# フーリエ変換

# 目次

- 離散フーリエ変換
- 高速フーリエ変換

# フーリエ変換
# はじめに

- 地球物理の世界では, 現象を周期や振幅を用いて議論することが多々ある. また, エネルギーのやり取りを, しばしば波数空間で議論することもある. さらに, 気象業界では, 微分方程式を数値的に解く際に, 波数空間で解くことも珍しくない（例えば, スペクトル法）.
- これらにおいて用いる基本的な概念がフーリエ変換であり, それを離散化した離散フーリエ変換である. ここでは, 離散フーリエ変換について考え, 周波数解析を体験しよう.

# 離散フーリエ変換 (1)

- 扱うデータは離散的なので, フーリエ変換というよりは, フーリエ級数を扱っていることに近いだろう.
  - フーリエ変換

$$\tilde{f}(k) = \int_{-\infty}^{\infty} f(x)\, e^{ikx} dx, \qquad f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \tilde{f}(k)\, e^{-ikx} dk$$

  - フーリエ級数

$$\tilde{f}_n = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x)\, e^{inx} dx, \qquad f(x) = \sum_{n=-\infty}^{\infty} \tilde{f}_n e^{-inx}$$

# 離散フーリエ変換 (2)

- フーリエ級数

$$\tilde{f}_n = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)\, e^{inx} dx, \qquad f(x) = \sum_{n=-\infty}^{\infty} \tilde{f}_n e^{-inx}$$

- フーリエ級数を基に, 下のように離散化する.

$$\tilde{f}_n = \frac{1}{N(\Delta x)} \sum_{m=0}^{N-1} f_m e^{ik_n x_m} (\Delta x)$$

$$= \frac{1}{N(\Delta x)} \sum_{m=0}^{N-1} f_m e^{i\left(n\frac{2\pi}{N(\Delta x)}\right)(m(\Delta x))} (\Delta x)$$

$$= \frac{1}{N} \sum_{m=0}^{N-1} f_m e^{i\frac{2\pi mn}{N}}$$

$$f_m = \sum_{n=0}^{N-1} \tilde{f}_n e^{-ik_n x_m}$$

$$= \sum_{n=0}^{N-1} \tilde{f}_n e^{-i\left(n\frac{2\pi}{N(\Delta x)}\right)(m(\Delta x))}$$

$$= \sum_{n=0}^{N-1} \tilde{f}_n e^{-i\frac{2\pi mn}{N}}$$

# 離散フーリエ変換 (1)

- フーリエ変換

$$\tilde{f}(k) = \int_{-\infty}^{\infty} f(x)\, e^{ikx} dx, \qquad f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \tilde{f}(k)\, e^{-ikx} dk$$

- は, 下のように離散化される.

$$\tilde{f}_n = \frac{1}{(\Delta x)} \sum_{m=0}^{N-1} f_m e^{ik_n x_m} (\Delta x)$$

$$= \frac{1}{(\Delta x)} \sum_{m=0}^{N-1} f_m e^{i\left(n\frac{2\pi}{N(\Delta x)}\right)(m(\Delta x))} (\Delta x)$$

$$= \sum_{m=0}^{N-1} f_m e^{i\frac{2\pi mn}{N}}$$

$$(n = 0, 1, \ldots, N-1)$$

$$f_m = \frac{1}{\frac{2\pi}{(\Delta x)}} \sum_{n=0}^{N-1} \tilde{f}_n e^{-ik_n x_m} (\Delta k)$$

$$= \frac{1}{\frac{2\pi}{(\Delta x)}} \sum_{n=0}^{N-1} \tilde{f}_n e^{-i\left(n\frac{2\pi}{N(\Delta x)}\right)(m(\Delta x))} \left(\frac{2\pi}{N(\Delta x)}\right)$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} \tilde{f}_n e^{-i\frac{2\pi mn}{N}}$$

# 練習問題

- 離散フーリエ変換のプログラムを作ってみよう.

# 離散フーリエ変換の単位

- フーリエ変換後の値の単位はフーリエ変換前と同じ.
- 「パワー」とは?

