デジタル信号処理

第7回　高速フーリエ変換

立命館大学
情報理工学部
李　亮

離散フーリエ変換の問題点

- 離散フーリエ変換

$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(-j\frac{2\pi nk}{N}\right)$$

  - Nを大きくすればスペクトル解析精度は向上
  - しかし、Nが大きくなると計算量は膨大
  - 計算時間がかかるとシステムの負荷大

⇒ 高速フーリエ変換の提案

高速フーリエ変換

- 高速フーリエ変換とは
  - FFT: Fast Fourier Transform
  - スペクトル解析を高速に行うためのアルゴリズム
  - 離散フーリエ変換と高速フーリエ変換のスペクトルは同じ
- 高速フーリエ変換の特徴
  - スペクトル解析のための計算回数/時間を削減
  - 解析結果は、離散フーリエ変換と同じ
  - 計算時間を短縮可能
  - ただし、データ数は $N = 2^P$ という制約あり
  - 計算量は
  - 離散フーリエ変換<O（N^2）>
  - 高速フーリエ変換<O（$N \log_2 N$）>

演習課題(1/1)

- 離散フーリエ変換(DFT)の計算回数をN^2、高速フーリエ変換(FFT)の計算回数を$N \log_2 N$としたとき、下記の計算回数表を埋めよ。

データ数N	DFT	FFT	DFT/FFT
32	?	?	?
64	?	?	?
128	?	?	?
256	?	?	?

ヒント: $N=32= 2^5$ よって$\log_2 N$ は?

FFTへの道のり

- どうやって高速にフーリエ変換を実現するか？
  - 鍵となるのは回転子

$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(-j\frac{2\pi nk}{N}\right)$$

$$W_N = \exp\left(-j\frac{2\pi}{N}\right)$$

$$X_k = \sum_{n=0}^{N-1} x_n \, W_N{}^{nk} = \sum_{n=0}^{N-1} x_n W_N{}^{m}$$

ただし $m = nk$

- この回転子 $W_N{}^m$ の性質が計算量削減の大きな役割を果たす

回転子 W_N^m の性質

$$X_k = \sum_{n=0}^{N-1} x_n W_N^m$$

- 周期性:
  - 回転子は周期Nの関数。よって任意のpに対して

$$W_N^m = W_N^{m \pm pN} \qquad (1)$$

- 指数乗算の性質:
  - p<mである任意のpに対して

$$W_N^m = W_N^p W_N^{m-p} \qquad (2)$$

  - N/2進めると符号が反転させたものと等しくなる

$$W_N^m = -W_N^{m+N/2} \qquad (3)$$

DFTアルゴリズム

- N=4とすると

$$X_k = \sum_{n=0}^{3} x_n \exp\left(-j\frac{2\pi nk}{4}\right)$$
$$= \sum_{n=0}^{3} x_n W_4{}^m \quad \text{ただし } m = nk$$

- K=0,1,2,3として行列を使って展開すると

$$\begin{array}{cccccc}
 & X_k & & W_4{}^m = W_4{}^{nk} & x_n & \\
\begin{array}{c} k=0 \\ k=1 \\ k=2 \\ k=3 \end{array} & \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} & = & \begin{bmatrix} W_4{}^0 & W_4{}^0 & W_4{}^0 & W_4{}^0 \\ W_4{}^0 & W_4{}^1 & W_4{}^2 & W_4{}^3 \\ W_4{}^0 & W_4{}^2 & W_4{}^4 & W_4{}^6 \\ W_4{}^0 & W_4{}^3 & W_4{}^6 & W_4{}^9 \end{bmatrix} & \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} & \begin{array}{c} n=0 \\ n=1 \\ n=2 \\ n=3 \end{array} \\
 & & & \begin{array}{cccc} n=0 & n=1 & n=2 & n=3 \end{array} & &
\end{array}$$

