第2章　MATLABの基本フレームワーク

2.1　はじめに

本章では，組み込みプログラムの開発において，中心的な役割を果たすMATLABの基本フレームワークについて解説します．

このMATLABの基本フレームワークはSimulinkに受け継がれ，そして，組み込みプログラム自身に継承されます．要するに，ルーツから勉強を始めるということです．

これまでにMATLABを使った経験があり，MATLABの基本フレームワークは十分に身についているという方は，本章をスキップして第3章へ進んでください．

2.2　MATLABのスタート

それでは，MATLABを操作するところから始めます．

ここでは，MATLAB 7.0.1（R 14）SP1，Windows版を使います．

本書において取り扱う機能は，MATLABの基本的な機能に限るので，無理にバージョンを合わせる必要はありません．皆さんの手元に異なるバージョンのMATLABがある場合は，それを使ってください．

ただし，バージョンが異なると，ユーザ・インターフェース（ダイアログなど）の画面が異なる場合があるので，その点は注意してください．

もし，皆さんの手元にMATLABがなければ，30日間有効の試用版を使ってみることができます．入手方法は，サイバネットシステムのWebサイト（http://www.cybernet.co.jp/matlab/product/beta.shtml）を参照してください．

私が調べた範囲内では，MATLABの日本語化作業は現在進行形の状態です．ユーザ・インターフェースの日本語化とドキュメントの日本語化が並行して進められています．

しかし，それら作業の足並みは，必ずしもそろっているとは言えません．そのために，ダイアログなどのユーザ・インターフェースは英語表示なのに，それに対応するドキュメントの説明は日本語，あるいは逆に，ユーザ・インターフェースは日本語表示で，それに対応するドキュメントの説明は英語とい

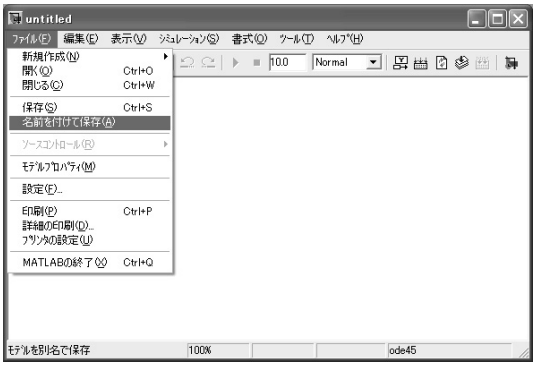

図2-1　Simulinkのモデル・ウィンドウ

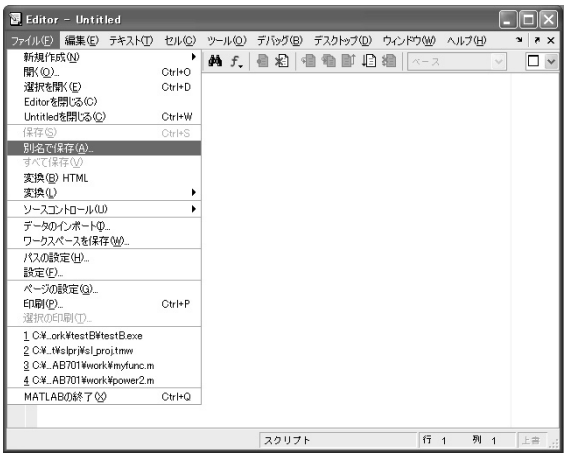

図2-2　MATLABエディタの画面

うような，ちぐはぐな状況にぶつかることがあります．こういう場合には，内容をよく理解して，英語と日本語の対応を考える必要があります．

しかし逆に言えば，この状況は，MATLABに関してかなり多くの量のドキュメントが存在するということの証明にもなります．

メーカは懸命に日本語化作業を進めているのですが，それを超える量のドキュメントが生産されているために，必要な日本語化作業が追いつかないという状況になっているのです．

