

MATLAB/Simulink

简单マニュアル

北海学園大学工学部電子情報工学科

(2010年版)

第0章 MATLABの使用にあたって

0.1 起動と終了

起動は、MATLABのアイコンをクリックする。起動すると、**図 0.1** のMATLABの起動画面が開く。MATLABでは、種々のコマンドはコマンドウィンドウ (Command Window) から実行する。この画面の左端に現れる「>>」は「MATLABプロンプト」と呼ばれ、コマンドはここに入力する。また、結果もここに返される。なお、終了は、右上の×をクリックするか、あるいは >> exit を入力する。

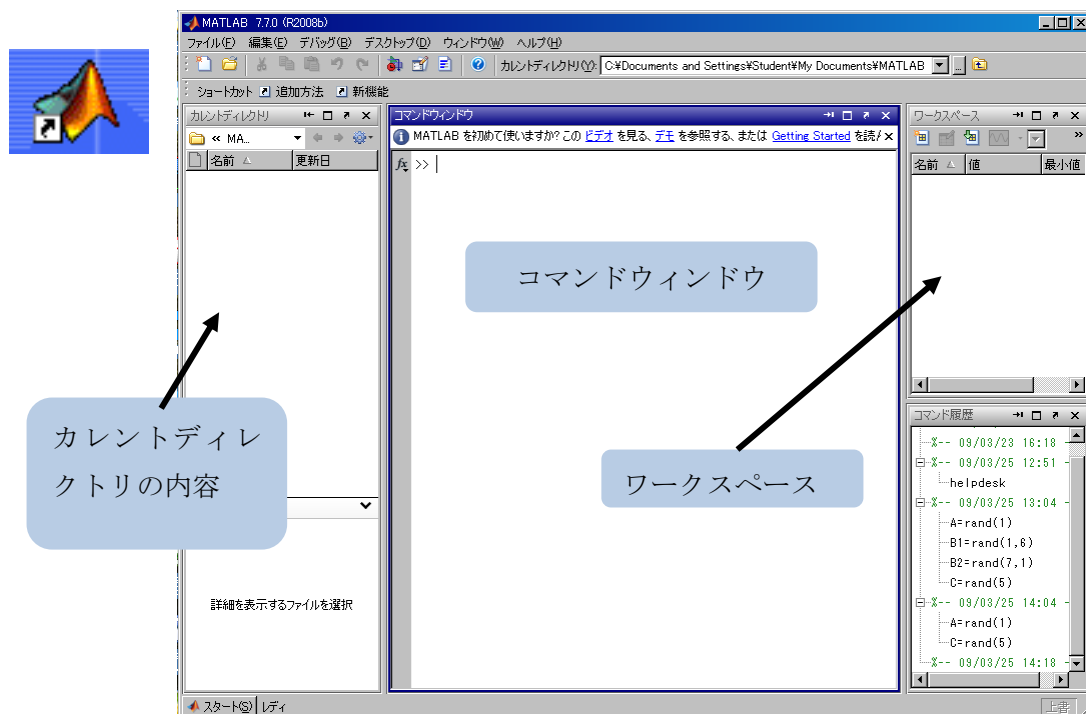


図 0.1 MATLAB のアイコン (左上) と起動画面の MATLAB デスクトップ (右) :
Version 7.7.0 (R2008b)

MATLAB デスクトップでは、コマンドウィンドウの他に次のウィンドウ (これらをデスクトップツールという) が利用できる。すなわち、

- ワークスペース (Workspace)
 - 自動管理される変数の属性を表示するウィンドウ (⇒第 2 章)
- カレント・ディレクトリ (Current Directory)
 - 作業をしているディレクトリ (フォルダ) 内のファイルの一覧を示すウィンドウ

- ・ コマンドヒストリ (Command History)

..... コマンドウィンドウで実行したコマンドの履歴を示すウィンドウ

0.2 カレント・ディレクトリの変更・設定

MATLAB では、コマンドウィンドウのツールバーにカレント・ディレクトリ欄がある。カレント・ディレクトリとは、「いま、作業をしているディレクトリ (フォルダ)」という意味であり、現実には、作成したプログラムやデータが納められるフォルダである。したがって、カレント・ディレクトリを各自の USB メモリ内の希望するフォルダに設定すると、間違いなく自分の作成したプログラムやデータは自分のフォルダに保存される。

カレント・ディレクトリの設定の手順を図 0.2 に示す。カレント・ディレクトリが自分のフォルダに設定されると、図 0.2 の左側にみられるように、フォルダ内のファイル一覧が、デスクトップツールであるカレント・ディレクトリに表示される。

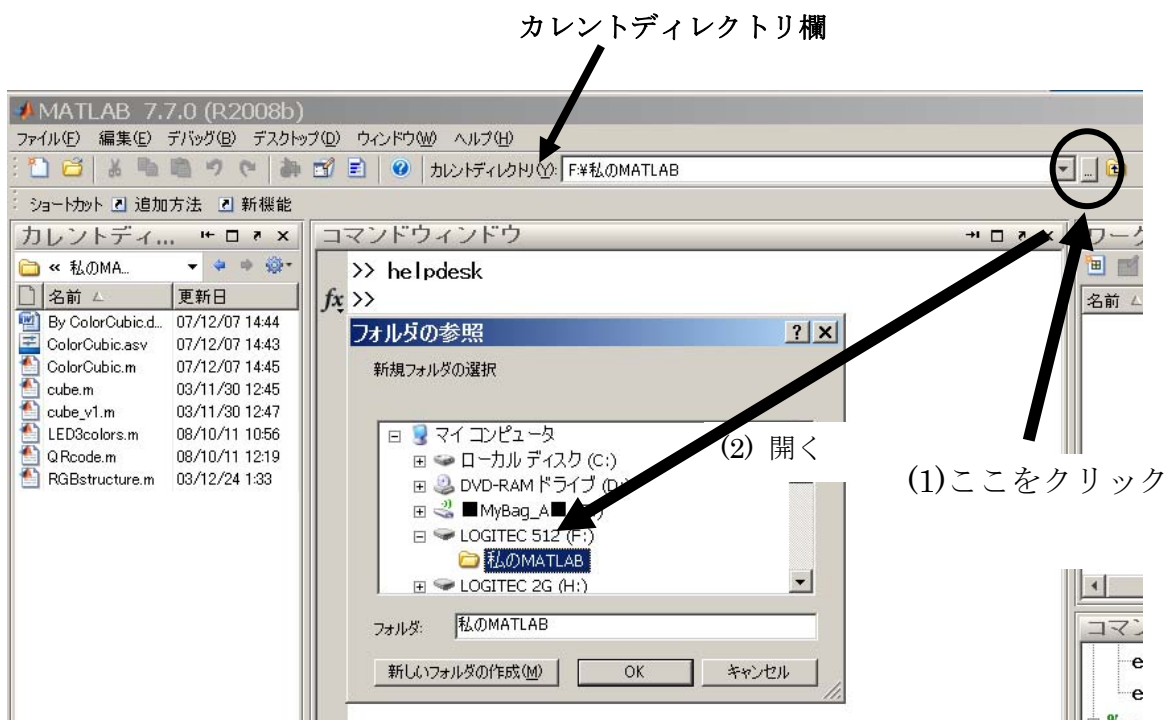


図 0.2 カレントディレクトリの変更・設定. 図の手順で、「フォルダの参照」を開いて各自の USB メモリ内のフォルダを指定する.

0.3 MATLAB デスクトップの扱い方

デフォルトでは、MATLAB を起動すると、図 0.1 の配置の MATLAB デスクトップになっている。しかし、ワークスペースのようなデスクトップツールは、各自の使い勝手に合わせて図 0.3 のように個別に取り出すことができる。

