情報科学科２年生応用線形代数のためのMATLAB 入門

2014年５月

数理･計算科学専攻，情報科学科
福田光浩

目次

1. 概要
2. Command Window
3. 数値と演算記号,help
4. ベクトルと行列
5. 線形方程式系
6. 固有値と固有ベクトル
7. Graphics
8. Toolbox等
9. Programming の際の注意

1. 概要

- MATLAB は超高級電卓
  - 特に，行列の和，逆行列等の線形代数演算を装備
- ままざまなグラフが描ける
- MATLAB はプログラミング言語
  - プログラミングが”超簡単” --- ベクトル，行列を含むアルゴリズムの記述に適している.

    MATLAB なら C = A*B;

    C, JAVA なら

```
for (int i=0; i<n; ++i) {
    for (int j=0; j<n; ++j) {
        C[i][j] = 0.0;
        for (int k=0; k<n; ++k) {
            C[i][j] = C[i][j] + A[i][k]*B[k][j];
}}}
```

  - C, JAVA等に比べて（繰り返し・反復演算の）処理速度は遅い.
  - ただし，プログラミングについては述べない．下記のHP 参照.
- ここで述べるのはMATLABのごく一部の機能.
- 数理科学，工学の研究に極めて強力な道具.
- MATLABは有料であるが，類似したソフトウェアとしてGNU Octaveが存在する．Linux,Mac,そしてcygwinを通してWindowsにもインストール可能.

2. Command Window

- MATLAB を起動するには，アプリケーション,MATLAB内のMATLABアイコンを マウスボタンでクリックすればよい.
- 起動するとcommand windowでさまざまなcommand（命令）が実行可能.
- “>>”の後にcommandを記述.
- Help から様々な情報が得られる.

- 変数を使うことができる.
- 変数名は英字で始まり，英字、数字，_からなる31文字以内．ローマ字の大文字と小文字は区別される．文字はすべて半角文字.
- MATLABの終了は >> **exit**

```
>> s = 1 + 2
s =
3
>> fun = sin(pi/4)
fun =
0.7071
>> s + fun
ans =
3.7071
```

```
>> format long
>> fun
fun =
0.70710678118655
>> format short
>> fun
fun =
0.7071
```

3. 数値と演算記号, help

- 整数, 実数, 複素数が使用可能
- 小文字 i が虚数単位.
- % の後はコメントで無視される.

```
>> fun = sin(pi/4)  % =1/sqrt(2), pi=3.14･･･
fun =
0.7071
```

```
>> xint = 10
xint =
10
>> xreal = 10.01
xreal =
10.0100
>> xcomplex = i + xreal
xcomplex =
10.0100 + 1.0000i
```

```
>> realmin, realmax
ans =
2.2251e-308
ans =
1.7977e+308
```

演算記号

- 加算 : +
- 減算: -
- 乗算: ＊
- 除算: /　または ＼
- すべて半角文字（ここでは，印字の都合上＊と＼は全角を使っていることに注意！）

```
>> a = (2/3+1)＊4
a =
6.6667
```

```
>> ld = 2╲3, rd = 2/3
ld =
1.5000
rd =
0.6667
```

(╲, / はベクトル，行列の演算に拡張される．後述）

- べき乗

```
>> a = 2.5^3
a =
15.6250
```

- help

>> help inv

INV Matrix inverse.

INV(X) is the inverse of the square matrix X. A warning message is printed if X is badly scaled or nearly singular.

See also SLASH, PINV, COND, CONDEST, LSQNONNEG, LSCOV.

Overloaded methods

help sym/inv.m

- Command が2行以上にまたがるときは,
  … で行の最後をつなぐ.

```
>> x = sin(1) - sin(2) + sin(3) - sin(4) …
+ sin(5) - sin(6) + sin(7) - sin(8) …
+ sin(9) - sin(10)
x =
0.7744
```

- 結果をprintしないときのcommand末は ;
- 結果をprintするときは , または空白

```
>> u = 2 + 3, v=u+6; v+1 % v=11
u =
5
ans =
12
```

4. ベクトルと行列

- 横（行）ベクトル

```
>> a = [1 2 3] % or, a=[1, 2, 3]
a =
1 2 3
```

- 縦（列）ベクトル

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

- **内積，転置**; a=[1 2 3], b=[1; 1; 2]

```
>> b’＊b, a＊b % b’ = bの複素共役転置
ans =
6
ans =
9
```

