第 2 章

MATLAB の基礎 1 — 基本操作

本文書は MATLAB を用いて，手計算だけでは難しい複雑な線型計算を行うための準備訓練を行う。この後はベクトルと行列を多用する手法を学ぶので，手軽にこれらの計算が行える MATLAB の線型計算機能の概要を知らねばならない。以下，ここに提示した実行例は自分の手で入力し，何が実行され，どういう結果が出たのか，そして，その結果の正しさをどう確認するのかを理解して頂きたい。

2.1 MATLAB とは？

MATLAB[3] はアメリカ合衆国にある MathWorks 社が開発した統合型数値計算ソフトウェアである。C/C++, Fortran のようなコンパイラ言語からは複雑な呼び出し方が必要となる LAPACK のような数値計算ライブラリを，使いやすいスクリプト言語を介して簡単に呼び出すことができるようにしたもので，現在では Scilab[4] や Octave 等，様々な統合型数値計算ソフトウェアが存在するが，その源流となった最古参のソフトウェアがこの MATLAB である。

MATLAB は商品なので，使用に当たっては適切な対価を支払い，使用のためのライセンスを得る必要がある。インストール時にもライセンスホルダーの情報が必要となり，インストール後にはインターネットを通じてのアクティベーションが必須である。インストールとライセンスについての情報は，適宜，担当者（講義の担当教員等）から得ること。

2018 年 1 月現在の最新バージョンは R2017b で，Windows 10 や，macOS, Linux 上で使用できる。今のところ，スマートフォンやタブレット (Android, iOS) 上でネイティブに動作するバージョンは存在しない。モバイル環境で使いたい場合は，MATLAB Mobile といった Web 経由で使用するツールが提供されており，これを使うことになる。

かように, 40 年近い歴史を誇り, 現在も現役の強力な統合型数値計算ツールである MATLAB は，数値計算だけでなく，グラフィックスや音声，理工学分野に特化したツールを多数揃える

に至った。MuPAD[7] に由来する数式処理機能も統合し，近年ではデータサイエンスや機械学習，AI を手軽に利用できるツールも備え，その用途は広がりを見せている。

2.2　IEEE754-1985 単精度・倍精度浮動小数点数のデータ構造

科学技術計算では，有限桁の小数を表現するために浮動小数点数 (floating-point number) 形式を利用する。現在のコンピューター上で良く用いられているのは IEEE で規格化されているもので，一般には IEEE754 standard と呼ばれているものである。

この IEEE754 規格の浮動小数点数の基数 (base) は 2 であり，図 2.1 に示すような有限桁の浮動小数点表現が定められている。単精度・倍精度の仮数部が実際の bit 列より 1bit 分長いのは，最初の bit が必ず 1 になるように正規化されるため，実際にはその分は削ってあるからである。これをケチ表現と呼ぶ。C や Fortran 等のプログラミング言語で単精度・倍精度の値を

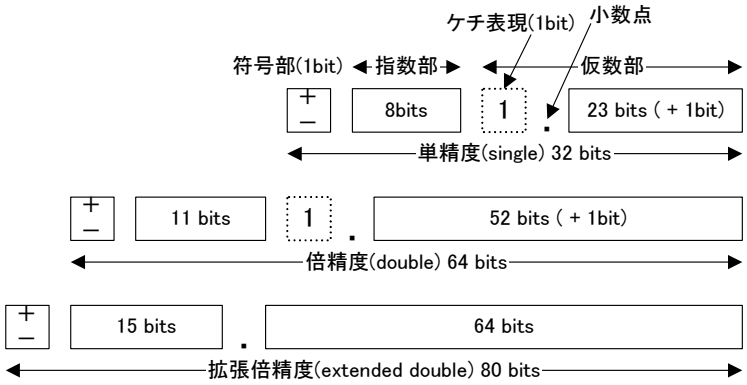

図 2.1　IEEE754 standard

10 進数の文字列として出力すると，例えば 10.65387 は

```
10.65387
0.1065387e+02
1.065387e+01
1.065387e+1
1.065387e1
1.065387E+01
1.065387D+01
1.065387Q+01
```