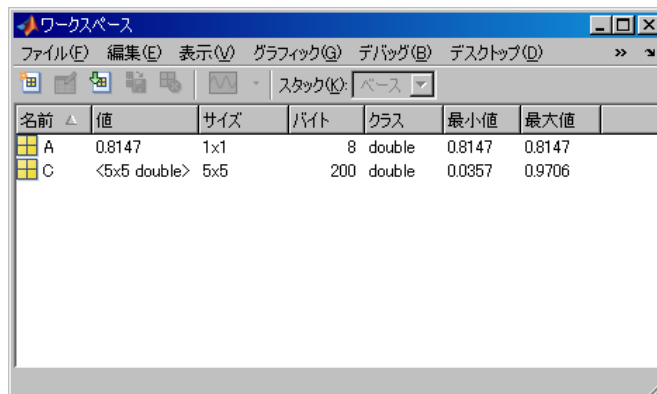


図 0.3 MATLAB デスクトップから飛び出たワークスペース

図 0.4 は、ワークスペースのウィンドウが飛び出たあとの MATLAB デスクトップのレイアウトで、この配置は使う人の都合

でいろいろなレイアウトになる。これをデフォルトの配置に戻すには、メニューバーから

「デスクトップ」 → 「デスクトップのレイアウト」 → 「デフォルト」

と設定する。

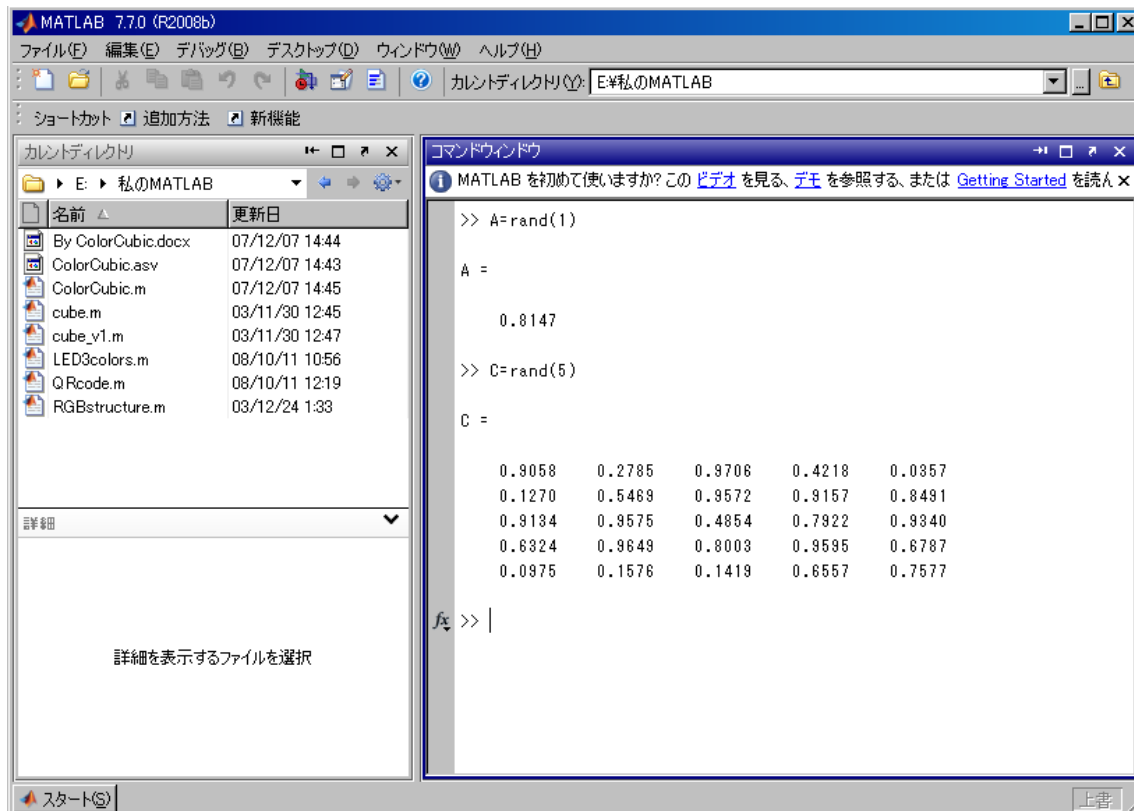


図 0.4 MATLAB デスクトップのひとつのレイアウト

0.4 ヘルプの使い方

MATLAB では、ヘルプが充実している。このヘルプ機能を体験するために、つぎの実習 0.1 を実行してみよう。

実習 0.1 MATLAB を起動し、コマンドウィンドウで次の演算を実行しなさい。

- (1) `» 1+2+3`
- (2) `» pi`
- (3) `» x=0/0`
- (4) `» y=1/0`

この実行結果では、ans, pi, NaN, Inf という表 0.1 に示す MATLAB 特有の予約変数や定数が出てくる。

ans	直前のコマンドの答え
i, j	虚数単位
Inf, inf	∞ (無限大)
NaN	Not-a-number (不定値)
pi	π (3.14159265358979)

表 0.1 MATLAB の予約定数・変数

ここにみられるような意味不明な記号やコマンドが現れたときには、表 0.1 を知らなければならぬかという、そうではなく MATLAB のヘルプ機能を使って意味を知ることができる。たとえば、pi の意味が知りたいときには、コマンドウィンドウから

```
» help pi
```


を実行すると、

```
PI      3.1415926535897....
```

が表示され、それが円周率 π であることがわかる。

また、コマンドウィンドウから

```
» helpdesk
```

と入力すると  0.5 のヘルプ画面が現れる。この起動は、メニューバーの「ヘルプ」からも起動できる。そして、ここから全ての MATLAB 関数の意味や利用法を分野別および関数名から知ることができる。

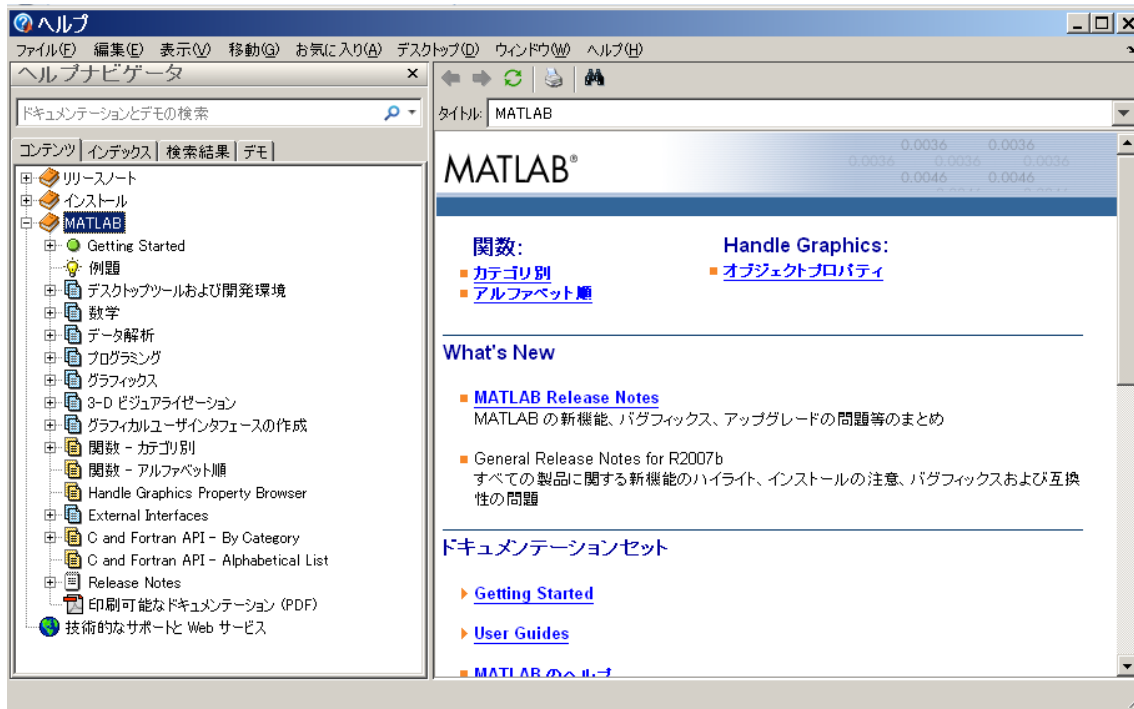


図 0.5 MATLAB 日本語版ヘルプの起動画面 (R2007)

知っているると便利！

- コマンドウィンドウから `help pi` を実行したあとで、矢印キーの上向きキー「↑」を押してみよう。過去に実行したコマンドが現れる。これを利用して、コマンドをキー入力せずに再実行できます。
- コマンドの実行は、カレント・ディレクトリにあるファイル（M ファイル）からもできます。そこにあるファイルを右クリックし実行することができます。
- また、コマンド履歴にある履歴からも実行できます。たとえば、そこに `help pi` があれば、それをクリックしてみると分かります。

第1章 変数と演算

1.1 スカラー、ベクトル、行列

MATLAB には、一様乱数を発生する関数 `rand` がある。この詳細には触れないが、これを用いて MATLAB の変数の扱いを説明する。

実習 1.1 Command Window から以下の結果を確認せよ。

- (1) `》 A=rand(1)`
- (2) `》 B1=rand(1,6)`
- (3) `》 B2=rand(7,1)`
- (4) `》 C=rand(5)`

(1)の A は1個の数で、これをスカラー（数）という。(2)の B1 は行（横）ベクトル、(3)の B2 は列（縦）ベクトル、(4)の C は、 5×5 のサイズの行列である。つまり、

- ・ スカラー数： ふつうの1個の数
- ・ ベクトル： 電流を変えたときの出力のデータや、電圧を変えたときの角度などで、グラフに表示できる複数の数の組。
- ・ 行列： 数値の配列で、数表のデータや画像データは行列で表現される。行列が一般的で、スカラーは 1×1 の行列、ベクトルは $1 \times m$ (あるいは、 $m \times 1$) の行列である。

