# データ解析・技術計算言語 MATLAB の活用法

総合情報基盤センター　教授　高井正三

MATLAB（「MATrix LABoratory」の略で，『マットラブ』と発音）は，科学技術計算，可視化およびプログラミングのための言語と対話型環境を提供し，データ解析，アルゴリズム開発，各種数式モデルやアプリケーション開発，更には最近の Big Data 解析にも活用されています．本稿では，基本的な使い方と計算結果を可視化するプログラミング方法を中心に解説していきます．

MATLAB は，その名のとおり Matrix（行列）を基本的なデータとして扱うプログラミング言語で，ベクトルや行列で定式化した問題を解くのに適しています．従って，この言語を習得するには行列計算に関する基礎的な知識が必要です．一般的に言えることですが，習得に時間と労力をかけたプログラミング言語は，コードを書くのが容易で，貴方の最も強力で便利な「プログラミング言語」となることでしょう．本稿では，1）MATLAB で何ができるか，2）MATLAB の起動，終了，対話型プログラミングの操作，3）MATLAB とのデータの入出力方法，4）プログラム・ファイル M-Files の編集，デバッグ方法，5）ベクトル，行列，線形代数の計算方法，6）多項式計算と補間，7）MATLAB 関数，8）データ解析と統計処理，そして，9）データをグラフ化，可視化する方法を解説します．最後に演習問題を用意しましたので，各自挑戦して解いてみて下さい．

では，始めましょう！「習うより，慣れろ！」です．Getting Started! Practice Makes Perfect!

## 1．MATLAB で何ができるか

### 1.1　MATLAB とは

MATLAB は，表題に示すとおり，データ解析・科学技術のための数値計算と，データ解析し，グラフ化するなどの可視化ツールを提供し，各種モデルや様々なアプリケーション・システム，アルゴリズムを開発するための高水準プログラミング言語と，その対話型開発環境を提供しています．

### 1.2　新しいアイデアを探す

Mathworks 社によれば，MATLAB は，信号処理，通信，画像処理，ビデオ処理，制御システム，テストおよび測定，金融工学，情報生命科学など，幅広い分野での利用が可能で，アイデアを探して可視化したり，共同作業を行ったりできます．

### 1.3　アイデアを具体化する

さらに，MATLAB を活用できるプロジェクトには，例えば，スマート電力網を構築するためのエネルギー消費のモデル化，超音速車両のための制御アルゴリズムの開発，ハリケーンの進路と強さを可視化するための気象データの解析，抗生物質の最適な使用量を同定するための数百万のシミュレーションの実行など，があるといいます．

### 1.4　MATLAB 環境の準備

本学では，総合情報基盤センターから MATLAB R2014b の Floating License 版が希望者に配布されていますので，インストール用のライセンス・キーとネットワーク・ライセンス・キーを入手して，各自の PC にコピーしてから，MATLAB システムをインストールして下さい．既に，MATLAB R2015b 版がリリースされていますが，本学では対応が遅れているようです．

では，数値計算，データの可視化，プログラミングとアルゴリズム開発，アプリケーション開発と配布，等の方法を，MATLAB プログラミングを体験しながら具体的にみていきましょう．

## 2．MATLAB の起動と終了，プログラミング

### 2.1　MATLAB の起動

Windows 10 PC 上で MATLAB を起動するには，(1) スタート・ボタンから「すべてのアプリ」をクリックして，メニューMATLAB をクリックするか，(2) スタート・ボタンから MATLAB タイルをクリックするか，(3) Desk Top 画面で MATLAB アイコンをダブル・クリックします（図 2.1）．

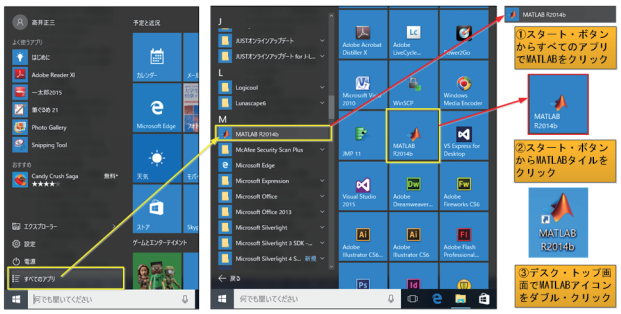



図 2.1　MATLAB を起動する 3 つの方法

起動時に MATLAB システムは，ライセンス・サーバに対して，起動するためのネットワーク・ライセンスを取得しにいきますので，起動に多少の時間がかか

ります．ライセンスは Floating License で，総合情報基盤センターで取得している Floating License 数は 50 個です．従って，50 人を超える利用者が同時に MATLAB を起動することはできません．

しばらくすると，図 2.2 のような MATLAB のロゴが表示され，ライセンス認証を経て，図 2.3 のような MATLAB デスクトップ（初期画面）が現れますので，コマンド・ウィンドウのプロンプトに，コマンドや数式，MATLAB 関数，スクリプトなどを入力し，対話しながら作業を進めていくことができます．

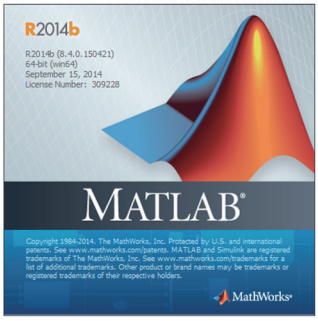



図 2.2　MATLAB のロゴ表示

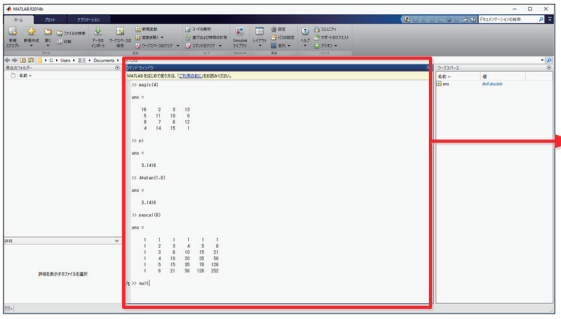


図 2.3　MATLAB のデスクトップ画面

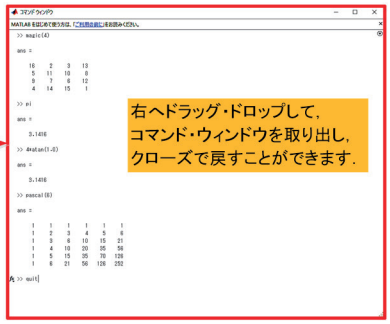



図 2.4　取り出したコマンド・ウィンドウ画面

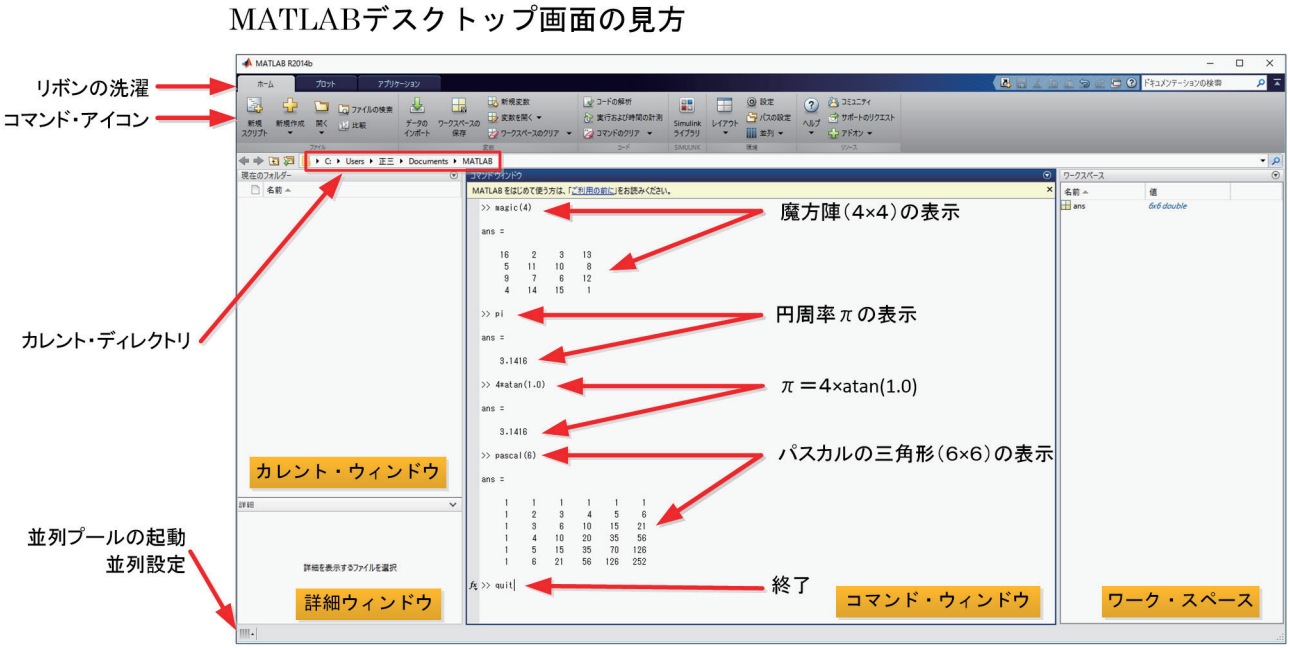



図 2.5　MATLAB のデスク・トップ画面

## 2.2　MATLAB コマンドの入力

では，以下の様にコマンドを入力してみましょう．

```
>> magic (4)
```

とタイプし，[Enter] キーを押します．magic (4) は 4×4 の魔方陣を生成する関数です．

```
>> pi
```

とタイプし，［Enter］キーを押します．pi は円周率を表します．

```
>> 4*atan (1.0)
```

とタイプし，[Enter] キーを押します．ArcTan (1.0) × 4 ＝円周率を計算し，表示します．

```
>> pascal (6)
```

とタイプし，［Enter］キーを押します．pascal (6) は 6×6 の Pascal 行列を生成します．Pascal 行列はパスカルの三角形から得られる整数要素をもつ対称な正定行列です．

## 2.3　MATLAB の終了

MATLAB を終了するには，「quit」コマンドを

```
>> quit
```

とタイプし，［Enter］キーを押します．直ちに MATLAB を終了します．

## 2.4　MATLAB プログラミングの方法

MATLAB のプログラミングは対話形式で行います．

### （1）MATLAB デスクトップ

MATLAB を起動する場合，MATLAB デスクトップは，図 2.5 に示すように現れます．利用者は，デス

クトップの外観を，レイアウト・ボタン・メニューからコマンド履歴ウィンドウを追加するなど（図 2.7）自由に変更することができます．図 2.4 はコマンド・ウィンドウをデスクトップから，ドラッグ・ドロップして，取り出した例です．

**（2）カレント・ディレクトリの変更**

カレント・ディレクトリを変更する場合は，予め「D:¥MATLAB_Work」の様な作業用フォルダーを作成し，フォルダー参照アイコンをクリックして「新規フォルダー選択」ダイアログを表示し，図 2.6 の様に変更します．

