1　フーリエ級数

1.1　ベクトル演算

3 次元空間内のベクトル演算について次の図 1 のような状況を考える.

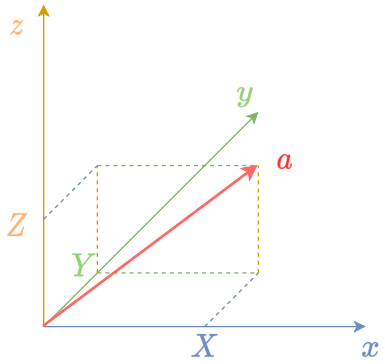

図 1

このとき x, y, z 軸方向の単位ベクトルをそれぞれ $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ とすれば, ベクトル $\boldsymbol{a}$ は

$$\boldsymbol{a} = X\boldsymbol{i} + Y\boldsymbol{j} + Z\boldsymbol{k} \tag{1}$$

として表すことができる. このように任意の 3 次元ベクトルは 3 つの単位ベクトルの実数倍の和として表現することができ, 単位ベクトル同士は互いに直交しているため同じ単位ベクトル $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ を用いる場合式 (1) 以外の方法で表現することはできない. これは n 次元ベクトルについても同様で, 互いに直交した n 個の単位ベクトルを使うことである n 次元ベクトル $\boldsymbol{a}$ を一意に表現することが可能である. また, 式 (1) の係数 X, Y, Z はそれぞれ $X = \boldsymbol{a} \cdot \boldsymbol{i}, Y = \boldsymbol{a} \cdot \boldsymbol{j}, Z = \boldsymbol{a} \cdot \boldsymbol{k}$ により与えられる.

1.2　関数への拡張

連続な関数 $f(x)$ の区間 $[a, b]$ において, この区間を n 個の微小な区間に分け, その距離を Δx としたときに関数 $f(x)$ が $n+1$ 次元ベクトル $\Big[f(a), f(a+\Delta x), f(a+2\Delta x), \ldots, f(a+n\Delta x)(= f(b))\Big]$ を作ると考える. このとき関数 $f(x)$ と関数 $g(x)$ が

$$\sum_{k=0}^{n} f(a+k\Delta x)g(a+k\Delta x) = 0$$

を満たせば, これはこの 2 つの関数が区間 $[a, b]$ で作り出すベクトルの内積が 0 であることを表し, このとき 2 つの関数が作り出すベクトルは直交している. これを $\Delta x \to 0$ とし, 積分として書けば

$$\int_a^b f(x)g(x)\mathrm{d}x = 0$$

となり, これを関数の直交条件と考えることができる. 従って, 関数 $f(x)$ の区間 $[a, b]$ について点を n 個取った場合, この関数によって生成されるベクトルは直交する n 個の関数によって生成されるベクトルの実数倍の

和として表現することができるということであり, これはすなわち $f(x)$ を互いに直交する関数群を用いて近似することが可能であるということである.

1.3 直交関数系

直交関数系には三角関数がよく使われ、例えば区間 $[-L, L]$ において sin 関数と sin 関数は

$$\begin{aligned}
&\int_{-L}^{L} \sin\left(\frac{n\pi x}{L}\right) \sin\left(\frac{m\pi x}{L}\right) \mathrm{d}x \quad (n, m = 0, 1, 2, \dots), (n \neq m) \\
&\quad = \frac{1}{2} \int_{-L}^{L} \cos\left(\frac{\pi x(n-m)}{L}\right) - \cos\left(\frac{\pi x(n+m)}{L}\right) \mathrm{d}x \\
&\quad = \frac{1}{2} \left[\left(\frac{L}{\pi(n-m)} \sin\left(\frac{\pi x(n-m)}{L}\right)\right) - \left(\frac{L}{\pi(n+m)} \sin\left(\frac{\pi x(n+m)}{L}\right)\right)\right]_{-L}^{L} \\
&\quad = 0 \quad \left(\because \sin\left(\frac{\pi x(n-m)}{L}\right)\Big|_{x=L} = 0 \right)
\end{aligned}$$

cos 関数と cos 関数に関しては同様にして

$$\begin{aligned}
&\int_{-L}^{L} \cos\left(\frac{n\pi x}{L}\right) \cos\left(\frac{m\pi x}{L}\right) \mathrm{d}x \quad (n, m = 0, 1, 2, \dots), (n \neq m) \\
&\quad = \frac{1}{2} \int_{-L}^{L} \cos\left(\frac{\pi x(n-m)}{L}\right) + \cos\left(\frac{\pi x(n+m)}{L}\right) \mathrm{d}x \\
&\quad = 0
\end{aligned}$$

cos 関数と sin 関数はその積が奇関数なので $[-L, L]$ では 0 について対称性があるため

$$\int_{-L}^{L} \cos\left(\frac{n\pi x}{L}\right) \sin\left(\frac{m\pi x}{L}\right) \mathrm{d}x = 0 \quad (n, m = 0, 1, 2, \dots), (n \neq m)$$

より三角関数が直交関数系であることが分かる.

1.4 フーリエ級数展開

三角関数の直交関数系を用いることで n 個の互いに直交する関数を容易に作ることができ, 適当な係数 a_n, b_n を用いて $f(x)$ を区間 $[-L, L]$ で

$$f(x) \sim \sum_{n=0}^{\infty} \left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right)$$

と表すことができる. $\sim$ は近似を示している. 係数 a_n, b_n は再現するベクトルと単位ベクトルの内積を取ればよいので,

$$\begin{aligned}
a_n &= \int_{-L}^{L} f(x) \cos\left(\frac{n\pi x}{L}\right) \mathrm{d}x \\
b_n &= \int_{-L}^{L} f(x) \sin\left(\frac{n\pi x}{L}\right) \mathrm{d}x
\end{aligned}$$

となるが, 単位ベクトルは規格化されている必要があるため

$$\begin{aligned}
&\int_{-L}^{L} \cos^2\left(\frac{n\pi x}{L}\right) \mathrm{d}x \quad (n = 0, 1, 2, \dots) \\
&\quad = \frac{1}{2}\int_{-L}^{L} 1 + \cos\left(\frac{2n\pi x}{L}\right) \mathrm{d}x \\
&\quad = \frac{1}{2}\left[x + \frac{L}{2n\pi}\sin\left(\frac{2n\pi x}{L}\right)\right]_{-L}^{L} \\
&\quad = L \\
&\int_{-L}^{L} \sin^2\left(\frac{n\pi x}{L}\right) \mathrm{d}x \quad (n = 0, 1, 2, \dots) \\
&\quad = \frac{1}{2}\int_{-L}^{L} 1 - \cos\left(\frac{2n\pi x}{L}\right) \mathrm{d}x \\
&\quad = L
\end{aligned}$$

より実際は

$$\begin{aligned}
a_n &= \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(\frac{n\pi x}{L}\right) \mathrm{d}x \\
b_n &= \frac{1}{L}\int_{-L}^{L} f(x)\sin\left(\frac{n\pi x}{L}\right) \mathrm{d}x
\end{aligned}$$

である. $n = 0$ について, この時

$$f(x) \sim \frac{\alpha}{2} + \sum_{n=1}^{\infty}\left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right)$$

として a_0 を改めて求めてみると

$$\begin{aligned}
a_0 &= \frac{1}{L}\left\{f(x)\cdot\cos\left(\frac{0\pi x}{L}\right)\right\} = \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(\frac{0\pi x}{L}\right) \mathrm{d}x \\
&= \frac{1}{L}\int_{-L}^{L} \frac{\alpha}{2} \mathrm{d}x \quad \left(\because a_n \cos\left(\frac{n\pi x}{L}\right)\cos\left(\frac{m\pi x}{L}\right) = 0, b_n \sin\left(\frac{n\pi x}{L}\right)\cos\left(\frac{m\pi x}{L}\right) = 0 \quad (n \neq m)\right) \\
&= \frac{\alpha}{2L}\cdot 2L = \alpha
\end{aligned}$$

より $\alpha = a_0$ となるため, 初項を $\dfrac{a_0}{2}$ として級数展開を記述することが多い. 以上のことをまとめると関数 $f(x)$ は区間 $[-L, L]$ について三角関数を用いて

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right) \tag{2}$$

ただし

$$a_n = \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(\frac{n\pi x}{L}\right) \mathrm{d}x \tag{3}$$

$$b_n = \frac{1}{L}\int_{-L}^{L} f(x)\sin\left(\frac{n\pi x}{L}\right) \mathrm{d}x \tag{4}$$

となり, これが関数 $f(x)$ のフーリエ級数展開である.

