数値解析基礎

工学博士　安　田　仁　彦　著

コロナ社

ま え が き

工学の分野で数値解析の果たす役割はますます重要になっている。本書は，数値解析の広範な内容のうち，機械系あるいはその関連の工学の分野で重要と思われる内容を中心に，数値解析法の基礎を解説したものである。

今日，数値解析のためのツールが発達し，使いやすくなっている。しかし汎用のツールでは対処できない問題も多く，これに対処できる力を養うため，また，ツールで用意されている種々の方法を適切に利用する指針を得るため，さらに，得られた結果を適切に評価する数量的感覚を養うため，数値解析法を学ぶことの重要性はいささかも減じていない。

これまで著者は何冊かの著書を出版し，幸い好評をいただいてきた。本書で数値解析法を解説するにあたっても，著者のこれまでの執筆方針を踏襲し，易から難へとゆっくり話をすすめるよう心がけた。また数値解析の特徴を示す部分をていねいに説明し，数値解析の面白さを伝えられるよう心がけた。さらに章の順序を，数値解析の考え方に少しずつ慣れていただくという観点から定めた。これらの執筆方針がどの程度実現されているかは読者に判断していただくほかはないが，本書が読みやすく，いささかでも感動を与えられるものとなっていることを祈念している。

本書は，著者が長年にわたって企業で行ってきた研修のためのノートをもとにして，名古屋大学や愛知工業大学での講義，研究指導の際の資料を追加し，全体を整理して書き上げたものである。研修や講義に際して，多くの受講者，学生からいただいた質問やコメントは，本書を仕上げるにあたってきわめて有用であった。

本書の草稿に対して，多くの同僚から貴重なコメントをいただき，本書を改善するのに役立たせていただいた。お名前はここでは省略させていただくが，これらの方々に感謝申し上げたい。

数値解析を学ぶ際に，また関連分野で数値解析を活用する際に，本書が読者に少しでも役立つならば，著者の大きな喜びである。

2008 年 10 月

安田　仁彦

目　　次

1. 数値解析の基礎

2. 非線形方程式

3. 補 間 多 項 式

4. 数 値 積 分

5. 連立1次方程式

6. 行列の固有値問題

7. 常微分方程式

8. 偏微分方程式

1 数値解析の基礎

数値解析は，工学の分野で出会う現実の数学的問題の解を得る強力な手段である。この章で，数値解析をコンピュータで行うときに必要となる基礎事項を述べる。

1.1 数 値 解 析

工学の分野で出会う現実の数学的問題の解を理論的に求めることは一般には難しい。このためコンピュータを用い，適当な**数値解析**（numerical analysis）の方法によって問題を扱う。数値解析法は問題解決の強力な手段となっている。

数値解析は一連の計算によって行われる。この一連の計算を行うための計算手順を**アルゴリズム**（algorithm）という。このアルゴリズムに従ってコンピュータに計算を行わせるため，コンピュータが理解できる言語を用いてアルゴリズムを具体的に記述したものを**プログラム**（program）という。アルゴリズムはプログラムの基本をなす。アルゴリズムの良否によってプログラムの良否が決まり，これによって，同じコンピュータを用いても数値解析の精度や計算時間が異なる。

本書の目的は，数値解析を利用して問題解決をしたい読者に，基本的で優れたアルゴリズムをわかりやすく紹介して，問題に適したアルゴリズムを考えるヒントを提供すること，多数あるアルゴリズムの適切な利用法を示すこと，数値解析によって得られる結果を適切に評価する数量的感覚を養っていただくこ