各自で，自分専用の使用環境を整え，diary ファイルや M-File の保存場所を確保しましょう．

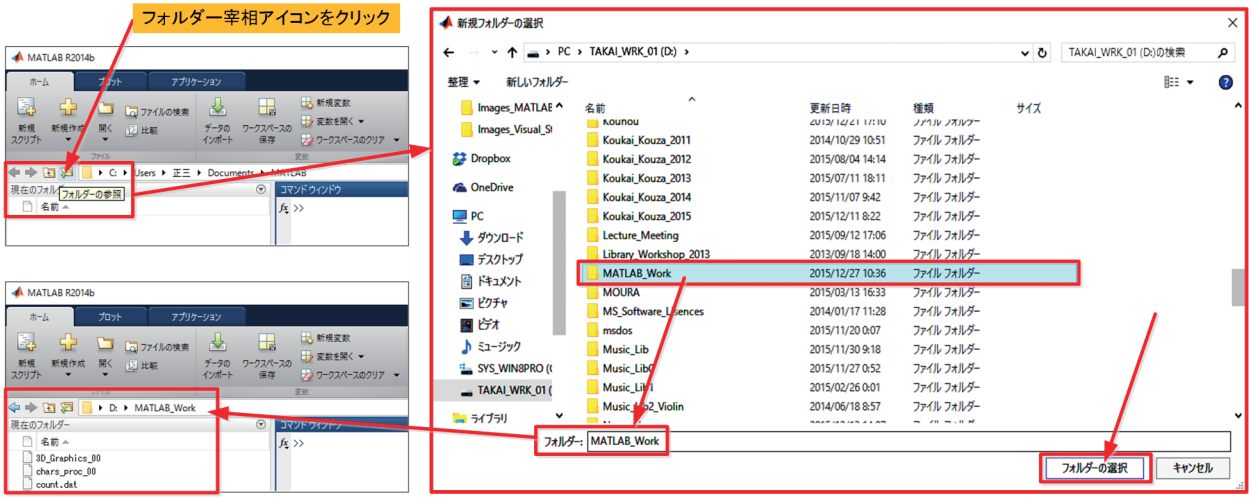



図 2.6　MATLAB のカレント・ディレクトリの変更

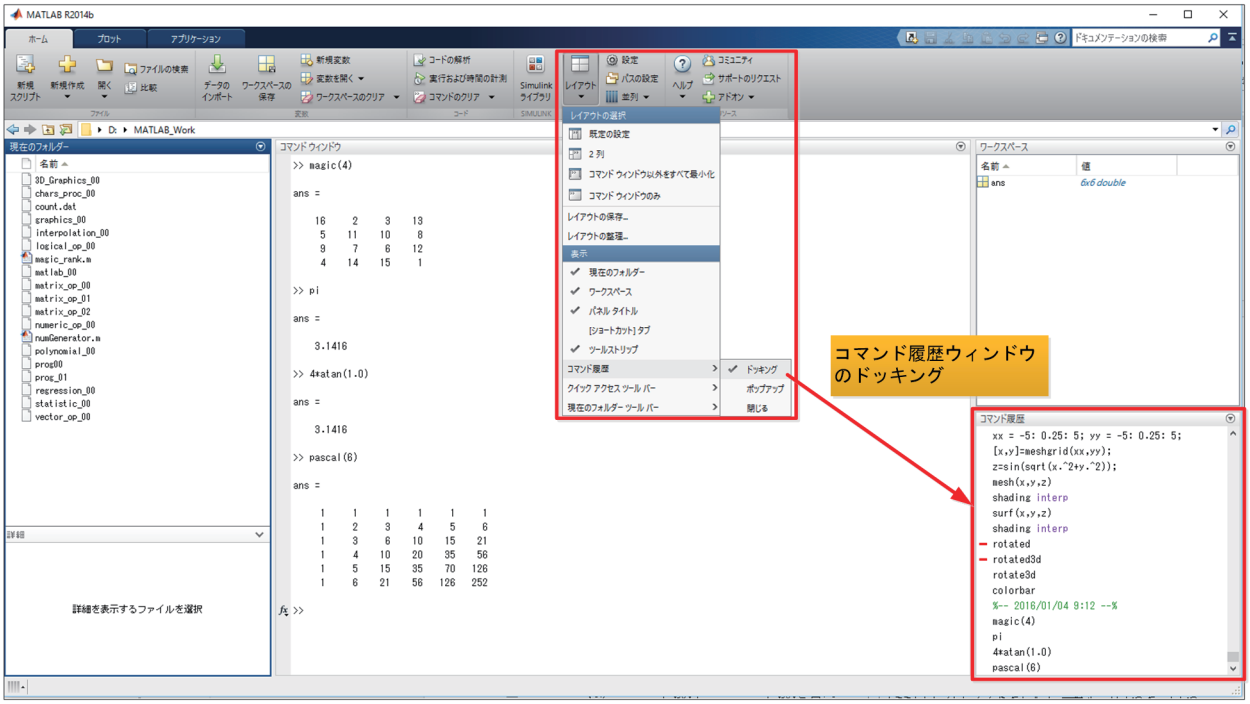



図 2.7　MATLAB デスクトップにコマンド履歴ウィドウをドッキング

**（3）対話型計算履歴を採る diary コマンド**

**>> diary　filename**

（filename：履歴を保存するファイル名）

とタイプすると，カレント・ディレクトリ上に，指定したファイル名でキー・インの記録を開始します．

**>> diary　off**

とタイプすると，キー・インの履歴をカレント・ディレクトリ上に，指定のファイル名で保存します．

**（4）式**

MATLAB は数学的な式を使いますが，他のプログラム言語と異なり，これらの式はすべての行列を含んでいます．式は，「変数 ，数字，演算子，関数」から

構成されています.

**1）変数**

MATLABは，タイプの宣言や次元を宣言するステートメントを必要としていません．MATLABは新しい変数名を使おうとするとき，自動的に変数を作成し，適切な大きさのストレージを割当てます．変数がすでに存在していると，MATLABは必要なら，その内容を変更し，新しいストレージを割当てます．例えば，

```
num_students = 25
```

は，1行1列のnum_studentsと名付けた行列を作成し，その単一要素に値25をストアします．

変数名は，一つの文字を先頭に，その後にいくつかの文字，数字やアンダースコアを続けて表わされます．MATLABは，1つの変数名として，最初の31文字のみを使います．MATLABは，大文字，小文字の区別を行います．すなわち，Aとaは，同じ変数ではありません．ある変数に割当てられた行列を表示するには，単に変数名を入力してください．

**2）数字**

MATLABは，通常の小数点表示を行います．これは，小数点をもち，数字の先頭にプラスまたはマイナスの符号を付けます．科学的な記法として，10のべき乗のスケール・ファクタを設定するeを使います．虚数は，iまたはjをサフィックスとして使います．正しく表現された数字の例を次に示します．

```
3          -99          0.0001
9.6397238  1.60210e-20  6.02252e23
1i         3.14159j     3e5i
```

すべての数字は，IEEE浮動小数点標準で設定されるlong書式を使って内部的にはストアされます．浮動小数点数は，概ね16桁の数字の有限精度をもち，1.0e−308から1.0e+308の有限な範囲に入ります．

3）演算子

式には，馴染みの深い代数演算子と優先順位則を使います．

| | | | |
|---|---|---|---|
| + | 和 | a+b | a+3 |
| − | 差 | a-b | a-3 |
| .* | 乗算 | a.*b | a.*3 |
| ./ | 除算 | a./b | a./3 |
| .¥ | 左除算 | a.¥b | a.¥3 |

（円記号¥は＼（バック・スラッシュ）の日本語表記）

| | | |
|---|---|---|
| .^ | べき乗 | c.^3 |
| * | 行列の乗算 | A*B |
| / | 行列の除算 | A/B |
| .' | 転置 | B.' |
| ¥ | 行列の左除算 | B¥A equal A/B |

（円記号¥は＼（バック・スラッシュ）の日本語表記）

| | | |
|---|---|---|
| ^ | 行列のべき乗 | A^3 |
| ' | 和複素共役転置 | B' |
| () | 演算順序の設定 | (a-b).*3 |

4）関数

MATLABは，abs，sqrt，exp，sin等を含む多くの標準的な初等数学関数を用意しています．負の数の平方根や対数は，エラーにはなりません．適切な複素数で表される結果が自動的に出力されます．

MATLABは，Bessel関数やGamma関数を含んだ，よりアドバンスドな数学関数も用意しています．これらの関数のほとんどは，複素数と共にも使えます．

初等数学関数の一覧を得るには，

```
>> help elfun
```

とタイプします．そして，よりアドバンスドな数学関数や行列関数の一覧を得るのは，

```
>> help specfun
>> help elmat
```

とタイプインしてください．例えば，sqrt, sin等のいくつかの関数は組込み関数になっています．これらは，MATLABのコアの一部で，非常に効率よく作られていますが，計算の詳細を見ることはできません．gamma, sinh等の他の関数は，M-ファイルとして実行されます．これらは，ユーザが内容を見ることができ，必要なら，内容を変更することができます．

いくつかの特別な関数は，利用可能な定数値を用意しています．

| | |
|---|---|
| pi | 3.14159265.... |
| ans | 直前に実行したコマンドの答え |
| i | 虚数単位 |
| j | 虚軸単位 |
| eps | 浮動小数点相対精度，IEEE浮動小数点数形式で，2^(−52) ≒ 2.22e-16 |
| realmin | 最も小さな浮動小数点数，IEEE浮動小数点数形式で，2^(−1022) ≒ 2.2251e-308 |
| realmax | 最も大きな浮動小数点数，IEEE浮動小数点数形式で，2^(1024) ≒1.7977e+308 |
| inf | 無限大 |
| NaN | Not-a-Number不定値 |

無限大は，ゼロでない数をゼロで割ること，または，オーバフローを巧く定義する数学的表現，すなわちrealmaxを超える数を定義する表現として使います．Not-a-Numberは，0/0またはinf-infのような巧く定

義できない数学的な値を計算するときに作成されます.

**（5）コメント行**

MATLABのコメント文は,%を用いて記述します.

```
>> kekka          % result of computation
```

## 3．MATLABの基本的な演算

### 3.1 数値演算

コマンド・ウィンドウ上で通常の数値定数による演算を実行してみましょう（図3.1~2）.

「>>」はプロンプトといい，一般に「けっと，けっと（cket cket）」と発音する入力促進記号です．では，各自で以下の演算にトライしてみて下さい.

（注:実際の画面表示では１行毎に空白行が挿入されますが,以下の記述では空行を非表示にしています.）

```
>> 456 + 123
ans =
   579          <==== 加算

>> 456 - 123, 456 .* 123
ans =
   333          <==== 減算
ans =
       56088    <==== 乗算

>> 456 ./ 123, 123 .¥ 456
ans =

    3.7073      <==== 普通の除算
ans =
    3.7073      <==== 右の数値を左で除算

>> 2.^10
ans =
        1024    <==== K(Kiro)
>> 2.^20
ans =
     1048576    <==== M(Mega)
>> 2.^30
ans =
  1.0737e+009   <==== G(Giga)
>> 2.^40
ans =
  1.0995e+012   <==== T(Tera)

>> realmax, realmin
ans =
  1.7977e+308       <==== 最大実数
ans =
  2.2251e-308       <==== 最小実数
>> 2.^ 10000
ans =
   Inf      <==== 無限大（infinity）
>> 2.^5000 - 2.^10000
ans =
   NaN      <==== Not a Number

>> 1:5  <==== ベクトル［1 2 3 4 5］発生
ans =
     1     2     3     4     5
>> prod(1:5)
ans =
   120  <= ベクトル［1 2 3 4 5］の要素の積

>> 5 - 2.*i
ans =
   5.0000 - 2.0000i
```

図3.1 MATLABを使っての数値演算例（1）

```
>> x = 10 + 6.*i; y = 5 - 2.*i;
>> z1 = x + y, z2 = x - y
z1 =
  15.0000 + 4.0000i
z2 =
   5.0000 + 8.0000i

>> z3 = x .* y, z4 = x ./ y
z3 =
  62.0000 +10.0000i
z4 =
   1.3103 + 1.7241i

>> sqrt(-1), sqrt(3.0)
ans =
       0 + 1.0000i    <== 虚数
ans =
    1.73211
```

図3.2 MATLABを使っての数値演算例（2）

### 3.2 論理演算

論理演算子と論理関数は次の通りです（表3.3）．使用例は図3.4の通りです.

表3.3 MATLABの論理演算子と論理関数

| 論理演算子 | 演算 | 論理関数 | 関数の説明 |
|---|---|---|---|
| & | AND | XOR（a,b） | 排他的論理和<br>XOR（1,1）→0,<br>XOR（1,0）→1,<br>XOR（0,1）→1,<br>XOR（0,0）→0 |
| \| | OR | all（A） | 引数の要素のすべてが真またはゼロでないなら１を返す |
| ~ | NOT | any（A） | 引数の要素のどれか1つが真またはゼロでないなら１を返す |

```
>> u = [1 0 2 3 0 5] ;
>> v = [5 6 1 0 0 7] ;
>> u, v
ans =
   1   0   2   3   0   5
ans =
   5   6   1   0   0   7

>> u & v
ans =
   1   0   1   0   0   1

>> u | v
ans =
  1   1   1   1   0   1
```

```
>> ~u
ans =
   0   1   0   0   1   0

>> a = 1; b = 1; xor (a,b)
ans =
   0

>> A = [0 1 2;3 5 0]
A =
   0   1   2
   3   5   0
>> all (A)
ans =
   0   1   0

>> v = [5 0 8] ;
>> any (v)
ans =
   1
```

図 3.4　MATLAB を使っての論理演算と論理関数の例

### 3.3　MATLAB の基本ベクトル演算

MATLAB の演算対象データ形式はベクトルなどの配列です．ここでは 2 次方程式

$$ax^2 + bx + c = 0$$

を解の公式：

$$x1, x2 = (-b \pm \sqrt{b^2 - 4ac}) / 2a$$

を用いて解くプログラムを実行してみましょう（図 3.5）．

```
>> A=1.0; B=-3.0; C=4.0;

>> X1=(-B+(B.^2-4.*A.*C).^0.5)./(2.*A)
X1 =
   1.5000 + 1.3229i

>> X2=(-B-(B.^2-4.*A.*C).^0.5)./(2.*A)
X2 =
   1.5000 - 1.3229i
>> P= [1.0 -3.0 4.0] ;

>> R=roots (P)  %<=多項式の根を求める関数
R =
   1.5000 + 1.3229i
   1.5000 - 1.3229i
```

図 3.5　MATLAB を使っての 2 次方程式を解く

続いて，3 組の 2 次方程式を同時に解いてみましょう（図 3.6）．

$$x^2 - 3x + 4 = 0$$
$$3.5x^2 + 5.7x + 8 = 0$$
$$5x^2 - 5x + 8 = 0$$

```
>> A = [1 3.5 5]; B = [-3 5.7 -5]; C = [4 8 8];  %<=== 3組の係数をベクトルにして入力

>> X1 = (-B + (B .^2 - 4 .* A .* C) .^ 0.5) ./ (2 .* A)

X1 =
   1.5000 + 1.3229i  -0.8143 + 1.2738i   0.5000 + 1.1619i

>> X2 = (-B - (B .^2 - 4 .* A .* C) .^ 0.5) ./ (2 .* A)

X2 =
   1.5000 - 1.3229i  -0.8143 - 1.2738i   0.5000 - 1.1619i
```

図 3.6　MATLAB を使っての 3 組の 2 次方程式を解く

ここで，**2.*A** は 2 と A の各成分の積を成分とするベクトル［2*1　2*3.5　2*5］を与えます．一般に $A = [a_1 a_2 \cdots a_n]$ のとき，c.* $A$ は $[ca_1 ca_2 \cdots ca_n]$ という行ベクトルを与えます．

表 3.7　MATLAB 演算子の演算における優先順位

| 順位 | 演算子 |
|---|---|
| 1 | かっこ（） |
| 2 | 転置（.'），べき乗（.^），複素共役転置（'），行列べき乗（^） |
| 3 | 単項プラス（+），単項マイナス（-），論理否定（~） |
| 4 | 乗算（.*），除算（./），左除算（.¥），行列乗算（*），行列除算（/），行列左除算（¥） |
| 5 | 加算（+），減算（−） |
| 6 | コロン演算子（:） |
| 7 | 関係演算子 < <= > >= == ~= |
| 8 | 論理積（&） |
| 9 | 論理和（\|） |

