演算

FFTは複素加算2回，複素乗算1回の**バタフライ演算**を基本演算とし，これの組み合わせで構成される。Nが2のべき乗であれば同じ手順で計算可能。



DFT vs. FFT：計算量の比較

- バタフライ演算を基本演算とするFFTは N が2のべき乗の場合に適用可能でありその総数は $N/2 \log_2 N$ となる。
- 一般的な N の場合の計算量
 - DFT：複素加算 $N(N - 1)$, 複素乗算 N^2
 - FFT：複素加算 $N \log_2 N$, 複素乗算 $N/2 \log_2 N$
- 計算オーダ
 - DFT： $\mathcal{O}(N^2)$
 - FFT： $\mathcal{O}(N \log_2 N)$

FFT のためのビットリバース

- FFT では信号の並び替えが必要 → ビットを逆順にすることで実現可能

ビットリバース			
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

10進数 2進数

復習：時間領域の畳み込み演算は z 領域で乗算になる (これはフーリエ変換, ラプラス変換でも共通する)

$$\mathcal{Z}[h[n] * x[n]] = H(z)X(z)$$

$x = [1, 3, 2]$, $h = [2, 2, 3]$ とする. 畳み込み結果 $y[n]$ を求めよ.

$h[n]$	2	-2	3		
$x[n]$	1	3	2		
$x[n-0]$					
$x[n-1]$					
$x[n-2]$					
$h[0]x[n-0]$					
$h[1]x[n-1]$					
$h[2]x[n-2]$					
$y[n]$					

計算手順：

- 0シフト: $x[n-0] \cdot h[0]$
- 1シフト: $x[n-1] \cdot h[1]$
- 2シフト: $x[n-2] \cdot h[2]$
- 各列で足す

$$H(z)$$

$$X(z)$$

$$X(z)z^{-0}$$

$$X(z)z^{-1}$$

$$X(z)z^{-2}$$

$$h[0]X(z)z^{-0}$$

$$h[1]X(z)z^{-1}$$

$$h[2]X(z)z^{-2}$$

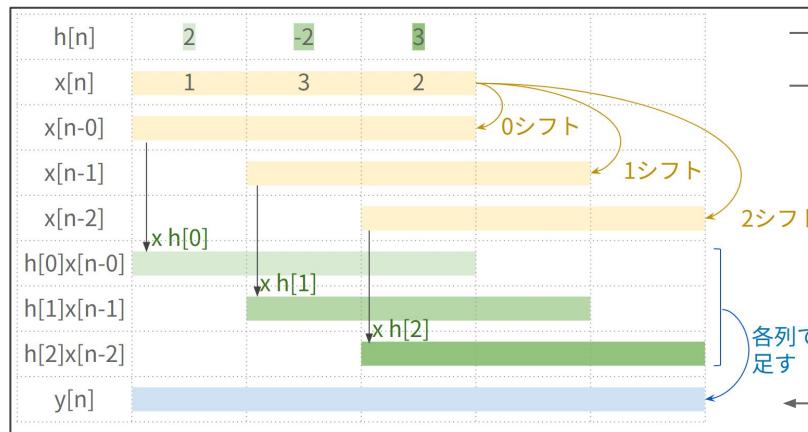
$$(h[0]z^{-0} + h[1]z^{-1} + h[2]z^{-2})X(z)$$

$$H(z)$$

第1回のスライドから

FFTは畠み込みの計算量を抑えるのにも役立つ

愚直に計算すると $O(N^2)$



$\mathcal{O}(N \log_2 N)$

FFT

$H[k]$

$X[k]$

k 每に複素乗算

逆FFT (IFFT)

