# MATLAB 講習会 (基礎) 資料

この資料は，サイバネットシステム株式会社の Web Course のページ

http://www.cybernet.co.jp/matlab/support/webcourse/

を，MATLAB 講習会のために許諾を得て一部改変して利用させていただいているものです．できるだけ現在の学術情報メディアセンターのバージョンにあわせて書いているつもりですが，お気づきの点等ございましたら，furutani@kuee.kyoto-u.ac.jp までお知らせください．

# 第 1 回 MATLAB 基本操作

## 1.1 MATLAB の起動と終了

MATLAB は、基本的には Windows 環境（Microsoft Windows、Macintosh、X Windows System 等）で実行されます。

MATLAB の起動方法は、各システムによって異なります。MS-Windows 版、及び Macintosh 版では、MATLAB のアイコンをダブルクリックし、X Windows システムからの UNIX 版ではシェルプロンプト上で matlab とタイプして起動します。なお、Windows 上で使用する際は、スタートボタンをクリックし、MATLAB for Windows から MATLAB を選択して起動します。また、ショートカットを作っておくと、アイコンをダブルクリックすることによって起動することができます。

下記は、MATLAB が起動された時の共通の画面です。MATLAB 起動時には、一瞬 MATLAB のロゴウィンドウがフラッシュします。また、>> が MATLAB の起動された時のプロンプトです。以後、MATLAB プロンプトと呼びます。

```
                    < M A T L A B >
        Copyright 1984-2003 The MathWorks, Inc.
    Version 6.5.1.199709 Release 13 (Service Pack 1)
                      Aug  4 2003


  To get started, select "MATLAB Help" from the Help menu.
>>
```

MATLAB を終了するには、MATLAB プロンプトから quit コマンド、または exit コマンドを実行します。

Windows 版や Macintosh 版では、ウィンドウシステムのメニューバーにある『File』メニュー内から終了することもできます。

## 1.2 データの定義

MATLAB でのデータの取り扱いや適応範囲は、非常に柔軟です。MATLAB 上で扱う変数には、以下のような特徴があります。

- スカラやベクトルデータを含めて、データはすべて行列として取り扱われます。
- 変数の型宣言 (実数、複素数、文字)、および配列の宣言は不要です。
- データは、全て倍精度の浮動小数点型で取り扱われます。

MATLAB でのデータの入力 (定義) には、以下のような方法があります。

1．直接数値を入力する方法
2．MATLAB の関数を利用する方法
3．ファイルから読み込む方法

今回は、1 と 2 の方法について学びます。3 の方法については、「第 3 回 データの入出力」で詳しく取り上げます。

1． 直接数値を入力する方法

MATLAB のプロンプトに対して、直接キーボードから入力する方法です。変数を入力する場合には、以下のような約束があります。

- 行列の要素間は、ブランク (空文字) またはカンマ (,) で区切ります。
- 行列の各行の終了は、改行、またはセミコロン (;) で定義します。
- 要素全体を鍵括弧 ([ ]) で囲んで定義します。
- 1 行で記述しきれない場合は、MATLAB の継続記号として、最後にピリオド 3 つ以上 (...) を連続して付けてつぎの行に継続します。
- 計算結果を画面 (MATLAB コマンドウィンドウ) 上に表示させない場合は、ステートメントの最後にセミコロン (;) を付けます。

**例題 1.2.1**

つぎに示す行列を定義してみます。1) では、3 × 3 の実数行列 A を定義し、2) では、複素行列 C を定義し ます。

1)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

2)

$$C = \begin{bmatrix} 1+2i & 3-2i \\ 2-i & 4+3i \end{bmatrix}$$

1)

```
>> A = [1 2 3;4 5 6;7 8 0]
A =
        1   2   3
        4   5   6
        7   8   0
```

ここでは、A という変数名の行列を定義し ました。要素間をブランクで、行間はセミコロン (;) で区切り、要素全体を鍵括弧 ([ ]) で囲んで定義しています。この場合、最後にセミコロンを付けていませんので、定義された行列が画面に表示されます。MATLAB では、文字列 (または文字変数) は大文字と小文字を区別し ますので、A と a という 2 つの変数を別のものとして定義することができます。また、数字から始まる変数名は、認識されません。

2)

```
>> C = [1+2i, 3-2i
        2-i, 4+3i]
C =
      1.0000 + 2.0000i   3.0000 - 2.0000i
      2.0000 - 1.0000i   4.0000 + 3.0000i
```

ここでは、要素間をカンマ (,) で区切り、行間は改行で定義しています。複素行列を定義する場合、虚数単位には i または j を使います (表示は i になります)。

上記例題で使った虚数単位 i のような変数は、永久変数として MATLAB があらかじめ持っている変数です。このような変数には、pi(円周率)、eps(浮動小数点相対精度) 等があります。変数を定義するときには、これらと同じ変数名をつけないよう注意してください。これらと同じ名前の変数を定義したときは、新たに定義された変数が参照されてし まいます。

また、MATLAB では、無限大や Not-a-Number も扱うことができます。無限大は Inf、Not-a-Number は NaN で定義し ます。

2. MATLAB の関数を使う方法

MATLAB は、一般行列関数や特殊行列関数として下記のような関数を提供しています。また、行列要素の抽出や付け足し等の操作も容易に行えます。

| | |
|---|---|
| zeros | 零行列を作成 |
| ones | 要素が全て 1 の行列を作成 |
| eye | 単位行列を作成 |
| diag | 対角行列を作成 |
| magic | 魔方陣行列を作成 |
| : | 線形等間隔のベクトルを作成 |

**例題 1.2.2**

MATLAB の行列関数を用いて、様々な行列を定義し、その要素に対する操作を行います。

1) 零行列、対角行列等の基本行列の定義

2) 等間隔ベクトルの扱い

3) 行列の大きさ

4) 行列要素の抽出、入れ替え、追加

1) 2 行 3 列の零行列 X1 と、2,5,7 を対角要素に持つ対角行列 X2 を定義し ます。

```
>> X1 = zeros(2, 3)
X1 =
```

```
        0   0   0
        0   0   0

>> X2 = diag([2 5 7])
X2 =
        2   0   0
        0   5   0
        0   0   7
```

関数 zeros は、指定した大きさの零行列を作成します。入力引数として、行数と列数を与えます。また、1 つの数値のみを与えた場合は、正方行列を作成します。すべての要素が 1 の行列を作成する関数 ones、単位行列を作成する関数 eye も同様です。

関数 diag は、与えられたベクトルを対角要素とする対角行列を作成します。また、逆に行列の対角要素を取り出すこともできます。

2）等間隔のベクトルを定義します。Y1 として、25 から 0 まで-5 刻みで変化する数列を定義し、Y2 として 1 から 5 まで 1 ずつ変化する数列を定義します。Y3 は、0 から 3 刻みに増加する数列で、要素が最終値と一致しない場合の例です。

```
>> Y1 = 25:-5:0
Y1 =
       25   20   15   10    5    0

>> Y2 = 1:5
Y2 =
        1    2    3    4    5

>> Y3 = 0:3:10
Y3 =
        0    3    6    9
```

等差数列のベクトルを定義する場合、コロン記号 (:) を用いることができます。初期値、公差、最終値をそれぞれコロン記号で区切って定義します。公差を省略した場合は、公差 1 のベクトルになります。Y3 のように最終値が数列要素と一致しない場合、最終値以下で条件を満たす値がベクトルとして出力されます。また関数 linspace,logspace を使ってベクトルを定義することもできます。

3）行列の大きさを調べます。

```
>> d = size(X1)
d =
       2    3
```

行列の大きさを知るには、関数 size を使います。出力引数を 1 つ与えると、2 要素のベクトルとして [行数 列数] の形で出力されます。出力引数を 2 つ与えると、行数と列数がそれぞれスカラ値として出力されます。また、行数と列数のうち大きい方を出力する関数 length もあります。

4）行列要素の抽出や、入れ替え、追加を行います。ここでは、はじめに 3 × 3 の乱数行列 Z1 を定義し、Z1 に対し、これらの操作を行います。

```
>> Z1 = rand(3)
Z1 =
        0.9501    0.4860    0.4565
        0.2311    0.8913    0.0185
        0.6068    0.7621    0.8214

>> Z1(3, 2) = 1
Z1 =
        0.9501    0.4860    0.4565
        0.2311    0.8913    0.0185
```

```
        0.6068    1.0000    0.8214

>> b = Z1(:, 1)
b =
        0.9501
        0.2311
        0.6068

>> Y2(10) = 10
Y2 =
        1    2    3    4    5    0    0    0    0   10

>> Z2 = [X2 Z1]
Z2 =
        2.0000         0         0    0.9501    0.4860    0.4565
             0    5.0000         0    0.2311    0.8913    0.0185
             0         0    7.0000    0.6068    1.0000    0.8214
```

行と列のインデックスを与えることによって、その要素を抽出したり入れ替えることができます。コロン記号は、すべての行 (または列) を意味しますので、変数 b には Z1 のすべての行の 1 列目の要素が入ります。Z1(:) とすると、Z1 のすべての要素を列ベクトルとして表します。なお、関数 rand は 0 と 1 の間の一様乱数を発生させます。行列要素の追加は、ある特定のインデックスに入る数値を与えることによっても行うことができます (Y2 参照)。この時、指定しなかった要素には 0 が入ります。また、Z2 のように要素数が合う場合は、行列を要素として入れることもできます。要素数が合わない場合には、下記のようなエラーメッセージが表示されます。

```
>> [A X1]
??? Error using ==> horzcat
All matrices on a row in the bracketed expression must have the
 same number of rows.
```

### 1.3 文字列の扱い

MATLAB では、文字データを扱うこともできます。文字列は、シングルクォートではさみ込んで定義します。なお、MATLAB では、データの型や配列の大きさの宣言は必要ありませんので、文字列も数値データ同様の定義ができます。

また、MATLAB で文字列を扱うために、下記のような関数が用意されています。

| | |
|---|---|
| eval | 文字列の実行 |
| str2mat | 文字列から文字行列を作成 |
| int2str | 整数値を文字列に変換 |
| num2str | 数値を文字列に変換 |
| str2num | 文字列を数値に変換 |

**例題 1.3.1**

下記の 2 行の文字行列 name を定義してみます。

$$\text{name} = \begin{bmatrix} \text{matlab} \\ \text{simulink} \end{bmatrix}$$

```
>> name1 = 'matlab  ';
>> name2 = 'simulink';
>> name = [name1 ; name2]

name =

matlab
simulink
```

**例題 1.3.2**

例題 1.3.1 で定義した文字行列 name を、MATLAB の関数 str2mat を使って定義します。

```
>> name1 = 'matlab';
>> name2 = 'simulink';
>> name = str2mat(name1, name2);
```

関数 str2mat を利用して定義する場合、文字数を合わせる必要はありません。

## 1.4 データの管理

MATLAB では、配列宣言などのメモリ管理をユーザに要求しません。MATLAB にデータが入力 (定義) されると、そのデータのためのメモリが割り当てられます。この割り当てられたメモリ領域をワークスペースと呼びます。この領域にあるデータは、MATLAB を終了するか、または clear コマンドが実行されるまで保持されます。なお、既に割り当てられている変数と同じ変数名を使用した場合、たとえ異なる配列の行列であっても、ワーニング無しに上書きしてしまいますので注意してください。