実習 1.2 スカラー数に関して以下の演算を試みよ。

- (1) `》 a=5;`
`》 b=60;`
`》 A=30;`
`》 X=a*b/A`
- (2) `》 r=5; S=pi*r^2`
- (3) `》 x=sin(pi/2)`

(説明)

(1)では、`a`、`b`、`A` が定義され、`X=a*b/A` が計算される。このとき、後ろにセミコロンの「;」がついた `a`、`b`、`A` の値は表示されない。セミicolonがない `X` の値だけが表示される。また、これらは `》 a=5; b=60; A=30; X=a*b/A` と1行に書いてもよい。

(2)は、半径 `r` の円の面積 `S` の計算で、予約定数 `pi` が使われている。

(3)は、`sin(90°)` の計算である。

以上のことから、つぎのことが分かる。

- MATLAB では、単に変数名のみをかくと、その変数の数値が表示される。表示する必要がない変数には、変数名の直後にセミコロン「;」を付ける。
- 大文字と小文字の変数名は区別される。つまり、`x` と `X` は同じ変数ではない。また、変数として認識できるのは、31文字までの変数名である。
- MATLAB には、あらかじめ定義されている定数（あるいは変数）がある（表 0.1）。これらは、特別の事情がない限り変数名として使用しない方がよい。
- 表 1.1 にあるような、三角関数や対数などの演算は、`》x=sin(pi/2)` あるいは `》y=log(x)` のように、数値や変数を `()` に入れてかく。

分類	関数名	記述例
三角関数	<code>sin</code> (正弦), <code>cos</code> (余弦), <code>tan</code> (正接), <code>csc</code> (余割), <code>sec</code> (正割), <code>cot</code> (余接)	<code>Y=sin(X)</code>
指数・対数関数	<code>exp</code> (指数), <code>log</code> (自然対数), <code>log2</code> (底 2 の対数), <code>log10</code> (常用対数)	<code>Y=exp(X)</code>
絶対値・平方根	<code>abs</code> (絶対値), <code>sqrt</code> (平方根)	<code>Y=sqrt(X)</code>

表 1.1 初等数学関数 (一部)

+, -などの演算記号は電卓などで用いられるものとほぼ同じであるが、行列演算関数と呼ばれる (表 1.2)。この中で、左割りとは、`5¥45=9` のように左の数で右の数を割る演算である。

+	加算
-	減算
*	乗算
/	わり算
¥	左割り
^	べき乗

表 1.2 基本的な行列演算関数

なお、表 1.2 の行列演算関数はスカラー数だけではなく、一般の行列にも用いられる。

1.2 ベクトルの定義

等間隔なベクトルは、コロン「:」を用いて定義する。

実習 1.3 次のコマンドを実行し、等間隔ベクトルが定義されることを確認せよ。

- (1) `》v1=1:10`
- (2) `》v2=-12:4:10`
- (3) `》v3=-2.5:0.5:1.7`

一般的には、等間隔なベクトルは、

$$v = (\text{初期値}) : (\text{間隔}) : (\text{最終値})$$

で定義される。ただし、間隔が1のときには、 $v1$ のように（間隔）は省略できる。また、 $v3$ のように、（初期値）や（最終値）は0や負でもよく、また整数でなくともよい。

等間隔ベクトルは、また、関数 `linspace` を用いて定義することもできる。

実習 1.4 次のコマンドを実行し、6個の要素からなる等間隔ベクトルが定義されることを確認せよ。

(1) `》 v=linspace(50,80, 6)`

ここでの関数 `linspace` は、はじめから要素数が決まっているときに用いられる。いまの場合、初期値が50、最終値が80で、その間に要素数が6個の等間隔ベクトルを定義する。このように、等間隔ベクトルは表 1.3 に示す2通りの方法で定義できる。

関数名など	機能	記述例
:	等間隔ベクトル	$x=m1:m2$ (間隔1のベクトル, 初期値 $m1$, 最終値 $m2$) $x=m1:a:m2$ (初期値 $m1$, 間隔 a , 最終値 $m2$)
<code>linspace</code>	等間隔のベクトル (要素数が与えられたとき)	$x=linspace(m1,m2,n)$ (初期値 $m1$, 最終値 $m2$, 個数 n)

表 1.3 等間隔ベクトルを与える2つの関数

等間隔ベクトルの有難味は、実習 2.1 でよく分かります！

つぎに、測定データなどをキーボードから変数 x として定義し、それをグラフにプロットする場合を考える。

実習 1.5 次の5ステップを実行せよ。

- (1) `》 x=[2.3, 3.4,5.5,8.1,.....
17.3, 29.5, 50.4];`
- (2) `》 x(1)`
- (3) `》 x(end)`
- (4) `》 n=length(x)`
- (5) `》 plot(x)`

(説明)

- ・ベクトルは要素全体を[.....]とカッコで囲んで、
変数名=[ベクトル要素]

の形で定義する。

- ・要素間は、ブランク (空欄) またはカンマ「,」で区切る。
- ・長い定義文などのときには、ピリオドを「...」と3つ (あるいは、それ以上) 付けると、つぎの行に継続できる。
- ・定義文の最後にセミコロン「;」をつけて実行すると、結果は画面表示されない。
- ・配列要素は、通常のように、x(1), x(2)... と指定する。
- ・x(end)は、配列の最終要素。この end は MATLAB の特殊用法である。
- ・length(x)は、x の要素数を与える (⇒1.6 節)。
- ・plot(x)は、配列番号を横軸にとって、x のグラフを描く (⇒第4章)。

1.3 行列の定義

行列も、要素全体を[.....]とカッコで囲んで、
変数名=[行列要素]

の形で定義する。このとき、行列要素は、

[行列要素]=[1行目のベクトル要素 ; 2行目のベクトル要素 ;]

のように、ベクトル要素をセミコロン「;」で区切ってかく。

実習 1.6 つぎの2つを実行し、A と B の行列が同一であることを確認せよ。

- (1) » A=[1,2,3;4,5,6;7,8,9]
- (2) » B=[1:3; 4:6; 7:9]

MATLAB には、種々の行列やベクトルの作成に便利な「一般行列関数」が用意されている (表 1.4)。これらを用いると、種々の行列が効率的に作成できる。

なお、表 1.4 において、引数を一つだけ指定すると、それで指定されたサイズの2次元正方行列が得られる。

関数名	機能	記述例
ones	すべての要素が1の行列	X=ones(m,n) X=ones(m) (m×mの正方行列)
zeros	すべての要素が0の行列	X=zeros(m,n) X=zeros(m) (m×mの正方行列)
eye	単位行列	X=eye(m,n) X=eye(m) (m×mの正方行列)
diag	対角行列	X=diag([x1,x2,...,xn]) ([x1,x2,...,xn] は対角要素ベクトル)
rand	0と1の間の一様乱数	X=rand(m,n) X=rand(m) (m×mの正方行列)

randn	平均値が 0, 標準偏差が 1 の正規 (ガウス) 乱数行列	X=randn(m,n) X=randn(m) (m×m の正方行列)
-------	-----------------------------------	--

表 1.4 代表的な行列を与える一般行列関数

実習 1.7 以下を実行して, 関数 zeros と関数 diag を確認せよ.

- (1) » A=zeros(3); A(1,1)=1; A(2,2)=2; A(3,3)=3;
を定義せよ.
- (2) » B=diag([1,2,3]);
を定義せよ.
- (3) » A, および » B と入力し, 両者が同一の行列であることを確認しなさい.

1.4 行列の演算

実習 1.8 以下を実行して, 2 行 2 列の簡単な行列演算を確かめよ.

- (1) » A=[1,0;2,3] を定義せよ.
- (2) » B=[0,2;1,4] を定義せよ.
- (3) » X=A+B を求めよ.
- (4) » Y=A*B を求めよ.
- (5) » Z=A.*B を求めよ.

(説明)

$$A = \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 2 \\ 1 & 4 \end{pmatrix}$$

の行列である. したがって, (3)は

$$X = \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} + \begin{pmatrix} 0 & 2 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 1+0 & 0+2 \\ 2+1 & 3+4 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 3 & 7 \end{pmatrix}$$

となる. このように, 2つの行列の和 (あるいは差) は, 要素ごとの和 (差) として得られる.

(4)の行列の積 Y は

$$Y = \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} * \begin{pmatrix} 0 & 2 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 \times 0 + 0 \times 1 & 1 \times 2 + 0 \times 4 \\ 2 \times 0 + 3 \times 1 & 2 \times 2 + 3 \times 4 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 3 & 16 \end{pmatrix}$$

である.