1.5 複素フーリエ級数展開

$\omega = \dfrac{\pi}{L}$ とおいたときの三角関数で与えられたフーリエ級数展開

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \Big(a_n \cos(n\omega x) + b_n \sin(n\omega x) \Big)$$

はオイラーの公式 $e^{it} = \cos t + i \sin t$ より与えられる三角関数の表現

$$\cos t = \frac{e^{it} + e^{-it}}{2}$$
$$\sin t = \frac{e^{it} - e^{-it}}{2i} = -i\frac{e^{it} - e^{-it}}{2}$$

を用いて

$$\begin{aligned} f(x) &\sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \Big(a_n \frac{e^{in\omega x} + e^{-in\omega x}}{2} + b_n(-i)\frac{e^{in\omega x} - e^{-in\omega x}}{2} \Big) \\ &\sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \Big(\frac{a_n - ib_n}{2} e^{in\omega x} + \frac{a_n + ib_n}{2} e^{-in\omega x} \Big) \end{aligned}$$

と表すことができる. ここで $c_n = \dfrac{a_n - ib_n}{2}$ とすれば

$$a_{-n} = \frac{1}{L}\int_{-L}^{L} f(x) \cos\left(\frac{(-n)\pi x}{L}\right) \mathrm{d}x = \frac{1}{L}\int_{-L}^{L} f(x) \cos\left(\frac{n\pi x}{L}\right) \mathrm{d}x = a_n$$
$$b_{-n} = \frac{1}{L}\int_{-L}^{L} f(x) \sin\left(\frac{(-n)\pi x}{L}\right) \mathrm{d}x = -\frac{1}{L}\int_{-L}^{L} f(x) \sin\left(\frac{n\pi x}{L}\right) \mathrm{d}x = -b_n$$

より

$$\sum_{n=1}^{\infty} c_n = \sum_{n=-\infty}^{-1} c_{-n} = \sum_{n=-\infty}^{-1} c_n^*$$

が得られるから,

$$\begin{aligned} f(x) &\sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \Big(c_n e^{in\omega x} + c_n^* e^{-in\omega x} \Big) \\ &\sim \frac{a_0}{2} + \sum_{n=1}^{\infty} c_n e^{in\omega x} + \sum_{n=-\infty}^{-1} c_n e^{in\omega x} \end{aligned}$$

と変形でき, $n = 0$ を $\dfrac{a_0}{2} = c_0$ とおけば結局

$$f(x) \sim \sum_{n=-\infty}^{\infty} c_n e^{in\omega x} \tag{5}$$

である. また, c_n はその定義から自明に

$$\begin{aligned} c_n &= \frac{a_n - ib_n}{2} \\ &= \frac{1}{2L}\left\{\int_{-L}^{L} f(x)\cos(n\omega x)\mathrm{d}x - i\int_{-L}^{L} f(x)\sin(n\omega x)\right\} \\ &= \frac{1}{2L}\left\{\int_{-L}^{L} f(x)\Big(\cos(n\omega x) - i\sin(n\omega x)\mathrm{d}x\Big)\right\} \\ &= \frac{1}{2L}\int_{-L}^{L} f(x)e^{-in\omega x}\mathrm{d}x \end{aligned} \tag{6}$$

である. c_n についてその絶対値と偏角 ϕ_n はそれぞれ

$$\begin{aligned} |c_n| &= \frac{1}{2}\sqrt{a_n^2 + b_n^2} \\ \phi_n &= \tan^{-1}\left(\frac{a_n}{-b_n}\right) \end{aligned}$$

であり, これより

$$c_n = |c_n|\, e^{i\phi}$$

を得ることができる.

2 フーリエ積分

フーリエ級数展開について $L \to \infty$ とした場合を考える.

$$f(x) \sim \sum_{n=-\infty}^{\infty} c_n e^{in\omega x}$$

について

$$c_n = \frac{1}{2L}\int_{-L}^{L} f(x)e^{-in\omega x}\mathrm{d}x$$

を代入すれば

$$f(x) \sim \sum_{n=-\infty}^{\infty}\left(\frac{1}{2L}\int_{-L}^{L} f(x)e^{-in\omega x}\mathrm{d}x\right)e^{in\omega x}$$

となる. ここで, $\dfrac{\pi}{L} = \omega$ としていたからこの式は

$$f(x) \sim \sum_{n=-\infty}^{\infty}\left(\frac{1}{2\pi}\int_{-L}^{L} f(x)e^{-in\omega x}\mathrm{d}x\right)\omega e^{in\omega x}$$

と書き換えることができる. L が大きくなるとき ω が微小になることに注意し, これを $\Delta\omega$ と書き直せば

$$f(x) \sim \sum_{n=-\infty}^{\infty}\left(\frac{1}{2\pi}\int_{-L}^{L} f(x)e^{-in\Delta\omega x}\mathrm{d}x\right)e^{in\Delta\omega x}\Delta\omega$$

であり, $L \to \infty$ のとき $\Delta\omega \to \mathrm{d}\omega$ であるから, これを積分に書き換えれば

$$f(x) \sim \frac{1}{2\pi} \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} f(x) e^{-i\omega x} \mathrm{d}x \right) e^{i\omega x} \mathrm{d}\omega \tag{7}$$

が得られる. また

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} \mathrm{d}x \tag{8}$$

と定義するとよく知られる非周期関数のフーリエ表現式を得ることができる.

$$f(x) \sim \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega x} \mathrm{d}\omega \tag{9}$$

3 フーリエ変換

非周期関数のフーリエ表現式における $F(\omega)$ は $f(x)$ のフーリエ変換として知られている.

3.1 フーリエ変換の性質

フーリエ変換には以下の性質がある. ただし $\mathcal{F}\Big[f(x)\Big] = F(\omega)$ とする.

3.1.1 直線性

a_1, a_2 が実数のとき

$$\mathcal{F}\Big[a_1 f_1(x) + a_2 f_2(x)\Big] = a_1 F_1(\omega) + a_2 F_2(\omega) \tag{10}$$

3.1.2 縮尺性

$a \neq 0$ の実数であるとき

$$\begin{aligned} \mathcal{F}\Big[f(ax)\Big] &= \int_{-\infty}^{\infty} f(ax) e^{-i\omega x} \mathrm{d}x \\ &= \frac{1}{a} \int_{-\infty}^{\infty} f(t) e^{-i(\omega/a)t} \mathrm{d}t \quad (\because ax = t) \\ &= \frac{1}{|a|} F\left(\frac{\omega}{a}\right) \end{aligned} \tag{11}$$

3.1.3 逆

$$\begin{aligned} \mathcal{F}\Big[f(-x)\Big] &= \frac{1}{|-1|} F\left(\frac{\omega}{-1}\right) \\ &= F(-\omega) \end{aligned} \tag{12}$$

3.1.4 時間推移

$$\begin{aligned}\mathcal{F}\Big[f(x-x_0)\Big] &= \int_{-\infty}^{\infty} f(x-x_0)e^{-i\omega x}\mathrm{d}x \\ &= \int_{-\infty}^{\infty} f(t)e^{-i\omega(x_0+t)}\mathrm{d}t \quad (\because x-x_0=t) \\ &= e^{-i\omega x_0}F(\omega) \end{aligned} \tag{13}$$