ワークスペース内にどのような変数が定義されているかを知るには、who コマンドを使います。また、whos コマンドを使うと、さらに詳しい情報を得ることができます。左から変数名、大きさ、要素数、使用メモリ、密度 (スパース行列の場合、密度が表示されます)、複素行列か否かという情報を得ることができます。

```
>> whos

 Name         Size         Bytes  Class

 A            3x3             72  double array
 C            2x2             64  double array (complex)
 X1           2x3             48  double array
 X2           3x3             72  double array
 Y1           1x6             48  double array
 Y2           1x10            80  double array
 Y3           1x4             32  double array
 Z1           3x3             72  double array
 Z2           3x6            144  double array
 b            3x1             24  double array
 d            1x2             16  double array
 name         2x8             32  char array
 name1        1x6             12  char array
 name2        1x8             16  char array

Grand total is 110 elements using 732 bytes
```

任意の変数をワークスペースから消去するには、'clear 変数名' とします。なお、'clear' だけ実行すると、ワークスペース内の変数全てが消去されます。また、'clear 関数名' とすると、メモリ上に読み込まれていた M-ファイルを消去します。

```
>> clear name2
>> who
Your variables are:

A         X2        Y3        b         name1
C         Y1        Z1        d
X1        Y2        Z2        name
```

---

コラム　変数名

データを定義する場合、出力変数名を指定しないとテンポラリ変数 ans に格納されます。この変数が必要な場合は、何らかの変数に移しておく必要があります。

```
>> magic(3)
ans =
        8     1     6
        3     5     7
        4     9     2
>> A = ans
A =
        8     1     6
        3     5     7
        4     9     2
```

---

### 1.5 基本的な行列演算

MATLAB では、行列の四則演算を行うために、下記のような関数が用意されています。また、四則演算だけではなく、行列操作を行う関数や基本的な数学関数などもあります。

| | |
|---|---|
| + | 加算 |
| − | 減算 |
| * | 乗算 |
| / | 右割り |
| \ | 左割り |
| ^ | べき乗 |
| ' | 転置 |
| reshape | 行列の変形 |
| fliplr | 左右方向への行列要素の入れ替え |
| flipud | 上下方向への行列要素の入れ替え |
| inv | 逆行列 |
| abs | 絶対値 |
| log | 自然対数 |
| log10 | 常用対数 |
| exp | 指数関数 |
| sqrt | 平方根 |

**例題 1.5.1**

つぎの各行列演算を行います。1) では行列の加算を行い、2) では行列積を求めます。3) では行列の対応する要素毎を乗算します。

1)

$$\begin{bmatrix} 1 & 6 & 3 \\ 7 & 0 & 2 \\ 4 & 5 & 9 \end{bmatrix} + \begin{bmatrix} 7 & 2 & -1 \\ 6 & 8 & 3 \\ 9 & 2 & 2 \end{bmatrix}$$

2)

$$\begin{bmatrix} 2 & 6 \\ 8 & 3 \\ 7 & 1 \end{bmatrix} \begin{bmatrix} 2 & 8 & 7 \\ 6 & 3 & 1 \end{bmatrix}$$

3)

$A = \begin{bmatrix} 3 & 6 \\ -2 & 1 \end{bmatrix}$ $B = \begin{bmatrix} 8 & 0 \\ 5 & 7 \end{bmatrix}$ としたとき、要素毎の積を求める。

$$\begin{bmatrix} A_{11} \times B_{11} & A_{12} \times B_{12} \\ A_{21} \times B_{21} & A_{22} \times B_{22} \end{bmatrix}$$

1)

```
>> X = [1 6 3;7 0 2;4 5 9]
X =
```

```
     1     6     3
     7     0     2
     4     5     9
>> Y = [7 2 -1;6 8 3;9 2 2]
Y =
     7     2    -1
     6     8     3
     9     2     2
>> X + Y
ans =
     8     8     2
    13     8     5
    13     7    11
```

このような行列演算を行う場合、MATLAB では for ループを組んで要素毎に繰り返し演算を行うのではなく、一度に行列全体に対して演算を行うことができます。これは、加算を行うときだけではなく、他の行列演算に対しても有効ですし、例えば exp(X) を実行するだけで、行列 X の各要素の値に対する指数関数の値を、一度に求めることもできます。

2)

```
>> A = [2 6;8 3;7 1];
>> A * A'
ans =
    40    34    20
    34    73    59
    20    59    50
```

掛け合わせる 2 つの行列の値に注目すると、これらの行列はお互いに転置の関係にあることが分かります。MATLAB では、行列を転置するにはシングルクォート (') で表します。また複素行列 C に対して C' を実行すると、C の共役転置行列が得られます。複素行列を転置するには、C.' と実行します。

3)

```
>> A = [3 6;-2 1];
>> B = [8 0;5 7];
>> A .* B
ans =
    24     0
   -10     7
```

この演算のように、要素毎の乗算を行う場合、演算子 $*$ の前にドット (.) を付けます。ドットを付けて要素毎の演算を行うものとして、このほかに除算 (/)、逆割り (\)、べき乗 (^) の演算があります。

**例題 1.5.2**

つぎの連立一次方程式を解きます。

$$\begin{cases} x + 5y &= 7 \\ 2x + 4y &= 8 \end{cases}$$

```
>> A = [1 5;2 4];
>> b = [7;8];
>> n = rank(A)
n =
        2
>> inv(A) * b
ans =
    2.0000
    1.0000
```

連立一次方程式を解く場合は、このように逆行列を用います。このとき、行列がフルランクでなければなりません。また、A\bとしても求められます。解として、x=2,y=1が得られたことが分かります。

下記は、行列がフルランクではない例です。この場合、関数invから値は出力されますが、正しい解が求まらないことがあります。

$$\begin{cases} 16x_1 + 2x_2 + 3x_3 + 13x_4 &= -25 \\ 5x_1 + 11x_2 + 10x_3 + 8x_4 &= 62 \\ 9x_1 + 7x_2 + 6x_3 + 12x_4 &= 82 \\ 4x_1 + 14x_2 + 15x_3 + x_4 &= -85 \end{cases}$$

```
>> A = magic(4);
>> b = [-25 62 82 -85]';
>> n = rank(A)
n =
     3
>> x = inv(A) * b

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.306145e-17.

x =
  -11
   -4
  -12
   17
>> A * x
ans =
   17
    5
   61
 -151
```

### 1.6 宿題

1）以下のデータをMATLABの関数を使って定義してみましょう（1〜3）。

1-1)

$$\begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{bmatrix}$$

1-2)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 & 0 & 3 \end{bmatrix}$$

1-3)

$$\begin{bmatrix} 2 & 2 & 2 & 2 \\ 4 & 4 & 4 & 4 \\ 6 & 6 & 6 & 6 \\ 8 & 8 & 8 & 8 \\ 10 & 10 & 10 & 10 \end{bmatrix}$$

2）つぎの連立方程式の解を求め、画面上に下記のように表示しましょう。

Answer : x=□, y=□, z=□

$$\begin{cases} 2x + y + 5z &= 5 \\ 2x + 2y + 3z &= 7 \\ x + 3y + 3z &= 6 \end{cases}$$

3）関数rootsを使わずに、次の二次方程式の解を求めましょう。

$x^2 + 6x + 10 = 0$

ヒント

1-1）データが 0.1 から 0.6 まで 0.1 刻みに並んでいることに注目しましょう。行列の変形を行うには、関数 reshape を使います。

1-2）2〜4 行目の 4〜6 列目の対角要素に、1,2,3 と数値が並んでいることに注目しましょう。

1-3）それぞれの列の要素は、等間隔の数列になっています。また、1 列目から 4 列目まで同じ値になっていることに注目しましょう。

2）画面上に表示するには、disp コマンドを使います。

3）二次方程式の解の公式により求めてみましょう。

# 第 2 回 プログラミング機能の利用

## 2.1 データ解析の手順

データ解析を行う場合、一般的にはつぎに示す手順にしたがって行います。

step1　データ入力
step2　データ処理
step3　結果の可視化

MATLAB を使ってデータ解析を行う場合も、この手順に従います。ここでは、簡単な例題を用いて、MATLAB を使ったデータ解析の手順を確認してみましょう。

**例題**

ある実験から得られた、2 分ごとの温度変化のデータ (下表) があります。このデータを多項式で近似し、オリジナルのデータと近似値との差の二乗和を誤差として求めます。また、オリジナルのデータ点と、近似曲線を表示して比較してみましょう。

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 時間 | t | 0 | 2.0 | 4.0 | 6.0 | 8.0 | 10.0 | 12.0 | 14.0 | 16.0 | 18.0 | 20.0 |
| 温度 | f | 37.5 | 37.8 | 38.3 | 39.5 | 40.6 | 41.1 | 40.8 | 41.2 | 41.0 | 41.0 | 41.2 |

step1 データ入力

時間データを t、温度データを f として、それぞれ MATLAB 上で定義します。

```
>> t = [0 2.0 4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0];
>> f = [37.5 37.8 38.3 39.5 40.6 41.1 40.8 41.2 41.0 41.0 41.2];
```

step2 データ処理

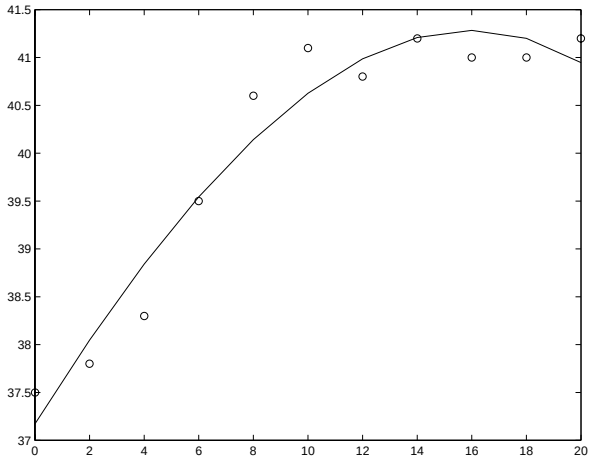
上記で定義したデータを 3 次の多項式で近似し、二乗和誤差がどのくらいになるか計算してみます。多項式近似を行うには、関数 polyfit を使います。この関数から出力される p は、求められた多項式の係数ベクトルです。関数 polyval では、求められた多項式に時間の値 t を代入し、温度の近似値 F を求めます。求める二乗和誤差は、E で示される値です。

```
>> p = polyfit(t, f, 3);
>> F = polyval(p, t);
>> E = sum((f - F) .^ 2) / length(f)
E =
        0.1015
```

step3 結果の可視化

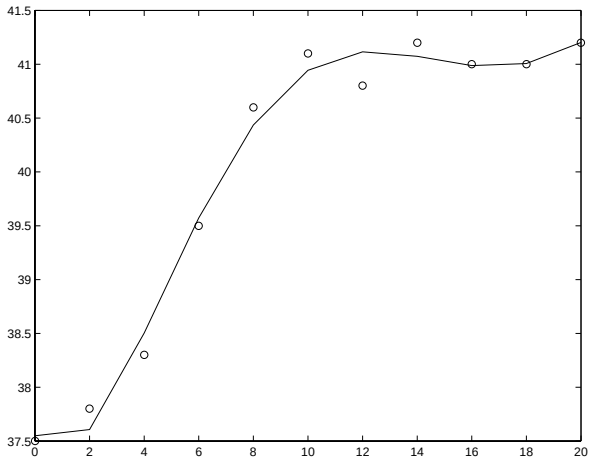
オリジナルのデータと近似点を比較してみます。2 次元のデータを表示するには、関数 plot を使います。青の丸印で示される点がオリジナルのデータ点で、緑色の曲線が近似点を示しています。

```
>> plot(t, f, 'o', t, F)
```




MATLAB を使うと、上記のような方法でデータ解析を行うことができます。しかし、結果を見て分かるように、上記で行った近似はあまり良い近似とはいえません。近似多項式の次数を高くして、もう一度 step 2 以降の作業を行ってみます。今度は、5 次の多項式で近似してみます。

```
>> p = polyfit(t, f, 5);
>> F = polyval(p, t);
>> E = sum((f - F) .^ 2) / length(f)
E =
        0.0230
>> plot(t, f, 'o', t, F)
```