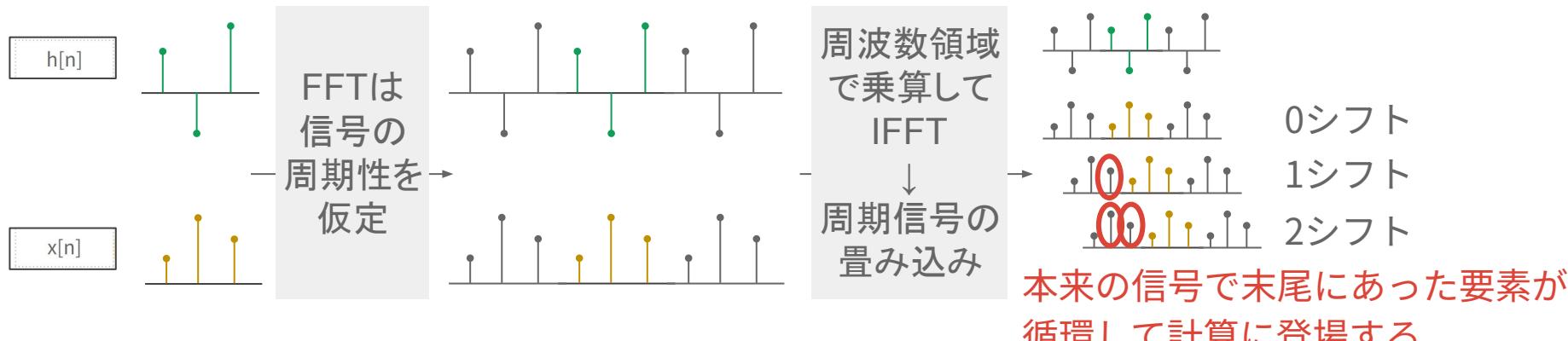
$\mathcal{O}(N \log_2 N)$

ただし

- $h[n]$ と $x[n]$ の長さは、等しく、かつ 2 のべき乗でなければならない
 - FFT の条件
- $h[n]$ と $x[n]$ の長さは、畠み込みの結果の長さを超えないければならない
 - 長さ N と長さ M の畠み込み結果の長さは $N + M - 1$