## 3.5　MATLAB の行列演算

MATLAB は行列の演算を最も得意としています．

（1）生成

MATLAB では関数を使うことにより，異なった種類の行列を生成することができます（図 3.8）．

```
>> A=pascal (3)   <=pascal 行列 (対称行列)
A =
     1     1     1
     1     2     3
     1     3     6
>> B=magic (3)   <= 魔方陣による非対称行列
B =
     8     1     6
     3     5     7
     4     9     2
>> C=fix (10*rand (3,2))   <==== [0,1] の一様乱数を 10 倍し，3×2 の長方形行列
C =
     9     4
     2     8
     6     7

>> u = [3; 1; 4]      <==== m× 1 行列の列ベクトル（m＝3)
u =
     3
     1
     4
>> v= [2 0 -1]      <==== 1 × n 行列の
                          行ベクトル（n＝3)
v =
     2     0    -1
>> s=7          <=== 1 × 1 行列のスカラー
s =
     7
```

図 3.8　行列生成の例

（2）加算と減算

```
>> X = A + B                <==== 加算
X =
     9     2     7
     4     7    10
     5    12     8
>> Y = X - A          <==== 減算（結果はB）
Y =
     8     1     6
     3     5     7
     4     9     2
>> X = A + C    <==== 二つの行列の行数・
     列数が同じ大きさでない
     ??? エラー: ==> +
     行列の次元は同じである必要があります．
>> w = v + s
w =
     9     7     6
```

図 3.9　行列演算，加算，減算の例

（3）ベクトル積と転置

同じ長さの行ベクトルと列ベクトルは，どちらか一方のベクトルを基準に他のベクトルを順番に従って，乗算していきます．結果は，スカラー，すなわち，内積になるか，行列すなわち外積のどちらかになります．

```
>> x = v*u
x =
     2
>> x = u*v
x =
     6     0    -3
     2     0    -1
     8     0    -4
>> x = v'
x =
     2
     0
    -1
```

図 3.10　ベクトルの積と転置の例

複素数ベクトルまたは行列 z に対して，z'は複素共役転置を定義します．共役を取らない複素転置は，他の配列演算と同様に，z.'で定義します．

例えば，

```
>> z = [1+2i 3+4i]   <==== 複素数行列
z =
   1.0000 + 2.0000i   3.0000 + 4.0000i
>> z'          <==== 複素共役転置
ans =
   1.0000 - 2.0000i
   3.0000 - 4.0000i
z.'          <====共役を取らない複素転置
ans =
   1.0000 + 2.0000i
   3.0000 + 4.0000i
```

図 3.11　複素ベクトル演算の例

### （4）行列の乗算

行列の乗算は，その中に含まれている線形変換の構成を反映する一つの方法で，連立線形方程式のコンパクトな表現です．行列積 C=AB は，A の列次元が B の行次元と等しいときか，またはどちらか一方がスカラーのとき，定義されます．A が m 行 p 列で，B が p 行 n 列の行列ならば，積 C は m 行 n 列の行列になります．積は，MATLAB の for ループ，コロン記法，ベクトルのドット積を使って定義されます．

```
for i = 1:m
    for j = 1:n
       C (i,j)  = A (i,:) *B (:,j) ;
    end
end
```

図 3.12　行列の乗算の定義

MATLAB は，行列乗算を定義するのに，単一アスタリスクを使います．つぎの二つの例は，行列積が累積になっていないことを示すものです．すなわち，AB は，通常，BA と異なります．

```
A =
     1     1     1
     1     2     3
     1     3     6
B =
     8     1     6
     3     5     7
     4     9     2
u =
     3
     1
     4
v =
     2     0    -1
s =
     7
C =
     9     4
     2     8
     6     7
```

図 3.13　行列生成の例

```
>> X=A*B                <====行列は，右の列ベクトルと左の行ベクトルとの乗算になります．
X =
    15    15    15
    26    38    26
    41    70    39
>> Y=B*A                <====行列は，右の列ベクトルと左の行ベクトルとの乗算になります．
Y =
    15    28    47
    15    34    60
    15    28    43
```

```
>> x = A*u
x =
     8
    17
    30
>> y=v*B                <====長方形行列乗算は，次元の整合性を満足していなければなりません．
y =
    12    -7    10
>> X = A*C
X =
    17    19
    31    41
    51    70
>> Y = C*A    <==== 次元が異なっています．
??? エラー: ==> *
% 内部行列の次元は同じである必要があります．
他には，スカラーによる乗算ができます．
>> w = s*v
w =
    14     0    -7
```

図 3.14　行列積の例

### （5）単位行列

一般に受け入れられる数学記法では，対角要素が 1 で，他の要素が 0 である種々の大きさの行列を，単位行列と言い，大文字 I を使って定義します．これらの行列は，次元の整合性が保たれている範囲で，AI=A また IA=A の性質をもっています．MATLAB の元々のバージョンでは，大文字，小文字の区別がなく，i は既にサブスクリプトや複素数単位として使っていたので，単位行列として I を使うことができませんでした．それで，語呂合わせを使って記号を作りました．すなわち，I と同じ発音をもつ eye を使うことにしました．関数 eye（m,n）は，m 行 n 列の長方形単位行列を出力し，eye（n）は，n 行 n 列の正方単位行列を出力します（図 3.15）．

```
>>  eye (3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

図 3.15　単位行列の例

### （6）Kronecker テンソル積

2 つの行列の Kronecker（「クロネッカー」と言います）積，kron（X,Y）は，X の要素と Y の要素の中で取り得る可能な組み合わせ積から作成できる大きな行列になります．

X が m 行 n 列で，Y が p 行 q 列ならば，kron (X,Y) は mp 行 nq 列の行列になります．要素は，つぎの順番で並べられます．

［X（1,1）*Y　X（1,2）*Y　...　X（1,n）*Y
　　・・・
X（m,1）*Y　X（m,2）*Y　...　X（m,n）*Y］

Kronecker 積は，0 と 1 からなる行列を使って，小さな行列の繰り返しコピーを作成します．たとえば，X が，2 行 2 列の行列で，I が 2 行 2 列の単位行列のとき，2 つの行列 kron（X,I）と kron（I,X）は，つぎのようになります（図 3.16）．

```
>> X = [1 2; 3 4]
X =
     1     2
     3     4
>> I = eye (2,2)
I =
     1     0
     0     1
>> kron (X,I)
ans =
     1     0     2     0
     0     1     0     2
     3     0     4     0
     0     3     0     4
>> kron (I,X)
ans =
     1     2     0     0
     3     4     0     0
     0     0     1     2
     0     0     3     4
```

図 3.16　Kronecker 積の例

## 3.6　線形代数計算

### （1）連立方程式

次の連立方程式を行列方程式で表すと，

$$2x_1 + x_2 + x_3 = 1$$
$$-x_1 + 2x_2 + x_3 = 1$$
$$-x_1 - x_2 + 2x_3 = 1$$

$$A = \begin{pmatrix} 2 & 1 & 1 \\ -1 & 2 & 1 \\ -1 & -1 & 2 \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$AX = B$$

$X = A \backslash B$ は，行列方程式 $AX = B$ の解を表します．

MATLAB では，次のように計算します（図 3.17）．

```
>> A = [2 1 1;-1 2 1;-1 -1 2]
A =
     2     1     1
    -1     2     1
    -1    -1     2
>> B = [1;1;1]
B =
     1
     1
     1
>> X= A¥B
X =
    0.0714
    0.2143
    0.6429
>> A*X
ans =
     1
     1
     1
```

図 3.17　行列方程式の解の計算例

### （2）逆行列と行列式

A が正方で，正則ならば，方程式 AX＝I と XA＝I は，同じ解 X を持ちます．この解は，A の逆行列と呼ばれ，$A^{-1}$ で表し，関数 inv で計算できます．

行列の行列式は理論的な考察やある種の数式計算で利用可能ですが，そのスケーリングや丸め誤差の性質が，数値的な計算に信頼性を低下させます．その条件下で，関数 det は，正方行列の行列式を計算します（図 3.18）．

再度，A は対称で，整数要素で，行列式が 1 であるので，逆行列の行列式も 1 になります．

一方，X の要素の細かいチェック，または，有理形式（format rat）を使用すると，これらは 360 である整数を割ったものになることがわかります（図 3.19）．

```
>> A = pascal (3)
A =
     1     1     1
     1     2     3
     1     3     6
>> d = det (A)
d =
     1
>> X = inv (A)
X =
     3    -3     1
    -3     5    -2
     1    -2     1
```

図 3.18　行列式と逆行列の例

```
>> B = magic (3)
B =
     8     1     6
     3     5     7
     4     9     2
>> d = det (B)
d =
  -360
```

```
>> X = inv (B)
X =
     0.1472   -0.1444    0.0639
    -0.0611    0.0222    0.1056
    -0.0194    0.1889   -0.1028
```

図 3.19　行列式と逆行列の例２

A が正方で，正則ならば，丸め誤差を考えないで，X = inv(A)*B は，理論的には X = A¥B と等価で，Y =B*inv(A)は Y = B/A と等価です.

しかし，バック・スラッシュとスラッシュを含む計算が好まれます．これは，計算時間や，メモリが小さく，よりエラーの検出が可能であるためです.

（1）の連立方程式を，逆行列を用いて解くと，図 3.20 のようになります.

```
>> X=inv(A)*B
X =
    0.0714
    0.2143
    0.6429
>> A*X
ans =
    1.0000
    1.0000
    1.0000
```

図 3.20　連立方程式を逆行列で解く

### （3）LU 分解，コレスキー分解，QR 分解

MATLAB の線形方程式機能は，3 つの基本的な行列分解をベースにしています.

1）対称，正定行列に対しては，コレスキー分解

2）一般的な正方行列に対しては，LU 分解法
(Gauss 消去法)

3）長方形行列に対しては，直交化

これら 3 つの分解は，関数 chol，lu，qr を使って行うことができます. これらの 3 つの因子分解すべては, 対角要素の上部または下部のどちらかのすべての要素がゼロである三角行列を使います. 三角行列を含む線形方程式システムは，前置代入，または，後退代入のどちらかを使って，簡単に，容易に解くことができます.

### 1）LU 分解

（1）の連立方程式を LU 分解法（ガウスの消去法）で解くと，図 3.21 のようになります.

```
>> A =  [2 1 1;-1 2 1;-1 -1 2]
A =
     2     1     1
    -1     2     1
    -1    -1     2
>> B = [1; 1; 1]
B =
     1
     1
     1
>> [L, U] = lu(A)
L =
    1.0000         0         0
   -0.5000    1.0000         0
   -0.5000   -0.2000    1.0000

U =
    2.0000    1.0000    1.0000
         0    2.5000    1.5000
         0         0    2.8000

>> X = U ¥ (L ¥ B)
X =
    0.0714
    0.2143
    0.6429

>> A * X
ans =
     1
     1
     1
```

図 3.21　連立方程式を LU 分解で解く

### 2）Cholesky 分解

コレスキー分解は, 対称行列を三角行列とその転置行列との積として表現します.

$$A = R'R$$

ここで, R は, 上三角行列です. 対称行列すべてが，この方法で分解できることはありません．すなわち，因子分解による行列は正値行列です．これは，A の対角要素は正で，非対角要素は"あまり大きくない"ものです．Pascal 行列は，興味のある例題です．この章を通して，例題の行列 A は，3 行 3 列の Pascal 行列です．ちょっと，一時的ですが，6 行 6 列の行列を考えましょう.

A の要素は，二項係数になります．各々の要素は，その上と左の要素の和になります. コレスキー分解は，図.3.22 のようになります.

要素は，再び，二項係数になっています. R'*R が A に等しいことは，二項係数の積の和を含んでいることを示すものです.

```
>> A = pascal (6)
A =
     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
>> R = chol (A)
```

```
R =
     1     1     1     1     1     1
     0     1     2     3     4     5
     0     0     1     3     6    10
     0     0     0     1     4    10
     0     0     0     0     1     5
     0     0     0     0     0     1
```

図 3.21　行列 A のコレスキー分解の計算例

コレスキー分解は，複素数行列にも適用できます．コレスキー分解を行った複素数行列は A' = A を満足し，Hermitian positive definite と言われます．

線形システム

$$A * x = b$$

は，

$$R'R * x = b$$

で置き換えることで，コレスキー分解を行うことができます．バック・スラッシュ（日本では¥記号）演算子は，三角システムで使えるので，つぎのようにして簡単に解くことができます．

$$x = R \setminus (R' \setminus b)$$

A が，n 行 n 列の行列の場合，chol（A）の計算の誤差は$O(n^3)$になりますが，連続バック・スラッシュによる解の誤差は，たった$O(n^2)$です．

**3）QR 分解**

直交行列，または，直交性の列をもつ行列は，各列が単位長さをもち，お互いが直交関係になる実数行列に分解されます．Q が直交であれば，

$$Q'Q = 1$$

になります．最も簡単な直交行列は，2 次元の座標回転変換に使うものです．

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

複素数行列に対して，対応する項はユニタリです．直交ユニタリ行列は，長さと角度が保存され，そのため誤差が大きくならないので，数値計算に対して望ましいものです．

直交または QR 因子分解は，任意の直交行列を一つの直交行列またはユニタリ行列と一つの上三角行列の積で表現します．列の並べ替えも含まれています．

$$A = QR \quad \text{または} \quad AP = QR$$

で，ここで，Q は直交またはユニタリ行列で，R は上三角行列で，P は並べ替え行列です．

線形システムは，列よりも多くの行をもつ長方形行列を含んでいます．

すなわち，m 行 n 列で，m >n です．

フルサイズ QR 因子分解は，m 行 m 列の正方直交行列 Q と m 行 n 列上三角行列 R の積になります（図 3.22）．

```
>> C = [9 4; 2 8; 6 7]
C =
     9     4
     2     8
     6     7
>> [Q,R] = qr(C)
Q =

   -0.8182    0.3999   -0.4131
   -0.1818   -0.8616   -0.4739
   -0.5455   -0.3126    0.7777
R =
  -11.0000   -8.5455
         0   -7.4817
         0         0
```

図 3.22　行列の QR 分解の計算例

## 4．MATLAB での文字処理

### 4.1　文字列

（1）キャラクタとテキスト

MATLABR へテキストを入力するには，シングル・コードを利用します．例えば，図 4.1 の結果は，今まで取り扱ってきた数値行列や配列と異なる種類のものです．

