本稿は Linux Japan 誌 2002 年 06 月号に掲載された記事に補筆修正したものです.

数値ツールで行列計算

春先は毎年新年度の講義の準備があって苦しいやら楽しいやらで過ごしています. 今年は行列計算パッケージについて下調べをしていたので, そのあたりのお話しをします. 行列を扱うことを主眼においた数値ツールは MATLAB[1]W^3 に尽きると思いますが, もちろんオープンとなっているツールしか眼中にはありません. さて MATLAB 互換の数値ツールといえば, Octave[2]W^3 や Scilab[3]W^3 あるいは Rlab[4]W^3 が直に思い浮かびますが, これら有名どころからは書式が一番似ていると思われる Octave を紹介しましょう. また, もっと行列に特化している MaTX[5]W^3 も是非取り上げねばと思っていました. MaTX の作者は日本の方ですから開発者のドキュメントも日本語で馴染みやすいし, また書籍が縁あって筆者の勤務する大学の出版局から出ていますので, まあ親近感もあったりして (もちろん出版局からの依頼があったわけではありません, 念のため). それと, ちょと方向性が違っていますが任意精度計算も可能という大きな特徴がある PARI-GP[6]W^3 を加えて, 行列の扱い方の違いを中心にまとめていきます.

大石先生の「Linux 数値計算ツール」[7]W^3 にもありますが, 行列計算のプログラムはアルゴリズムを理解する目的で自分で少し書いてみることに意味はありますが, 実用的なものまで体系的に仕上げるのは無謀というものです. やはり, 定評のある汎用ライブラリ (LAPACK, BLAS) を用いるべきでしょう. さらに, 実際に問題を解く場合にも, ライブラリを直に使ったプログラムを組むことよりも, 数値計算ツールのスクリプトとして仕上げた方が, 可読性もまたそれゆえ可搬性も高くなることが期待できます.

配列の生成

数学の講義では, スカラーからベクトルそして行列 (あるいはテンソル) へと進展します. ここでも, ベクトルの扱いを先にまとめておきます.

行ベクトル

まずは, 人が手で書くように定義しましょう. 行の先頭に Octave, MaTX, PARI-GP を略して O,M,G と記載することにします.

```
OMG: A=[1,2,3]
OM: A=[1 2 3] も可
```

ただし型が違っていて, PARI-GP ではベクトルが, Octave と MaTX では行列が生成されます. 変数あるいはオブジェクトの型の情報は

```
O: whos A
   prot  type           rows   cols  name
   ====  ====           ====   ====  ====
    rwd  matrix            1      3  A
M: whos A
   Name         Class     Size(Value)
      A      Re_Matrix     1 x   3
G: type(A)
   %2 = "t_VEC"
```

として知ることができます (Octave には `typeinfo()` 関数もあります). MaTX では配列への型変換が以下のように可能です.

```
M: A=Array([1,2,3])
```

Octave と MaTX では範囲と増分を指定して宣言を行う関数が用意されています. 例えば, 0 から始まって増分 1 で 10 までの値を要素とする `A` は

```
OM: A=[0:1:10]
O: A=0:1:10 も可 (本来は範囲指定子)
```

と定義します. Octave では相変わらず行列で MaTX では今度は配列です. 型はさておき, 画面上には 0 から 10 までが **1 行**に 11 個表示されます (増分が省略された場合, 例えば `[1:10]` のような記述では増分は 1 と解釈されます). 数学的に素直な扱いは次数 11 の行ベクトル (横ベクトル) と考えるのがよいでしょう. しかし, 一般にはベクトルを行列として扱うこともままありますから, その場合には 1×11 行列と考えます. これは,

```
OM: A(i) または A(1,i)
```

として i 番目の要素にアクセスできることからも明らかです. さて, この方法ですと $-\pi$ から π までを 10 等分する配列は

```
OM: A=[-pi:pi/5:pi] (MaTX では PI)
O: A=-pi:pi/5:pi も可 (型は範囲です)
```

などとなります. ここでも要素の数は 11 ですからご注意. 割り切れない場合にはどなるでしょう. 例えば

```
[0:1.1:10] や [0:0.1:pi]
```

とした場合, 要素の数 (また最後の要素の値) がどうなるか確かめてください. 増分を与えるのが面倒ならば, 要素数を与えて行ベクトルを生成する組み込み関数を使いましょう.

```
OM: x=linspace(0.0,2*pi,101)
```

両端が要素に含まれますから, 区間の分割数は要素数より 1 小さいくなります. したがって, 区間をきれいに 100 に分割するには要素数を 101 としなければなりません.