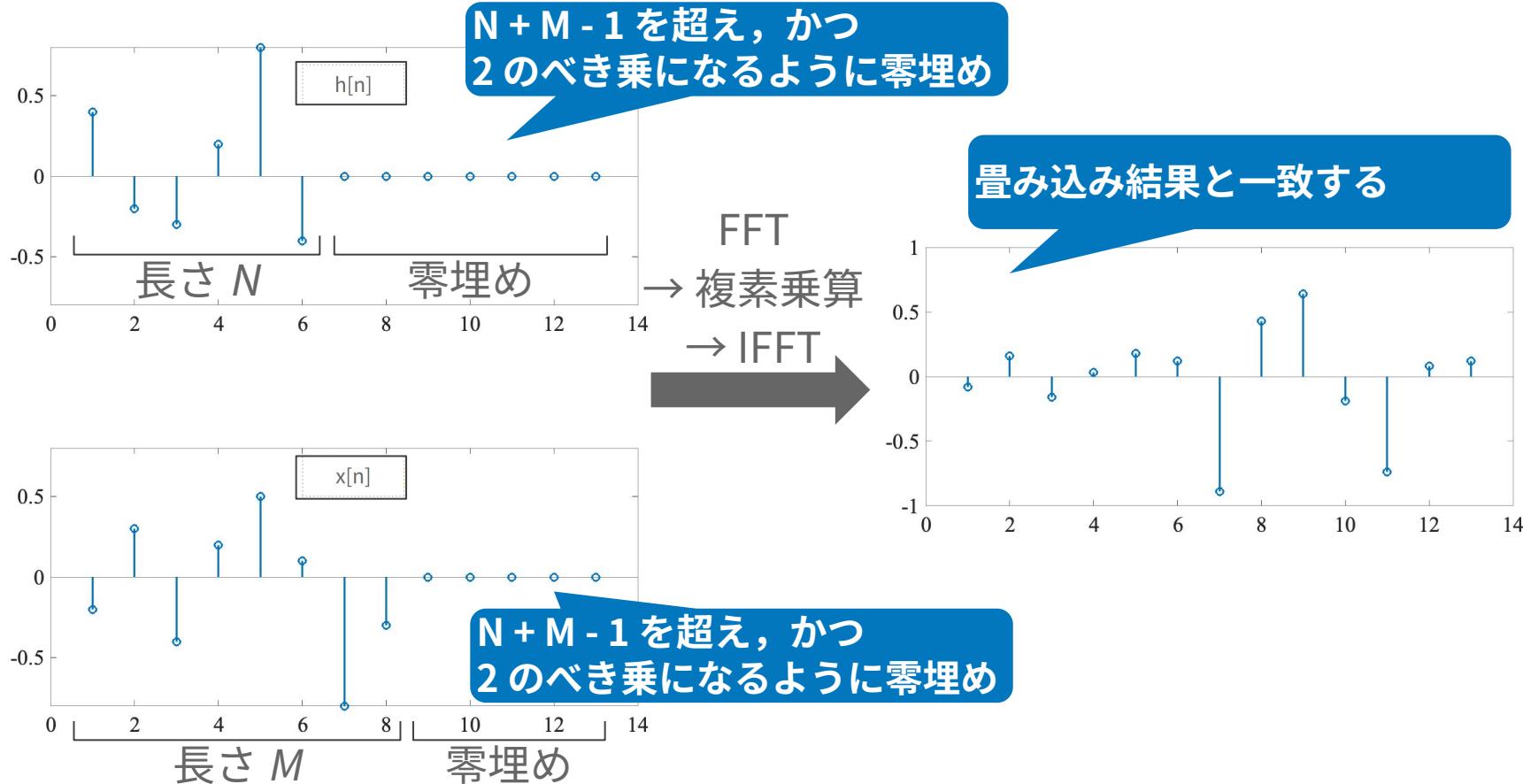
畳み込みを FFT (DFT) でやるときの注意

FFT (DFT) は信号の周期性を仮定するため、信号長が短いときは畳み込み結果が循環する（循環畳み込みと呼ばれる）



故に、循環の影響を受けないようにしたい. → 零埋め (zero padding)

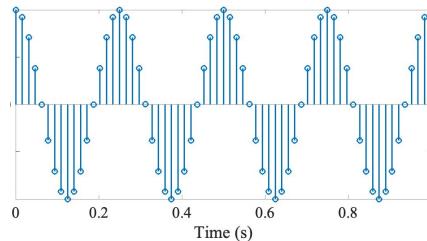
零埋め



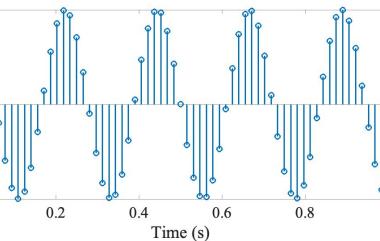
窓関数と短時間フーリエ変換 (window function and short-term Fourier transform)

離散Fourier変換の問題： 信号長が 1 sec (左) の場合 vs. 信号長が 2 sec (右) の場合

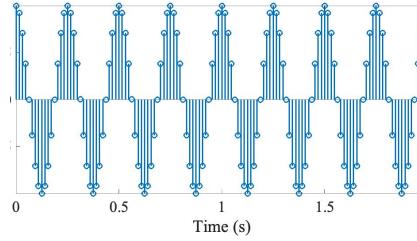
信号1
(周波数 4 Hz)



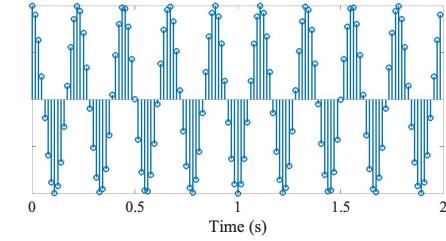
信号2
(周波数 4.5 Hz)



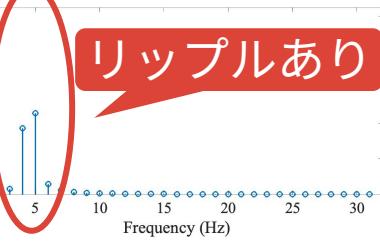
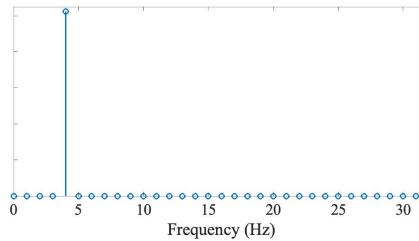
信号1
(周波数 4 Hz)



