X.2 MATLAB to GAUSS の基礎　　ver. 0.2

GAUSS と MATLAB はしばしば補完的に扱われることが多いのですが、同じ行列を扱う言語という点では共通しているものの、基本的な行列に関する文法はまったく違います。一言で言えば、MATLAB は理工系の言語であって、GAUSS は計量データ専用の言語であるという目的の違いから始まっています。一般的に Matlab のファイル拡張子は.m です。

画面表示

MATLAB	GAUSS
disp(a)	print a;
disp('OK')	print "OK";
error(' Variable must be real.')	errorlog "Variable must be real."

Matlab は、表示に関して言えば、丸括弧系の言語です。一方 GAUSS は直接続ける系統の言語です。変数 a の中味を表示するのではなくて、OK という文字列を表示するのには、Matlab では丸括弧の中に一重の引用符' 'で文字列を包みます。一方、GAUSS では二重の引用符" "で包みます。エラーログで abort する場合も同様な違いがあります。

コメント行

MATLAB	GAUSS
% This is a comment.	/* This is a comment. */
k=1; % This is a comment.	k=1; @ This is a comment. @

Matlab では%のマーク以降のその行はコメントとして無視されます。一方、GAUSS では、行の概念はなく/* から*/まで（または@から@まで）の部分がコメントの部分です。

行列表記

MATLAB	GAUSS
a=[1 2 3]	a={1 2 3};

GAUSS では、[ ]の四角括弧は常に配列ディメンションを表すのに用います。したがって、Matlab の[ ]の四角括弧の中味の数字は、GAUSS では{ }の括弧に包む必要があります。上のどちらの表記も括弧内で区切りのしるしがなければ１×３の行ベクトルを表します。GAUSS では、a={1,2,3};とするとカンマごとに改行の意味で、３×１の列ベクトルになります。

行列の行

MATLAB	GAUSS
b=[1 2 3; 4 5 6; 7 8 9];	b={1 2 3, 4 5 6, 7 8 9};
c=[1 2 3 4 5 6 7 8 9];	

Matlab ではセミコロン；を行列データの改行を表して、そこで１行という意味をもちますが、GAUSS ではカンマ,が常に「そこまで１行」という意味を持ちます。Matllab では３行またがって区切りのしるしなしに表記してもそれぞれが行をなします。上の例では b も c も等しく３×３の行列を表現しています。GAUSS では基本的にプログラムに行の概念がないので、カンマで区切らずに Matlab の例のように３行にまたがって行列を表記しようとすれば、１×９の行ベクトルにしかなりません。

配列表記

MATLAB	GAUSS
a(1)	a[1]
b(2,3)	b[2,3]

Matlab では配列を表すのに丸括弧( )を使いますが、GAUSS では一貫して、丸括弧( )は組込み関数および proc の引数に、四角括弧[ ]は配列に使われるように厳格に区別されます。GAUSS では通常 a[1]とは a が列ベクトルである場合の第１要素の意味ですが、行ベクトルの場合の第１要素でもプログラム的には有効です。

サブ行列

MATLAB	GAUSS
b(1:2,2:3)	b[1:2,2:3]

同様に Matlab では行列の一部分を扱うのには、丸括弧で１行目「から」２行目まで、２列目「から」３列目までという意味で、コロン：を使いますが、GAUSS でも同様にコロン：を使います。ただし GAUSS では配列は常に四角括弧[ ]です。GAUSS では、丸括弧( )には引数、四角括弧[ ]には配列、{　} 括弧には行列の実際の要素が常に入ります。

行列のワイルドカード

MATLAB	GAUSS
b(:,2)	b[.,2]

Matlab ではワイルドカードを表すのにコロン：を用いますが、GAUSS ではドットマーク.が一貫してワイルドカードを表します。GAUSS ではコロン：には「何々から何々まで」の意味しかありません。上の場合どちらも行列ｂの２列目の「すべて」の要素をさします。

行列のマージ

MATLAB	GAUSS
[b c]	b~c
[b;c]	b|c

Matlab ではマージはすべて四角括弧[ ]の中で行なわれます。何もつけずに複数の行列を四角括弧の中に入れると水平方向のマージになります。カンマをつける場合もあります。Matlab での行列の改行を意味するセミコロン；のしるしを置くと、（改行されて）上下の垂直方向のマージになります。GAUSS では水平方向のマージは～、垂直方向のマージは｜です。（Matlab では｜は or の意味、～は NOT の意味のようにまったく別の事柄に使われますので注意が必要です。）

組込み関数

MATLAB	GAUSS
m=mean(x);	m=meanc(x);
s=std(x);	s=stdc(x);
sm=sum(x);	sm=sumc(x);
cs=cumsum(x);	cs=cumsumc(x);