PARI-GP では行ベクトルの生成について, 手書きする以外に明示的な関数が用意されています. 例えば

```
G: A=vector(101,i,0.01*(i-1)*Pi)
```

とすると, $0.0, 0.001\pi, 0.02\pi, ..., \pi$ の行ベクトルが得られます. また, ベクトル (1 次元配列) にアクセスするには C 言語と同じく

```
G: A[i]
```

とします. 行列とはオブジェクトが異なりますから, `A[1,j]` でアクセスしようとするととエラーとなります. 厳密な扱いをしているといえましょう.

こうしてみると型の問題もさることながら, 配列の大きさを直接指定することに今までとは違ったセンスが要求されることに気づきます. 実は, 閉区間の両端を含むような配列を宣言するには, このような方法が理にかなっていることは明らかです. 一般のプログラミング言語では 分割数+1 個のサイズを宣言しなければならないからです.

しかしプログラミング言語の癖に慣れているとどうしても直接要素数を指定したくなります. それには別の方法があり, 行列のところでまとめて説明します.

ここまでですと，プログラミング言語に比べてあまり利点が感じられないかもしれませんが，関数が絡んでくると便利さに感動します．すなわち，関数にベクトル（行列）を与えると，同じ次元の関数値のベクトル（行列）を生成するのです．Octave，MaTX，PARI-GP いずれも

```
OMG: x=[0.0,0.5,1.0];sin(x) または
     sin([0.0,0.5,1.0])
```

としてみてください．行ベクトル $(\sin(0.0), \sin(0.5), \sin(1.0))$ が得られます．

列ベクトル

行列演算では行ベクトルよりも列ベクトル（縦ベクトル）を用いることが多いです．すでに行ベクトルがある場合には，転置命令を用います．

```
O: A'
M: A' または trans(A)
G: A~ または mattranspose(A)
```

とすると，11 行にわたって要素 1 つずつが表示されます．もちろん，最初から

```
O: [1,2,3]'
M: [1,2,3]' または trans([1,2,3])
G: [1,2,3]~ または
   mattranspose([1,2,3])
```

と記述できます．個々の要素を直接代入しながら宣言するには，

```
OG: A = [1;2;3]
M: A = [[1][2][3]]
```

と記述します．`[[1][2][3]]` は PARI-GP ではエラーとなるのでまだ被害が少ないのですが，Octave では行ベクトル `[1,2,3]` となってしまいますから要注意です．PARI-GP では列ベクトルの生成関数があります．

```
G: A=vectorv(101,i,0.01*(i-1)*Pi)
```

また Octave では，いきなり列ベクトルあるいは行ベクトルの要素を定義できます．紛らわしいですが，

```
for i=1:10
  A(i)=i^2;
endfor
A
```

で列ベクトルが，

```
for i=1:10
  C(1,i)=i^2;
endfor
C
```

で行ベクトルが定義できることを確かめてみてください．MaTX では宣言されていない配列への代入はエラーとなります．

行列の生成

まず紙と鉛筆で書くように定義する方法から（宣言は不要です）．MaTX は書式が違います．

```
OG: A =[1,2,3; 4,5,6; 7,8,9]
M: A = [[1 2 3][4 5 6][7 8 9]]
```

PARI-GP では，ベクトルや行列の連結を行う関数

```
G: X = connect(u,v)
```

が用意されています．Octave や MaTX ではベクトルを並べて行列を作成する関数はありません．しかし，ベクトルの積をとれば行列ができます．すなわち，一般に $N \times K$ と $K \times M$ の行列の積は $N \times M$ の行列ですから，N 次の列ベクトルと M 次の行ベクトルをかければ良いことになります．Octave と PARI-GP では

```
O: x=[1,2,3]; C=x'*x
G: x=[1,2,3]; C=x~*x
```

とすることができます．MaTX では型にこだわり，以下のように配列（Array）を行列（Matrix）に型変換しなければなりません．

```
M: C = Matrix(x')*Matrix(x)
```