のようになる。`e` や `E`，`D`，`Q` の前が 10 進数に変換後の仮数部，後ろが指数部を示している。

有限桁の浮動小数点数は離散的に存在するため，表現可能な数の間には隙間ができる。この隙間は指数部が大きくなるにつれて，絶対的な距離は大きくなるが相対的にはほぼ同じという特徴がある。この間隔をマシンイプシロン (machine epsilon) と呼ぶ。これにも幾つかの定義方法があるが，本書では以下のように定義する。

定義 2.1 (マシンイプシロン)

1 の次に大きい最小の正の有限桁の浮動小数点数を $1 + \varepsilon_M$ と表現する時，この ε_M をマシンイ

プシロン (machine epsilon) と呼ぶ。特に基数 m で仮数部が p 桁の場合,

$$\varepsilon_M = m^{-(p-1)} \tag{2.1}$$

となる。

従って，IEEE754 standard の場合のマシンイプシロンは

単精度 $2^{-23} \approx 1.1920928955078125 \times 10^{-7}$
倍精度 $2^{-52} \approx 2.220446049250313080847263336181 6 \times 10^{-16}$

となる。

この IEEE754 単精度，倍精度浮動小数点表現をを用いてコンピューターで実行される計算には，マシンイプシロン程度の丸め誤差 (round-off error) が含まれているものと想定しておくと，ゴミのような数値が計算結果の下の桁に現れることも納得できる筈である。

2.3 絶対値と丸め誤差

絶対値の数学的性質については既に述べた。

本書で扱うコンピュータ上の線型計算は，前述したように有限桁の浮動小数点数に基づくものである。従って，計算過程において，短い桁に「丸め」られることが多い。そのため，実際に得られた数値 $\widetilde{a}$ と，真の値 a とのズレが生じる。このズレを誤差 (error) と呼び，誤差を伴う数値 $\widetilde{a}$ を真値 a に対する近似値 (approximation) と呼ぶ。

例えば，円周率 $\pi = 3.14159265...$ という無限小数で表現するほかない実数は，π が持つ数学的性質を持った特殊なデータとして記号的に扱うか，有限桁の小数として表現する他ない。前述したように，本書では後者の取り扱い方法のみを扱うことにしているので，以降，コンピュータ上で扱う実数を要素として持つデータ型（実数型 (float, double)，複素数型 (complex)，ベクトル型，行列型）の要素は有限桁の小数として表現されるものとする。

長い桁数の小数を短い桁の小数に変換することを丸める (rounding, round-off) と呼ぶ。例えば π を有効数字 5 桁の浮動小数点数 $\widetilde{\pi}$ に丸める処理は，6 桁目の値を四捨五入して行うことが標準的である。

$$\widetilde{\pi} = 3.1416 \approx \pi$$

この場合，真値 (真の値) は π であり，近似値は $\widetilde{\pi}$ である。

ここで発生する真値と近似値のずれを誤差 (error) と呼ぶ。ここでは近似値 $\widetilde{a}$ に含まれる誤差を

$$E(a) = a - \widetilde{a}$$

と表現することにする。

通常，誤差は符号より大きさのみを調べるために用いられる。そこで誤差の絶対値

$$aE(\widetilde{a}) = |E(\widetilde{a})| = |a - \widetilde{a}|$$

を絶対誤差 (absolute error) と呼ぶ。また，近似値に含まれる誤差の割合を調べる際には，相対誤差 (relative error) と呼ばれる次の量 rE が使用される。

$$rE(\widetilde{a}) = \begin{cases} \frac{aE(a)}{|a|} = \left|\frac{a-\widetilde{a}}{a}\right| & (a \neq 0) \\ aE(a) = |a - \widetilde{a}| & (a = 0) \end{cases}$$

数学的にはこの定義で良いが，実際には真値 a が不明なことが多い。従って，その場合は $\widetilde{a}$ より真値に近い (と保証できる) 近似値 $\widetilde{a}'$ を用いて，絶対誤差も相対誤差も，その近似値

$$\begin{aligned} aE(\widetilde{a}) &\approx |\widetilde{a}' - \widetilde{a}| \\ rE(\widetilde{a}) &\approx \begin{cases} \left|\frac{\widetilde{a}'-\widetilde{a}}{\widetilde{a}'}\right| & (\widetilde{a}' \not\approx 0) \\ |\widetilde{a}' - \widetilde{a}| & (\widetilde{a}' \approx 0) \end{cases} \end{aligned}$$