- **要素ごとの積**

```
>> a.＊a
ans =
1  4  9
```

- **要素ごとのべき乗**

```
>> a.^3
ans =
1  8  27
```

- **ベクトルの長さ**; a=[1 2 3], b=[1; 1; 2]

```
>> length(a), length(b)
ans =
3
ans =
3
```

・ Euclid ノルム

```
>> a = [1 2 3]; normOFa = norm(a)
normOFa =
3.7417
>> sqrt(a * a')
ans =
3.7417
>> unitVector = a / normOFa
unitVector =
0.2673  0.5345  0.8018
```

- 行列

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

- 行列のサイズ

```
>> size(A)
ans =
2   3
```

- **行列式**

```
>> A = [1 2; 3 4]; det(A) %=1∗4 - 2∗3
ans =
-2
>> det(A') %=det(A)
ans =
-2
```

- **特殊な行列**

```
>> n=4; N = -2:n
N =
-2 -1 0 1 2 3 4
>> zeroVector=zeros(1,n)
zeroVector =
0 0 0 0
```

```
>> vectorOfOnes=ones(1,5)
vectorOfOnes =
1 1 1 1 1
>> matrixOfOnes=ones(3,4)
matrixOfOnes =
1 1 1 1
1 1 1 1
1 1 1 1
```

- **単位行列**

```
>> idMatrix=eye(3,3)
idMatrix =
1 0 0
0 1 0
0 0 1
```

- **部分行列**（ : はすべての行または列を表す）

```
>> D=idMatrix([3 1],:) % idMatrixの3,1行
D =
0 0 1
1 0 0
```

・ 行列の演算

```
>> A=[1 2; 3 4]; B=ones(2,2); C=A * B+3 * B
C =
 6    6
10 10
>> D1=C * [2;3], D2=[2, 1] * A
D1 =
30
50
D2 =
5  8
```

・ 要素ごとのかけ算，割り算，べき乗

```
>> A=[1 2; 3 4];B=2*ones(2,2);C=A.*B
C =
2  4
6  8
>> A2=C./B, C2=C.^B
A2 =
1  2
3  4
C2 =
  4  16
36  64
```

· 对角行列

```
>> d=[1 3 5]; D=diag(b)
D =
1  0  0
0  3  0
0  0  5
>> d1 = diag(D)
d1 =
1
3
5
```

5. 線形方程式系: A x = b

```
>> A=[3 1; 2 4]; b=[8; 3]; x=A¥b
x =
 2.9000
-0.7000
>> r=b-A * x
r =
 1.0e-15 *
 0.8882
-0.4441
```

・ LU分解(A=LU, L:下3角, U:上3角)

```
>> A=[3 1; 2 4]; [L, U] =lu(A)
L =
1.0000         0
0.6667  1.0000
U =
3.0000  1.0000
     0  3.3333
>> B=L * U
B =
 3   1
 2   4
```

- **逆行列**

```
>> A = [3 1; 2 5]; B = inv(A)
B =
 0.3846  -0.0769
-0.1538   0.2308
>> C = A * B
C =
 1.0000         0
-0.0000   1.0000
```

6. 固有値と固有ベクトル

```
>> A = [2 1;1 2];
>> [P, D] = eig(A)
P =   0.7071  0.7071
     -0.7071  0.7071
D =  1    0
     0    3
```

(D の対角が固有値, Pの列が固有ベクトル）

```
>> lambda_1 = D(1,1); %固有値
>> p_1 = P(:,1); %固有ベクトル
>> Ap_1 = A * p_1, lambda_1 * p_1
Ap_1 =  0.7071
       -0.7071
ans =  0.7071
      -0.7071
>> norm(A * P(:,2) - D(2,2) * P(:,2))
ans = 0.0000
```

```
>> P’ * P %P.’ → P’
ans = 1.0000  -0.0000
        -0.0000   1.0000
```

