第II部

対話型利用：電卓のように

第2章　MATLABの基礎

MATLABのインストールが終わり起動すると，いろいろ見慣れないものが出てきます．Webを経由する場合もほぼ同じだと思います．まず電卓並みの作業から始めてみましょう．

2.1　MATLABの起動と利用のスタート

2.1.1　MATLABの起動

インストールが完了すると，PCの画面上にMATLABのアイコンが現れます．これをクリックしてMATLABを起動すると，複数のウィンドウ，ブラウザーが開きます．各ウィンドウ，ブラウザーの形式は，[ホーム]→[設定]→[MATLAB]の中で変更できます．

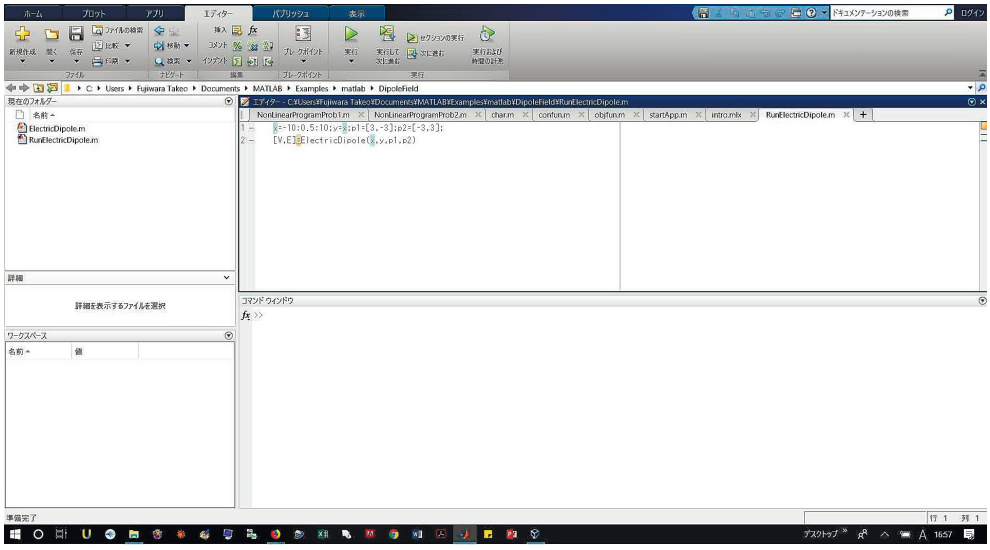

図 2.1: MATLABを起動すると，複数のMATLABウィンドウが開く．

- 「コマンド」ウィンドウ：ここでは対話型に，コマンドを入力して結果を得ることができる．
- 「エディター」: 自分が作成したファイルを編集することができる．新しい計算プログラムを作成，保存するためにもこのウィンドウを使う．
- 「ワークスペース」ブラウザー：現在使用している変数の一覧などが表示される．
- 「現在のフォルダー」ブラウザー：現在，どのフォルダーを使用しているかが表示され，またフォルダーをここで変更することもできる．
- 「コマンド履歴」ウィンドウ：コンソールから入力したコマンドが，入力の順にしたがって表示される．
- 「Figure」 ウィンドウ：図を描くと開く．

2.1.2　MATLAB の使用で迷ったとき

2.1.2.1　誰かに聞く

MATLAB を使用しているとき，どうやったらよいか，どのような関数が用意されているかなどに迷う場合には，色々なやり方で調べることができます．
(1) 製品付属のヘルプ文書を，(オンライン接続しない状態でも）ドキュメンテーション・キーワードで調べることができます．
(2) オンラインヘルプ
(3) オンラインマニュアル
(4) MATLAB ユーザーグループへの質問
(5) ネット検索
(6) 東京大学ユーザーグループへのメールによる問い合わせ
いずれが最も有効であるか，使いやすいかは各自で判断してください．
(2)∼(4) は，MathWorks Account が必要です．MATLAB マニュアルは以下にあります．
https://jp.mathworks.com/help/

(6) は私たち東京大学 MATLAB ユーザーグループが用意した質問サイトです．情報システム本部のサイト
https://www.u-tokyo.ac.jp/adm/dics/ja/matlabcwl.html
の中の WebForm を利用してください．東大ユーザーグループの中の誰かがお答えします.

2.1.2.2 マニュアルについて

MathWorks 社が提供している日本語マニュアルは，英語マニュアルの翻訳です．そのため分かり難いあるいは不正確なことがあるようです．日本語マニュアルが分かり難いと感じたときには，英語マニュアルにも当たるようにしてください.
マニュアルを全部読もうと考えるのは止めましょう．むしろ**一般の web 検索で,「MATLAB　****」と入れる**と効率的にマニュアルの箇所を探すことができます．****は MATLAB のキーワードか数学の術語です.
計算をするときには，マニュアルに書いてあることをウのみにせず，検算をやって確かめてください.