サイズを確保する（宣言する）ためだけに，わざわざベクトルのかけ算をするのは面倒です．むしろ定義済みの行列を宣言するのがよいでしょう．例えば，要素が全て 0 または 1 である行列

```
O: zeros(N,M), zeros(N)
   ones(N,M), ones(N)
M: Z(N,M), Z(N)
   ONE(N,M), ONE(N)
```

が使えます．引数が 1 つの場合には正方行列と解釈されます．単位行列（対角成分が 1 で他成分がゼロ）も有用です．

```
O: eye(N,M), eye(N), eye
M: I(N,M), I(N)
```

Octave の eye（=1）は MATLAB との互換のために設けてあると help に説明があります．あるいは，(0,1) 内の乱数を要素とする行列を生成する組み込み関数は両者に共通ですから記憶しやすいかもしれません．ただし，MatX では配列として生成されるので，Matrix で変換しなければなりません．

```
O: rand(N,M), rand(N)
M: Matrix(rand(N,M)), Matrix(rand(N))
```

PARI-GP では行列の明示的な生成関数

```
G: matrix(N,M,i,j,expr(i,j))
```

を用いれば，上記の行列を全て得ることが可能です．すなわち

```
zeros = matrix(N,M,i,j,0)
ones = matrix(N,M,i,j,1)
rand = matrix(N,M,i,j,random)
eye = matrix(N,M,i,j,if(i==j,1,0))
```

と定義できます．ただし，単位正方行列の生成関数だけはあらかじめ `matid(N)` という名前で組み込まれています．

配列を確保してから（Octave では配列を確保していなくとも）個々の要素を多重ループ内で代入するという，プログラミング言語と同等の方法は筆者にもすぐ書くことができます．for ループは，MaTX では C 言語と全く同じ記述が可能ですが，Octave では **for** *var*=*start*:*end* **...** **endfor** という書式です．

```
for i = 1:10
  for j = 1:6
    A(i,j)=sin(0.1*i)*sin(0.1*j);
  endfor
endfor
```

このように $f(i)$, $g(j)$ の直積となっている場合には簡単に

```
A=sin([0.1:0.1:1.0]')*sin([0.1:0.1:0.6])
```

と書くことができます．もっとも，らしくするために無理やり書いている状態でして，筆者は for ループを使う方が安心できます．

行列要素へのアクセス

行列の要素へのアクセスは成分 1 つ単位だけではなく，ベクトル単位，さらに一般化して範囲を指定して行えるようになっています．PARI-GP においては，部分行列の取り出しにはそのための関数があります．範囲指定も独特ですし，実は`"i1..i2"` という直感的な範囲指定ではなく自然数も書けます．するとそれを 2 進数とみて，ビットが立っている部分が取り出される，すなわち，任意の位置を指定して一辺に取り出されるというちょっと凝ったつくりです．

```
OM: A(i,j), A(i,:), A(:,j),
    A(i1:i2,j1:j2)
G:  A[i,j], A[i, ], A[ ,j]
    vecextract(A,"i1..i2","j1..j2")
```

なお，MaTX では行列を 1 次元配列にみて `A((i-1)*M+j)` としてアクセスできます．

範囲指定

Octave と MaTX の範囲指定は基本的には両端と増分の 3 つから成り立ちます．増分を省略すると 1 とみなされます．また増分に負の数を指定できますから，-1 ならば並べ順を反転できます．

```
OM: [10:-2:1], A(2:5,10:-1:1)
```

などで確認してください．

ベクトルと行列の演算

演算子（とくに加減乗除 `+,-,*,/`）の多重定義により，ノートに記した数学の式がそのまま記述できることがツールの便利な点と思います．表記に関しては，除算（逆行列の乗算）のみが少し違います．

```
OMG: A+B, A-B,, A*B, B*A,
     A^(-1)*B, A*B^(-1)
OM: B\A, B/A, inv(B)*A, B*inv(A)
MG: 1/A*B, A*1/B
M:  A~*B, a*B~
```

左除算（左から逆行列をかける）を表す演算子 “\” は MATLAB 互換のツールならではのものです．実は除算は逆行列を求めて乗算しているわけではないので，その意味では正直なのかもしれません．rand(N) を用いて正方乱数行列 `A,R` を生成し，以下の式を実行してください．

```
OM: A^(-1)*R-A\R または R*A^(-1)-R/A
```