```
>> s = 'Hello'
s =
Hello
>> a = double (s)
a =
    72   101   108   108   111
>> t = char (a)
t =
Hello
```

図 4.1　MATLAB を使ってのテキストの入力例

これは，1 行 5 列のキャラクタ配列です．内部的に，キャラクタは数字として格納されますが，浮動小数点書式ではありません．ステートメント

```
>> a = double (s)
```

は，キャラクタ配列を ASCII コードの浮動小数点表現を含む数値行列に変換します．ステートメント

```
>> s = char (a)
```

は，図 4.1 のように逆の変換を行います．

数字をキャラクタに変換することにより，ユーザのコンピューターで使用可能なフォントの種類を調べることができます．

基本的な ASCII キャラクタセット（図 4.2）の中のプリント可能なキャラクタは，整数 32:126 で表わされます（32 よりも小さい整数と 127 は，プリントで

きない制御キャラクタを表わします).


▼ ASCII文字コード

| 文字 | 10進 | 16進 | 文字 | 10進 | 16進 | 文字 | 10進 | 16進 | 文字 | 10進 | 16進 | 文字 | 10進 | 16進 | 文字 | 10進 | 16進 | 文字 | 10進 | 16進 | 文字 | 10進 | 16進 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NUL | 0 | 00 | DLE | 16 | 10 | SP | 32 | 20 | 0 | 48 | 30 | @ | 64 | 40 | P | 80 | 50 | ` | 96 | 60 | p | 112 | 70 |
| SOH | 1 | 01 | DC1 | 17 | 11 | ! | 33 | 21 | 1 | 49 | 31 | A | 65 | 41 | Q | 81 | 51 | a | 97 | 61 | q | 113 | 71 |
| STX | 2 | 02 | DC2 | 18 | 12 | " | 34 | 22 | 2 | 50 | 32 | B | 66 | 42 | R | 82 | 52 | b | 98 | 62 | r | 114 | 72 |
| ETX | 3 | 03 | DC3 | 19 | 13 | # | 35 | 23 | 3 | 51 | 33 | C | 67 | 43 | S | 83 | 53 | c | 99 | 63 | s | 115 | 73 |
| EOT | 4 | 04 | DC4 | 20 | 14 | $ | 36 | 24 | 4 | 52 | 34 | D | 68 | 44 | T | 84 | 54 | d | 100 | 64 | t | 116 | 74 |
| ENQ | 5 | 05 | NAK | 21 | 15 | % | 37 | 25 | 5 | 53 | 35 | E | 69 | 45 | U | 85 | 55 | e | 101 | 65 | u | 117 | 75 |
| ACK | 6 | 06 | SYN | 22 | 16 | & | 38 | 26 | 6 | 54 | 36 | F | 70 | 46 | V | 86 | 56 | f | 102 | 66 | v | 118 | 76 |
| BEL | 7 | 07 | ETB | 23 | 17 | ' | 39 | 27 | 7 | 55 | 37 | G | 71 | 47 | W | 87 | 57 | g | 103 | 67 | w | 119 | 77 |
| BS | 8 | 08 | CAN | 24 | 18 | ( | 40 | 28 | 8 | 56 | 38 | H | 72 | 48 | X | 88 | 58 | h | 104 | 68 | x | 120 | 78 |
| HT | 9 | 09 | EM | 25 | 19 | ) | 41 | 29 | 9 | 57 | 39 | I | 73 | 49 | Y | 89 | 59 | i | 105 | 69 | y | 121 | 79 |
| LF* | 10 | 0a | SUB | 26 | 1a | * | 42 | 2a | : | 58 | 3a | J | 74 | 4a | Z | 90 | 5a | j | 106 | 6a | z | 122 | 7a |
| VT | 11 | 0b | ESC | 27 | 1b | + | 43 | 2b | ; | 59 | 3b | K | 75 | 4b | [ | 91 | 5b | k | 107 | 6b | { | 123 | 7b |
| FF* | 12 | 0c | FS | 28 | 1c | , | 44 | 2c | < | 60 | 3c | L | 76 | 4c | \¥ | 92 | 5c | l | 108 | 6c | \| | 124 | 7c |
| CR | 13 | 0d | GS | 29 | 1d | - | 45 | 2d | = | 61 | 3d | M | 77 | 4d | ] | 93 | 5d | m | 109 | 6d | } | 125 | 7d |
| SO | 14 | 0e | RS | 30 | 1e | . | 46 | 2e | > | 62 | 3e | N | 78 | 4e | ^ | 94 | 5e | n | 110 | 6e | ~ | 126 | 7e |
| SI | 15 | 0f | US | 31 | 1f | / | 47 | 2f | ? | 63 | 3f | O | 79 | 4f | _ | 95 | 5f | o | 111 | 6f | DEL | 127 | 7f |

* LFはNL、FFはNPと呼ばれることもある。
* 赤字は制御文字、SPは空白文字(スペース)、黒字と緑字は図形文字。
* 緑字はISO 646で割り当ての変更が認められており、例えば日本ではバックスラッシュが円記号になっている。

図 4.2　ASCII コード表

これらの整数は，つぎのように適切な 6 行 16 列の配列に並べることができます．6×16=96 なので，32:127 まで指定し，

```
F = reshape(32:127,16,6)';
```

として，これらの整数がキャラクタとして解釈されるとき，結果は現在使われているフォントに依存します．このコードを表示するために，つぎのステートメントをタイプします（図 4.3）．

```
char (F)
```

```
>> F = reshape (32:127,16,6) ';
>> char (F)
ans =
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[¥]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
```

図 4.3　MATLAB を使ってのテキストのフォントの表示

```
>> h = [s, ' world']
```

は，横方向に文字列を連結し，つぎの結果を出力します．

```
h =
   Hello world
```

ステートメント

```
>> v = [s; 'world']
```

は，縦方向に文字列を連結し，つぎの結果を出力します．

```
v =
   Hello
   world
```

h の中で 'w' の前にブランクが挿入されていなければならないことと，v の中で 2 つの単語は同じ長さでなければならないことに注意してください．結果の配列は，共にキャラクタ配列で，h は 1 行 11 列で，v は 2 行 5 列です．

異なる長さのラインを含むテキストの内容を取り扱うには，キャラクタ配列を付け加えて同じ長さにするか，文字列のセル配列にするかの 2 つの方法があります．関数 char は，複数行のラインを入力でき，各々のラインがすべて同じ長さになるようにブランクを加えることができます．そして，各ラインが行ごとに分割されたキャラクタ配列を作成します．たとえば，

```
S = char ('A','rolling','stone',
    'gathers','momentum.')
```

は，5 行 9 列のキャラクタ配列です．

```
S =
A
rolling
stone
gathers
momentum.
```

S の最初の 4 つの行には，各行を同じ長さにするために必要なブランクがあります．また，1 つのセル配列にテキストを格納することができます．たとえば，

```
C = {'A';'rolling';'stone';
    'gathers';'momentum.'}
```

は，5 行 1 列のセル配列になります．

```
C =
    'A'
    'rolling'
    'stone'
    'gathers'
    'momentum.'
```

ブランクが追加されたキャラクタ配列を，つぎのコマンドで文字列からなるセル配列に変換できます．

```
C = cellstr (S)
```

そして，逆も可能です．

```
S = char (C)
```

## 4.2　文字列操作関数

文字列操作する関数には次のものがあります．

表 4.4　MATLAB 演算子の文字列操作関数

| カテゴリ | 関数名 | 詳細説明 |
|---|---|---|
| 一般的なもの | blanks | ブランク文字列 |
| | cellstr | キャラクタ配列から文字列のセル配列を作成 |
| | char | キャラクタ配列の作成（文字列） |

| | | |
|---|---|---|
| | eval | MATLABR 表現を使った文字列の実行 |
| 文字列と数字の変換 | double | 文字列を数値コードに変換 |
| | num2str | 数字を文字列に変換 |
| 文字演算 | strcmp | 文字列の比較 |
| | strrep | 文字列の置き換え |
| | strcat | 文字列の連結 |
| | upper | 大文字に変換 |
| | lower | 小文字に変換 |

## 5．MATLAB プログラミング

### 5.1　制御フロー

MATLAB には，通常のプログラムをするために繰り返しや分岐の構文が用意されています．これらの構文は処理の流れをコントロールすることから制御フロー文と呼ばれています．MATLAB には 8 つの制御フロー文があります．

1）if 文は else や elseif とともに使い，論理条件をもとに 1 つのグループ化した文を実行します．
2）switch 文は case や otherwise とともに使い，ある論理条件の値に従って，種々のグループに分かれた文を実行します．
3）while 文は，ある論理条件を満足する間，1 つのグループ化した文を実行します．
4）for 文は，設定した回数だけ 1 つのグループ化した文を実行します．
5）continue 文は，for または while の次の繰り返しまで続けて，残りの文をスキップします．
6）break 文は，for または while ループの実行を停止します．
7）try...chatch 構文は，実行中にエラーが検出される場合，フロー制御を変更します．
8）return 文は，実行を起動関数に戻します．

なお，総てのフローを制御するブロックの終わりを示すのに end 文を使います．

#### （1）if, else, elseif

a が偶数の場合，”a is even”と表示して，2 で割った商を計算してみましょう（図 5.1 上段）．

rem (a,2) は，a を 2 で割った余りを計算します．

disp は () の中を表示する関数です．

さらに，a が奇数の場合，”a is odd”と表示して，2 で割った商を計算してみましょう（図 5.1 下段）．

fix（a/2）は，a を 2 で割った結果の小数点以下を切り捨てます．

```
if rem (a,2)  == 0
    disp ('a is even')
    b = a/2;
end
```

```
if rem (a,2)  == 0
    disp ('a is even')
    b = a/2;
else
    disp ('a is odd')
    b = fix (a/2) ;
end
```

図 5.1　if, else 構文の例題プログラム

#### （2）switch

```
switch input_num
    case -1
        disp ('negative one') ;
    case 0
        disp ('zero') ;
    case 1
        disp ('positive one') ;
    otherwise
        disp ('other value') ;   <==== -1,0,1 以外の場合に表示します.
end
```

図 5.2　switch 構文の例題プログラム

#### （3）while ループ

```
n = 1;
while prod (1:n)  < 1e100
   % 1 から n までの積が 未満である間,
   % n を 1 増やします.
   n = n + 1;
end
```

図 5.3　while ループ構文の例題プログラム

#### （4）for ループ

```
for m = 1:3
    for n = 1:3
        x (m,n)  = m + n*i;
    end
end
```

```
x =
  1.0000 + 1.0000i   1.0000 + 2.0000i   1.0000 + 3.0000i
  2.0000 + 1.0000i   2.0000 + 2.0000i   2.0000 + 3.0000i
  3.0000 + 1.0000i   3.0000 + 2.0000i   3.0000 + 3.0000i
```

図 5.4　for ループ構文の例題プログラム

#### （5）continue ループのスキップ

continue 文は，for または while ループの次の繰り返しまで，コントロールを続け，ループ本体の中の残りの文を跳ばします．入れ子のループでは，continue は，それを囲む for または while ループの次の繰り返しまで，コントロールを続けます．

```
Tom Watson
101 S. Main St.
```

```
Anytown, USA
Mary B. Smith
123 Home Street

% --- comment
Hometown, UK
```

図 5.5　master.m の内容

例題は，図 5.5 の master.m という名のファイルの中に，ブランク行やコメント行があった場合，行数をカウントしないプログラムです（図 5.6）.

実行結果は「6 lines」です.

```
fid = fopen ('master.m','r') ;
count = 0;
while ~feof (fid)
    line = fgetl (fid) ;
    if isempty(line) | strncmp(line,'%',1)
        continue
    end
    count = count + 1;
end
disp (sprintf ('%d lines',count)) ;
```

図 5.6　continue 構文の例題プログラム

### （6）break ループの停止

break 文に出会うと，実行はループの外の次の文を実行します. 入れ子になったループの中で, break は, 最も内側のループのみを停止します（図 5.7）.

```
x = 2
while 1
    if x > 1000000
        break;
    end
    x = x^2;
end
x =
  4.2950e+009
```

図 5.7　break 構文＝ループの停止

### （7）try ... catch

try ... catch 構文の一般的な型は, 次のとおりです. エラーが発生するまで，try と catch の間の文が実行されます．エラーが発生後，catch ... end 間の文が実行されます. lasterr を使って, エラーの原因をしらべることができます（図 5.8）.

```
try,
   statement,
   ...,
   statement,
catch,
   statement,
   ...,
   statement,
end
```

図 5.8　try...catch 構文

### （8）return

return は，コマンドの実行の流れを停止し，コントロールを起動した関数またはキーボードに戻します. return 文はまた，キーボード・モードを終了させるために使用されます.

## 5.2　MATLAB スクリプト・ファイル

MATLAB は, 対話形式の計算環境と同様に強力なプログラミング言語です. MATLAB 言語の中のコードを含むファイルを M-File（エム・ファイル）と言います. MATLAB システムのエディタや普通のテキスト・エディタを使って M-ファイルを作ります．そして，それらを他の MATLAB 関数またはコマンド同様に使うことができます.

M-ファイルには 2 種類のものがあります.

1）スクリプト : このファイルは，入力引数を受け入れたり，出力引数を出力したりしません．このファイルは, ワークスペースの中のデータを使います.

2）ファンクション : このファイルは，入力引数を受け入れ, 出力引数を出力します. 内部変数は, 関数のローカル変数です.

あなたが MATLAB のプログラミングを始めたばかりならば，カレント・ディレクトリ上に実行しようとする M-ファイルを作成してください．あなた自身の M-ファイルが沢山完成したなら，他のディレクトリにそれらをまとめて, 個人的な Toolbox を作成して下さい．そして，MATLAB のサーチ・パスに加えて下さい.

関数名が重複したら，MATLAB はサーチ・パスの中で最初に現れるファイルを実行します.

例えば，myfunction.m の内容を表示するには,

```
>> type myfunction
```

のようにタイプインします.

### （1）スクリプト

あなたがスクリプトを読み込むと，MATLAB は, 単にファイルの中のコマンドを実行します. スクリプトは, ワークスペースの中に存在するデータを取り扱うか，または，演算するための新しいデータを作成します．スクリプトは，出力引数を出力しませんが，作成する変数はワークスペースに残り, その後の計算に使われます.

（例－1）Switch 文の例（EX_Switch_00.m）

```
>> type EX_Switch_00
XS = [-1 0 1];
for i= 1:3
X = XS(i)
switch sign(X)
  case 1
    disp('X is positive.');
  case -1
    disp('X is negative.');
  otherwise
    disp('X is zero.');
end
end

>> EX_Switch_00
X =
    -1
X is negative.

