離散フーリエ変換

数値計算に向けた解説書

シキノ *

June 18, 2022

Contents

*https://slpr.sakura.ne.jp/qp/, https://twitter.com/sikinote

1 まとめ

フーリエ変換

$$\begin{cases} f(k) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi kx} dx & \text{(1.1a)} \\ f(x) = \int_{-\infty}^{\infty} f(k) e^{i2\pi kx} dk & \text{(1.1b)} \end{cases}$$

位置

$$x = [-\infty, \infty] \quad \text{(1.2a)}$$

波数

$$k = [-\infty, \infty] \quad \text{(1.3a)}$$

畳み込み

$$\begin{aligned} (f * g)(x) &\equiv \int_{-\infty}^{\infty} f(\tau) g(x - \tau) d\tau & \text{(1.4a)} \\ (f * g)(k) &\equiv \mathcal{F}[(f * g)(x)](k) \\ &= f(k) g(k) & \text{(1.4b)} \end{aligned}$$

離散フーリエ変換

$$\begin{cases} f(k_n) = \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} & \text{(1.5a)} \\ f(x_m) = \frac{1}{N} \sum_{n=0}^{N-1} f(k_n) e^{i2\pi nm/N} & \text{(1.5b)} \end{cases}$$

位置

$$x = [0, X] \quad \text{(1.6a)}$$

$$x_m = m\Delta x, (m = 0, 1, \cdots, N-1) \quad \text{(1.6b)}$$

$$\Delta x = \frac{X}{N} = \frac{1}{K} \quad \text{(1.6c)}$$

波数

$$k = [0, K], \quad \text{(1.7a)}$$

$$k_n = n\Delta k, \ (n = 0, 1, \cdots, N-1) \quad \text{(1.7b)}$$

$$\Delta k = \frac{K}{N} = \frac{1}{X}, \quad \text{(1.7c)}$$

$$\Delta x \Delta k = \frac{1}{N}, \quad KX = N \quad \text{(1.8)}$$

$$f(x_m \pm X) = f(x_m), \ f(k_m \pm K) = f(k_n) \quad \text{(1.9)}$$

畳み込み

$$\begin{aligned} (f * g)_{\mathrm{d}}(x_n) &\equiv \sum_l f(x_l) g(x_n - x_l) & \text{(1.10a)} \\ (f * g)_{\mathrm{d}}(k_n) &\equiv \mathcal{F}[(f * g)_{\mathrm{d}}(x)](k_n) \\ &= f(k_n) g(k_n) & \text{(1.10b)} \end{aligned}$$

2 連続の場合

2.1 フーリエ変換 (Fourier-Transform, FT)

2.1.1 本稿のフーリエ変換の定義

本稿ではフーリエ変換を以下の演算と定義します[1].

$$\begin{cases} f(k) = \displaystyle\int_{-\infty}^{\infty} f(x) e^{-i2\pi kx} dx & \text{(2.1a)} \\ f(x) = \displaystyle\int_{-\infty}^{\infty} f(k) e^{i2\pi kx} dk & \text{(2.1b)} \end{cases}$$

ここで, Eq. (2.1a) を順方向フーリエ変換, Eq. (2.1b) を逆方向フーリエ変換と呼びます. 多くの場合で x は位置（時間）, k は波数（周波数）と呼ばれます.

フーリエ変換は右辺の積分を省略して

$$\begin{cases} f(k) = \mathcal{F}\left[f(x)\right](k) & \text{(2.2a)} \\ f(x) = \mathcal{F}^{-1}\left[f(k)\right](x) & \text{(2.2b)} \end{cases}$$

と演算子 $\mathcal{F}, \mathcal{F}^{-1}$ を用いて表記したりします.

フーリエ変換は, 基底関数 $\varphi_k(x) = e^{i2\pi kx}$ への射影と考えることができます. つまり Eq. (2.1b) とは,

『適当な関数 $f(x)$ があるとき, $f(x)$ を $\varphi_k(x)$ の $f(k)$ 倍の和（積分）で書いてください.』

という問題なわけです[2]. ここで, $\varphi_k(x)$ にかかる係数は x に依存しない k の関数 $f(k)$ であり, それは Eq. (2.1a) から求められる, という操作がフーリエ変換なわけです.

[1] フーリエ変換の定義は分野によって異なります. 基本的には, その分野で何が目的を元に決められます. 数値計算の分野では計算量が少なくなることが素晴らしいことので, 積分記号の前に余計な係数が付かないこと等から, 指数関数の肩に 2π が付けられたものが使われます. 物理学（特に量子力学）では, 角周波数で表現すると式の見た目が良くなるので指数関数の肩に 2π が付かないものが使われます.

[2] なかなか積分は和と同じということをイメージするのは慣れが必要なので, 人によっては離散フーリエ変換 (節 3.1) の方がイメージしやすいかもしれません.

2.1.2 フーリエ変換に関係する簡単な式と特徴

- **逆方向フーリエ変換から順方向フーリエ変換を導く**

  逆方向フーリエ変換の結果だけから, 順方向フーリエ変換の結果を導くことができます[3]. Eq. (2.1b) の左から $e^{-i2\pi k'x}$ を掛けて, そのあと全位置空間 x で積分すれば, $f(k)$ について導くことができます. つまり,

$$\begin{aligned}
\int_{-\infty}^{\infty} dx e^{-i2\pi k'x} f(x) &= \int_{-\infty}^{\infty} dx e^{-i2\pi k'x} \cdot \int_{-\infty}^{\infty} dk f(k) e^{i2\pi kx} && \text{(2.3a)}\\
&= \int_{-\infty}^{\infty} dk f(k) \int_{-\infty}^{\infty} dx e^{i2\pi(k-k')x} && \text{(2.3b)}\\
&= \int_{-\infty}^{\infty} dk f(k)\delta(k-k') && \text{(2.3c)}\\
&= f(k') && \text{(2.3d)}
\end{aligned}$$

  となります.k' を改めて k と書けば,

$$f(k) = \int_{-\infty}^{\infty} dx e^{-i2\pi kx} f(x) \tag{2.3e}$$

  となり, 順方向フーリエ変換が導けます.

- **フーリエ変換はユニタリー変換**

  フーリエ変換の重要な特徴の一つはユニタリー変換であることです. つまり, フーリエ変換して逆変換すると元の関数に戻ります. 式で示せば,

$$\mathcal{F}^{-1}\mathcal{F}f(x) = f(x), \quad \rightarrow \quad \mathcal{F}^{-1}\mathcal{F} = 1 \tag{2.4}$$

  ということです. 実際に考えてみますと

$$\begin{aligned}
\mathcal{F}^{-1}\mathcal{F}f(x) &= \mathcal{F}^{-1} \int_{-\infty}^{\infty} dx' f(x') e^{-i2\pi kx'} && \text{(2.5a)}\\
&= \int_{-\infty}^{\infty} dk \left[\int_{-\infty}^{\infty} dx' f(x') e^{-i2\pi kx'} \right] e^{i2\pi kx} && \text{(2.5b)}\\
&= \int_{-\infty}^{\infty} dx' f(x') \int_{-\infty}^{\infty} dk' e^{i2\pi k(x-x')} && \text{(2.5c)}\\
&= \int_{-\infty}^{\infty} dx' f(x')\delta(x-x') && \text{(2.5d)}\\
&= f(x) && \text{(2.5e)}
\end{aligned}$$

  となります.

[3] 順方向から逆方向を導いてもどちらでも構いません

2.1.3 フーリエ変換の例

実際にフーリエ変換を Eq. (2.1) に従って実行してみましょう.

$$f(x) = \cos(2\pi 3x) \tag{2.6}$$

という, 波数 3 を持つ波をフーリエ変換してみます. フーリエ変換すると波数の成分が出てくるはずなので, 波数 3 が答えとして出てくることを期待します.

順方向フーリエ変換を行うと [4],

$$\begin{aligned} f(k) &= \int_{-\infty}^{\infty} dx \cos(2\pi 3x) e^{-i2\pi kx} && \text{(2.7a)} \\ &= \int_{-\infty}^{\infty} dx \frac{e^{i2\pi 3x} + e^{-i2\pi 3x}}{2} e^{-i2\pi kx} && \text{(2.7b)} \\ &= \frac{1}{2} \int_{-\infty}^{\infty} dx \left[e^{i2\pi(3-k)x} + e^{-i2\pi(3+k)x} \right] && \text{(2.7c)} \\ &= \frac{1}{2} \left[\delta(k-3) + \delta(k+3) \right] && \text{(2.7d)} \end{aligned}$$

と求められます. つまり, $\cos(2\pi 3x)$ をフーリエ変換したら $k = \pm 3$ に正の振幅を持ったピークが出てくる, ということです. フーリエ変換によって, 波形の波数に値が出てくる, ということが確認できました.

[4] 積分の文字 dx の順番を積分の直後に置いておりますが, 同じ意味です. 何個も積分記号が混在すると, どの変数がどの範囲か分からなくなるため, 積分記号の直後に置くことにしています.

2.2 畳み込み (Convolution)

2.2.1 畳み込みの定義

畳み込み（もしくは畳み込み積分）とは, 関数 $f(x), g(x)$ が定義されているときに以下の積分

$$(f*g)(x) \equiv \int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau \tag{2.8a}$$

$$= \int_{-\infty}^{\infty} f(x-\tau)g(\tau)d\tau \tag{2.8b}$$

を計算することです. 関数 f と関数 g の畳み込み, と言います.

畳み込みとは『関数と関数を混ぜ込んだ結果』みたいなイメージです. Eq. (2.8a) から Eq. (2.8b) へは, $x-\tau$ を新たな変数として変数変換すれば導けます. τ の範囲は $(-\infty, \infty)$ が基本であり本稿はそれに倣いますが, 問題によっては有限範囲や 1 周期を考える場合もあります.

2.2.2 畳み込みとフーリエ変換

畳み込み積分 (2.8a) をフーリエ変換された関数で表現してみましょう [5].

条件と問題を整理すると以下の通りになります.

$$(f*g)(x) = \int d\tau f(\tau)g(x-\tau) \tag{2.9a}$$

$$f(x) = \int_{-\infty}^{\infty} dk f(k)e^{i2\pi kx} \tag{2.9b}$$

$$g(x) = \int_{-\infty}^{\infty} dk g(k)e^{i2\pi kx} \tag{2.9c}$$

Eq. (2.9b), (2.9c) を Eq. (2.9a) に代入すれば,

$$(f*g)(x) = \int_{-\infty}^{\infty} d\tau f(\tau)g(x-\tau) \tag{2.10a}$$

$$= \int d\tau \left[\int_{-\infty}^{\infty} dk f(k)e^{i2\pi k\tau}\right]\left[\int_{-\infty}^{\infty} dk' g(k')e^{i2\pi k'(x-\tau)}\right] \tag{2.10b}$$

$$= \int_{-\infty}^{\infty} dk \int_{-\infty}^{\infty} dk' \left[f(k)g(k')e^{i2\pi k'x}\int d\tau e^{i2\pi(k-k')\tau}\right] \tag{2.10c}$$

$$= \int_{-\infty}^{\infty} dk \int_{-\infty}^{\infty} dk' \left[f(k)g(k')e^{i2\pi k'x}\delta(k-k')\right] \tag{2.10d}$$

$$= \int_{-\infty}^{\infty} dk f(k)g(k)e^{i2\pi kx} \tag{2.10e}$$

[5] なぜフーリエ変換しようと思ったかは聞かないでください. 先人が畳み込みを計算しようと思って考えついて, それが綺麗だった, 程度の理由だと思います.

と簡単になります. つまり, $e^{i2\pi kx}$ の前にかかる係数がフーリエ変換結果ですので, 畳み込みの結果をフーリエ変換すると,

$$\mathcal{F}[(f*g)(x)](k) = \int_{-\infty}^{\infty}(f*g)(x)e^{-i2\pi kx}dx \tag{2.11a}$$
$$= f(k)g(k) \tag{2.11b}$$

となります.

以上より『畳み込みの計算結果をフーリエ変換したい！』となったら, 畳み込みに用いる二つの関数をそれぞれ独立にフーリエ変換し, フーリエ変換結果をかけ合わせればよい, ということです.

もしくは, 畳み込みの計算はその逆方向フーリエ変換なので

$$(f*g)(x) = \int d\tau f(\tau)g(x-\tau) \tag{2.12a}$$
$$= \mathcal{F}^{-1}[f(k)g(k)](x) \tag{2.12b}$$

で計算できます.

2.2.3 畳み込みの例

ここでは畳み込みの例を考えてみます. 2つの関数

$$f(x) = x^5e^{-x^2} \tag{2.13}$$
$$g(x) = e^{-4x^2} \tag{2.14}$$

の畳み込みを考えてみます.Eq. (2.8a) に代入してそのまま積分を計算すると,

$$h(x) = \int f(x)\cdot g(x-\tau)d\tau \tag{2.15a}$$
$$= \int \tau^5e^{-\tau^2}\cdot e^{-4(x-\tau)^2}d\tau \tag{2.15b}$$
$$= \frac{1}{3125}\sqrt{\frac{\pi}{5}}x(1024x^4+1600x^2+375)e^{-\frac{4}{5}x^2} \tag{2.15c}$$

となります.

一方で, それぞれの関数のフーリエ変換を掛け合わせて逆変換を行う Eq. (2.11), (2.12a) に従って求めてみたいと思います. Eq. (2.13) のそれぞれのフーリエ変換は,

$$f(k) = -\frac{i\pi^{3/2}}{4}(4\pi^4k^5-20\pi^2k^3+15k)e^{-\pi^2k^2} \tag{2.16}$$
$$g(k) = \frac{1}{2}\sqrt{\pi}e^{-\pi^2k^2/4} \tag{2.17}$$

より [6],

$$h(k) = f(k)g(k) \tag{2.18}$$

を逆変換して,

$$\int_{-\infty}^{\infty} dk h(k) e^{i2\pi kx} \tag{2.19a}$$

$$= -\frac{i\pi^{3/2}}{4}\left(\frac{64ix(64x^4 - 400x^2 + 375)}{3125\sqrt{5}\pi}e^{-4x^2/5} + \frac{32ix(8x^2 - 15)}{25\sqrt{5}\pi}e^{-4x^2/5} + \frac{12ix}{\sqrt{5}\pi}e^{-4x^2/5}\right) \tag{2.19b}$$

$$= \frac{1}{3125}\sqrt{\frac{\pi}{5}}x\left(1024x^4 + 1600x^2 + 375\right)e^{-4x^2/5} \tag{2.19c}$$

と求められ, Eq. (2.15c) と一致することが確認できます.

[6] $f(x)$ のフーリエ変換 https://www.wolframalpha.com/input?i=integral+from+-infty+to+infty+of+x%5E5+exp%28-x%5E2%29*exp%28-i*2*pi*k*x%29
$g(x)$ のフーリエ変換 https://www.wolframalpha.com/input?i=integral+from+-infty+to+infty+of+exp%28-4*x%5E2%29*exp%28-i*2*pi*k*x%29

3 離散の場合

3.1 離散フーリエ変換 (Discrete Fourier-Transform, DFT)

離散フーリエ変換とは, 位置・波数の両方が有限区間に制限された中で行われるフーリエ変換と考えて良いです.

つまり, 適当な離散的な基底関数 $\varphi_n(x), (n = 0, 1, \cdots)$ を使用し, 離散的な点 $x = x_m, (m = 0, 1, \cdots)$ 上の値を用いて関数 $f(x_m)$ を

$$f(x_m) = \sum_n f(k_n)\varphi_n(x_m) \tag{3.1}$$

として書いたときに, $f(x_m)$ と $\varphi_n(x_m)$ が既知で, 係数 $f(k_n)$ が未知の時, これを求めたいというのが離散フーリエ変換です. 本節の出発点は Eq. (2.1) とし, 順に離散的な表現に移行していきます. 順方向離散フーリエ変換は, ユニタリー性を保つために係数倍の自由度を指定する必要がありますので, 後程の項 3.1.3 で説明します.

本で離散フーリエ変換を知りたい方は, [2] などが良いかと思います.

