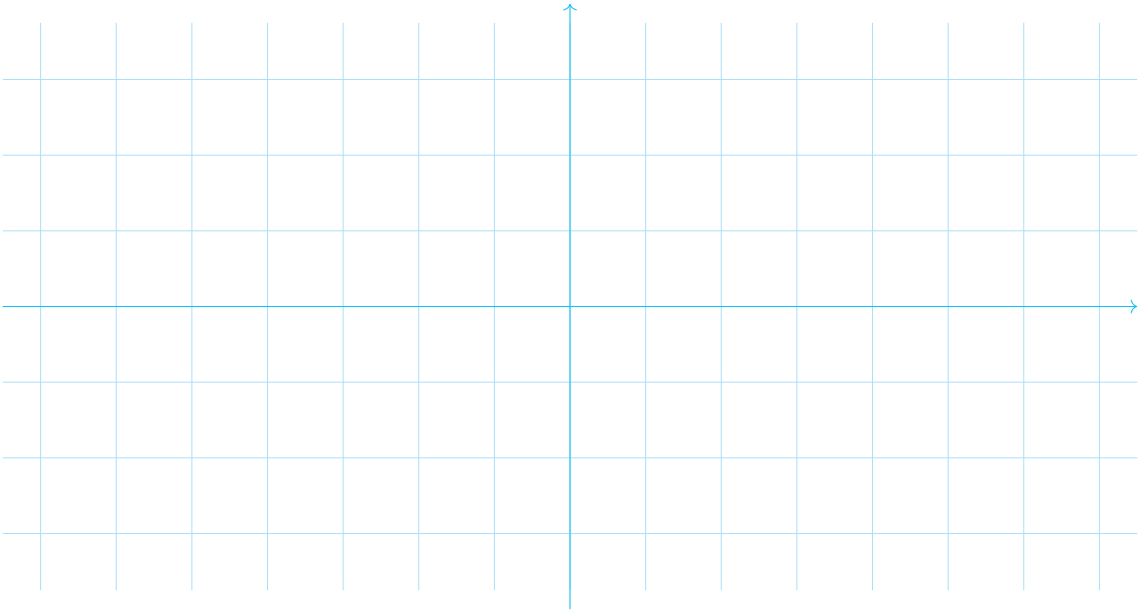
memo



信号処理

授業開始までしばらくお待ちください。

信号処理
Signal Processing

『計算機演習』

rebrand.ly/sigproc

小林裕之

大阪工業大学 RD 学部システムデザイン工学科

OSAKA INSTITUTE OF TECHNOLOGY

10 of 14

a LaTeX + Beamer slideshow

授業の受講に関して

- 講義資料（スライド等）は **Google Drive** (https://rebrand.ly/sigproc) に置く (紙の配布資料は行わない)。授業前には虫喰い状態のスライドのみを提供するが、授業後に uncoveredフォルダに穴埋め版を置くので復習に活用されたい。アカウントの問題等でアクセスできないときのために https://www.oit.ac.jp/rd/labs/kobayashi-lab/~yagshi/lectures/にも置いておく。
- ミニレポートは**Google Forms** (https://forms.gle/1PdgpYDpnahECqkY6) に提出。

授業の進め方

- 出席そのものは評価せず。極論するとテストのみ出席で他は全欠席でも A 評価はあり得る。
- 基本的には**中間演習**と**期末試験**で評価。
- 毎回ミニレポートを課す。出す者は提出期間を厳守すること。
- 試験の不合格者は**毎回のミニレポート**と**出席**で少し救済する。
  (**しっかりした内容の**ミニレポートを概ね 9 割以上提出し、かつ大学の出欠管理システムで 8 割以上遅刻せず出席していた場合最大 10 点程度の救済。提出数や出席数が少ない場合は救済幅が縮小する。いずれかが 7 割を下回ったら一切救済しない。締め切り後の提出は認めない。)
- スライド穴埋め版はその回の授業終了後に公開。
- **授業中に**スライドの誤りを見つけて指摘してくれた者には、誤り一箇所につき先着一名様限り 100 点満点 1 点相当の加点を行う。(ただしごく軽微なものなど、内容によっては加点しない場合もあり。)

FFT ふたたび

さらに効率の良い FFT: いわゆるバタフライ演算

計算量はこの前やったのと同じです。

- 前回の FFT 実装は式のまま、トップダウンに再帰的に実装した。
- それはそれで非常に美しいがパフォーマンス（特にメモリ効率）的に特別優れているとも言えない。
- そこでもっと効率の良い実装を行う。マイコン等でも十分使える。

バタフライ FFT 実装 Step 1: 並べ替え

問.

