第1章　乱暴な序論

以下の解説は，FFT に至るまでの概説であるが，予備知識のない方々には分からない数学記号や用語が断りなしに出てくる。しかしそれなしには解説も不可能になってしまうし，本書を通読して頂ければ「あ，そうだったのか」と納得して抱ける筈であるので，あえて余計な解説は付け足さないようにした。従って，この序論は本書を一通り学んだ後に再読して頂くためのもの，とお考え頂きたい。

1.1　FFT を理解するためのキーワード

高速 Fourier[1] 変換 (Fast Fourier Transform)，略して FFT をスクラッチから説明しようとすると，次の事柄がどうしても必要になってくる。

1. 複素解析の初歩的な知識
2. 「関数空間」の基本的な概念
3. コンピュータが実行できる「計算」についての知識

説明の仕方によっては，複素解析を丸っきりすっとばすことも可能だろうが，標準的な考え方としてはあまり教育的とは思われない。トリッキィな説明は，書いている本人にとってはこの上ない愉悦を与えてくれるものだが，聞かされる方にしてみれば迷惑以外の何物でもない，という場合も多い。よって，ここではごく常識的な解説を行うことにした。
以下，FFT なるものを理解するための概説を述べることにしたい。

1.2　複素解析とは?

普通の大学教養課程における解析学，あるいは微分積分学と呼ばれる講義では，実数 ($\mathbb{R}$) を独立変数および従属変数とする関数を対象とした微分や積分の取り扱

[1] 本書では日本人以外の人物名を欧文で表記する。

いについて学ぶことになっている。教師によっては「極限」「連続」という概念をきっちり再定義してから，それを土台とした厳密な理論体系を作りあげるような講義をすることもあろうが，最終的には高校で習った微分積分の延長線上の話に落ち着くので，高校時代，理系クラスに属していた学生さんにとってはとっつきやすい内容ではある。目新しいところといえばせいぜい

- Taylor の定理 (Taylor 展開，Maclaurin 展開)
- 偏微分
- 重積分と累次積分

でないか。主として多変数 (といっても講義ではせいぜい 2 変数どまりだろう) 関数の微積分のところが新しく習う内容でないかと思う。どのみち，実変数 (変数は実数しか取らないという意味) の扱いにとどまっていることには変わりない。

複素解析は，このような実変数関数の微分積分を土台として構築された，複素数を変数とする関数，主として「正則 (holomorphic)」関数を扱う解析学である。その結果として重要な 3 つの定理をここに挙げておく。

Euler の公式 任意の $\theta \in \mathbb{R}$ に対して

$$\exp(\theta\sqrt{-1}) = \cos\theta + \sqrt{-1}\,\sin\theta$$

が成立する。

Cauchy の積分定理 ある領域 $D \subset \mathbb{C}$ で定義される正則関数 $f(z) \in \mathbb{C}$ に対し，D 内部における任意の単純閉曲線 C において

$$\int_C f(z)dz = 0$$

である。

Cauchy の積分公式 単純閉曲線 C とその内部で正則な関数 $f(z)$ と，C の内部の任意の点 z において，

$$f(z) = \frac{1}{2\pi\sqrt{-1}}\int_C \frac{f(w)}{w-z}dw$$

となる。

FFT を理解する上で重要なのは Euler の公式だが，もののついでに Cauchy の積分定理と公式も触れて頂きたい。そうでないと複素解析の「美しさ」が理解できないし，実数にへばり付いていたのでは分からない,「上空からの眺め」を体験したことにならないからである。FFT が複素形式を標準形とするのも，理論に対する美的感覚がそうさせていると言えるのである。

1.3 関数空間とは？

一言でまとめると，関数そのものを集合とする無限次元の線形空間である。行列とベクトルを教えてくれる線型代数学は，高校の教育課程ではほとんど扱わなくなっているが,「システム」というものを理解する上では欠かせない道具立てである。