2.2 変数と簡単な計算

以下のステートメントをコマンドウィンドウに表示されているコマンドプロンプト >> の後ろに入力し実行（Return キー) すれば，結果が表示されます。

(a) 変数　*a, b* と　*a+b*　(b) 行列と簡単な行列演算

```
>> a=1
a =
     1

>> b=2.5
b =
    2.5000
>> a+b
ans =
    3.5000
```

```
>> A=[16 3+i 2 13;5 10-i 11 8;9 6 7+2i 12;4 15 14-4i 1]
A =
  16.0000 + 0.0000i   3.0000 + 1.0000i   2.0000 + 0.0000i  13.0000 + 0.0000i
   5.0000 + 0.0000i  10.0000 - 1.0000i  11.0000 + 0.0000i   8.0000 + 0.0000i
   9.0000 + 0.0000i   6.0000 + 0.0000i   7.0000 + 2.0000i  12.0000 + 0.0000i
   4.0000 + 0.0000i  15.0000 + 0.0000i  14.0000 - 4.0000i   1.0000 + 0.0000i
>> inv(A)
ans =
   1.4762 + 1.8175i   3.6349 + 5.6587i  -3.9048 - 5.5952i  -1.4127 - 1.7540i
  -3.2381 + 0.8095i  -9.7143 + 1.8095i   9.7143 - 2.0000i   3.2381 - 1.0000i
   3.2381 - 0.3095i   9.7143 - 0.3095i  -9.7143 + 0.5000i  -3.2381 + 0.5000i
  -1.4286 - 2.1270i  -3.5873 - 6.5873i   3.9048 + 6.5238i   1.4127 + 2.0635i
```

```
>> A'
ans =
  16.0000 + 0.0000i   5.0000 + 0.0000i   9.0000 + 0.0000i   4.0000 + 0.0000i
   3.0000 - 1.0000i  10.0000 + 1.0000i   6.0000 + 0.0000i  15.0000 + 0.0000i
   2.0000 + 0.0000i  11.0000 + 0.0000i   7.0000 - 2.0000i  14.0000 + 4.0000i
  13.0000 + 0.0000i   8.0000 + 0.0000i  12.0000 + 0.0000i   1.0000 + 0.0000i
>> A.'
ans =
  16.0000 + 0.0000i   5.0000 + 0.0000i   9.0000 + 0.0000i   4.0000 + 0.0000i
   3.0000 + 1.0000i  10.0000 - 1.0000i   6.0000 + 0.0000i  15.0000 + 0.0000i
   2.0000 + 0.0000i  11.0000 + 0.0000i   7.0000 + 2.0000i  14.0000 - 4.0000i
  13.0000 + 0.0000i   8.0000 + 0.0000i  12.0000 + 0.0000i   1.0000 + 0.0000i
```

図 2.2: (a) 変数 a, b と $a+b$．(b) 行列の定義と簡単な演算，転置行列.

2.2.1　変数と加減乗除，べき乗，初等数学関数

（図 2.2a)

- >> がコマンドライン，その後ろが結果．変数 a, b の値を設定.
- $a + b$ と打ち込めば，$a + b$ の値が返される．
- その値を例えば c としたければ，$c = a + b$ と打ち込めばよい.
- 乗除算はそれぞれ，$a * b$，a/b.
- x^n は x^n と書く．
- 様々な初等関数は既に定義されている：$\sin(a)$ など.
- 複素数 $x + \mathrm{i}y$ は $x + yi$ あるいは $0.1 + 0.6i$ 等と書く．i の代わりに j と書いてもよい.

演算子の一覧表は
https://jp.mathworks.com/help/matlab/matlab_prog/matlab-operators-and-special-characters.html

2.3　行列と行列演算

2.2.1　ベクトルおよび行列

（図 2.2b)

- 行列の定義：各要素はコンマ (,) またはスペース（空白）で区切る．
- セミコロン (;) で区切ると，次の行に移ることを示す.
- 行列を転置するにはドットダッシュ (.') を付ける．これは**数学の記号（上付きの t または T）**とは異なる．
- 行列のエルミート共役をとるにはダッシュ (') を付ける．実行列ではドットダッシュ (.') と同じ．これは**数学の記号（上付きの*）**とは異なる．
- 行ベクトルは 1 行 n 列の行列として定義.

- 列ベクトルは n 行 1 列の行列として定義するか，あるいは行ベクトルを転置する．
- A の逆行列は $\mathrm{inv}(A)$.