上記の例題では、パラメータ (多項式の次数) を変更して、一連のコマンド群を繰り返し実行しました。MATLAB ではこのように同じ実行を繰り返し行う場合には、M-ファイルと呼ばれるファイルを作成しておくと、効率よく作業を進めることができます。M-ファイルについては、次節以降で詳しく取り上げます。

### 2.2 コマンドの一括処理 (スクリプト M-ファイル)

前節で取り上げた例題のように、一連の操作を繰り返し実行する場合、コマンド群を再度利用可能な形に変更することができます。コマンド群を実行順にファイルに書き綴り保存しておくと、プロンプトに対してファイル名をタイプインするだけで、複数のコマンドが一括処理されます。

ファイルを作成する場合、テキストエディタを使います。Windows 版、Macintosh 版の場合は、メニューバーの中の「File」から、「New」—「M-file」を選択してエディタを起動することもできます (任意のエディタを登録することも可能です)。

ここでは、前節で取り上げた例題をファイルに保存し、そのコマンド群の一括処理を行ってみます。ファイルを保存する場合、MATLAB で実行出来る形式にするため、拡張子 m を付けます。

では、下記のコマンド群をファイルとして保存してみましょう。ファイル名には、fit.m と付けます。

(ファイル名 fit.m)

```
t = [0 2.0 4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0];
f = [37.5 37.8 38.3 39.5 40.6 41.1 40.8 41.2 41.0 41.0 41.2];
p = polyfit(t, f, n);
F = polyval(p, t);
E = sum((f - F) .^ 2) / length(f)
t1 = 0:20;
f1 = polyval(p, t1);
plot(t, f, 'o', t1, f1)
```

作成したファイルを実行するには、拡張子を除いたファイル名のみをタイプインします。ただしこの場合は、関数 polyfit で与える多項式の次数 n を、パラメータとしているので (ファイル内 3 行目)、はじめに n を定義してから実行します。

```
>> n = 3;
>> fit
E =
        0.1015
```

上記を実行すると、二乗和誤差 E が表示され、Figure ウィンドウにグラフが描かれます。

また、近似多項式の次数を変更するには、n を定義し直してただちに実行することができます。

```
>> n = 5;
>> fit
E =
        0.0230
```

MATLAB では、このようにコマンドを一括処理するいわゆるバッチファイル的なものを、スクリプト M-ファイルと呼んでいます。スクリプト M-ファイル内のコマンドは、上から順に実行され、定義された変数も MATLAB のワークスペースに追加されます。

### 2.3 機能拡張 (ファンクション M-ファイル)

前節では、スクリプト M-ファイルを作成することによって、パラメータを変更しながら実行を繰り返し行う方法を学びました。この節では、変更したいパラメータを、入力引数として持つ関数を作成してみましょう。

新たな関数を作成する場合も、テキストエディタを使って、拡張子 m のファイルを作成します。

(ファイル名 funfit.m)

```
function E = funfit(t, f, n)
% Fits a polynomial to (t, f) data.
% E = funfit(t, f, n)
% t,f : Input data
% n : Degree of polynomial
% E : Error sum of squares
p = polyfit(t, f, n);
F = polyval(p, t);
E = sum((f - F) .^ 2) / length(f);
plot(t, f, 'o', t, F)
```

関数を作成するとき、ファイルの先頭行で function 宣言をします。上記のファイル関数では、E が出力引数で、t,f,n が入力引数、funfit が関数名であることを示しています。

ファイル内の%記号は、コメント行であることを表し、実行されません。関数の使い方等を記述しておくと、ヘルプ機能 (2.5 節 参照) を用いてコマンドウィンドウに表示することもできます。

では、上記の関数をファイル名 funfit.m として作成し、実行してみましょう。

```
>> E = funfit(t, f, 3)
```

```
E =
        0.1015
```

実行するには、必要な入力引数を与えます (上記の場合、変数 t,f があらかじめ定義されているものとします)。この結果、スクリプト M-ファイルを実行したときと同じように、二乗和誤差が求まり、Figure ウィンドウにグラフが表示されます。ただし、この関数を実行しても、関数内で定義された出力引数以外の変数 (ここでは p,F) は、関数内のローカル変数として扱われ、ワークスペースには残りません。

MATLAB では、このように新たにユーザ関数を容易に作成することができます。このようなファイルを、ファンクション M-ファイルと呼んでいます。

---

コラム　組み込み関数と MEX-ファイル

MATLAB 上で実行出来る関数には、M-ファイルのほかに、組み込み関数と MEX-ファイルがあります。

- 組み込み関数 (Built-in Function)
  LINPACK,EISPACK プロジェクトで開発された行列計算のアルゴリズムを、C 言語で書き直した実行ファイルのことを組み込み関数と呼んでいます。数学関数、ファイル入出力関数、グラフィックス関数、プログラミング関数等があります。これらの基本的な関数を MATLAB の核として、M-ファイルが作成されています。
- MEX-ファイル
  C 言語や FORTRAN で作成されたプログラムを、MATLAB 上で実行可能なオブジェクトとして作成したファイルを MEX-ファイルと呼んでいます。計算の高速化、ソフトウェア資産の継承、アルゴリズムの隠蔽等のために利用されます。詳細は「エクスターナルインタフェースガイド」を参照して下さい。なお、コンパイラバージョンの制限等があるので、注意して下さい。

---

### 2.4 M-ファイル

前節までで学んだ M-ファイルについて、特徴をもう一度まとめておきましょう。MATLAB が提供する 500 以上の関数のうち、ほとんどが M-ファイルとして提供されています。M-ファイルを理解することによって、MATLAB の関数構造について理解しましょう。

- M-ファイルは、スクリプト M-ファイルとファンクション M-ファイルの 2 種類があります。スクリプト M-ファイルは、ファイル内のコマンド群を順に実行するもので、一連のコマンドを一括処理する場合などに用いられます。ファンクション M-ファイルは、ファイルの先頭で function 宣言を行い、新たな機能を持つ関数を作成する場合に用いられます。
- M-ファイルは、インタプリタにより処理されるので、コンパイルやリンクを行う必要はなく、拡張子 m を付けたファイルに保存するだけで、MATLAB 上で実行可能となります。
- M-ファイルはテキスト形式のファイルなので、エディタを使って作成したり、既存の M-ファイルの内容を確認することができます。MATLAB がオリジナルで提供している関数 (M-ファイル) についても、同様にアルゴリズムの参照や内容の編集を行うことができるので、ユーザのニーズに合わせてカスタマイズすることも容易に行えます。
- M-ファイル内に% 記号がある場合、その行はコメント行となり実行されません。またファイルの先頭部分にあるコメント行は、help コマンドによりコマンドウィンドウ上に表示することができます。関数の使い方等の情報を書き込んでおくと便利です。
- M-ファイル内では、if 文や for 文、while 文のような制御文を扱うこともできます。さらにファンクション M-ファイルでは、再帰関数の呼び出しを行うこともできます。

### 2.5 ファイル管理に関する機能

この節では、つぎの 3 つの項目について取り上げます。

1．ヘルプ機能
2．MATLAB サーチパス
3．ファイル検索

1．　ヘルプ機能

MATLABでは、オンラインヘルプ機能としてhelpコマンドがあります。引数がディレクトリの場合は、そのディレクトリ（またはToolbox）に含まれる関数一覧を出力し、関数の場合は、その使用方法の概要を見ることができます。これらは、ユーザが登録することもできます。ディレクトリ内の関数を一覧するには、そのディレクトリ下にcontents.mを作成しておくと、連続したコメント行（%で始まる）で記述されている内容が出力されます。また、関数の概要を表示するには、そのファンクションM-ファイルの先頭の連続したコメント行（%で始まる行）に、その内容を記述しておきます。

| | |
|---|---|
| >> help | MATLABサーチパス上にあるディレクトリ一覧を表示 |
| >> help <directory> | そのディレクトリに含まれる関数一覧を表示 |
| >> help <function> | その関数についての概要を表示 |
| >> helpwin | helpウィンドウを開く（マウスで操作可能） |

MATLABの関数は、機能別のディレクトリ構造なので、ディレクトリ一覧の中から目的の関数を検索することができます。

またキーワード検索する場合は、lookforコマンドを利用します。lookforコマンドでは、M-ファイルの最初のコメント行内にそのキーワードが含まれているものが検索されます。

2. MATLABサーチパス

MATLAB上で利用するM-ファイルやMEX-ファイル、データファイル等は、カレントディレクトリ（フォルダ）、またはMATLABのサーチパス上になければなりません。ディレクトリをパスに追加するには、下記のように行います。ただし、日本語や空文字を含むディレクトリ名には対応できません。ディレクトリ名は、英数半角8文字以内でつけるようにしてください。

```
>> path(path,'追加ディレクトリ名')
```

または

```
>> path('追加ディレクトリ名',path)
```

MATLABサーチパスをデフォルト設定にする場合は、MATLAB起動ディレクトリ下にstartup.mファイルを作成し、ファイル内にこの設定を記述して下さい。このファイルは、MATLAB起動時にいつも実行されるので、サーチパスだけでなく、様々なデフォルト設定を指定することができます。

3. ファイル検索

whichコマンドを用いると、カレントディレクトリまたはMATLABサーチパス上にあるMATLABファイルを検索し、絶対パスを表示します。ファイル作成時に、既存のファイルと同じファイル名を付けてしまうことがないように、whichコマンドで確認してください。

M-ファイルの場合

```
>> which polyfit
/usr/local/opt/matlab6.51/toolbox/matlab/polyfun/polyfit.m
```

組み込み関数の場合

```
>> which zeros
zeros is a built-in function.
```

ファイルが存在しない場合

```
>> which maximum
maximum not found.
```

なお、MATLABファイルが検索される優先順位は、

カレントディレクトリ ＞ 組み込み関数 ＞ MATLABサーチパス

となります。検索されたカレントディレクトリやMATLABサーチパス上に同じファイル名のM-ファイルとMEX-ファイルが存在する場合は、通常MEX-ファイルが優先されます。また関数と同じ名前の変数が定義されていると、その変数が参照されてしまいますので、関数名や変数名を決定する際には注意してください。

## 2.6 例題

**例題 2.6.1**

2.1 節で取り上げた多項式近似の例題で、二乗和誤差が 0.01 以下になるような、最小次数の近似多項式を求めます (この場合、最大次数は 10 次とします)。

(ファイル名 exp261.m)

```
t = 0:2:20;
f = [37.5 37.8 38.3 39.5 40.6 41.1 40.8 41.2 41.0 41.0 41.2];
for n = 1:10
      p = polyfit(t, f, n);
      F = polyval(p, t);
      E = sum((f - F) .^ 2) / length(f);
      if E <= 0.01
            break
      end
end
n
```

上記を実行すると、7 次の多項式が誤差の条件を満たす最小次数の多項式であるという結果が得られます。

```
>> exp261
n =
     7
```

この例題では、M-ファイル内で for 文、if 文を使っています。for 文では、n が 1 から 10 まで 1 ずつ増加する間、end までの各ステートメントを実行します。if 文は、条件 E<=0.01 が満たされるとき、end までのステートメントが実行されます。また、ここでは break コマンドも使っています。break は for 文や while 文で構成されたループを停止させ、ループから抜け出します。ここでは、if 文の条件が満たされるとき、for 文のループから抜け出すという意味です。