# ナイキスト周波数

- 一定の時間間隔, $T$, で温度を測定することを考える.
- 現象の時間スケールに比べて温度の測定間隔が大きい場合, 現象の特徴を捉えられない. つまり, 捉えたい時間スケールに対して, 測定間隔には上限がある（測定する周波数＝測定する間隔の逆数には下限がある）.
- 逆に言えば, ある間隔での測定データにおいて意味のある周波数には上限がある.
- この周波数をナイキスト周波数と呼ぶ.
- ナイキスト周波数, $f$, は, 測定間隔, $T$, を使って下のように書ける.

$$f = \frac{2\pi}{2T}$$

# エイリアシング

- 一定の時間間隔, $T$, で温度を測定することを考える.
- ここで測定される温度に, ナイキスト周波数以上の周波数を持った現象が含まれている場合, その周波数成分は別の周波数の成分の波として検出されてしまう.
- この現象をエイリアシングと呼ぶ.

- したがって, 測定時には, 捉えたい周波数以外の周波数成分をフィルタを使うなどして除去することが必要になる.

# 高速フーリエ変換 (1)

- 工夫することで, フーリエ変換を高速に実行する / フーリエ変換の演算量を減らす方法が考えられている.
  - 代表的なのは, Cooley-Tukey 型アルゴリズム (Cooley and Tukey, 1965)

- 以下では, データ数 N が 2 のべき乗の場合についてのみ考える.
  - それ以外の場合には, N の因数に対応した演算を行えばよい.

# 高速フーリエ変換 (2)

$$\tilde{f}_n = \frac{1}{(\Delta x)} \sum_{m=0}^{N-1} f_m e^{ik_n x_m} (\Delta x) = \frac{1}{(\Delta x)} \sum_{m=0}^{N-1} f_m e^{i\left(n\frac{2\pi}{N(\Delta x)}\right)(m(\Delta x))} (\Delta x) = \sum_{m=0}^{N-1} f_m e^{i\frac{2\pi mn}{N}}$$

$$= \sum_{\substack{m=0 \\ (even)}}^{N-1} f_m e^{i\frac{2\pi mn}{N}} + \sum_{\substack{m=0 \\ (odd)}}^{N-1} f_m e^{i\frac{2\pi mn}{N}}$$

$$= \sum_{\substack{m=0 \\ (even)}}^{2N'-1} f_m e^{i\frac{2\pi mn}{2N'}} + \sum_{\substack{m=0 \\ (odd)}}^{2N'-1} f_m e^{i\frac{2\pi mn}{2N'}} \qquad (N = 2N')$$

$$= \sum_{m'=0}^{N'-1} f_{2m'} e^{i\frac{2\pi(2m')n}{2N'}} + \sum_{m'=0}^{N'-1} f_{2m'+1} e^{i\frac{2\pi(2m'+1)n}{2N'}}$$

$$= \sum_{m'=0}^{N'-1} f_{2m'} e^{i\frac{2\pi(2m')n}{2N'}} + e^{i\frac{2\pi n}{2N'}} \sum_{m'=0}^{N'-1} f_{2m'+1} e^{i\frac{2\pi(2m')n}{2N'}}$$

$$= \sum_{m'=0}^{N'-1} f_{e,m'} e^{i\frac{2\pi m'n}{N'}} + e^{i\frac{2\pi n}{N}} \sum_{m'=0}^{N'-1} f_{o,m'} e^{i\frac{2\pi m'n}{N'}} \qquad (n = 0, 1, \dots, N-1)$$

# 高速フーリエ変換 (2)

$$\tilde{f}_n = \sum_{m\prime=0}^{N\prime-1} f_{e,m\prime} e^{i\frac{2\pi m\prime n}{N\prime}} + e^{i\frac{2\pi n}{N}} \sum_{m\prime=0}^{N\prime-1} f_{o,m\prime} e^{i\frac{2\pi m\prime n}{N\prime}} \qquad (n = 0, 1, \dots, N-1)$$