2.3.2 行列演算

MATLAB は行列が重要な役割を果たします．そのため通常の線形代数における演算規則と異なる MATLAB 特有な計算もあります．その他の言語，例えば Python，でも利用されます．

```
>> A=[1 2 3;4 5 6;7 8 9]
A =
   1   2   3
   4   5   6
   7   8   9
>> A+10
ans =
   11   12   13
   14   15   16
   17   18   19
>> sin(A)
ans =
   0.8415   0.9093   0.1411
  -0.7568  -0.9589  -0.2794
   0.6570   0.9894   0.4121
```

```
>> B=[sin(1) sin(2) sin(3);
      sin(4) sin(5) sin(6);
      sin(7) sin(8) sin(9)]
B =
   0.8415   0.9093   0.1411
  -0.7568  -0.9589  -0.2794
   0.6570   0.9894   0.4121
>> B==sin(A)
ans =
  3×3 の logical 配列
   1   1   1
   1   1   1
   1   1   1
```

```
>> A*B
ans =
   1.2988   1.9595   0.8186
   3.5238   4.7787   1.6401
   5.7488   7.5979   2.4616
>> A.*B
ans =
   0.8415   1.8186   0.4234
  -3.0272  -4.7946  -1.6765
   4.5989   7.9149   3.7091
```

図 2.3: (左) 行列 A の定義，$A+10$, $\sin(A)$. (中央）$B=\sin(A_{ij})$ および B が $\sin(A)$ となっていることの確認．（右）$A*B$, $A.*B$

行列 A を定義します（図 2.3）.

- 行列 A の i,j 要素は $A(i,j)$. $A_{ij}=A(i,j)$
- $A+10$ は，A の各要素に 10 が足されます．$(A+10)_{ij}=A_{ij}+10$. **これは数学での規則とは異なります**.
- $\sin(A)$ の各要素は
$$(\sin(A))_{ij}=\sin(A_{ij})$$
となります．**通常，数学での定義は 2.3.3 です**.

- A, B がそれぞれ同じ大きさの行列であるとき，$A * B$ は通常の行列の掛け算

$$(A * B)_{ij} = \sum_k A_{ik} B_{kj}$$

です．
しかし，MATLAB 特有の演算 $A. * B$ は要素ごとの掛け算，

$$(A. * B)_{ij} = A_{ij} B_{ij}$$

です．これを特に**アダマール積** (Hadamard product) または**シュール積**といいます．

- .^　および　./　も同様な要素ごとの演算です．これを**アダマール冪** (Hadamard power)，**アダマール除算** (Hadamard division) と呼びます．

- 行列 A, B に対して $a = [A, B]$ あるいは $a = [A\ B]$（A と B の間にはスペース）は，行列 A と B を連結させ，A と B を横に並べます．

- $b = [A; B]$ は，行列 A と B を連結させ，A と B を縦に並べます．

(図 2.3)

MATLAB で A.^ (-1) と A^ (-1) は違うものです．また 1/A と入れると次のように返ってきます．

エラー: /
行列の次元は一致しなければなりません。

これらは上に示したことから直ぐに理解できるはずです．試みに A として例えば 2×2 の行列を入れていろいろ試してみて下さい．

アダマール積その他の演算は注意しなくてはなりませんが，慣れると便利です．また途中で使った「$A == B$」は，A と B は等しいか，という**関係演算子**といわれるものです．等しい要素の所は 1，異なる要素の所は 0 を返します．後で出てきますが,「$f(x) == 0$」は「恒等的に $(x) = 0$ という関係が成り立つ」という意味になります．

2.3.3 行列関数について

行列 A が関係する計算については，MATLAB では要素ごとの計算をします．しかし数学（線形代数）では次のように定義します．

$$A^n = \overbrace{A * A * \cdots * A}^{n\text{ 個の }A} \tag{2.1a}$$

$$\mathrm{e}^{\mathrm{A}} = E + A + \frac{1}{2!}A^2 + \cdots + \frac{1}{n!}A^n + \cdots \tag{2.1b}$$

$$\sin A = A + \frac{1}{3!}A^3 + \cdots + \frac{1}{(2n+1)!}A^{2n+1} + \cdots \tag{2.1c}$$

$$\cos A = E + \frac{1}{2!}A^2 + \cdots + \frac{1}{(2n)!}A^{2n} + \cdots \tag{2.1d}$$

これらを行列関数と呼び，MATLAB では別に定義をします．これについてはもう少し後で説明します．

第3章　線形代数：初級編

3.1　連立1次方程式

3.1.1　逆行列の計算

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{cases}$$

という連立1次方程式は，行列とベクトルを用いて次のように書けます.

$$A\boldsymbol{x} = \boldsymbol{b}$$

ここで A は $n \times n$ 行列, $\boldsymbol{x}$, $\boldsymbol{b}$ は n 次元縦ベクトルです．これに左から逆

```
>> A = fix(10*rand(3,3))
A =
    7    6    0
    7    1    2
    3    7    0
>> rank(A)
ans =
    3
>> B=eig(A)
B =
   11.7274
    1.0960
   -4.8235
```

```
>> A00=det(A)
A00 =
  -62.0000

>> C=inv(A)
C =
    0.2258   -0.0000   -0.1935
   -0.0968   -0.0000    0.2258
   -0.7419    0.5000    0.5645
```

```
>>  D=A*C;
>>  E=[1 0 0;0 1 0;0 0 1];
>> D==E
ans =

  3×3 の logical 配列

   1   0   1
   1   1   1
   1   0   1
```

図 3.1: 逆行列の計算．(左) 行列 A の定義，階数，固有値. (中央）行列 A の行列式および $C = A^{-1}$. （右）$AC =$ 単位行列 になることの確認.