で代用する。

この相対誤差 (あるいは相対誤差の近似値) を用いて，近似値の有効桁数 (有効数字, valid digits) を次のように定義できる。10 進数で表現される近似値 $\widetilde{a}$ の有効桁数は，相対誤差 $rE(\widetilde{a})$ を用いて

$$d = -\log_{10} rE(\widetilde{a})$$

と表現される。例えば，$\widetilde{\pi} = 3.14159$ の場合，真値 π の代わりに $\widetilde{\pi}' = 3.1415926535$ を用いると，絶対誤差，相対誤差，(10 進表現の) 有効桁数は

$$\begin{aligned} aE(\widetilde{\pi}) &\approx 0.0000026535 = 2.6535 \cdot 10^{-6} \\ aE(\widetilde{\pi}) &\approx aE(\widetilde{\pi})/\widetilde{\pi}' \approx 8.44635 \cdot 10^{-7} \\ d = -\log_{10} rE(\widetilde{\pi}) &\approx 6.073 \end{aligned}$$

となる。大体 10 進 6 桁ぴったり真値と一致する，と考えてよい。

2.4　MATLAB の基本演算機能

本章では図 2.2 に示す MATLAB R2017b の Windows 用 GUI 版に基づいて解説する。
GUI 版は図 2.2 に示すように次の 4 つのウィンドウから構成される。

1. ファイルブラウザ・・・MATLAB スクリプト (.m という拡張子を持つ MATLAB 命令から成るプログラム) 等を呼び出す。
2. コマンドウィンドウ・・・MATLAB 命令（コマンド）を直接打ち込むことができ，標準入出力はこのコンソール上で行われる。

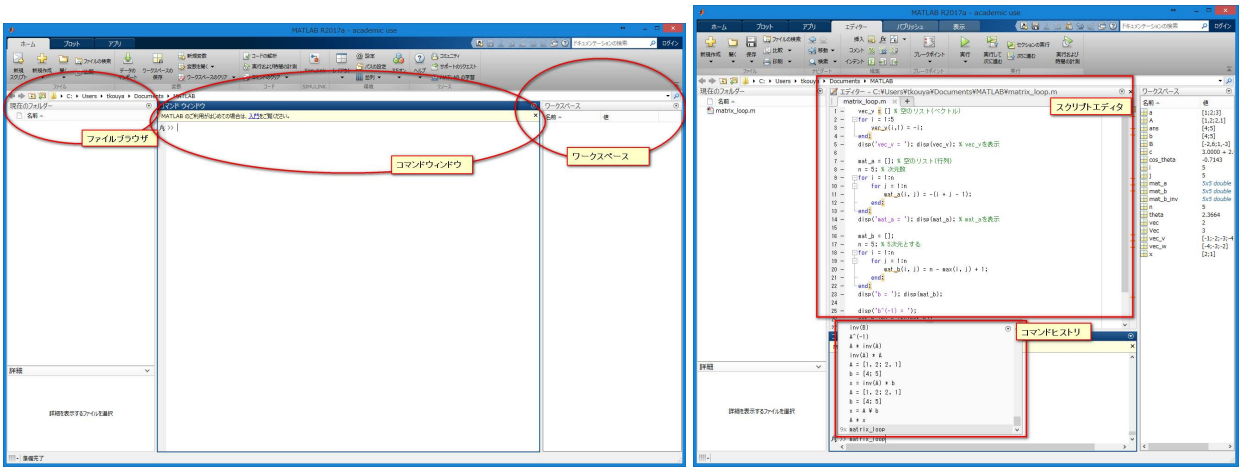

図 2.2　MATLAB の画面：Windows 用 GUI 版の場合 (左：初期画面，右：スクリプトエディタを表示した場合の画面)

3. ワークスペース・・・定義済みの MATLAB 変数一覧。
4. コマンドヒストリ（履歴）・・・コンソールに打ち込んだ MATLAB 命令を表示。特定の命令をダブルクリックすることで呼び出すことができる