3.1.1 位置を無限区間から有限区間へ制限 → 波数の刻み幅 Δk の決定

まずは位置だけを区間 $[x_a, x_b], (x_b > x_a)$ に制限して, その他の区間では $[x_a, x_b]$ が滑らかに繋がるように周期的に存在していると仮定します.

連続の場合の類推から, 離散フーリエ変換は基底関数

$$\varphi_n(x) = e^{i2\pi k_n(x - x_{\text{off}})} \tag{3.2}$$

への射影となります[7]. ここで x_{off} は基底関数を平行移動させる任意の値 ($x_{\text{off}} = 0$ でも, $x_{\text{off}} = x_a$ でも OK), 波数 k_n は離散的で,

$$k_n = n \cdot \Delta k, \ (n = 0, \pm 1, \pm 2, \cdots) \tag{3.3}$$

$$\Delta k \equiv \frac{1}{x_b - x_a} \tag{3.4}$$

だけが存在できます. ここで, Δk は波数の間隔です. k が離散的であることと Δk の値はどこから決まったかというと, 位置 $[a, b]$ で周期的であるという条件から決められます. つまり, 区間の端で滑らかにつながるための周期境界条件

$$\begin{cases} \varphi_n(x_a) = \varphi_n(x_b) & \text{(3.5a)} \\ \left.\dfrac{d\varphi_n(x)}{dx}\right|_{x=x_a} = \left.\dfrac{d\varphi_n(x)}{dx}\right|_{x=x_b} & \text{(3.5b)} \end{cases}$$

[7] x_{off} の off は offset の意味で使用しています.

を満たすような波数 $k = k_n$ しか存在しないと考える訳です. この答えが Eq. (3.3) になる訳です.

このままでも良いのですが, 位置だけ制限してなぜ波数は無限のままにするのか？という疑問が出てきます[8]. なので波数も有限の区間で考えましょう.

3.1.2 波数を無限区間から有限区間へ制限 → 位置の刻み幅 Δx の決定

基底関数 (3.2) に波数の最小値・最大値を制限した場合, どのような制限が掛かるか考えてみましょう. Eq. (3.3) の制限から, 波数 k の最小値を $k_{\min}$, 最大値を $k_{\max}$ とします. 式で表せば,

$$k_{\min} = N_{\min}\Delta k \tag{3.6a}$$

$$k_{\max} = N_{\max}\Delta k \tag{3.6b}$$

です. ここで, $N_{\min}$, $N_{\max}$ はそれぞれの波数に対応するインデックスで $N_{\max} > N_{\min}$ です. 波数 k の取り得る範囲とその範囲の大きさ k_{range} は

$$k = [k_{\min},\ k_{\max}] = [N_{\min}\Delta k, N_{\max}\Delta k] \tag{3.7}$$

$$k_{\mathrm{range}} \equiv k_{\max} - k_{\min} = (N_{\max} - N_{\min})\Delta k \tag{3.8}$$

となります.

波数の区間を制限すると, その範囲外の波数を持つ関数は範囲内に強制的に収められてしまいます. つまり, 範囲外の波数を持つ基底関数と範囲内の波数を持つ基底関数とが区別されなくなることを意味します. 式で表せば,

$$e^{i2\pi k_n(x-x_{\mathrm{off}})} = e^{i2\pi(k_n+K)(x-x_{\mathrm{off}})}, \tag{3.9}$$

ただし, $x = [x_a, x_b]$ で, $k_n + K$ は Eq. (3.7) の範囲外

となってしまう定数 K が存在する, ということです. この K の値はいつも Eq. (3.7) の範囲外であるという条件から,

$$K = (N_{\max} - N_{\min})\Delta k \tag{3.10}$$

[8] 対称性が悪い, とも言いますね. ある関数は位置で考え無ければならない, という制限はどこにもないのです. 多くの人が扱いやすいと感じるだけだから位置に注目しているだけであって, ある関数を波数で考えてもいいじゃないかという人もいるのです. どちらも優劣が無いので, 片方を制限したらもう片方も制限しない理由が特に無いのです. 逆に, わざと片方を連続のまま扱うことも可能で, 実際に量子力学の分野では利用されてたりします.

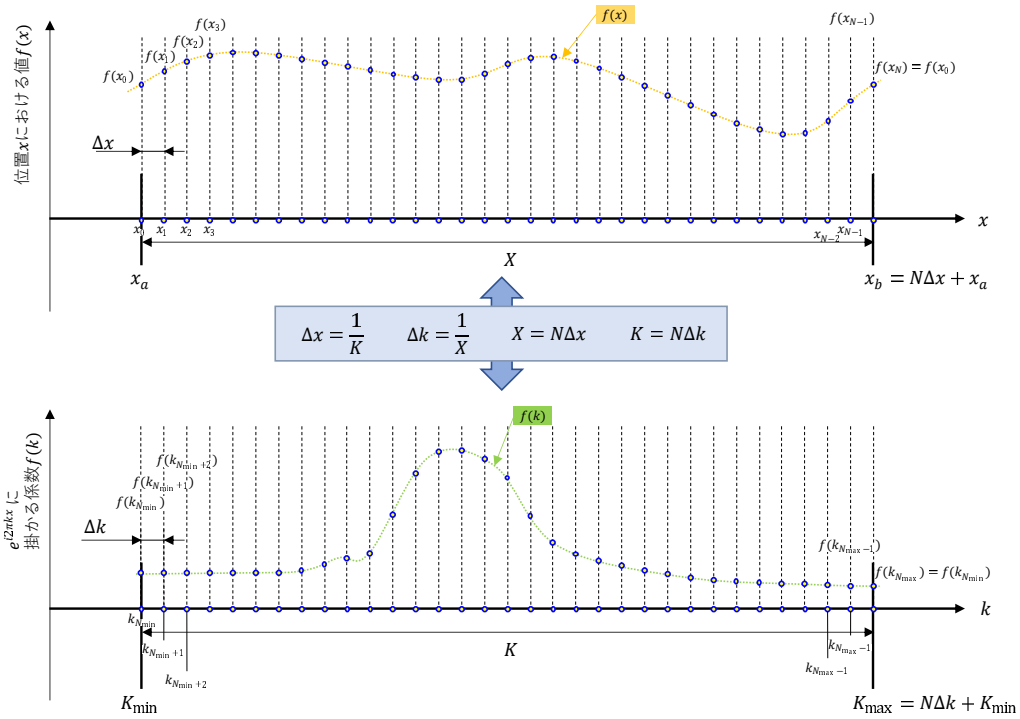

Figure 1:　項 3.1.1, 3.1.2 で定義した分点の位置, 関係式等の図示.

となることは想像に難くないでしょう. 区間より大きくとって $K > (N_{\max} - N_{\min})\Delta k$ としても良いですが, 大きくとった際のメリットは特にないので通常は等号が成り立つときを K とします.

さて, Eq. (3.10) のよう K を決定すると, 具体的に k_n のインデックス n の最大値を決めることができます. 安直に考えれば, $n = N_{\min}, N_{\min}+1, \cdots, N_{\max}-1, N_{\max}$ ですが, 端の $N_{\min}$ と $N_{\max}$ に置ける関数値は周期境界条件 (3.9) により

$$f(N_{\min}\Delta k) = f(N_{\max}\Delta k + K) \tag{3.11}$$

が成立しています. なので, 実際のインデックスは $N_{\min}$ から $N_{\max}-1$ までで打ち止めとなります. つまり, 波数は

$$k_n = n\Delta k + l \cdot K, \tag{3.12a}$$

$$(n = N_{\min}, N_{\min}+1, \cdots, N_{\max}-1) \tag{3.12b}$$

となるわけです. ここで, 波数は K の整数倍 $l, (l = 0, \pm 1, \cdots)$ だけずれても良いので, それを考慮しています.

また Eq. (3.9) より

$$e^{i2\pi K(x-x_{\text{off}})} = 1 \ \rightarrow \ K(x - x_{\text{off}}) = m, \ (m \text{ は整数}) \tag{3.13}$$

の関係式がいつでも成立していなければなりません．K, m は既に決まっているため, Eq. (3.13) は x に対して制限を加える条件になっています．つまり, $x - x_{\mathrm{off}}$ はいつでも $1/K(\equiv \Delta x)$ の整数倍になっていなければならない, という条件です. 式で表せば,

$$x_m = m\Delta x + x_{\mathrm{off}},\ (m = 0, 1, \cdots, N_{\mathrm{max}} - N_{\mathrm{min}} - 1) \tag{3.14a}$$

$$\Delta x = \frac{1}{K} = \frac{x_b - x_a}{N_{\mathrm{max}} - N_{\mathrm{min}}} \tag{3.14b}$$

となります. ここで, $N_{\mathrm{max}} - N_{\mathrm{min}} - 1$ の”-1” が入っているのは, $m = 0$ と $m = N_{\mathrm{max}} - N_{\mathrm{min}}$ が周期境界条件からいつも同じ値になるため, 除いても問題ないためです．また K を Eq. (3.10) のように決定したことにより, $x_{\mathrm{off}} = x_a$ の制限が付きました．$m = 0$ の時に $x_0 = x_{\mathrm{off}}$ となり, x_0 は範囲の左端である x_a に等しいからです.

以上で求めた関係について, Fig. 1 に示しましたので参考にしてください.

3.1.3 順方向/逆方向離散フーリエ変換の表記

フーリエ級数 $f(k_n)$ による $f(x_m)$ の表現は Eq. (3.1) のように書けます. そこでは具体的な点数などは決めていませんでしたが, ここまで来てようやくそれらを指定することができました.

逆方向離散フーリエ変換は Eq. (3.1) の通りで, 分点の位置や数は Eq. (3.12b) に従います. つまり,

$$f(x_m) = \sum_{n=N_{\min}}^{N_{\max}-1} f(k_n) e^{i2\pi k_n (x_m - x_a)} \tag{3.15a}$$

となります. 先に述べた x_{off} は x_a に等しいという事実を用いました. 順方向離散フーリエ変換を求めるためには, 連続フーリエ変換を求めた際に使用した Eq. (2.5) を真似すればよいです. Eq. (3.15) の両辺から $e^{-i2\pi k_{n'}(x_m - x_a)}$ を掛けて m に対して和を取ってみます. すると,

$$\sum_{m=0}^{N-1} e^{-i2\pi k_{n'}(x_m - x_a)} \cdot f(x_m) = N \cdot f(k_{n'}) \tag{3.16}$$

となり, 順方向離散フーリエ変換が導かれます. 計算過程は Appendix A.2 を参照してください.

以上から, 順方向/逆方向離散フーリエ変換は

$$\begin{cases} f(k_n) = \dfrac{1}{N} \displaystyle\sum_{m=0}^{N-1} f(x_m) e^{-i2\pi k_n x_m} & \text{(3.17a)} \\ f(x_m) = \displaystyle\sum_{n=N_{\min}}^{N_{\max}-1} f(k_n) e^{i2\pi k_n x_m} & \text{(3.17b)} \end{cases}$$

となります.

連続フーリエ変換 (2.1) とは異なり係数 $1/N$ が存在し, 非常に, 非常に気持ち悪いです. ですが, この $\frac{1}{N}$ が存在しないとフーリエ変換のユニタリー性が保たれなくなります. 式で書くなら次の通りです.

$$\mathcal{F}'^{-1}\mathcal{F}' f(x) = N f(x) \tag{3.18}$$

ここで, $\mathcal{F}', \mathcal{F}'^{-1}$ は Eq. (3.17) で $1/N$ が存在しない順方向, 逆方向離散フーリエ変換です. これの右辺に入る N を出てこないようにするために両辺に $1/N$ を掛け, この定数が離散フーリエ変換の定義に入り込んできます. そのため, $1/N$ は必ず順方向に入る訳ではなく, 逆方向の方に入っても良いですし, 両方に入っても良いです.

この係数 $1/N$ と指数関数の肩に 2π を含めるか否かの派閥のせいで, 本によって離散フーリエ変換の定義がバラバラです. 同じ著者でも気を付けてください. いくつか例を挙げてみましょう.

例 a) **対称性を重視する形式**

例えば数学的に最もきれいな形にしようとすれば対称性を考えて

$$\begin{cases} f(k_n) = \dfrac{1}{\sqrt{N}} \displaystyle\sum_{m=0}^{N-1} f(x_m) e^{-i2\pi k_n x_m} & \text{(3.19a)} \\ f(x_m) = \dfrac{1}{\sqrt{N}} \displaystyle\sum_{n=N_{\min}}^{N_{\max}-1} f(k_n) e^{i2\pi k_n x_m} & \text{(3.19b)} \end{cases}$$

が良いでしょう. ですがどうもなかなか好まれないようです. 理由は様々ですが, ルートを書くのが面倒, プログラムの出力を見た際に分かりにくいというのがあります. フーリエ変換のテストとして, $f(x_m) = e^{i2\pi k_0 x}$ を実行してみると, Eq. (3.19a) の解は $f(k_0) = \sqrt{N}$ となります. 例えば $N = 5000$ とかで実行したら $\sqrt{5000}$ の答えは計算しなきゃいけません. ですが $1/\sqrt{N}$ が無ければ, そのまま $f(k_0) = 5000$ が値として出て嬉しいため, $1/\sqrt{N}$ はなかなか採用され無いのかな…と思います.

例 b) **連続フーリエ変換との対応を重視する形式**

連続フーリエ変換の形と見比べると, 以下の形式でも良いことが分かります.

$$\begin{cases} f(k_n) = \displaystyle\sum_{m=0}^{N-1} f(x_m) e^{-i2\pi k_n x_m} \Delta x & \text{(3.20a)} \\ f(x_m) = \displaystyle\sum_{n=N_{\min}}^{N_{\max}-1} f(k_n) e^{i2\pi k_n x_m} \Delta k & \text{(3.20b)} \end{cases}$$

これは, フーリエ変換の積分を区分求積のように考えて離散フーリエ変換にしているように見えます. この書き方は連続フーリエ変換の答えがそのまま離散フーリエ変換の答えとなり, 係数倍の違いも生じません. つまり, $\Delta x \Delta k = 1/N$ が成立するため, これも意味がある分け方です. そのため連続フーリエ変換の純粋な離散表現を行いたい場合はこれが良いでしょう. しかし, マイナーです. 好んで使う人はなかなかいない気がします (著者はこの表記が大好きです).

例 c) **可読性を重視する形式**

もっともよく使用されるのが逆方向に $\frac{1}{N}$ を入れる方法です.

$$\begin{cases} f(k_n) = \displaystyle\sum_{m=0}^{N-1} f(x_m) e^{-i2\pi k_n x_m} & \text{(3.21a)} \\ f(x_m) = \dfrac{1}{N} \displaystyle\sum_{n=N_{\min}}^{N_{\max}-1} f(k_n) e^{i2\pi k_n x_m} & \text{(3.21b)} \end{cases}$$

順方向をできるだけ簡単に表記しつつ, 逆方向に全てを押し付けた形ですね. 次の項以降でもこれを考えて行こうと思います.

3.1.4 離散フーリエ変換の簡略化

離散フーリエ変換に制限を加えることで, 表記がもう少し簡単になります. まずはこれまでの結果をまとめてみましょう. 前節 3.1.1, 3.1.2 の結果より, 離散フーリエ変換に関係する式は次の Eq. (3.22) から Eq. (3.27) の通りとなります.