この例題は、while 文を使って行うこともできます。

(ファイル名 exp2611.m)

```
t = 0:2:20;
f = [37.5 37.8 38.3 39.5 40.6 41.1 40.8 41.2 41.0 41.0 41.2];
n = 0;
E = inf;
while E > 0.01
      n = n + 1;
      p = polyfit(t, f, n);
      F = polyval(p, t);
      E = sum((f - F) .^ 2) / length(f);
end
n
```

while 文では、条件 E>0.01 が満たされている間、end までの各ステートメントが実行されます。

**例題 2.6.2**

フィボナッチ数列を求める関数を作成します。フィボナッチ数列とは、下記の条件を満たす数列です。

$$a_1 = 1, \quad a_2 = 1, \quad a_n = a_{n-1} + a_{n-2} \quad (n \geq 3)$$

(ファイル名 fibfun.m)

```
function f = fibfun(n)
% FIBFUN for calculating Fibonacci numbers.
% Incidentally, the name Fibonacci comes from
```

```
% Fillius Bonassi, or "son of Bonassus".
if n > 2
     f = fibfun(n-1) + fibfun(n-2);
else
     f = 1;
end
```

例えば、フィボナッチ数列の 10 項目の値を求めるには、下記のように実行します。

```
>> f = fibfun(10)
f =
    55
```

この例題では、if 文を使って条件分岐を行っています。また、再帰関数の呼び出しを行っています。

**例題 2.6.3**

与えられたベクトルの各要素間の差を求める関数を作成します。

（ファイル名 myminus.m）

```
function Y = myminus(X)
% Y = [X(2)-X(1) X(3)-X(2) ... X(n)-X(n-1)]
Y   = [];
for i = 1:length(X)-1
     s = X(i+1) - X(i);
     Y = [Y, s];
end
```

この関数を使って、5 要素の乱数ベクトルの各要素間の差を求めてみます。

```
>> X = rand(1,5)
X =
    0.9554    0.5561    0.1482    0.9833    0.4088
>> Y = myminus(X)
Y =
   -0.3993   -0.4080    0.8352   -0.5745
```

出力引数 Y は、はじめに空行列として定義しておきます。空行列は、大きさが 0 の行列としてワークスペース内に存在しますので、定義されていないこととは意味が異なります。for ループ内では、各要素間の差を計算し、変数 s に出力しています。そして、空行列として定義していた Y に、値を順に追加してゆきます。Y は順に 1 要素、2 要素、… のベクトルとなります。

---

コラム　文字数制限

MATLAB のコマンドライン上での制限は、直接 MATLAB プロンプト上にタイプ入力する場合も、M-ファイルで実行する場合も 256 文字です。

また、継続記号を利用した場合、バッファでの制限は継続記号 (...) を除いて 4096 文字です。これは例えば大きな行列を定義したり、256 文字を超える式を扱う時、...　を用いて継続させますが、この継続記号分を除いてカウントされます。

---

## 2.7 宿題

1）例題 2.6.2 で作成した関数 fibfun を使って、n 項目までのフィボナッチ数列をベクトルとして出力する関数 fib を作成しましょう。

2）n 以下の素数を求める関数を作成しましょう。

ヒント

1）出力されるベクトルを、はじめに空行列として定義して、要素を順に追加してゆきましょう（例題 2.6.3 参照）。

2）

```
整数列      2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
            1  1  1  1  1  1  1  1  1  1  1  1  1  1  1

i=2 のとき *     0     0     0     0     0     0     0
i=3 のとき    *        0        0        0        0
i=4   ×

結果        1  1  0  1  0  1  0  0  0  1  0  1  0  0  0
```

全て True(1) として始めます。i=2 からループ計算を行い、i の倍数を False(0) にしてゆきます。なお、既に False(0) になっている i については、ループ計算を行う必要はありません。

## 第 3 回 データの入出力

### 3.1 ファイル形式

MATLAB のファイル入出力機能を利用する場合、データファイルの種類によって方法が異なります。ファイルは、アスキファイルとバイナリファイルに分類され、またそれぞれのファイルについてもいくつかのケースに分類されます。それぞれのファイルの扱いについて、次節以降で形式別に紹介します。

バイナリファイルを扱う場合、読み込む (または書き込む) データの型を分かっていることが前提となります。また、MATLAB 固有のバイナリ形式ファイルもあり、MAT-ファイルと呼んでいます。

### 3.2 アスキファイルの扱い

アスキファイルの利点は、容易に編集することができることです。しかしバイナリファイルに比べ、ファイルサイズが大きくなってしまいます。

#### 3.2.1 アスキファイルへの書き込み

まず、アスキファイルに、数値データのみを書き込む場合を考えます。

**例題 3.2.1**

ワークスペース内の変数 X をアスキファイル e321.dat に、有効桁数 16 桁で書き込みます。

```
>> X
X =
    1.1650    0.3516    0.0591
    0.6268   -0.6965    1.7971
    0.0751    1.6961    0.2641

>> save e321.dat X -ascii -double
```

アスキファイルに数値データを書き込むには、save コマンドに-ascii オプションを付けて実行します。save コマンドは、"save ファイル名 変数名 オプション"のように使用します。デフォルトでは有効桁数 8 桁で書き込みますが、-double オプションを付けると、有効桁数 16 桁で書き込みます。ここで作成されたファイル e321.dat の内容を、type コマンドを使って確認してみましょう。

```
>> type e321.dat
  1.1649535105006560e+000  3.5160690276852340e-001  5.9059777981351200e-002
  6.2683908263243000e-001 -6.9651253516368220e-001  1.7970717836948180e+000
  7.5080154677683210e-002  1.6961424807470760e+000  2.6406852881722590e-001
```

save コマンドによって作成されたアスキファイルには、ヘッダーがないので変数名等の情報は残りません。また、複数の変数を 1 つのファイルに保存することはできますが、ファイル内では変数間の区切りは認識されません (読み込むときに、元の複数の行列として読み込めません)。また、複素行列についても適用されないので、注意してください。

以下は複素行列を save コマンドによってアスキファイルに書き込んだ例です。実部のみしか保存されていません (Version 6 以降では以下に示すような Warning が出ます)。複素行列を保存するには、実部と虚部に分けてそれぞれ別のファイルに保存してください。

```
>> C
```

```
C =
   1.0000 + 2.0000i   3.0000 - 2.0000i
   2.0000 - 1.0000i   4.0000 + 3.0000i

>> save comp.dat C -ascii
Warning: Imaginary part of complex variable 'C' not saved to ASCII file.
>> type comp.dat
  1.0000000e+00  3.0000000e+00
  2.0000000e+00  4.0000000e+00
```

ではつぎに、文字列を含むデータをアスキファイルに書き込みます。

**例題 3.2.2**

データ x の各値に対する正弦値をヘッダ情報とともに、アスキファイル e322.dat に書き込みます。

```
>> x=0:0.1:1;
>> y=[x;sin(x)];
>> fid=fopen('e322.dat','w');
>> fprintf(fid,'%s\n','Example for File I/O');
>> fprintf(fid,'%s\n','x    Sine');
>> fprintf(fid,'%3.1f %6.3f\r\n',y);
>> fclose(fid)

ans =
     0
```

文字列を含むデータをアスキファイルに書き込む場合や、フォーマットを指定して書き込むには、fprintf コマンドを使います。その際、はじめに fopen コマンドによって書き込むファイルを開きます。ここで与える'w' はファイルのアクセスモードで、書き込みモードであることを表しています。

fprintf コマンドの 1 つ目の入力引数には、fopen から出力される値を与えます。2 つ目の引数では書式を指定します。%s は文字列、%f は固定小数点表記、\n は改行コードを表しています。%と f の間の数値は、精度を指定するもので、%3.1f なら小数点以下 1 桁を持った 3 文字の固定小数点表記になります。なお、データがファイルに書き込まれたとき、MATLAB 上での配列は転置されます。

fopen コマンドによって開かれたファイルは、fclose コマンドによって閉じます。このとき出力される 0 という値は、ファイルがきちんと閉じられたことを示しています。

**例題 3.2.3**

変数 data1,data2,･･･,data10 がワークスペース内に存在しているとき、アスキファイル file1.dat には変数 data1 を、file2.dat には変数 data2 を、･･･ file10.dat には data10 をそれぞれ書き込みます。

```
>> f_name = 'file';
>> extension = '.dat';
>> d_name = 'data';
>> for n = 1:10
     filename = [f_name, int2str(n), extension];
     dataname = [d_name, int2str(n)];
     eval(['save ',filename,' ',dataname,' -ascii'])
end
```

eval コマンドは、与えられた文字列をステートメントやコマンドとして解釈し、それを実行します。上記では例えば n=3 のとき、"save file3.dat data3 -ascii"が実行されることになります。

### 3.2.2 アスキファイルの読み込み

前節では、アスキファイルにデータを書き込む方法についてとりあげましたが、今度はファイルからデータを読み込む方法について考えます。まず、数値のみからなるデータを MATLAB に読み込みましょう。

**例題 3.2.4**

例題 3.2.1 で作成したアスキファイル e321.dat から、データを読み込みます。

```
>> load e321.dat
>> e321

e321 =

    1.1650    0.3516    0.0591
    0.6268   -0.6965    1.7971
    0.0751    1.6961    0.2641
```

アスキファイルを読み込むには、load コマンドを使います。このとき、読み込まれたデータの変数名は、拡張子を除いた変数名になります。またこのコマンドは、ファイル内のデータを 1 つの変数として読み込むので、データは行列として読み込める形でなければなりません。例えば、下記のようなデータの場合は読み込めません。

data.txt

```
21.8959   83.0965    6.6842
 4.7045    3.4572   41.7486
67.8865    5.3462   68.6773
67.9296   52.9700   58.8977
93.4693   67.1149   93.0436
38.3502    0.7698
```

```
>> load data.txt
??? Error using ==> load
Number of columns on line 6 of ASCII file data.txt
must be the same as previous lines.
```

では、文字列を含むファイルや、書式付きのファイルからデータを読み込みましょう。

**例題 3.2.5**

例題 3.2.2 で作成したアスキファイル e322.dat から、数値データを変数 data322 として読み込みます。

```
>>  fid = fopen('e322.dat','r');
>>  fgets(fid);
>>  fgets(fid);
>>  data322 = fscanf(fid, '%f %f', [2 inf])'

data322 =

         0         0
    0.1000    0.1000
    0.2000    0.1990
    0.3000    0.2960
    0.4000    0.3890
    0.5000    0.4790
    0.6000    0.5650
    0.7000    0.6440
    0.8000    0.7170
    0.9000    0.7830
    1.0000    0.8410

>>  fclose(fid);
```

アスキファイルにデータを書き込むときと同様に、fopen コマンドによってファイルを開きます。ファイル e322.dat の 1,2 行目は文字列なので、fgets コマンドによって 1 行毎読み込んでおきます。3 行

目以降の数値データを読み込むには、fscanf コマンドを使います。この場合、2 つ目の引数で書式を指定し、3 つ目の引数で読み込むデータの大きさを指定します。[2 inf] とあるのは、2 行の行列として EOF まで読み込むという意味です。

### 3.3 MAT-ファイルの扱い

MAT-ファイルは、MATLAB が持つバイナリ形式のファイルです。MAT-ファイルには、実数以外にも複素数や文字データ、スパース行列も保存することができ、変数名の情報も保たれます。また、各対応プラットフォーム (UNIX,Windows,Macintosh) の MATLAB から利用することができます。これが M-ファイルと共にマルチプラットフォームのアプリケーションソフトウェア MATLAB と言われる理由です。

では、例題を用いて MAT-ファイルの扱いを確認しましょう。

**例題 3.3.1**

実数、複素数、文字データ、Inf,NaN 等を含むデータを定義して、MAT-ファイル e331.mat にこれらを保存します。

```
>> clear
>> dat1 = diag([pi inf NaN]);
>> dat2 = (2 + i) * [1:0.1:2]';
>> dat3 = str2mat('MATLAB', 'Basic', 'Seminar');
>> save e331
>> clear
>> who
>> load e331
>> who

Your variables are:

dat1      dat2      dat3
```