以下，MATLAB の基本機能を見ていくことにする。使用方法としては，MATLAB を起動したのち，

- コンソールに直接コマンドを打ち込んで 1 行ずつ実行する
- MATLAB 付属のテキストエディタを起動し，実行したいコマンドを連ねたプログラム (MATLAB スクリプト) を作成してまとめて実行する

という二つのやり方がある。簡単な実行例はコンソールに直接打ち込み，長い MATLAB スクリプトはエディタで作成して実行，というように適宜使い分けるようにするとよい。ここではそのように使い分けを行う。

MATLAB が行う計算は，特に指定しない限り，全て IEEE754 倍精度 (8 Byte, 2 進 53bits, 10 進約 15 桁) の実数型で行われる。データを記憶するための変数 (variable) は，特に型宣言することなく，スクリプトの任意の位置で使用できる。大文字小文字の区別はあるので，`Vec` と `vec` は異なる変数として扱われる。

```
>> Vec = 3 ←ここを打ち込む
 Vec   = ← 変数 Vec に
         ←
     3   ← 3 が入っているという意味

>> vec = 2 ←ここを打ち込む
 vec   =
```

```
    2
```

一度使用した変数は clear 命令を使って消去しない限り記憶し続ける。

```
>> Vec          ← 変数 Vec に格納されているデータを表示
 Vec  =

       3

>> clear Vec ← 変数 Vec を消去

>> Vec
関数または変数 'Vec' が未定義です。 ← 変数 Vec が消えているのでエラーとなる

もしかして: ← 変数 Vec に似ている名前の変数があれば
>> vec        ← サジェスチョンをしてくれる
```

変数の値をコンソールに出力したくないときにはセミコロン; を使用する。

```
>> Vec = 3 ← 変数 Vec に 3 を格納し，コンソールに Vec の値を出力
 Vec  =

    3

>> Vec; ← Vec の値を表示しない

>> Vec  ← Vec の値を表示
 Vec  =

    3
```

変数に値が入力されたら，それを用いて演算ができる。四則演算は下記のように実行する。式に含まれる半角スペースはあってもなくてもよいが，見易さのため，必要に応じて入れる癖をつけておくとよい。

```
>> Vec + vec ← Vec と vec の和
 ans  =

    5

>> Vec - vec ← Vec と vec の差
 ans  =

    1

>> Vec * vec ← Vec と vec の積
 ans  =

    6

>> Vec / vec ← Vec と vec の商
 ans  =

    1.5000
```

MATLAB ではべき乗 (x^y) の機能も備わっている。例えば $\sqrt{2}$ は

```
>> 2^(1/2)
 ans  =

    1.4142

>> 2^(0.5)
 ans  =

    1.4142

>> sqrt(2) ← 平方根を計算する MATLAB 関数
 ans  =

    1.4142
```

と計算できる。
なお，値の表示方法は特に指定していない限り 10 桁と設定されているが，format 関数を使うことで倍精度一杯，16 ケタの表示が可能となる。

```
>> 2^(1/3) ← 2 の立方根を計算
 ans  =

    1.2599

>> format long ← 有効数字の表示桁数を長くする

>> 2^(1/3)  ← もう一度計算
 ans  =

    1.259921049894873  ←有効数字一杯の桁数まで表示
```

$2^{1/3} = 1.259921049894873\underline{16}47672106072...$ となるので，16 桁目まで正しく，17 桁目から誤差が混入していることが分かる。

値の表示形式を元に戻す時には format short と指定すればよい。

```
>> format short
>> 2^(1/3)

ans =

    1.2599
```

問題 2.1

1. 次の計算を MATLAB で行え。短い桁数と長い桁数の両方を出力し，その結果を書くこと。
   (a) 2^{10}
   (b) 3^{-10}
   (c) $1/2 + 1/3$
   (d) $10! = 1 \times 2 \times \cdots \times 10$
2. 自然対数の底 $e = 2.71828...$ を真値とする。その近似値 $\widetilde{e} = 2.7183$ が与えられた時，$\widetilde{e}$