(A:対称行列 ===>固有値は実数，Pは直交行列）

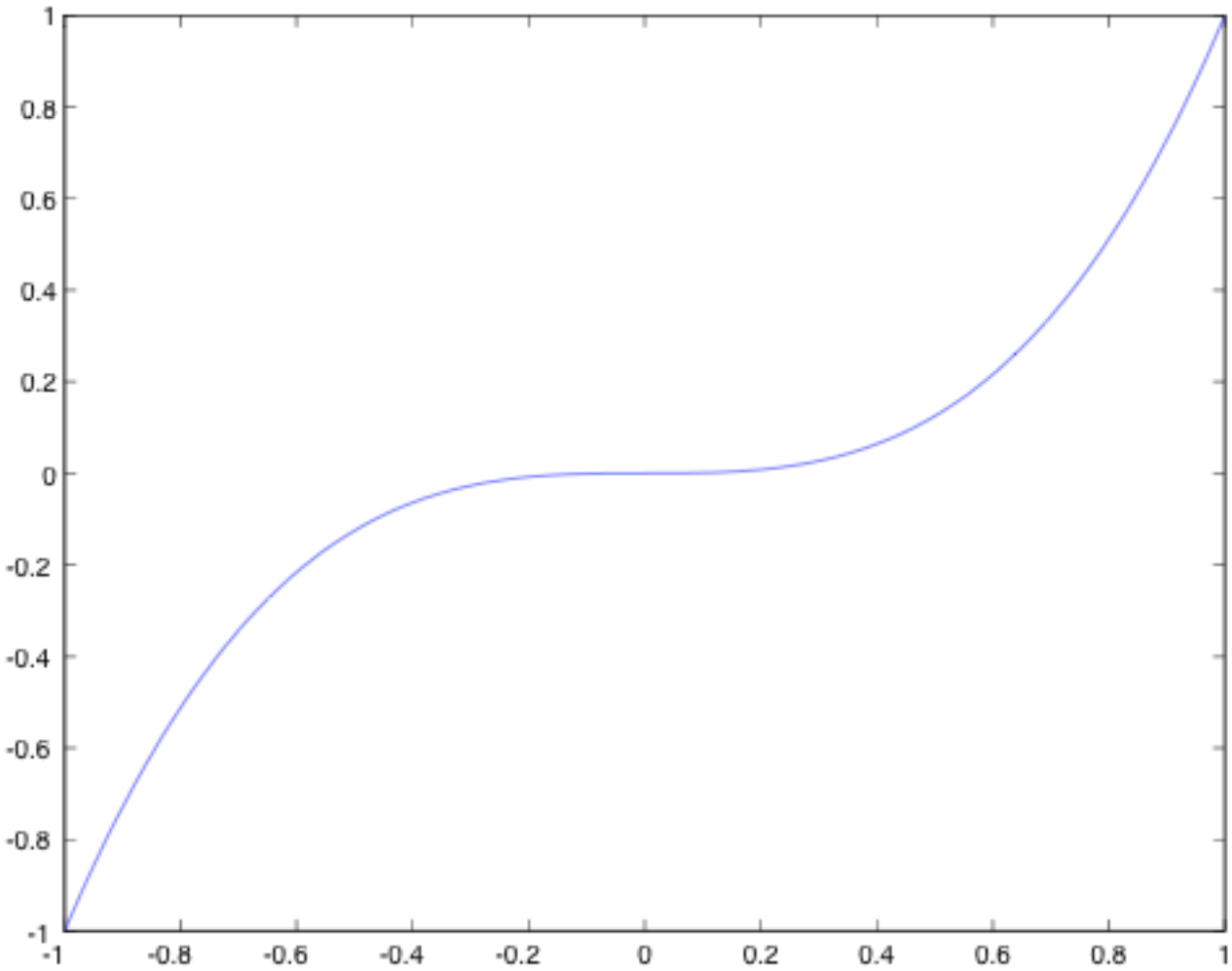
```
>> P’ * A * P % = D %P.’ → P’
ans = 1.0000  -0.0000
        0.0000   3.0000
```