我々の日常生活を過ごす空間は 3 次元と呼ばれている。空間のどこかに原点 O なるものを設置し，そこを基準としてどのぐらい離れているかによって位置を決める訳だが，その位置決めには 3 つの実数パラメータ x_1, x_2, x_3 が与えられればよい，という空間である。これを座標 (coodinates) と呼び，ベクトル表記では縦に並べて

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

と書く。しかしこれは一つの基底 (bassis) とのセットで表記されるべきものであり，通常は

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},\ \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

という標準的な正規直交基底 (orthonormal basis) を念頭においた上で，ベクトル $\mathbf{x}$ は

$$\mathbf{x} = \sum_{i=1}^{3} x_i \mathbf{e}_i$$

と表記できる，ということを意味している。つまり座標とは基底があって初めて定まるものと言える。ちなみに正規直交とは,「正規」つまり長さが 1 を意味し,「直交」とは直角に交わっていることを意味する。数学的に表現すればユークリッドノルム (Euclid norm)$\|\cdot\|_2$ と内積 (inner product)$(\cdot,\ \cdot)$ を用いて

$$\|\mathbf{e}_i\|_2 = \sqrt{(\mathbf{e}_i,\ \mathbf{e}_i)} = 1,\ (\mathbf{e}_i,\ \mathbf{e}_j) = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$

ということになる。ベクトル同士の内積がゼロになる時，このベクトル同士は直交していると言う。

逆に基底が決まれば，それによって座標が決まる。具体的には内積を用いて

$$x_i = (\mathbf{x},\ \mathbf{e}_i)\ (i = 1, 2, 3)$$

と表現できる。

さて，基底というベクトルのセットは一次独立 (linearly independent) でありさえすれば何でもよく，例えば

$$\mathbf{e}'_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix},\ \mathbf{e}'_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix},\ \mathbf{e}'_3 = \mathbf{e}_3$$

であっても基底としての資格を持つ。従ってどんな 3 次元ベクトル $\mathbf{x}$ でも $\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3$ という基底によって

$$\mathbf{x} = \sum_{i=1}^{3} x'_i \mathbf{e}'_i$$

という形で表現できる。当然のことながら，これは

$$\mathbf{x} = \sum_{i=1}^{3} (\mathbf{x},\ \mathbf{e}'_i)\mathbf{e}'_i$$

とも表現できることになる。内積の計算 $(\mathbf{x},\ \mathbf{y})$ を具体的に書けば，先の標準正規直交基底で定まった座標値 $\mathbf{x} = [x_1\ x_2\ x_3]^T, \mathbf{y} = [y_1\ y_2\ y_3]^T$ を用いると

$$(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{3} x_i \cdot \overline{y_i}$$

となる。

この時，3×3 正方行列 A によって

$$A\mathbf{x} = \mathbf{x}' \tag{1.1}$$

という関係が導かれる。つまり基底ベクトルそのものが

$$A\mathbf{e}_i = \mathbf{e}'_i\ (i = 1, 2, 3)$$

と変換されたとも言える。このような「線型変換」は，有限次元の線形空間であれば必ず行列の形で表現できる，ということを理解してもらうのが「線形代数学」の最終目標なのだが，最近はそこまでやる教師は少ないらしい。しかし本来はそういうことなのである。何を言いたいかというと，線型空間とは何か，ということを復習したかったのである。まとめると

- 空でないベクトルの集合である
- 基底が決まればベクトルの座標が決まる

- ベクトルの長さ=ノルム，内積が定義されている
- ベクトル同士の加減算，スカラー倍が可能
- 線型変換は基底の変換として定義できる

となる。相当ずさんな言い方だが，これが線形空間の特徴である。そして「関数空間」とは，ベクトルの代わりに関数を用い，以上の特徴をもつ無限次元，つまり基底が加算無限個存在する空間のことなのである。

