数学／数式処理システム MATLAB の使い方（２）

MATLAB では数値的な計算やグラフ作成だけでなく、数式処理、つまり文字式の上での各種計算、例えば因数分解、展開、記号的な微分・積分、極限計算、級数計算などもできる。

実のところ、それができるというのは、解析学でやる問題のうちの計算問題はほとんどそれで解けてしまうということでもある。ちょうど、電卓があれば小学校の計算問題はそれで片づいてしまうようなものである。

と、ここであわてて教育的注意。
まず、コンピュータでできるということと、自分で手計算でできる、というより、そういった練習を通じて各種の関数やそれらの上の計算がどんなふうになるかの感覚をつかむこととは別問題である。したがって世の中ではコンピュータでできてしまうからといって、自分で手（や頭）を動かしてやってみることの価値がなくなるわけではない。次に、純粋に計算としてできるのは、実は問題のごく一部にすぎない。もっと本質的なのは、問題をどう分析するか、どのようにして数式の形で表すかといった点で、そこまでできてしまえば、あとはそれこそコンピュータでもできることになってしまう。

とまあ、以上を前置きにして、具体的な数式処理計算の仕方を紹介しよう。

1　簡単な例

前回説明したような数値的な処理と、文字式計算のような数式処理とは同じ MATLAB の中といっても全く別物と考えたほうがよい。ただそれでも、MATLAB の新しいヴァージョンになって、数式処理はだいぶやりやすくなった（前のヴァージョンはもっと大変だった）。

まず $y = \sin^2 x$ という関数を考えてみよう。これを定義するには、

```
>> syms x y
>> y = sin(x)^2
```

と打てばよい。すると画面には

```
>> y = sin(x)^2

y =

sin(x)^2
```

のように表示される（以下ではスペース節約のため、間の空白行は省略する）。

これを x で微分してみよう。

```
>> diff(y,x)
ans =
2*sin(x)*cos(x)
```

今度は x で積分してみよう。

```
>> int(y,x)
ans =
-1/2*sin(x)*cos(x)+1/2*x
```

次に $\displaystyle\int_0^\pi ydx$ という定積分を求めてみる。

```
>> int(y,x,0,pi)
ans =
1/2*pi
```

といった具合である（ただし、pi は円周率 π のこと）。

順番に説明しよう。

1. 数式処理の中で使う記号は、すべてあらかじめ宣言しなければならない。それが最初の "syms x y" である。ここで「記号」というのは、数式の中の変数、定数両方を指す。例えば

   $$y = ax + b$$

   という 1 次関数を定義する場合、x だけでなく、a, b も宣言しておかなければならない：

   ```
   >> syms x a b
   >> y = a*x + b
   ```

   コンピュータにとっては変数・定数（を表す文字）といった区別はないからである。

   ただ、上の例では y は宣言していなかったことに注意しよう。実は値が代入される変数、つまり従属変数は、宣言しなくても数式変数として用いることができる（もちろん宣言することもできるし、したほうがいいだろう）。

   数式変数と数値変数（前回のグラフ書きに用いたような変数）とは区別して使うように注意すること。例えば同じ x という変数名を、あるときは数式変数、あるときは数値変数として使ったりするとエラーや混乱の原因になる。数式変数は小文字、数値変数は大文字で始まるものを用いるといった使い分けをすればいいだろう。

2. 上の数式で、掛け算やべき乗の記号として '*' や '^' を使ったことに注意。前回のグラフ書きのときは '.*' や '.^' のように前にピリオドがついていた。割り算の '/' も同様である。理由の説明は長くなるので省略するが、区別には注意すること。

3. 前回とのもう 1 つの違いは、

   ```
   >> y = sin(x)^2
   ```

   のように、行の最後にセミコロン ';' がついていない点である（前回の例ではすべてついていた）。

   この違いは、計算結果を表示するかどうかの違いである。セミコロンがない場合は、上の例のように結果が表示される。ただしここで言う「結果」とは入力の真下にでる計算結果そのもののことで、グラフ表示などは別である。計算によっては結果表示がないものもあり、その場合にはセミコロンがなくても何も表示されない（上の syms、あるいは前回の mesh 等、グラフ表示コマンドなど）。