MAT-ファイルに保存するには、save コマンドを使います。使い方はアスキファイルに保存する場合と同じですが、ファイル名の拡張子 mat は省略することができます。また、保存する変数名を省略すると、ワークスペース内の変数すべてが指定した MAT-ファイルに保存されます。さらに引数を与えずに save コマンドを実行した場合には、ワークスペース内の変数すべてを matlab.mat という名の MAT-ファイルに保存します。

MAT-ファイルからデータを読み込むには、load コマンドを使います。load コマンドも save コマンド同様、デフォルトでは MAT-ファイルを扱います。load コマンドによって読み込まれたデータは、保存したときの変数名のまま読み込まれます。

---

コラム　MAT-ルーチン

C 言語や FORTRAN のユーザプログラムと MATLAB で利用するデータを共通化するために、MATLAB データ形式の MAT-ファイルへの入出力ルーチンが提供されています。MAT-ファイルを利用すると、変数情報もヘッダに収納されているのでデータの情報をコンパクトにすることができます。

詳細は「エクスターナルインタフェースガイド」を参照下さい。なお、コンパイラ等の制限があるので、注意して下さい。

---

### 3.4 バイナリファイルの扱い

前節では、MATLAB がもつバイナリ形式の MAT-ファイルについて取り上げましたが、他の形式のバイナリファイルも扱うことができます。

#### 3.4.1 バイナリファイルへの書き込み

**例題 3.4.1**

ワークスペース内の変数 Y を、16 ビットの符号付き整数型としてバイナリファイル e341.bin に書き込みます。

```
>> Y
Y =
    92    39   -14    45   -79
   124    64    62  -165    63
    23    87    98    12    82
  -110    18  -112  -103   -18
    10   -32   -55    12    56

>> fid=fopen('e341.bin','w');
>> fwrite(fid,Y,'int16');
>> fclose(fid);
```

バイナリファイルにデータを書き込むには、fwrite コマンドを使います。fwrite コマンドに与える引数は、1 つ目に fopen からの出力引数 (ファイル識別子)、2 つ目に書き込むデータ (変数名)、3 つ目に書式を指定します。指定できる書式は、MATLAB リファレンスガイドの fread の項目を参照してください。

### 3.4.2 バイナリファイルの読み込み

**例題 3.4.2**

例題 3.4.1 で作成したバイナリファイル e341.bin から、変数 data342 としてデータを読み込みます。

```
>> fid=fopen('e341.bin','r');
>> data342=fread(fid,[5 5],'int16')
data342 =
    92    39   -14    45   -79
   124    64    62  -165    63
    23    87    98    12    82
  -110    18  -112  -103   -18
    10   -32   -55    12    56

>> fclose(fid);
```

バイナリファイルからデータを読み込むには、fread コマンドを使います。fread コマンドでは、読み込む変数名を指定し、1 つ目の引数には fopen からの出力引数を与えます。2 つ目の引数は読み込むデータの大きさを表します。大きさは fprintf コマンドと同じように、inf で指定することもできます。3 つ目の引数では、データの書式を指定します。

---

コラム　DDE(Dynamic Data Exchange) 機能

MATLAB では、Excel などとのデータのやり取りを動的に行うコマンドが用意されています。例えば、Excel を起動してデータファイル (data.xls) を開き、Excel 上の 100 行 5 列目のデータを MATLAB のワークスペース変数 Z に読み込むのは、次の方法でできます。なお、この機能は Windows 版に限ります。

```
>>  channel=ddeinit('excel', 'data.xls');
>>  Z = ddereq(channel, 'r1c1:r100c5');
>>  rc = ddeterm(channel);
```

---

## 3.5 デバッグ機能

前節までとは異なり、この節では MATLAB のデバッグ機能についてご紹介します。

ユーザがファンクション M-ファイルを作る際に、エラー発生時の関数内部のローカル変数のチェックを行う場合等の有効な機能として、MATLAB のデバッグ関数があります。デバッグモードに入ると、プロンプトは K>>になり、このモードでローカル変数を確認することができます。

MATLAB のデバッグ関数には、下記のようなものがあります。

| | |
|---|---|
| dbstop | ブレイクポイントの設定 |
| dbclear | ブレイクポイントの削除 |
| dbcont | 実行の再開 |
| dbdown | ローカルワークスペース環境の変更 |
| dbstack | 呼出されている関数の表示 |
| dbstatus | すべてのブレイクポイントの表示 |
| dbstep | ステップ毎の実行 |
| dbtype | 行番号付き M-ファイルの内容表示 |
| dbup | ローカルワークスペース環境の変更 |
| dbquit | デバッグモードの終了 |

**例題 3.5.1**

関数 quad を使って、関数 humps の数値積分を行います。このとき、関数 quad の 44 行目でデバッグモードに入り、ローカル変数を確認してみましょう。

```
>> dbstop at 58 in quad
>> s1 = quad('humps',0,1);
58  x = [a a+h a+2*h (a+b)/2 b-2*h b-h b];
K>>
```

関数 quad 内のローカル変数情報を見ます。

```
K>> who
Your variables are:

a          f          h          trace
b          funfcn     tol        varargin
```

dbstep コマンドを使って実行を進めます。このとき、ポイントは表示される行の先頭に存在しています。

```
K>> dbstep
59  y = feval(f, x, varargin{:});
K>> dbstep
60  fcnt = 7;
```

デバッグモードから抜け、デバッガの設定を解除します。

```
K>> dbquit
>> dbclear at 58 in quad
```

### 3.6 宿題

1）データファイル hw31.dat の 1 列目と 2 列目のデータの相関係数を計算し、求めた相関係数行列をこのファイルの最後に追加しましょう。

hw31.dat

```
1.2070338e+001 2.3656885e+001
1.3988543e+001 3.7257483e+000
1.5906748e+001 2.7441618e+001
1.7824953e+001 9.9603869e+000
1.9743158e+001 2.6091244e+001
2.1661363e+001 2.9865461e+001
2.3105423e+001 2.1345158e+001
      :              :
      :              :
      :              :
```

2）以下のようなアスキファイル hw32.dat があるとき、ヘッダー内のデータ番号を変数名とし（d1001）、ヘッダー内に書かれている大きさでファイル内の数値データを読み込みましょう。

hw32.dat

```
Data No:1001
Row:2
Coloumn:4
5.8898
9.3044
8.4617
5.2693
0.9196
  :
  :
  :
```

ヒント

1）相関係数を求めるには、関数 corrcoef を使います。また、ファイルにデータを追加保存するときは、fopen コマンドのオプション引数を確認してください。

2）ヘッダー行を読み込むには、fgetl コマンドを使い、読み込んだ文字列内の値を数値に変換するには、関数 str2num を使います。

# 第 4 回 データ処理

## 4.1 統計処理

MATLAB では、初等的な統計処理を行う関数として、下記のようなものがあります。

| | |
|---|---|
| max | 最大値 |
| min | 最小値 |
| mean | 平均値 |
| median | 中央値 |
| std | 標準偏差 |
| sort | 昇順整列 |
| sum | 和 |
| prod | 積 |
| cumsum | 累積和 |
| cumprod | 累積積 |

**例題 4.1.1**

ある測定データ stat.dat(アスキファイル) があります、このデータを読み込み、最小値を求めます。

stat.dat

```
156.2000  154.9000  168.5000
180.5000  160.4000  158.3000
149.8000  172.0000  172.2000
155.6000  176.4000  182.1000
171.8000  174.3000  171.9000
170.0000  168.5000  178.3000
166.5000  162.5000  168.2000
158.1000  177.0000  173.8000
176.5000  171.6000  176.0000
161.1000  169.4000  168.1000
```

```
>> load stat.dat
>> size(stat)
ans =
    10     3
```

```
>> min(stat)
ans =
  149.8000  154.9000  158.3000
```

このデータは、3 列の行列データです。このようなデータに対して関数 min を使うと、各列の最小値を出力します。上表の統計関数はすべて、入力引数が行列のときは列ごとに対する演算を行います。行列データに対して、データ全体の最小値を求めるには、下記のように行います。関数 min に与える変数を stat(:) とすることによって、1 次元配列化しています。

```
>> min(stat(:))
ans =
  149.8000
```

また、行ごとに対する演算は、下記のように行います。

```
>> min(stat')'
ans =
  154.9000
  158.3000
  149.8000
  155.6000
      :
      :
```

**例題 4.1.2**

データを正規化する関数を作成します。ここでは、元のデータから平均値を引き、標準偏差で割った値を正規化された値とします。

normv.m

```
function y = normv(x)
% NORMV Returns a normalized value
m = mean(x);
mm = m(ones(length(x),1),:);
s = std(x);
ms = s(ones(length(x),1),:);
y = (x-mm)./ms;
```

この関数内では、入力値の平均値と標準偏差をそれぞれ求めています。そして入力値からそれぞれ平均値 m を引き、さらにその値を標準偏差 s で割っています。では、実際にデータの正規化を行ってみましょう。データは例題 4.1.1 の stat.dat を用います。

```
>> load stat.dat
>> Y=normv(stat)
Y =
   -0.8391   -1.8953   -0.4923
    1.5854   -1.1399   -2.0419
   -1.4776    0.4532    0.0699
   -0.8989    1.0575    1.5740
    0.7174    0.7691    0.0243
    0.5378   -0.0275    0.9967
    0.1886   -0.8515   -0.5378
   -0.6495    1.1399    0.3130
    1.1863    0.3983    0.6472
   -0.3502    0.0961   -0.5530
```

### 4.2 欠測データの扱い

測定器を使って計測を行うとき、何らかの理由でデータを取得できない場合があります。このような欠測データを MATLAB では、Not-a-Number(NaN) として扱うことができます。

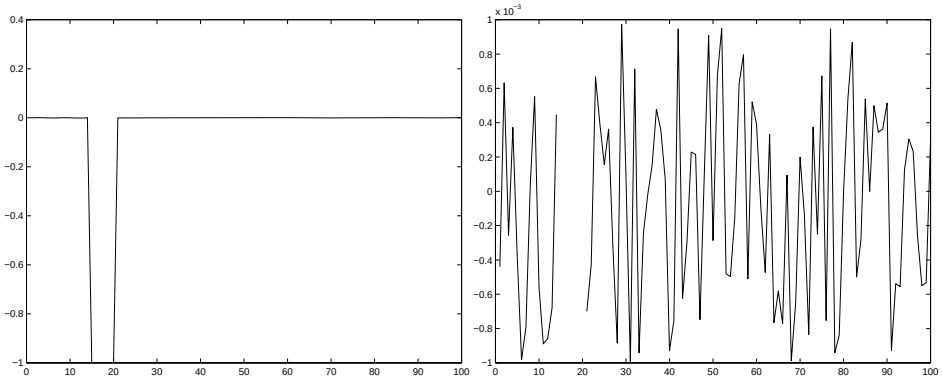
**例題 4.2.1**

計測データのレンジが 1.e-3 で、計測不能時 (15〜20 番目) には-1が入っているデータ M が measure.mat にあります。このデータを plot コマンドを用いてそのまま表示すると左図のようになり、表示範囲を変更しなければデータの変化を読み取ることができません。計測不能時のデータを NaN として表示してみます。

measure.mat

```
>> load measure                  % 1
>> plot(M)                       % 2

>> n_nan = find(M == -1);        % 3
>> M(n_nan) = M(n_nan) * NaN;    % 4
>> figure, plot(M)               % 5
```



関数 find によって、データが-1 となっている部分を探します。このとき出力される n_nan は、-1 となっている要素のインデックスがベクトルとして得られます (% 3)。つぎに n_nan で示される要素について、値を NaN に変更します (% 4)。このとき、等号の左辺と右辺では、要素数を合わせることに注意してください。この結果を表示すると、図の中でデータが抜けている部分があることに注目してください (右図)。