私がMATLABを使っていて，ちょっと困ったなと思うことがありました．例えば，同じ英文に対する日本語訳が場所によって異なることがあります．

細かい違いですが，例えば，Simulinkのモデル・ウィンドウの画面でメニューのファイルをクリックすると，**図2-1**に示すように，英語の［Save As］は，［名前をつけて保存］という日本語に翻訳されています．

それでは，MATLABのエディタを開きます．**図2-2**に示すように，同じ［Save As］が，ここでは［別名で保存］という日本語に翻訳されています．

このようなことがあるので，内容を十分に理解した上で，慎重に操作を進める必要があります．

本書において，画面に関する用語は，私が現在使っているMATLABの画面に表示されたとおりの文字を使用します．MATLABの画面に英語が書かれていれば英語を使い，画面に日本語が書かれていれば日本語を使います．例えば，ダイアログのボタンにビルドと書いてあれば，［ビルド］と記します．Buildと書いてあれば，［Build］と記します．

MATLABの一つの英単語に対して，場所によって異なる日本語が対応する場合は，本書においては，画面に応じて異なる日本語表記を使います．

本書の表記が混乱しているのではなくて，MATLABの日本語訳が異なっていることを，あらかじめ理解しておいてください．

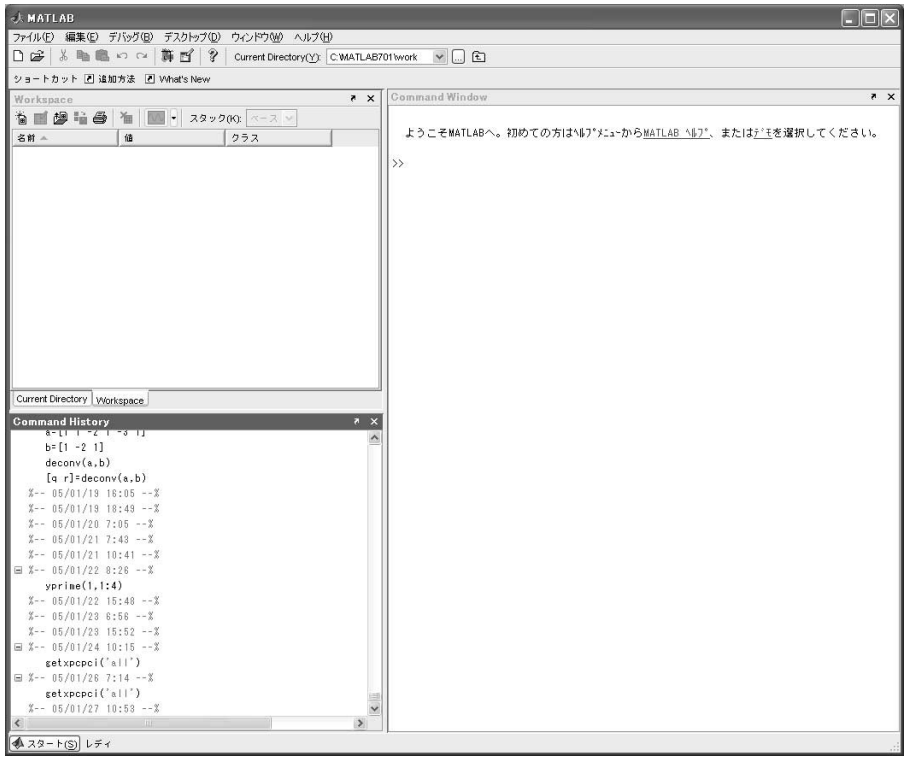

図2-3　MATLABの起動画面

それでは，PCの前に座ってMATLABを起動します．すると，**図2-3**のような画面が表示されます．

図2-3は，MATLABを標準インストールして，最初に立ち上げたときの画面です(以下，この状態をデフォルトの画面と呼ぶ)．画面内のウィンドウは，自由に開閉や移動ができるので，皆さんが見る画面が**図2-3**とまったく同じでなくても心配することはありません．