1. 離散フーリエ変換

$$\begin{cases} f(k_n) = \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi k_n x_m} & \text{(3.22a)} \\ f(x_m) = \frac{1}{N} \sum_{n=N_{\min}}^{N_{\max}-1} f(k_n) e^{i2\pi k_n x_m} & \text{(3.22b)} \end{cases}$$

2. 位置　区間 $[x_a, x_b]$

$$x_m = m\Delta x + x_a, \ (m = 0, 1, \cdots, N-1) \quad \text{(3.23a)}$$

$$\Delta x = \frac{1}{K} = \frac{x_b - x_a}{N} \quad \text{(3.23b)}$$

$$N \equiv N_{\max} - N_{\min} \quad \text{(3.23c)}$$

3. 波数　区間 $[k_{\min}, k_{\max}]$, $K = k_{\max} - k_{\min} = N\Delta k$

$$k_n = n\Delta k + lK, \ (n = N_{\min}, N_{\min}+1, \cdots, N_{\max}-1) \quad \text{(3.24a)}$$

$$\Delta k = \frac{K}{N} = \frac{1}{x_b - x_a} \quad \text{(3.24b)}$$

4. 関数の周期境界条件

$$f(x_m \pm X) = f(x_m), \qquad (\text{for all } m) \quad \text{(3.25a)}$$

$$f(k_n \pm K) = f(k_n), \qquad (\text{for all } n) \quad \text{(3.25b)}$$

5. 関係式

$$\Delta x \Delta k = \frac{1}{N}, \quad XK = N \quad \text{(3.26)}$$

6. 基底関数

$$\varphi_n(x) = e^{i2\pi k_n (x - x_a)} \quad \text{(3.27)}$$

離散フーリエ変換の指数関数部を詳細に見ていきましょう. つまり,

$$e^{i2\pi k_n x_m} \tag{3.28}$$

に注目します. 具体的な表記 Eq. (3.23), (3.24) を代入していくと,

$$\begin{aligned} e^{i2\pi k_n x_m} &= e^{i2\pi(n\Delta k+lK)(m\Delta x+x_a)} && \text{(3.29a)} \\ &= e^{i2\pi nm\Delta k\Delta x}\cdot e^{i2\pi lmK\Delta x}\cdot e^{i2\pi(n\Delta k+lK)x_a} && \text{(3.29b)} \\ &= e^{i2\pi nm/N}\cdot e^{i2\pi(n\Delta k+lK)x_a} && \text{(3.29c)} \end{aligned}$$

となります. ここで, $\Delta x\Delta k = 1/N,\ \Delta x = 1/K$ を用いました. もし $x_a = 0$ であれば非常に簡単になりますので, x_a を 0 にとりましょう. つまり, 基底関数はいつも

$$\varphi_n(x) = e^{i2\pi k_n x} \tag{3.30}$$

であり, 離散フーリエ変換は

$$\begin{cases} f(k_n) = \displaystyle\sum_{m=0}^{N-1} f(x_m)e^{-i2\pi nm/N} & \text{(3.31a)} \\ f(x_m) = \dfrac{1}{N}\displaystyle\sum_{n=N_{\min}}^{N_{\min}+N-1} f(k_n)e^{i2\pi nm/N} & \text{(3.31b)} \end{cases}$$

と書くことができます. しかしながら, 位置の分点は

$$x_m = m\Delta x,\ (m = 0, 1, \cdots, N-1) \tag{3.32}$$

に変更されました. つまり, x 軸の範囲はもはや $[x_a, x_b]$ と任意の範囲ではなくなり, $[0, N\Delta x]$ の範囲でいつも記述されていなければならないことを意味しています. 簡単にいうなれば, $x_{m=0} = 0$ を満たす決め方しか許されない代わりに簡単な表現 (3.31) を手に入れた, ということです. 良いところは, 分点の位置に左右されないことですね.Δx や Δk など x, k の情報に左右されないのです.

また, 波数についても範囲を決めましょう. 波数は範囲が $K = k_{\max} - k_{\min}$ であること以外制限がなく, 決めた範囲以外は K で周期的なので, これまでの議論から一意に決めることはできません.

$N_{\min}$ の決め方や, $x_{m=0} \neq 0$ とする取り方にしたい場合にはフーリエ変換の表記が変わりますので, それらについて一通り紹介します.

- **$N_{\mathrm{min}} = 0$ にとる場合**

  まずは最も多用される, $N_{\mathrm{min}} = 0$ とする場合を考えます. 何が便利かといえば, 逆離散フーリエ変換の式で添え字 n が 0 から $N-1$ として表現でき, すっきりした表記になることが便利です. 波数は $N_{\mathrm{min}} = 0$ に決め, 以下の通りになります.

$$k = [lK, (l+1)K], \tag{3.33a}$$

$$k_n = n\Delta k + lK, \quad n = 0, \cdots, N-1, \tag{3.33b}$$

$$K = N\Delta k, \quad \Delta k = \frac{K}{N} = \frac{1}{N\Delta x} \tag{3.33c}$$

  となります. 多くの場合で $l = 0$ が選ばれます. 波数の範囲は N_{min} によって自動的に決まるので, 離散フーリエ変換は,

$$\begin{cases} f(k_n) = \displaystyle\sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} & (3.34a) \\ f(x_m) = \dfrac{1}{N} \displaystyle\sum_{n=0}^{N-1} f(k_n) e^{i2\pi nm/N} & (3.34b) \end{cases}$$

  と書くことができます.x, k の範囲はそれぞれ $x = [0, X], k = [0, K]$ です. 範囲は $x_{m=0} = 0, k_{n=0} = 0$ である必要があります. 波数 k については特に考えずに周期境界条件を適用しても構いません [9]. つまり, k の範囲を $k = [-K/2, K/2]$ で考えたい場合には添え字は $k_{n=0} = 0$ を満たすように,

$$k = [-K/2, K/2], \tag{3.35a}$$

$$k_n = \begin{cases} N = \text{偶数の時} \begin{cases} n\Delta k, & n = 0, \cdots, N/2-1, \\ n\Delta k - K, & n = N/2, \cdots, N-1, \end{cases} \\ N = \text{奇数の時} \begin{cases} n\Delta k, & n = 0, \cdots, (N-1)/2, \\ n\Delta k - K, & n = (N-1)/2+1, \cdots, N-1, \end{cases} \end{cases} \tag{3.35b}$$

  とすればよいです. 離散フーリエ変換は Eq. (3.34) を使用できます.

[9] 位置については, 不連続関数を取った場合に不連続点の位置が変わってしまいます. そのため, 考えて周期境界条件を適用しなければなりません. 詳細は, 次の項目**位置の始まりを変更したい場合**を参照してください.

- **位置の左右対称に変更したい場合**

  最後に位置の始まりを左右対称にしたい場合を考えます. これは Eq. (3.32) で指摘したとおり, Eq. (3.29c) の後ろの指数関数がゼロにならなくなり ($x_a \neq 0$ となり), 表現が複雑になってしまいます. 勝手に変更することはできません. それを承知のうえでも $x_{m=0} = -X/2$ のようにして, $x = [-X/2, X/2]$ で考えたい場合を考えていきましょう.

  これを解決するためには関数を $-X/2$ だけ平行移動してしまえばよいのです. つまり, 今考えたい新しい位置の分点 x'_m は

$$x' = [-X/2, X/2], \tag{3.36a}$$

$$x'_m = m\Delta x - X/2, \quad m = 0, \cdots, N-1, \tag{3.36b}$$

  としたいです.Eq. (3.22) に立ち戻ると, 今の状況は $x_a = -X/2$ に相当するので $e^{i2\pi k_n x'_m}$ を考えていくと

$$\begin{aligned} e^{i2\pi k_n x'_m} &= e^{i2\pi nm/N} \cdot e^{i2\pi(n\Delta k + lK)(-X/2)} && (3.37a)\\ &= e^{i2\pi nm/N} \cdot e^{i2\pi(-nN\Delta k\Delta x/2 - lXK/2)} && (3.37b)\\ &= e^{i2\pi nm/N} \cdot e^{-i\pi(n+lN)} && (3.37c)\\ &= e^{i2\pi nm/N} \cdot (-1)^{n+lN} && (3.37d) \end{aligned}$$

  となります. 以上から, 区間 $[-X/2, X/2]$ で定義された関数の離散フーリエ変換は Eq. (3.34) に代入すれば, 離散フーリエ変換は

$$\begin{cases} (-1)^{n+lN} f(k_n) = \displaystyle\sum_{m=0}^{N-1} f(x'_m) e^{-i2\pi nm/N} \\ f(x'_m) = \dfrac{1}{N} \displaystyle\sum_{n=0}^{N-1} (-1)^{n+lN} f(k_n) e^{i2\pi nm/N} \end{cases} \tag{3.38a}$$

  となります. ここで, x は Eq. (3.36) に従い決定されます. 波数の範囲は Eq. (3.33) を考えて $k = [0, K]$ としても良いですし, Eq. (3.35) に従って $k = [-K/2, K/2]$ としても構いません ($l = 0$ を取った場合). 波数の周期境界条件を利用して範囲をずらす場合は, 位置の場合と違って離散フーリエ変換の係数に影響を及ぼさないからです. 別の言い方では, 位置の起点 x_a は基底関数の起点と関係があったので簡単にはいきませんが, 波数の起点は基底関数に関係ないです.

以上で, 位置が $[0, X]$ もしくは $[-X/2, X/2]$ で使えることを見てきましたが, もっと一般的な範囲では見ていないので, それを考えてみましょう. これを考える意義は例えば,

$x = 0$ の近傍ではなくて遠い地点に波形の大部分がある場合, 離散フーリエ変換 (3.31) は使用できないのではないか? という問題に応えるためです. 既にみたように, Eq. (3.22) の x_a を変更すれば良いのです.

これは平行移動の処理をしてしまえばよく, 平行移動した分だけを別で与えれば良いのだ, ということで解決することができます.

このアイデアを具体的に数式化しましょう. 本当にフーリエ変換したい波形 $f(x)$ が定義域 $[x_a, x_a + X]$ で定義されているとします. つまり,

$$x' = [x_a, x_a + X], \tag{3.39a}$$

$$x'_m = m\Delta x + x_a, \quad m = 0, \cdots, N-1, \tag{3.39b}$$

というわけです. $e^{i2\pi k_n x'_m}$ は

$$e^{i2\pi k_n x'_m} = e^{i2\pi nm/N} \cdot e^{i2\pi k_n x_a} \tag{3.40}$$

以上に進めることができません. よって, 離散フーリエ変換は

$$\begin{cases} e^{i2\pi k_n x_a} f(k_n) = \displaystyle\sum_{m=0}^{N-1} f(x'_m) e^{-i2\pi nm/N} \\ f(x'_m) = \dfrac{1}{N} \displaystyle\sum_{n=0}^{N-1} \Big[e^{i2\pi k_n x_a} f(k_n) \Big] e^{i2\pi nm/N} \end{cases} \tag{3.41a}$$

の形となります. 通常の数値計算などでは, Eq. (3.41a) の右辺の形で実装されていることが多いです. よって, 本来のフーリエ変換結果である $f(k)$ が欲しい場合は, 左辺に現れる余計な位相をキャンセルするようにこの因子を掛けなければなりません.

3.1.5 本稿の離散フーリエ変換の定義

本稿の離散フーリエ変換は Eq. (3.34) に従って以下の通りに定義します.

- 離散フーリエ変換

$$\begin{cases} f(k_n) = \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} & \text{(3.42a)} \\ f(x_m) = \frac{1}{N} \sum_{n=0}^{N-1} f(k_n) e^{i2\pi nm/N} & \text{(3.42b)} \end{cases}$$

- 位置　区間 $[0, X]$

$$x_m = m\Delta x, \ (m = 0, 1, \cdots, N-1) \tag{3.43a}$$
$$\Delta x = \frac{X}{N} = \frac{1}{K} \tag{3.43b}$$

- 波数　区間 $[lK, (l+1)K]$,

$$k_n = n\Delta k + lK, \ (n = 0, 1, \cdots, N-1) \tag{3.44a}$$
$$\Delta k = \frac{K}{N} = \frac{1}{X}, \tag{3.44b}$$
$$K = N\Delta k = \frac{N}{X} \tag{3.44c}$$

$l = 0, \pm 1, \pm 2, \cdots$ は任意の整数で, 典型的には $l = 0$ がとられ, 本稿でも注記が無い限り $l = 0$ とします. この l の値によってオーバーサンプリングやダウンサンプリングとして解釈されます.

- 関数の周期境界条件

$$f(x_m \pm X) = f(x_m), \quad (\text{for all } m) \tag{3.45a}$$
$$f(k_n \pm K) = f(k_n), \quad (\text{for all } n) \tag{3.45b}$$

- 関係式

$$\Delta x \Delta k = \frac{1}{N}, \quad XK = N \tag{3.46}$$

- 基底関数

$$\varphi_n(x) = e^{i2\pi k_n x}, \tag{3.47a}$$

特に $x = x_m$の分点上では,

$$\varphi_n(x_m) = e^{i2\pi nm/N}, \tag{3.47b}$$

となります.

任意の区間におけるフーリエ変換を知りたい場合は項 3.39 で示した通りの方法と注意で実装できます. また, 今後多くの場合で採用されている $l = 0, N =$ (偶数) に絞り考えていきます.

3.1.6 離散フーリエ変換の例

項 2.1.3 と同じように, 同じ関数を用いて離散フーリエ変換を実施してみましょう.

$$f(x) = \cos(2\pi 3x) \tag{3.48}$$

という, 波数 3 を持つ波 (周期 $X_p = \frac{2\pi}{2\pi\times 3} = 1/3$ を持つ波) を離散フーリエ変換してみます. ここでは, 離散フーリエ変換の区間 Eq. (3.43) と $f(x)$ の周期が一致する場合と, 一致しない場合の 2 つの例を考えてみます.

- **離散フーリエ変換の区間 Eq. (3.43) と $f(x)$ の周期が一致する場合**

  離散フーリエ変換の区間を $x = [0, qX_p]$ と波形の周期の整数倍として考えてみます. ここで, q は正の整数で $q = 1, 2, \cdots$ を取るものとします. 離散フーリエ変換における特徴的な量は以下のように決まります.

$$\Delta x = \frac{qX_p}{N},\ \Delta k = \frac{1}{qX_p}, \tag{3.49}$$

$$K = \frac{N}{qX_p} \ \rightarrow\ k = \left[0, \frac{N}{qX_p}\right] \tag{3.50}$$

  まずは順方向離散フーリエ変換をしてみます.Eq. (3.42a) に代入して,

$$\begin{aligned} f(k_n) &= \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} && \text{(3.51a)}\\ &= \sum_{m=0}^{N-1} \cos(2\pi 3 \cdot m\frac{qX_p}{N}) e^{-i2\pi nm/N} && \text{(3.51b)}\\ &= \sum_{m=0}^{N-1} \cos(2\pi \frac{qm}{N}) e^{-i2\pi nm/N} && \text{(3.51c)}\\ &= \frac{1}{2}\sum_{m=0}^{N-1} \left[e^{i2\pi m\frac{(q-n)}{N}} + e^{-i2\pi m\frac{(q+n)}{N}}\right] && \text{(3.51d)}\\ &= \frac{N}{2}\left(\delta_{n,q} + \delta_{n,-q}\right) && \text{(3.51e)} \end{aligned}$$

  となります. ここで, $\delta_{i,j}$ はクロネッカーのデルタで

$$\delta_{i,j} = \begin{cases} 1, & (i = j) \\ 0, & (i \neq j) \end{cases} \tag{3.52}$$

  です.$q > 0$ であることから, Eq. (3.55) の右辺の $\delta_{n,-q}$ で $n = -q$ になることはないと思われますが, 波数に対する周期境界条件 $f(k_n) = f(k_n \pm K)$ があることから,

$\delta_{n,-q+N}$ とするべきで, 値を持つことになります. つまり, 現在の定義で順方向離散フーリエ変換の結果は

$$f(k_n) = \frac{N}{2}\left(\delta_{n,q} + \delta_{n,-q+N}\right), \ (n = 0, 1, \cdots, N-1) \tag{3.53}$$

となります. ここで, 等比級数の総和を用いて

$$\sum_{m=0}^{N-1} e^{i2\pi m(n-n')/N} \quad = \quad N\delta_{n,n'} \tag{3.54}$$

を用いました.

Eq. (3.53) より, $n = q, N-q$ の位置にピークが現れることが分かります. これはつまり, 波数としては

$$k_n = n\Delta k = \begin{cases} q\Delta k = q\dfrac{1}{qX_p} = 3 \\ (N-q)\Delta k = N\Delta k - q\dfrac{1}{qX_p} = K-3 \end{cases} \tag{3.55}$$

となり, 波数 3 と $K-3$ にピークが出ることが分かります. 周期境界条件により, 波数に K の整数倍を足したり引いたりしても良いので, 分かりやすい表現では, 波数の成分 ± 3 を持つことが分かります. この 3 は元々の波形の波数が 3 だから, ということで期待する通りの結果であることが分かります.