行列 A^{-1} をかけて次式を得ます.

$$\boldsymbol{x} = A^{-1}\boldsymbol{b}.$$

ただし行列が大きくなった場合の実際の計算では，A の逆行列を求め，それを $\boldsymbol{b}$ に掛けるという方法はお勧めできません．その理由は，逆行列を直接計算するという方法は計算時間の面からもまた精度の面からいっても，なかなか問題があるからです．この問題は後で触れることにしましょう.

図 3.1に行列計算のいくつかを示します.

- rand は $(0, 1)$ 区間の一様乱数.
- $X = \mathrm{rand}(n_1, \cdots, n_N)$ は，$(n_1 \times \cdots \times n_N)$ 個の乱数が格納された N 次元の $n_1 \times \cdots \times n_N$ 配列を返す.
- fix は 0 方向に少数以下を丸める関数（$\mathrm{fix}(1.9) = 1$, $\mathrm{fix}(-0.9) = 0$).
- $\mathrm{rank}(A)$ は行列の階数（ランク）を返す.
- コマンドの後に ;（セミコロン）を入力すると，実行結果は印字されない.

最後に $D == E$ としたとき，3×3 の総ての成分が 1 であることを期待しました．しかしいくつかの成分は 0（等しくない）という答えを返しました．何故でしょうか.（もし分からなければ $D - E$ と入れてみてください．）二つの実数の比較には注意が必要です.

3.1.2 連立 1 次方程式の MATLAB における適切な解法

$$A\boldsymbol{x} = \boldsymbol{b}$$

の解法は（$\boldsymbol{x} = A^{-1}\boldsymbol{b}$ ではなく）

$$\mathrm{x} = \mathrm{A}\backslash\mathrm{b}$$

です．ここではどのようなアルゴリズムを用いるか説明しません（第 11章を参照）が，行列の扱いは，精度と計算時間が最も典型的に計算手順に依存します．ですから，逆行列を計算することはせずに，上のように書くことを強くお勧めします．これについては，第 11章で，再び触れることにします.

3.2　固有値および固有ベクトル

$n \times n$ 行列 A が与えられたとき次式が得られたとしましょう．これは n この変数（$x_1, x_2, \cdots, x_n$）を持った n 元の連立 1 次方程式です．

$$(A - \lambda E)\boldsymbol{x} = 0\ . \tag{3.1}$$

λ は数，E は $n \times n$ 単位行列，$\boldsymbol{x}$ は縦行列です：$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T$．これから λ （**固有値**）とそれに対応した $\boldsymbol{x}$（**固有ベクトル**）を求めようというのが問題です．
このとき自明でない解 $\boldsymbol{x} \neq 0$ が存在するためには，$(A - \lambda E)$ が逆行列を持たないことが必要充分です．従って必要十分条件は

$$\det(A - \lambda E) = 0 \tag{3.2}$$

です．λ を固有値，$\boldsymbol{x}$ を固有ベクトルといいます（図 3.2）．

```
>> A = fix(10*rand(3,3))
A =
    9    7    0
    2    3    0
    7    5    5
>> rank(A)
ans =
    3
>> eig(A)
ans =
   5.0000
   1.2042
  10.7958
```

```
>> [V D]=eig(A)
V =
         0    0.6479   -0.5672
         0   -0.7215   -0.1455
    1.0000   -0.2443   -0.8106
D =
    5.0000         0         0
         0    1.2042         0
         0         0   10.7958
```

図 3.2: (左) 行列 A を一様乱数を用いて定義，その階数と固有値．（右）行列 A の固有値を収めた対角行列 D と，固有ベクトル（縦ベクトル）を並べて作った行列 V．

- $\mathrm{rank}(A)$ は行列 A の階数（rank）を返す．
- $e = \mathrm{eig}(A)$ は，正方行列 A の固有値を要素とする列ベクトルを返す．
- $[V, D] = \mathrm{eig}(A)$ は，固有値からなる対角行列 D と，対応する右固有ベクトルを列にもつ行列 V (つまり $A * V = V * D$) を返す．

第4章　シンボリックな計算

数学で必要なことは，数値計算だけではありません．通常の計算機上の演算は数値に対して行われます．しかしシンボリック演算では数式に対して数式の記号のままの演算が行われます．
MATLAB は**シンボリック演算（数式処理）**の機能も備えています（Symbolic Math Toolbox をインストールすることが必要です）．シンボリック演算をするためには，変数や行列，関数がシンボリックなもの（記号のままで処理されるべきもの）であることを宣言する必要があります．

4.1　シンボリック変数とシンボリック演算

4.1.1　シンボリック変数の定義とシンボリックな処理

シンボリック演算の例を図 4.1に示しましょう．

```
>> s=sym('s')
>> expand((s+1)^3)
ans =
s^3 + 3*s^2 + 3*s + 1
```

```
>> syms y z
>> expand((y+1)^3)
ans =
y^3 + 3*y^2 + 3*y + 1
>> simplify(z^3+3*z^2+3*z+1)
ans =
(z + 1)^3
```

図 4.1: シンボリック変数の定義と計算.