MATLABの画面は，デフォルトの状態において，三つのウィンドウから構成されます．それらは，

［Command Window］……コマンド・ウィンドウ

［Command History］ ……コマンド履歴

［Current Directory］/［Workspace］…カレント・ディレクトリ/ワーク・スペース

です．

［Current Directory］と［Workspace］は2重化されているので，タブによりどちらかのウィンドウを選択します．

MATLABを使う場合は，キーボードからコマンドを入力します．コマンドはMATLABに対する命令です．UNIXやLINUXのシェル，Windowsのコマンド・プロンプトの操作に似ています．

コマンドの入力を行う場所は，［Command Window］です．

［Command Window］は，デフォルトの状態でMATLABのペインの右側に置かれます．

［Command Window］の上部左端に，

```
>>
```

という記号があります．これはプロンプト（入力を催促する記号）です．

それでは，コマンドを入力して，処理を実行してみます．例えば，キーボードから，

```
>> 3+5
```

と入力します．入力の最後に，Enterキーを押します．

すると，**図2-4**に示すように，処理結果が画面上に表示されます．

そこで，［Command History］（デフォルトの状態では，画面左側下部）のウィンドウを見てください．使用したコマンド（この場合3+5）が記録されています（**図2-5**）．このウィンドウを見ることによって，過去にどのようなコマンドを使用したか，その履歴を調べることができます．

それでは，**図2-6**に示した［Workspace］のウィンドウを見ます．

図2-3の左側上部のウィンドウです．ここにansという変数が記録されています．

この変数が見えないときは，ウィンドウは［Current Directory］になっています．ウィンドウ下部のタブをクリックして，ウィンドウを［Workspace］に変更してください．

コマンドラインにおいて，とくにansという文字列を入力したわけではないのですが，ユーザが変数を指定しなかった場合，MATLABはansという一時的な変数を生成して，これに結果を入れ，［Workspace］に格納します．

［Workspace］内のデータを見ると，変数ansはdouble型の配列で，8バイト（64ビット）のメモリを使

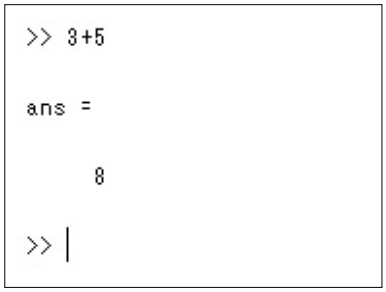

図2-4　3＋5の処理結果

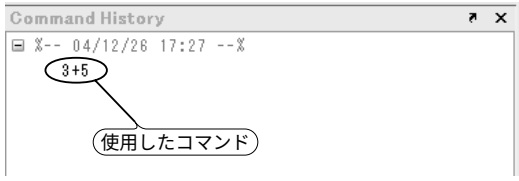

図2-5　コマンドの記録

図2-6　ワークスペース

用していることがわかります．

ansには8という数値を入れたので，変数の型からいうとスカラですが，MATLABはスカラを1行1列のマトリックスとして登録します．MATLABにおいて，すべての変数はdouble型のマトリックスとして扱われます．これが第1のポイントです．

double型は64ビットの浮動小数点数であり，指数部16ビット，数値部は48ビットです．48ビットは，およそ，$2^{48} \approx 1.5\times10^{14}$となるので，有効桁数は14～15桁程度です．

組み込み環境において，例えば，A-D変換器の入力データの型はint，スイッチのON/OFF状態を表す型はboolになります．このような場合は，データの型をdoubleから，明示的に変更する必要があります．