一方, (5)の Z をあたえるピリオドが付いた積「.*」は, MATLAB 独特なもので

$$Z = \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} .* \begin{pmatrix} 0 & 2 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 \times 0 & 0 \times 2 \\ 2 \times 1 & 3 \times 4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 2 & 12 \end{pmatrix}$$

を与える. つまり, 対応する要素ごとの積からなる行列を結果する. この演算は, 画像処理などにおいてよく用いられる.

除算の場合も, 要素ごとのわり算のときにはピリオドを付けた演算子「./」を用いる. また, べき乗の演算も同様で,

$$A^2 = \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 \times 1 + 0 \times 2 & 1 \times 0 + 0 \times 3 \\ 2 \times 1 + 3 \times 2 & 3 \times 0 + 3 \times 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 8 & 9 \end{pmatrix}$$

であり、

$$A.^2 = \begin{pmatrix} 1 \times 1 & 0 \times 0 \\ 2 \times 2 & 3 \times 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 4 & 9 \end{pmatrix}$$

である。このように、ピリオド「.」があるか無いかで結果が全く異なるので注意しなければならない。

1.5 行列操作関数

MATLAB には、行列の要素数を求めたり、行列を変形するための行列操作関数がある（表 1.5）。

関数名	行列操作の意味	記述例
length	ベクトルの長さ（要素数）	n=length(v)
size	行列の大きさ（配列数）	n=size(X) あるいは [m,n]=size(X)
:	行列要素のインデックス (行あるいは列のすべての意味)	v=X(3,:) (ベクトル v は行列 X の 3 行目)
reshape	行列の大きさの変更	Y=reshape(X,m,n) (X を m 行 n 列に変換)
flipud	行列要素の上下の入れ替え	Y=flipud(X)
fliplr	行列要素の左右の入れ替え	Y=fliplr(X)
'	行と列の入れ替え（転置）	Y=X'
inv	逆行列	Y=inv(X)
conj	複素数行列の要素の複素共役	Y=conj(X)
rot90	行列の 90° 回転	Y=rot90(X)

表 1.5 代表的な行列操作関数

実習 1.9 以下を実行し、行列操作関数を理解せよ。

- (1) ベクトル v=1:12 を定義しなさい。
- (2) n=length(v)を確認しなさい。
- (3) V=reshape(v,3,4)を試みよ。
- (4) m=size(V)を試みよ。
- (5) V1=fliplr(V)を試みよ。

第2章 変数の管理とデータの型

MATLAB では、使用する変数の型や配列数をあらかじめ宣言する煩わしさが無い。これは、変数を定義するたびに MATLAB が自動的に変数にメモリを割り当てて管理を行っているからである。これをみるために、次の実習問題からはじめる。

実習 2.1 以下を実行せよ。

- ```
(1) >> clear; close all; x=-40:40; y=sin(2*pi*x/20); plot(x,y)
(2) >> clear; close all; x=-5:0.1:5; y=x.^2; plot(x,y)
(3) >> clear; close all; x=-5:0.1:5; y=exp(-x.^2/2).*cos(2*pi*x); plot(x,y)
```

(説明)

- (1) は、 $-40 \leq x \leq 40$  の範囲で  $y = \sin(2\pi x / 20)$  を描く。  
(2) は、 $-5 \leq x \leq 5$  の範囲で放物線  $y = x^2$  を描く。  
(3) は、 $-5 \leq x \leq 5$  の範囲で  $y = \exp(-x^2 / 2) \cos(2\pi x)$  を描く。

`clear` は過去に使用した変数をすべてクリアし、`close all` は開かれているフィギュア・ウィンドウ (Figure Window) をすべて閉じるコマンド。`plot(x,y)` は、横軸に  $x$ 、縦軸に  $y$  をとったグラフを表示するコマンドで、図 2.1 は実習 2.1(1)の実行結果。MATLAB の `plot` 文では、縦軸の最大値が自動的に計算され、その値で規格化されたグラフがフィギュア・ウィンドウに表示される。なお、`plot` 文にはいくつかのオプションがあり、グラフのスタイルを設定することもできる (⇒第4章)。

実習 2.1 の実行では、いずれも 2 つの変数  $x$ 、 $y$  が定義された。これらの変数の属性 (プロパティ) は図 2.2 に示すワークスペース・ブラウザで確認できる。たとえば、実習 2.1 の(1)の実行後には、ワークスペース・ブラウザには、 $x$ 、 $y$  ともサイズが  $1 \times 81$ 、バイト数が 648 バイト、クラス (データの型) が `double` であることが示される。

なお、`clear` コマンドは、ワークスペースからすべての変数を消去する。

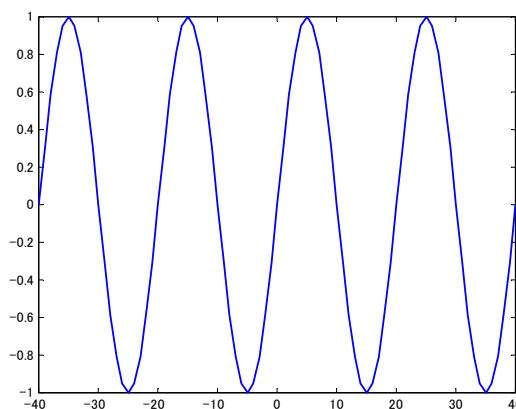


図 2.1 実習 2.1(1)の結果

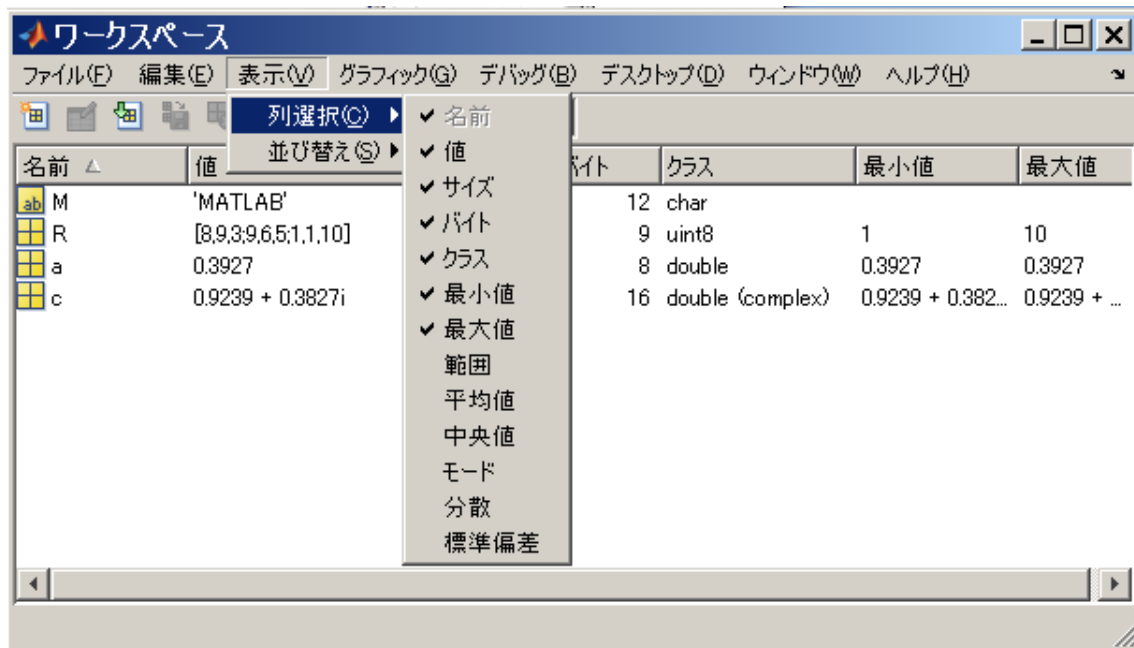


図 2.2 ワークスペース・ブラウザのダイアログボックス

ワークスペース・ブラウザには、変数名のほかに値、サイズ、バイト数、クラス、最小値、最大値などが選択的に示される。ここで、クラスとは変数の型のことで、主なクラスとしては以下のものがある。

double     .... 倍精度の数値行列（デフォルト値）  
char       .... 文字配列（各キャラクターは 16-bit）  
uint8      .... 符号なし 8-bit 整数データ  
cell       .... セル配列  
struct     .... 構造体

**実習 2.2** 以下を実行し、ワークスペース・ブラウザで R と X の属性を確認せよ。

```

>> R=rand(10);
>> X=uint8(100*R);
>> imagesc(X); colormap(bone)

```

（説明）

2 行目：uint8 は、double（倍精度）の変数を非負の 8 ビット整数に変換するコマンドである。

3 行目：imagesc 文は、2 次元配列 X を画像表示するコマンドのひとつである。この命令では、配列 X の最大値と最小値が自動計算され、データの値は [0,255] の範囲に規格化された結果がフィギュア・ウィンドウに表示される。なお、colormap(bone) はその色調を与えるコマンドである。