ここで,

$$e^{i\frac{2\pi m\prime}{N\prime}\left(n+\frac{2N\prime}{2}\right)} = e^{i\frac{2\pi m\prime}{N\prime}n}$$
$$e^{i\frac{2\pi}{N}\left(n+\frac{N}{2}\right)} = -e^{i\frac{2\pi}{N}n}$$

であることを考慮すると,

$$\tilde{f}_n = \sum_{m'=0}^{N'-1} f_{e,m'} e^{i\frac{2\pi m' n}{N'}} + e^{i\frac{2\pi n}{N}} \sum_{m'=0}^{N'-1} f_{o,m'} e^{i\frac{2\pi m' n}{N'}} = \tilde{f}_{e,n} + e^{i\frac{2\pi n}{N}} \tilde{f}_{o,n} \quad \left(n = 0,1, \dots, \frac{N}{2} - 1\right)$$

$$\tilde{f}_{n+\frac{N}{2}} = \sum_{m'=0}^{N'-1} f_{e,m'} e^{i\frac{2\pi m'}{N'}\left(n+\frac{N}{2}\right)} + e^{i\frac{2\pi}{N}\left(n+\frac{N}{2}\right)} \sum_{m'=0}^{N'-1} f_{o,m'} e^{i\frac{2\pi m'}{N'}\left(n+\frac{N}{2}\right)}$$

$$= \sum_{m'=0}^{N'-1} f_{e,m'} e^{i\frac{2\pi m' n}{N'}} - e^{i\frac{2\pi n}{N}} \sum_{m'=0}^{N'-1} f_{o,m'} e^{i\frac{2\pi m' n}{N'}} = \tilde{f}_{e,n} - e^{i\frac{2\pi n}{N}} \tilde{f}_{o,n} \quad \left(n = 0, 1, \dots, \frac{N}{2} - 1\right)$$

となる.

# 高速フーリエ変換 (3)

つまり, データ数 N のフーリエ変換は, データ数 N’=N/2 のフーリエ変換二つ

$$\tilde{f}_{e,n} = \sum_{m\prime=0}^{N\prime-1} f_{e,m\prime} e^{i\frac{2\pi m\prime n}{N\prime}}, \qquad \tilde{f}_{o,n} = \sum_{m\prime=0}^{N\prime-1} f_{o,m\prime} e^{i\frac{2\pi m\prime n}{N\prime}} \quad \left(n = 0, 1, \ldots, \frac{N}{2} - 1\right)$$

を使って計算できる.

さらにデータ数 N/2 のフーリエ変換は, データ数 N/4 のフーリエ変換二つを使って計算できる.
さらにデータ数 N/4 のフーリエ変換は, データ数 N/8 のフーリエ変換二つを使って計算できる.
…
さらにデータ数 2 のフーリエ変換は, データ数 1 のフーリエ変換二つを使って計算できる.
データ数 1 のフーリエ変換は下のようになる.

$$\tilde{f}_0 = \sum_{m=0}^{0} f_m e^{i\frac{2\pi mn}{2}} = f_0$$

# 高速フーリエ変換の演算量 (1)

- データ数 N の離散フーリエ変換における演算量（ここでは掛け算の数に注目する）は, N^2 である.
- ここで, データ数 N の離散フーリエ変換の演算量を NFFT(N) と表すことにすると, 高速フーリエ変換のアルゴリズムでは下の関係が成り立つ.
  - NFFT(N) = 2 * NFFT(N/2) + N/2
  - ただし, NFFT(1) = 0 である.
- これより,
  - NFFT(2^n) = 2 * NFFT(2^(n-1)) + 2^(n-1)
  - これより,
    - n = 1; NFFT(2) = 2 * NFFT(1) + 1 = 1
    - n = 2; NFFT(4) = 2 * NFFT(2) + 2 = 2 * 1 + 2 = 4
    - n = 3; NFFT(8) = 2 * NFFT(4) + 4 = 2 * 4 + 4 = 12
    - n = 4; NFFT(16) = 2 * NFFT(8) + 8 = 2 * 12 + 8 = 32
    - n = 5; NFFT(32) = 2 * NFFT(16) + 16 = 2 * 32 + 16 = 80
    - …
  - これは, 一般化すると, n >= 2 のとき, 下のように表される.
    - NFFT(N) = N/2 log2N = 2^(n-1) log2 2^n = 2^(n-1) * n