答えは 0 にはなりません．逆行列を求めずに，正確にしかも速く除算を実行するアルゴリズムが実装されているからです．Octave, MaTX では `inv(A)`≡`A^(-1)`，つまり左右から乗算されても同じ結果を出します．MaTX の `A~` も同じです．ところが MaTX の `1/A` は左からの乗算に対しては `X*1/A=X/A` と計算されようです．PARI-GP では `1/A` ≡ `A^(-1)` です．

行列と指数関数

p を整数とすれば，正方行列の累乗 A^p は行列 A を p 回かけたものとして定義できますし，A^{-p} は A の逆行列 A^{-1} を p 回かけたものと定義できます．もちろん $A^0 = I$（単位行列）です

```
OMG: A^p  p：整数
```

行列の累乗が定義されれば，行列指数関数は

$$e^A = I + A + \frac{A^2}{2!} + \cdots + \frac{A^p}{p!} +$$

と定義され，MaTX の `exp(A)` はこれを採用しています．実は，この定義通りに計算するのはかなり面倒です．もう 1 つの定義の仕方があります，要素ごとの指数をとるもので，

$$e^A = \begin{pmatrix} e^{a_{11}} & e^{a_{12}} & \cdots \\ e^{a_{21}} & e^{a_{22}} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

これは簡単に計算できますね．Octave と PARI-GP では `exp(A)` はこの要素ごとの指数関数となっています．また Octave では行列指数は `expm()` と命名されています．ベクトルでは要素の指数しか定義できませんから，スカラー，ベクトル，行列を通じて定義が同一となる方を選択したのだと思います．実際，行列そのものを変数にした関数には後ろに `m` をつけて区別しています． `logm()` や `sqrtm()` などがこれにあたります．

要素ごとの演算

というわけで，線形代数の講義では教わらないのですが要素ごとの乗除算がなくてはならないものとして行列計算ツールにはあります．記号 ‘‘.’’ を前に付けて区別します．加減算も一応ありますが同じ結果となるのであまり意味がないでしょう．PARI-GP はこの演算がありません．

```
OM: A.*B  A./B
```

突然グラフィック

このあとは，連立方程式の解，固有値と言及しなければならないでしょうが，少し疲れましたので，グラフィックに話題を移しましょう (^^; この 3 つのツールはグラフィック部分を Gnuplot に任せています．正確にいえば，PARI-GP は自前のものあるのですがちょっと使い難いので Gnuplot を使うようにコンパイルし直すことを薦めます．

PARI-GP で Gnuplot を使う

以下の手順に従って，Gnuplot を使うようにコンパイルしましょう．

1. PARI-GP のソース (バージョン 2.1.3) を展開し，そこに gnuplot というディレクトリを作成し，gnuplot-3.7.* のソースを展開する．
2. まず gnuplot のディレクトリに移り，configure，make により gnuplot をコンパイルする．
3. 以下のように，PARI-GP で必要なオブジェクトファイルをまとめたライブラリを作成する．
   `ar cr libgnuplot.a version.o util.o term.o bitmap.o stdfn.o`
4. `libgnuplot.a` を `/usr/lib` または `/usr/local/lib` などに置く．
5. PARI-GP のトップディレクトリ（１つ上）に戻って，Olinux686/Makefile の `PLOTLIBS` の項に `-lgnuplot` を追加する（場合によっては，`-lgd -lpng -lz` も）．
6. Gnuplot を使うように `./Configure --graphic=gnuplot` とした後，`make; make install` する．

プロット命令

これで三者とも Gnuplot 自身あるいはその一部を用いてグラフを描くことになりましたから，同じ関数を描かせてみましょう．まずは，Octave ですが，なるべく MATLAB と類似の記述で次のように書けます（図 1）．

List 1 Octave のプロットスクリプト

```
x=linspace(-2*pi,2*pi,101);
xm=linspace(-2*pi,2*pi,40);
ym=xm;
f=sin(2.5*x).*cos(x);
# グリッドデータの作成
fm=3*(sin(xm)./xm)'*(sin(ym)./ym)
# 題，軸，範囲の設定
title("Octave");
xlabel("X-label");
ylabel("Y-label");
axis([-2*pi, 2*pi]);
grid("on");
# 区画 (1,1) に２次元プロット
subplot(1,2,1)
plot(x,f,";sin(2.5x)cos(x);")
# 区画 (1,2) に３次元プロット
subplot(1,2,2)
mesh(xm,ym,fm)
# 表示の維持
pause
```

