Data compression using the Fourier transform

Keita Haruyama

2025年10月28日

目次

1　はじめに

本書では、フーリエ変換を応用したデータ圧縮アルゴリズムの草案をまとめている。対象とするのは、膨大なデータ量を持つビッグデータ環境における効率的な圧縮手法であり、フーリエ変換を用いることでデータの本質的な特徴を抽出・再構成し、冗長部分を削減することを目的としている。

1.1　背景

近年、データ解析や機械学習の発展に伴い、数 TB 規模のビッグデータを扱う機会が増えている。これらのデータは大企業のハイスペックサーバ上で解析される一方、個人が所有しているローカル PC の環境では処理能力やメモリ容量が不足しており、全データを高精度に扱うことは現実的ではない。

現状では、ビッグデータの解析には専用の高性能マシンが必要であり、ノート PC 等の一般的な環境では、検証すら十分に行えない場合が多い。また、ビッグデータ解析に対応できるエンジニアは、上記のような検証環境の制限により、常に人材が不足している状況にある。

本データ圧縮技術は、この課題を解決するための手法として提案される。数 TB のデータを数 MB レベルまで圧縮することで、多くのエンジニアが手元の環境でも大容量データの解析を容易に行えるようにすることを目指す。

この実現にあたっては、「すべてのデータを正確に再現する必要は必ずしもない」という観点に基づき、一定の精度を犠牲にしても高い圧縮率を達成できる手法を検討した。その結果、フーリエ変換を基盤とするデータ圧縮アプローチに注目した。

2　データ圧縮の構想概要

データ圧縮の構想について、データの関数化を行うことで、データの圧縮を行う。

2.1　データの関数化とは

データ関数化とは、データという「点や行の集まり」を、ひとつの関数として表現・保存するという発想である。すなわち、個々の観測値を点列として保持する代わりに、それらを生成する関数（もしくは近似関数）を求めることで、情報をより圧縮的に表現できる。

たとえば、次のようなデータが与えられているとする。

x	1	2	3	4	5
y	2	4	6	8	10

それらの点をプロットすると以下のようになる。

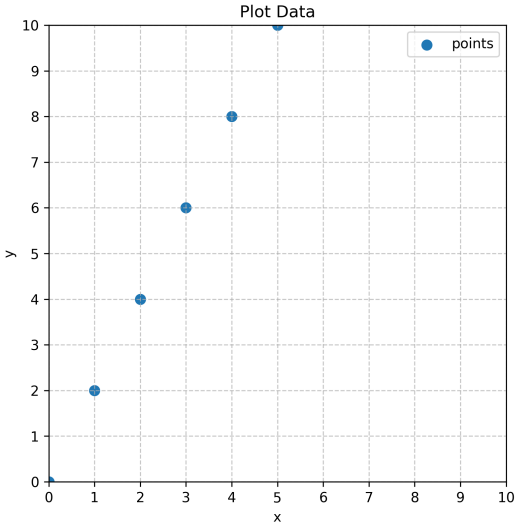

図 1　サンプル画像

この x と y 関係性は、次のような一次関数で表すことができる。

$$y = 2x \tag{2.1}$$

この関数表現の利点は、$y = 2x$ という情報だけを保持しておけば、大量のデータを保存する必要がなくなる点である。

例えば、x が 0 から 1 億までのデータを持っていた場合、それに対応するすべての y の値を個別に保存すると非常に大きな容量が必要となる。

さらに、データの集計や特定の値（例：$x = 3000$ のときの y の値）を取り出す処理には、データベースの参照や計算リソースが必要である。

しかし、$y = 2x$ という関数の形で情報を保持していれば、必要な時に任意の値をすぐに計算して取り出すことができる。そして集計した値を求めるときは積分として変換する、等の数値計算としても利用可能となる。