# 高速フーリエ変換の演算量 (2)

- あるいは,

$$NFFT(N) = 2NFFT\left(\frac{N}{2}\right) + N$$

- より,

$$\begin{aligned}
NFFT\left(2^n\right) &= 2NFFT\left(2^{n-1}\right) + 2^n \\
&= 2\left\{2NFFT\left(2^{n-2}\right) + 2^{n-1}\right\} + 2^n \\
&= 2^2 NFFT\left(2^{n-2}\right) + 2 \cdot 2^n \\
&= 2^2\left\{2NFFT\left(2^{n-3}\right) + 2^{n-2}\right\} + 2 \cdot 2^n \\
&= 2^3 NFFT\left(2^{n-3}\right) + 3 \cdot 2^n \\
&= \cdots \\
&= 2^n NFFT\left(1\right) + n \cdot 2^n \\
&= n \cdot 2^n \\
&= \log_2 2^n \cdot N
\end{aligned}$$

# 高速フーリエ変換プログラムの概要

- 入力
  - データ(複素数)配列 1 つ(or 実数配列 2 つ)　: $f_m \ (m = 0, 1, \dots, N-1)$
- 出力
  - データ(複素数)配列 1 つ(or 実数配列 2 つ)　: $\tilde{f}_n \ (n = 0,1, \dots, N-1)$
- 処理手順
  - 入力データ数が 1 のとき
    - 入力データを出力
      - $\tilde{f}_n = f_m \ (n = m = 0,1, \dots, N-1)$
  - 入力のデータ数が 2 以上のとき
    - 入力データ配列を 2 つに分割
      - 奇数番目データの配列　: $f_{o,m} = f_{2m} \ \left(m = 0,1, \dots, \frac{N}{2} - 1\right)$
      - 偶数番目データの配列　: $f_{e,m} = f_{2m+1} \ \left(m = 0,1, \dots, \frac{N}{2} - 1\right)$
    - 分割した配列を引数として高速フーリエ変換プログラムを呼び出す.
    - データ数 N/2 のフーリエ変換の出力, $\tilde{f}_{e,n}, \tilde{f}_{o,n} \ \left(n = 0,1, \dots, \frac{N}{2} - 1\right)$ を用いて, データ数 N のフーリエ変換の出力, $\tilde{f}_n \ (n = 0,1, \dots, N-1)$, を計算.
      - $\tilde{f}_n = \tilde{f}_{e,n} + e^{i\frac{2\pi n}{N}} \tilde{f}_{o,n} \ \left(n = 0,1, \dots, \frac{N}{2} - 1\right)$
      - $\tilde{f}_{n+\frac{N}{2}} = \tilde{f}_{e,n} - e^{i\frac{2\pi n}{N}} \tilde{f}_{o,n} \ \left(n = 0, 1, \dots, \frac{N}{2} - 1\right)$

# Fortran90 再帰的手続き

- Fortran77 では手続きを再帰的に呼び出すことができなかった.
- Fortran90 では可能である.
- プログラム例を右に示す. このプログラムは, 再帰的呼び出しを用いて, 6! を計算する.

```
program main
  implicit none
  integer :: Num

  Num = 1
  call factorial( 6, Num )

  write( 6, * ) Num
end program main

recursive subroutine factorial( n, Num )
  implicit none
  integer, intent(in) :: n
  integer, intent(inout) :: Num

  Num = Num * n
  if ( n > 1 ) then
    call factorial( n-1, Num )
  end if
end subroutine factorial
```

# 練習問題

- 高速フーリエ変換のプログラムを作ってみよう.

# 2 次元フーリエ変換

- 2 回やれば良い.

# 練習問題

- 2 次元フーリエ変換のプログラムを作ってみよう.

# 参考文献・ページ

- http://www.mk.ecei.tohoku.ac.jp/jspmatlab/pdf/matdsp2.pdf