3.1.5 周波数推移

$$\begin{aligned}\mathcal{F}\Big[f(x)e^{i\omega_0 x}\Big] &= \int_{-\infty}^{\infty} f(x)e^{i\omega_0 x}e^{-i\omega x}\mathrm{d}x \\ &= \int_{-\infty}^{\infty} f(x)e^{-i(\omega-\omega_0)x}\mathrm{d}x \\ &= F(\omega-\omega_0) \end{aligned} \tag{14}$$

3.1.6 対称性

$$\begin{aligned} f(x) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega x}\mathrm{d}\omega \\ f(-x) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{-i\omega x}\mathrm{d}\omega \quad (\because x=-x) \\ 2\pi f(-\omega) &= \int_{-\infty}^{\infty} F(x)e^{-i\omega x} \quad (\because x \leftrightarrow \omega) \\ &= \mathcal{F}\Big[F(x)\Big] \end{aligned} \tag{15}$$

3.1.7 導関数

$x \to \pm\infty$ で $f(x) \to 0$ とする.

$$\begin{aligned}\mathcal{F}\Big[f'(x)\Big] &= \int_{-\infty}^{\infty} f'(x)e^{-i\omega x}\mathrm{d}x \\ &= f(x)e^{-i\omega x}\Big|_{-\infty}^{\infty} + j\omega\int_{-\infty}^{\infty} f(x)e^{-i\omega x} \\ &= i\omega F(\omega) \end{aligned} \tag{16}$$

3.1.8 積分

$\omega \neq 0$ かつ $\int_{-\infty}^{\infty} f(x)\mathrm{d}x = F(0) = 0$ とする. $\phi(x) = \int_{-\infty}^{x} f(x')\mathrm{d}x'$ について

$$\mathcal{F}\Big[\phi'(x)\Big] = \mathcal{F}\Big[f(x)\Big] = i\omega\Phi(\omega)$$

従って

$$\mathcal{F}\Big[\int_{-\infty}^{x} f(x')\mathrm{d}x'\Big] = \frac{1}{i\omega}F(\omega) \tag{17}$$

ただし $x \to \infty$ で

$$\lim_{x\to\infty}\phi(x) = \int_{-\infty}^{\infty} f(x)\mathrm{d}x = F(\omega)\Big|_{\omega=0} = 0$$

である.

3.2 畳み込み

関数 $f_1(t), f_2(t)$ の畳み込みは

$$f(t) = \int_{-\infty}^{\infty} f_1(x)f_2(t-x)\mathrm{d}x \tag{18}$$

により定義される. 記号的には

$$f(t) = f_1(t) * f_2(t) \tag{19}$$

の形で表される. 特に $f_1(t) = 0, t < 0$ かつ $f_2(t) = 0, t < 0$ である場合

$$f(t) = f_1(t) * f_2(t) = \int_{0}^{t} f_1(x)f_2(t-x)\mathrm{d}x \tag{20}$$

となる. 畳み込みで成り立つ法則, 定理を以下に述べる.

3.2.1 可換則

$$\begin{aligned} f_1(t) * f_2(t) &= \int_{-\infty}^{\infty} f_1(x)f_2(t-x)\mathrm{d}x \\ &= \int_{-\infty}^{\infty} f_1(t-y)f_2(y)\mathrm{d}y \quad (\because y = t-x) \\ &= f_2(t) * f_1(t) \end{aligned} \tag{21}$$

3.2.2 結合則

$$\Big[f_1(t) * f_2(t)\Big] * f_3(t) = f_1(t) * \Big[f_2(t) * f_3(t)\Big]$$

について $g(t) = f_1(t) * f_2(t), h(t) = f_2(t) * f_3(t)$ とおくと,

$$g(t) = \int_{-\infty}^{\infty} f_1(y)f_2(t-y)\mathrm{d}y$$

だから

$$\begin{aligned} g(t) * f_3(t) &= \int_{-\infty}^{\infty} g(x) f_3(t-x) \mathrm{d}x \\ &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f_1(y) f_2(x-y) \mathrm{d}y \right] f_3(t-x) \mathrm{d}x \end{aligned}$$

である．$z = x - y$ を代入して積分の順序を入れ替えれば

$$g(t) * f_3(t) = \int_{-\infty}^{\infty} f_1(y) \left[\int_{-\infty}^{\infty} f_2(z) f_3(t-y-z) \mathrm{d}z \right] \mathrm{d}y$$

が得られる．ここで

$$h(t-y) = \int_{-\infty}^{\infty} f_2(z) f_3(t-y-z) \mathrm{d}z$$

に注意すれば

$$g(t) * f_3(t) = \int_{-\infty}^{\infty} f_1(y) h(t-y) \mathrm{d}y = f_1(t) * h(t)$$

より

$$\Big[f_1(t) * f_2(t) \Big] * f_3(t) = f_1(t) * \Big[f_2(t) * f_3(t) \Big] \tag{22}$$

を得ることができる．

3.2.3 デルタ関数との畳み込み

交換則より

$$\begin{aligned} f(t) * \delta(t) &= \int_{-\infty}^{\infty} f(x) \delta(t-x) \mathrm{d}x \\ &= \int_{-\infty}^{\infty} \delta(x) f(t-x) \mathrm{d}x \end{aligned}$$

デルタ関数について

$$\int_{-\infty}^{\infty} \delta(t-t_0) \phi(t) \mathrm{d}t = \int_{-\infty}^{\infty} \delta(t) \phi(t+t_0) \mathrm{d}t = \phi(t_0)$$

が成り立つから

$$\begin{aligned} \int_{-\infty}^{\infty} f(x) \delta(t-x) \mathrm{d}x &= \int_{-\infty}^{\infty} f(x) \delta(-(x-t)) \mathrm{d}x \\ &= \int_{-\infty}^{\infty} f(x) \delta(x-t) \mathrm{d}x \quad (\because \delta(x) = \delta(-x)) \\ &= f(t) \end{aligned}$$

より

$$f(t) * \delta(t) = \delta(t) * f(t) = f(t) \tag{23}$$

である. また,

$$f(t) * \delta(t-T) = \delta(t-T)f(t) = \int_{-\infty}^{\infty} \delta(x-T)f(t-x)\mathrm{d}x$$

について $y = x - T, \tau = t - T$ とおけば

$$\begin{aligned} f(t) * \delta(t-T) &= \int_{-\infty}^{\infty} \delta(y)f(t-T-y)\mathrm{d}y \\ &= \int_{-\infty}^{\infty} \delta(y)f(\tau-y)\mathrm{d}y \\ &= f(\tau) \end{aligned}$$

であるから

$$f(t) * \delta(t-T) = f(t-T) \tag{24}$$

が得られる. 同様に

$$f(t-t_1) * \delta(t-t_2) = \delta(t-t_2) * f(t-t_1) = \int_{-\infty}^{\infty} \delta(x-t_2)f(t-x-t_1)\mathrm{d}x$$

について $y = x - t_2, \tau = t - t_2 - t_1$ とおけば

$$\begin{aligned} f(t-t_1) * \delta(t-t_2) &= \int_{-\infty}^{\infty} \delta(y)f(t-y-t_2-t_1)\mathrm{d}y \\ &= \int_{-\infty}^{\infty} \delta(y)f(\tau-y)\mathrm{d}y \\ &= f(\tau) \end{aligned}$$

であるから

$$f(t-t_1) * \delta(t-t_2) = f(t-t_2-t_1) \tag{25}$$

が得られる.