X =
     0
X is zero.

X =
     1
X is positive.
```

図 5.9　スクリプト・ファイル EX_Switch_00.m と実行結果

スクリプトは，plot コマンド等を使って，グラフィカル出力を作成することもできます．

（例－2）魔方陣のランク計算（magicrank.m）

例えば，つぎの MATLAB コマンドを含んだ magicrank.m と呼ばれるファイルを，テキスト／エディタを使って作成し，MATLAB 作業用フォルダー「D:¥MATLAB_Work」に保存します（図 5.10）．

```
% 魔方陣のランクの計算
r = zeros (1,32);
for n = 3:32
  r(n) = rank(magic(n));
end
r
bar(r)
```

図 5.10　スクリプト・ファイル magicrank.m

```
>> type magicrank.m

% 魔方陣のランクの計算
r = zeros (1,32);
for n = 3:32
  r(n) = rank(magic(n));
end
r
bar(r)
>> magicrank
r =

  1 列から 20 列

    0    0    3    3    5    5    7    3    9    7   11    3   13    9   15
3   17   11   19    3

  21 列から 32 列

   21   13   23    3   25   15   27    3   29   17   31
```

図 5.11　スクリプトの実行結果出力

ステートメント magicrank により，MATLAB はコマンドを実行し（図 5.11），最初の 30 個の魔方陣のランクを計算し，結果を棒グラフにプロット表示します（図 5.12）．ファイルの実行が終了すると，変数 n と r がワークスペースに残ります．

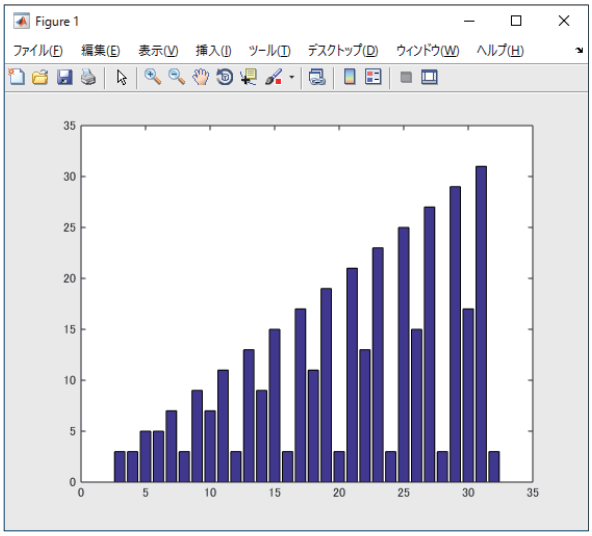



図 5.12　スクリプトの実行結果のグラフ

## （2）ファンクション

**ファンクションは，入力引数をもち，出力引数を出力する M-ファイルです．**

**M-ファイルの名前と関数の名前は同じものにしてください．ファンクション M-ファイルは，それ自身がもつワークスペースの中で変数を使い，MATLAB コマンドプロンプトでアクセスするワークスペースと区別します．**

良い例題は，rank を使って示されます．M-ファイル rank.m は，「toolbox/matlab/matfun」のディレクトリにあります．ファイルの内容は，type rank コマンドで表示されます．内容は図.5.13 のとおりです．

ファンクション M-ファイルの最初の行は，function と言うキーワードで始まります．これは，関数名と引数の順序を与えるものです．この例題では，入力引数は 2 つで，出力引数は 1 です．

最初のブランク行または実行可能行までの数行は，ヘルプ・テキストを与えるコメント行です．これらの行は，help rank とタイプすることで，表示されます．

ヘルプ・テキストの最初の行は, H1 ラインと言われ, lookfor コマンドを使うか, またはディレクトリ上の help により MATLAB が表示するものです.

ファイルの残りは, 関数を定義している実行可能な MATLAB コードです. 最初の行の中の変数 r,A,tol と同様に関数の中に導入されている変数 s は, すべて関数のローカル変数です. すなわち, MATLAB ワークスペースの中の変数とは異なるものです.

この例題は, MATLAB 関数の一つの見方を示すもので, 通常, 他のプログラミング言語の中には見られません. すなわち, 引数の数を可変にできます. rank 関数は, 種々の方法で使うことができます.

```
rank (A)
r = rank (A)
r = rank (A,1.e-6)
```

```
>> type rank

function r = rank(A,tol)
%RANK   Matrix rank.
%   RANK(A) provides an estimate of the number of linearly
%   independent rows or columns of a matrix A.
%   RANK(A,tol) is the number of singular values of A
%   that are larger than tol.
%   RANK(A) uses the default tol = max(size(A)) * eps(norm(A)).
%
%    Class support for input A:
%       float: double, single

%   Copyright 1984-2007 The MathWorks, Inc.

s = svd(A);
if nargin==1
   tol = max(size(A)) * eps(max(s));
end
r = sum(s > tol);
```

図. 5. 13　M ファイル rank. m の内容表示

多くの M-ファイルは, この方法で実行します. 出力引数を設定しなければ, 結果は ans にストアされます. 2 番目の入力引数が設定されなければ, 関数はデフォルト値を使って計算します. 関数の中で, nargin と nargout と名付けられた 2 つの量は, 関数の中で特殊な使い方で, 入力引数の数や出力引数の数を出力するものです. rank 関数は nargin を使っていますが, nargout は使う必要がありません.

## 5.3　MATLAB ファイル入出力

次のようなテキスト・データを, grades.txt というファイル名で, Z:¥MATLAB_Work¥grades.txt として保存されていたとします. このテキスト・データを次のように読み込みます（図.5.14～17）.

```
   John    Ann     Martin  Rob
   88.4    91.5    89.2    77.3
   83.2    88.0    67.8    91.0
   77.8    76.3            92.5
   92.1    96.4    81.2    84.6
```

図 5. 14　テキスト・ファイル grades. txt

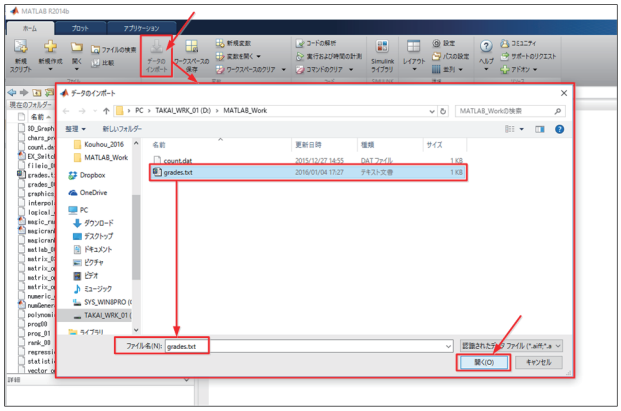


図 5. 15　データのインポートを開始

インポートするテキスト・ファイル名 grades. txt を指定

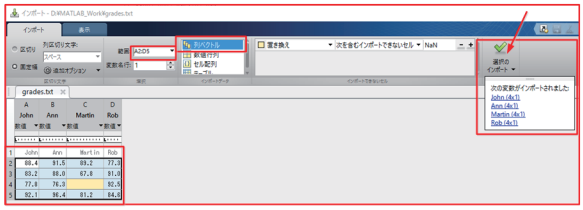


図 5. 16　インポート・ダイアログでの操作とインポートの実施

```
>> John, Ann, Martin, Rob
John =
   88.4000
   83.2000
   77.8000
   92.1000
Ann =
   91.5000
   88.0000
   76.3000
   96.4000
Martin =
   89.2000
   67.8000
       NaN
   81.2000
Rob =
   77.3000
   91.0000
   92.5000
   84.6000
S = [John, Ann, Martin, Rob]
S =
   88.4000   91.5000   89.2000   77.3000
   83.2000   88.0000   67.8000   91.0000
   77.8000   76.3000       NaN   92.5000
   92.1000   96.4000   81.2000   84.6000
```

図 5. 17　インポート・データを確認する

表 5.18　ASCII データを入力するための関数

| 関　数 | データ・タイプ | デリミタ | 返す値の数 | 記　　述 |
|---|---|---|---|---|
| csvread | 数値データ | コンマ | 1 | 表計算データ |
| dlmread | 数値データ | 任意の文字 | 1 | 柔軟で使い易い |
| fscanf | 英字，数字 | 任意の文字 | 1 | 行単位で入力，ファイルの識別子，読み込み後 fclose 要 |
| load | 数値データ | スペース | 1 | 使い易い |
| textread | 英文字，数字 | 任意の文字 | 複数 | 柔軟で，パワフルで，使い易い，format ストリング使用 |

表 5.19　入力可能なファイル Format と入力コマンド

| データのサンプル | ファイル拡張子 | 記述 |
|---|---|---|
| 1 2 3 4 5<br>6 7 8 9 10 | .txt<br>.dat　or other | Import Wizard による入力 |
| 1; 2; 3; 4; 5;<br>6; 7; 8; 9; 10<br>or<br>1, 2, 3, 4, 5<br>6, 7, 8, 9, 10 | .txt<br>.dat<br>.cvs<br>or other | load,<br>dlmread<br>csvread |
| Ann Type1 12.34 45 Yes<br>Joe Type2 45.67 67 No | .txt<br>.dat　or other | textread 関数 |
| Grade1　Grade2　Grade3<br>91.5　89.2　77.3<br>88.0　67.8　91.0<br>67.3　78.1　92.5 | .txt<br>.dat<br>or other | Import Wizard による入力<br>textread |

表 5.20　出力可能なファイル Format と出力コマンド

| データのサンプル | ファイル拡張子 | 記述 |
|---|---|---|
| 1 2 3 4 5<br>6 7 8 9 10 | .txt<br>.dat　or other | dlmwrite<br>diary |
| 1; 2; 3; 4; 5;<br>6; 7; 8; 9; 10 | .txt<br>.dat | dlmwrite<br>セミコロンをデリミタに使用 |

表 5.21　ASCII データを出力するための関数

| 関　　数 | データ・タイプ | デリミタ | 記述 |
|---|---|---|---|
| diary | 数値データ/セル配列 | スペース | 小さな配列，エディタが必要 |
| dlmwrite | 数値データ | 任意の文字 | 柔軟で使い易い |
| fprintf | 英字，数字 | 任意の文字 | 行単位で出力，ファイルの識別子，fopen と fclose が必要 |
| save | 数値データ | タブ／スペース | 使い易い，高精度で出力する |

以下にデータの出力例を示します.

```
>> A = [ 1 2 3 4; 5 6 7 8] ;
>> save my_data.out A -ASCII
>> dlmwrite ('my_data2.out',A,';')
>> diary my_data3.out
>> A
A =
     1     2     3     4
     5     6     7     8
>> diary off
```

ファイル名：my_data.out 高精度形式で出力されます.

```
1.0000000e+000  2.0000000e+000  3.0000000e+000  4.0000000e+000
5.0000000e+000  6.0000000e+000  7.0000000e+000  8.0000000e+000
```

ファイル名：my_data2.out デリミタにセミコロン(;) を使用しています.

```
1;2;3;4
5;6;7;8
```

ファイル名：my_data3.out は右側のとおり，表示形式のまま記録されます.