2.3 MATLABの関数

ピタゴラスの定理を使って直角三角形の斜辺の長さを計算します．**図2-7**に示すように，直角三角形の底辺の長さをa，高さをbとします．

直角三角形の斜辺cは，ピタゴラスの定理によって，

$$c=\sqrt{a^2+b^2}$$

となります．

［Command Window］から，まず，データを，

```
>> a=3;b=4;
```

と入力し，続いて，

```
>> c=sqrt(a^2+b^2)
```

と入力します．答えは，当然，

```
c=5
```

です．

sqrt( )は平方根を求めるためのMATLAB関数（MATLAB function）です．

例えば，コマンドとして，

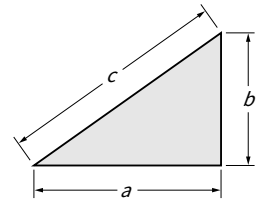

図2-7 ピタゴラスの定理

```
>> sqrt(2)
```

と入力すると，答えは，

```
ans=1.4142
```

となります．

MATLABの構成要素は関数です．必要とする関数がMATLAB内に用意されていれば，その関数の名前と引き数をコマンドラインに書き込むことによって，処理が行われ，計算結果が出力されます．逆の言い方をすれば，MATLABに用意されていない関数は計算できません．

MATLABの基本は関数です．MATLABは関数の集合です．ここが第2のポイントです．

いま，仮に観測データに対して，ウェーブレット(wavelet)の計算をしたいとします．ウェーブレットを計算する関数は，MATLABに標準では用意されていません．自分でプログラムを作ってMATLAB上で実行するか，あるいはMATLABプロダクト・ファミリに用意されているWavelet Toolboxを購入するか，どちらかを選択します．

MATLABに対して，Wavelet Toolboxを購入して追加インストールすると，ウェーブレットの計算に必要な関数がMATLABコマンドライン上において使用可能になります．

MATLABのツール・ボックスは，MATLABに対して，特定の分野で使用する関数を提供し，MATLABの適用範囲を拡張します．

物理学の世界において，物体の状態を表すために，順序付けられた数値の組(tuple)を使います．こ

■ 2.4 ベクトルとマトリックス

れをベクトル(vector)といいます．例えば，(1, 2, 3)は一つのベクトルです．このベクトルは**図2-8**に示すように，空間内の一つの点の位置を示します．

要素が3のベクトルを3次元ベクトル(three dimensional vector)といいます．

物理学における力，トルク，回転角速度などは，大きさと方向をもつ量なので，これも3次元ベクト

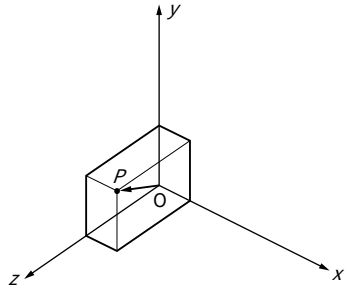

図2-8　3次元空間内のベクトル

ルによって表現します.

組み込み系は現実の物理世界を相手にするので，組み込み系のプログラムを作る際に，3次元ベクトルの数学を十分に理解する必要があります.

MATLABにおいてベクトルを作ります.

コマンドラインにおいて,

```
>> v=[1 2 3]
```

と入力します.

［Workspace］にベクトルvが登録されます．厳密に言うと，vは1行3列のマトリックスです．スカラが1行1列のマトリックスになったのと同じ論理です.

$$v=(1,2,3)$$

ベクトルはカギ・カッコ［ ］によって囲み，各要素はスペースで区切ります.

```
>> v=[1,2,3]
```

のように各要素をコンマで区切っても同じ結果が得られます.

今度は，コマンドラインにおいて,

```
>> x=[1; 2; 3]
```

と入力します.

［Workspace］にベクトルxが登録されました．ベクトルの各要素をセミコロン；で区切ると縦ベクトルができます．これは3行1列のベクトルです．vとxは異なるベクトルです.

$$x=\begin{vmatrix}1\\2\\3\end{vmatrix}$$

連続した数値のベクトルを作る際には，コマンドラインにおいて,

```
>> w=[1:10]
```

と入力すると，$w=(1,2,3,4,5,6,7,8,9,10)$というベクトルができます.