3.2.4 時間畳み込み定理

$f_1(t) * f_2(t)$ のフーリエ変換を考えると

$$\begin{aligned} \mathcal{F}\Big[f_1(t) * f_2(t)\Big] &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f_1(x)f_2(t-x)\mathrm{d}x\right] e^{-i\omega t}\mathrm{d}t \\ &= \int_{-\infty}^{\infty} f_1(x) \left[\int_{-\infty}^{\infty} f_2(t-x)e^{-i\omega t}\mathrm{d}t\right] \mathrm{d}x \\ &= \left[\int_{-\infty}^{\infty} f_1(x)e^{-i\omega x}\mathrm{d}x\right] F_2(\omega) \quad \Big(\because \mathcal{F}\Big[f(x-x_0)\Big] = e^{-i\omega x_0}F(\omega)\Big) \\ &= F_1(\omega)F_2(\omega) \end{aligned} \tag{26}$$

3.2.5 周波数畳み込み定理

$F_1(\omega) * F_2(\omega)$ のフーリエ逆変換を考えると

$$\begin{aligned}\mathcal{F}^{-1}\Big[F_1(\omega) * F_2(\omega)\Big] &= \mathcal{F}^{-1}\Big[\int_{-\infty}^{\infty} F_1(y)F_2(\omega - y)\mathrm{d}y\Big] \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} F_1(y)F_2(\omega - y)\mathrm{d}y\right] e^{i\omega t}\mathrm{d}\omega\end{aligned}$$

$x = \omega - y$ とすれば

$$\begin{aligned}\mathcal{F}^{-1}\Big[F_1(\omega) * F_2(\omega)\Big] &= \frac{1}{2\pi}\int_{-\infty}^{\infty} F_1(y)\left[\int_{-\infty}^{\infty} F_2(x)e^{i(x+y)t}\mathrm{d}x\right]\mathrm{d}y \\ &= 2\pi\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} F_1(y)e^{iyt}\mathrm{d}y\right]\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} F_2(x)e^{ixt}\mathrm{d}x\right] \\ &= 2\pi f_1(t)f_2(t) \qquad (27)\end{aligned}$$

であり, つまり

$$\mathcal{F}\Big[f_1(t)f_2(t)\Big] = \frac{1}{2\pi}F_1(\omega) * F_2(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F_1(y)F_2(\omega - y)\mathrm{d}y \qquad (28)$$

である.

4 離散フーリエ変換 (DFT)

4.1 離散フーリエ係数

関数 $f(x)$ の複素フーリエ級数展開

$$f(x) \sim \sum_{n=-\infty}^{\infty} c_n e^{in\omega x}$$

における

$$c_n = \frac{1}{2L}\int_{-L}^{L} f(x)e^{-in\omega x}\mathrm{d}x$$

を離散化することを考える. 一周期 $[0, 2L]$ を N 等分したとき $h = \dfrac{2L}{N}, x_j = jh \quad (j = 0, 1, 2, ...)$, また $f_j = f(x_j)$ とおく. $f(x)$ の区間 $[a, b]$ での積分の台形則による近似が

$$\int_a^b f(x)\mathrm{d}x \sim \frac{h}{2}\left(f_0 + 2\sum_{i=i}^{n-1} f_i + f_n\right), \ \left(h = \frac{b-a}{n}\right)$$

で与えらえれ, 周期関数においては $f_0 = f_n$ であることに注意すれば c_n についての台形則による近似は

$$c_n \sim C_n = \frac{1}{2L}h\sum_{j=0}^{N-1} f_j e^{-in\omega x_j} \qquad (29)$$

となる. さらに定数について新しく $\xi = e^{i\omega h}$ とおき, $h = \dfrac{2L}{N}$ を展開すれば

$$C_n = \frac{1}{N}\sum_{j=0}^{N-1} f_j \xi^{-nj} \tag{30}$$

として f の離散フーリエ係数 C_n が得られる.

4.2 ξ の性質

$\omega = \dfrac{\pi}{L}, h = \dfrac{2L}{N}, \xi = e^{i\omega h}$ であったから

$$\xi = \exp\left(\frac{2\pi i}{N}\right)$$

が得られる. 複素数 $z = re^{i\alpha}, w = se^{i\beta}$ についてその積が

$$\begin{aligned} zw &= re^{i\alpha} se^{i\beta} \\ &= rse^{i(\alpha+\beta)} \end{aligned}$$

で与えられることを考えれば z の n 乗は

$$z^n = r^n e^{in\alpha}$$

であり, $\gamma = z^n$ としたときその n 乗根は

$$\sqrt[n]{\gamma} = \sqrt[n]{r^n}\exp\left(\frac{in\alpha}{n}\right)$$

である. すなわち 1 についてその n 乗根は

$$\sqrt[n]{1} = \exp\left(\frac{2\pi m i}{n}\right) \quad (m = 0, \pm 1, \pm 2, \ldots)$$

であり, ξ がこの $m = 1$ の場合であることが分かる. ξ は m について $m = n\alpha + \beta$ であれば,

$$\begin{aligned} \xi^{n\alpha+\beta} &= \exp\left(\frac{2\pi n\alpha i}{n}\right)\exp\left(\frac{2\pi\beta i}{n}\right) \\ &= e^{2\pi\alpha i}\xi^{\beta} \\ &= \xi^{\beta} \end{aligned}$$

より m_1, m_2 について

$$\xi^{m_1} = \xi^{m_2} \quad m_1 \equiv m_2 \pmod n \tag{31}$$

であることが分かる. また, 初項 a, 公比 r の等比級数の n 項までの和が $\dfrac{a(r^n-1)}{r-1}$ で与えられ, $\xi = \sqrt[n]{1}$ であることから

$$\sum_{m=0}^{n-1}\xi^{mj} = \begin{cases} n & j \equiv 0 \pmod n \\ 0 & j \not\equiv 0 \pmod n \end{cases} \tag{32}$$

が導出される.

4.3 離散フーリエ変換

$\boldsymbol{f} = \begin{bmatrix} f_0, f_1, \cdots, f_{N-1} \end{bmatrix}^\top$ に対する離散フーリエ係数 $\boldsymbol{C} = \begin{bmatrix} C_0, C_1, \cdots, C_{N-1} \end{bmatrix}^\top$ を $\boldsymbol{f}$ の離散フーリエ変換と呼ぶ. C_0, C_1, C_2 について計算結果を示すと,

$$
\begin{aligned}
C_0 &= \frac{1}{N}\Big(f_0\xi^{-0\cdot 0} + f_1\xi^{-0\cdot 1} + f_2\xi^{-0\cdot 2} + \cdots\Big) = \frac{1}{N}\Big(f_0\xi^{0} + f_1\xi^{0} + f_2\xi^{0} + \cdots\Big) \\
C_1 &= \frac{1}{N}\Big(f_0\xi^{-1\cdot 0} + f_1\xi^{-1\cdot 1} + f_2\xi^{-1\cdot 2} + \cdots\Big) = \frac{1}{N}\Big(f_0\xi^{0} + f_1\xi^{-1} + f_2\xi^{-2} + \cdots\Big) \\
C_2 &= \frac{1}{N}\Big(f_0\xi^{-2\cdot 0} + f_1\xi^{-2\cdot 1} + f_2\xi^{-2\cdot 2} + \cdots\Big) = \frac{1}{N}\Big(f_0\xi^{0} + f_1\xi^{-2} + f_2\xi^{-4} + \cdots\Big)
\end{aligned}
$$

となる. これは行列演算としてまとめて

$$
\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ \vdots \\ C_{N-1} \end{pmatrix} = \frac{1}{N}\begin{pmatrix} \xi^0 & \xi^0 & \xi^0 & \cdots & \xi^0 \\ \xi^0 & \xi^{-1} & \xi^{-2} & \cdots & \xi^{-1\cdot(N-1)} \\ \xi^0 & \xi^{-2} & \xi^{-4} & \cdots & \xi^{-2\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \xi^0 & \xi^{-(N-1)\cdot 1} & \xi^{-(N-1)\cdot 2} & \cdots & \xi^{-(N-1)(N-1)} \end{pmatrix}\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{pmatrix} \tag{33}
$$