```
     1     2     3     4
     5     6     7     8
```

# ６．多項式計算と補間

## 6.1　多項式

MATLAB は，多項式の根，多項式の値，カーブ・フッティングなど，標準的な多項式演算のための関数を提供しています．

### （1）多項式の表現

MATLAB は，次数の高い係数から行ベクトルとして多項式を表現します．

例えば，次の方程式：

$$x^3 - 2x - 5 = 0$$

の多項式

$$p(z) = x^3 - 2x - 5$$

（この方程式は，フランス・アカデミーで Newton 法を最初に表現するときに Wallis が使った）

を MATLAB に入力し，多項式の根を求める roots 関数と，設定した値を根とする多項式の値を求める poly 関数を使って，値を比較してみましょう（図 6.1）．

```
>> p = [1 0 -2 -5];
>> r = roots (p)
r =
   2.0946
  -1.0473 + 1.1359i
  -1.0473 - 1.1359i
>> p2 = poly (r)
p2 =
   1.0000        0   -2.0000   -5.0000
```

図. 6. 1　多項式の根との計算

```
>> q = polyder (p)
q =
     3     0    -2
```

図. 6. 2　多項式の微係数の計算

### （2）多項式の微分

関数 polyder は，任意の多項式の微係数を計算します（図 6.2）．

### （3）カーブ・フィッティング

関数 polyfit は，データ群に適合する最小二乗法による多項式係数を出力します．

**p = polyfit (x,y,n)**

ここで，x と y は適合されるデータ群の x データ，y データで，n は出力される多項式の次数です．

今，次の x と y のテスト・データを与え，多項式の次数を 3 として，推定値を計算し（図 6.3），グラフ出力します（図 6.4）．

```
>> x = [1 2 3 4 5];
>> y = [5.5 43.1 128 290.7 498.4];
>> p = polyfit (x,y,3)
p =
   -0.1917   31.5821  -60.3262   35.3400
>> x2 = 1:.1:5;
>> y2 = polyval (p,x2);
>> plot (x,y,'o',x2,y2)
>> grid on
```

図. 6. 3　最小二乗法によるカーブ・フィッティング

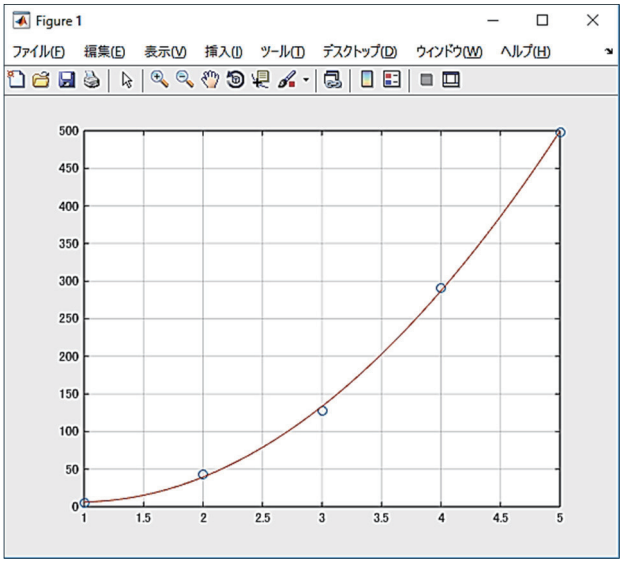



図. 6. 4　カーブ・フィッティングのグラフ出力

## 6.2　補間（Interpolation）

MATLAB は，実行速度やメモリしようとデータ・フィットの平滑化とのバランスを考えた数種類の補間法を提供しています．補間（内挿ともいう）は，既知のデータ点間の値を推定するプロセスです．このような作業は，信号処理，イメージ処理等の分野で重要なアプリケーションになります．詳しくは help を参照して下さい．

### （1）１次元補間

ここでは，データ解析やカーブ・フィッティングに重要な演算である多項式による補間を実行する関数 interp1 を紹介します．この関数は，与えられたデータに対して，データ間を多項式で近似し，希望する補間点で適切な関数を推定します．この最も一般的な型は，

**yi = interp1 (x,y,xi,method)**

ここで，y は，ある関数値を含むベクトルで，x は，y の値が与えられる点を含む y と同じ長さのベクトルです．xi は，補間される点を含むベクトルです．method は，補間法を設定するオプションの文字列です．以下の補間法を選択すると，他の処理より計算時

間とスペースが必要になることは覚えておいてください．また，これらの必要な事柄と結果の求まるスムージング状態との間でトレード・オフを行うことが必要です．

１） Nearest neighbor interpolation （method = 'nearest'）: この方法は，補間する点の最も近くにあるデータを使います．最も高速な方法ですが，スムージングと言う意味では，最も悪い結果を出力します．

２）Linear interpolation （method = 'linear'）: この方法は，存在しているデータ点のお互いの組毎に別々の関数をフィットさせ，xi で設定されている点で，その関数値を計算して出力します．これは，interp1 関数に対するデフォルトの方法です．nearest neighbor 法よりも多くのメモリを使い，実行時間もわずかですが多くなります．Nearest neighbor 補間法と異なり，結果は連続ですが，頂点で勾配が変わります．

３）Cubic spline interpolation （method = 'spline'）: この方法は，存在しているデータ点の各組の間を種々のキュービック関数でフィットさせ，関数 spline を使って，データ点でキュービック・スプライン補間を行います．よく使われる補間法で，実行時間を比較的必要とします．しかし，Cubic 内挿よりはメモリを必要としません．この手法は，すべての内挿法の中で最も平滑なものを作成します．入力データが一様でなかったり，ある部分に他の部分よりデータが集中していると，予期せぬ結果を出力する場合があります．

４）Cubic interpolation （method = 'pchip' または 'cubic'）: これらの方法は，同じものです．関数 pchip を使って，ベクトル x と y とを区分的なキュービック・エルミート補間を行います．これらの方法は，単調で，データの型を保ちます．neighbor 法や linear 法のどちらよりもメモリおよび実行時間を必要とします．しかし，内挿されたデータやその微係数は共に連続です．

xi の任意の要素が，x で張られる区間の外に位置する場合，設定した補間法は，外部補間を実行するものとして使われます．また，

**yi = interp1 (x,Y,xi,method,extrapval)**

は，外部補間された値を extrapval で置き換えます．NaN が，extrapval に対して，しばしば使われます．

なお，すべての方法は，一様分布していないデータに対しても機能します（図 6.5）．

（例：ランダムな測定データのでスプライン補間）

```
>> x = [15 35 55 80 115];
>> y = [32.5 143.2 55.6 93.7 64.8];
>> x, y
x =
    15    35    55    80   115
y =
   32.5000  143.2000   55.6000   93.7000   64.8000
>> x2 = 10:0.1:120;
>> y2 = interp1(x,y,x2,'spline');
>> plot(x,y,'o',x2,y2)
>> grid on
```

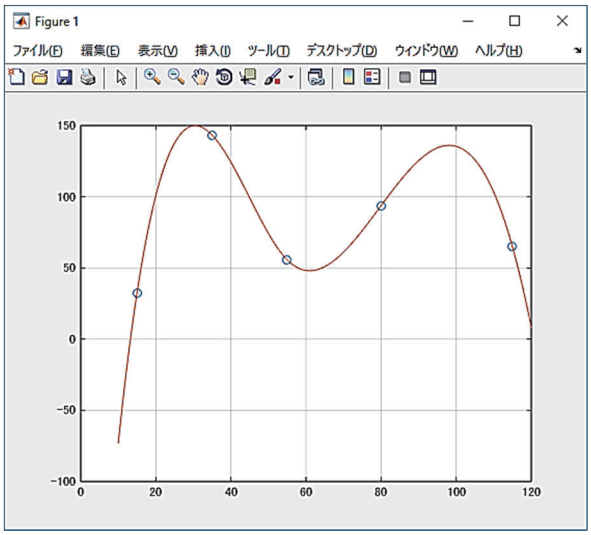


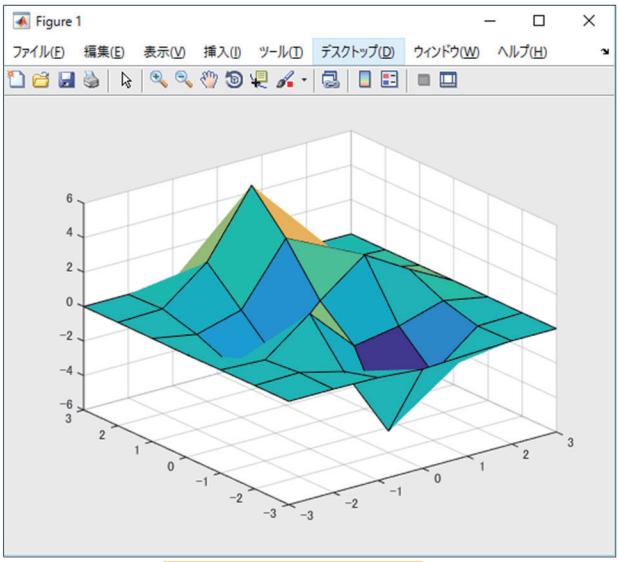

図 6.5　Spline 補間の実行例（上）とグラフ出力（下）

## （2）補間法の比較

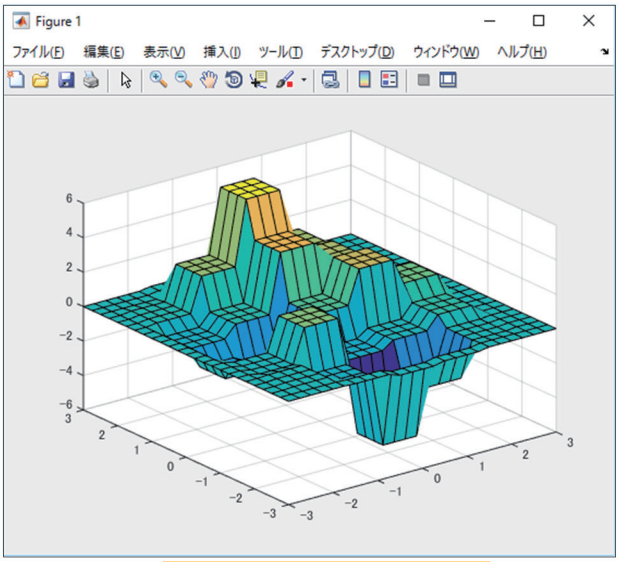
この例題は，７×７のデータ行列に２次元補間法を適用したもの比較します．ここでは解像度の peaks 関数を使用してデータを作成し，３次元メッシュ表示と等高線表示で補間法を比較します（図 6.6，6.7）．

```
>> [x,y] = meshgrid(-3:1:3);
>> z = peaks(x,y);
>> surf(x,y,z)
>> [xi,yi] = meshgrid(-3:0.25:3);
>> zi1 = interp2(x,y,z,xi,yi,'nearest');
>> surf(xi,yi,zi1)
>> zi2 = interp2(x,y,z,xi,yi,'bilinear');
>> surf(xi,yi,zi2)
>> zi3 = interp2(x,y,z,xi,yi,'bicubic');
>> surf(xi,yi,zi3)
>> contour(xi,yi,zi1)
>> contour(xi,yi,zi2)
>> contour(xi,yi,zi3)
```

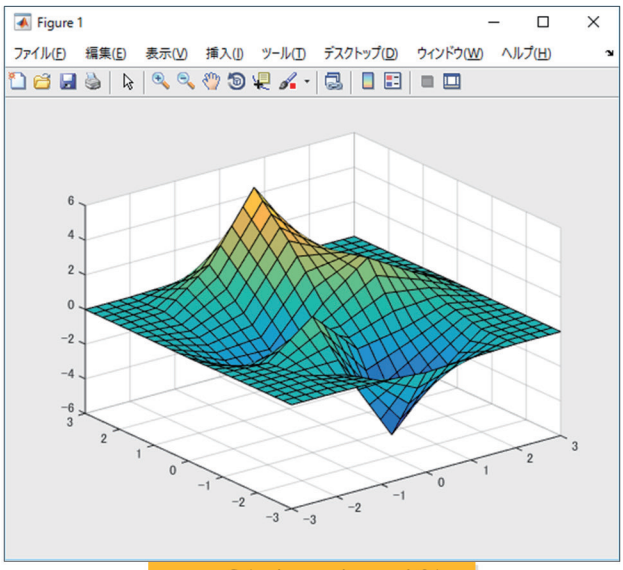
図 6.6　2 次元補間の実行例



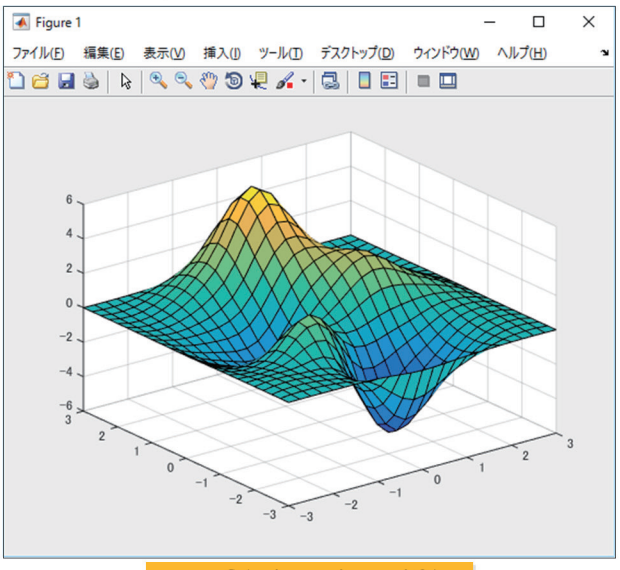

surf(x, y, z)



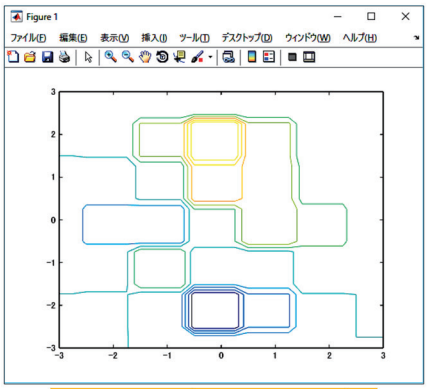

surf(xi, yi, zi1)
% nearest



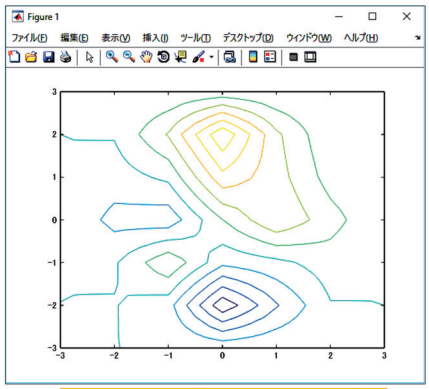

surf(xi, yi, zi2)
% bilinear



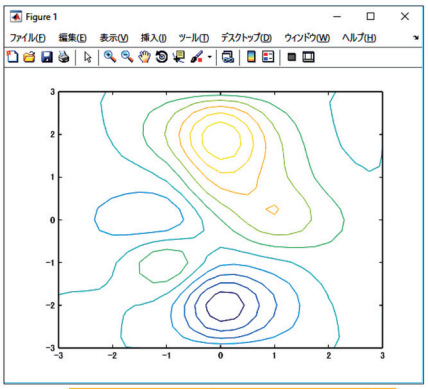

surf(xi, yi, zi3)
% bicubic




contour (xi,y1,zi1)
% nearest




contour (xi,y1,zi2)
% bilinear




contour (xi,y1,zi3)
% bicubic

図 6.7　2 次元補間の実行結果のグラフ出力

# 7．MATLAB によるデータ解析，統計処理

## 7.1　MATLAB によるデータ解析

MATLAB ではツールボックスが特化したデータ解析アプリケーション用の豊富な機能（表 7.1）を提供しています．

表 7.1　データ解析アプリケーションの機能

| Toolbox | データ解析アプリケーション |
|---|---|
| 最適化 | 非線形カーブ・フィッティングと回帰 |
| 信号処理 | 信号処理，フィルタリング，周波数解析 |
| スプライン | カーブ・フィッティングと回帰 |
| 統計 | 統計解析，非線形カーブ・フィッティングと回帰 |
| システム同定 | パラメトリック/ARMA モデリング |
| 波動 | 波動解析 |

## 7.2　記述統計

MATLAB には記述統計に関する関数が用意されています．関数は，平均値，最大値，最小値，標準偏差，中央値，レンジの計算を提供します．基本的に列方向にデータ解析を行ないます（表 7.3，図 7.4，図 7.5）．

ここでは交通量サンプル・データセットとして，24 時間に渡り，3 地点で観測した交通量（表 7.2）を使用します．このデータをファイル名 count.dat を付けて保存します．

表 7.2　3 地点の交通量サンプル

| 時刻 | 第 1 | 第 2 | 第 3 | 時刻 | 第 1 | 第 2 | 第 3 |
|---|---|---|---|---|---|---|---|
| 01h00 | 11 | 11 | 9 | 13h00 | 18 | 19 | 29 |
| 02h00 | 7 | 13 | 11 | 14h00 | 17 | 15 | 18 |
| 03h00 | 14 | 17 | 20 | 15h00 | 32 | 36 | 48 |
| 04h00 | 11 | 13 | 9 | 16h00 | 42 | 47 | 10 |
| 05h00 | 43 | 51 | 69 | 17h00 | 57 | 65 | 92 |
| 06h00 | 38 | 46 | 76 | 18h00 | 44 | 66 | 151 |
| 07h00 | 61 | 132 | 186 | 19h00 | 44 | 55 | 90 |
| 08h00 | 75 | 135 | 180 | 20h00 | 114 | 145 | 257 |
| 09h00 | 38 | 88 | 115 | 21h00 | 35 | 58 | 68 |
| 10h00 | 28 | 36 | 55 | 22h00 | 11 | 12 | 15 |
| 11h00 | 12 | 12 | 14 | 23h00 | 13 | 9 | 15 |
| 12h00 | 18 | 27 | 30 | 24h00 | 10 | 9 | 7 |

表 7.3　計算結果の統計値

| 交通量統計値 | 第 1 地点 | 第 2 地点 | 第 3 地点 |
|---|---|---|---|
| 平均値　mean | 33.0417 | 46.5417 | 65.5833 |
| 最大値　max | 114 | 145 | 257 |
| 最小値　min | 7 | 9 | 7 |
| 標準偏差　std | 25.3265 | 41.4057 | 68.0281 |
| 中央値 median | 30.0000 | 36.0000 | 39.0000 |
| レンジ　range | 107 | 136 | 250 |
| サイズ | 24 | 24 | 24 |

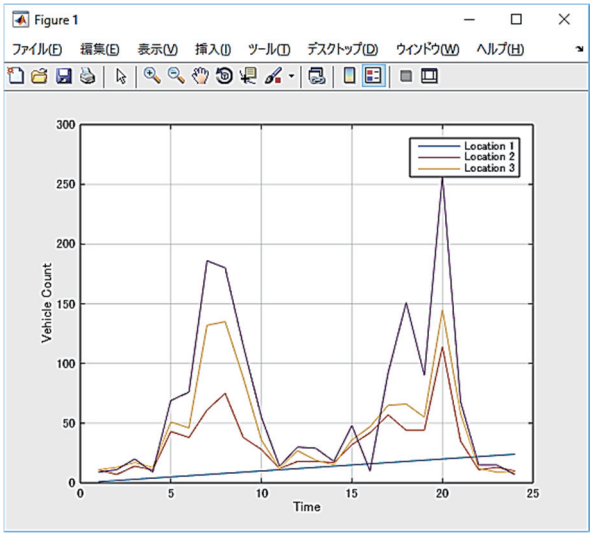



図 7.4　交通量グラフ