   練習：上の例でセミコロンを付けるとどうなるか、また前回の例でセミコロンをとるとどうなるかを調べてみよ。

4. 表示される結果は、入力が

   ```
   >> y = sin(x)^2
   ```

   のように変数への代入文になっている場合には

   ```
   y =
   sin(x)^2
   ```

のように「 変数 = ... 」という形式になるが、

```
sin(x)^2
```

のように式だけの場合には

```
ans =
sin(x)^2
```

のように ans という変数の値という形式になる。

ans というのは特別な変数で、直前に評価された (代入文でない) 式の結果が代入される。したがって次の式を打てば値も変わる。しかし次の式の中で ans を使うことができる。

```
>> x^2
ans = x^2               注：これ以下では ans = の後ろで改行せず、同じ行にまとめる。
>> ans^2
ans = x^4
>> ans^2
ans = x^8
```

5. diff, int は微分、積分を行う組み込み関数である。これについては次節で説明する。

6. 上の実行例からもわかるように、$1/2$ のような分数 (有理数) での処理が行われる。

   また pi というのは円周率 π のことである。これ自体も変数であるので "pi = 1" といった代入が可能だが、そういったことはやるべきではない。同様の組み込み定数としては inf $(= \infty)$ がある。

   練習： inf+inf, inf-inf などがどうなるか試してみよ。

2 微分と積分

数式として定義された変数 (従属変数) に対し、記号的な微分・積分が計算できる。

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

は f を x について微分する。この際、f の定義式の中にある他の記号は全て定数と見なされる。したがって diff はそのまま、偏微分の計算になっている。

```
>> syms a b x
>> f = a*x + b;
>> diff(f,x)
ans = a
>> diff(f,a)
ans = x
>> diff(f,b)
ans = 1
>> diff(f)
ans = a
```

上は順に、$\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial a}, \dfrac{\partial f}{\partial b}$ を計算したものになっている。最後の例からわかるように、微分する変数を省略することも可能で、その場合にはシステムの側でその変数を推定する。しかし思わぬまちがいの原因にもなるので、初心のうちは微分変数をきちんと書くようにしたほうがよい。

微分した結果も数式だから、それを再び微分することもできる。それが高階の微分になる。

```
>> syms x y
>> f = x^2*y^3;
>> diff(f,x)
ans = 2*x*y^3
>> diff(ans,x)
ans = 2*y^3
>> diff(diff(f,x),x)
ans = 2*y^3
>> diff(f,x,2)
ans = 2*y^3
```

最後の形のように、“diff(f,x,n)”（n は自然数）とすると、n 階の微分が計算できる。同じ f について：

```
>> diff(diff(f,x),y)
ans = 6*x*y^2
>> diff(diff(f,y),x)
ans = 6*x*y^2
```

となり、$\dfrac{\partial^2 f}{\partial x \partial y} = \dfrac{\partial^2 f}{\partial y \partial x}$ となることがわかる。

積分についても同様で、“int(f,x)” は f の x による不定積分を計算する。ただし、微分と違って不定積分は必ずしも計算できるとは限らず、被積分関数によっては特殊関数として表示されたり（$\int e^{-x^2} dx$ など）、それでも計算できない場合には元の式：int(f,x) がそのまま返される。

定積分を計算するには、上端を a、下端を b として、“int(f,x,b,a)” とすればよい（a, b の順番に注意）。

```
>> int(x^2,x,0,1)
ans = 1/3
```

なお積分でも積分変数以外の記号は定数と見なされる。したがって重積分などにも利用可能である。

定積分の計算には、不定積分を求めてから次の subs 関数を使う方法もある。

3　その他の基本機能

3.1　代入

与えられた数式の変数に数値や式を代入するには subs を使う。

```
>> syms x y
>> f = x^2*y^3:
>> subs(f,x,2)
ans = 4*y^3
>> subs(f,x,y)
ans = y^5
>> subs(f,y,x+1)
ans = x^2*(x+1)^3
>> subs(sin(x),x,pi)
ans = 0
```

subs の第 1 引数は代入対象となる変数や数式、第 2 引数は代入される変数、第 3 引数は代入する数値や数式である。数式中の複数の変数に同時に代入を行うには次のようにする。

```
>> subs(sin(x)+cos(y), {x,y}, {1,2})
ans = sin(1)+cos(2)
```