と書き直すことができる. ここで

$$
W = \frac{1}{N}\begin{pmatrix} \xi^0 & \xi^0 & \xi^0 & \cdots & \xi^0 \\ \xi^0 & \xi^{-1} & \xi^{-2} & \cdots & \xi^{-1\cdot(N-1)} \\ \xi^0 & \xi^{-2} & \xi^{-4} & \cdots & \xi^{-2\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \xi^0 & \xi^{-(N-1)\cdot 1} & \xi^{-(N-1)\cdot 2} & \cdots & \xi^{-(N-1)(N-1)} \end{pmatrix}
$$

とおけばこれは結局

$$
\boldsymbol{C} = W\boldsymbol{f} \tag{34}
$$

である. また,

$$
-nj\alpha \equiv 0 \,(\mathrm{mod}\, N) \tag{35}
$$

であれば式 (32) より, 離散フーリエ逆変換を行うために必要な W の逆行列 W^{-1} がその転置の複素共役をとり, N を掛けたもので与えられることが分かる.

$$
W^{-1} = \begin{pmatrix} \xi^0 & \xi^0 & \xi^0 & \cdots & \xi^0 \\ \xi^0 & \xi^1 & \xi^2 & \cdots & \xi^{1\cdot(N-1)} \\ \xi^0 & \xi^2 & \xi^4 & \cdots & \xi^{2\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \xi^0 & \xi^{(N-1)\cdot 1} & \xi^{(N-1)\cdot 2} & \cdots & \xi^{(N-1)(N-1)} \end{pmatrix} \tag{36}
$$

実際, WW^{-1} の (k_1, k_2) 成分について計算してみると

$$
\begin{aligned}
WW^{-1}_{(k_1,k_2)} &= \xi^0\xi^0 + \xi^{-(k_1-1)\cdot 1}\xi^{1\cdot(k_2-1)} + \cdots + \xi^{-(k_1-1)(N-1)}\xi^{(N-1)(k_2-1)} \\
&= \xi^0 + \xi^{k_2-k_1} + \cdots + \xi^{(N-1)(k_2-k_1)} \\
&= \sum_{n=0}^{N-1} \xi^{n(k_2-k_1)}
\end{aligned}
$$

より初項 $\xi^0 = 1$, 公比 ξ の等比級数によって与えられるので, $k_2 - k_1 = 0$ となる対角成分以外は全て 0 となることが分かる. 従って

$$f_j = \sum_{n=0}^{N-1} C_n \xi^{nj}$$

である.

5 高速フーリエ変換 (FFT)

5.1 Cooley-Tukey FFT

離散フーリエ係数

$$f_j = \sum_{n=0}^{N-1} C_n \xi^{jn} \tag{37}$$

について N が r_1 と r_2 による合成, すなわち $N = r_1 r_2$ であるとする. このとき式 (37) について n, j は 0 から $N-1$ までを動くから

$$\begin{aligned} n, j &= 0, 1, \ldots, r_1 r_2 - 1 \\ &= 0, 1, \ldots, (r_1 - 1) r_2 + r_2 - 1 \\ &= 0, 1, \ldots, (r_2 - 1) r_1 + r_1 - 1 \end{aligned}$$

より

$$j = j_1 r_1 + j_0 \quad \Big(j_0 = 0, 1, \ldots, r_1 - 1, \quad j_1 = 0, 1, \ldots, r_2 - 1 \Big) \tag{38}$$

$$n = n_1 r_2 + n_0 \quad \Big(n_0 = 0, 1, \ldots, r_2 - 1, \quad n_1 = 0, 1, \ldots, r_1 - 1 \Big) \tag{39}$$

であるとすると,

$$\begin{aligned} jn &= j_1 r_1 n_1 r_2 + j_1 r_1 n_0 + j_0 n_1 r_2 + j_0 n_0 \\ &= n_1 r_2 (j_1 r_1 + j_0) + n_0 (j_1 r_1 + j_0) \\ &= j n_1 r_2 + j n_0 \end{aligned} \tag{40}$$

より式 (37) を

$$f_{j_1, j_0} = \sum_{n_0} \sum_{n_1} C_{n_1, n_0} \xi^{j n_1 r_2} \xi^{j n_0} \tag{41}$$

と書き直すことができる. ξ について

$$\begin{aligned} j n_1 r_2 &= j_1 n_1 r_1 r_2 + j_0 n_1 r_2 \\ &= j_1 n_1 N + j_0 n_1 r_2 \end{aligned}$$

より

$$\xi^{j n_1 r_2} = \exp\left(\frac{2\pi j_1 n_1 N i}{N} \right) \xi^{j_0 n_1 r_2} \tag{42}$$

$$= \xi^{j_0 n_1 r_2} \tag{43}$$

であるから, ここで新しく

$$C_{j_0,n_0} = \sum_{n_1} C_{n_1,n_0} \xi^{j_0 n_1 r_2} \tag{44}$$

と定義すれば f_{j_1,j_0} について

$$f_{j_1,j_0} = \sum_{n_0} C_{j_0,n_0} \xi^{j n_0} \tag{45}$$

$$= \sum_{n_0} C_{j_0,n_0} \xi^{(j_1 r_1 + j_0) n_0} \tag{46}$$

が得られる. C_{j_0,n_0} は $j_0 = 0, 1, \ldots, r_1 - 1, n_0 = 0, 1, \ldots, r_2 - 1$ より $r_1 r_2 = N$ 個の要素を持ち, それぞれの要素を得るには $n_1 = 0, 1, \ldots, r_1 - 1$ より r_1 回の演算が必要なので C_{j_0,n_0} を得るには合計で $N r_1$ 回の演算が必要である. 同様に C_{j_0,n_0} を用いて f_{j_1,j_0} を計算するためには $N r_2$ 回の演算が必要であるから, C_{n_1,n_0} を用いて f_{j_1,j_0} を計算するためには

$$T = N(r_1 + r_2) \tag{47}$$

回の演算が必要であることが分かる. 式 (46) が式 (37) とほぼ同じ形になっていることに注目すれば, 式 (46) を同様の手順で m ステップに分解することにより, f_j を求めるための計算量が

$$T = N(r_1 + r_2 + \cdots + r_m) \tag{48}$$

で与えられることが分かる. ただし

$$N = r_1 r_2 \cdots r_m \tag{49}$$

である. N を分解をしたとき, r_j が他の全ての r と等しければ分解の回数 m は $m = \log_r N$ によって与えられる. 従って式 (48) は $(r_1 + r_2 + \cdots + r_m) = r \log_r N$ であるから,

$$T = N \cdot r \log_r N \tag{50}$$

と書き直すことができる. さらに, $N = r^m s^n t^p \cdots$ であるなら

$$\frac{T}{N} = rm + sn + tp + \cdots \tag{51}$$

$$\log_2 N = m \log_2 r + n \log_2 s + p \log_2 t + \cdots \tag{52}$$

であり,

$$\frac{T}{N \log_2 N} = \frac{rm + sn + tp + \cdots}{m \log_2 r + n \log_2 s + p \log_2 t + \cdots}$$

より $\dfrac{T}{N \log_2 N}$ は $\dfrac{r}{\log_2 r}, \dfrac{s}{\log_2 s}, \dfrac{t}{\log_2 t}$ の加重平均である.

$\dfrac{r}{\log_2 r}$ について r を変化させてゆくと値は以下のように動く.

r	2	3	4	5	6	7	8	9	10
$\dfrac{r}{\log_2 r}$	2.00	1.88	2.00	2.15	2.31	2.49	2.67	2.82	3.01

表を見ると $r=3$ の場合が最も効率的に計算できることが分かるが, $r=2$ や $r=4$ の場合と比べて約 6% 効率が良いだけなので, 実際に計算する場合はより便利な $r=2$ や $r=4$ を使う.