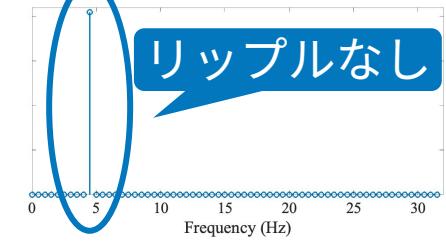
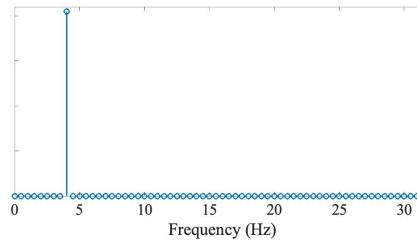
信号2
(周波数 4.5 Hz)



パワースペクトル



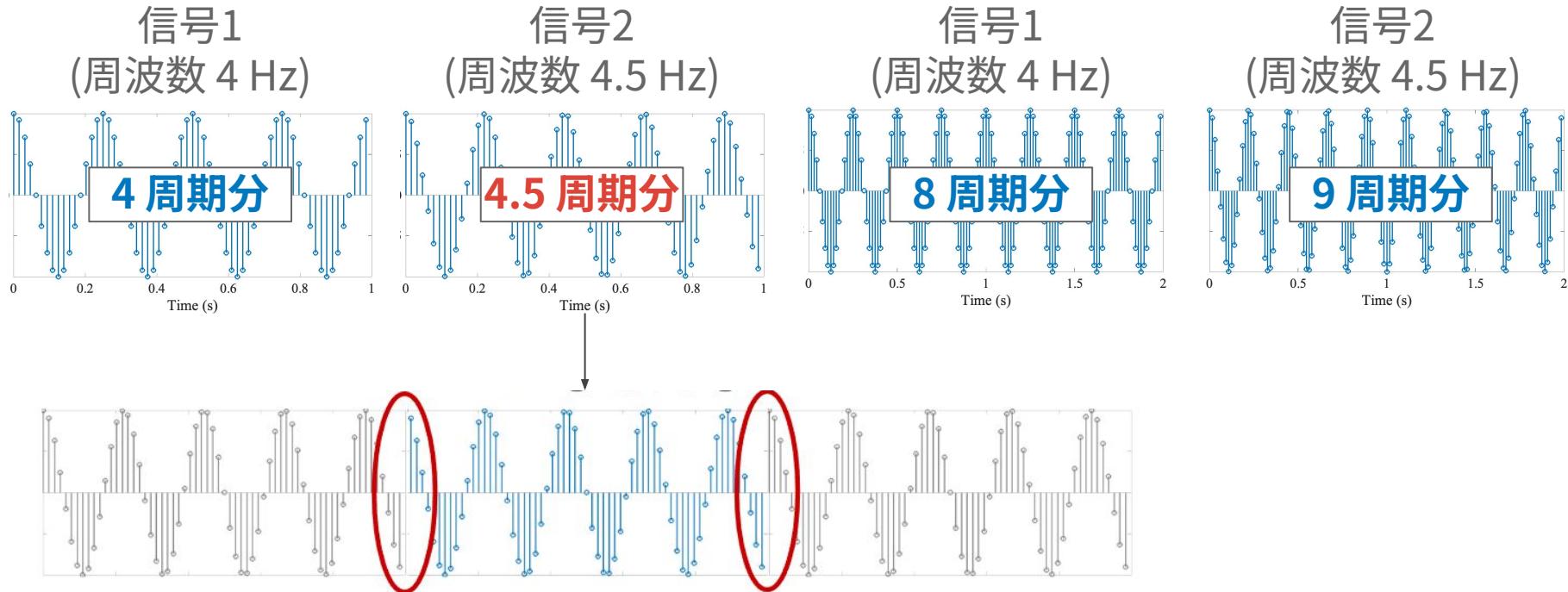
パワースペクトル



周波数 4.5 Hz の近傍にスペクトルが
染み出てしまっている…

同じ信号のはずなのに、この場合は
染み出していない。何故？

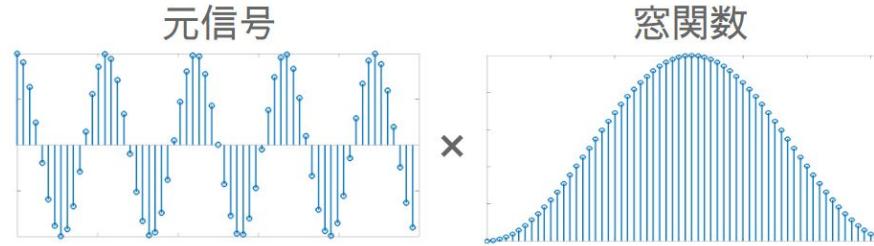
離散Fourier変換の問題： 信号長が 1 sec (左) の場合 vs. 信号長が 2 sec (右) の場合



離散Fourier変換は時間領域での周期性を仮定するため、切り出された信号の両端のつながり方によってはリップルが生じる。

窓関数の役割

- 窓関数によるリップルの抑圧
 - 信号長を対象に合わせて変更することは通常できないため、両端のつながりの影響を少なくする窓関数を信号に乗算してリップルを抑制する。



- 離散窓 Fourier 変換（代表例は次のスライド）
 - 離散時間信号 $x[n]$ に対し、信号区間の中心から離れるほど減衰する重み関数（窓関数） $w[n]$ を乗じて離散 Fourier 変換

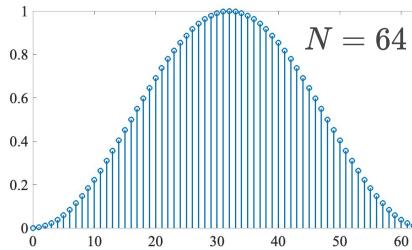
$$X[k] = \sum_{n=0}^{N-1} w[n] x[n] \exp\left(-j \frac{2\pi k n}{N}\right)$$

窓関数の乗算

よく使われる窓関数

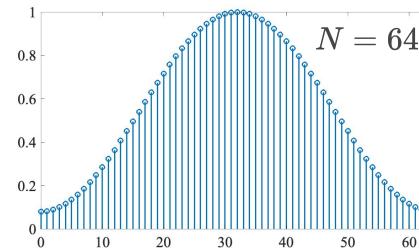