最後に連続フーリエ変換との比較をしましょう. 連続フーリエ変換の結果は Eq. (2.7) の通り, デルタ関数を用いて

$$f(k) = \frac{1}{2}[\delta(k-3) + \delta(k+3)] \tag{3.56}$$

となります. それに対し, 離散フーリエ変換では Eq. (3.53) の通りで

$$f(k_n) = \frac{N}{2}\left(\delta_{k_n,3} + \delta_{k_n,-3}\right) \tag{3.57}$$

です. 両者を比較するとデルタ関数がクロネッカーのデルタに対応していることが分かります. 実際には比較するためには, 離散フーリエ変換で $\frac{1}{N}$ に押し込めた $\Delta x\Delta k$ のうち, Δx を掛けなければなりません (Eq. (3.20a) を参照). つまり,

$$\delta(k-3) \to N\delta_{k_n,3}\Delta x \tag{3.58}$$

に対応することが分かります.

- **離散フーリエ変換の区間 Eq. (3.43) と $f(x)$ の周期が一致しない場合**

  続いて, 離散フーリエ変換の区間を $x = [0, X]$ として, 任意の範囲で考えてみます. 特に, X が周期の無理数倍と仮定して, 必ず周期境界条件を満たさない場合を仮定します[10].

  離散フーリエ変換における特徴的な量は以下のように決まります.

$$\Delta x = \frac{X}{N},\ \Delta k = \frac{1}{X}, \tag{3.59}$$

$$K = \frac{N}{X} \ \rightarrow \ [0, K] = \left[0, \frac{N}{X}\right] \tag{3.60}$$

  順方向離散フーリエ変換をしてみます.Eq. (3.42a) に代入して,

$$\begin{aligned}
f(k_n) &= \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} && \text{(3.61a)}\\
&= \sum_{m=0}^{N-1} \cos(2\pi 3 \cdot m\frac{X}{N}) e^{-i2\pi nm/N} && \text{(3.61b)}\\
&= \frac{1}{2}\sum_{m=0}^{N-1} \left[e^{i2\pi m\frac{(3X-n)}{N}} + e^{-i2\pi m\frac{(3X+n)}{N}} \right] && \text{(3.61c)}\\
&= \frac{1}{2}\left(\frac{e^{i2\pi(3X-n)} - 1}{e^{i2\pi(3X-n)/N} - 1} + \frac{e^{-i2\pi(3X+n)} - 1}{e^{-i2\pi(3X+n)/N} - 1} \right) && \text{(3.61d)}\\
&= \frac{1}{2}\left(\frac{e^{i2\pi 3X} - 1}{e^{-i2\pi(k_n-3)\Delta x} - 1} + \frac{e^{-i2\pi 3X} - 1}{e^{-i2\pi(k_n+3-K)\Delta x} - 1} \right) && \text{(3.61e)}
\end{aligned}$$

  となります. 注記しますが, X は $f(x)$ の周期の無理数倍であることから, 離散フーリエ変換で参照される波数 $k_n = n\Delta k = \frac{n}{X}$ が 3 の整数倍に等しくなることはないため, Eq. (3.61e) の分母が発散することはなく, Eq. (3.61c) から Eq. (3.61d) の計算が可能になっています. また, Eq. (3.61e) の右辺第二項の分母に突如現れた K は有ってもなくても影響しませんが, 分母の発散が分かりやすいように, また定義域に収まるように追加しました.

  では結果 (3.61e) が何を示すのか考えていきましょう. 右辺の分母に着目すると, $k_n = 3, K-3$ に近い場合にゼロに近づくため, $f(k_n)$ の値が大きくなることが予想されます. 以降, $k_n = 3, K-3$ というのは面倒なので, K だけずらして $k_n = \pm 3$ と呼びます.

  以上から, $f(k_n)$ は $k_n = \pm 3$ 周りにある程度の幅を持ったピークを持つ関数となることが分かります. 幅を持つということは, $k = \pm 3$ 以外の成分が出ていることを意

[10] 周期境界条件を満たさない条件なら半整数倍, もしくは有理数倍にすればよいですが, そうしない理由について Appendix B をご覧ください.

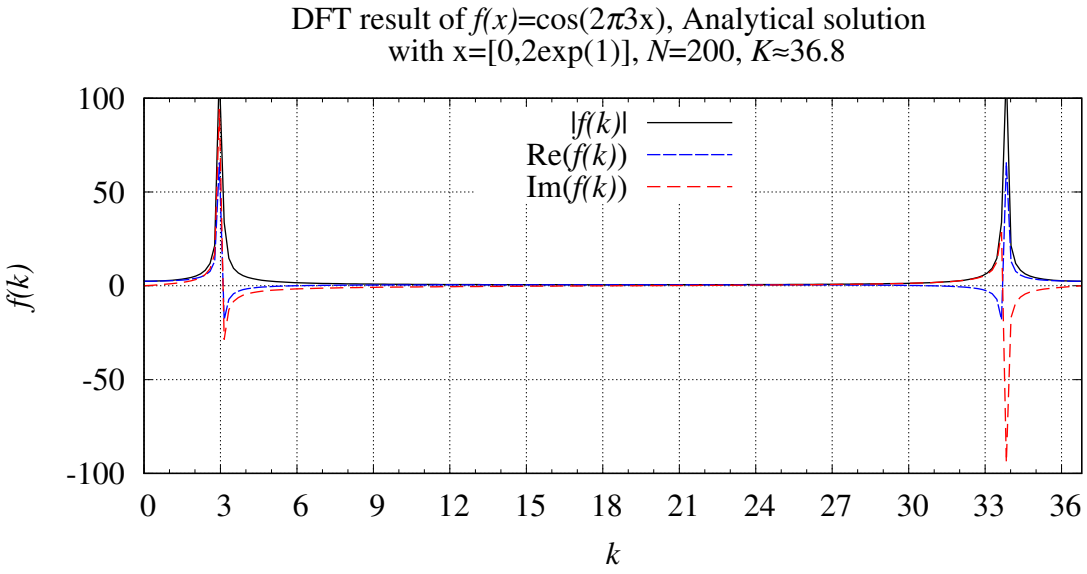

Figure 2:　Eq. (3.48) の DFT 結果 (2) の図示. $X = [0, 2\exp(1)], N = 200, K = 1/\Delta x \approx 36.8$ として計算

味しています. 元々の波形 (3.48) には $k = \pm 3$ 以外の成分が含まれていなかったのに, どうしてそれ以外の波数が出てくるのでしょうか. これは周期境界条件を満たした波形を離散フーリエ変換していないことが原因です. 離散フーリエ変換の基底関数が周期境界条件を満たしているため, そこから構成される $f(x)$ も周期境界条件 (3.5) を満たしていなければなりません. そうでなければ, 離散フーリエ変換は適用するべきではなく, Eq. (3.61e) は適用するべきではないときの結果となります.

しかし, この適用するべきではないという事実は理不尽です. なぜなら, 多くの場合ではその関数の波数を知りたいために離散フーリエ変換を行うのですが, その離散フーリエ変換を行うためにはその関数の波数を知っていなければならないというのは理不尽極まりないのです. もし関数の波数を知らない場合に, 正確な離散フーリエ変換を行いたければ, 離散フーリエ変換の範囲 X は有理数で表現できなくなる値まで大きくとらなければなりません. つまり, $X \to \infty$ として無限大の区間で行わなければならず, これはもはや離散フーリエ変換ではなくなります.

多くの場合では有限に収めなければならないため, 妥協します. つまり, 正確ではないけど Eq. (3.61e) はほとんど正解に近いはずだと考える訳です. 図で Eq. (3.61e) を書けば, Fig. 2 のようになります. $k = 3$ と, $k \approx 33.8$ に幅を持つピークが現れていることが分かります. もしも k の範囲を正負で考えて K だけずらせば, $k = \pm 3$ となります. これは, もともと考えたかった波形が持つピークの位置となります. ですが, 幅を持って広がるため, Eq. (3.55) のようにその周波数だけピークになる, ということにはなりません.

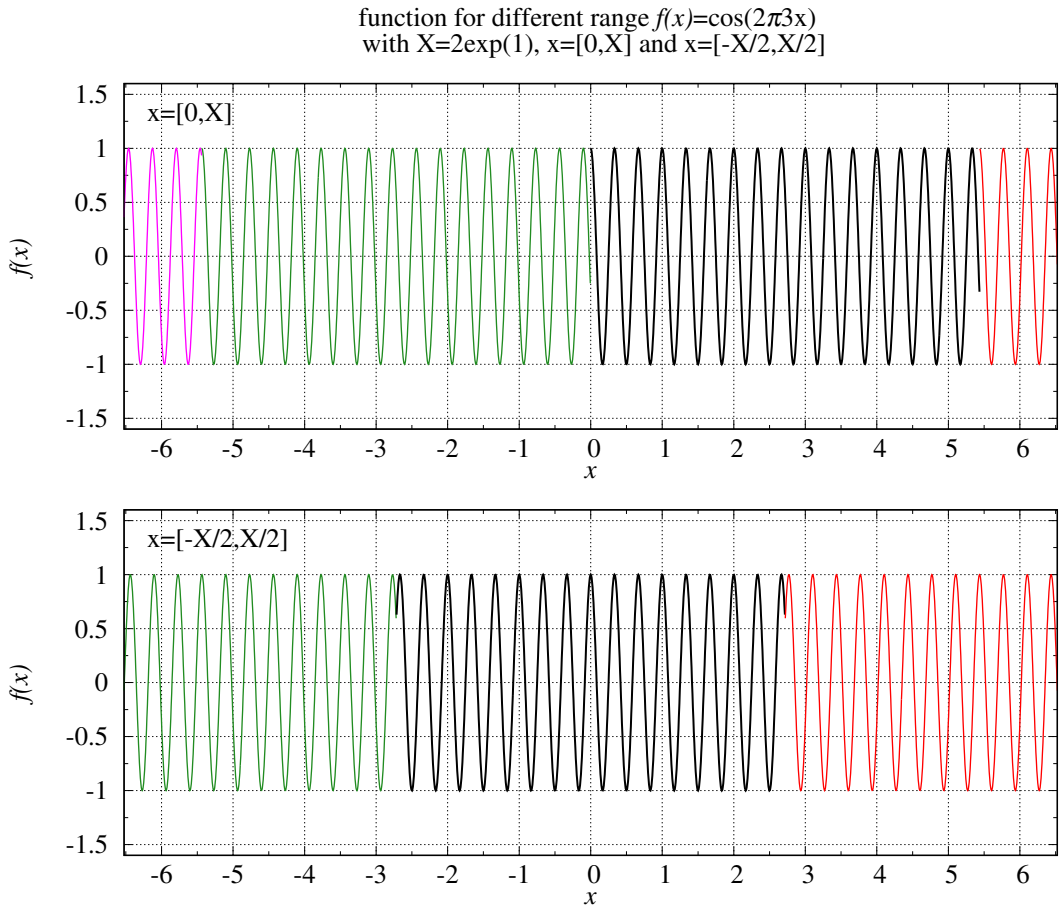

Figure 3: 周期関数の異なる範囲の離散フーリエ変換を考えた場合. (上)$x = [0, X]$, (下)$x = [-X/2, X/2]$. 色ごとに X の範囲を示す.

- **周期が一致しないかつ左右対称の区間を選んだ場合**

今までは位置の範囲が $x = [0, X]$ でしたが, $x = [-X/2, X/2]$ の場合を考えてみましょう. この 2 つは同じ結論にはなりません. その理由は, 不連続点が $x = 0$ に来るか,$x = X/2$ に来るかの違いがあるためです.Fig. 3 にその比較を示しました. この違いによって, 離散フーリエ変換が認識する関数の周期に差が生じます. この場合のフーリエ変換は, Eq. (3.38a) でみた形で計算しなければなりません. つまり, 離散

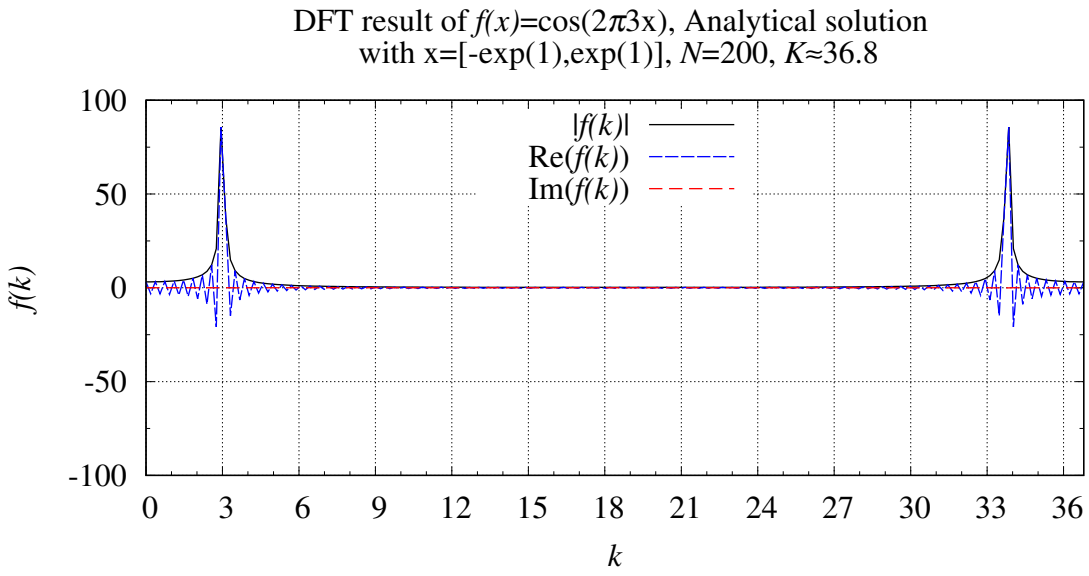

Figure 4:　Eq. (3.48) を左右対称の区間で離散フーリエ変換した解析解 (3.63) の図示.　$X = [-\exp(1), \exp(1)], N = 200, K = 1/\Delta x \approx 36.8$ として計算

フーリエ変換の結果は $l = 0$ を考えて,

$$(-1)^n f(k_n) = \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} \tag{3.62a}$$

$$= \sum_{m=0}^{N-1} \cos(2\pi 3 \cdot (m\Delta x - X/2)) e^{-i2\pi nm/N} \tag{3.62b}$$

$$= \frac{1}{2}\left(e^{-i\pi 3X} \frac{e^{i2\pi 3X} - 1}{e^{-i2\pi(k_n - 3)\Delta x} - 1} + e^{i\pi 3X} \frac{e^{-i2\pi 3X} - 1}{e^{-i2\pi(k_n + 3 - K)\Delta x} - 1} \right) \tag{3.62c}$$

より,

$$f(k_n) = (-1)^n \cdot \frac{1}{2} \cdot \left(e^{-i\pi 3X} \frac{e^{i2\pi 3X} - 1}{e^{-i2\pi(k_n - 3)\Delta x} - 1} + e^{i\pi 3X} \frac{e^{-i2\pi 3X} - 1}{e^{-i2\pi(k_n + 3 - K)\Delta x} - 1} \right) \tag{3.63}$$

となります.Eq. (3.61e) と比べて $(-1)^n$ と, 指数関数が追加された形になります. 実際に Eq. (3.63) を図示すると, Fig. 4 のようになります.