結果からわかるように、これは数式中の x には 1、y には 2 を代入したことになる。なお：

```
>> subs(subs(x^2*y^3, x, 1), y, 2)
ans = 8
>> subs(subs(x^2*y^3, y, 2), x, 1)
ans = 8
```

となって subs も入れ子にして使うことができるが、これは一般にはうまくいかない。

```
>> syms x y
>> f = x^2+y^3;
>> subs(f, {x,y}, {y,x})
ans = y^2+x^3
>> subs(subs(f,x,y),y,x)
ans = x^2+x^3
>> subs(subs(f,y,x),x,y)
ans = y^2+y^3
```

練習：なぜ上のようになるか考えてみよ。
また subs(subs(sin(x)+cos(y),x,1),y,2) とするとどうなるかを調べ、その理由を考えてみよ。

3.2 式の変形

同じ数式でも様々な形に変形することができる。代表的なものが因数分解と、その逆の展開である。

```
>> syms x y
>> f = (x+1)^5
f = (x+1)^5
>> expand(f)
ans = x^5+5*x^4+10*x^3+10*x^2+5*x+1
>> factor(ans)
ans = (x+1)^5
```

上のように、expand は式の展開を、factor は因数分解を行う。ちなみに同じように見えるが、展開が機械的で簡単なのに比べて、因数分解は数式処理の問題としてはきわめて難しい部類に入る（もちろんここで言う因数分解は、高校などでやる簡単なものでなく、変数が多数あり、次数も高い式に対する因数分解を指す）。例えば factor(expand((x+y+1)^100) がどれだけの計算か、考えてみよ（ちなみに実際にやってみたら、数分かかった）。なお、factor(n)（n は自然数）とすれば、n の素因数分解が返される。

数式をできるだけ簡単な形にする関数として、simple(f) がある。これはいろいろな式変形を試して結果の文字数が一番少ないものを返す。各種の式変形の結果も表示されるので、どういった変形があるかを見ることもできる。

4 数値計算、グラフ作成との関係

f が x だけを変数に含むなら、ezplot(f) によってそのグラフを表示することができる。x 以外に定数を表す記号があるなら、あらかじめ subs によって数値を代入しておく必要がある。単に ezplot(f) とした場合には x の定義域をシステムが自動的に選択するが、ezplot(f,a,b) とすれば x が [a,b] の範囲でグラフが表示される。1 変数関数のグラフをいろいろ細かい設定を与えて表示するには plot を使うが、説明は省略する。

2 変数関数の場合にはグラフ表示はもっと面倒になる。この場合には前回の 2 次元グラフ表示できる形に直してやること、つまり数式の形の表現を数値に直してやることが必要になる。詳細は省略するが、例えば $f(x,y) = x^2 - y^2$ のグラフを表示するには次のようにする。

```
>> syms x y
>> f = x^2-y^2;
>> [X,Y] = meshgrid([-1:0.2:1]);
>> Z = double(subs(f,{x,y},{X,Y}));
>> mesh(X,Y,Z)
```

ポイントとなるのは、X,Y に 2 次元格子を作ること、subs によってその各点に数値を代入すること、そして double によって数式として表された格子を数値格子に直すことである。なお上の subs の計算は大変時間がかかるので注意 (格子の点の数をあまり多くしないこと)。

5 セッションの記録

matlab で実行している内容をファイルに記録するには、diary コマンドを使えばよい。matlab の中で "diary ファイル名"、"diary off" とすれば、その間の表示がファイルに記録される。ただしここに記録されるのは文字の入出力だけで、グラフなどの画像表示は残らない。

また MATLAB に入る前に script コマンドを起動する方法もある。

```
dream% script
Script started, file is typescript
dream% matlab
>> ...
>> quit
   ...
dream% exit
Script done, file is typescript
```

と表示され、typescript というファイルにその間の端末上でのやりとりがすべて記録される。ただし、改行文字やバックスペース文字などもそのまま記録されるので実際にはその編集が必要であるが、詳細は省略する。編集を行うためのコマンドを用意しておく予定。