Matlab と GAUSS は同じような組込み関数が存在すれば、使い方はほとんど同じです。ただし、Matlab ではそのものの意味を表す言葉が関数名になっているのに対して、GAUSS では「その名前＋c」というふうに列 column ごとの計算であることを明記して組込み関数名が作られていることがほとんどです。上の例では、ｘの平均、標準偏差、合計、累積和をそれぞれ計算し、m,s,sm,cs というふうに自分でつけた名前の変数に代入しています。

組込み関数（違いがあるもの）

MATLAB	GAUSS
n=length(x);	n=rows(x);　k=cols(x);
x1=log(x);	x1=ln(x);
x2=log10(x);	x2=log(x);

上の例では、Matlab と GAUSS の組込み関数に若干の違いがあるところです。ここでも数学系とデータを扱う計量系の言語の違いが見られます。行数を見るときには、Matlab では length と使いますが､GAUSS では rows を使いますし列数を見るのに cols という関数も用意されています。ログに関しては、Matlab と GAUSS はいい方が違いますので注意が必要です。Matlab に精通した国内の研究者のように、Matlab は自然対数に log を使います。常用対数には log10 が別に用意されています。GAUSS では厳密に自然対数 ln の意味には ln という関数が、常用対数 log には log という関数が文字通り割り当てられています。計量の世界では、自然対数と常用対数はまったく違います。Log-Likelihhood などで登場するログは exponential に事前対数をとったときの前に来る部分とそうでない部分とに計算上関連しているので、必ず GAUSS では自然対数の ln を使わなくてはいけません。誤解のないようにしてください。　そのほか、文字列の処理の仕方が matlab ではすべて組込み関数の引数として処理されますが、GAUSS では演算子に＄のマークをつけることによって、直接に足し合わせたり比べたりできます。

シークエンス数列

MATLAB	GAUSS
i=1:10	i=seqa(1,1,10)'
j=1:0.1:2	j=seqa(1,0.1,11)'

Matlab では数列を表すのに始まりと終わりの数を指定します。始め「から」終わりまでという意味でコロン：を用います。上の i の場合には i=[1 2 3 4 5 6 7 8 9 10]ということを表しています。この場合のステップは 1 に固定されています。なお、上の j のようにコロンとコロンの間にさらにステップ幅を指定すると、その数刻みでシークエンスになります。上の j の場合 j=[1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2]ということを表しています。一方、GAUSS ではそのような記号言語的表記方法はありません。その代わり、組込み関数 seqa を用いて作り出します。その場合には、始めと終わりの指定ではなくて、「始め」と「ステップ」と「個数」の指定が必要になります。つまり seqa(1,1,10)であれば、1 から 1 ステップで 10 個分ということになります。ただし、GAUSS は計量データを念頭に専門に作られた言語体系なので、出てくるシークエンスはデータと同じような縦方向の列ベクトルになります。したがって、数学のように行ベクトルがほしいのであれば、転置させてダッシュ´をつける必要があります。なお、同様に 1 から 0.1 ステップで j のようなシークエンスを GAUSS で作る場合、距離とポール数の問題でててくるように、その個数は 10 個ではなくて 11 個になることに注意してください。1 から 0.1 ステップで 10 個分しか作らなければ、結果は当然のことながら、1 から始まって 1.9 で終わってしまいます。

関数の作り方

MATLAB	GAUSS
function [b,se,t]=ols(y,x)	proc(3)=ols(y,x); local b,e,s2hat,varb,se,t;
. . . . .	. . . . .
	retp(b,se,t); endp;

Matlab では、GAUSS のように proc で関数を作るようなとき、function 機能を使います。この場合、retp( );や endp;に相当する部分はありません。GAUSS で retp の丸括弧の中に

入るリターン変数は、Matlab では冒頭で四角括弧[ ]の中に書いてやります。GAUSS では C 言語の { } のクローズに代わるものとして endp;が必ず必要になります。Matlab では特に必要のない変数のローカル宣言が、GAUSS では必要になります。ローカル宣言ではリターン変数のほか、インプット変数を除くその procedure 内で使うすべての変数を宣言しなくてはなりません。

例(サンプルの平均と標準偏差の計算)

MATLAB	GAUSS
function [m,sd] = stat(x)	proc(2)=stat(x);
	local n,m,sd;
n=length(x);	n=rows(x);
m=sum(x)/n;	m=sumc(x)/n;
sd=sqrt(sum((x-m).^2)/n);	sd=sqrt(sumc(x-m)^2)/n);
	retp(m,sd);
	endp;