3.2 離散畳み込み (Convolution)

節2.2で考えた畳み込みを離散的に考えてみましょう. 離散畳み込みとは, 関数$f(x), g(x)$の値が, 同じ区間で離散的かつ等間隔の離散点にだけ値が与えられているときに以下の和を計算することです.

$$(f * g)_{\mathrm{d}}(x_n) \equiv \sum_l f(x_l) g(x_n - x_l) \tag{3.64a}$$

$f(x), g(x)$の定義域は同じ範囲です. 一点注意しておかなければならない点は, 連続の場合に定義した畳み込み(2.8a)をそのまま離散的にしたのではないことです.Eq. (2.8a)をそのまま離散化するならば, 例えば

$$\begin{aligned}(f * g)(x_n) &= \int_{-\infty}^{\infty} f(\tau) g(x_n - \tau) d\tau && (3.65a)\\ &\approx \sum_p f(x_p) g(x_n - x_p) \Delta x && (3.65b)\\ &= (f * g)_{\mathrm{d}}(x_n) \cdot \Delta x && (3.65c)\end{aligned}$$

となります. ですが, 多くの場合ではEq. (3.64a)のようにΔxを消去したものを離散畳み込みと呼んでいるので, 本稿もそれに倣います.

さて, Eq. (3.64a)で添え字の範囲などを明確にを指定しなかったのには理由があります. これは, Eq. (3.64a)の定義において$x_n - x_l$が定義域から飛び出してしまう場合が存在するためです. この飛び出した範囲の$g(x)$をどのように処理するかで, 畳み込みの結果が変わります. 主な決め方は2つあり, 関数を周期的と考える『循環畳み込み』, もしくは定義域外はゼロと考える『直線畳み込み』です.

- **循環畳み込み**

  循環畳み込みは定義外の領域を周期的であるとする方法です. つまり,

  $$g(x_n \pm X) = g(x_n), (\text{for all } n) \tag{3.66}$$

  として考えます (fについても同様です). ここで, Xは定義域の大きさです.

- **直線畳み込み**

  直線畳み込みは定義外の領域をゼロであるとする方法です. つまり,

  $$g(x_n) = \begin{cases} g(x_n), & (x_n\text{は定義域内}) \\ 0, & (\text{otherwise}) \end{cases} \tag{3.67}$$

  として考えます.

本稿では特に**循環畳み込み**を考えていきます. これは離散フーリエ変換を利用して畳み込みを計算したいという意図があるため, 同じ周期境界条件を持つものを仮定したいためです.

3.2.1 離散畳み込みの定義

本稿における離散畳み込みは循環畳み込みを想定し, 以下のように定義します.

$$(f * g)_{\mathrm{d}}(x_m) \equiv \sum_{p=0}^{N-1} f(x_p) g(x_m - x_p) \tag{3.68a}$$

$$f(x_m \pm X) = f(x_m) \tag{3.68b}$$

$$g(x_m \pm X) = g(x_m) \tag{3.68c}$$

ここで, X は定義域の区間であり, 位置の分点と共に以下の通り定義します.

$$x = [0, X] \tag{3.69a}$$

$$x_m = m\Delta x, \quad (m = 0, 1, 2 \cdots, N-1) \tag{3.69b}$$

$$\Delta x = \frac{X}{N} \tag{3.69c}$$

もし, 別の定義域で定義された波形を考えたい場合は, 平行移動して $[0, X]$ の範囲に移動させて計算し, その後全て計算し終えたら平行移動した分だけ戻すことで実行できます.

3.2.2 離散畳み込みと離散フーリエ変換

項 2.2.2 の連続の場合で考えたように, 離散畳み込みを離散フーリエ変換された関数で考えてみましょう. 離散フーリエ変換の定義は Eq. (3.42a), (3.42b) に従います. 循環畳み込みを考えたいので, 関数は区間で周期的であるから関数 f, g は

$$f(x_m) = \frac{1}{N} \sum_{n=0}^{N-1} f(k_n) e^{i2\pi nm/N} \tag{3.70a}$$

$$g(x_m) = \frac{1}{N} \sum_{n=0}^{N-1} g(k_n) e^{i2\pi nm/N} \tag{3.70b}$$

と書けています.Eq. (3.70a), (3.70b) を Eq. (3.68a) に代入すれば,

$$
\begin{aligned}
(f*g)_{\mathrm{d}}(x_n) &= \sum_{p=0}^{N-1} f(x_p)g(x_m-x_p) && \text{(3.71a)}\\
&= \sum_{p=0}^{N-1}\left[\frac{1}{N}\sum_{n=0}^{N-1} f(k_n)e^{i2\pi np/N}\right]\left[\frac{1}{N}\sum_{n'=0}^{N-1} g(k_{n'})e^{i2\pi n'(m-p)/N}\right] && \text{(3.71b)}\\
&= \frac{1}{N^2}\sum_{n=0}^{N-1} f(k_n)\sum_{n'=0}^{N-1} g(k_{n'})e^{i2\pi n'm/N}\sum_{p=0}^{N-1} e^{i2\pi(n-n')p/N} && \text{(3.71c)}\\
&= \frac{1}{N^2}\sum_{n=0}^{N-1} f(k_n)\sum_{n'=0}^{N-1} g(k_{n'})e^{i2\pi n'm/N}N\delta_{n,n'} && \text{(3.71d)}\\
&= \frac{1}{N}\sum_{n=0}^{N-1} f(k_n)g(k_n)e^{i2\pi nm/N} && \text{(3.71e)}
\end{aligned}
$$

と計算することができます. つまり, 関数 $f(k_n)g(k_n)$ の逆変換が離散畳み込みの結果を与える, といっています. 結局, 項 2.2.2 と同じ結論が得られるわけですね. 畳み込みの離散フーリエ変換の結果なので, $(f*g)_{\mathrm{d}}(k_n)\equiv f(k_n)g(k_n)$ と書きましょう. 実際に畳み込みをフーリエ変換すると以下のようになります.

$$
\begin{aligned}
(f*g)_{\mathrm{d}}(k_n)\equiv f(k_n)g(k_n) &= \mathcal{F}[(f*g)_{\mathrm{d}}(x)](k_n) && \text{(3.72a)}\\
&= \sum_{m=0}^{N-1}\left[\sum_{p=0}^{N-1} f(x_p)g(x_m-x_p)\right]e^{-i2\pi nm/N} && \text{(3.72b)}\\
&= \sum_{p=0}^{N-1} f(x_p)e^{-i2\pi np/N}\sum_{m=0}^{N-1} g(x_m-x_p)e^{-i2\pi n(m-p)/N} && \text{(3.72c)}\\
&= \sum_{p=0}^{N-1} f(x_p)e^{-i2\pi np/N}\sum_{m'=0}^{N-1} g(x_{m'})e^{-i2\pi nm'/N} && \text{(3.72d)}\\
&= \sum_{p=0}^{N-1} f(x_p)e^{-i2\pi np/N}\cdot g(k_n) && \text{(3.72e)}\\
&= f(k_n)g(k_n) && \text{(3.72f)}
\end{aligned}
$$

ここで, Eq. (3.72c) から Eq. (3.72d) に移る際に循環畳み込みの性質 (3.68c) を利用しています. これも項 2.2.2 と同様の結論が得られたことになります. 畳み込み積分 (3.68a) 自体を知りたければ, Eq. (3.72f) を逆変換することで得ることができます.

3.2.3 任意区間における離散畳み込みと離散フーリエ変換

任意区間における離散畳み込みと離散フーリエ変換の関係を考えてみます.

$x=0$ を中心とした関数なので $x=[x_a, x_a+X]$ で定義されたフーリエ変換 Eq. (3.41a) を元に考えていきます.$l=0$ として再掲すれば

$$\begin{cases} e^{i2\pi k_n x_a} f(k_n) = \displaystyle\sum_{m=0}^{N-1} f(x'_m) e^{-i2\pi nm/N} \\ f(x'_m) = \dfrac{1}{N} \displaystyle\sum_{n=0}^{N-1} \left[e^{i2\pi k_n x_a} f(k_n) \right] e^{i2\pi nm/N} \end{cases} \tag{3.73a}$$

位置, 波数の位置はそれぞれ Eq. (3.74), (3.75) のように決めています.

$$x' = [x_a, x_a + X], \tag{3.74a}$$

$$x'_m = m\Delta x + x_a, \quad m = 0, \cdots, N-1, \tag{3.74b}$$

$$k = [0, K], \tag{3.75a}$$

$$k_n = n\Delta k, \quad n = 0, \cdots, N-1, \tag{3.75b}$$

$x_m - x_p = (m-p)\Delta x$ を計算する際は $x_a = 0$ と考えることに注意して

$$(f * g)_{\mathrm{d}}(x'_m) = \sum_{p=0}^{N-1} f(x'_p) g(x'_m - x'_p) \tag{3.76a}$$

$$= \sum_{p=0}^{N-1} \left[\frac{1}{N} \sum_{n=0}^{N-1} e^{i2\pi k_n x_a} f(k_n) e^{i2\pi np/N} \right] \left[\frac{1}{N} \sum_{n'=0}^{N-1} g(k_{n'}) e^{i2\pi n'(m-p)/N} \right] \tag{3.76b}$$

$$= \frac{1}{N^2} \sum_{n=0}^{N-1} e^{i2\pi k_n x_a} f(k_n) \sum_{n'=0}^{N-1} g(k_{n'}) e^{i2\pi n'm/N} \sum_{p=0}^{N-1} e^{i2\pi(n-n')p/N} \tag{3.76c}$$

$$= \frac{1}{N^2} \sum_{n=0}^{N-1} e^{i2\pi k_n x_a} f(k_n) \sum_{n'=0}^{N-1} g(k_{n'}) e^{i2\pi n'm/N} \cdot N\delta_{n,n'} \tag{3.76d}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} e^{-i2\pi k_n x_a} \left[e^{i2\pi k_n x_a} f(k_n) \right] \left[e^{i2\pi k_n x_a} g(k_n) \right] e^{i2\pi nm/N} \tag{3.76e}$$

となります.Eq. (3.76e) の中括弧 [ ] 内は離散フーリエ変換 Eq. (3.73) の結果に対応します. つまり, フーリエ変換を通して畳み込みを計算する場合は, 位相因子 $e^{-i2\pi k_n x_a}$ を付与しなければならないことに注意してください. 連続畳み込みと対応付ける際は, さらに積分の刻み幅である Δx を最後の結果に掛けることも忘れないようにしましょう.

3.2.4 離散畳み込みの例

ここでは離散畳み込みの例を考えてみます. 問題は連続の畳み込み (2.2.3) と同じように 2 つの関数

$$f(x) = x^5 e^{-x^2} \tag{3.77}$$

$$g(x) = e^{-4x^2} \tag{3.78}$$

の離散畳み込み

$$(f*g)_{\mathrm{d}}(x_m) = \sum_p f(x_p)g(x_m - x_p) \tag{3.79}$$

を考えてみます.

今回考える関数は $x=0$ を中心とした関数なので, $x=[x_a, x_a+X]$ で定義されたフーリエ変換 Eq. (3.73) を元に考えていきます. ここで x'_m と k_n は, Eq. (3.74), (3.75) で決めます.

それぞれの連続フーリエ変換を離散化することで簡易的に求めましょう. この対応は, $-x_a, x_a+X >> 1$ のみで成り立ちます. 連続の結果は項 2.2.3 で得た結果をただ離散化すればよいので,

$$f(k) = -\frac{i\pi^{3/2}}{4}(4\pi^4k^5 - 20\pi^2k^3 + 15k)e^{-\pi^2k^2}\cdot\frac{1}{\Delta x} \tag{3.80}$$

$$g(k) = \frac{1}{2}\sqrt{\pi}e^{-\pi^2k^2/4}\cdot\frac{1}{\Delta x} \tag{3.81}$$

となります. ここで $1/\Delta x$ は連続から離散の表現に伴って生じる, $\int dx \to \sum \Delta x$ への変形時に生じる Δx を左辺に移した形です. Eq. (3.79) の解も同じように離散化すれば,

$$(f*g)_{\mathrm{d}}(x_m) = \sum_p f(x_p)g(x_m - x_p) \tag{3.82}$$

$$= \frac{1}{3125}\sqrt{\frac{\pi}{5}}x_m\left(1024x_m^4 + 1600x_m^2 + 375\right)e^{-4x_m^2/5}\cdot\frac{1}{\Delta x} \tag{3.83}$$

を得ます.

4 プログラム

本章では畳み込みを実行するプログラムを載せます. 言語は Fortran90 です.

4.1 Intel MKL を利用した離散 (高速) フーリエ変換

4.1.1 Intel MKL の離散フーリエ変換の定義

Intel MKL の離散フーリエ変換は, Eq. (3.42) で見た形で定義されています.

- 離散フーリエ変換

$$\begin{cases} f(k_n) = \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} & \text{(4.1a)} \\ f(x_m) = \sum_{n=0}^{N-1} f(k_n) e^{i2\pi nm/N} & \text{(4.1b)} \end{cases}$$

- 位置

$$x = [0, X] \tag{4.2a}$$
$$x_m = m\Delta x, \ (m = 0, 1, \cdots, N-1) \tag{4.2b}$$

ここで, Δx は以下で定義されます.

$$\Delta x = \frac{X}{N} = \frac{1}{K} \tag{4.3}$$

- 波数

$$k = [0, K] \tag{4.4a}$$
$$k_n = n\Delta k + lK, \ (n = 0, 1, \cdots, N-1) \tag{4.4b}$$

ここで, $\Delta k, K$ は以下で定義されます.

$$\Delta k = \frac{K}{N} = \frac{1}{X}, \tag{4.5a}$$
$$K = N\Delta k = \frac{N}{X} \tag{4.5b}$$

- 関数の周期境界条件

$$f(x_m \pm X) = f(x_m), \qquad (\text{for all } m) \tag{4.6a}$$
$$f(k_n \pm K) = f(k_n), \qquad (\text{for all } n) \tag{4.6b}$$

- 関係式

$$\Delta x \Delta k = \frac{1}{N}, \quad XK = N \tag{4.7}$$

- 基底関数

$$\varphi_n(x) = e^{i2\pi k_n x}, \tag{4.8a}$$

特に $x = x_m$ の分点上では

$$\varphi_n(x_m) = e^{i2\pi nm/N}, \tag{4.8b}$$

となります.