Hanning 窓

$$w[n] = \begin{cases} 0.5 - 0.5 \cos\left(\frac{2\pi n}{N}\right), & 0 \leq n \leq N \\ 0, & \text{otherwise} \end{cases}$$



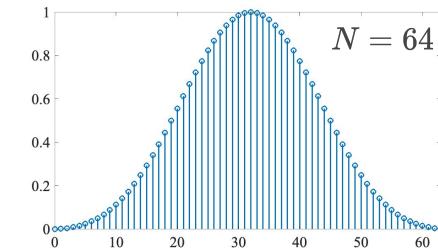
Hamming 窓

$$w[n] = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi n}{N}\right), & 0 \leq n \leq N \\ 0, & \text{otherwise} \end{cases}$$



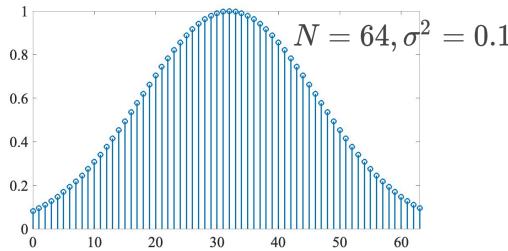
Blackman 窓

$$w[n] = \begin{cases} 0.42 - 0.5 \cos\left(\frac{2\pi n}{N}\right) + 0.08 \cos\left(\frac{4\pi n}{N}\right), & 0 \leq n \leq N \\ 0, & \text{otherwise} \end{cases}$$



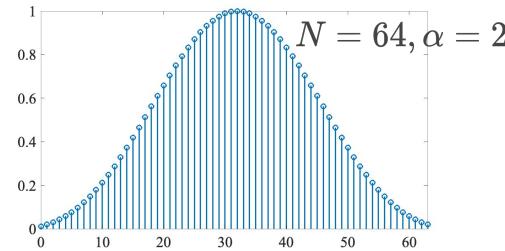
Gauss 窓

$$w[n] = \begin{cases} \exp\left(-\frac{1}{\sigma^2} \frac{(n-\frac{N}{2})^2}{N^2}\right), & 0 \leq n \leq N \\ 0, & \text{otherwise} \end{cases}$$