Matlab の場合、往々にしてデータｘはリスト（または行ベクトル）で与えられていることが多いので、GAUSS の上で考える場合には、まずそれが列なのか行なのかをはっきりとさせる必要があります。上の例では、GAUSS ではいつでもデータは列で与えられているものとしています。これが GAUSS の常道です。例えば、x={1,2,3,4,5};として与えられている場合、GAUSS ではこれを５×１の行列または列ベクトルと呼びます。その点は厳密です。同様に、合計を求めるときには、Matlab では sum を用いて、平均ｍを求めるにはそれをデータの個数ｎで割ります。GAUSS も同様にそう行ないます。しかしながら、sumc とｃの文字を関数につけます。ｍ＝の行をまとめて m=meanc(x);と行なうこともできますが、ここでもｃの文字をつけます。これはｘが複数列の場合（計量経済学の場合ほとんどがそうなっている）その列ごとに GAUSS は計算します。例えば、x={1 9,2 9,3 9,4 9,5 9,6 9};と２列のデータが行列として与えられている時、ｍの計算結果は２×１の列ベクトル{3,9}として出てきます。これこそが GAUSS の本質です。なお、Matlab でついているべき乗^の前のドットは GAUSS では省略することが慣例となっていますが、つけてもかまいません。また、GAUSS では、インプット以外の変数を扱う場合、いかなる場合でも local 宣言を関数内ではしなくてはなりません。また、アウトプットが存在する場合、いかなる場合でも retp( )でリターンを返す変数を明示した上で、冒頭の proc の後の数字にそのリターンの数を一致させなくてはなりません。（１個のリターンの場合には proc stat(x)などと省略が可能です。GAUSS では proc から始まって endp;までが１つのかたまりになります。別関数プログラムファイルを作ることなしに、１つのプログラムファイルに整然と数多くの

関数を書けるように意図されたものが GAUSS です。
条件分岐

MATLAB	GAUSS
if b>1	if b>0;
y=1;	y=1;
else	else;
y=0;	y=0;
end	endif;

Matlab も GAUSS も条件分岐の構造は同じである。しかし、Matlab では条件分岐やループで end というふうに置いて、そこまでがクローズであることを示すのに対して、GAUSS では If クローズに対しては endif;を、do ループに対しては endo;をというふうに専門のそこまでというしるしがある。また、重大な違いには、Matlab では関数や方程式の後にしかセミコロン；をつけないが、GAUSS では制御文字や条件文の後にもいちいちセミコロン；をつける必要があります。単独の else の後や複数分岐の elseif 文も同様です。

Do ループ

MATLAB	GAUSS
k=1;	k=1;
while k<=10	do while k<=10;
. . . . .	. . . . .
k=k+1;	k=k+1;
end	endo;

Matlab では条件文と同様に、end で終わる。制御文にはセミコロンはつけない。一方、GAUSS では制御文を含むすべての文にセミコロン；をつける。また do を while の前につけるとともに、do の end という意味で endo;で終わる。

For ループ

MATLAB	GAUSS

```
k=0;                                    k=0;
for i=1:10                              for i (1,10,1);
    k=k+i;                                  k=k+i;
end                                     endfor;
```

Matlab では i にシークエンスの１から 10 が来るとします。ステップ数１は省略できます。For ループも最後は end で終わります。一方、GAUSS では for ループのシークエンスの指定はイコールサインをつけずに for の直後に置いて、丸括弧の中に(始点,終点,ステップ)の順に指定します。順番が Matlab と異なるとともに、ステップはたとえ１であっても省略はできません。終わりは、GAUSS では、for の end ということで endofor;になります。