4.1.2 プログラム

Intel MKL を利用したプログラムを書いてみましょう．例えば Program (1) のようになります．分かりやすさを中心に書いていますので, 無駄があります．プログラム中の変数 Np が $2, 3, 5, 7$ の乗数になると高速フーリエ変換として自動的に実行されます．MKL の詳しい使い方はマニュアル [1] をご覧ください．

Listing 1: Intel MKL を用いた Fortran90 のプログラム例

```
1  program main
2    implicit none
3    integer::m
4    double precision::dx,Xrange
5    double precision,allocatable::x(:)
6    integer::n
7    double precision::dk,Krange
8    double precision,allocatable::k(:)
9    integer::Np
10   complex(kind(0d0)),allocatable::f(:)
11   complex(kind(0d0)),external::func
12
13   Np = 200 ! Number of DFT points
14   Xrange = 2d0*exp(1d0) ! x=[0,Xrange]
15
16   ! Initialize and allocation for DFT
17   allocate(x(0:Np-1),k(0:Np-1),f(0:Np-1))
18   x = 0d0
19   k = 0d0
20   f = dcmplx(0d0,0d0)
21
22   ! Define position x
23   dx = Xrange / Np
24   call dft_abscissa(Np, x, dx) ! x=[0, Xrange]
25
26   ! Calculate k abscissa
27   Krange = 1d0/dx
28   dk = Krange / Np
29   call dft_abscissa(Np, k, dk) ! k=[0, Krange]
30
31   ! Calculate function f at x=x(m)
32   do m = 0,Np-1
33      f(m) = func(x(m))
34   enddo
35
36   ! Write down f(x)
37   call dft_write("before_fx.d",Np,x,f)
38
39   ! Forward dft (f is overwrited)
40   call dft(Np,f,"forward")
41
42   ! Write down f(k)
43   call dft_write("fk.d",Np,k,f) ! output k=[0,Krange]
44   call dft_write_center0("fk_center0.d",Np,k,f) ! output k=[-Krange/2,Krange/2]
45
46   ! Backward dft (f is overwrited)
47   call dft(Np,f,"backward")
48
49   ! Write down f(x)
50   call dft_write("after_fx.d",Np,x,f)
51
52   stop
53 end program main
54
55 complex(kind(0d0)) function func(x)
56   implicit none
57   double precision,intent(in)::x
58   !
59   ! Defined x=[0,X]
60   !
61   double precision::pi=dacos(-1d0)
62
63   func=dcmplx(dcos(2d0*pi*3d0*x),0d0)
64   !func=dcmplx(dsin(2d0*pi*5d0*x),0d0)
65   !func=exp(dcmplx(0d0,2d0*pi*1*x))
```

```
66
67    return
68  end function func
69
70  !---------------------------------
71
72  subroutine dft_abscissa(N, w, dw)
73    implicit none
74    integer,intent(in)::N
75    double precision,intent(in)::dw
76    double precision,intent(out)::w(0:N-1)
77    !
78    !w=[0, W]
79    ! w(0) = 0
80    ! w(N-1) = (N-1)*dw = W-dw
81    !
82    integer::j
83
84    do j = 0,N-1
85       w(j) = j*dw
86    enddo
87
88    return
89  end subroutine dft_abscissa
90
91  include "/opt/mkl/include/mkl_dfti.f90"
92  subroutine dft(N,f,FB)
93    use MKL_DFTI
94    implicit none
95    integer,intent(in)::N
96    complex(kind(0d0)),intent(inout)::f(0:N-1)
97    character(*),intent(in)::FB
98    !
99    !sikinote
100   ! author : sikino
101   ! date : 2022/06/11
102   !
103   !n: Number of DFT points.
104   !f(i): value of data at i
105   !FB: "forward" -> Forward DFT
106   ! "backward" -> Backward DFT
107   integer::Status
108   TYPE(DFTI_DESCRIPTOR),POINTER::hand
109
110   Status = DftiCreateDescriptor(hand,DFTI_DOUBLE,DFTI_COMPLEX,1,N)
111   Status = DftiSetValue(hand,DFTI_FORWARD_SCALE,1d0)
112   Status = DftiSetValue(hand,DFTI_BACKWARD_SCALE,1d0/dble(N))
113   Status = DftiCommitDescriptor(hand)
114
115   if(trim(FB)=="forward")then
116      Status = DftiComputeForward(hand,f)
117   elseif(trim(FB)=="backward")then
118      Status = DftiComputeBackward(hand,f)
119   else
120      write(6,*)"DFT string different"
121      stop
122   endif
123   Status = DftiFreeDescriptor(hand)
124
125   return
126 end subroutine dft
127
128 subroutine dft_write(fname,N,w,f)
129   implicit none
130   character(*),intent(in)::fname
131   integer,intent(in)::N
132   double precision,intent(in)::w(0:N-1)
133   complex(kind(0d0)),intent(in)::f(0:N-1)
134
135   ! w = [0,W]
136   ! w(0) = 0
137   ! --> output range w=[0,W]
138   integer::j
139
140   open(21,file=trim(fname))
141   do j = 0,N-1
142      write(21,'(3e26.16e3)')w(j),dble(f(j)),dimag(f(j))
143   enddo
144   close(21)
145
146   return
```

```
147 end subroutine dft_write
148
149
150 subroutine dft_write_center0(fname,N,w,f)
151   implicit none
152   character(*),intent(in)::fname
153   integer,intent(in)::N
154   double precision,intent(in)::w(0:N-1)
155   complex(kind(0d0)),intent(in)::f(0:N-1)
156
157   ! w = [0,W]
158   ! w(0) = 0
159   ! --> output range w=[-W/2,W/2]
160   integer::j,Nhalf
161   double precision::dw,Wrange
162
163   if(mod(N,2)==0)then
164      Nhalf = N/2
165   else
166      Nhalf = (N-1)/2
167   endif
168
169   dw = abs(w(1)-w(0))
170   Wrange = N*dw
171
172   open(21,file=trim(fname))
173   do j = Nhalf,N-1
174      write(21,'(3e26.16e3)')w(j)-Wrange,dble(f(j)),dimag(f(j))
175   enddo
176   do j = 0,Nhalf-1
177      write(21,'(3e26.16e3)')w(j),dble(f(j)),dimag(f(j))
178   enddo
179   close(21)
180
181   return
182 end subroutine dft_write_center0
```

4.1.3 MKL を利用した離散フーリエ変換の計算例

実際に離散フーリエ変換をプログラム上で実行してみます. 波の形は Eq. (2.1.3) や Eq. (3.48) で解いた時と同じ,

$$f(x) = \cos(2\pi 3x) \tag{4.9}$$

とします.

- **離散フーリエ変換の区間と $f(x)$ の周期が一致する場合**

  波の周期と x 軸の範囲 X が整数倍なので, この解析解は Eq. (3.55) より,

$$f(k_n) = \frac{N}{2}\left(\delta_{n,q} + \delta_{n,-q}\right) \tag{4.10}$$

$$= \frac{N}{2}\left(\delta_{k_n,k_q} + \delta_{k_n,k_{-q}}\right) \tag{4.11}$$

  となります. 具体的に $X = [0, 5], N = 200, K = 1/\Delta x = 40$ として計算することを考えると $q = 15$ なので, $\pm q\Delta k = \pm 3$ となります. よって,

$$f(k_n) = 100\left(\delta_{k_n,3} + \delta_{k_n,-3}\right) \tag{4.12}$$

  となります. 数値的に DFT を実行した結果は Fig. 5 のようになります. 数値計算上でも Eq. (4.12) の通りになっていることが確認できます.

- **離散フーリエ変換の区間と $f(x)$ の周期が一致しない場合**

  Eq. (4.9) で DFT の区間が波の周期の無理数倍である時かつ, 区間が $x = [0, X]$ の場合, 解析解は

$$f(k_n) = \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} \tag{4.13a}$$

$$= \frac{1}{2}\left(\frac{e^{i2\pi 3X} - 1}{e^{-i2\pi(k_n-3)\Delta x} - 1} + \frac{e^{-i2\pi 3X} - 1}{e^{-i2\pi(k_n+3)\Delta x} - 1}\right) \tag{4.13b}$$

  となります. 数値的に DFT を実行した結果は Fig. 6 のようになります. 図中でフーリエ変換結果を比較している通り Eq. (4.13) と解析解が一致していることが分かります.

  また, Eq. (4.9) で DFT の区間が波の周期の無理数倍である時かつ, 区間が $x = [-X/2, X/2]$ の場合, 解析解は Eq. (3.63) で見た通り

$$f(k_n) = (-1)^n \cdot \frac{1}{2} \cdot \left(e^{-i\pi 3X}\frac{e^{i2\pi 3X} - 1}{e^{-i2\pi(k_n-3)\Delta x} - 1} + e^{i\pi 3X}\frac{e^{-i2\pi 3X} - 1}{e^{-i2\pi(k_n+3-K)\Delta x} - 1}\right) \tag{4.14}$$

  となります. 数値的に DFT を実行した結果は Fig. 7 のようになります. 図中でフーリエ変換結果を比較している通り Eq. (4.14) と解析解が一致していることが分かります.

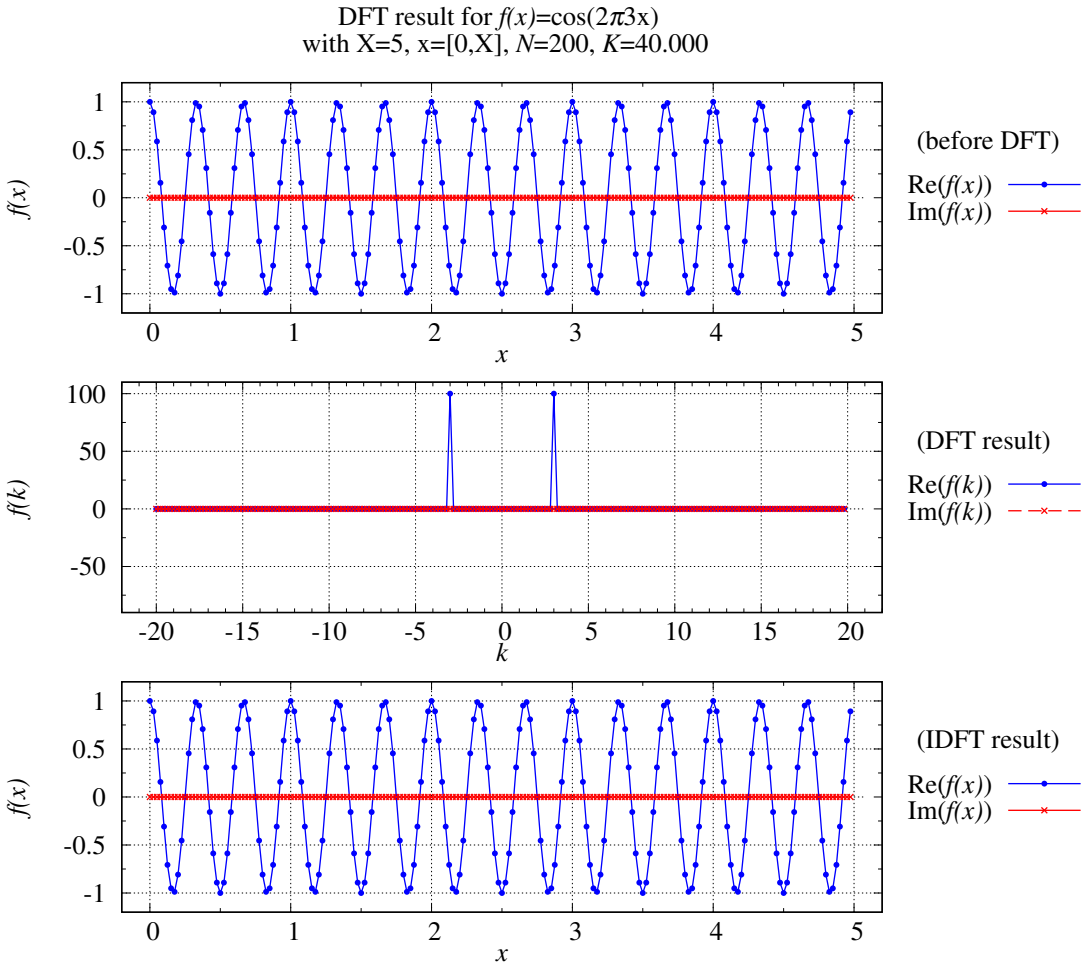

Figure 5: Eq. (4.9) の DFT 結果. $X = [0, 5], N = 200, K = 1/\Delta x = 40$ として, プログラム (1) を用いて計算.

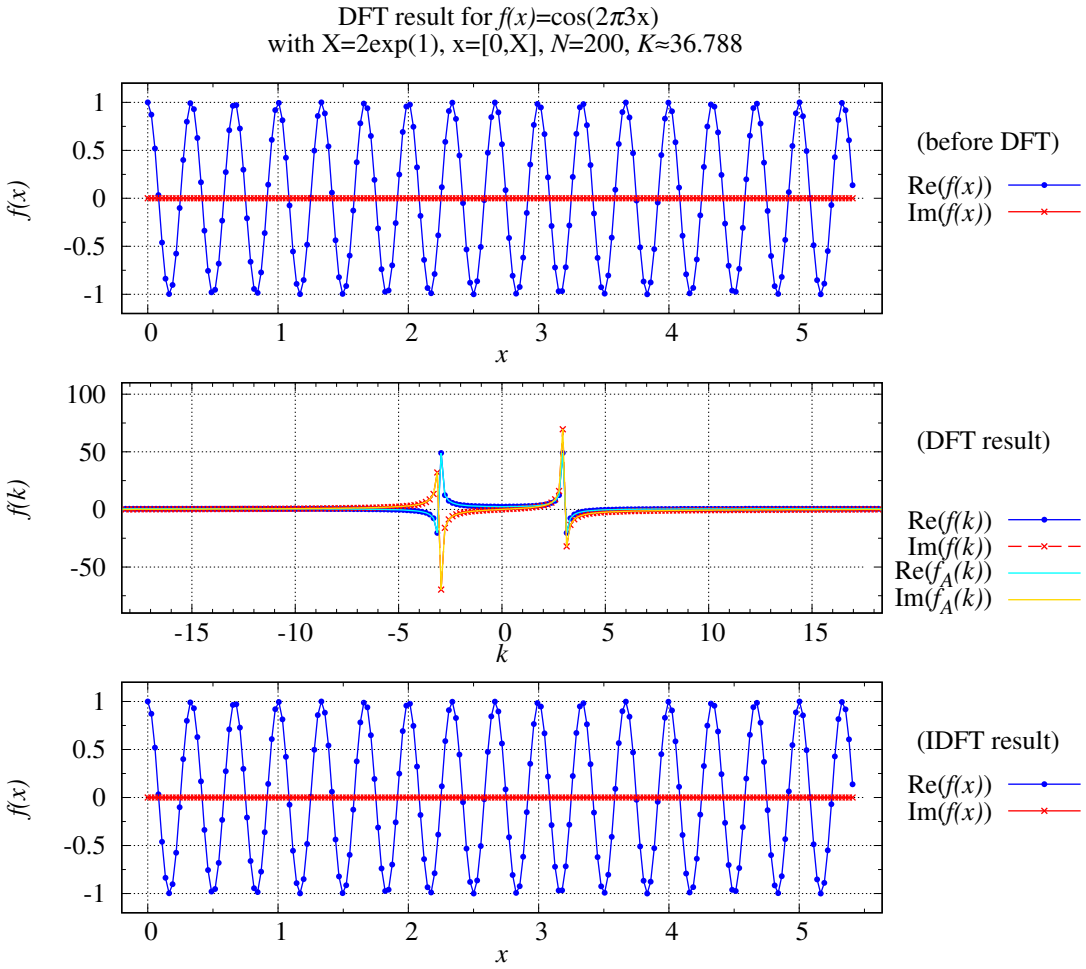

Figure 6: Eq. (4.9) の DFT 結果. $X = [0, 2\exp(1)], N = 200, K = 1/\Delta x \approx 36.8$ として, プログラム (1) を用いて計算. 図中 $f_A(k)$ は, 解析解 (4.13) を表す.

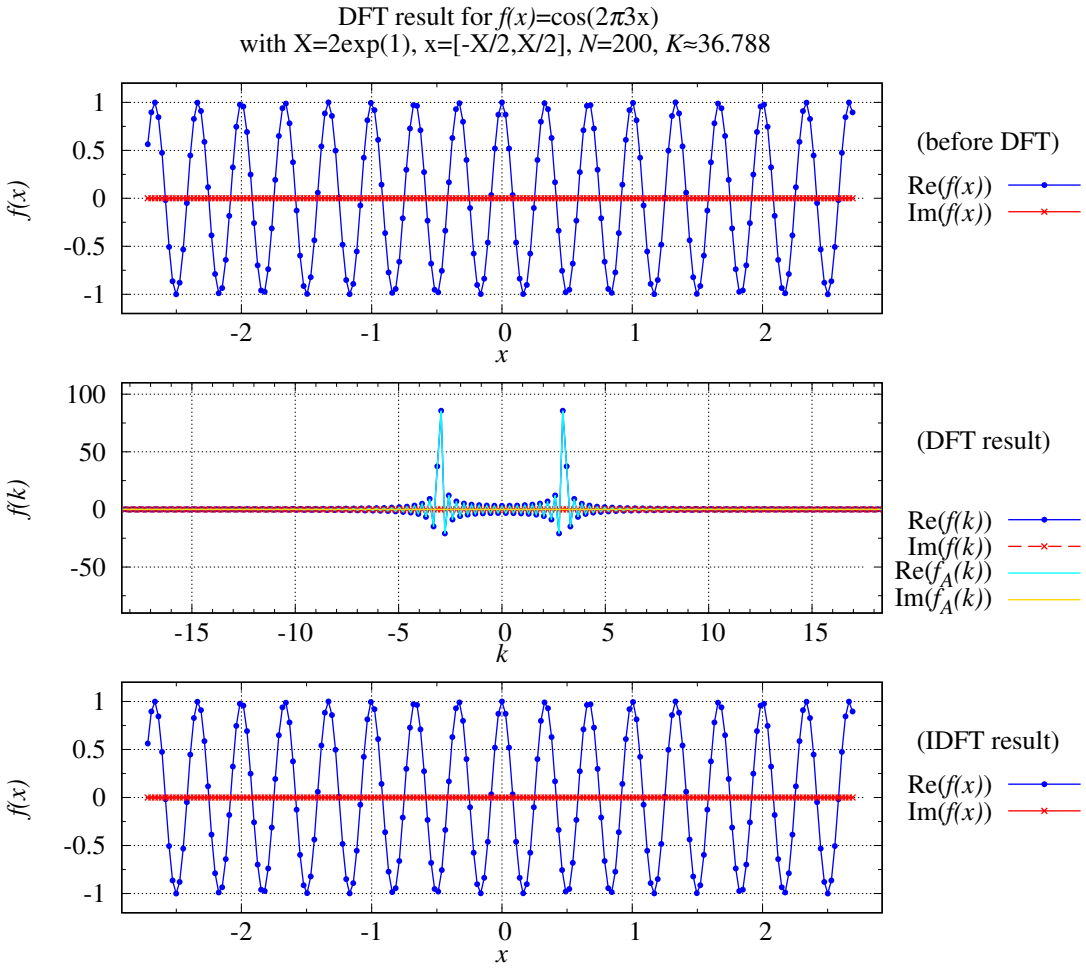

Figure 7: Eq. (4.14) の DFT 結果. $X = [-\exp(1), \exp(1)], N = 200, K = 1/\Delta x \approx 36.8$ として, プログラム (5) を用いて計算. 図中 $f_A(k)$ は, 解析解 (4.14) を表す.