$N=2^m$ である場合, n, j が式 (38) および (39) のように $r-1$ までの値をとることを考えれば 0 から $N-1$ までの数は二進数のように表現できるから n, j について $n_k, j_k \quad (k=0,1,\ldots,m-1)$ を用いて

$$j = j_0 + j_1\cdot 2 + j_2\cdot 2^2 + \cdots + j_{m-1}\cdot 2^{m-1} \quad (j_k = 0,1) \tag{53}$$

$$n = n_0 + n_1\cdot 2 + n_2\cdot 2^2 + \cdots + n_{m-1}\cdot 2^{m-1} \quad (n_k = 0,1) \tag{54}$$

である. 従って

$$jn = jn_0 + j(n_1\cdot 2) + j(n_2\cdot 2^2) + \cdots + j(n_{m-1}\cdot 2^{m-1}) \tag{55}$$

であるから式 (37) を m 分割することにより f_j について

$$f_{j_{m-1},\ldots,j_0} = \sum_{n_0}\sum_{n_1}\cdots\sum_{n_{m-1}} C_{n_{m-1},\ldots,n_0}\xi^{jn_{m-1}\cdot 2^{m-1}+\cdots+jn_0} \tag{56}$$

が得られる. また, $l=1,2,\ldots m$ とすると ξ について

$$\begin{aligned}
\xi^{jn_{m-l}\cdot 2^{m-l}} &= \xi^{j_0 n_{m-l}\cdot 2^{m-l}}\xi^{(j_1\cdot 2+j_2\cdot 2^2+\cdots+j_{m-1}\cdot 2^{m-1})n_{m-l}\cdot 2^{m-l}} \\
&= \xi^{j_0 n_{m-l}\cdot 2^{m-l}}\exp\left(\frac{2\pi j_1\cdot 2n_{m-l}\cdot 2^{m-l}i}{N}\right)\cdots\exp\left(\frac{2\pi j_l\cdot 2^l n_{m-l}\cdot 2^{m-l}i}{N}\right)\cdots \\
&= \xi^{j_0 n_{m-l}\cdot 2^{m-l}}\xi^{(j_1\cdot 2+\cdots+j_{l-1}\cdot 2^{l-1})n_{m-l}\cdot 2^{m-l}}\exp\left(\frac{2\pi j_l n_{m-l}Ni}{N}\right)\cdots \\
&= \xi^{(j_0+j_1\cdot 2+\cdots+j_{l-1}\cdot 2^{l-1})n_{m-l}\cdot 2^{m-l}}
\end{aligned} \tag{57}$$

を得ることができる. ここで式 (57) より n_{m-1} での C の総和は

$$\sum_{n_{m-1}} C_{n_{m-1},\ldots,n_0}\xi^{j_0 n_{m-1}\cdot 2^{m-1}} \tag{58}$$

と書き直せるから,

$$C_1 = \sum_{n_{m-1}} C_{n_{m-1},\ldots,n_0}\xi^{j_0 n_{m-1}\cdot 2^{m-1}} \tag{59}$$

と新たに定義すれば C_l について

$$C_l = \sum_{n_{m-l}} C_{l-1}\xi^{(j_0+j_1\cdot 2+\cdots+j_{l-1}\cdot 2^{l-1})n_{m-l}\cdot 2^{m-l}} \tag{60}$$

が得られる. $n_{m-l}=0,1$ であるからこれを展開してみると

$$\begin{aligned}
C_l &= C_{l-1} + C_{l-1}\xi^{(j_0+j_1\cdot 2+\cdots+j_{l-3}\cdot 2^{l-3})\cdot 2^{m-l}} \exp\left(\frac{2\pi j_{l-1}2^{l-1}2^{m-l}i}{N}\right)\exp\left(\frac{2\pi j_{l-2}2^{l-2}2^{m-l}i}{N}\right)\\
&= C_{l-1} + C_{l-1}\xi^{(j_0+j_1\cdot 2+\cdots+j_{l-3}\cdot 2^{l-3})\cdot 2^{m-l}} \exp\left(\frac{\pi j_{l-1}Ni}{N}\right)\exp\left(\frac{\pi j_{l-2}Ni}{2N}\right)\\
&= C_{l-1} + C_{l-1}\xi^{(j_0+j_1\cdot 2+\cdots+j_{l-3}\cdot 2^{l-3})\cdot 2^{m-l}} (-1)^{j_{l-1}} i^{j_{l-2}} \qquad (61)
\end{aligned}$$

となる. $l=m$ で

$$C_m = C_{m-1} + C_{m-1}\xi^{j_0+\cdots+j_{m-1}\cdot 2^{m-1}} \qquad (62)$$

であるから, 式 (56) の f_j は

$$f_{j_{m-1},\ldots,j_0} = C_{j_0,\ldots,j_{m-1}} \qquad (63)$$

によって与えられる.

5.2 FFT プログラム

要素数 $N=2^m$ の配列 $\{c_0, c_1, \ldots, c_{2^m-1}\}$ を高速フーリエ変換することを考える. 配列の分割の仕方を考えたとき, 偶数番目と奇数番目の要素で分割すると c_n の離散フーリエ変換は

$$\begin{aligned}
f_j &= \sum_{m=0}^{N/2-1} c_{2m}\xi^{2mj} + \sum_{m=0}^{N/2-1} c_{2m+1}\xi^{(2m+1)j}\\
&= \sum_{m=0}^{N/2-1} c_{2m}\exp\left(\frac{2\pi i 2mj}{N}\right) + \sum_{m=0}^{N/2-1} c_{2m+1}\exp\left(\frac{2\pi i 2mj}{N}\right)\exp\left(\frac{2\pi ij}{N}\right)\\
&= \sum_{m=0}^{N/2-1} c_{2m}\exp\left(\frac{2\pi imj}{N/2}\right) + \sum_{m=0}^{N/2-1} c_{2m+1}\exp\left(\frac{2\pi imj}{N/2}\right)\exp\left(\frac{2\pi ij}{N}\right) \qquad (64)
\end{aligned}$$

となる. c_{2m}, c_{2m+1} を各要素に持つ要素数 $N/2$ の 2 つの配列

$$\begin{aligned}
c_{k_1} &= \{c_0, c_2, \ldots, c_{2m}\}\\
c_{k_2} &= \{c_1, c_3, \ldots, c_{2m+1}\}
\end{aligned}$$

を新しく定義しなおせば,

$$\begin{aligned}
&\sum_{m=0}^{N/2-1} c_{2m}\exp\left(\frac{2\pi imj}{N/2}\right)\\
&\sum_{m=0}^{N/2-1} c_{2m+1}\exp\left(\frac{2\pi imj}{N/2}\right)
\end{aligned}$$

は $C_k = \sum_k c_k\xi^{kj}$ として

$$\begin{aligned}
f_j &= \left\{\sum_{k_1} c_{k_1}\xi^{k_1 j}\right\} + \left\{\sum_{k_2} c_{k_2}\xi^{k_2 j}\right\}\xi^j \qquad (65)\\
&= C_{k_1} + C_{k_2}\xi^j \qquad (66)
\end{aligned}$$

と書き直すことができる．この新たに定義した配列 c_{k_1}, c_{k_2} について C_{k_1} および C_{k_2} は離散フーリエ変換の形になっているから，さらに偶数番目 (e) と奇数番目 (o) の要素で分割すると

$$f_j = \left\{C_{k_1}(e) + C_{k_1}(o)\xi^j\right\} + \left\{C_{k_2}(e) + C_{k_2}(o)\xi^j\right\} \tag{67}$$

となり，この分割を繰り返すことにより f_j を求めることができる．要素数が 1 の時の離散フーリエ変換はその定義により

$$f_0 = \sum_{n=0}^{0} C_0 \xi^{j\cdot 0} = C_0 \tag{68}$$

であるから，結局 FFT のプログラムは

$$f_j = \begin{cases} C_0 & N = 1 \\ C_k(e) + C_k(o)\xi^j & N > 1 \end{cases} \tag{69}$$

を実装することで得られる．また，離散フーリエ逆変換は

$$C_n = \frac{1}{N}\sum_{j=0}^{N-1} f_j \xi^{-nj}$$

より

$$NC_n = \begin{cases} f_0 & N = 1 \\ f_k(e) + f_k(o)\xi^{-j} & N > 1 \end{cases} \tag{70}$$