**例題 4.2.2**

例題 4.2.1 で求めた NaN を含むデータ M の平均値を求めます。

```
>> mean(M)

ans =
   NaN
```

データ内に NaN が含まれているとき、MATLAB はオペレーション結果にも NaN を出力します。このようなデータの平均値を求めるために、新しい関数を作成します。

meannan.m

```
function [y,n] = meannan(x)
% Mean ignoring NaNs
xn=length(x);
idx = find(isnan(x));
x(idx) = [];
y = mean(x);
n = xn-length(idx);
```

この関数の 3 行目では、関数 find を使ってデータ内の NaN を探しますが、find(x==NaN) のように論理演算の中で NaN を使うことはできません。データ内の NaN を探すには関数 isnan を使い、その出力を関数 find に与えます。

4 行目では、値が NaN である要素を空行列にして、データ x から NaN を取除いています。

```
>> meannan(M)

ans =
   -8.3352e-05
```

### 4.3 データ補間

いくつかの指定した点を通る曲線を求めたりする場合、データ補間を行います。MATLAB ではデータ補間を行う関数として、1 次元データを扱う関数 interp1 と 2 次元データを扱う関数 interp2 があります。

**例題 4.3.1**

ロボットアームがホームポジションから動き始め、ある点で物を握り、握ったまま移動してある点で物を放して初期位置に戻ってくる動きを観測した (x,y) 座標データと、各座標点でのコードデータを points.mat として用意しています。

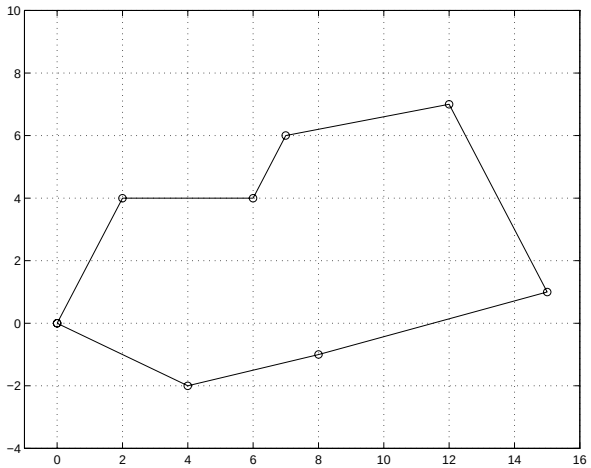
points.mat

| x | y | code | | コードの解釈 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 初期位置 |
| 2 | 4 | 1 | 1 | 経由点 |
| 6 | 4 | 1 | 2 | 物を握る位置 |
| 7 | 6 | 2 | 3 | 物を放す位置 |
| 12 | 7 | 1 | | |
| 15 | 1 | 3 | | |
| 8 | -1 | 1 | | |
| 4 | -2 | 1 | | |
| 0 | 0 | 0 | | |

ロボットアームは、各座標点間を直線的に動きます。アームの動きをスムーズにするために、スプライン補間を用いて滑らかな軌跡を求めてみます。このとき、全体を下記の 3 つの部分に区切って補間を行います。

PATH 1　初期位置～物を握る位置
PATH 2　物を握る位置～物を放す位置
PATH 3　物を放す位置～初期位置

```
>> load points                    % 1
>> plot(x, y, x, y, 'ro')         % 2
>> axis([-1 16 -4 10]), grid      % 3
```

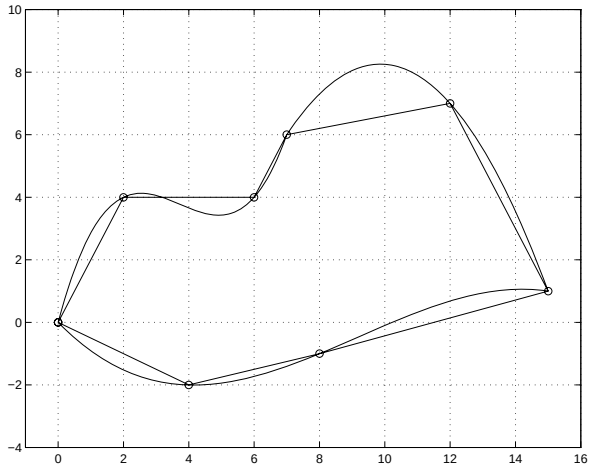


```
>> grasp = find(code == 2);                                    %  4
>> release = find(code == 3);                                  %  5
>> xend = length(x);                                           %  6
>> x1 = x(1:grasp);         y1 = y(1:grasp);                   %  7
>> x2 = x(grasp:release);   y2 = y(grasp:release);             %  8
>> x3 = x(release:xend);    y3 = y(release:xend);              %  9
>> t1 = x(1):0.1*sign(x(grasp)-x(1)):x(grasp);                 % 10
>> t2 = x(grasp):0.1*sign(x(release)-x(grasp)):x(release);     % 11
>> t3 = x(release):0.1*sign(x(xend)-x(release)):x(xend);       % 12
```

(% 4,5) では x,y のデータを 3 つの部分に区切るために、code が 2 と 3 を示す要素のインデックスを得ます。(% 7,8,9) では、x,y のデータを 3 つに区切って定義しています。

(% 10,11,12) では、補間後の x 方向のデータを定義しています。t1,t2,t3 がそれぞれ増加列かそれとも減少列なのかを判断し、刻み幅に的確な符号を与えるために関数 sign を使っています。

```
>> s1 = interp1(x1, y1, t1, 'spline');                          % 13
>> s2 = interp1(x2, y2, t2, 'spline');                          % 14
>> s3 = interp1(x3, y3, t3, 'spline');                          % 15
>> plot([t1 t2 t3], [s1 s2 s3], 'r', x, y, 'b', x, y, 'bo')     % 16
>> axis([-1 16 -4 10]), grid                                    % 17
```




(% 13,14,15) では、関数 interp1 を使ってそれぞれの区間で補間を行っています。ここではスプライン補間を行うので、関数の 4 つ目の引数で'spline' と指定します。関数 interp1 はこの他に、線形補間 (linear)、キュービック補間 (cubic) を行うことができます。なお、デフォルトでは線形補間を行います。関数 interp1 を使う場合、x 方向のデータは単調増加 (また減少) 列でなければならないので、注意してください。

画面上の図の赤色の曲線が、補間によって求めた曲線を示しています (青色は元の直線的な軌跡)。求めた軌跡は、逆運動学解析等にも用いられます。

---

コラム　データの丸め

MATLAB では、データの丸めを行う関数として、下記のものが用意されています。

| | 丸め関数 |
|---|---|
| ceil | 正の無限大方向への丸め |
| fix | ゼロ方向への丸め |
| floor | 負の無限大方向への丸め |
| round | 最も近い整数への丸め |

それぞれの関数を使って、データの丸めを行い、結果を比較します。

```
>> X  =  [2.3 -37.418 192.73 -5.62];
>> ceil(X)
```

```
ans =
        3   -37   193    -5

>> fix(X)
ans =
        2   -37   192    -5

>> floor(X)
ans =
        2   -38   192    -6

>> round(X)
ans =
        2   -37   193    -6
```

---

### 4.4 信号処理

MATLAB では、サンプリングされた信号をベクトルデータとして扱い、下記に示す関数を用いて信号処理を行うことができます。なお Signal Processing Toolbox では、更に高度な信号処理関数を提供しています。

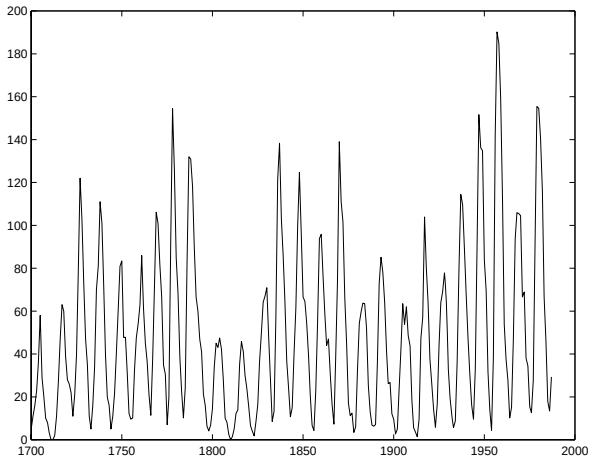
| | |
|---|---|
| abs | 複素数の大きさ（絶対値） |
| angle | 位相角 |
| fft | 高速フーリエ変換 |
| ifft | 逆高速フーリエ変換 |
| fft2 | 2 次元高速フーリエ変換 |
| ifft2 | 2 次元逆高速フーリエ変換 |
| filter | フィルタ操作 |
| conv | コンボリューション |
| decov | デコンボリューション |

**例題 4.4.1**

太陽の黒点の活動について、1700 年〜1987 年までの間、1 年毎に観測した Wolfer 数データが、sunspot.dat にあります。黒点の動きは、周期的であることが知られていますが、フーリエ変換を用いて、黒点の動きの近似周期を求めます。なお、データファイル sunspot.dat は MATLAB のデモファイルとして提供されています。

はじめにデータをロードして表示します。

```
>> load sunspot.dat                    %  1
>> year = sunspot(:,1);                %  2
>> wolfer = sunspot(:,2);              %  3
>> plot(year,wolfer)                   %  4
```

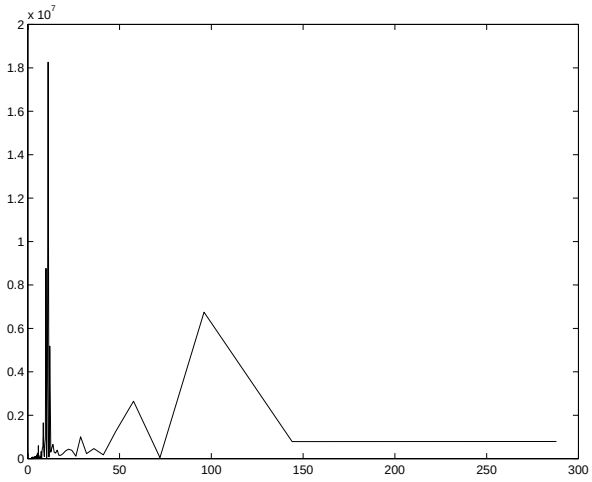


```
>> Y = fft(wolfer);                    %  5
>> n = length(Y);                      %  6
>> Y(1)=[];                            %  7
>> power = abs(Y(1:n/2)).^2;           %  8
>> nyquist = 1/2;                      %  9
>> freq = (1:n/2)/(n/2)*nyquist;       % 10
>> period = 1./freq;                   % 11
>> plot(period,power);                 % 12
```

関数 fft で Wolfer 数データをフーリエ変換しています (% 5)。フーリエ変換によって求めた値 Y は複素数値ですが、Y の 1 つ目の値はデータの和になっているので取り除きます (% 7)。Y は前半と後半でデータが折り返しになっているので、半分のデータを用いてパワーを求めます (% 8)。

ナイキスト周波数を 1/2(1/年) として周波数ベクトル freq を定義し (% 10)、その逆数をとって周期 period としています (% 11)。

(% 12) では、周期に対するパワーをプロットしています。約 10 年のところにピークがあることがわかりますが、実際に何年のところにピークがあるのか計算してみます。




```
>> index = find(power==max(power));   % 13
>> period(index)                      % 14
ans =
        11.0769
```

関数 find を使って、power の何番目の値で最大値となるか調べています (% 13)。index に対応する period の値が求める周期になり、太陽の黒点は約 11 年周期で活動することが分かりました。