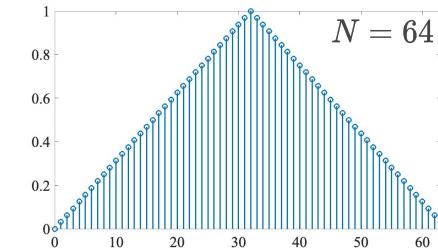
Kaiser 窓

$$w[n] = \begin{cases} \frac{I_0\left(\pi\alpha\sqrt{1-\left(\frac{2n}{N}-1\right)^2}\right)}{I_0(\pi\alpha)}, & 0 \leq n \leq N \\ 0, & \text{otherwise} \end{cases}$$



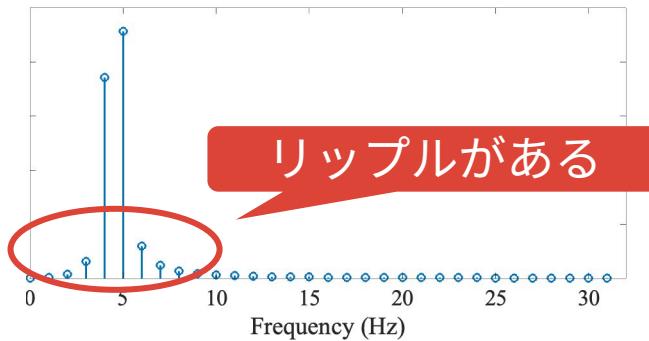
Bartlett 窓(三角窓)

$$w[n] = \begin{cases} \frac{2n}{N}, & 0 \leq n \leq \frac{N}{2} \\ 2 - \frac{2n}{N}, & \frac{N}{2} < n \leq N \\ 0, & \text{otherwise} \end{cases}$$