このように、データを単なる数値の列としてではなく関数の形式で表現することで、本質的な構造を保ちながらデータ量を大幅に削減できる可能性がある。

2.2　複雑なプロットデータの場合

上記で述べた一次関数による表現は、単純な直線的関係のデータにのみ適用可能である。しかし、現実のデータはより複雑であり、単純な直線では表現できない場合が多い。

例えば、図 2 に示すような複雑な点列が存在するとする。このようなデータを通る関数を直線で表現することはできない。

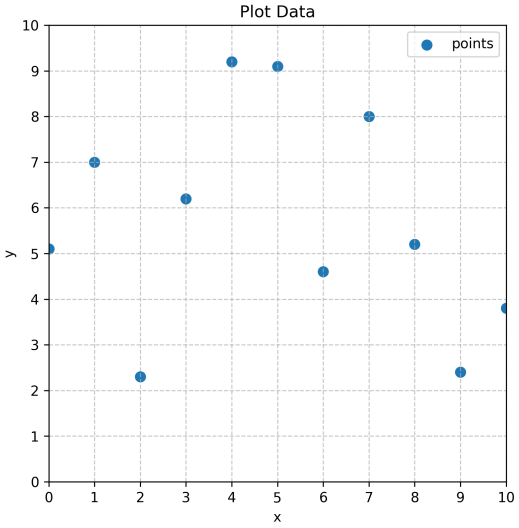

図 2　複雑な点のプロット

これらを通る関数は直線で表現することができない。

これらの複雑なプロットデータを関数によって表現する場合、必ず近似誤差が生じることになる。しかし、ビッグデータの性質上、個々のデータ点の正確性よりも、全体のデータ容量や傾向が予測精度に与える影響の方が大きい。したがって、100% 正確なデータでなければ解析精度が低下する、ということは基本的に起こらないと考えられる。

本研究では、この関数近似による誤差の許容範囲と、データ圧縮による予測精度への影響を検証することを視野に入れている。すなわち、データを基にした予測精度の正確性と圧縮率とのトレードオフを踏まえた上で、効率的なデータ関数化の手法を構築することを目的とする。

2.3　フーリエ級数展開を用いた関数表現

こうした複雑なプロットデータを表現するために、フーリエ級数展開を用いる。フーリエ級数展開とは、任意の周期関数を正弦関数（sin）や余弦関数（cos）の線型結合として表す手法である。

言い換えれば、複雑に見える波形も、異なる周波数の単純な波の重ね合わせとして再現できるという考え方である。

フーリエ級数展開の定義は以下である。

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \Big(a_n \cos(nx) + b_n \sin(nx) \Big) \tag{2.2}$$

$$a_0 = \frac{1}{\pi} \int_0^{2\pi} f(x)\, dx \tag{2.3}$$

$$a_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos(nx)\, dx, \quad n \geq 1 \tag{2.4}$$

$$b_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin(nx)\, dx, \quad n \geq 1 \tag{2.5}$$

直感的には、次のように理解できる。単純な正弦波・余弦波である $\sin x$ や $\cos x$ の形は滑らかで単純だが、これを高さや周期を変えながら重ねることで、複雑な波形や点列の形状を再現できる。

この手法を用いることでデータを関数形式で保持しつつ、元データの量を大幅に削減できると目論んでいる。

ただし、フーリエ級数展開では本来、無限個の正弦波・余弦波を組み合わせる必要がある。このままでは無限の計算が必要となってしまうため、実際には適切な項数で打ち切り、精度と計算コストのバランスをとる必要がある。

一方で、フーリエ級数展開の大きな利点は、一度係数を計算してしまえば、データが追加されない限り再計算の必要がない点である。初回の関数化には一定の計算時間を要する場合があるが、関数が完成すれば以降はその関数を用いて任意の値を取得できる。この性質により、データアクセスや集計の効率化が大幅に向上する。

3　2 次元の数値データにおける検証

まずは、シンプルなデータ検証から行う。x に対して y という数値が割り当てられているという状況を考える。

そのため、まずは単純な数値データに対して、関数化を行う。ビッグデータは、数値・文字や構造・非構造等の複雑な形式をとっているため、のちに、文字データや非構造化データについても取り扱う予定である。

3.1　単純なデータの準備

まず、x が 0 から 100 までで、ランダムなデータを投入したデータを準備する。

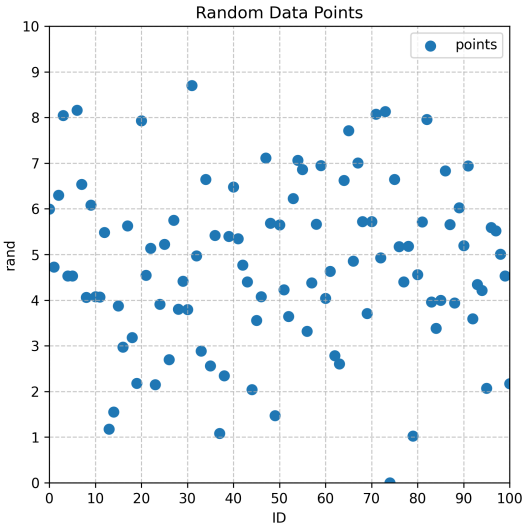

図 3　x が 0 から 100 までのランダムなデータ

上記の点を一旦、以下のようなステップ関数として扱う。(x 軸は見やすさのため 0 から 20 までの表示とする。)