コロン：を使うと連続した値のベクトルになります．この場合，コマンドラインにおいて，［ ］なしで,

```
>> w=1:10
```

と入力しても，同じベクトルが得られます.

これは，コロン：の演算子が，［ ］と同じ機能を持っているからです．どちらの記法でも同じ結果が得られるので，好きなほうを使ってください.

コマンドラインにおいて，

```
>>v=[1:2:10]
```

と入力すると，$v=(1,3,5,7,9)$ というベクトルができます．1を出発点として，2刻みで，10あるいはそれ以下の数(この場合9)までの数値が設定されます．

C言語などのコンピュータ言語を使ってプログラムを作る際には，あらかじめ変数を宣言する必要がありました．例えば，

```
int a,b;
float f;
```

などです．

MATLABでは，変数の宣言は不要です．ベクトルは，コンピュータ言語における配列に該当します．また，コンピュータ言語において配列を使う際には，あらかじめ配列の大きさを宣言する必要がありました．例えば，

```
float a[100];
```

などです．このプログラムにおいて，配列aに，200個のデータを格納することはできません．プログラムを書き直して，配列の次元を修正したうえで，再コンパイルする必要があります．

MATLABでは，そのような配列の大きさを宣言する必要はありません．例えば，

```
v=[1 2 3]
```

とすれば，要素数が3の配列vが作られ，さらに，

```
v=[3 4 5 6 7 8 9 10 11 12]
```

とすれば，配列vは要素数10の配列に変わります．

配列はシステムが管理するので，ユーザは配列の大きさを考える必要ありません．

これは，とても便利な機能ですが，大きなプログラムを作る際には，予期しないエラーを呼び込む原因になることがあるので，注意が必要です．これが第3のポイントです．

二つのベクトル$a=(a_1,a_2,a_3)$，$b=(b_1,b_2,b_3)$ のスカラ積は，公式$a\cdot b=a_1b_1+a_2b_2+a_3b_3$を用いて計算します．

コマンドラインにおいて，

```
>> a=[1 2 3];b= [4 5 6] ;
>> c=dot(a,b)
```

と入力すると，c=32となり，二つのベクトルのスカラ積が計算できます．dot(a,b)は，スカラ積を

計算するMATLAB関数です．ここで，コマンドの最後にセミコロン；を付けると，画面へのプリントは省略されます．画面へのプリントが不要のとき(例えば，大量のデータをMATLABに読み込むときなど)には，コマンドの最後に；を付けます．

ピタゴラスの定理を使うと，3次元空間におけるベクトルの長さは，

$\|a\| = \sqrt{a_1^2 + a_2^2 + a_3^2}$ ，$\|b\| = \sqrt{b_1^2 + b_2^2 + b_3^2}$

となります．これを，とくに，ユークリッドのノルムと言います．

MATLABにおいて，ベクトル$a = (1,2,3)$ のノルムを計算する場合，コマンドラインにおいて，

```
>> a=[1 2 3];
>> norm(a)
```

と入力します．

答えは，$\sqrt{14}$ なので，

$ans = 3.7417$

となります．

関数norm( )の引き数は，ベクトルです．

二つのベクトル$a = (a_1, a_2, a_3)$，$b = (b_1, b_2, b_3)$ のベクトル積は，次の公式，

$a \times b = (a_2 b_3 - a_3 b_2, a_3 b_1 - a_1 b_3, a_1 b_2 - a_2 b_1)$

によって計算します．

ベクトル積は，一つのベクトルを与えます．

このベクトルは，二つのベクトルに直交し，その長さは二つのベクトルが作る平行四辺形の面積に等しくなります(**図2-9**)．

二つのベクトル$a = (1,2,3)$，$b = (-1,1,-2)$ のベクトル積を計算します．コマンドラインにおいて，

```
>> a=[1 2 3];b=[-1 1 -2];
>> c=cross(a,b)
```