例えば実関数 $\exp(x) = e^x$ は $x = 0$ の周りで Taylor 展開，つまり Maclaurin 展開できて

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} \tag{1.2}$$

と表現できるが，これは $x^0 = 1,\ x^1,\ ...,\ x^n, ...$ という正規でも直交もしていないが基底となっている関数を用いて表現されたものと言うことができる。これは複素数に拡張しても成立する。

本書では最終的に，実数を引数とする，$I = [0,\ 2\pi] \subset \mathbb{R}$ で定義された周期複素関数 $f(\theta) = f(\theta \pm 2\pi) \in \mathbb{C}$ を対象とするが，これが全てこの区間 I で積分可能であれば，関数 f と g の内積を

$$(f,\ g) = \int_0^{2\pi} f(\theta)\overline{g(\theta)}\, d\theta$$

とし，これに基づいて関数 f のユークリッドノルムも

$$\|f\|_2 = \sqrt{(f,\ f)} = \sqrt{\int_0^{2\pi} |f(\theta)|^2\, d\theta}$$

と定義できる。

そして内積とノルムが定義されたことによって，関数空間においても直交基底というものが定義でき，例えば

$$e_0(\theta) = \frac{1}{\sqrt{2\pi}} \exp(0\theta\sqrt{-1}),\ e_1(\theta) = \frac{1}{\sqrt{2\pi}} \exp(1\theta\sqrt{-1}),\ ...,\ e_n(\theta) = \frac{1}{\sqrt{2\pi}} \exp(n\theta\sqrt{-1}),\ ...$$

がそれにあたる。

当然，基底が決まれば「座標」も決まる訳で，関数 $f(\theta)$ が与えられれば

$$f(\theta) = \sum_{n=0}^{\infty} (f, e_n) e_n(\theta)\, d\theta = \sum_{n=0}^{\infty} c_n \exp(n\theta\sqrt{-1})$$

と表現でき，各座標値，すなわち係数 c_n は

$$c_n = \frac{1}{\sqrt{2\pi}}(f,\, e_n) = \frac{1}{2\pi}\int_0^{2\pi} f(\theta)\exp(-n\theta\sqrt{-1})\,d\theta \tag{1.3}$$

となる訳である。このように，ある関数 f が与えられた時に上記の級数に変換することを，Fourier 変換と呼び，結果として得られた級数を Fourier 級数，係数 (座標値)c_n を Fourier 係数と呼ぶ。この Fourier 変換は無限次元の線型変換の一種とも言える。

理論的にはかなりずさんな表現をしたが，高速でない「普通の」Fourier 変換は，関数を無限次元に展開したときの表現の一種ということがご理解いただけただろうか？

1.4　理論は分かったが，実際の計算は？

以上で述べたことをまとめると，Fourier 変換のためには

1. 実引数の複素関数 $f(\theta)$ が与えられる
2. Fourier 係数 $c_n = \frac{1}{2\pi}\int_0^{2\pi} f(\theta)\exp(-n\theta\sqrt{-1})d\theta$ を計算する

という手順で計算すればいいことになる。さてこれをコンピュータで「効率的に」行うにはどうしたらいいだろうか？

しかし，そもそも Fourier 変換を「厳密に」実行することは不可能である。その理由は 2 点ある。

- 無限個の c_n を有限の時間内で得ることは不可能である。
- 定積分可能な関数ではあっても，その原始関数を有限個の初等関数の代数的な組み合わせで表現することはできないことがある。

最初の理由は説明するまでもないだろう。従って，現実問題としては適当に大きな $N \in \mathbb{N}$ を決めて，有限個の $c_0, c_1, ..., c_{N-1}$ を得ることで我慢するほかない。つまり (無限)Fourier 級数の「近似 (approximation)」との有限 Fourier 級数に留める必要がある。