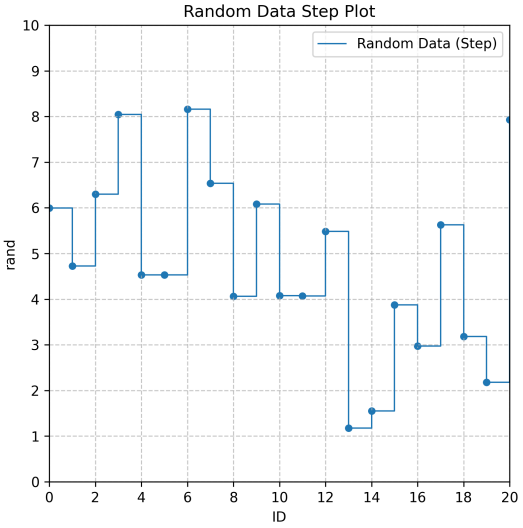

図 4　x が 0 から 20 までのランダムなデータ

ステップ関数として扱う理由は、フーリエ級数展開の係数計算の短縮の目的がある。
これらの近似関数をフーリエ級数展開で表現する。

3.2　検証結果 (記載中)

一旦検証結果のみ記載する。

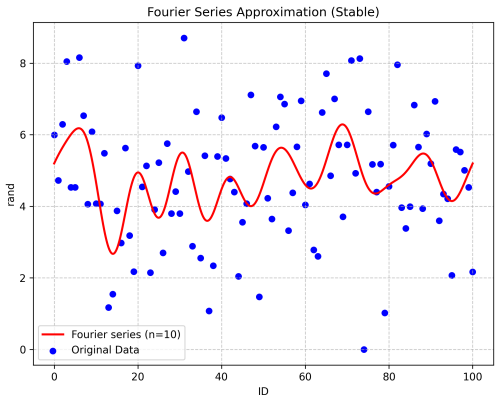

図 5　基底 10 次までの近似

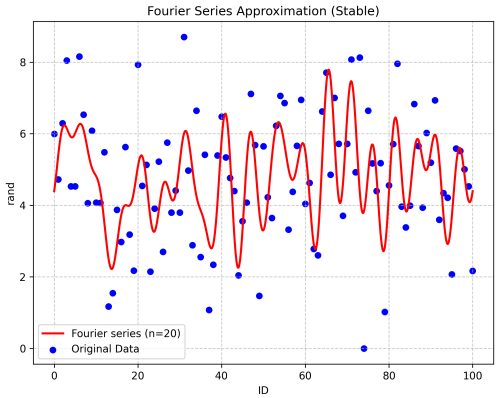

図 6　基底 20 次までの近似

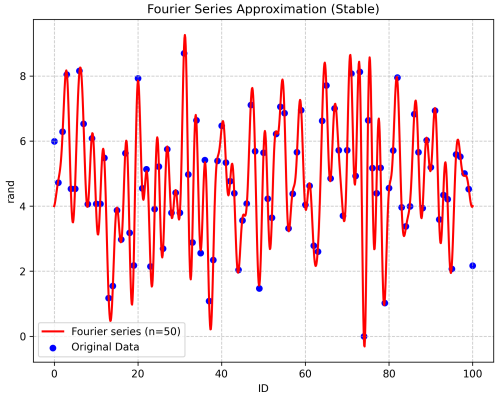

図 7　基底 50 次までの近似

0-20 までの間のプロット

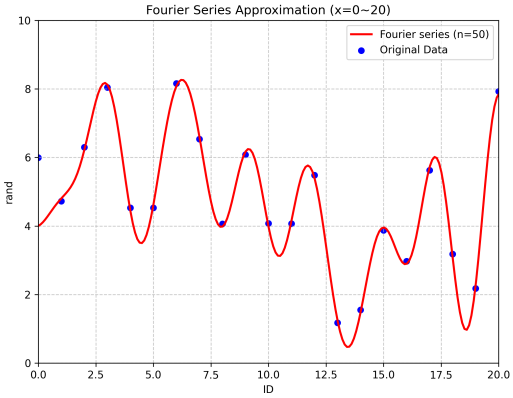

図 8　基底 50 次までの近似 x が 0 から 20 までの間

付録 A　フーリエ級数展開の基礎復習

本ページでは、LaTeX や数学の初歩的な内容をまとめます。フーリエ級数や積分などを学ぶ前に、基礎を復習することができます。

A.1　数式モードと文章モード

- 数式モード：‘...‘（インライン）、‘

...

‘（独立表示）- 文章モード：普通のテキスト
例：

- インライン： $y = \sin x$
- 独立表示：

$$f(x) = \frac{1}{\pi} \int_0^{2\pi} f(t) \sin(nt)\, dt$$

A.2　よく使う数学コマンド

- $\frac{a}{b}$ ：分数
- $\sum_{n=1}^{\infty}$ ：総和
- $\int_0^{2\pi}$ ：積分
- $\sin, \cos, \log$ ：立体関数

A.3　文章中での関数表記

- 正しい書き方：

- 正：「正弦関数（$\sin x$）」
- 誤：‘$sinx$‘ （変数扱いになり斜体）

A.4　図の挿入の基本

“‘latex

図 9　サンプル画像

付録 B　Python 積分の数値計算

付録 C　Python 基本コード復習