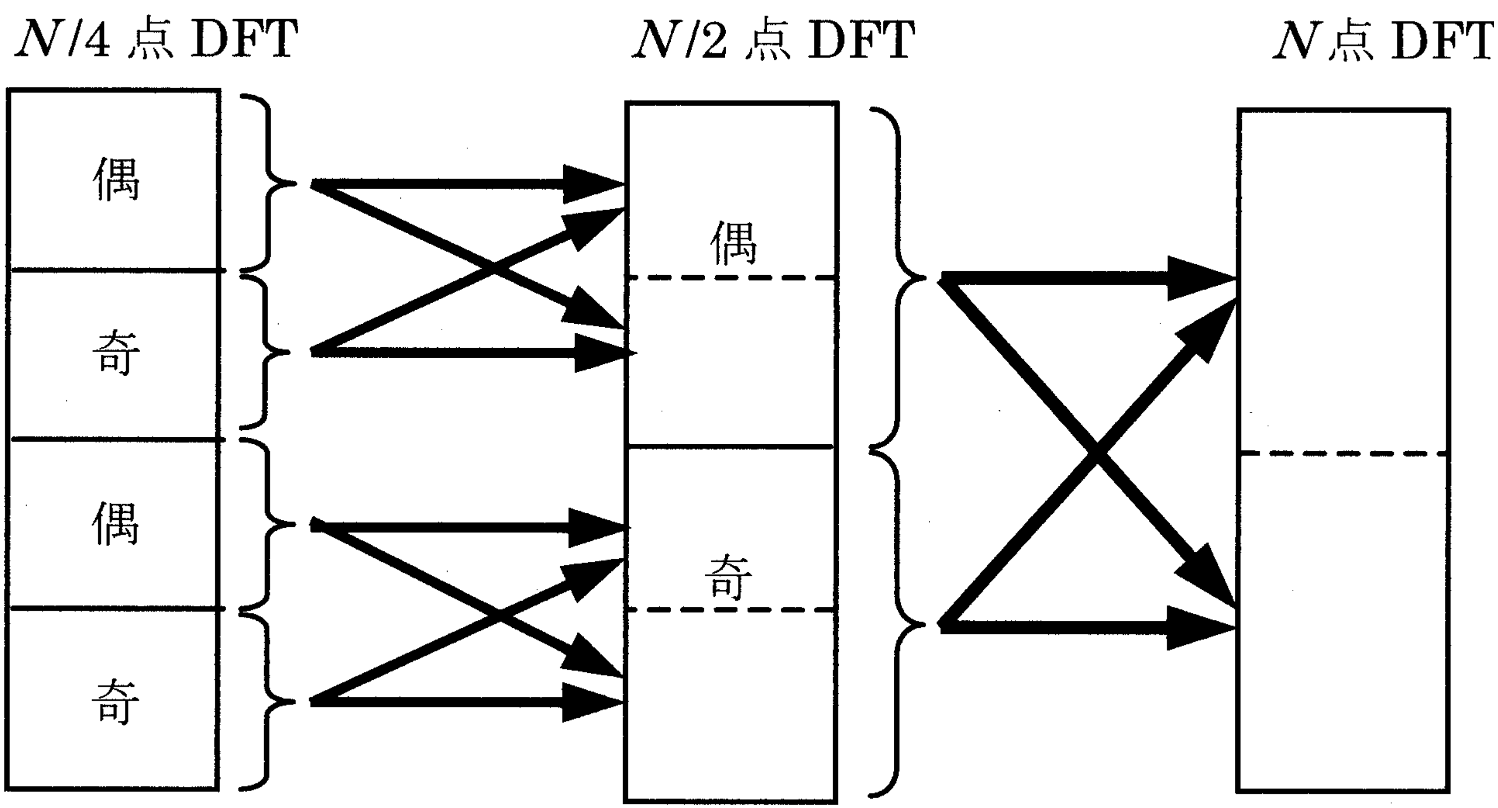
- データ数をNとすると積和演算回数はN^2回となる

FFTアルゴリズム（1）

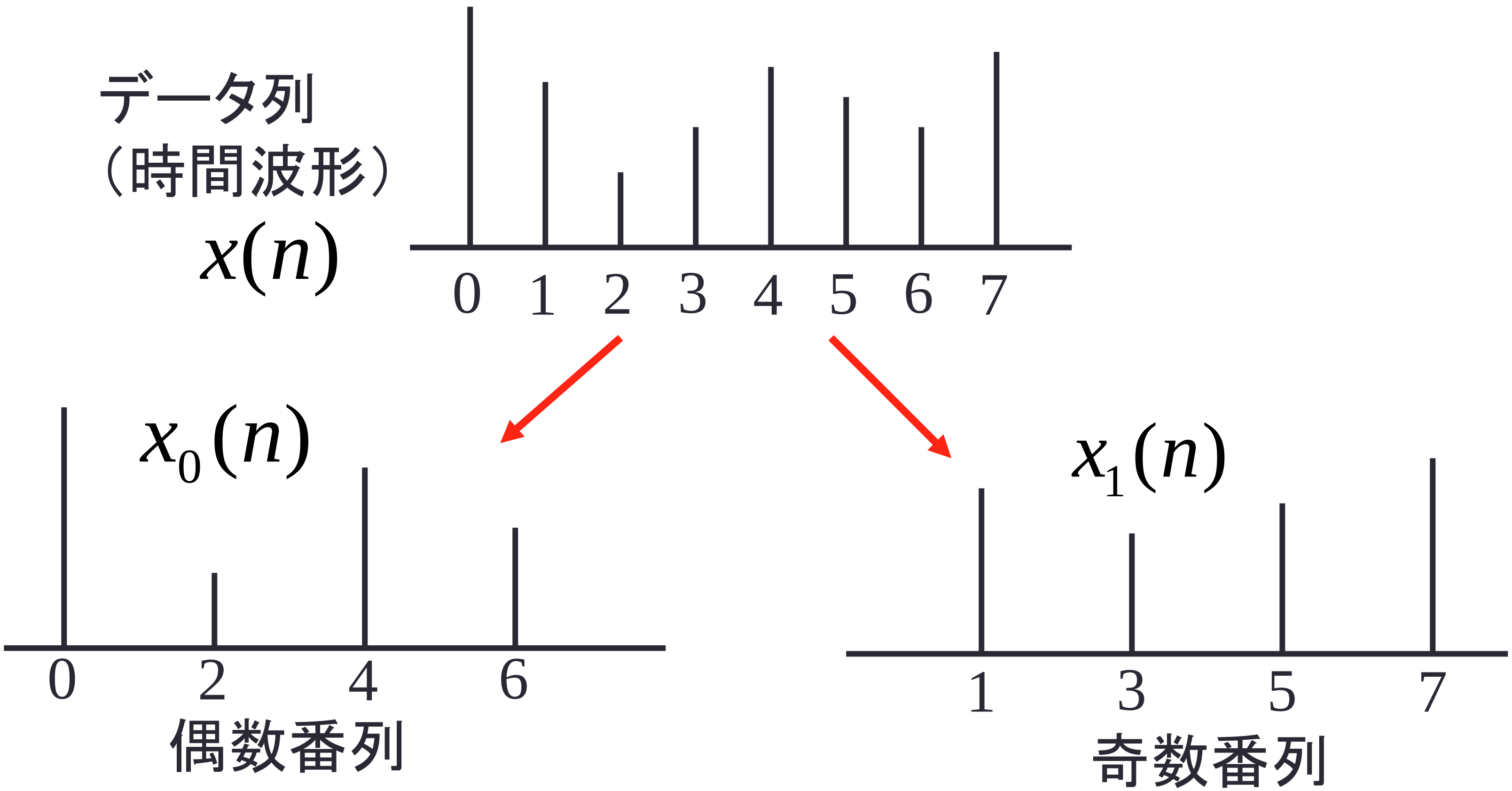
- ポイントはデータ分割と回転子の特徴
  - データ分割することで効率よく変換を行う
    - ソートアルゴリズムであれば、
      - バブルソートのような総当りのアルゴリズムがDFT
      - クイックソートのようなデータ分割型のアルゴリズムがFFT
    - と同じ考え方である
  - データ分割のポイント
    - データを分割することでデータ長Nを見かけ上、小さくしてDFTを繰り返し行う。
    - 例えばN＝32点のとき
      - N=32点のDFTの計算回数は1024回
      - データ分割して例えば16点のDFTを2回行うとN=16点のDFTの計算回数は256回(N^2)なので、計算回数は256 × 2 = 512回
      - 実際にはNをできるだけ小さく分割して、あとからマージする。