であるから，式 (69) の実装で j の符号を反転させたものを用いて求めたものに $\frac{1}{N}$ を掛けることで得ることができる．これを C++ を用いて実装すると次のようになる．

```cpp
#include <vector>
#include <complex>
#include <cmath>

using complex = std::complex<double>;
constexpr double pi = 3.14159265358979323L;

std::vector<complex>
fft(const std::vector<complex>& c, const bool inverse = false) {
    std::size_t N = c.size();                       // N = 2^m
    double N_inv = 1.0 / static_cast<double>(N);    // Inverse of N
    std::vector<complex> f(N);                      // FFT result
    int inv = inverse ? -1 : 1;                     // Inverse flag

    // When N=1, f_0=c_0
    if(N <= 1) return c;

    // Decompose by even-odd
    std::vector<complex> even(N / 2), odd(N / 2);
    for(int i = 0; i < N / 2; ++i) {
        even.at(i) = c.at(2 * i);
```

```
        odd.at(i) = c.at((2 * i) + 1);
    }
    even = fft(even, inverse);
    odd = fft(odd, inverse);

    // Calculate C_k(e) + C_k(o) * xi^j
    double theta = (2.0 * pi * N_inv * inv);
    const auto xi = [&](int j) {
        return complex(std::cos(theta * j), std::sin(theta * j));
    };
    for(int j = 0; j < N; ++j) {
        int k = j % (N / 2);
        f.at(j) = even.at(k) + (odd.at(k) * xi(j));
    }
    return f;
}
```

以下では必要な定数等の定義を行っている．各行では要素数 N の取得，$\dfrac{1}{N}$ の定義，FFT を行った結果を保存するための配列の定義，逆変換を行う場合は -1 の用意を行っている．

```
std::size_t N = c.size();                          // N = 2^m
double N_inv = 1.0 / static_cast<double>(N);       // Inverse of N
std::vector<complex> f(N);                         // FFT result
int inv = inverse ? -1 : 1;                        // Inverse flag
```

以下は $N = 1$ の場合の特別な処理である．式 (69) より $N = 1$ のときはフーリエ変換を行う配列そのものがフーリエ変換の結果であるから受け取った配列をそのまま結果として返している．

```
if(N <= 1) return c;
```

以下では配列要素の分解，すなわち

$$f_j = \left\{ \sum_{k_1} c_{k_1} \xi^{k_1 j} \right\} + \left\{ \sum_{k_2} c_{k_2} \xi^{k_2 j} \right\} \xi^j$$

における c_{k_1}, c_{k_2} を求めている．各行では要素数が $\dfrac{N}{2}$ の偶数番目および奇数番目の要素を受け取るための配列の定義，各要素の代入，再帰関数による分解の繰り返しを行っている．

```
std::vector<complex> even(N / 2), odd(N / 2);
for(int i = 0; i < N / 2; ++i) {
    even.at(i) = c.at(2 * i);
    odd.at(i) = c.at((2 * i) + 1);
}
even = fft(even, inverse);
odd = fft(odd, inverse);
```

以下では

$$C_k(e) + C_k(o)\xi^j$$

の計算を行っている．theta は ξ の指数部であり，逆変換の際は -1 が掛かるため inv が掛かっている．また，xi は j を受け取り ξ^j を返す関数である．

```
double theta = (2.0 * pi * N_inv * inv);
const auto xi = [&](int j) {
    return complex(std::cos(theta * j), std::sin(theta * j));
};
for(int j = 0; j < N; ++j) {
    int k = j % (N / 2);
    f.at(j) = even.at(k) + (odd.at(k) * xi(j));
}
return f;
```

最後に

```
int k = j % (N / 2);
f.at(j) = even.at(k) + (odd.at(k) * xi(j));
```

における k の取り方の正当性を示す．以下は式 (64) を再掲したものである．

$$f_j = \sum_{m=0}^{N/2-1} c_{2m} \exp\left(\frac{2\pi imj}{N/2}\right) + \sum_{m=0}^{N/2-1} c_{2m+1} \exp\left(\frac{2\pi imj}{N/2}\right) \exp\left(\frac{2\pi ij}{N}\right)$$

この式について $j > \dfrac{N}{2}$ について考える．$j = \dfrac{N}{2} + \alpha \quad (\alpha = 1, 2, \ldots, \dfrac{N}{2} - 1)$ としてこれを書き直すと

$$\begin{aligned}
f_j &= \sum_{m=0}^{N/2-1} c_{2m} \exp\left(\frac{2\pi im(N/2+\alpha)}{N/2}\right) \\
&\qquad + \sum_{m=0}^{N/2-1} c_{2m+1} \exp\left(\frac{2\pi im(N/2+\alpha)}{N/2}\right) \exp\left(\frac{2\pi ij}{N}\right) \\
&= \sum_{m=0}^{N/2-1} c_{2m} \exp\Big(2\pi im\Big) \exp\left(\frac{2\pi im\alpha}{N/2}\right) \\
&\qquad + \sum_{m=0}^{N/2-1} c_{2m+1} \exp\Big(2\pi im\Big) \exp\left(\frac{2\pi im\alpha}{N/2}\right) \exp\left(\frac{2\pi ij}{N}\right) \\
&= \sum_{m=0}^{N/2-1} c_{2m} \exp\left(\frac{2\pi im\alpha}{N/2}\right) - \sum_{m=0}^{N/2-1} c_{2m+1} \exp\left(\frac{2\pi im\alpha}{N/2}\right) \exp\left(\frac{2\pi ij}{N}\right)
\end{aligned} \tag{71}$$

より

$$\begin{aligned}
f_j &= C_\alpha(e) + C_\alpha(o)\xi^j \\
&= C_{j \bmod (N/2)}(e) + C_{j \bmod (N/2)}(o)\xi^j
\end{aligned} \tag{72}$$

を得ることができるため，以下の行において k は $j \bmod (N/2)$ となっている．

```
int k = j % (N / 2);
f.at(j) = even.at(k) + (odd.at(k) * xi(j));
```

5.3 多項式の掛け算と畳み込みの高速化

多項式

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n$$
$$Q(x) = q_0 + q_1 x + q_2 x^2 + \cdots + q_n x^n$$

について $P(x)Q(x)$ を考える. 実際に計算してみると

$$\begin{aligned} P(x)Q(x) &= p_0 q_0 + (p_0 q_1 + p_1 q_0)x + \cdots + \sum_{i=0}^{n} p_i q_{n-i} x^n + \sum_{i=1}^{n} (p_i q_n + p_n q_i) x^{n+i} \\ &= \sum_{m=0}^{n} \sum_{i=0}^{m} p_i q_{m-i} x^m + \sum_{i=1}^{n} (p_i q_n + p_n q_i) x^{n+i} \end{aligned}$$

となる. ここで, $P(x), Q(x)$ について係数が 0 である $x^{n+1}, \ldots, x^{2n}$ までの項を追加し

$$P(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_{2n} x^{2n}$$
$$Q(x) = q_0 + q_1 x + q_2 x^2 + \cdots + q_{2n} x^{2n}$$

とすることで $P(x)Q(x)$ について x^{2n} までの項を

$$P(x)Q(x)_{2n} = \sum_{m=0}^{2n} \sum_{i=0}^{m} p_i q_{m-i} x^m \tag{73}$$

と書き直すことができる. この計算量は $O(n^2)$ であるが, 式 (73) の x^m の係数が畳み込み

$$P(m) * Q(m) = \int_{-\infty}^{\infty} P(i)Q(m-i)\mathrm{d}i$$

を離散化したものになっていることに注目し, 時間畳み込み定理

$$\mathcal{F}\Big[f_1(t) * f_2(t)\Big] = F_1(\omega)F_2(\omega) \tag{74}$$

を思い出せば, $P(x)Q(x)$ の係数は $P(x), Q(x)$ それぞれの係数の配列をフーリエ変換したものの積より得ることができることが分かるので, FFT を用いることでこの計算量を $O(n \log n)$ まで減らすことができる.