### 4.5 宿題

1) 整数 n の階乗を求める関数を作りましょう。

2) ある速度を観測した下記のようなデータがあります。このデータを用いて何らかの処理をするために、0 秒〜60 秒後までのデータを 5 秒間隔に補間しましょう。

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 時間 | 0 | 10 | 16 | 24 | 31 | 38 | 44 | 50 | 53 | 57 | 63 |
| 速度 | 0 | 21 | 60 | 65 | 78 | 109 | 102 | 98 | 97 | 96 | 96 |

ヒント

1) ベクトルデータの累積積を求める関数 cumprod を使います。

2) データの補間を行う関数 interp1 を使います。

## 第 5 回 グラフィックス機能

### 5.1 高レベルグラフィックス関数

MATLAB でもっとも一般的に利用されているグラフィックス関数で、シンプルな命令で高度で複雑な描写を行います。ここまでに紹介した、2 次元プロット関数 plot などはこれに当たります。これらはある

程度のデフォルト値を持ち、また関数に与えたデータにある程度柔軟に対応するので、ユーザは最低限必要な命令を指定するだけで済みます。

| 2次元プロット関数 | | 3次元プロット関数 | |
|---|---|---|---|
| plot | 線形プロット | plot3 | 線形プロット |
| loglog | 両対数プロット | mesh | メッシュプロット |
| semilogx | x片対数プロット | surf | 表面プロット |
| semilogy | y片対数プロット | contour3 | コンタープロット |
| contour | コンタープロット | waterfall | waterfallプロット |
| polar | 極座標プロット | slice | 断面プロット |
| bar | 棒グラフ | | |
| errorbar | エラーバー表示 | | |
| hist | ヒストグラム表示 | | |

**例題 5.1.1**

plot関数を使って2次元データの視覚化をします。

```
>> x = 0:0.02:1;
>> plot(x, humps(x))
```

plot関数は2次元データの視覚化を行います。座標軸は、指定されたデータからオートスケーリングされますので、ユーザはデータを指定するだけで描写することができます。

複数データを同時にプロットさせるには、plot(x,y,s,x1,y1,s1)のようにx,yデータの組を並べます。また、データが行列として与えられた場合も複数データと判断し、列ごとのデータでプロットを行います。

また3番目の引数で、ラインタイプやマーカーの種類、色を指定するキャラクタ文字列を指定することができます。省略した場合は、実線で6色で色分けをして表示します。オプション引数には、下表のキャラクタ文字列が指定できます。

| 色の指定 | | 線・マークの指定 | |
|---|---|---|---|
| y | 黄 | . | 点 |
| m | 赤紫 | o | 円 |
| c | シアン | x | ×印 |
| r | 赤 | + | ＋印 |
| g | 緑 | * | ＊印 |
| b | 青 | - | 実線 |
| w | 白 | : | 点線 |
| k | 黒 | -. | 鎖線 |
| | | -- | 破線 |

キャラクタ文字列は、シングルクォートで囲んで指定します。また、色と線種を同時に指定することもできます。例えば、plot(x,y,'c+')はシアンの+印でプロットを行います。

---

コラム　グラフィックスの印刷

MATLABのグラフィックスをプリンタに出力するには、次の2種類があります。

- コマンドラインからの出力

  MATLABのプロンプトでprintコマンドを実行します。

  ```
  print [ -ddevice] [ -options ]
  ```

  出力タイプは -d オプションで指定します。デフォルトではPostScriptで出力されます。出力タイプの一覧はhelp printで見ることができます。複数ウィンドウが開いている場合は、カレントウィンドウがプリントされます。ウィンドウを指定する場合は、 -f オプションを利用します。

  ```
  print -f     % Figure window の場合
  ```

  また、ファイルに出力する場合は、出力ファイル名を指定します。

  ```
  print
  ```

- メニューからの出力
  Windows 版、または Macintosh 版 MATLAB ではメニューからの出力が行えます。この場合は OS のプリンタドライバを用いますので、任意にプリンタに出力することが出来ます。

---

### 5.2 装飾関数

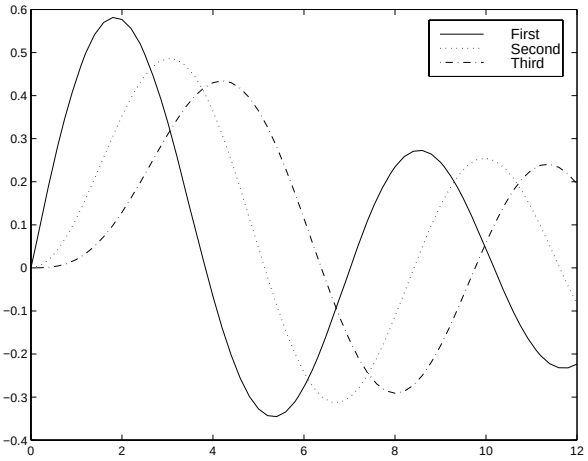
MATLAB でデータ解析を行った結果を用いて資料等を作成する場合、下記のような関数を用いてグラフの装飾を行うことができます。

| 装飾関数 | |
|---|---|
| title | タイトル表示 |
| xlabel(ylabel,zlabel) | 軸ラベル |
| grid | グリッドライン |
| text | 文字列の表示 |
| subplot | 分割プロット |
| legend | 凡例作成 |

**例題 5.2.1**

複数のデータをプロットし、関数 legend で各データに凡例を付けます。

```
>> x = 0:.2:12;
>> plot(x,bessel(1,x),x,bessel(2,x),':',x,bessel(3,x),'-.');
>> legend('First','Second','Third');
```




関数 legend は凡例を作成します。各ラインの文字列を引数として指定します。凡例ウィンドウの位置は、マウスで任意の位置に移動できます。また、オプション引数-1 を与えると、軸の外側に凡例を作成することもできます。

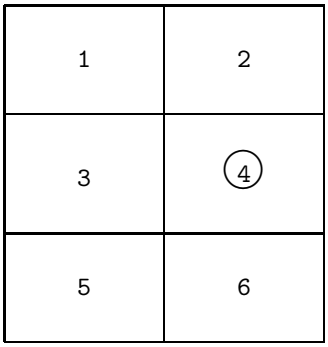
**例題 5.2.2**

subplot 関数でウィンドウを分割し、データの比較を行います。

```
>> t = 0:0.1:10;
>> subplot(2,1,1)
>> plot(t,sin(t)), grid
>> subplot(2,1,2)
>> plot(t,sin(2*t)), grid
```

subplot(m,n,p) はウィンドウを m × n に分割した p 番目の位置に座標軸を作成します。

例えば、subplot(3,2,4) はウィンドウを 3 × 2 分割して 4 番目の位置に描写します。デフォルトの座標軸に戻すには subplot(1,1,1) または clf と入力して下さい。




### 5.3 グラフィックスウィンドウの管理

MATLABのグラフィックスウィンドウは複数開くことができます。新しいウィンドウを開くには、figureコマンドを実行します。また、閉じる場合にはcloseコマンドを実行します。

| | ウィンドウ管理コマンド |
|---|---|
| figure | 新規ウインドウを作成 |
| close | ウィンドウを閉じる |
| clf | 画面上のオブジェクト消去 |

**例題 5.3.1**

複数のウィンドウを開いて、いくつかの3Dグラフィックス表現を比較してみます。

```
>> [X,Y] = meshgrid(-2:.2:2, -2:.2:2);
>> Z = X .* exp(-X.^2 - Y.^2);
>> surf(X,Y,Z)
>> figure
>> meshc(X,Y,Z)
```

MATLABでグラフィックス表示を行う場合、既にグラフィックスウィンドウが開かれているならそのウィンドウに新しい情報を表示します(既存の情報は失われます)。グラフィックスウィンドウに表示されている情報を残しておきたいときは、figureコマンドによって新しいウィンドウを開き、そこに新たな情報を表示します。

複数のウィンドウが存在する場合、どのウィンドウに描写するかは重要な問題です。ウィンドウは、ウィンドウのタイトルに表記されている番号で管理されます。MATLAB が現在認識しているウィンドウ(描写するウィンドウ)をカレントウィンドウと呼びます。複数枚ウィンドウが存在する場合、gcf(Get Current Figure)コマンドで出力される番号がカレントウィンドウの番号になります。

カレントウィンドウを変更するには、figureコマンドにfigure番号を引数として与えます。また、マウスでクリックしても変更できます。

### 5.4 オブジェクト指向のグラフィックス

MATLABでプロットしたグラフィックスに対して"ライン幅を変更したい"、"ラベルフォントを変更したい"などの要求がある場合、低レベルグラフィックス関数を使って、詳細な要望に柔軟に対処することが出来ます。MATLABの高度なグラフィックス機能を利用して、より詳細な設定を行うためには、まずMATLABのオブジェクト指向のグラフィックス構造を理解する必要があります。

MATLABでは、グラフィックスを構成する各単位(figure,axes,line,textなど)をオブジェクトと呼んでいます。

各オブジェクトは、それぞれプロパティと呼ばれる属性を持っています。例えば位置、色、大きさなどの値がプロパティとして設定されています。ユーザはこれらのプロパティ値を変更することによって、グラフィックスに対して細かい設定を行うことができます。

では、実際にグラフィックスオブジェクトを作成してみましょう。グラフィックスオブジェクトは、低レベルグラフィックス関数と呼ばれる関数(下表参照)によって作成されます。このとき、各プロパティの値を任意に設定して作成することができます。

低レベルグラフィックス関数
(Handle Graphics オブジェクト)

| | |
|---|---|
| figure | グラフィックスウィンドウ |
| axes | 座標軸 |
| line | ライン |
| text | テキスト |
| patch | パッチ |
| surface | サーフェス |
| image | イメージ |
| uicontrol | ユーザインタフェースコントロール |
| uimenu | ユーザインタフェースメニュー |

**例題 5.4.1**

ライン幅を指定して、正弦波を表す line オブジェクトを作成します。

```
>> t = 0:0.1:10;
>> line(t, sin(t), 'linewidth', 10)
```

プロパティ値を指定してオブジェクトを作成するには、設定するプロパティ名とその値を引数として与えます。

MATLAB のグラフィックスオブジェクトは、上記のような低レベルグラフィックス関数を基に作成されます。低レベルグラフィックス関数では、様々なプロパティ値を指定することができるので、詳細な設定を行うことができます。また、5.1 節で取り上げた高レベルグラフィックス関数は、低レベルグラフィックス関数を組み合わせて作成されています。それぞれのプロパティを、使いやすいようにあらかじめ設定してあります。単なるデータ表示には高レベル関数が便利なのに対して、低レベル関数は細かな指定をしたグラフィックスオブジェクトを作成するときに便利です。

---

コラム　オブジェクトの階層構造

すべてのオブジェクトは階層構造を組んで管理されています。階層の上下関係は parent と children プロパティで示されます。

---

## 5.5 オブジェクトの管理

前節により、オブジェクト作成時に任意にプロパティ値を指定できることが分かりましたが、今度は作成後に設定されているプロパティ値を取得したり、変更する方法について考えます。

プロパティ値について操作を行うには、どのオブジェクトに対する設定なのか認識していなければなりません。各オブジェクトは、他と識別するための認識番号 (handle) を持っています。MATLAB は、このハンドルでオブジェクトを管理しています。root(Computer Screen) にはゼロ、figure には整数、text,axes,line などには実数が与えられます。

ハンドルの取得方法

オブジェクトのハンドルを取得するには、以下のような方法があります。

1. オブジェクト作成時に取得する

```
>> t = 0:0.1:10;
>> h = plot(t,sin(t))
h =

   57.0001
```

plot,mesh 等グラフィック表示を行う関数を実行する際に指定した出力引数が、作成されたオブジェクト (plot の場合は line オブジェクト、mesh の場合は surface オブジェクト) のハンドルになります。