- x をシンボリック変数として定義するためには syms x あるいは $x = \mathrm{sym}('x')$ である．
- expand は式の展開.
- simplify は因数分解など（代数的な単純化）を行う.

行列をシンボリック行列として定義するときは次のようにします.

$$\mathrm{A} = \mathrm{sym}('\mathrm{A}', [2\ 4])$$

これで 2×4 行列 A が定義され，A の i, j 成分は Ai_j という名前になります．sym は行インデックスと列インデックスをアンダースコア underscore（_）で区切りますが，これを変えることもできます.

4.1.2　シンボリックな求解

シンボリックな演算によって解を求めることも可能です．ここでは代数方程式の解の一般的な式を求めることと，三角関数の式の解を求める方法を示してみましょう（図 4.2).

```
>> syms a b c x
>> eqn=a*x^2+b*x+c==0;
>> sol=solve(eqn)
sol =
 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
>> sol2=solve(eqn,x)
sol2 =
 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

```
>> syms x
>> eqn=sin(x)==1
eqn =
sin(x) == 1
>> sol3=solve(eqn,x)
Sol3 =
pi/2
```

図 4.2: シンボリックな求解.

- eqn= $\cdots$ == $\cdots$ は恒等式を定義し，それを 'eqn' と名付ける.
- solve(eqn) は恒等式 'eqn' を解く．複数の変数がある場合，変数は 'symvar' によって検索され、アルファベット順に並べて返されるとマニュアルにはある．[1] 複数のシンボリック変数がある場合には，何について解くかを，symvar で書かせて確認するか，あるいは明示的に指定するか，いずれかを勧める.
- solve(eqn,x) は恒等式 eqn を変数 x について解くというコマンドである.

[1] しかし実際にそのルールどおりになっているのかはよく理解できません.

4.2 シンボリック関数演算

シンボリックな表現を用いれば，微積分その他の計算が，数式として行えます.

4.2.1 微分

シンボリックな変数あるいは関数を用いれば，解析的な微分を行うことができます．そのときは演算子

$$\mathrm{diff}(f),\ \mathrm{diff}(f,x)$$

を用います．f は関数です．n 次導関数は

$$\mathrm{diff}(f,n)$$

です.
さらに関数 $f(x)$ について

$$\mathrm{eq2}{=}\mathrm{diff}(f,2)$$

として，2 階微分を求めた後で，$\mathrm{eq2}(x_0)$ を求めれば，$x = x_0$ における 2 回微分係数の値が与えられます.（図 4.3）

```
>> syms x n
>> f(x,n)=x^n
f(x, n) =
x^n
>> df1(x,n)=diff(f,x)
df1(x, n) =
n*x^(n - 1)
>> df1(2,n)
ans =
2^(n - 1)*n
>> df2(x,n)=diff(f,2)
df2(x, n) =
n*x^(n - 2)*(n - 1)
```

```
>> syms x n
>> g(x,n)=n*x^(n-1)
g(x, n) =
n*x^(n - 1)

>> intg(x,n)=int(g,x)
intg(x, n) =
piecewise(n == 0, 0, n ~= 0, x^n)

>> int(g,[1,2])
ans(n) =
piecewise(n == 0, 0, n ~= 0, 2^n - 1)
```

図 4.3: シンボリックな微積分.

4.2.2　偏微分

偏微分に関しては，特に特別なことはありません．

4.2.2.1　偏微分

例えば2変数 x, y の関数 $f(x, y)$ を

```
syms x y
f=x^ 2+x*y+3*y^ 3
```

と定義します．x による（偏）微分は

```
diff(f,x)
```

y による（偏）微分は

```
diff(f,y)
```

です．高階（偏）微分は

```
diff(f,x1,x2, •••• ,xn)
```

と書き，これは f を x1, x2, •••• , xn の順番で微分します．

【問題】

$$f(x, y) = \frac{xy(x^2 - y^2)}{x^2 + y^2}$$

の1階偏微分係数，2階偏微分係数などを計算し，手計算の結果と比べてみてください．

4.2.2.2　多変数関数の勾配

多変数関数 $f(x_1, x_2, \cdots, x_n)$ に関する偏微分（勾配）に関しては，gradient というコマンドがあります．

```
syms x1 x2 ··· xn
[fx1, fx2, ··· , fxn] = gradient(f)
```

f(x) は n 次元配列です．

4.2.2.3 ヤコビアン（ヤコビ行列式）

一般の変数変換を考えてみます.

$$x = x(u, v)$$
$$y = y(u, v)$$

変換により x, y に関する積分を u, v についての積分に変換することを考えましょう．これは一般に

$$\int f(x, y)dxdy = \int f(x(u, v), y(x, v))|J|dudv$$

と書けます．$|J|$ はヤコビ行列式 J の絶対値であり，二つの空間 (x, y) と (u, v) の面素の比です：

$$J = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{pmatrix} = \frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u}.$$

例えば次のように入力します：