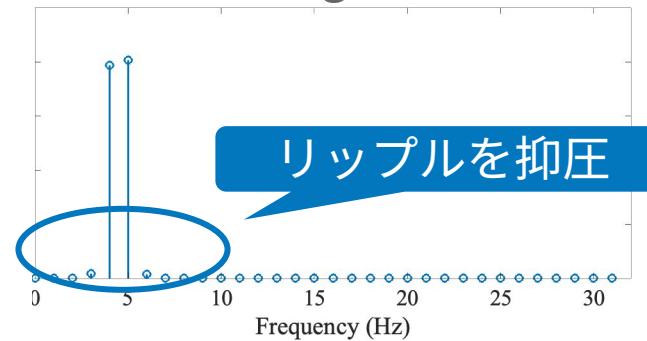


窓関数の効果：1 sec, 4.5 Hz 余弦波に窓関数を適用

矩形窓の場合



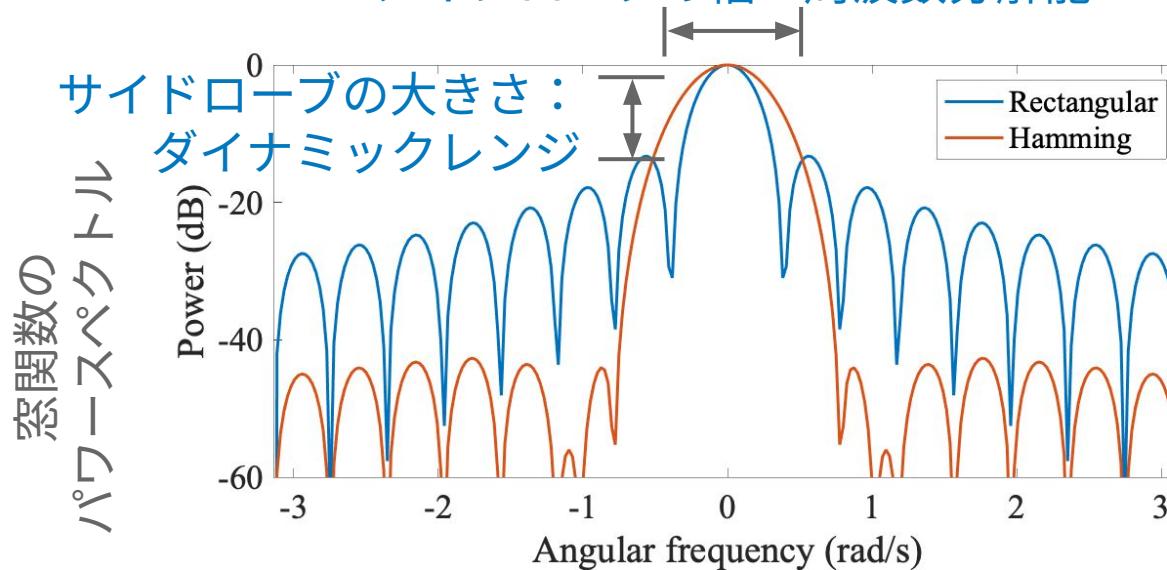
Hamming窓の場合



(時間区間を切り出してDFT
することは、矩形窓をかけて
いることと等価)

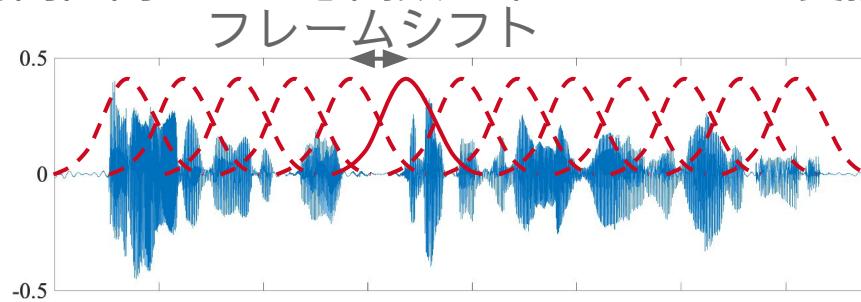
窓関数の性能

- 窓関数をかけることで元の信号が変わってしまうのでは?
 - 窓関数をかけることは、周波数領域における畳み込み
 - 窓関数の性能は周波数分解能とダイナミックレンジによって決まる
 - これらはトレードオフの関係にあるため、目的に合わせて選択する
- メインローブの幅：周波数分解能



短時間 Fourier 変換： 時間変化する信号に対する分析

- 時々刻々と変化する時系列信号を考える。
 - 長い時間区間の信号では、スペクトルの時間変化に意味がある場合が多い
 - (時間区間全体のスペクトルはあまり意味を持たない)
- 短時間 Fourier 変換 (short-time Fourier transform: STFT)
 - 短時間の時間区間ごとに窓関数を乗じて Fourier 変換