```
>> load count.dat
>> [n,p]=size(count)
n =
    24
p =
     4
>> t=1:n;
>> set(0,'defaultaxeslinestyleorder','-|--|-.')
>> set(0,'defaultaxescolororder',[0 0 0])
>> plot(t,count),legend('Location 1','Location 2','Location 3',0)
>> xlabel('Time'), ylabel('Vehicle Count'), grid on
>> meanv=mean(count), maxv=max(count), minv=min(count)
meanv =
   12.5000   33.0417   46.5417   65.5833
maxv =
    24   114   145   257
minv =
     1     7     9     7
>> stdev=std(count), cntrv=median(count),rangev=range(count)
stdev =
    7.0711   25.3265   41.4057   68.0281
cntrv =
   12.5000   30.0000   36.0000   39.0000
rangev =
    23   107   136   250
```

図 7.5　データ・ファイル count. dat を読み込んで交通量グラフ表示と記述統計の計算

## 7.3　回帰分析カーブ・フィッティング

いま，サンプル・データをプロットしたものが，図 7.7 です．このデータに合う多項式関数

$$y = a_0 + a_1 t + a_2 t^2$$

の係数 $a_0, a_1, a_2$ を，最小 2 乗法を用いて求め，これを $0 \le t \le 2.5$ の範囲で，0.1 間隔で計算して，サンプル・データと重ねて表示したグラフが図 7.8 です．

このMATLABコードが次の図7.6です.

```
>> t = [0 .3 .8 1.1 1.6 2.3]';
>> y = [0.5 0.82 1.14 1.25 1.35 1.40]';
>> plot(t,y,'o'), grid on
>> X = [ones(size(t))  t  t.^2]
X =
    1.0000         0         0
    1.0000    0.3000    0.0900
    1.0000    0.8000    0.6400
    1.0000    1.1000    1.2100
    1.0000    1.6000    2.5600
    1.0000    2.3000    5.2900
>> a = X¥y
a =
    0.5318
    0.9191
   -0.2387
>> T = (0:0.1:2.5)';
>> Y = [ones(size(T))  T  T.^2]*a;
>> plot(T,Y,'-',t,y,'o'), grid on
```

図7.6　サンプル・データを最小2乗法で多項式近似

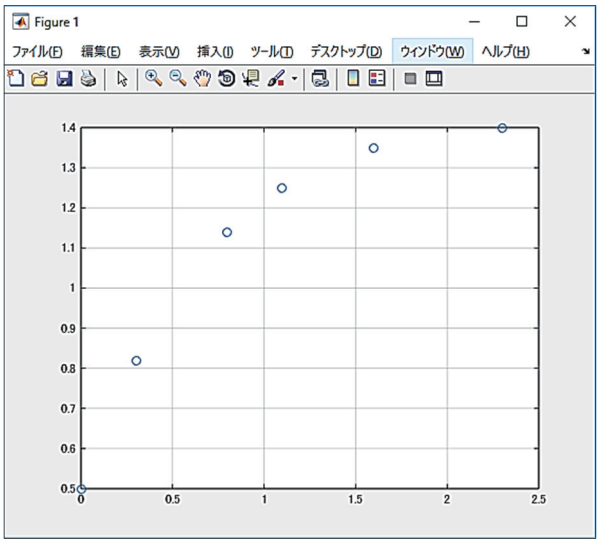



図7.7　サンプル・データのプロット

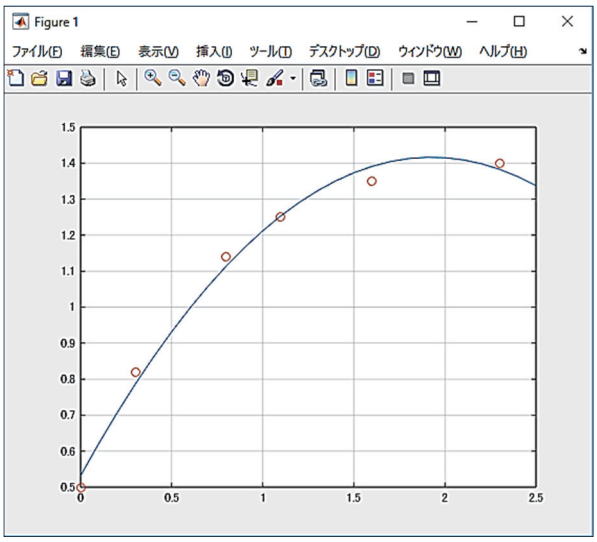



図7.8　2次多項式で近似したグラフ

# 8．MATLABグラフィックスと可視化

## 8.1　グラフィックス

MATLABは，実験データやシミュレーション結果をグラフ表示したり，同じウィンドウの中に複数のプロットを表示したり，3次元メッシュやサーフェス表示，画像表示などを自由に取り扱うことができます.

### （1）グラフの作成

2次元グラフを作成するには，x軸方向のデータとそのxに対応するy軸方向のデータがあれば，plotというデータを描く関数でデータを表示するだけです.グラフの背景に格子表示したり，軸の目盛りを変更したり，描画にマーク（印）を使ったり，カラー表示する作業は，正常なグラフを表示した後で行いましょう.

ここで，$0 \le x \le 4\pi$ の範囲で，

$$y = \sin(x)$$
$$y1 = \cos(x)$$
$$y2 = \sin(x - 0.5)$$
$$y3 = \cos(2x)$$
$$y4 = \sin(x) + \cos(2x)$$

を，π/90 毎に4πラジアンまで，描画してみましょう．まず，$y = \sin(x)$ を描きます（図8.1）.

続いて，3つのグラフを重ねて表示してみましょう（図8.2～3）.

```
>> x=0:pi/90:4*pi;
>> y=sin(x);
>> y1=cos(x);
>> y2=sin(x-0.5);
>> y3=cos(2*x);
>> y4=sin(x)+cos(2*x);
>> plot(x,y), grid on      <==== 図8.2
>> plot(x,y,x,y2,x,y3), grid on      <==== 図8.3
>> plot(x,y,x,y3,x,y4), grid on      <==== 図8.4
```

図8.1　SIN，COS関数をグラフ化するコード

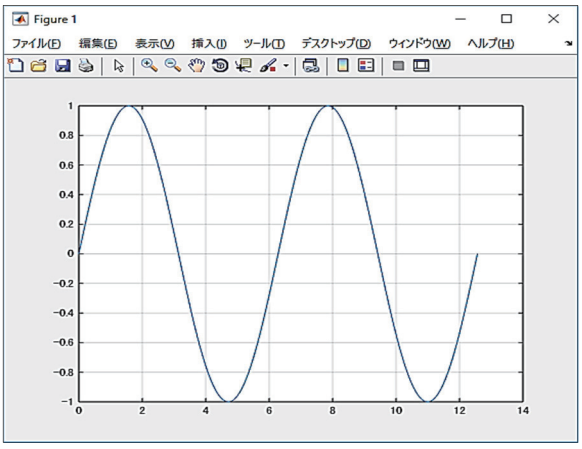



図8.2　　SINカーブ

続いて，3つのグラフを重ねて表示してみましょう

(図 8.3～4).

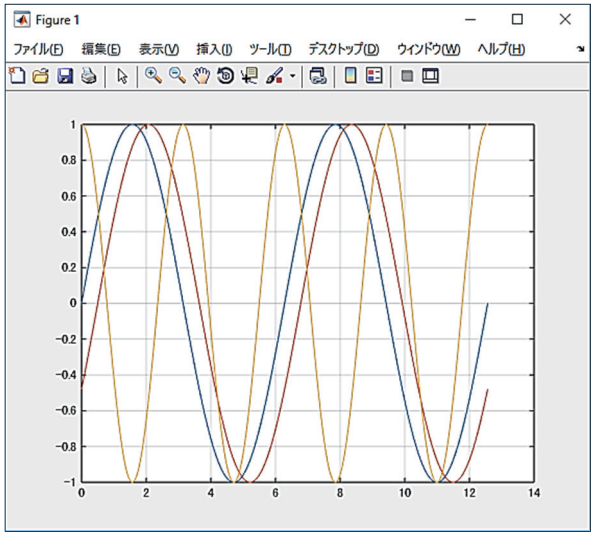



図 8.3　sin(x), sin(x-0.5), cos(2x) カーブ

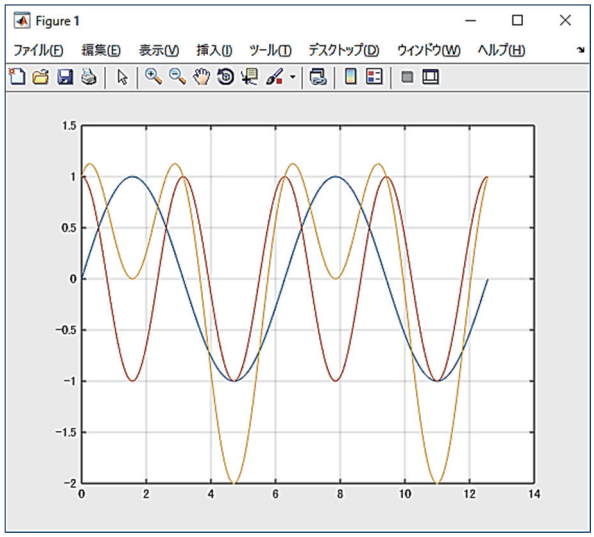



図 8.4　sin(x), cos(2x), sin(x)+cos(2x) カーブ

続いて，ラインの型，軸の目盛りの変更，軸名，グラフのタイトルを描いてみましょう（図 8.5）.