```
syms u v
x = u.^ 2 + v.^ 2
y = 2 * x. * y
jacobian([x, y], [u, v])
```

4.2.3 テイラー級数展開

微分学における重要な結果の 1 つは，テイラー級数展開です．MATLAB でテイラー級数展開を行うのは簡単です:

```
>> syms x
>> f=log(1+x)
>> taylor(f,'Order',8)
ans =
x^ 7/7 - x^ 6/6 + x^ 5/5 - x^ 4/4 + x^ 3/3 - x^ 2/2 + x
```

また，テイラー展開を任意の点の周りで行うことも可能です：

```
>> syms x
>> taylor(log(1+x),x,'ExpansionPoint',0)
ans =
x^ 5/5 - x^ 4/4 + x^ 3/3 - x^ 2/2 + x
>> taylor(log(1+x),x,'ExpansionPoint',1)
ans =
x/2 + log(2) - (x - 1)^ 2/8 + (x - 1)^ 3/24 - (x - 1)^ 4/64
+ (x - 1)^ 5/160 - 1/2
```

【問題】対数関数 log(1+x) を x=0 の他，幾つかの点例えば x=1/2 の周りでテイラー展開してみてください．さらにそれらを図に描くことで，収束の様子（あるいは元の関数値との比較）を調べてみましょう．

上のように**式または関数**を図に描くときは，

```
fplot(f,[x1 x2])
```

を使います．(plot(x,y) は点 (x, y) を線分で結ぶコマンドです.）[x1 x2] はグラフを描く変数の値の範囲を示します．これを省略するとデフォルトの範囲 [-5 5] で描画します．

4.2.4　積分

数式処理による積分（不定積分）は

```
int(expr,var)
```

です．シンボリックなスカラー変数 var について 関数 expr の不定積分を計算します．
さらにこれの定積分の値を求めたいときは，例えば積分区間を $[1.0, 2.0]$ とするならば

```
int(expr, [1,2])
```

とします．(図 4.3)
~= は不等号 $\neq$ です．

2 重積分および 3 重積分用のコマンド

```
integral2, integral3
```

等も用意されています．一般には，積分の方がシンボリック演算のプログラムを用意するのは難しいようです．

第5章　グラフ

5.1　MATLABの描画機能

計算結果をグラフに描くことができれば，私たちの理解が進みます．ですから計算の結果をグラフにしてみることを勧めます．
色々な場合を図にするのは，手作業で実行するのは難しいものです．コンピュータプログラムでこれが実行できるようになって，作業が著しく簡単になりました．しかし，同時にグラフの意味するところを時間をかけて吟味することが疎かになったというのも事実です．昔は手間をかけてグラフ描きながら，いろいろなことを考えていたということなのでしょう．空いた時間を考えることに使うよう，心してかからなくてはなりません．

5.1.1　グラフの種類

プロットルーティンには x 軸，y 軸を共に**線形**とするものの他，**片対数**プロットである semilogx, semilogy，**両対数**プロットである loglog があります．
また線の**太さと種類，色**，点に**マーカ**ー付与，などができます．さらに軸に**軸ラベル**を描くことや，**図のタイトル**なども書くことができます．

プロットは目的により様々なものがあります．

- 等高線図,
- 局面
- 3 次元空間のベクトル図（矢印入り）

などなどです．ここではそれらを例示しませんが，インターネットで ‘MATLAB プロットのタイプ’ と検索してみてください．下のようなサイトが見つかるでしょう．

https://jp.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html
この中から必要なタイプのものを選ぶと良いでしょう．

5.1.2　2 次元プロット

最も一般的な 2 次元プロットについて述べてみましょう．これは独立変数 x に対する関数 $f(x)$ の振る舞いを図にします（図 5.1）．

```
>> x=[0:0.1:2*pi];
>> y=[sin(x);cos(x)];
>> plot(x,y)
```

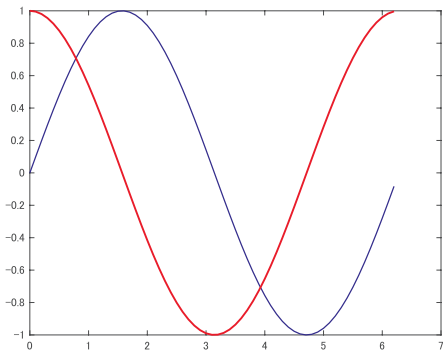

図 5.1: 2 次元のグラフプロット．コマンドと $\sin(x)$, $\cos(x)$ の図．

- ここで変数 x はベクトルとして定義．[0:0.1:2*pi] は，0 から 2π まで 0.1 刻みで与えるということ．x がベクトルであるから $\sin(x)$, $\cos(x)$ はベクトルとして定義される．このように定義されることは，行列のところで説明した．**ここが MATLAB 特有の関数定義の妙味の 1 つである．**
- コマンドの最後にセミコロン（；）を入力すると，計算結果は出力されない．
- 最後の描画コマンド $\mathrm{plot}(x, y)$ は 2 次元の点 (x_i, y_i) を直線でつなぐ，2 次元プロットのためのコマンド．これにより，横軸に x，縦軸に y をとった図が，Figure ウィンドウに出力される．
- clf コマンドで Figure ウィンドウはクリアされる．

5.1.3 陰関数プロット

ここで便利な機能を示しておきましょう.