とである。以下の各章で，種々の問題に対する基本的なアルゴリズムを紹介する。

各論に入る前に，ここで一般的にアルゴリズムが備えるべき条件を，つぎの問題を例にして考えよう。

問題　係数 $a_n, a_{n-1}, \cdots, a_0$ を与えられた定数として，変数 x の n 次多項式

$$f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0 \tag{1.1}$$

がある。変数 x にいろいろな値を与えて $f(x)$ の値を求めるアルゴリズムを考えよ。

この問題に対して，つぎのような計算手順が考えられる。変数 x に値を与えてまず $x^2, x^3, \cdots, x^n$ を計算する。つぎに，$x, x^2, x^3, \cdots, x^n$ の各値にそれぞれ定数 $a_1, a_2, \cdots, a_n$ を掛けて $a_1x, a_2x^2, \cdots, a_nx^n$ を計算する。最後に，これらの各値と a_0 の値の和を求める。この場合に，結果が得られるまでの演算回数を調べると，x^n を求めるまでの乗算 $n-1$ 回，$a_1x, a_2x^2, \cdots, a_nx^n$ を求めるための乗算 n 回，和を求めるための加算 n 回で，全部で $3n-1$ 回である。

同じ問題に対して，別の計算手順を考えよう。与えられた関数 $f(x)$ は

$$\left.\begin{aligned}
y_1 &= a_nx+a_{n-1}\\
y_2 &= y_1x+a_{n-2}=a_nx^2+a_{n-1}x+a_{n-2}\\
&\ \vdots\\
y_n &= y_{n-1}x+a_0=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0
\end{aligned}\right\} \tag{1.2}$$

の y_n によって与えられることに注目する。そこでまず，$y_1=a_nx+a_{n-1}$ を求める。つぎに，得られた y_1 を用いて $y_2=y_1x+a_{n-2}$ を求める。続いて，得られた y_2 を用いて $y_3=y_2x+a_{n-3}$ を求める。以下，これを繰り返して $y_n=y_{n-1}x+a_0$ を求める。これが求める結果である。この結果を得るまでの演算回数は，1 回の乗算と 1 回の加算とを n 回繰り返した $2n$ 回である。

両者の計算手順のよしあしを比較してみよう。まず演算回数は，前者より後者の方が少ない。また前者の手順では，途中の値をどこかに記憶しておく必要があって手間がかかるのに対し，後者の手順では，途中の値をそのままつぎの

計算に用いるので，あらためて記憶させる手間がいらない。このように，後者の手順は演算回数と手間の点で優れており，よいアルゴリズムといえる。

一般にアルゴリズムとしては，演算回数が少なく，手間のかからないものがよい。

1.2　コンピュータ内の数値表現

コンピュータの内部では，すべての数値は2進数に置き換えて処理される。これは，2進数が電子回路のON / OFFと対応させることができて都合がよいからである。2進数の1桁を**ビット**（bit）という。

コンピュータの記憶容量は有限であるから，一つ一つの数値に無制限に桁数を与えることはできない。コンピュータ内で数値がどのように表現されているかを見ておこう。

コンピュータ内で数値は，以下に述べる整数型と実数型の二つに分類され，それぞれに異なった表現が与えられる。この表現は，整数型については早くからほとんどのコンピュータで同じであったが，実数型については，1970年代まではIBM方式が，1985年以降はIEEE方式が主流となっている。細かいところはコンピュータによって異なるが，以下，表現の一例を示す。

整数型（integer type）の数値とは，10進数でいう0と正負の整数をいう。整数型の数値のため，コンピュータ内では**図1.1**に示すように，ふつう32ビットを割り当てる。32ビットのうち1ビットを符号部に使い，残りの31ビットを0または1として2進数で数値を表現する。扱える整数型の数値の絶対値の最大値は，31ビットがすべて1となる数である。この値を10進法で求めると

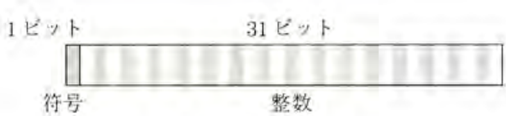

図1.1　整数型の数値

$$I_{\max}=2^0+2^1+\cdots+2^{30}=2\,147\,483\,647 \tag{1.3}$$

となる。この数値は約21億で，整数で行うふつうの計算では，この値は十分大きい。

実数型（real type）の数値とは，10進数でいう1.23や-4.56のような小数点を持つ数値をいう。実数型の数値は，コンピュータ内で

$$d=\underbrace{\pm}_{\text{符号部}}\ \underbrace{(0.d_1d_2\cdots d_N)}_{\text{仮数部}}\times 2^{\overbrace{p}^{\text{指数部}}} \tag{1.4}$$