と入力します．結果は，

$c = (-7,-1,3)$

となります．

ベクトルの順序を逆にして，

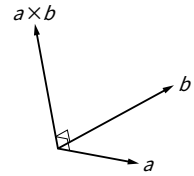

図2-9　二つのベクトルのベクトル積

```
>> d=cross(b,a)
```

と入力すると，

$d=(7,1,-3)$

となります．符号が反転し，逆向きのベクトルになりました．

ベクトル積において，順序を変えると，

$a\times b\neq b\times a$

となります．

二つのベクトル$a=(a_1,a_2,a_3)$，$b=(b_1,b_2,b_3)$が作る平面の方程式を導きます．

平面の方程式は，

$Ax+By+Cz+D=0$

です．ここで，A，B，Cは平面の法線ベクトルの成分，Dは平面から原点までの距離なので，ベクトル積の公式を利用して，

$A=a_2b_3-a_3b_2$

$B=a_3b_1-a_1b_3$

$C=a_1b_2-a_2b_1$

と置くと，

$D=-(A,B,C)\cdot(a_1,a_2,a_3)$

となります．$D=-(A,B,C)\cdot(b_1,b_2,b_3)$としても同じ結果が得られます．

二つのベクトル$a=(1,2,3)$，$b=(-1,1,-1)$が作る平面の方程式を計算します．

$n=(A,B,C)$と置くと，

$n=a\times b$

$D=-n\cdot a$

を計算します．

コマンドラインにおいて，

```
>> a=[1 2 3];b=[-1 1 -1];
>> n=cross(a,b)
```

と入力すると，

```
n=(-5,-2,3)
```

となります．dを計算するために，

```
>> d=-dot(n,a)
```

と入力すると，

```
d=0
```

となります．平面の方程式は，

$$-5x-2y+3z=0$$

となります．二つのベクトルが原点を通るので定数項Dはゼロです．

ベクトルを拡張してマトリックスを作ります．コマンドラインにおいて，

```
>> a=[4 1 3;7 9 2;8 6 5]
```

と入力すると，［Workspace］に，

$$a=\begin{vmatrix}4 & 1 & 3\\ 7 & 9 & 2\\ 8 & 6 & 5\end{vmatrix}$$

というマトリックスができます．

マトリックスを入力する場合，行の区切り記号としてセミコロン；を使用します．ベクトルの場合と同じです．

マトリックスとベクトルの積を求めます．コマンドラインにおいて，

```
>> b=[1;-1;2];
```

と入力して，［Workspace］内にベクトル$\boldsymbol{b}$を作り，次に，このベクトルに対してマトリックスをかけ，

```
>> c=a*b
```

と入力すると，

$$c=\begin{vmatrix}9\\ 2\\ 12\end{vmatrix}$$

というベクトル$\boldsymbol{c}$が得られます．

演算子*は，この場合，ベクトルとマトリックスの乗算を行います．

ベクトル$\boldsymbol{b}$をマトリックスに書き換えます．コマンドラインにおいて，

```
>> b=[-1 1 9;3 -2 -2;-4 5 7]
```

と入力します．すると，［Workspace］に，マトリックス，

$$b=\begin{vmatrix}-1 & 1 & 9\\ 3 & -2 & -2\\ -4 & 5 & 7\end{vmatrix}$$

ができます．そこで，コマンドラインにおいて，

```
>> a*b
```

と入力すると，結果は，

$$ans = \begin{vmatrix} -13 & 17 & 55 \\ 12 & -15 & 9 \\ -10 & 21 & 95 \end{vmatrix}$$

となります．

演算子*は，この場合，マトリックスとマトリックスの乗算を行います．

MATLABは，すべての変数をマトリックスとして扱うので，ベクトルとベクトル，ベクトルとマトリックス，マトリックスとマトリックスの演算を区別する必要はありません．同じ演算子*を用います．