の絶対誤差，相対誤差，10 進有効桁数を MATLAB で求めよ。

3. 真値 $a = \sqrt{2000}$ に対して，10 進 10 桁に丸めた値を $\widetilde{a}$ とする。この時，絶対誤差，相対誤差，10 進有効桁数を求めよ。(ヒント: `round(sqrt(2000), 10, 'significant')` として $\widetilde{a}$ を求めよ。)

2.5　入出力

他のプログラミング言語同様，MATLAB でも標準入力 (コンソールからキーボード入力) と標準出力 (コンソールへ出力) が可能である。以下，"sample_input.m"という名前の MATLAB スクリプト (11 行分) としてエディタから入力して適宜実行すること。なおスクリプト中のコメント文は %(パーセント記号) 以降に記入すること。

■標準入力　ユーザからの入力を待ち，入力値を変数に渡すには下記のように input 関数を用いる。**行番号を表わす「1: 」は入力しないこと！**

```
 1: % 入力
 2: a = input('a を入力 > '); % コンソールから入力
 3: b = input('b を入力 > ');
```

この結果，変数 a, b にそれぞれキーボードからの入力値が入るようになる。

■標準出力　上記のスクリプトの 3 行目にに続いて，入力された変数 a, b の値を標準出力に表示するためには下記のように disp 関数（形式指定なし），もしくは fprintf 関数（形式指定あり）を使用する。なお，改行を表わす\n の「\」(バックスラッシュ）は日本語環境では半角の円マーク「¥」として打つこと。

```
 4: (空行)
 5: % 出力
 6: disp('a = '); disp(a);        % 改行付き文字列として出力
 7: fprintf('b = %25.17e\n', b); % C の printf, fprintf 関数と同じ
```

上記までの部分をスクリプトエディタ打ち込み，メニューバーから「実行」をクリックするとスクリプトが実行される。a に 3 を，b に 4 を入力すると下記のような結果を得る。

```
a を入力 > 3 ← 3 をキーボードから入力
b を入力 > 4 ← 4 をキーボードから入力
a =
     3

b =   4.00000000000000000e+00
```

同様に，"a + b"，"a - b"の計算結果を表示させるには次のようにすればよい。

```
 8: (空行)
 9: % 計算と出力
10: disp('a + b = '); disp(a + b);
11: fprintf('a - b = %25.17e\n', a - b);
```

問題 2.2

c と d に値を入力し，$c*d$ と c/d の計算結果を返す MATLAB スクリプト (sample_input2.m) を作れ。また，このスクリプトを使って，c, d が下記の値になる時の計算結果を書け。

1. $c = 10, d = 30$
2. $c = 0, d = -15$
3. $c = 5, d = 0$
4. $c = 123456789, d = 987654321$

2.6 定数と複素数の定義・演算

MATLAB に限らず，プログラミング言語等でも，よく使用する定数はあらかじめ定義され，ユーザはそれを呼び出すだけで使えることが多い。MATLAB の場合，定数として `pi` を円周率 $\pi = 3.141592...$ として使用できる。

```
>> format long
>> pi

ans =

   3.141592653589793
```

自然対数の底 $e = 2.71828...$ は，`exp(1)`，即ち，指数関数 $\exp(x) = e^x$ を使用して得ることができる。

```
>> exp(1)

ans =

   2.718281828459046
```

複素数の定義に使用される虚数単位 `i` $= \sqrt{-1}$ も，定数 `i(j` も使用可) として定義されている。

```
>> i

ans =

  0.000000000000000 + 1.000000000000000i

>> i^2

ans =

    -1

>> j

ans =

  0.000000000000000 + 1.000000000000000i

>> j^2
```

```
ans =

    -1
```

これを用いて複素数の定義もできる。

```
>> format short
>> c = 3 + 2i

c =

   3.0000 + 2.0000i
```

なお，複素数定義関数 complex を用いて

```
>> c = complex(3, 2)

c =

   3.0000 + 2.0000i
```

としてもよい。

問題 2.3

$c = 2 + 3\mathrm{i}, \mathrm{d} = 2\mathrm{i}$ である時，次の計算を行え。

1. $\mathrm{ic} + (3 + 2\mathrm{i})\mathrm{d}$
2. $(\sqrt{2} + \sqrt{3}\,\mathrm{i})\mathrm{cd}$