5.4 畳み込み演算の高速化の例

以下は AtCoder Typical Contest 001 C - 高速フーリエ変換の問題である.

> AtCoder 食堂では、定食のメニューを検討しています。
> - 主菜は、価格が i 円のものが A_i 種類あります $(1 \leqq i \leqq N)$。
> - 副菜は、価格が i 円のものが B_i 種類あります $(1 \leqq i \leqq N)$。
>
> 定食は、主菜と副菜を 1 種類ずつ選んで構成します。 定食の価格は、選んだ主菜と副菜の価格の和とします。各 $k(1 \leqq k \leqq 2N)$ について、価格が k 円になる定食の種類の数を計算して下さい。

この問題においてちょうど k 円になる組み合わせの個数 C_k は

$$C_k = \sum_{i=0}^{k} A_i B_{k-i} \tag{75}$$

であるから, これを $2N$ 円まで求めようとすると式 (73) と同様に A_{2n}, B_{2n} まであると考えれば

$$C = \sum_{i=0}^{k} A_i B_{k-i} \quad (k = 0, 1, \ldots, 2N) \tag{76}$$

を計算すれば良いことが分かる. $k = k_n$ の場合を x^{k_n} としてこれを多項式として表せば

$$\sum_{k=0}^{2N} \sum_{i=0}^{k} A_i B_{k-i}$$

である. 従って A_i, B_i の離散フーリエ変換を a_i, b_i として,

$$C = \mathcal{F}^{-1}\Big[a_i b_i\Big] \quad (k = 0, 1, \ldots, 2N) \tag{77}$$

を得ることができる. つまりこの問題は畳み込み処理さえ実装すれば容易に解くことができて, 実際に C++ で実装したものが関数 convolution である.

```
std::vector<complex>
convolution(const std::vector<complex>& f, const std::vector<complex>& g) {
    std::size_t N = 1;
    while(N < (f.size() + g.size() - 1)) N *= 2;

    std::vector<complex> F(N), G(N);
    for(int i = 0; i < f.size(); ++i) {
        F.at(i) = f.at(i);
        G.at(i) = g.at(i);
    }
    F = fft(F);
    G = fft(G);

    std::vector<complex> conv(N);
    for(int i = 0; i < N; ++i)
        conv.at(i) = F.at(i) * G.at(i);

    conv = fft(conv, -1);
    for(int i = 0; i < N; ++i)
        conv.at(i) *= complex(1.0 / static_cast<double>(conv.size()));
    return conv;
}
```

5.5 多倍長整数の演算

式 (73) に $x = 10$ を代入すれば整数 $a_i \dots a_1 a_0, b_j \dots b_1 b_0$ を配列 $A = \{a_i, \dots, a_1, a_0\}, B = \{b_i, \dots, b_1, b_0\}$ として表すことで FFT を用いて多倍長整数の掛け算を実装することができる. 具体的に

$$9987958341098962462 4 \times 82646219652732371529 \tag{78}$$

を計算するプログラムを以下に示す.

```
std::vector<int>
multiply(const std::vector<int>& a, const std::vector<int>& b) {
    std::vector<complex> ac(a.size()), bc(b.size());
    for(int i = 0; i < a.size(); ++i) ac.at(i) = complex(a.at(i));
    for(int i = 0; i < b.size(); ++i) bc.at(i) = complex(b.at(i));

    std::vector<complex> rc = convolution(ac, bc);
    std::vector<int> r(rc.size());
    for(int i = 0; i < rc.size(); ++i) r.at(i) = std::round(rc.at(i).real());
    return r;
}

int main() {
    std::vector<int>
        a = {4, 2, 6, 4, 2, 6, 9, 8, 9, 0, 1, 4, 3, 8, 5, 9, 7, 8, 9, 9};
    std::vector<int>
        b = {9, 2, 5, 1, 7, 3, 2, 3, 7, 2, 5, 6, 9, 1, 2, 6, 4, 6, 2, 8};

    std::vector<int> m = multiply(a, b);
    for(int i = 0; i < m.size() - 1; ++i) {
        if(m.at(i) >= 10) {
            int k = m.at(i) / 10;
            m.at(i) -= 10 * k;
            m.at(i + 1) += k;
        }
    }
    while(m.back() >= 10) {
        int k = m.back() / 10;
        m.back() -= 10 * k;
        m.push_back(k);
    }
    while(m.size() > 1 && m.back() == 0) {
        m.pop_back();
    }

    for(int i = m.size() - 1; i >= 0; --i) std::cout << m.at(i);
    std::cout << std::endl;
    return 0;
}
```

ただし

```
std::vector<int> m = multiply(a, b);
```

以下は繰り上がり処理である. また, 複素数値として計算される離散フーリエ変換の実部は目的の整数がちょうど得られるとは限らないので

```
r.at(i) = std::round(rc.at(i).real());
```

のように round 関数を用いてその四捨五入を行っている.
実際にこのプログラムを実行すると値として

$$825466998940805287058672141763701493009 6 \tag{79}$$

が得られ, これは確かに正しい計算結果であると分かる.

なお, 上記のプログラムでは桁数が違う場合は桁数が小さい方を 0 を埋めることで大きい方と同じ桁数まで持ってゆく必要があり, 例えば 1234567×1234 は 1234567×0001234 に直して計算する. 従って $P(10), Q(10)$ の桁数をそれぞれ N_P, N_Q として $N_{\max} = \max(N_P, N_Q)$ のとき計算量は $O(N_{\max} \log N_{\max})$ である.
しかし,

$$\begin{aligned} P(x) &= p_0 + p_1 x + p_2 x^2 + \cdots + p_i x^i \\ Q(x) &= q_0 + q_1 x + q_2 x^2 \end{aligned}$$

として $P(x)Q(x)$ を計算すると

$$\begin{aligned} P(x)Q(x) &= p_0(q_0 + q_1 x + q_2 x^2) + p_1 x(q_0 + q_1 x + q_2 x^2) + \cdots + p_i x^i (q_0 + q_1 x + q_2 x^2) \\ &= (p_0 + p_1 x + p_2 x^2)Q(x) + (p_3 x^3 + p_4 x^4 + p_5 x^5)Q(x) + \cdots \\ &= \sum_{m=0}^{0\cdot 3+2\cdot 2} \sum_{i=0}^{m} p_i q_{m-i} x^m + \sum_{m=3}^{1\cdot 3+2\cdot 2} \sum_{i=3}^{m} p_i q_{m-i} x^m + \cdots \end{aligned} \tag{80}$$

のようにして複数の畳み込みへと分解することができるからこれを計算することで $N_{\max} \log N_{\min}$ まで改善することができる. ただし $N_{\min} = \min(N_P, N_Q)$ である.

参考文献

[1] J.W.Cooley, J.W.Tukey: An Algorithm for the Machine Calculation of Complex Fourier Series, Mathematics of Computation, Vol.19 1965.

[2] 神永正博. Python で学ぶフーリエ解析と信号処理. コロナ社, 2020.

[3] Hsu, Hwei P. (Hwei Piao), and 佐藤平八. フーリエ解析. 森北出版, 1979.

[4] 桂田祐史. 画像処理とフーリエ変換 (2016). (2022/08/04, 閲覧).

[5] フーリエ変換・FFT 入門. (2022/08/01, 閲覧).

[6] デルタ関数. (2022/08/03, 閲覧).

[7] 「数値積分法」, 台形公式. (2022/08/03, 閲覧).

[8] Python プログラミング（ステップ 8・関数・高速フーリエ変換）. (2022/08/14, 閲覧).

[9] AtCoder Typical Contest 001, C - 高速フーリエ変換. (2022/08/14, 閲覧).

[10] 超高速！多倍長整数の計算手法【前編：大きな数の四則計算を圧倒的な速度で！】. (2022/08/22, 閲覧).