## 第3章 Mファイルとプログラミング

### 3.1 Mファイルとは

ここまでは、コマンドウィンドウ (Command Window) からの実行であったが、一度 MATLAB を閉じるとすべての実行内容は消失する。プログラム内容を保存するためには、Mファイルというテキストファイルにそれを記述する。このMファイルとは、次のように決められたファイルである。

- Mファイルは、プログラム内容を記述したテキストファイルである。
- ファイル名の拡張子は、「.m」と約束されている。
- 保存したファイル名 (拡張子を除いたもの) をコマンドウィンドウから入力すると、プログラムが実行される。

このように、Mファイルはプログラム内容を記述したソースファイルであるとともに、コマンドウィンドウから実行できるコマンドでもある。つまり、MATLAB では、CやFortranとは異なり、コンパイルやリンクは必要ない。したがってexeファイルも作成されない。

つまり、Mファイルの内容は一連のコマンドを決められた手順で記述したものと考えてよい。コマンドウィンドウからMファイル名をコマンドとして入力すると、その記述内容を順次実行していく。

### 3.2 Mファイルの作成・編集

Mファイルは、コマンドウィンドウでエディタを起動して作成する。新規のMファイルを作成するときには、コマンドウィンドウから

```
» edit
```

と入力すると、エディタが起動する。そこに目的のプログラムを記述し、拡張子を「.m」としたファイル名をつけて保存する。これでMファイルができる。また、既存のMファイル「filename.m」を再編集するときには

```
» edit filename
```

と入力するとMファイルが呼び出される。このとき拡張子.mは不要である。既存のMファイルを呼び出そうとしてエラーがでる場合がある。その原因のほとんどは、そのファイルがカレント・ディレクトリに存在しないことによる。この場合には、カレント・ディレクトリをそのファイルが存在するディレクトリに変更して、再実行する(⇒第0章)。

### 3.3 M ファイルの実例

実は M ファイルには 2 種類ある。スクリプト M ファイルとファンクション M ファイルである（通常、単に M ファイルと言うと両方あるいはスクリプト M ファイルを指す）。ここでは、スクリプト M ファイルについて説明する。

簡単に言うと、スクリプト M ファイルは、入出力変数を持たないコマンドの集まりからなるテキストファイルである。エディタでこの M ファイルを作成し、コマンドウィンドウからファイル名を入力すると、そこに書かれたコマンドが順次実行される。

スクリプト M ファイルの一例として簡単な 2 元連立方程式

$$\begin{aligned} 4x - y &= 4 \\ 2x + y &= 14 \end{aligned} \quad (3.1)$$

を取りあげる。この方程式は、

$$A = \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4 \\ 14 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$$

とすると、

$$A\mathbf{x} = \mathbf{b} \quad (3.2)$$

とかける。解のベクトル  $\mathbf{x}$  は、(3.2)の両辺に逆行列  $A^{-1}$  を乗じるか、あるいは左割り行って

$$\mathbf{x} = A^{-1}\mathbf{b} \quad \text{あるいは} \quad \mathbf{x} = A \setminus \mathbf{b} \quad (3.3)$$

として得られる。むろん結果は同じである。次の実習 3.1 は、この問題を解くスクリプト M ファイルである。

**実習 3.1** エディタを起動し、次の M ファイルを作成し、実行しなさい。

```
%File name: Lineq.m
%Linear equation: 4x-y=4
% 2x+y=14
A=[4 -1
 2 1];
b=[4
 14];
x=inv(A)*b
x=A\b
%End of file
```

このプログラムの実行は、コマンドウィンドウで `》lineq` と M ファイル名を入力す

る。結果は、式(3.3)で与えられる二つの解法による解が示される。

実習 3.1 からスクリプト M ファイルは以下の特徴をもっていることが分かる。

- 行番号はない。
- %記号に続く記述は、コメントであって実行されない。
- 入出力変数がない。

なお、入出力変数をもつプログラムが、ファンクション M ファイルである。しかし、これについては、ここでは記さない。

**【演習問題 3.1】** 実習 3.1 を参考にしてつぎの 3 元連立方程式を解きなさい。

$$\begin{aligned}x + y + z &= -8 \\2x + 5y - z &= 5 \\x + 2y - 5z &= 9\end{aligned}$$

### 3.4 制御文

プログラミングでは、条件文 (if 文)、繰り返し文 (for 文と while 文)、分岐文 (switch 文) の制御文がしばしば用いられる。ここでは、if 文と for 文の一例を理解する。

**実習 3.2** エディタを起動し、次の M ファイルを作成し、実行しなさい。

```
% Filename: if_for.m
clear;
R=rand(1,7);
Rmin=inf;
for k=1:length(R)
 if R(k)<=Rmin; Rmin=R(k); kmin=k; end
end
R,Rmin,kmin
%End of file
```

(説明) このプログラムは、7 個の乱数 R を発生し、その中の最小値を求めるものである。ここでは、for 文の中に if 文が使われていて、for 文の繰り返し回数は乱数の個数 (すなわち、ベクトル R の要素数) である。

if 文では表 3.1 にある論理演算記号が使用できる。

|    |       |    |       |    |       |
|----|-------|----|-------|----|-------|
| >  | より大きい | <  | より小さい | >= | 以上    |
| <= | 以下    | == | 等しい   | ~= | 等しくない |
| &  | および   |    | または   |    |       |

表 3.1 論理演算記号

## 第4章 グラフの描画

### 4.1 プロット関数 (プロット文)

MATLAB では、グラフを描く幾つものコマンドがあるが、その中で `plot` 文が最もよく利用される 2 次元グラフを描く基本的なコマンドである。また、これによって 2 次元グラフ表示機能の大略を知ることができる。まず、実行例を示そう。

下の実習 4.1 は、ガウス関数

$$y = \exp(-ax^2) \quad (4.1)$$

のパラメータ  $a$  を変えて得られる 2 通りの  $y$  を描くプログラムである。これを通して `plot` 文の基本を理解できる。

**実習 4.1** `plot` 文の基本事項を理解するために、次のプログラムの M ファイルを作成し、実行しなさい

```
% Example of plot

clear; close all
x=-2:0.1:2;
y1=exp(-x.^2);
y2=exp(-4*x.^2);

figure(1)
plot(x,y1,x,y2,'LineWidth',3)
grid on

figure(2)
hold on
plot(x,y1,'-ob','LineWidth',2)
plot(x,y2,'--sr','LineWidth',2)
grid on
%End of file
```

(説明)

このプログラムでは、最初の `clear` コマンドですべての変数を初期化し (消去し)、`close all` ですべてのフィギュア・ウィンドウを初期化する (閉じる)。次に、 $x$  座標値を定義し、2 つの  $y$  座標値  $y1$ ,  $y2$  を定義している。

`figure(1)` で Figure 1 を指定し、

```
plot(x,y1,x,y2,'LineWidth',3)
```

でそこに2つの曲線を描く． つぎに， `figure(2)` で Figure No.2 を指定し，

```
plot(x,y1,'-ob','LineWidth',2)
plot(x,y2,'--sr','LineWidth',2)
```

で別々に2つの曲線を描く． この Figure 2 の結果が図 4.1 である．

なお，このプログラム中で `grid` コマンドが用いられている．これは，次節で述べるグラフの装飾関数の一つであって，

```
grid on
```

が実行されると，座標軸の目盛に一致した格子線がグラフ中に描かれる．格子線を消すときには `grid off` を実行する．

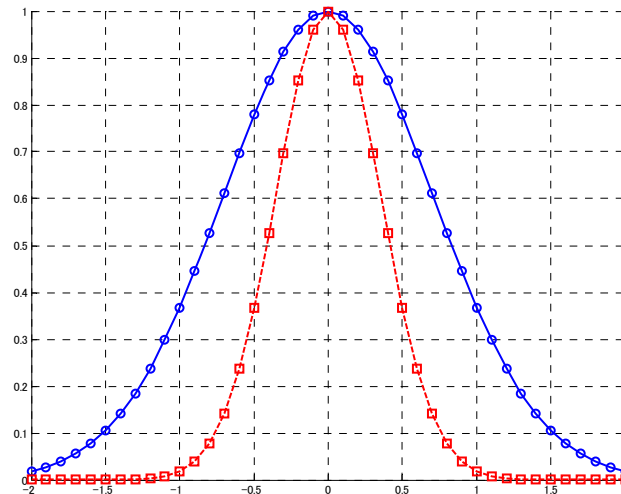


図 4.1 実習 4.1 の実行結果

さて，ここで `plot` 文の詳細を記す．表 4.1 は `plot` 関数（文）の代表的な記述をまとめたものである．

| 表 示                                            | 説 明                                                                               |
|------------------------------------------------|-----------------------------------------------------------------------------------|
| <code>plot(y)</code>                           | <code>y</code> の配列番号を横軸として <code>y</code> のグラフを描く                                 |
| <code>plot(x,y)</code>                         | <code>x</code> を横軸として <code>y</code> のグラフを描く                                      |
| <code>plot(x1,y1,x2,y2,...)</code>             | $(x_n, y_n)$ の組み合わせで複数のグラフを描く                                                     |
| <code>plot(x,y,LineSpec)</code>                | <code>LineSpec</code> で指定された線種，マーカー，色でグラフを描く（⇒表 4.2）                              |
| <code>plot(x,y,LineSpec,'LineWidth',lw)</code> | ' <code>LineWidth</code> ', <code>lw</code> によって， <code>lw</code> で指定された線幅でグラフを描く |