実は，`gplot`， `gset` コマンドにより生の Gnuplot が実行できますので，筆者はそちらで書く方が楽です．MaTX は体系をみると MATLAB に近いのですが，名前が違っています．MaTX は複数の描画面を持っているので，mgplot では描く面を指定します．このあたりは MATLAB よりは拡張されています．しかし，３次元プロット命令 `mesh()` や カラーマップの `surf()`（Gnuplot3.7.x にはない機能なので仕方ありませんが）などは未だ実装されていないです．

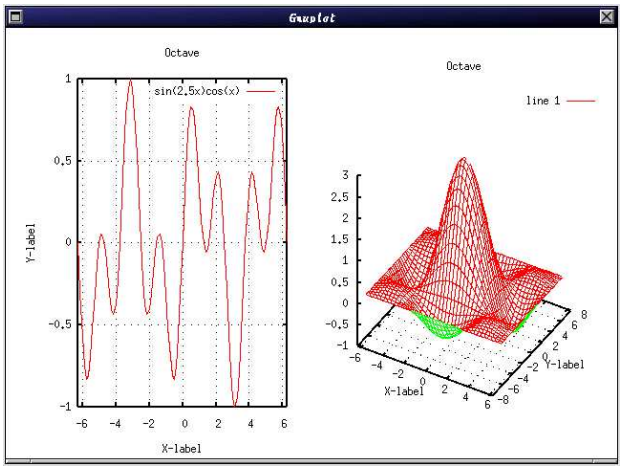

図 1　Octave による２次元，３次元プロット

List 2 MaTX のプロットスクリプト

```
x=linspace(-2*PI,2*PI,101);
f=sin(2.5*x)*cos(x);
mgplot_options(1,"-geometry 512x384");
mgplot_title(1,"MaTX");
mgplot_xlabel(1,"X-label");
mgplot_ylabel(1,"Y-label");
mgplot_range(1,-2*PI,2*PI);
mgplot_grid(1,1);
mgplot(1,x,f,{"sin(2.5x)cos(x)"});
pause
```

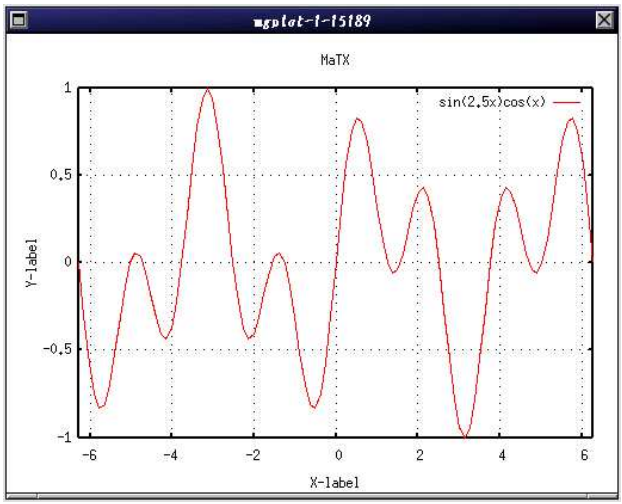

図 2　MaTX による２次元プロット

MaTX でも `mgplot_cmd()` により生の Gnuplot コマンドを使うことができます．PARI-GP は，tutorial では結構ページを割いて，例えば分割点を制御して $sin(x^7)$ などのプロットを正確に描く方法などを説明しています．しかしながら，見栄えの良いプロットを得ることができません．２次元プロット `ploth()` では，タイトルや軸名も描けません（図 3）

```
ploth(x=-2*Pi,2*Pi,sin(2.5*x)*cos(x));
system("sleep 10");
```