- $(0, 1, 2, \cdots, 2^N - 1)$ を偶数分と奇数部分に分けて並べると
  - $(0, 2, 4, \cdots)$ と $(1, 3, 5, \cdots)$ である。
  - 全部つなげると $(0, 2, 4, ..., 2^N - 2,\ 1, 3, 5, ...2^N - 1)$。
- それをさらに分けると
  - $(0, 4, 8, \cdots)$ と $(2, 6, 10, \cdots)$ と $(1, 5, 9, \cdots)$ と $(3, 7, 11, \cdots)$ である。
  - 全部つなげると $(0, 4, 8, \cdots, 2, 6, 10, \cdots, 1, 5, 9, \cdots, 3, 7, 11, \cdots)$ 。

この調子でそれ以上分けられないところまで分ける。

1. 全部つなげた並びはどうなるか考え、(ヒント: 偶数奇数なので 2 進数で考えるといいかも……。)
2. 与えられたリストそのようにを並び替える関数[a]を作成せよ。

[a] リストを引数とし、リストを return する関数

バタフライ FFT 実装 Step 2: 倍々サイズにまとめながら積み上げる

1. **0**段目は**1**点からなるN個のブロックに分け、それを**2**つずつ使って**計算**。
2. **1**段目は**2**点からなる$\frac{N}{2}$個のブロックに分け、それを**2**つずつ使って**計算**。
3. **2**段目は**4**点からなる$\frac{N}{4}$個のブロックに分け、それを**2**つずつ使って**計算**。
4. **k**段目は2^k点からなる$\frac{N}{2^k}$個のブロックに分け、それを**2**つずつ使って**計算**。
5. $M = \log_2 N - 1$段目は2^M点からなる 2 個のブロックに分け、その**2**つを使って**計算**。

『計算』のなか身

2 つのうち前半を X_e、後半を X_o としてあとは以下の #9 でやった以下の式そのまま

$$X[k] = \begin{cases} X_e[k] & +W_N^k X_o[k] \quad (0 \le k < \frac{N}{2}) \\ X_e[k-\frac{N}{2}] & +W_N^k X_o[k-\frac{N}{2}] \quad (\frac{N}{2} \le k < N) \end{cases}$$

numpy や scipy で実際に DFT を使う

実際の周波数と DFT 演算で得られるデータの関係

60 Hz と 25 Hz の正弦信号をサンプリングして周波数解析をする、というストーリーで numpy, scipy を使ってみよう。(実験で得られた 25Hz の信号に商用電源の 60Hz が重なっている、みたいな設定。)

- 振幅、位相、パワースペクトルを求めてみる。
- サンプリング周波数を変えてみる。
- エイリアシングを確認する。

「正弦信号をサンプリングして得られたデータ」の作り方

numpy を使います。

$x(t) = \sin(2\pi \times 60t) + 2\sin(2\pi \times 25t + 1)$ を 100 Hz でサンプリングした (という体のデータ)

```
import numpy as np
import matplotlib.pyplot as plt
fs = 100                                    # サンプリング周波数
duration = 2                                # 2 秒間のデータを取る
t = np.arange(0, 2, 1 / fs)                 # "arrange" じゃないよ
xn = np.sin(2 * np.pi * 60 * t) + 2 * np.sin(2 * np.pi * 25 * t + 1)
plt.plot(xn)
plt.show()
```

サンプリングしたデータを DFT し、パワスペクトルを見る。

numpy を使います。

前のページの続き

```
xk = np.fft.fft(xn)     # 複素数値 X[k] が得られる。
pk = xk * xk.conj()     # 複素共役をかけてパワを得る。
pkr = pk.real           # 実数部 (当然虚数部は 0 だが、型合わせのため)
plt.plot(pkr)
plt.show()
```

DFT したあとのデータの読み方

- 横軸の N(サンプル点数) が、f_s に当たる。
- 横軸の 1 ステップ（周波数分解能）は f_s/N である。
- つまり、長時間データを取ればそれだけ分解能が上がる。
- 半分より右側は情報的には意味がない (冗長である)。

問. 5Hz と 15Hz の正弦信号の合成信号を $f_s = 25$ Hz で 0.8 s サンプリングし、DFT を行った。それぞれの信号は DFT 後の周波数軸上の何番目のところ ($X[k]$ の k) にピークとして現れるか?

ミニレポート課題 (提出期間: 本日〜今週金曜)

p. 12の問を説明を添えて解け。計算 and/or PC 等による実際のデータを示して説明すること。

解答を PC 文書や手書きで作成し、PDF にして Google Forms (**https://forms.gle/1PdgpYDpnahECqkY6**) から提出せよ (要組織アカウントによるログイン)。ただし写真等の画像ファイルの場合は、解像度や露出・照明状態などを十分考慮し、きちんと読解可能なクオリティのものとすること。スマートフォンの場合はスキャナアプリの類の利用を必須とする。