問題は定積分の計算である。微分積分の教科書では，定積分 $\int_a^b g(x)dx$ の計算をする際には元の関数の原始関数 $G(x)$ ($G'(x) = g(x)$) を用いて

$$\int_a^b g(x)\,dx = G(b) - G(a)$$

とすればいい，と書いてあるが，この $G(x)$ が初等関数の有限個の代数的組み合わせで表現できるとは限らない。代表的なものとしては楕円積分

$$\int_0^k \frac{\sqrt{1-k^2x^2}}{\sqrt{1-x^2}}\,dx\ (0<k<1) \tag{1.4}$$

がある。このような事情は微分では発生しないため，現在ではプログラムで代数的に偏導関数を計算する自動微分 (automatic differentiation) という技法が定着している。しかし積分に関しては，厳密な計算は一般に不可能なので，定積分の値を得るには「近似」するほかない。そのためには定積分の定義に立ち返って考える必要がある。

よくある定積分の定義は，積分区間 $[a,b]$ を m 等分割して m 個の小区間 $[x_i, x_{i+1}]$ $(x_i = a+ih, h = (b-a)/m)$ ごとの関数の最小値 $g_i^{\min} = \min_{t\in[x_i,x_{i+1}]} g(t)$ と最大値 $g_i^{\max} = \max_{t\in[x_i,x_{i+1}]} g(t)$ を定め

$$\begin{aligned}\Delta_m^{\min} g[a,b] &= h\sum_{i=0}^{m-1} g_i^{\min}\\ \Delta_m^{\max} g[a,b] &= h\sum_{i=0}^{m-1} g_i^{\max}\end{aligned}$$

という二つの部分和を取り，$m\to+\infty$ の極限を取って

$$\Delta_\infty g[a,b] = \lim_{m\to+\infty}\Delta_m^{\min} g[a,b] = \lim_{m\to+\infty}\Delta_m^{\max} g[a,b]$$

とこの二つの和が一致するとき，その極限値 $\Delta_\infty g[a,b]$ を定積分の値とする，即ち

$$\Delta_\infty g[a,b] = \int_a^b g(x)\,dx$$

とする。この際，どうせ一致するなら，小区間ごとに端点の平均値 $g_i^{\mathrm{ave}} = (g(x_i) + g(x_{i+1}))/2$ の値をとって

$$\Delta_m^{\mathrm{ave}} g[a,b] = h\sum_{i=0}^{m-1} g_i^{\mathrm{ave}}$$

という和を作っても，$m\to\infty$ とすれば定積分値に一致する。そして，程々に大きな $M\in\mathbb{N}$ にしておけば

$$\Delta_M^{\mathrm{ave}} g[a,b] \approx \int_a^b g(x)\,dx$$

ということを期待しても，それほど的外れではないだろうという予測ができる。さらに言えば，$\Delta_M^{\min}$ や $\Delta_M^{\max}$ よりは「より近い」値になることが多いだろうというこ

とも想像できるだろう。このようにして「連続的 (continous)」な極限表現を「離散的 (discrete)」な有限表現にして初めて定積分は現実のコンピュータの計算に載せることができるのである。

以上を納得して頂いた，ということにすると，結局 Fourier 係数の計算では，$g(x) = f(x)\exp(-2\pi x\sqrt{-1})$ として，近似的に

$$c_k \approx C_k = \Delta_M^{\mathrm{ave}} g[0, 2\pi] = \frac{1}{M}\sum_{i=0}^{M-1} f_i \omega_M^{-ik} \tag{1.5}$$

を計算すればいいことになる。ここで f_i と ω_M は

$$f_i = f\left(\frac{2\pi i}{M}\right),\ \omega_M = \exp\left(\frac{2\pi}{M}\sqrt{-1}\right)$$