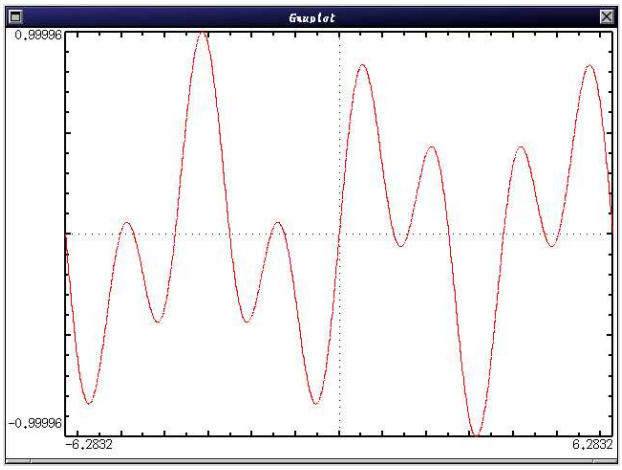

図 3　PARI-GP の ploth による２次元簡易プロット

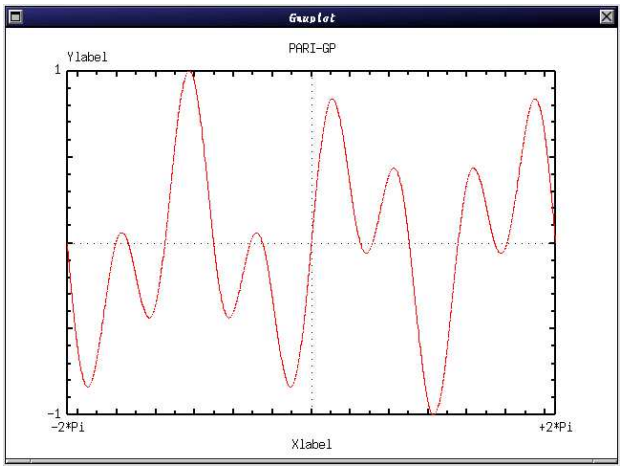

図 4　PARI-GP による２次元プロット

タイトルや軸名などを付けたかったら，次に示すように plotmove() でペンの位置を移動して，plotstring を用いて文字列を書くという低レベルのルーチンを使わなければならないのです（図 4）.

List 3　PARI-GP で低レベルコマンドを使った例

```
A=2; \\accumulator
T=3; \\temporay target
Ts=0.8; \\ scaling factor of T
To=0.1; \\ origin of T in A
relative=1;
\\ 文字揃え
bottom=0; vcenter=4; top=8;
left=0; center=1; right=2;
\\ 微調整
vgap=0.01;
hgap=0.01;

plotinit(A);
plotscale(A, 0,1, 0,1);
plotinit(T,Ts,Ts,relative);
plotrecth(T, x=-2*Pi,2*Pi, sin(2.5*x)*cos(x));
\\ タイトルと軸
plotmove(A, To + Ts/2,  1 - To/2);
plotstring(A, "PARI-GP", center+vcenter);
plotmove(A, To,  1 - To/2);
plotstring(A, "Ylabel", left+top);
plotmove(A, To + Ts/2 ,  To/2);
plotstring(A, "Xlabel", center+top);
\\ x,y の範囲
plotmove(A, To,  To/2);
plotstring(A, "-2*Pi", center+bottom);
plotmove(A, To + Ts,  To/2);
plotstring(A, "+2*Pi", center+bottom);
plotmove(A, To-hgap, To);
plotstring(A, "-1", right+vcenter);
plotmove(A, To-hgap, 1 - To);
plotstring(A, "1", right+vcenter);
\\ ウィンドウへの描画
plotdraw([A,0,0, T,To,To],relative)
\\psdraw([A,0,0, T,To,To],relative);
system("sleep 100")
```

まとめ

行列計算ツールを使いはじめるにあたって見通しを立てるつもりだったのですが，整理できませんでした．緒っから独自路線の PARI-GP は，MATLAB の体系を際だたせるために織り込んでみたのですが，どうも混乱させただけになってしまいました．MaTX は MATLAB 互換でない部分も結構あるし．やはり Scilab や Rlab を調べてみないと，どのようなツールを標準にするのがいいか判断できません．次回もこのネタになりそうです．

参考文献

[1] MathWorks 社の MATLAB Central. W3
http://www.mathworks.com/matlabcentral/

[2] GNU Octave の公式サイト. W3
http://www.octave.org/

[3] INRIA の Scilab 公開サイト. W3
http://www-rocq.inria.fr/scilab/

[4] SourceForge の Rlab プロジェクト. W3
http://rlab.sourceforge.net/

[5] MatX の公開サイト. W3
http://www.matx.org/

[6] PARI-GP の公式サイト. W3
http://www.parigp-home.de/

[7] 大石先生のページ. W3
http://www.oishi.info.waseda.ac.jp/ oishi/FAQ /FAQ.html