```
>> f=@(x,y) x.^2+x.*y+2*y.^2-1
>> fimplicit(f,[-1.5 1.5 -2 2])
```

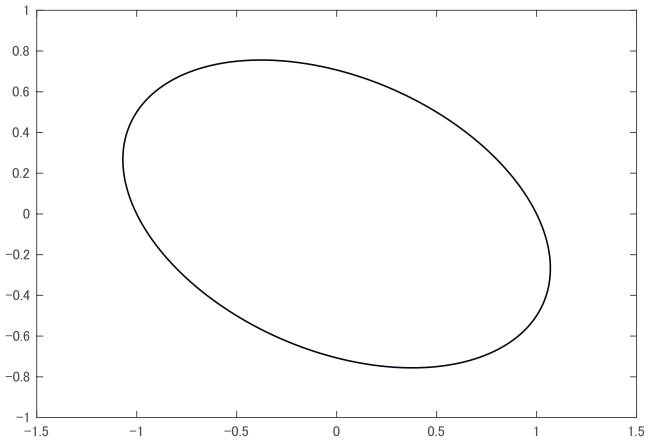

図 5.2: $f(x,y) = x^2 + xy + 2y^2 - 1 = 0$ の図.

図 5.2では，方程式 $f(x,y) = 0$ により定義された x の関数 $y(x)$ をプロットしています．このような機能を用いて図が得られれば，関数の性質を理解する大きな助けになるでしょう.

f = @(x,y) x.^ 2 + x.*y+2*y.^ 2 - 1

で（x,y）を変数とした関数 f （無名関数，anonymous function）が定義されます.

fimplicit(f,[-1.5 1.5 -1.0 1.0])

により，方程式 $f(x,y) = 0$ が決める曲線 (x,y) を，領域 $-1.5 \leq x \leq 1.5$, $-1.0 \leq y \leq 1.0$ で描きます.

5.1.4 3 次元プロット

3 次元空間で，上昇とともに半径が拡大する渦を描いてみましょう（図 5.3）.

plot3(x,y,z) は 3 次元座標データ (x_i, y_i, z_i) を直線でつなぎます．プロットされたデータは，**マウスの操作によりズーム，移動，回転など**ができます．3 次元グラフを様々な方向から見るなどしてより良い理解を得ることができます.