4.2 Intel MKL を利用した畳み込み

4.2.1 Intel MKL を利用した畳み込みの計算例

ここでは項 3.2.4 で計算した具体的に畳み込みを計算します. つまり,

$$f(x) = x^5 e^{-x^2} \tag{4.15}$$

$$g(x) = e^{-4x^2} \tag{4.16}$$

の離散畳み込み

$$h_A(x_m) \equiv (f * g)_{\mathrm{d}}(x_m) \tag{4.17}$$

$$= \sum_p f(x_p) g(x_m - x_p) \tag{4.18}$$

$$= \frac{1}{3125}\sqrt{\frac{\pi}{5}} x_m \Big(1024x_m^4 + 1600x_m^2 + 375\Big) e^{-4x_m^2/5} \cdot \frac{1}{\Delta x} \tag{4.19}$$

を考えます. 途中計算で現れる形も, 項 2.2.3 と同じように示せば

$$f_A(k) = -\frac{i\pi^{3/2}}{4}(4\pi^4 k^5 - 20\pi^2 k^3 + 15k) e^{-\pi^2 k^2} \cdot \frac{1}{\Delta x} \tag{4.20}$$

$$g_A(k) = \frac{1}{2}\sqrt{\pi} e^{-\pi^2 k^2/4} \cdot \frac{1}{\Delta x} \tag{4.21}$$

となります.

離散フーリエ変換の結果は Fig. 8 となります.

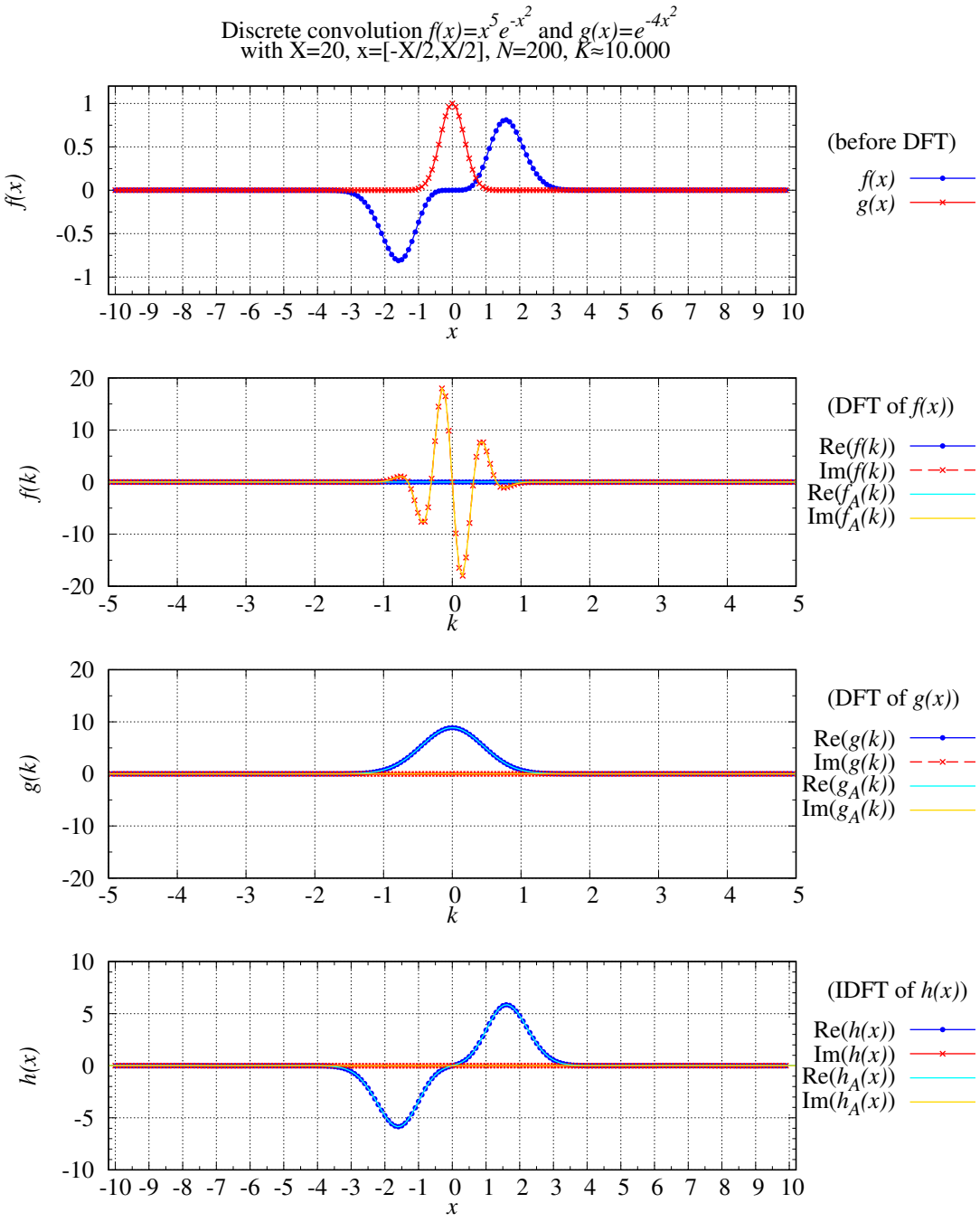

Figure 8: Eq. (4.17) に至るまでの計算過程. $X = [-10, 10], N = 200, K = 1/\Delta x \approx 10$ として, プログラム (6) を用いて計算. 図中 $f_A(k), g_A(k), h_A(k)$ は, 解析解 (4.14) を表す. 解は Eq. (3.80)Eq. (3.82)

4.3 DFT の計算速度

4.3.1 DFT の高速化

さて, DFT のコード (1) を見ると, 1 回の DFT ごとに Descrepter 等を作成しています. 複数回 DFT を実行する場合, これは一回だけで良いはずなのでその分計算時間が掛かってしまいそうです. 以下のコードを考えて比較してみましょう. 基本的にはコード (1) と同じですが, サブルーチン dft の中を取り出して main の中に移しています. (2) の中に, main の中の重要な部分だけを取り出して記述しました. また, 並列計算の影響もなくすために 1 スレッドだけで比較しています.

Listing 2: 離散フーリエ変換の点数 $N_p = 2^{18}$ として 10×100 回実行した時の計算時間を測定するプログラム

```
1
2   call mkl_set_num_threads(1) ! Number of threads for MKL parallel computation
3
4   ! DFT preparation with double-complex type of size N array
5   Status = DftiCreateDescriptor(hand,DFTI_DOUBLE,DFTI_COMPLEX,1,Np)
6   Status = DftiSetValue(hand,DFTI_FORWARD_SCALE,1d0)
7   Status = DftiSetValue(hand,DFTI_BACKWARD_SCALE,1d0/dble(Np))
8   Status = DftiCommitDescriptor(hand)
9
10  !---------------------------
11  f0=f
12  f1=f
13  time_fast=0e0
14  time_simp=0e0
15  do j=1,100
16     write(6,*)j
17     time_fast0=0e0
18     time_simp0=0e0
19     time_fast1=0e0
20     time_simp1=0e0
21
22     call cpu_time(time_simp0)
23     do i=1,10
24        call dft(Np,f0,"forward")
25        call dft(Np,f0,"backward")
26     enddo
27     call cpu_time(time_simp1)
28     time_simp = time_simp + time_simp1-time_simp0
29
30     call cpu_time(time_fast0)
31     do i=1,10
32        Status = DftiComputeForward(hand,f1)
33        Status = DftiComputeBackward(hand,f1)
34     enddo
35     call cpu_time(time_fast1)
36     time_fast = time_fast + time_fast1-time_fast0
37  enddo
38
39  !----------------------------
40  ! Free DFT settings
41  Status = DftiFreeDescriptor(hand)
42
43  write(6,*)"time simple [CPUsec] ",time_simp
44  write(6,*)"time fast [CPUsec] ",time_fast
```

実行結果は

Listing 3: 計算速度比較結果

```
1  time simple [CPUsec] 5.71431065
2  time fast [CPUsec] 4.00213337
```

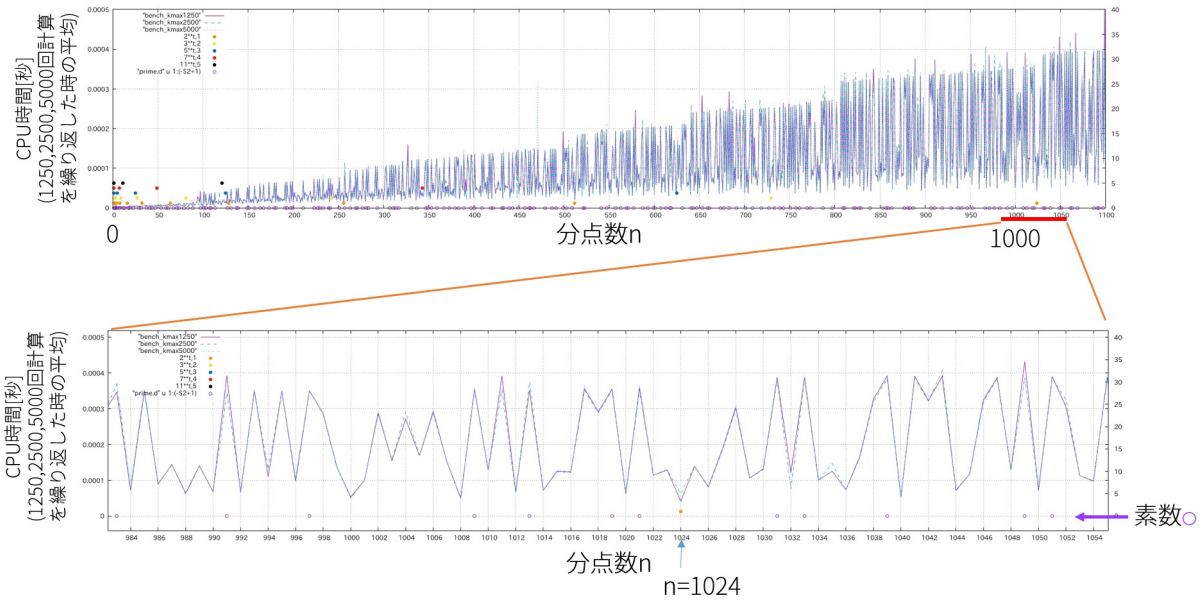

Figure 9: 同一計算機上でのベンチマーク結果. コンパイラ：ifort (IFORT) 16.0.2 20160204, MKL:/opt/intel/compilers_and_libraries_2016.2.181, 計算に用いたスレッド数は 1.

となりました. 可読性のために使用した dft 関数を使わなければ, 約 0.7 倍の計算時間となり高速化されます. 可読性を重視するか速度を重視するかは状況次第ですので, ここでは結果を出すにとどめておきます.

4.3.2 DFT の分点数に対する計算速度

MKL のルーチン（MKL は結局 fftw と呼ばれるルーチンを利用していますが…）は, 任意の数に対してフーリエ変換を実行することが可能です. つまり, 高速フーリエ変換だけサポートしているのではなく, 分点数が奇数でも素数でも使用可能です. ですが, 分点数がちょうど $2, 3, 5, 7, 11$ の乗数に等しいときに自動的に高速フーリエ変換が実行されます.

計算速度を調べたのが Fig. 9 です. 横軸に分点数, 縦軸に計算時間を示しています.

4.3.3 並列計算時の比較

さて, MKL は並列計算をサポートしています. 離散フーリエ変換も例外ではなく, 並列に CPU を使用することによって高速化することができます. 並列計算の準備は以下の一行プログラム内で指定するだけで良く, あとは勝手に実行されます.

Listing 4: MKL ルーチン実行時に使用するスレッド数の設定

```
1 call mkl_set_num_threads(Nthreads)
```

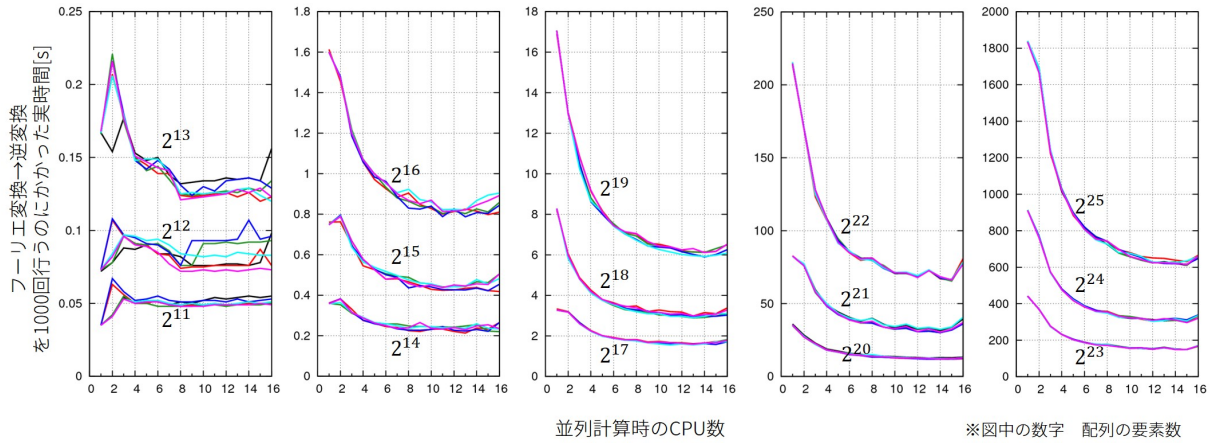

Figure 10: 同一計算機上でのベンチマーク結果. 計算時間の CPU 個数 (8 コア 16 スレッド) と配列の要素数依存性.

プログラムが始まったらすぐに指定すると良いでしょう. もしくは, MKL ルーチンの実行直前が良いでしょう.

Fig. 10 に, 並列計算の速度, 配列のサイズと実行時間の関係を示します.

”計算 1 回”とは（順方向→逆方向フーリエ変換）の 1 セットを”1 回”としています. 8 コア 16 スレッドの環境で実行したので, 図の場合では物理的な CPU の個数 8 個で計算速度はおおよそ打ち止めになっています.

Reference

[1] インテル (R) マス・カーネル・ライブラリーリファレンスマニュアル (2006 年)

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling and Brian P. Flannery, (丹慶勝市, 奥村晴彦, 佐藤俊郎, 小林誠 訳), 『ニューメリカルレシピ・イン・シー 日本語版—C 言語による数値計算のレシピ』, 技術評論社 (1993)
英語かつ過去版で良ければ，公式から無料で閲覧することができます (http://numerical.recipes/oldverswitcher.html).