である。ある程度大きな M であれば問題ないので，以降では $M = N$ とすることにしよう。

このように，有限 Fourier 変換の係数を離散的に計算したものを，離散 Fourier 変換 (DFT) と呼ぶ。まとめると

$$\text{(無限)Fourier 変換} \xrightarrow{\text{有限和}} \text{有限 Fourier 変換} \xrightarrow{\text{定積分の離散化}} \text{離散 Fourier 変換 (DFT)}$$

ということになる。

さすれば結局，DFT に必要なデータと計算は，N 個の複素数値

$$f_0,\ f_1,\ \ldots,\ f_{N-1} \in \mathbb{C}$$

が与えられた時に，N 個の複素数

$$C_0,\ C_1,\ \ldots,\ C_{N-1} \in \mathbb{C}$$

を，それぞれ

$$C_k = \frac{1}{N}\sum_{i=0}^{N-1} f_i \omega_N^{-ik}\ (k = 0, 1, \ldots, N-1)$$

として積和の計算をすることで得る，ということになる。

この際，必要な計算量は，あらかじめ ω_N^{-ik} を求めておいたとすれば，複素数の乗算が N 回，加算が $N-1$ 回になる。コンピュータに限らず，計算としては加算より乗算の方が時間がかかるので，以降では乗算の回数のみを考えることにする。そうすると，$C_0, C_1, \ldots, C_{N-1}$ の計算量を合計すると N^2 回必要ということになる。DFT に必要な計算時間はこの計算量に比例して決まるから，例えば $N = 1024$ の時に 1 秒 で計算が終了したとすると，$N = 4096 = 4 \times 1024$ の時にはおおよそ

$4^2 = 16$ 倍の計算時間がかかると予想できるので，16 秒 ということになる。もし $N = 1024 \times 1024$ であれば，

$$1024^2 \times 1 \text{ 秒} = 1048576 \text{ 秒} \approx 271 \text{ 時間} \approx 12 \text{ 日}$$

の計算時間ということになる。Fourier 変換は，周波数解析や画像処理には欠かせないツールなので，扱うデータは膨大なものになるのが普通である。むしろ，膨大なデータから「成分」を取り出せることが目的なので，計算時間は可能な限り短くしたい。

この N^2 に比例する計算量の DFT を高速にしたものを，FFT，高速 (離散)Fourier 変換と呼ぶ。1964 年に J.Cooley と J.Tukey によって定式化され，1965 年に 5 ページの論文にまとめられたものである。ここでは結果のみ述べるが，$\omega_N = \exp(2\pi\sqrt{-1}/N)$ の性質をうまく使うと計算量は $N \log_2 N$ に比例するところまで減らせることが明確になったのである。

そうすると，12 日もかかっていた計算は

$$1024 \log_2 1024 \times 1 \text{ 秒} = 1024 \times 10 \text{ 秒} = 10240 \text{ 秒} \approx 2.8 \text{ 時間}$$

で終了してしまうことになる。実際にはこれの定数倍の計算時間が必要になるが，それでも劇的な効果が，特に大規模データに対してはあることが分かる。

現実のデータは実数値であることがほとんどなので，わざわざ複素数で表現する理由が分からない，という声もあるかもしれない。それに対しては，「理論的な美的感覚」がそうさせているのだ，という答えで十分であろう。複素数で表現された理論は，そのまま実数に対しても適用できるのである。しかしその逆は成立しないのだ。

練習問題

1. (1.1) 式の正方行列 A を具体的に求めよ。その導出方法も説明せよ。
2. $k = 1/3$ の時の楕円積分の近似値を求めよ。ただし，10 進 2 桁以上の精度を持つように定めること。
3. $\exp(1.5)$ の値を先頭から二桁以上の精度で求めたい。(1.2) の右辺の級数を用いてこの値を求めるためには，項数 n はいくつ以上にする必要があるか？
4. $N = 1024$ の時に 0.5 ミリ秒 で FFT の計算が完了したとする。$N = 32768$ の時は何秒 (何時間) 程度かかると予想できるか答えよ。