数学において，マトリックスの転置行列(transposed matrix)は，通常，

$$a^T$$

と書きます．

転置行列が元の行列に一致する場合，その行列を対称行列(symmetric matrix)といいます．例えば，

$$s = \begin{vmatrix} 1 & 2 & 3 \\ 2 & 1 & 7 \\ 3 & 7 & 9 \end{vmatrix}$$

は対称行列です．

数学的に書くと，対称行列において，

$$s = s^T$$

となります．

要素が乱数のマトリックスを生成するために，コマンドラインにおいて，

```
>> r=rand(3)
```

と入力すると，3行3列の乱数を要素とするマトリックスができます．例えば，

```
0.9501    0.4860    0.4565
0.2311    0.8913    0.0185
0.6068    0.7621    0.8214
```

を得ます．各要素は，0と1の間の一様乱数です．

正規乱数が必要ならば，

```
>> n=randn(4)
```

とすると，

```
 -0.4326   -1.1465    0.3273   -0.5883
 -1.6656    1.1909    0.1746    2.1832
  0.1253    1.1892   -0.1867   -0.1364
  0.2877   -0.0376    0.7258    0.1139
```

となります.

正方行列でない場合は,

```
>> m=rand(2,3)
```

とすると，例えば,

$$m=\begin{vmatrix}0.9501 & 0.6068 & 0.8913\\ 0.2311 & 0.4860 & 0.7621\end{vmatrix}$$

のように，2行3列の一様乱数マトリックスが得られます.

配列の管理は，すべてMATLABが行うことについて，すでに述べました.

上で求めた2行3列のマトリックスmに対して,

```
>> m(:,2)=[ ]
```

とすると，マトリックスの2列が削除されて,

$$m=\begin{vmatrix}0.9501 & 0.8913\\ 0.2311 & 0.7621\end{vmatrix}$$

となります．mの第2列に空列[ ]を代入したので，この列が配列から削除されて，mは2行2列のマトリックスに変わりました.

配列の次元が変わるので注意してください.

今度は，コマンドラインから,

```
>> m(3,2)=1
```

と入力します．2行2列のマトリックスに対して，その範囲を超えて，3行2列の要素を指定しました.するとmは,

$$m=\begin{vmatrix}0.9501 & 0.8913\\ 0.2311 & 0.7621\\ 0 & 1\end{vmatrix}$$

となります.

mに対して3行2列の要素を加えたので，マトリックスは3行2列のマトリックスに変わりました.

3行1列の場所に0が入っていることに注意してください.

不用意に，マトリックスに対して誤った要素を入力すると，マトリックスの次元が変わり，結果とし

てとんでもないエラーが発生するので，十分に注意してください．

繰り返しますが，MATLABの配列は，通常のC++言語などのオブジェクト指向言語に慣れた人にとって，エラーを呼びやすい構造になっています．細心の注意が必要です．

では，もう一つの例を示します．コマンドラインにおいて，

```
>> v=[1 2 3];
```

としてベクトル，

$$v=(1,2,3)$$

を作ります．

ここで，

```
>> p=ones(5,1)
```

と入力します．すると，

$$p=\begin{vmatrix}1\\1\\1\\1\\1\end{vmatrix}$$

という5行1列のマトリックスができあがります．そこで，pをマトリックスの添え字として使って，

```
>> q=v(p,:)
```

と入力すると，

$$q=\begin{vmatrix}1&2&3\\1&2&3\\1&2&3\\1&2&3\\1&2&3\end{vmatrix}$$

という5行3列のマトリックスが得られました．

MATLABがどういう処理をしたか，皆さん各自で考えてみてください．

通常のプログラミング言語ならば，for文などを使ってループ処理を行わなければならないところですが，MATLABは自動的に処理を行うので，一つのコマンドがループ処理を実行します．便利といえば便利ですが，それがかえってまちがいを呼び込む原因になることもあるので，十分に注意する必要があります．