という形で表して，それぞれにビットを割り当てる。ここでNは仮数部のビット数，pは指数の大きさである。仮数部の中の小数点のつぎの数d_1は0でない数値，d_2から後は0または1である。d_1は0でない数値となるまでd_1より後にある数値を左へ詰め，詰めた分に応じて指数部のpを減らすという**正規化**（normalization）を行う。

実数型の数値に割り当てるビット数は，扱いたい数値の有効数字によって変えている。**単精度**（single precision）の数値では32ビットを，**倍精度**（double precision）の数値では64ビットを割り当てる。それぞれのビット数のうち，1ビットを符号部にあて，単精度の場合は残りの31ビット，倍精度の場合は残りの63ビットを仮数部と指数部にあてる。仮数部と指数部へ割り当てるビット数は，IEEE方式の場合，**図1.2**に示すように，単精度の場合8と23，倍精度の場合11と52である。

IEEE方式の実数型の数値の範囲を見積もってみよう。指数部に与えるビッ

図1.2　IEEE方式の実数型の数値

ト数が決まっていることから，式(1.4)の p の範囲が決められる。この範囲を正負でほぼ同じになるよう定め，単精度の場合，例えば

$$-126 \leqq p \leqq 128 \tag{1.5}$$

とする。このとき 10 進数で表して，数値の範囲は絶対値で最小約 10^{-38} から最大約 3×10^{38}，有効桁数は約 7 桁である。倍精度の場合，式(1.4)の p を例えば

$$-1\,022 \leqq p \leqq 1\,024 \tag{1.6}$$

とする。このとき 10 進数で表して，数値の範囲は絶対値で最小約 10^{-308} から最大約 3×10^{308}，有効桁数は約 16 桁である。

コンピュータは，上述のようにある値より大きい絶対値を持つ数値を記憶することはできない。この状態を，コンピュータは**オーバーフロー**（overflow）するという。計算の途中でこの状態になると，演算結果が正しいかどうかは保証されなくなる。また 0 を除いて，絶対値の最小値にも制約がある。この値より小さい絶対値を持つ数値を記憶させようとすると，コンピュータは**アンダーフロー**（underflow）となる。この数値に対しては，コンピュータはふつう 0 として扱うので，多くの場合問題にならない。

1.3 誤　　差

1.3.1 誤　　差

まず一般的な誤差を定義しておく。ある数の真の値を a，その近似値を x とするとき，両者の差を近似値 x の**誤差**（error）という。ふつう近似値が大きいとき誤差が正となるように両者の差をとり，誤差 e を

$$e = x - a \tag{1.7}$$

と定義する。この誤差の絶対値 $|e|$ を**絶対誤差**（absolute error）という。絶対誤差に対して

$$|e| \leqq \varepsilon \tag{1.8}$$

となる ε を見積もることができるとき，この ε を**誤差限界**（limit of error）

索　　引

――著者略歴――

1963年　名古屋大学工学部機械学科卒業
1968年　名古屋大学大学院工学研究科博士課程修了（機械工学専攻）
　　　　工学博士
1968年　名古屋大学助手
1970年　名古屋大学講師
1976年　名古屋大学助教授
1985年　名古屋大学教授
2004年　名古屋大学名誉教授
　　　　愛知工業大学教授
2013年　愛知工業大学退職

数値解析基礎

Fundamentals of Numerical Analysis

2008年12月30日　初版第1刷発行
2020年9月5日　初版第7刷発行

検印省略

著　者　安田仁彦（やすだ きみひこ）
発行者　株式会社　コロナ社
　　　　代表者　牛来真也
印刷所　壮光舎印刷株式会社
製本所　株式会社　グリーン

112-0011　東京都文京区千石4-46-10
発行所　株式会社　コロナ社
CORONA PUBLISHING CO., LTD.
Tokyo Japan
振替00140-8-14844・電話(03)3941-3131(代)
ホームページ　https://www.coronasha.co.jp

ISBN 978-4-339-06097-3　C3041　Printed in Japan　（横尾）