表 4.1 `plot` 関数の記述．`x`，`y` は実数のベクトル．

ベクトル  $y$  に関するプロット文の最も簡単な記述は

`plot(y)`

である。このとき、 $y$  が実数ベクトルならば  $y$  のインデックス（配列番号）を横軸にとってグラフを描く。

横軸と縦軸のデータが実数のベクトルの組  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$  ... で与えられるときプロット文の一般的な記述は、

|                                                                           |
|---------------------------------------------------------------------------|
| <code>plot(x1,y1,LineStyle1,x2,y2,LineStyle2,..., 'LineWidth', lw)</code> |
|---------------------------------------------------------------------------|

である。これによって、ベクトルの組  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$  ... の複数のグラフ（ライン）が一つのフィギュア・ウィンドウに描かれる。ここで、それぞれのLineStyle はラインスタイルの3成分（線種、マーカシンボル、および色）の指定子で、表 4.2 から選択して記述する。たとえば、線種を実線(-)，マーカをアスタリスク(\*)，色を赤(r)とするときには

`'-*r'`

と、シングルコートで挟んで記述する。このとき並びの順序は問われない。また、3成分の一部だけ指定しても良い。LineStyle を省略すると、デフォルトとして、複数のデータに対して青、緑、赤... の順でマーカなしの実線でグラフが描かれる。

ラインの線幅は、デフォルトでは 0.5 ポイント（1 ポイントは 1/3.2 インチ）である。plot 文の引数の最後に 'LineWidth',lw を記入すると線幅が lw に与えたポイント数に変更できる。

なお、plot 文では、座標軸に関して何も指定しないときには、各軸のデータの最小値、最大値を使って座標軸は自動設定される。もちろん axis コマンドを用いてマニュアル設定もできる。また、座標軸の線幅も希望のものに変えられる。これらについてはヘルプを参照するとよい。

| 線種 |    | マーカシンボル |        |   |       | 色 |      |
|----|----|---------|--------|---|-------|---|------|
| -  | 実線 | .       | 点      | v | 下向き矢印 | y | 黄    |
| :  | 点線 | o       | 円      | > | 右向き矢印 | m | マゼンタ |
| -. | 鎖線 | x       | x 印    | < | 左向き矢印 | c | シアン  |
| -- | 破線 | +       | プラス    | p | 五角形   | r | 赤    |
|    |    | *       | アスタリスク | h | 六角形   | g | 緑    |
|    |    | s       | 四角形    |   |       | b | 青    |
|    |    | d       | ダイヤモンド |   |       | w | 白    |
|    |    | ^       | 上向き矢印  |   |       | k | 黒    |

表 4.2 ラインスタイルを指定する 3 成分

上述したように、plot 文 1 行で一つのフィギュア・ウィンドウに複数のラインを

描くことができる。これとは別に、複数のラインは、`plot` 文を繰り返し記述して描くこともできる（実習 4.1 の前半）。このときには、`hold` コマンドを用いて `plot` 文とともに

`hold on`

を記述する。これによって、先に描かれたラインが保持され、引き続く `plot` 文のラインが追加される（実習 4.1 の後半）。

## 4.2 グラフの装飾と subplot 文

描いたグラフにタイトルや説明を与えるコマンドとして表 4.3 に挙げる装飾関数がある。この中でグリッドライン `grid` を除く装飾関数は、所定の文字列をグラフの中に書き込むコマンドである。

| 装飾関数                |        |                     |               |
|---------------------|--------|---------------------|---------------|
| <code>title</code>  | タイトル   | <code>text</code>   | 文字列表示         |
| <code>xlabel</code> | x 軸ラベル | <code>gtext</code>  | マウス指定による文字列表示 |
| <code>ylabel</code> | y 軸ラベル | <code>legend</code> | 凡例            |
| <code>zlabel</code> | z 軸ラベル | <code>grid</code>   | グリッドライン       |

表 4.3 グラフの装飾関数

次の実習 4.2 は、2つの正弦波信号

$$s_1 = \sin(2\pi f_1 t) \quad \text{と} \quad s_2 = \sin(2\pi f_2 t) \quad (4.2)$$

のビート信号

$$s = s_1 + s_2 \quad (4.3)$$

を計算し、グラフに描くプログラムである。これを通して、グラフの装飾関数を確認できる。

**実習 4.2** 次のプログラムの M ファイルを作成し、装飾関数と `subplot` 文を確認せよ。

```
% File name: Beat.m
```

```
clear; close all
```

```
T=250;
```

```
% Signal length
```

```
N=4000;
```

```
% Number of sampling
```

```
t=(1:N)/T;
```

```
f1=10;
```

```
f2=f1+1;
```

```
s1=sin(2*pi*f1*t);
```

```
s2=sin(2*pi*f2*t);
```

```
s=s1+s2; % Amplitude
```

```
figure(1)
```

```

subplot(3,1,1); plot(t,s1,'LineWidth',2); axis([0 , 2.5,-2, 2])
title('信号 1')
subplot(3,1,2); plot(t,s2,'LineWidth',2); axis([0, 2.5,-2 , 2])
title('信号 2')
subplot(3,1,3); plot(t,s,'LineWidth',2); axis([0 , 2.5,-2, 2])
title('ビット信号')
xlabel('周波数 [Hz]')
%End of file

```

なお、このプログラム中では、`subplot` 文が使われている。これは、ひとつのフィギュア・ウィンドウに複数のグラフを描くときに用いられる。たとえば、`subplot(3,1,2)` はフィギュア・ウィンドウを 3 行 1 列の描画窓に分割し、その 2 番目の窓にグラフを描くコマンドである。

また、`axis([xmin, xmax, ymin, ymax])` のコマンドは、x 軸と y 軸の座標値の最小値と最大値を指定するコマンドである。したがって、`axis([0, 2.5,-2, 2])` は x 軸座標が 0 ~ 2.5、y 座標が -2 ~ 2 の範囲でグラフが描かれる。

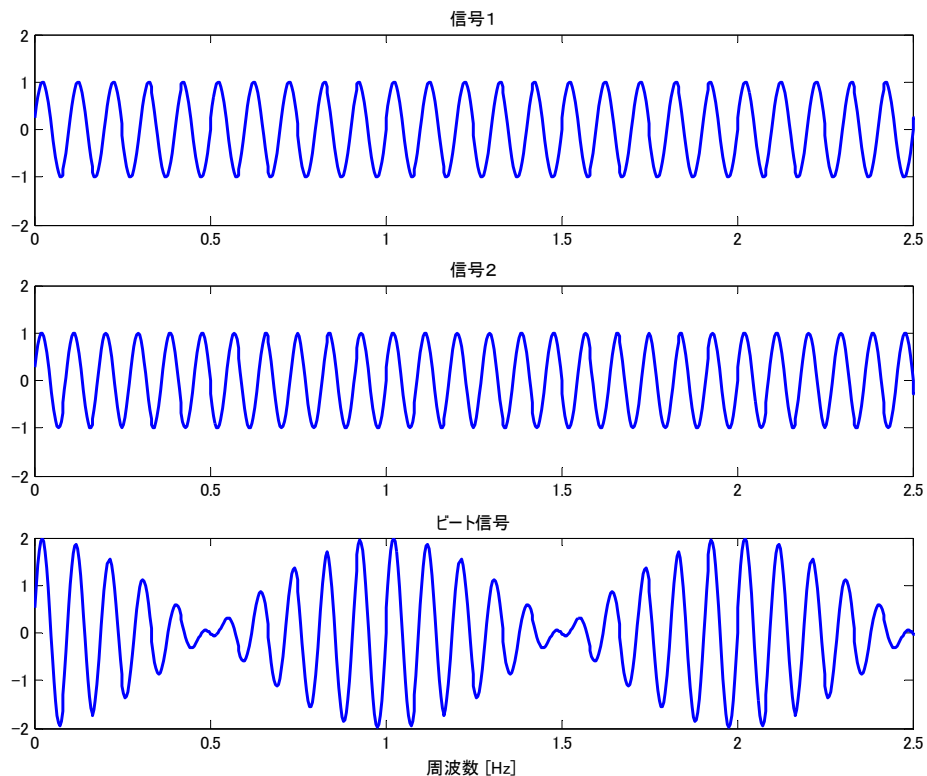


図 4.2 実習 4.2 の結果

## 第 5 章 Control System Toolbox

Control System Toolbox は制御系の解析と設計への応用を目標に作成された M ファイルパッケージであり、制御理論で使う種々の関数を MATLAB プロンプトから呼ぶだけで使用できる。使用可能な関数とその説明は

helpdesk→Control System Toolbox→Control System Toolbox 関数リファレンスをたどることで調べられる。また **help 関数名** で詳しい使用方法が表示される。