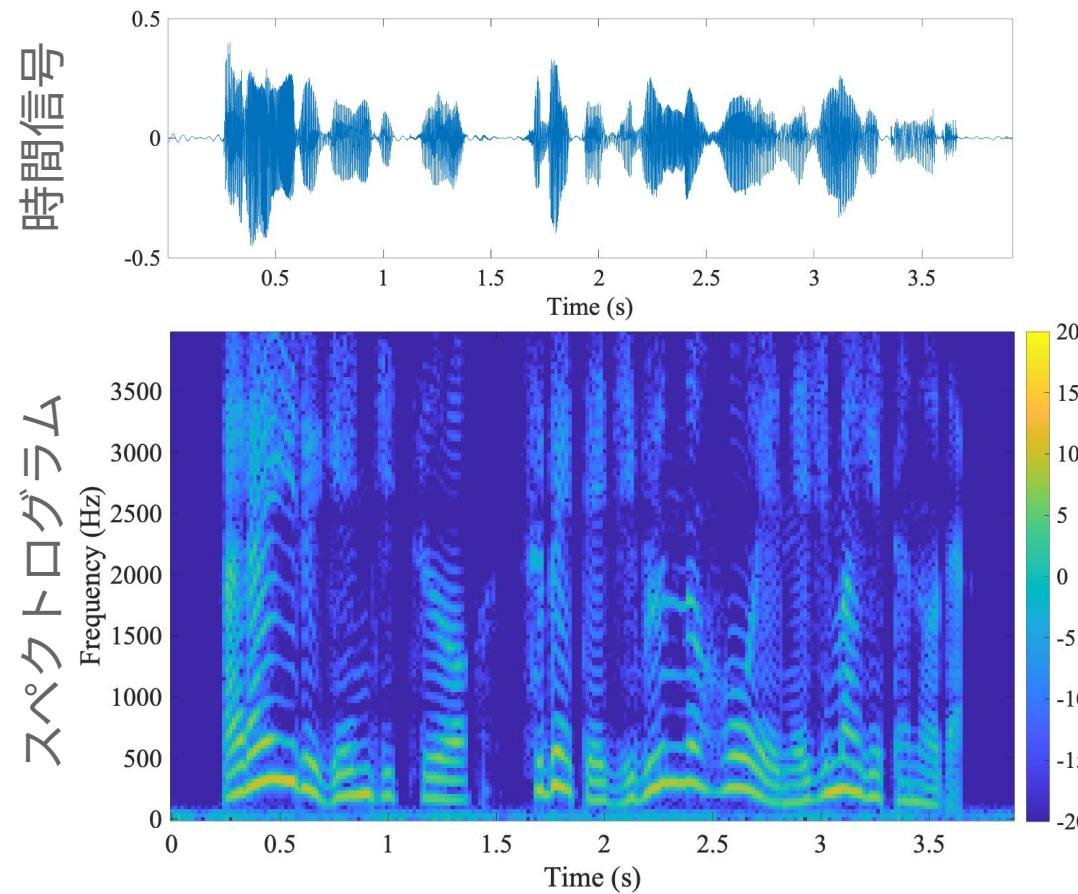


- 連続系での数式表現 (離散の場合も基本的に同じ)

$$X(\omega, t) = \int_{-\infty}^{\infty} x(\tau) w(\underline{\tau - t}) \exp(-j\omega\tau) d\tau$$

時刻 t を中心とした窓関数

短時間 Fourier 変換： スペクトログラム(時間-周波数と信号パワーの関係)



まとめ (matome)

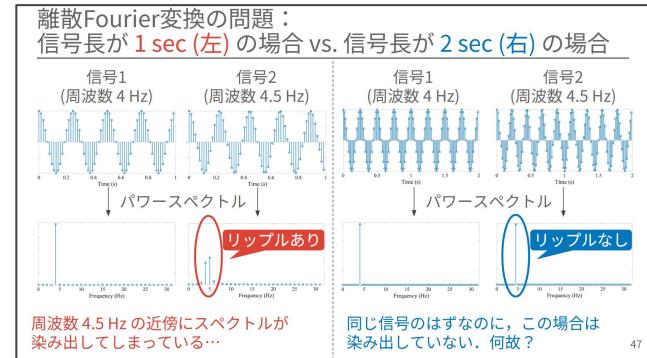
まとめ

- アナログフィルタからデジタルフィルタを作る
 - インパルス不变法，双一次変換
- 高速フーリエ変換
 - 信号長(タップ長)を 2^N に限定し，対称性を利用して計算量を小さくする
- 短時間フーリエ変換
 - 時々刻々とかわる信号を分析する

課題 (exercise)

演習

- ある周波数の正弦波を作成し、そのパワースペクトルにリップルが生じる場合と生じない場合を、プログラムで図示せよ。
また、矩形窓以外の窓関数でリップルを抑えられることを、プログラムで図示せよ。
- 下図の畠み込みを FFT を使って計算せよ。
すなわち、時間領域における畠み込みと、FFT を用いた畠み込みが一致することを示せ
(`numpy.fft.fft`, `numpy.fft.ifft`)



第 07 回について

第07回について

- 講義の直前に課題をアップロードします.
- いつも通り講義室に集合してください.
- 講義時間中にその課題を解いてください. 適宜, 話し合ったり資料を見てください. 質問はいつでも受け付けます.
- 一部の課題については, 解き方を解説します.
- 課題の提出は不要です. 講義後の課題もありません.