FFTアルゴリズム(2)

- FFTの基本的な流れ
  - 基本はデータ分割(偶数番と奇数番で分割)



FFTアルゴリズム（3）



時間波形を偶数番と奇数番で2つの系列に分解
すると、スペクトルは次ページのように表せる。
このとき必ずデータ列は　$N = 2^P$　の必要あり。

FFTアルゴリズム(4)

- データ列:　$x(n)$　　スペクトル:　$X(k)$
- 偶数番列:　$x_0(n)$　　偶数スペクトル:　$X_0(k)$
- 奇数番列:　$x_1(n)$　　奇数スペクトル:　$X_1(k)$

$$X(k) = \left\{ X_0(k) + e^{-j\frac{2\pi}{N}k} X_1(k) \right\}$$

$$X\left(\frac{N}{2}+k\right) = \left\{ X_0(k) - e^{-j\frac{2\pi}{N}k} X_1(k) \right\}$$

FFTアルゴリズム(5)

$$X(k) = \left\{ X_0(k) + e^{-j\frac{2\pi}{N}k} X_1(k) \right\}$$

$$X\left(\frac{N}{2}+k\right) = \left\{ X_0(k) - e^{-j\frac{2\pi}{N}k} X_1(k) \right\}$$

- 式より、N＝4とするとスペクトルは

$$X(0) = \left[X_0(0) + e^{-j\frac{\pi}{2}\times 0} X_1(0) \right] \qquad X(2) = \left[X_0(0) - e^{-j\frac{\pi}{2}\times 0} X_1(0) \right]$$

$$X(1) = \left[X_0(1) + e^{-j\frac{\pi}{2}\times 1} X_1(1) \right] \qquad X(3) = \left[X_0(1) - e^{-j\frac{\pi}{2}\times 1} X_1(1) \right]$$

ポイント： 分割を繰り返し行い、同じ項を繰り返し使う形にする

FFTアルゴリズム(6)

$$X(0) = \left[X_0(0) + e^{-j\frac{\pi}{2}\times 0} X_1(0) \right] \quad X(2) = \left[X_0(0) - e^{-j\frac{\pi}{2}\times 0} X_1(0) \right]$$

$$X(1) = \left[X_0(1) + e^{-j\frac{\pi}{2}\times 1} X_1(1) \right] \quad X(3) = \left[X_0(1) - e^{-j\frac{\pi}{2}\times 1} X_1(1) \right]$$

- 結果、計算回数はデータ数をNとすると
  - FFTの積和演算回数　$N\log_2 N$
  - DFTの積和演算回数　N^2

- 一般的に通常のスペクトル解析にはFFTを使う

FFTのまとめ

- 高速フーリエ変換の特徴
  - スペクトル解析を高速かつ少ない演算量で行える。
  - ポイントはデータ系列の分解と回転子の有効利用
    - FFTはデータ長を分割して短いデータ長のDFTを複数回行うことで高速化を実現(FFTはDFTの応用)
  - ただしデータ長は $N = 2^P$ の必要がある。
    - もし2^P でなければ、0付加(ゼロ詰め)してデータ長を 2^P にする
  - 通常のスペクトル解析ではFFTを使用する。
    - 実際の商品にもFFTアルゴリズムが使用されていることが多い
  - FFTもスペクトル解析手法の1つなので、窓掛けなどを行い、不連続点の影響を除去する必要あり。
    - DFTもFFTもアルゴリズムが異なるだけで結果は同じ
    - よって算出したスペクトルも同じ