```
>> z = 0:pi/50:10*pi;
>> sz = z.*sin(z);
>> cz = z.*cos(z);
>> plot3(sz,cz,z)
```

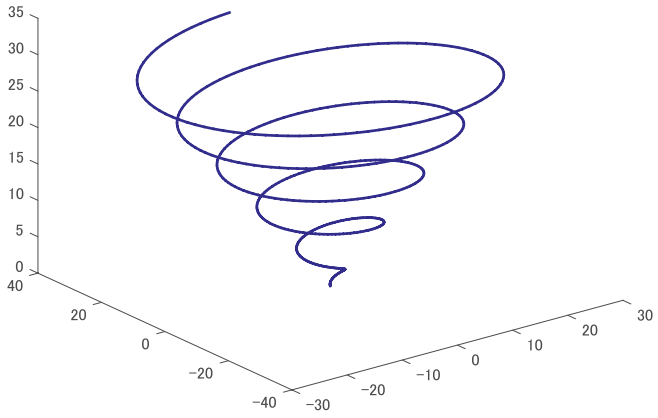

図 5.3: 3 次元プロット．コマンドと $x*\sin(z)$, $z*\cos(z)$ での図.

5.2　グラフを描く意味

関数の性質を知るために，その関数を可視化してみることは大いに意味があります.
数学的命題を考える際，証明という方法が重要ですし, 必要です．しかし，証明という段階が無くても，図に描けば十分であるというわけではありません．証明の前に，可視化という手順で性質を把握するのは，実際的で，重要な指針を得るという点からも必要です.

5.2.1　関数の極限，収束性の振る舞いを見る

対数関数

$$y = \log x$$

は

$$x = \exp y$$

の逆関数として定義されます.

$$\lim_{x\to 0+} \log x = -\infty \ , \quad \lim_{x\to\infty} \log x = +\infty$$

はよく知られた性質です．それでは $\alpha,\ \beta > 0$ としたとき

$$\lim_{x\to 0+} x^{\alpha} \log x \ , \quad \lim_{x\to\infty} x^{-\beta} \log x$$

はどうなるでしょう．極限値を持つのでしょうか.

最初の $\lim_{x\to 0+} x^\alpha \log x$ については $t = x^\alpha$ と置くと

$$x^\alpha \log x = \frac{1}{\alpha} t \log t$$

です（$x \sim 0+$ では $t \sim 0+$）．同様に $t = x^{-\beta}$ と置けば

$$x^{-\beta} \log x = -\frac{1}{\beta} t \log t$$

です（$x \sim +\infty$ では $t \sim 0+$）．上から，$\lim_{x\to 0+} x^\alpha \log x$ でも，あるいは $\lim_{x\to\infty} x^{-\beta} \log x$ の場合でも，$t \to 0+$ での $t \log t$ の振る舞いが問題であることが分かります（図 5.4）．

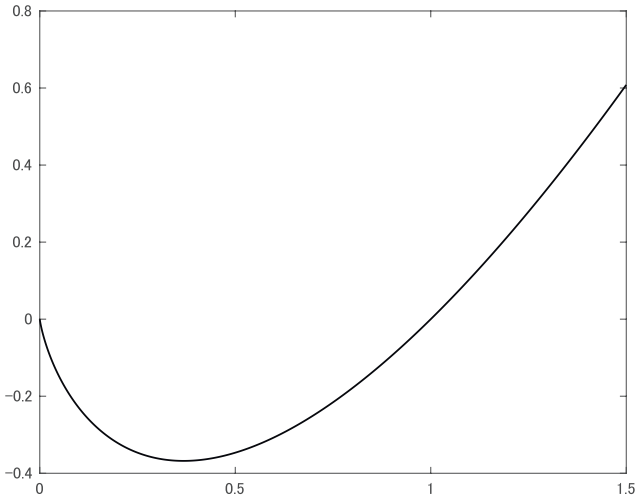

図 5.4: $t \log t$.

$u = \log t$ と置けば $t = \mathrm{e}^u$ ですから

$$t \log t = u\mathrm{e}^u$$

です．$x \to 0+$, $(t \to 0+)$ および $x \to \infty$, $(t \to 0+)$ のいずれでも $u \to -\infty$ であり，上式の e^u が（u に比べて）遥かに速く 0 となるので

$$\lim_{u\to -\infty} u\mathrm{e}^u = 0$$

です．こうして $\alpha,\ \beta > 0$ としたとき

$$\lim_{x\to 0+} x^\alpha \log x = 0\ , \quad \lim_{x\to\infty} x^{-\beta} \log x = 0$$

という極限値を持ちます．つまり $|\log x|$ は，$x \to 0+$ で x のどんな負冪よりゆっくりと無限大になり，$x \to +\infty$ で x のどんな正冪よりゆっくりと無限大になるのです．
一般には上のようにして式で示すのが一般的ですが，その前にここでやったように図に描いてみると，良く分かるのではないでしょうか．

5.2.2　関数の大局的な振る舞いを見る

きちんと証明するのは難しくても，関数の性質を大掴みに理解しようという際にも図は大いに役立ちます．ワイエルシュトラス関数（Weierstrass）

$$f(x) = \sum_{n=0}^{\infty} a^n \cos(b^n x) \ , \quad (0 < a < 1, b \text{ は奇数}, ab > 1 + \frac{3\pi}{2})$$

を考えてみましょう．これは「至る所で連続であるが至る所で微分不可能な関数」として知られています．
この級数が「至る所で連続である」ことは，$|a^n \cos(b^n x)| \leq |a|^n$, $|a| < 1$ であることからこの無限級数が収束することが言え，したがって一様連続であることはすぐに分かります．
「至る所で微分不可能な関数」であることを証明するにはもう少し手間がかかりますが，図を描いてみれば納得できると思います（図 5.5）．

```
a=0.5;
b=13.0;
t=[-pi/2*1:pi/10000.0:pi/2*1];
  x1=cos(t);
  x2=x1+cos(13*t);
  x3=x2+cos(13^2*t)
  plot(t,x1);grid on
  hold on
  plot(t,x2);grid on
  hold on
  plot(t,x3);grid on
  hold on
x=zeros(1,numel(t));
  for  n = 0:1:100
     x=x+ a^n*cos(b^n*t);
  end
  plot(t,x)
```

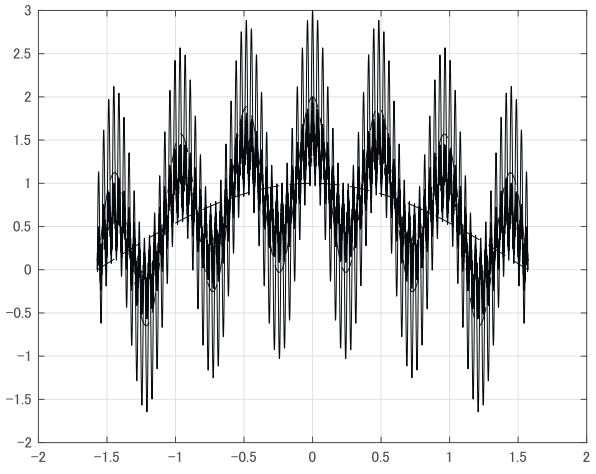

図 5.5: ワイエルシュトラス関数 ($a = 0.5,\ b = 13$).