### 5.1 伝達関数の表現

制御工学 I で学ぶ伝達関数法では制御システムを伝達関数で表現して、それに対し各種の操作を施す。伝達関数の詳しい説明は授業でおこなうが、ここではその MATLAB による表現を理解する。

伝達関数は tf 関数で表現する。一般に伝達関数は  $s$  (複素数) を変数とする複素多項式の比として表される関数 (複素有理関数と呼ぶ) となる。この比の分子 (numerator) が降べきの順に書かれた多項式

$$b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0$$

であるとき、この多項式は係数  $b_m b_{m-1} \cdots b_1 b_0$  を用いて

$$\text{num}=[b_m b_{m-1} \cdots b_1 b_0]$$

と定義される。このとき係数がなければ 0 を入れる。もしそれを入れなければ次数が下がった表現とされてしまう。同様に、分母 (denominator) が

$$a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0$$

であるときは、この多項式は係数  $a_n a_{n-1} \cdots a_1 a_0$  を用いて

$$\text{den}=[a_n a_{n-1} \cdots a_1 a_0]$$

と定義する。そして、sys と名付ける伝達関数は、上の num と den を用いて

$$\text{sys}=\text{tf}(\text{num},\text{den})$$

と書いて定義される。なお、直接

$$\text{sys}=\text{tf}([b_m b_{m-1} \cdots b_1 b_0],[a_n a_{n-1} \cdots a_1 a_0])$$

と書いてもよい。

例：伝達関数  $G(s) = \frac{s+15}{s^3+2s+10}$  を Control System Toolbox で表す。

```
>>num=[1 15], den=[1 0 2 10],sys=tf(num,den)
```

あるいは

```
>>sys=tf([1 15],[1 0 2 10])
```

$s^2$  の係数はないので, 0 を入れていることに注意. 定義された伝達関数は sys という名前でこの後で自由に使用できる. 詳しくは tf のヘルプを参照のこと.

### 実習 5.1

次の伝達関数を MATLAB 表現し, それぞれ sys1,sys2 の名前をつけなさい.

$$G_1(s) = \frac{1}{2s^4 + s^3 + 3s^2 + 5s + 9}, \quad G_2(s) = \frac{10}{s(s+1)(s+5)}$$

$G_2(s)$  の分母は多項式に展開して表現せよ.

## 5.2 ボード線図を描く関数 bode の使用

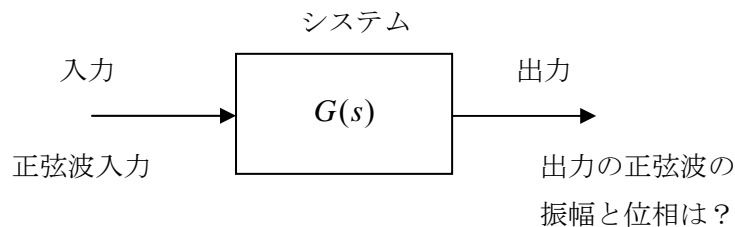


図 5.1 周波数応答

図 5.1 に示すように正弦波入力線形システムに加わると, 時間がたつて定常状態になったときの出力は正弦波となり, その振幅と位相が加わる周波数により変化する. 入力周波数を変えたときの応答正弦波の振幅と位相を表示する方法の一つがボード線図である. システムのボード線図による周波数応答の表示には関数 `bode` を使い,

```
>>bode(sys)
```

と入力する. この結果は別の Figure ウィンドウに表示される. 表示された結果はウィンドウ内の機能を使用して表現を変えることができる. 5.1 節の例に挙げた伝達関数

$$G(s) = \frac{s+15}{s^3 + 2s + 10}$$

の周波数応答のボード線図は, sys を定義して上のコマンドを実行すると図 5.2 がえられる.

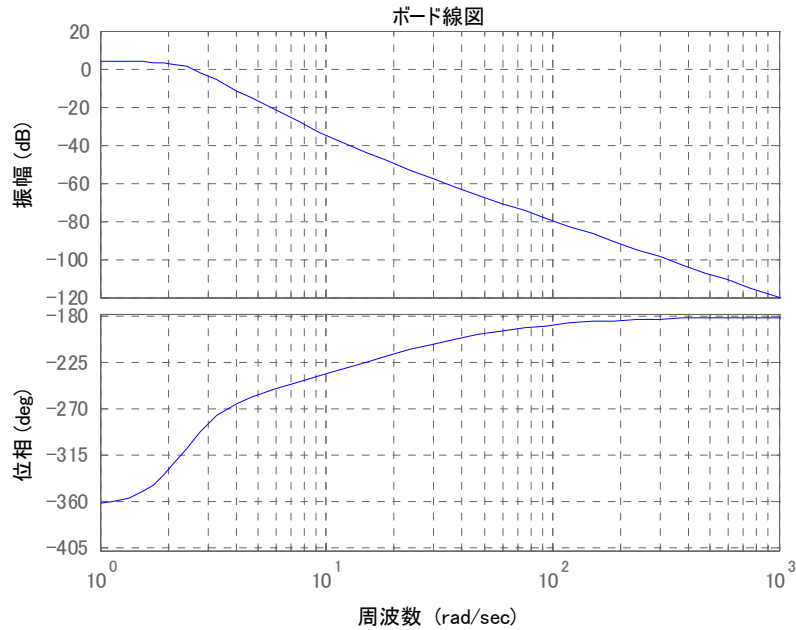


図 5.2 周波数応答のボード線図表現の一例

ここまで、MATLAB のコマンドラインからシステムの記述、ボード線図描画を逐次実行してきたが、次のような M ファイルを作成して実行してもよい。

```
%実習 5 の例
% Filename: Z5example
% G(s)=(s+15) / (s^3+2s+10)


clear; close all
%sys
num=[1]; % 伝達関数 G(s)の分子
den=[1,0,2,10]; % 伝達関数 G(s)の分母
sys=tf(num,den); % 伝達関数 G(s) (tf で伝達関数を定義)
bode(sys); % ボード線図の計算 → 描画
grid on;
%End of file
```

このように制御工学で計算に必要な大部分の関数は Control System Toolbox のパッケージとして予め M ファイルが用意されているので、その使い方をヘルプで調べることで容易に利用できる。

## 第6章 Simulink

Simulink は MATLAB に含まれるシミュレーションソフトであり、あらかじめ用意されたブロックを結線するだけで簡単にシミュレーション結果を見ることが出来る。

### 6.1 Simulink の立ち上げ

MATLAB コマンドウィンドウから `simulink` を入力する、あるいは上部の Simulink アイコン  をクリックすると、Simulink ライブラリブラウザが表示される。ブラウザには Common Used Blocks, Continuous など 16 個のグループがあり、これをクリックすると各グループのライブラリブロックが表示される。このウィンドウを常に表示させるには表示メニュー中の 「常に前面に表示」 を選択する。

### 6.2 シミュレーションモデルの作成とシミュレーション

- ①モデルウィンドウのオープン：ウィンドウメニューから新規作成のアイコンをクリックするか、ファイル／新規作成／モデルを選ぶとモデルウィンドウが現れる。このウィンドウを常に表示させるには「最大化」をクリックする。
- ②ライブラリブロックのコピー：モデルウィンドウにライブラリウィンドウから必要なライブラリブロックをドラッグする。あるいはライブラリを右クリックして「新規モデルウィンドウ(untitled)に追加」を選ぶ。
- ③ライブラリブロックの設定：各ライブラリブロックを右クリックしてブロックパラメータを選び、パラメータを設定する。
- ④ライブラリブロックの結合：各ブロックには入力と出力の端子記号<, >がついているので、どちらかの端子にカーソルをあて、マウスの左ボタンをおしたまま結合するブロックの端子までマウスを移動し左ボタンを離す。
- ⑤引き出し線：線の上でマウスの右ボタンをクリックすると線を引き出せる。
- ⑥Scope ブロックを右クリックしてブロックを開き、見やすい適当な位置に移動する。
- ⑦シミュレーションの設定と開始：ウィンドウメニューからシミュレーションをクリックして、パラメータを設定できる。メニューウィンドウのシミュレーション／開始をクリックするか、シミュレーションの開始アイコンをクリックするとシミュレーションを開始する。