プロット

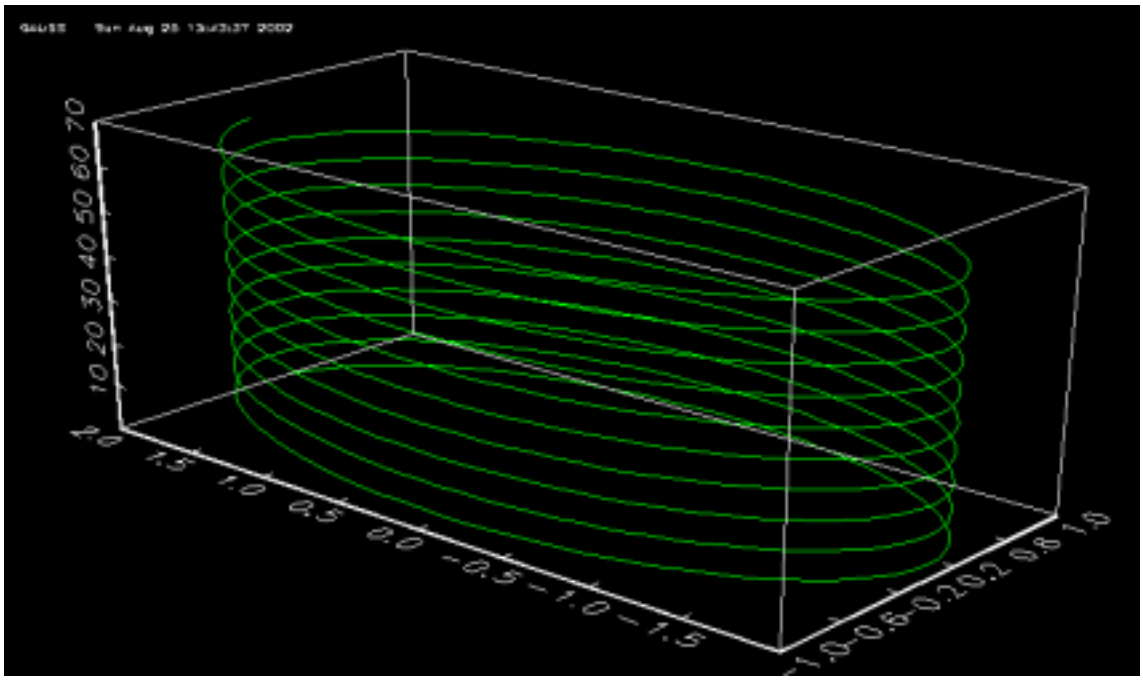
MATLAB	GAUSS
	library pgraph; graphset;
title('xy plot'),xtitle('x')	title("xy plot"); xlabel("x");
plot(x,y)	xy(x,y);

Matlab には plot 関数や plot3 関数があります。GAUSS にもかつて plot という内部関数がありましたが、今は近代言語と同じように、ライブラリーで呼び出す外部式のものに改められました。したがって、GAUSS では library の呼出しで、pgraph を呼んできて、そのグローバル変数を初期化するため graphset;としなくてはなりません Matlab では文字列はシングルの引用符で包みますが、GAUSS ではダブルの引用符で包んでタイトルや軸のラベルを指定します。

例(ナビオ阪急状の螺旋３次元プロット)

MATLAB	GAUSS
	library pgraph; graphset;
t=0:50/pi:20*pi	t=seqa(0,50/pi,1000);
plot3(sin(t),2*cos(t),t)	xyz(sin(t),2*cos(t),t);

Matlab の plot3 は、GAUSS では xyz によって実現します。GAUSS では pgraph のライブラリを呼び出してそのグローバル変数を初期化してからグラフを描く必要があるほか、シークエンスの第３要素は、Matlab では終点の値ですが、GAUSS では軌跡の数です。



なお、GAUSS には Matlab の ezplot の機能はありません。二次元ならば xy(x,y);、三次元ならば xyz(x,y,z);に対して、シークエンス等で全ての軌跡を与えてからグラフを描かせる必要があります。数学ではなくて、データにもとづいた計量のソフトであることがここからもわかります。

微分積分

MATLAB	GAUSS
syms x	
f=x^3	fn f(x)=x^3;
diff(f)	print gradp(&f,2);
int(f)	
int(f,0,2)	x1={0,2}; intquad1(&f,x1);

Matlab であるならば、シンボリックに微分や積分を解くことができます。しかしながら、計量を念頭に置いている GAUSS は数値微分や数値定積分しかできません。GAUSS では proc またはその簡易形である fn を用いて関数を定義してやってから、その関数をポインタを表す&のマークとともに gradp や intquad1 の中で呼び出して、微分の場合にはその地点の値（この場合は２）を与えて計算させます。また、積分の場合には始点と終点の入った２×１のベクトル変数（この場合は x1）を設定して計算させます。Matlab では有理数で答

えがでてきますが、GAUSS の場合数値的に計算しているので答えは小数で出てきます。GAUSS のできないことの１つにあげられるのが微分積分、それに数式の展開ですが、このことは計量経済学自体が数値的に計算解析する学問体系であることに加えて、GAUSS では数値的に求める数多くのアルゴリズムが考案されているので、統計計量経済学の用途にはこのことは弱点とはなりません。

最後に、Matlab は特に宣言をしないならば大文字と小文字の区別はあります。それに対して、GAUSS では大文字と小文字の区別の概念はまったくないので、勘違いのないようにしなければなりません。通常 GAUSS のプログラムは、すべて小文字で書かれることが慣例になっています。また、繰り返しになりますが、GAUSS では列単位に関数の計算が行なわれて、その結果が列として出てくる点に注意しなければいけません。これは、その後もその列ででてきた答えを用いてプログラムを拡張させていくために、わざわざ転置させる必要をなくしているためです。その点、GAUSS は極めて計量経済学を念頭に置いた言語と言えます。それでも GAUSS を使うのには１つ致命的であると言われ続けてきた配列が２次元までしか扱えなかった仕様問題は、GAUSS ver.5.0 からはN次元までの扱えるようになり解決しました。