A フーリエ変換→逆変換で元に戻ることの証明

A.1 連続の場合

$$
\begin{aligned}
f(x) &= \int_{-\infty}^{\infty} d\tilde{k} f(\tilde{k}) e^{i2\pi\tilde{k}x} && \text{(A.1)}\\
&= \int_{-\infty}^{\infty} d\tilde{k} \left[\int_{-\infty}^{\infty} dx' f(x') e^{-i2\pi\tilde{k}x'} \right] e^{i2\pi\tilde{k}x} && \text{(A.2)}\\
&= \int_{-\infty}^{\infty} dx' f(x') \int_{-\infty}^{\infty} d\tilde{k} e^{i2\pi\tilde{k}(x-x')} && \text{(A.3)}\\
&= \int_{-\infty}^{\infty} dx' f(x') \delta(x-x') && \text{(A.4)}\\
&= f(x) && \text{(A.5)}
\end{aligned}
$$

A.2 離散の場合

$$
\begin{aligned}
&\sum_{m=0}^{N-1} e^{-i2\pi k_{n'}(x_m - x_{\text{off}})} \cdot f(x_m) && \text{(A.6a)}\\
=&\sum_{m=0}^{N-1} e^{-i2\pi k_{n'}(x_m - x_{\text{off}})} \cdot \sum_{n=0}^{N-1} f(k_n) e^{i2\pi k_n (x_m - x_{\text{off}})} && \text{(A.6b)}\\
=&\sum_{n=0}^{N-1} f(k_n) \sum_{m=0}^{N-1} e^{i2\pi\cdot(k_n - k_{n'})\cdot(x_m - x_{\text{off}})} && \text{(A.6c)}\\
=&\sum_{n=0}^{N-1} f(k_n) \sum_{m=0}^{N-1} e^{i2\pi\cdot(n-n')\Delta k\cdot m\Delta x} && \text{(A.6d)}\\
=&\sum_{n=0}^{N-1} f(k_n) N\delta_{n,n'} && \text{(A.6e)}\\
=&N\cdot f(k_{n'}) && \text{(A.6f)}
\end{aligned}
$$

となります.Eq. (4.7) と, 等比級数の知識である

$$
\sum_{n=0}^{N-1} e^{i2\pi na} = \begin{cases} N, & (a=0) \\ \dfrac{e^{i2\pi aN}-1}{e^{i2\pi a}-1}, & (a\neq 0) \end{cases} \qquad \text{(A.7)}
$$

を用いて,

$$
\sum_{m=0}^{N-1} e^{i2\pi m(n-n')/N} = N\delta_{n,n'} \qquad \text{(A.8)}
$$

を用いています. ここで, $\delta_{i,j}$ はクロネッカーのデルタで

$$\delta_{i,j} = \begin{cases} 1, & (i = j) \\ 0, & (i \neq j) \end{cases} \tag{A.9}$$

です.

B DFT の具体例で考えた無理数倍について

項 3.1.6 で考えた, 離散フーリエ変換の区間が波の周期の無理数倍を仮定したことについてコメントします. 今考えたい計算は以下の通りです.

$$\begin{aligned} f(k_n) &= \sum_{m=0}^{N-1} f(x_m) e^{-i2\pi nm/N} && \text{(B.1a)} \\ &= \sum_{m=0}^{N-1} \cos(2\pi 3 \cdot m\frac{X}{N}) e^{-i2\pi nm/N} && \text{(B.1b)} \\ &= \frac{1}{2}\sum_{m=0}^{N-1} \left[e^{i2\pi m\frac{(3X-n)}{N}} + e^{-i2\pi m\frac{(3X+n)}{N}} \right] && \text{(B.1c)} \\ &= \frac{1}{2}\left(\frac{e^{i2\pi(3X-n)} - 1}{e^{i2\pi(3X-n)/N} - 1} + \frac{e^{-i2\pi(3X+n)} - 1}{e^{-i2\pi(3X+n)/N} - 1} \right) \quad ...? && \text{(B.1d)} \\ &= \frac{1}{2}\left(\frac{e^{i2\pi 3N\Delta x} - 1}{e^{-i2\pi(k_n-3)\Delta x} - 1} + \frac{e^{-i2\pi 3N\Delta x} - 1}{e^{-i2\pi(k_n+3)\Delta x} - 1} \right) && \text{(B.1e)} \end{aligned}$$

となります. Eq. (B.1c) から Eq. (B.1d) に移る際に $(3X \pm n)/N$ はゼロではないという条件を使用しています. 右辺の分母に注目すると, $k_n = \pm 3$ の時にどちらかの分母がゼロとなり発散, 計算が不能になります. つまり, Eq. (B.1c) ではどう頑張っても N が有限ならば値も有限ですが, Eq. (B.1d) では無限があり得るといっています. 明らかにおかしいですので, 本来 Eq. (B.1d) への変形は行うことができません.

つまりこの条件が言っているのはどんなに偶然でも $k_n = n\Delta k$ が 3 になった場合には Eq. (B.1e) を使用することはできないということです. すなわち, $f(x)$ の周期 X_p の整数倍となるときは適当な整数 n' を用いて

$$k_n = n\Delta k = n\frac{1}{X} = \frac{n'}{X_p} \tag{B.2}$$

が満たされるときは, Eq. (B.1e) は使えない, X について X が X_p の整数倍ではないときに

$$X = \frac{n}{n'} X_p \tag{B.3}$$

は使えないことを意味します. 以上をまとめると, Eq. (B.1e) が使用できるのは, 区間 $x = [0, X]$ の X は波の波数の無理数倍じゃなきゃダメです, ということです.

実際に離散フーリエ変換を行う上で無理数倍の区間指定はほとんど行いません. そのため, 具体例として挙げるのは適切ではないと判断し, 項 3.1.6 ではこれ以上の式変形を行っておりません.

C 任意区間の離散フーリエ変換プログラム

Listing 5: Intel MKL を用いた Fortran90 のプログラム例 (位置を範囲変更した一般の場合)

```
program main
  implicit none
  integer::m
  double precision::dx,Xrange,xa,xb
  double precision,allocatable::x(:)
  integer::n
  double precision::dk,Krange
  double precision,allocatable::k(:)
  integer::Np
  complex(kind(0d0)),allocatable::f(:)
  complex(kind(0d0)),external::func

  Np = 200 ! Number of DFT points
  xa = -exp(1d0)
  xb = exp(1d0)
  Xrange = xb-xa ! x=[xa, xa + Xrange]

  ! Initialize and allocation for DFT
  allocate(x(0:Np-1),k(0:Np-1),f(0:Np-1))
  x = 0d0
  k = 0d0
  f = dcmplx(0d0,0d0)

  ! Define position x
  dx = Xrange / Np
  call dft_abscissa_shift(Np, x, dx, xa) ! x=[xa, xa + Xrange]

  ! Calculate k abscissa
  Krange = 1d0/dx
  dk = Krange / Np
  call dft_abscissa(Np, k, dk) ! k=[0, Krange]

  ! Calculate function f at x=x(m)
  do m = 0,Np-1
     f(m) = func(x(m))
  enddo

  ! Write down f(x)
  call dft_write("before_fx.d",Np,x,f)

  ! Forward dft
  call dft(Np,f,"forward")

  ! Write down f(k)
  call dft_writefk_shift("fk.d",Np,k,f,xa) ! output k=[0,Krange]
  call dft_writefk_center0_shift("fk_center0.d",Np,k,f,xa) ! output k=[-Krange/2,Krange/2]

  ! Backward dft
  call dft(Np,f,"backward")

  ! Write down f(x)
  call dft_write("after_fx.d",Np,x,f)

  stop
end program main

complex(kind(0d0)) function func(x)
  implicit none
  double precision,intent(in)::x

  double precision::pi=dacos(-1d0)

  func=dcmplx(dcos(2d0*pi*3d0*x),0d0)

  return
end function func

!---------------------------------

subroutine dft_abscissa(N, w, dw)
  implicit none
  integer,intent(in)::N
  double precision,intent(in)::dw
  double precision,intent(out)::w(0:N-1)
```

```
75   !
76   !w=[0, W]
77   ! w(0) = 0
78   ! w(N-1) = (N-1)*dw = W-dw
79   !
80   integer::j
81
82   do j = 0,N-1
83      w(j) = j*dw
84   enddo
85
86   return
87 end subroutine dft_abscissa
88
89 subroutine dft_abscissa_shift(N, w, dw, wa)
90   implicit none
91   integer,intent(in)::N
92   double precision,intent(in)::dw,wa
93   double precision,intent(out)::w(0:N-1)
94   !
95   !w=[wa, wa + W]
96   ! w(0) = wa
97   ! w(N-1) = (N-1)*dw = W - dw + wa
98   !
99   integer::j
100
101  do j = 0,N-1
102     w(j) = j*dw + wa
103  enddo
104
105  return
106 end subroutine dft_abscissa_shift
107
108 include "/opt/mkl/include/mkl_dfti.f90"
109 subroutine dft(N,f,FB)
110   use MKL_DFTI
111   implicit none
112   integer,intent(in)::N
113   complex(kind(0d0)),intent(inout)::f(0:N-1)
114   character(*),intent(in)::FB
115   !
116   !sikinote
117   ! author : sikino
118   ! date : 2022/06/11
119   !
120   !n: Number of DFT points.
121   !f(i): value of data at i
122   !FB: "forward" -> Forward DFT
123   ! "backward" -> Backward DFT
124   integer::Status
125   TYPE(DFTI_DESCRIPTOR),POINTER::hand
126
127   Status = DftiCreateDescriptor(hand,DFTI_DOUBLE,DFTI_COMPLEX,1,N)
128   Status = DftiSetValue(hand,DFTI_FORWARD_SCALE,1d0)
129   Status = DftiSetValue(hand,DFTI_BACKWARD_SCALE,1d0/dble(N))
130   Status = DftiCommitDescriptor(hand)
131   if(trim(FB)=="forward")then
132      Status = DftiComputeForward(hand,f)
133   elseif(trim(FB)=="backward")then
134      Status = DftiComputeBackward(hand,f)
135   else
136      write(6,*)"DFT string different"
137      stop
138   endif
139   Status = DftiFreeDescriptor(hand)
140
141   return
142 end subroutine dft
143
144 subroutine dft_write(fname,N,w,f)
145   implicit none
146   character(*),intent(in)::fname
147   integer,intent(in)::N
148   double precision,intent(in)::w(0:N-1)
149   complex(kind(0d0)),intent(in)::f(0:N-1)
150
151   ! w = [wa,W]
152   ! w(0) = wa
153   ! --> output range w=[wa,wa+W]
154   integer::j
155
```

```
156   open(21,file=trim(fname))
157   do j = 0,N-1
158      write(21,'(3e26.16e3)')w(j),dble(f(j)),dimag(f(j))
159   enddo
160   close(21)
161
162   return
163 end subroutine dft_write
164
165 subroutine dft_writefk_shift(fname,N,k,f,xa)
166   implicit none
167   character(*),intent(in)::fname
168   integer,intent(in)::N
169   double precision,intent(in)::k(0:N-1),xa
170   complex(kind(0d0)),intent(in)::f(0:N-1)
171
172   ! k = [0,K]
173   ! k(0) = 0
174   ! --> output range k=[0,K]
175   integer::nn
176   complex(kind(0d0))::tmp
177   double precision,parameter::pi=dacos(-1d0)
178
179   open(21,file=trim(fname))
180   do nn = 0,N-1
181      tmp = exp(dcmplx(0d0,2d0*pi*k(nn)*xa))*f(nn)
182      write(21,'(3e26.16e3)')k(nn),dble(tmp),dimag(tmp)
183   enddo
184   close(21)
185
186   return
187 end subroutine dft_writefk_shift
188
189 subroutine dft_writefk_center0_shift(fname,N,k,f,xa)
190   implicit none
191   character(*),intent(in)::fname
192   integer,intent(in)::N
193   double precision,intent(in)::k(0:N-1),xa
194   complex(kind(0d0)),intent(in)::f(0:N-1)
195
196   ! k = [0,K]
197   ! k(0) = 0
198   ! --> output range k=[-K/2,K/2]
199   integer::nn,Nhalf
200   double precision::dk,Krange
201   complex(kind(0d0))::tmp
202   double precision,parameter::pi=dacos(-1d0)
203
204   if(mod(N,2)==0)then
205      Nhalf = N/2
206   else
207      Nhalf = (N-1)/2
208   endif
209
210   dk = abs(k(1)-k(0))
211   Krange = N*dk
212
213   open(21,file=trim(fname))
214   do nn = Nhalf,N-1
215      tmp = exp(dcmplx(0d0,2d0*pi*k(nn)*xa))*f(nn)
216      write(21,'(3e26.16e3)')k(nn)-Krange,dble(tmp),dimag(tmp)
217   enddo
218   do nn = 0,Nhalf-1
219      tmp = exp(dcmplx(0d0,2d0*pi*k(nn)*xa))*f(nn)
220      write(21,'(3e26.16e3)')k(nn),dble(tmp),dimag(tmp)
221   enddo
222   close(21)
223
224   return
225 end subroutine dft_writefk_center0_shift
```

D 任意区間の離散フーリエ変換を利用する畳み込みプログラム

いくつかのサブルーチンは Program (5) と共通なので, 省略しています.

Listing 6: Intel MKL を用いた畳み込みのプログラム例 (位置を範囲変更した一般の場合)

```
program main
  implicit none
  integer::m
  double precision::dx,Xrange,xa,xb
  double precision,allocatable::x(:)
  integer::n
  double precision::dk,Krange
  double precision,allocatable::k(:)
  integer::Np
  complex(kind(0d0)),allocatable::f(:),g(:),h(:)
  complex(kind(0d0)),external::func,gfunc
  double precision,parameter::pi=dacos(-1d0)

  Np = 200 ! Number of DFT points
  xa = -10d0
  xb = 10d0
  Xrange = xb - xa ! x=[xa,xa+Xrange]

  ! Initialize and allocation for DFT
  allocate(x(0:Np-1),k(0:Np-1),f(0:Np-1))
  x = 0d0
  k = 0d0
  f = dcmplx(0d0,0d0)

  allocate(g(0:Np-1),h(0:Np-1))
  g = dcmplx(0d0,0d0)
  h = dcmplx(0d0,0d0)

  ! Define position x
  dx = Xrange / Np
  call dft_abscissa_shift(Np, x, dx, xa) ! x=[xa, xa + Xrange]

  ! Calculate k abscissa
  Krange = 1d0/dx
  dk = Krange / Np
  call dft_abscissa(Np, k, dk) ! k=[0, Krange]

  ! Calculate function f at x=x(m)
  do m = 0,Np-1
     f(m) = func(x(m))
     g(m) = gfunc(x(m))
  enddo

  ! Write down f(x)
  call dft_write("before_fx.d",Np,x,f)
  call dft_write("before_gx.d",Np,x,g)

  ! Forward dft
  call dft(Np,f,"forward")
  call dft(Np,g,"forward")

  ! Multiply f(k) and g(k) and add the phase factor in k-space
  do n = 0,Np-1
     h(n) = exp(dcmplx(0d0,-2*pi*k(n)*xa))*f(n)*g(n)
  enddo

  ! Write down f(k), g(k), h(k)
  call dft_writefk_center0_shift("fk.d",Np,k,f,xa) ! output k=[-Krange/2,Krange/2]
  call dft_writefk_center0_shift("gk.d",Np,k,g,xa) ! output k=[-Krange/2,Krange/2]
  call dft_writefk_center0_shift("hk.d",Np,k,h,xa) ! output k=[-Krange/2,Krange/2]

  ! Backward dft
  call dft(Np,h,"backward")

  ! Write down h(x)
  call dft_write("after_hx.d",Np,x,h)

  stop
end program main

complex(kind(0d0)) function func(x)
  implicit none
```

```
73   double precision,intent(in)::x
74
75   func=dcmplx(x**5*exp(-x**2),0d0)
76
77   return
78 end function func
79
80 complex(kind(0d0)) function gfunc(x)
81   implicit none
82   double precision,intent(in)::x
83
84   gfunc=dcmplx(exp(-4*x**2),0d0)
85
86   return
87 end function gfunc
```