(Aの対角化）

7. Graphics

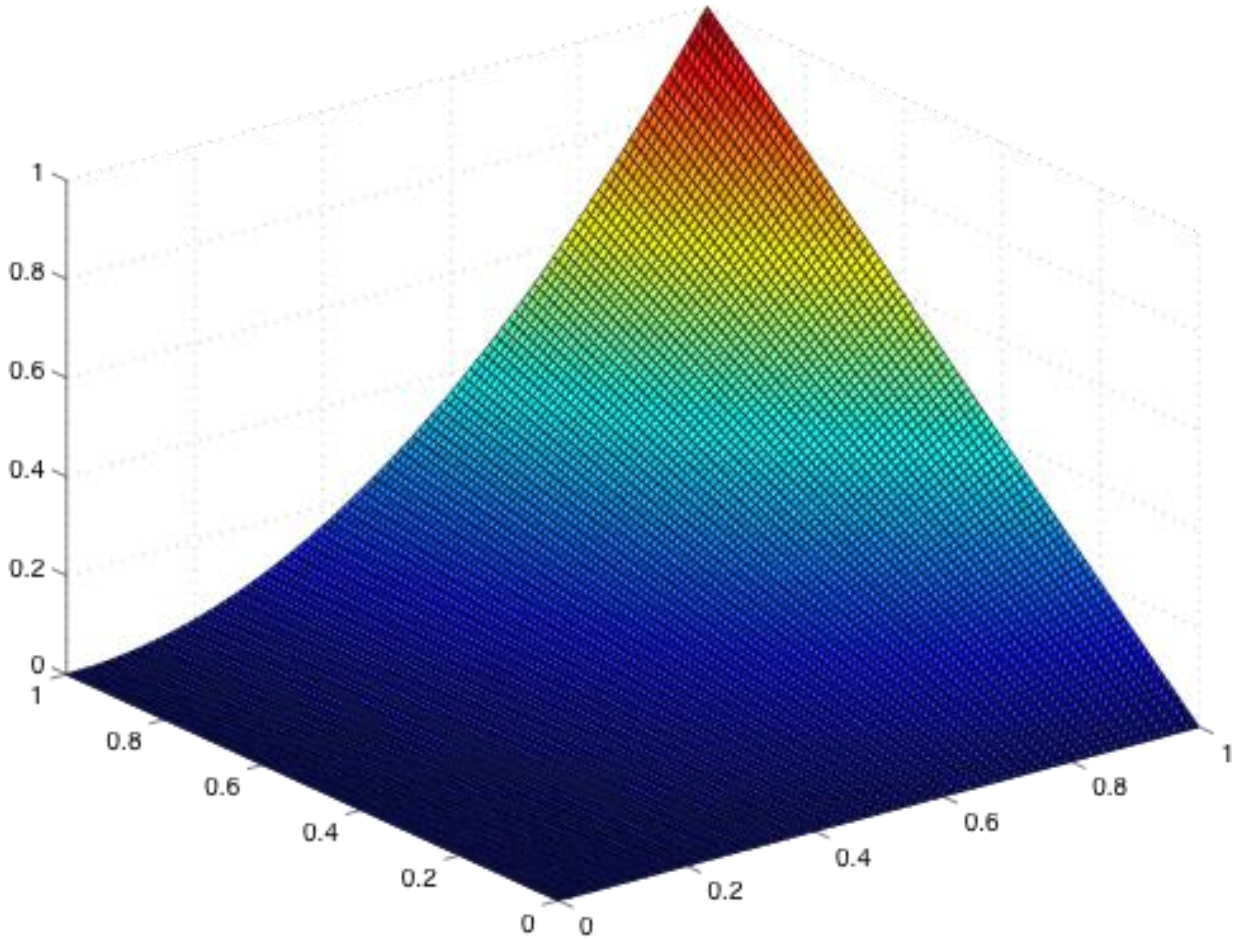
- さまざまな2次元graphics, 3次元graphicsのための関数が準備されている.
- 計算実験の可視化等に非常に有用.
- ここでは, 簡単な例をあげる.

```
>> x=0.01 * [-100:100]; plot(x,x.^3)
```



- 3次元Graphics
- z = x ∗ y^2, 0 <= x, y <=1.

```
>> x=0.01 ∗ [0:100]; y=x; z=x’ ∗ y.^2;
>> surf(x,y,z)
```


1
0.8
0.6
0.4
0.2
0
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

8. Toolbox等

- Optimization Toolbox --- 線形計画問題等の解法を含む.
- Symbolic Math Toolbox --- 多項式の演算等をサポート.
- その他のToolbox（有料）.
- MATLAB で既述された free software が多くある.

9. Programming の際の注意

- C, Java を知っていれば容易に programming できる.
- 多種，多様な関数が用意されているので，それらを有効利用するとよい.
- 有用な関数が多数ある．ほとんどの関数は行列を変数としている.

chol, rand, sort, max, min, sum, …

- 疎なベクトル，行列を簡単に扱える.

- 最初は速度を気にせずに分かりやすいprogram を組むこと.
- 高速化するには，MATLAB の組み込み関数を駆使して，繰り返し計算・反復計算を減らすこと. 10~100倍程度速くなることが頻繁に起きる.
- ただし，そのような工夫はかなり技巧的，かつ，職人芸的.
- http://www.mathworks.com/discovery/matlab-acceleration.html に結構有用な情報がある.