2. カレントオブジェクトのハンドルを取得する

```
>> t = 0:0.1:10;
```

```
>> plot(t,sin(t))
>> h = gca
h =

   58.0002

>> h1 = gco   % マウスで line オブジェクトをクリックしてから実行する
h =

   59.0002
```

gca コマンド (Get Current Axes) は、カレント軸のハンドルを取得します。また gco コマンド (Get Current Object) は、カレントオブジェクトのハンドルを取得します。オブジェクトの選択は、マウス操作によって行うことができます。

3. 階層構造を利用する

```
>> t = 0:0.1:10;
>> plot(t,sin(t))
>> h = get(gca,'children')
h =

   61.0004
```

オブジェクトの階層構造 (親子関係) を表すプロパティparent,children を利用してハンドルを取得します。get コマンドについては下記を参照してください。

MATLAB のグラフィックスオブジェクトは、ハンドルによって管理されているので、オブジェクト作成後にそのプロパティ値を自由に変更することができます。このように、オブジェクトのプロパティを操作する機能のことを、Handle GraphicsTM と呼んでいます。

各オブジェクトに設定されているプロパティ値を取得するには get コマンドを使います。また、プロパティ値を変更するには set コマンドを使います。

**例題 5.5.1**

ある正弦波グラフのオブジェクトを用いて、Handle Graphics 機能の操作方法を示します。

まず正弦波をプロットし、各オブジェクトのハンドル番号を取得します。

```
>> x = 0:0.1:10;  y = sin(x);
>> plot(x,y)
>> title('Sine Wave')
>> h = gcf;
>> h1 = gca;
>> h2 = get(h1,'children');
```

get コマンドを使って、axes オブジェクトに設定されたプロパティ値を表示します。プロパティ名を指定せずに get を実行したときは、そのオブジェクトに設定されているプロパティ値の一覧を得ることができます。

```
>> get(h1)
        AspectRatio = [NaN NaN]
        Box = on
        CLim = [0 1]
        CLimMode = auto
        Color = none
        CurrentPoint = [ (2 by 3) ]
        ColorOrder = [ (6 by 3) ]
        DrawMode = normal
        FontAngle = normal
        FontName = Helvetica
```

```
        FontSize = [12]
                :
                :
```

つぎに、axes オブジェクトの位置・大きさを変更します。これらを設定するのは、position プロパティです。position プロパティは、4 つの値をベクトルで与えます。値の単位は units プロパティによって決まります。ここでは normalized に設定されているので、Figure の大きさを 1 としたときの正規化された値を与えています。

そして、タイトルを日本語表示にします。この場合には、title のフォントに日本語に対応したものを設定します。なお、UNIX 版では日本語表示はできません。

```
>> set(h1, 'position', [0.1 0.1 0.5 0.6])
>> h3 = get(h1, 'title');
>> set(h3, 'string',' 正弦波', 'fontname',' ＭＳ ゴシック')
```

### 5.6 グラフィカルユーザインタフェース (GUI)

GUI(Graphical User Interface) は、Handre Graphics を利用して作成されたグラフィカルな作業環境をユーザに提供します。GUI を使うとメニューやボタンによって MATLAB の関数を容易に実行できます。この節では、このような GUI ツールの作成方法について学びます。

GUI ツールを構成するオブジェクトは、uicontrol,uimenu の 2 つの低レベルグラフィックス関数によって作成されます。

**例題 5.6.1**

カラーを変化させるボタンを作成します。

```
>> mesh(peaks);
>> h = uicontrol;
>> set(h,'string','color')
>> set(h,'callback','colormap(hot)')
```

この例題では、3 次元プロットのカラーを変更するプッシュボタンを作成しています。ボタンを作成するには、uicontrol 関数を使って作成します。ボタンが押されたときに実行するコマンドは、callback プロパティで与えます。ここでは、Figure に与えられた色の分布図 (colormap) を hot というマップに変更する設定をしています。デフォルトで hsv というマップが与えられていますが、プッシュボタンをクリックすることによってマップが変更されます。

uicontrol 関数は、style プロパティを変更することによってプッシュボタン以外にもラジオボタンやエディタブルテキスト、スライダーバーなどを作成することができます。

**例題 5.6.2**

GUI を利用したグラフ描写ツールを作成します。

myplot.m (Web ページからダウンロード可能)

このツールは下記の機能を持っています。

1. xdata,ydata を入力するエディタブルテキスト
2. プロットの種類 (plot,semilogx,semilogy,loglog,bar,polar) を選択するポップアップメニュー
3. グリッドラインを表示/非表示させるプッシュボタン
4. xlabel,ylabel,title を入力するエディタブルテキスト
5. 軸の範囲を指定するエディタブルテキスト

## 宿題の解答例

### 第 1 回

1-1)

```
M1=reshape(0.1:0.1:0.6,3,2)'
```

関数 reshape は、行列の変形を行いますが、reshape(0.1:0.1:0.6,2,3) と実行してしまうと、要素の並びが異なってしまいます。MATLAB では、列方向にインデックスを付けていくので、3 行 2 列に変形した後で、その行列を転置させます。

1-2)

```
M2(2:4,4:6)=diag(1:3); M2(4,4)=4
```

2〜4 行目の 4〜6 列目の対角要素は、関数 diag で定義します。このとき、指定しなかった要素には、自動的に 0 が入ります。また 4 行 4 列目の要素は、この後で 4 に入れ替えます。

1-3)

```
M3=(2:2:10)'; M3=M3(:,ones(1,4))
```

行列の各列は、2 から 10 まで 2 刻みで増加する数列になってます。まずこの列ベクトルを定義して、関数 ones を使って 4 列分コピーします。

2)

```
A=[2 1 5;2 2 3;1 3 3];
b=[5;7;6];
x=inv(A)*b;
disp(['Answer : x=' num2str(x(1)) ', y=' num2str(x(2)) ', z='
num2str(x(3))])
```

連立方程式は、逆行列を用いて解きます。その解を画面上に表示するには、関数 disp を使います。すべて文字列として表示するために、関数 num2str を使って、求めた解の数値を文字列に変換します。

3)

```
a=1; b=6; c=10;
x1=(-b+sqrt(b^2-4*a*c))/2*a
x2=(-b-sqrt(b^2-4*a*c))/2*a
```

二次方程式の解の公式にそのままあてはめます。平方根の計算には、関数 sqrt を使います。

### 第 2 回

1)

fib.m

```
function F = fib(n)
% FIBFUN for calculating Fibonacci numbers.
% Incidentally, the name Fibonacci comes from
% Fillius Bonassi, or "son of Bonassus".

F = [];
for i = 1:n
    f = fibfun(i);
    F = [F, f];
end
```

出力引数 F は、はじめに空行列として定義します。for ループ内では、初項から n 項目までのフィボナッチ数を f としてそれぞれ求め、その度 F の要素に追加してゆきます。

2)

prime.m

```
function [p, x] = prime(n)
% Find prime numbers.
% p : a prime number vector
% x : a number of p

s = ones(1, n);

for i = 2:sqrt(n)
    if s(i) == 1
        s(i^2:i:n) = 0 * (i^2:i:n);
    end
end

p = find(s);
x = sum(s);
```

1 から n までの整数について、素数なら 1(真)、そうでなければ 0(偽) を与えて判別してゆきます。初めはすべての整数に 1 を与えておきます。関数内の for ループでは、2 から n の平方根までの整数の中で 1 となっているものを探し、その整数の倍数を 0 に変更します。その結果、1 が与えられている整数が p として出力されます。また、x として 1 から n までの素数の数も出力します。

**第 3 回**

1)

```
load hw31.dat
C=corrcoef(hw31(:,1),hw31(:,2));
fid=fopen('hw31.dat','a');
fprintf(fid,'%f %f\r\n',C);
fclose(fid);
```

ダウンロードしたアスキーファイル hw31.dat から、データを読み込みます。そして、関数 corrcoef によって相関係数を求めます。求めた相関係数を元のファイルの最後に追加するために、hw31.dat ファイルを追加モード 'a' で開きます。

2)

```
fid=fopen('hw32.dat');
for i=1:3
  str=fgetl(fid);
  idx=findstr(str,':');
  if i==1
    dname=['d' str(idx+1:length(str))];
  else
    msize(i-1)=str2num(str(idx+1:length(str)));
  end
end
eval([dname '=fscanf(fid,''%f'',msize);']);
fclose(fid);
```

アスキーファイル hw32.dat の 3 行のヘッダー情報から、読み込むデータの変数名と大きさを指定します。読み込む変数名は、dname によって文字列として定義します。また、データの大きさは、$1 \times 2$ の行列 msize で数値データとして定義します。for ループ内の演算で定義された変数名と大きさを指定してデータを読み込むには、eval コマンド使います。

**第 4 回**

1)

```
function b = kai(N)
% factorial N!
if N <0 | rem(n, 1) | size(n, 1) ~="1" | size(n,2) ~="1   <BR">
error('N must be a natural number.')
elseif N == 0
  b = 1;
else
  prog = 1:N;
  x = cumprod(prog);
  b = x(N);
end
```

入力値が負の値、非整数値、スカラーではない値のときには、エラーメッセージを出力します。また入力値が 0 のときは、1 を出力するよう elseif 文で場合分けしています。

自然数が入力されたときは、関数 cumprod で 1 から自然数 N までの累積積を求め、その最後の値を出力します。

2)

```
t = [0 10 16 24 31 38 44 50 53 57 63];
v = [0 21 60 65 78 109 102 98 97 96 96];
t1 = 0:5:60;
v1 = interp1(t, v, t1, 'linear');
plot(t, v, 'o', t1, v1)
```

はじめに時間と速度のデータを変数 t,f として定義します。5 秒間隔のデータを得るために、時間データ t1 を 0 から 60 まで 5 刻みに増加する列として定義します。

関数 interp1 によって、5 秒間隔に補間データを求めます。上記例では線形補間を行っていますが、キュービック補間を行うにはオプション'cubic'、スプライン補間を行うにはオプション'spline' を指定します。

（参考）UNIX コマンドと mule（emacs）の操作

講習会で必要となる最小限の UNIX コマンドと mule（emacs）の操作法をあげておきます.

UNIX コマンド

- ls: ファイルリストの表示

  ls　　カレントディレクトリにあるファイルの表示
  ls /tmp　/tmp というディレクトリにあるファイルの表示

- cp: ファイルのコピー

  cp foo hoge　　foo を hoge という名前のファイルとしてコピー
  cp /tmp/hoge.dat .　/tmp/hoge.dat というファイルをカレントディレクトリにコピー
  cp /tmp/*.m .　/tmp というディレクトリにある m という拡張子のついたファイルをすべてカレントディレクトリにコピー

- less: ファイルの内容の確認

  less hoge　hoge というファイルの内容を見る
  （1 画面に収まらないときは，スペースで前進，b で後退できる）

mule（emacs）の操作法

- 移動（おそらくカーソルも使える）

  ctrl+n　下
  ctrl+p　上
  ctrl+f　左（前）
  ctrl+b　右（後）
  ctrl+a　行頭へ
  ctrl+e　行末へ

- マーク

  ctrl+スペース　マーク
  マークされた位置と（移動後の）カーソルとの間が指定範囲となる

- 削除

  ctrl+d　一字削除
  ctrl+k　行末まで削除（削除部分を記憶している）
  行頭で行えば一行削除
  ctrl+w　マークによる指定範囲を削除（削除部分を記憶している）

- ペースト

  ctrl+y　記憶部分を貼りつけ

- ファイル関係

  ctrl+x ctrl+f　ファイルの呼出
  ctrl+x ctrl+s　ファイルの保存
  ctrl+x ctrl+w　名前をつけて保存
  ctrl+x ctrl+c　終了