### 6.3 実習 6.1 : RC 直列回路の正弦波応答シミュレーション

2年生で履修の電子情報工学実験Ⅰ第4回目（オシロスコープの操作法Ⅱ）でおこなった図 6.1 のコンデンサと抵抗で構成する RC 直列回路を対象に，入力信号として正弦波を加えたときの出力信号の応答を求める問題を Simulink を用いてシミュレーションする。

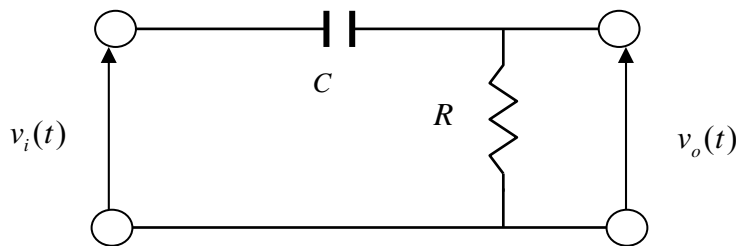


図 6.1 RC 直列回路

回路の伝達関数は  $G(s) = \frac{1}{RCs+1} = \frac{1}{Ts+1}$  で表現されるが，ここでは  $T = RC = 0.1$  とする。

#### 《正弦波応答シミュレーションの手続き》

##### 1. 起動：

MATLAB からピンクの入ったアイコンをクリック

- Simulink ライブラリブラウザが開く．表示を「常に前面に表示する」にする．
- その新規作成アイコンをクリックするとモデルウィンドウが開く．最大化する．

##### 2. ライブラリブロック（機器や回路などの部品）の準備：

- ① ライブラリブラウザにある Sources をクリック
  - 一覧から Sine Wave を選択し，モデルウィンドウ内に drag and drop.
- ② ライブラリブラウザにある Continuous をクリック
  - 一覧から Transfer Fcn を選択し，モデルウィンドウ内に drag and drop.
- ③ ライブラリブラウザにある Sinks をクリック
  - 一覧から Scope を選択し，モデルウィンドウ内に drag and drop.
- ④ 図 6.2 に示すように①～③の三つのライブラリをドラッグして並べる．

##### 3. 2チャンネルスコープの準備：

- モデルウィンドウの Scope をクリック → Scope ウィンドウが開く
- その左から 2 番目のアイコン《パラメータ》をクリック
- 開いたダイアログで座標軸数を 2 に設定して閉じる（OK）．

##### 4. 結線：

- ① Sine Wave をクリックし，次に Ctrl キーを押しながら Transfer Fcn をクリック．
- ② 次に，Ctrl キーを押したまま Scope をクリック．
- ③ Sine Wave と Transfer Fcn の間の線上で，マウスを右クリックし下方に線を出

す。

- ④ ③を繰り返して，**Scope** につなぐ．伝達関数ブロックをダブルクリックして開

き係数を  $G(s)=\frac{1}{0.1s+1}$  に設定する．つぎに正弦波ブロックを開き周波数を

10rad/s とする．図 6.2 のモデルウィンドウに示す Simulink モデルができる．

5. 実行：

メニューバーのシミュレーションから開始する．折れ点周波数での応答なので図 6.2

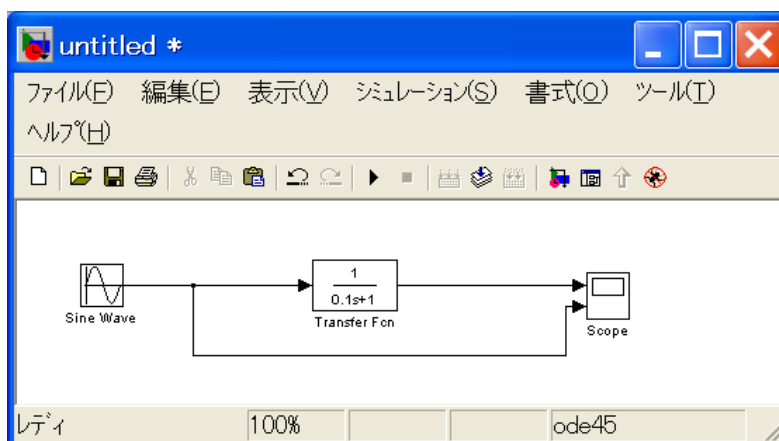
の Scope ウィンドウの出力波形は入力波形の  $\frac{1}{\sqrt{2}} \cong 0.707$  倍になっている．

6. 回路系のパラメータの設定・変更：

- ① モデルウィンドウの **Sine Wave** をダブルクリックし，周波数を変更できる．
- ② 同様に，**Transfer Fcn** をダブルクリックし，伝達関数のパラメータを変更できる．

7. 周波数を 1,5,20,50rad/s に変えてシミュレーションを実施してみよ．

モデルウィンドウ



Scope ウィンドウ

出力→  
入力  $\sin 10t$  →

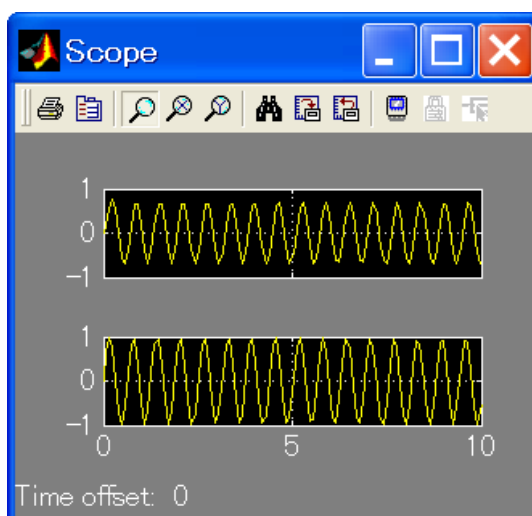
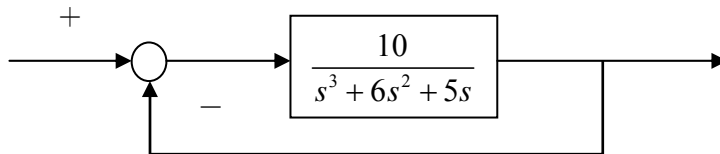


図 6.2 RC 直列回路シミュレーション

## 6.4 実習 6.2：フィードバック制御系のステップ応答

次の制御系を Simulink で表現し，単位ステップ入力に対する応答波形を求めよ．



### 《ステップ応答の手続き》

#### 1. 新規のモデルウィンドウ：

モデルウィンドウで新規作成のアイコンをクリック

→ 新規作成のモデルウィンドウが開く

#### 2. ライブラリーブロック（機器や回路などの部品）の準備：

① ライブラリブラウザにある Sources をクリック

→ 一覧から Step を選択し，モデルウィンドウ内に drag and drop.

② ライブラリブラウザにある Continuous をクリック

→ 一覧から Transfer Fcn を選択し，モデルウィンドウ内に drag and drop.

③ Transfer Fcn をダブルクリックして，次の伝達関数のパラメータを設定する．

$$G(s) = \frac{10}{s^3 + 6s^2 + 5s}$$

④ ライブラリブラウザにある Sinks をクリック

→ 一覧から Scope を選択し，モデルウィンドウ内に drag and drop.

⑤ ライブラリブラウザにある Math Operations をクリック

→ 一覧から Sum を選択し，モデルウィンドウ内の Step Transfer Fcn の間に drag and drop.

⑥ Sum をダブルクリックして，++を+-に変更する．

#### 3. 2チャンネルスコープの準備：

モデルウィンドウの Scope をクリック → Scope ウィンドウが開く

→ その左から2番目のアイコン《パラメータ》をクリック

→ 開いたダイアログで座標軸数を2に設定して閉じる（OK）．

#### 4. 結線：

① Step をクリックし，次に Ctrl キーを押しながら Sum をクリック．

② 次に，Ctrl キーを押したまま Transfer Fcn，そして Scope をクリック．

③ Transfer Fcn と Scope の間の線上で，マウスを右クリックし下方に線を出す．

④ ③を繰り返して，Sum の一端に繋ぐ．

⑤ 前と同様に，Step と Sum の間から右クリックで線を出し，Scope の2チャンネル

ル端子に繋ぐ.

#### 5. 実行 :

メニューバーのシミュレーションから開始する.

#### 参考書籍

・高井信勝：「信号処理」「画像処理」のための MATLAB 入門[増補版]，工学社，2000年

MATLAB そのものの使用方法がていねいに段階を追って説明されておりわかりやすい。付録と索引が充実しており，これを使って関数とコマンドを容易に探せる。参考書として大いにお奨めする。

・高谷邦夫：Matlab の総合応用（例題による解説），森北出版，2002年

MATLAB の基本から数値解析，確率統計，自動制御，シミュレーション，信号処理，画像処理，シンボリックな数学の応用に至る広い分野への適用を解説している。

・川田，西岡：MATLAB/Simulink によるわかりやすい制御工学，森北出版，2001年

伝達関数法を中心に MATLAB/Simulink の自動制御での使用方法を詳しく解説している。