```
>> plot(x, y, '-', x, y3, '--', x, y4, '+'), grid on
          <=== 図 8.6　ラインの型を変更
>> set(gca,'XTick',0:pi/2:4*pi)
>> set(gca,'XTicklabel',{'0','pi/2','pi'
       ,'3*pi/2','2*pi','5*pi/2','3*pi',
       '7*pi/2','4*pi'})
>> xlabel('0 <= x <= 4pi')
>> ylabel('sin(¥theta)')
>> title('Plot of sin(x), cos(2x), sin(x)+cos(2x)')
          <=== 図.8.7　軸の目盛りの変更,
                軸名，グラフのタイトル
```

図 8.5　SIN, COS 関数グラフを修飾するコード

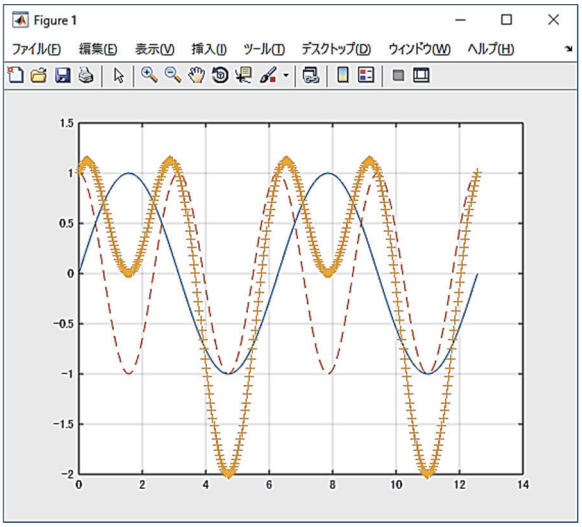



図 8.6　x, y3, y4 のライン型を変更

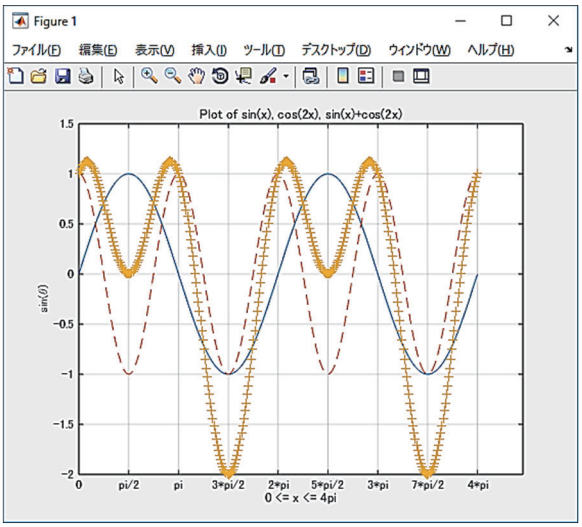



図 8.7　軸の目盛り，軸名，グラフのタイトルを追加

## 8.2　いろいろなグラフ

### （1）円グラフ（Pie Chart）

```
>> y=[1 2 3 4 5];
>> pie(y);
```

図 8.8　円グラフのコード

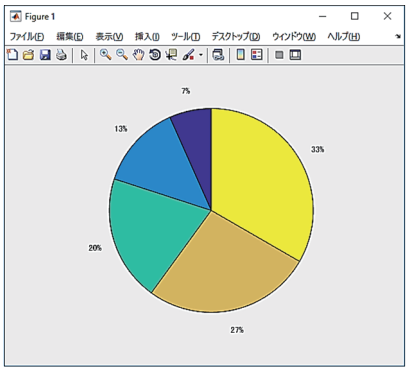



図 8.9　円グラフの表示

（2）棒グラフ（Bar Chart）と積み重ね

```
>> y=rand(5,3);
>> subplot(2,2,1); bar(y); title 'Virtical'
>> subplot(2,2,2); bar(y,'Stacked'); title 'Stack'
>> subplot(2,2,3); barh(y); title 'Horizontal'
>> subplot(2,2,4); barh(y,'Stacked'); title 'Stack'
```

図 8.10　棒グラフ，横棒グラフと積み重ねのコード

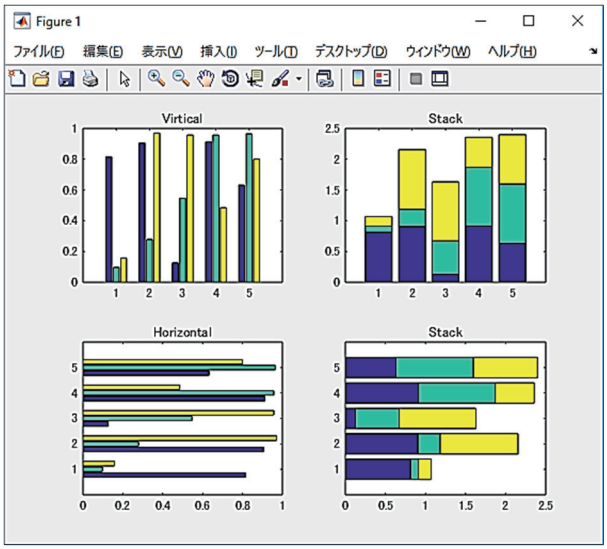



図 8.11　棒グラフ，横棒グラフと積み重ねのグラフ

（3）ベクトル場の表示

```
>> xx= -2:0.2:2; yy=-1.4:0.2:1.4;
>> [x,y]=meshgrid(xx,yy); z=exp(-(x.^2+y.^2));
>> [u,v]=gradient(z);
>> quiver(x,y,u,v)
>> quiver(x,y,u,v)
>> xlabel('x'); ylabel('y')
```

図 8.12　ベクトル場のような振動表示のコード

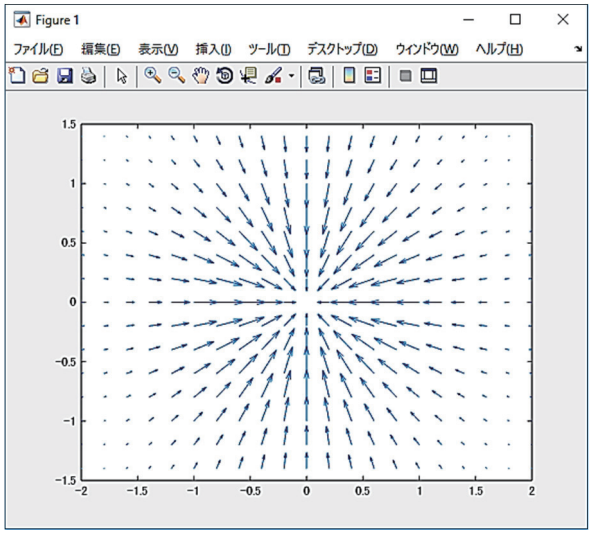



図 8.13　ベクトル場のような振動表示

## 8.3　3次元グラフィックス

3次元関数　$z = e^{-(x^2+y^2)}$

を $-2 \le x \le 2$，$-1.4 \le y \le 1.4$ の範囲で，メッシュ形式でカラー表示してみましょう（図 8.14）．

```
>> xx= -2:0.2:2; yy=-1.4:0.2:1.4;
>> [x,y]=meshgrid(xx,yy);
>> z=exp(-(x.^2+y.^2));
>> mesh(x,y,z)
>> xlabel('x'); ylabel('y'); zlabel('z')
>> colorbar
```

図 8.14　3次元メッシュ形式表示するコード

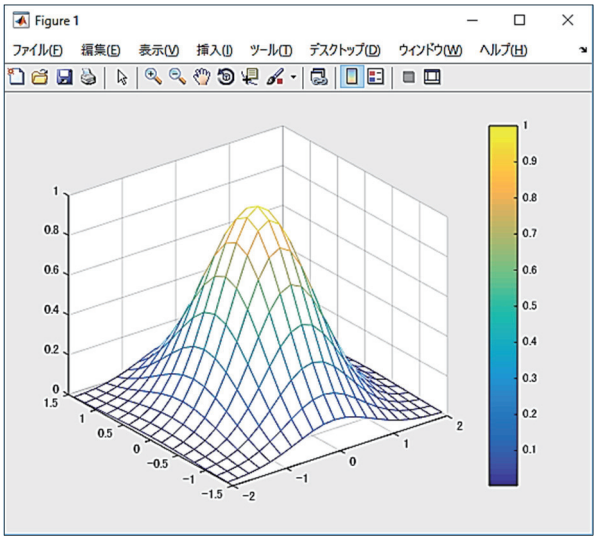



図 8.15　3次元メッシュ形式表示

## 9．演習問題

### 問題1　ベクトルと2次方程式の解

次の5組の2次方程式を，MATLAB を使用してできるだけ簡潔に解き，経過を記録します．この記録ファイルに以下のファイル名付けてみましょう．

$$x^2 + 2x + 3 = 0$$
$$2x^2 + 3x + 4 = 0$$
$$3x^2 + 4x + 5 = 0$$
$$4x^2 + 5x + 6 = 0$$
$$5x^2 + 6x + 7 = 0$$

ファイル名＝ユーザー名_MATLAB_1_99.txt

（99 は版番号で 00 から開始）

### 問題2　連立方程式の解法

次の2組の連立方程式を，MATLAB を用いて2つ以上の方法で簡潔に解き，その解を検算し，経過を記録します．以下のファイル名を付けてみましょう．

ファイル名＝ユーザー名_MATLAB_2_X_99.txt

（99 は版番号で 00 から開始，X は 1 または 2）

（1）４元連立方程式

$$\begin{pmatrix} -4 & 1 & 1 & 2 \\ 3 & 2 & -1 & 1 \\ -1 & 3 & 2 & -6 \\ 2 & -1 & 3 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9 \\ 8 \\ -13 \\ 17 \end{pmatrix}$$

（2）５元連立方程式

$$x_1 + x_2 + x_3 + 2x_4 + 3x_5 = 8$$
$$5x_1 + 0x_2 + 0x_3 + x_4 + 4x_5 = 8$$
$$x_1 + 2x_2 + 3x_3 + 4x_4 + 0x_5 = 10$$
$$2x_1 + 4x_2 + x_3 + 4x_4 + x_5 = 12$$
$$3x_1 + x_2 + 7x_3 + 5x_4 + 3x_5 = 19$$

## 問題３　３次スプライン補間

次のベクトル $x_i$ と $y_i$ のデータ作り，データ点に“＋”印でプロットし，$x_i(i = 1{,}16)$ における３次スプライン補間を spline 関数と（hold on）を使って，スプラインのグラフを描きなさい．次にｎ＝６次とｎ＝９次の多項式フィッティングで，ployfit を使って係数を求め，$x_i$ 値に対する $y_i$ 値を求め，このグラフに追加しなさい．

この係数グラフを Word に貼り付けて，次のファイル名を付けてみましょう．

ファイル名＝ユーザー名_MATLAB_3_99.docx

（99 は版番号で 00 から開始）

## 問題４　２次元グラフ

次の式で表されるデータを $0 \le x \le 4\pi$ まで求め，$x$ 軸は 1/４πおきに，ｙ軸は 0.25 ごとにグリッド表示し，グラフは重ねて表示します．表示された図形「Figure１」を Word に貼り付け，提出日付，ユーザー名，氏名を付記し，次のファイル名を付けて，WebCT に Upload して提出して下さい．

$$y1 = \sin(x),$$
$$y2 = 3\sin(x) + \cos(x)$$
$$y3 = 2\sin(2x) + 2\cos(x)$$

ファイル名＝ユーザー名_MATLAB_4_99.docx

（99 は版番号で 00 から開始）

## 問題５　３Ｄグラフ

$-5 \le x, y \le 5$ のとき，0.25 おきのメッシュ・グリッド上で $z = \sin\sqrt{x^2 + y^2}$ を MATLAB の３次元グラフィックスで表示します．通常，カーボーイ・ハット呼ばれる帽子の形が表示されます（図 9.1）．Figure１を Word に貼り付けます．次に３次元曲面オブジェクトを作成し，図 9.2 のように回転させて Figure１をコピーし，Word に貼り付けます．この Word に提出日付，ユーザー名，氏名を付記し，次のファイル名を付けてみましょう．

ファイル名＝ユーザー名_MATLAB_5_99.docx

（99 は版番号で 00 から開始）

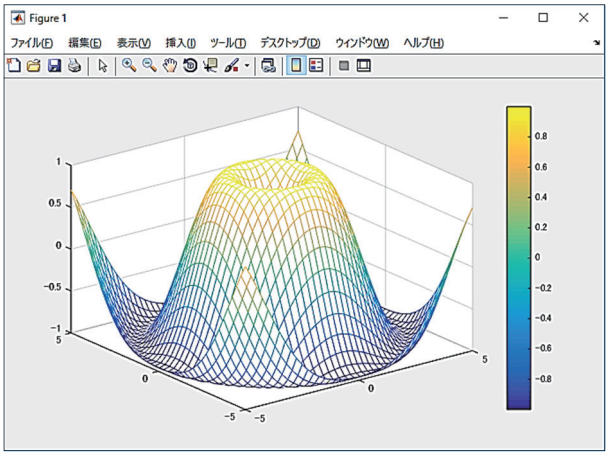



図 9.1　３次元メッシュ形式図形

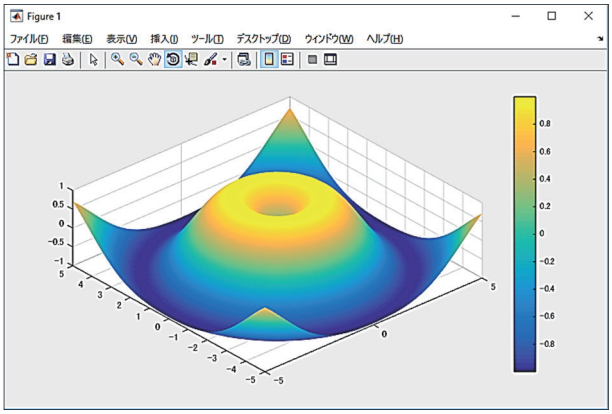



図 9.2　３次元曲面オブジェクトの回転図形

## 参考文献

［1］MATLAB ハンドブック，小林一行著，秀和システム，2000.11.01，ISBN4-87966-772-2

［2］はやわかり MATLAB，芦野隆一，レミ・ヴァイアンクール著，共立出版，1997.07.10，ISBN4-320-02875-9

［3］MATLAB と利用の実際―現代の応用数学と CG―，小国力著，サイエンス社，1995.04.25，ISBN4-7819-0763-6

［4］http://jp.mathworks.com/help/matlab/index.html

［5］http://web.cecs.pdx.edu/~gerry/MATLAB/