4.7 特殊関数とライブラリ GPE2 と GAUSS の対応関係　ver. 0.2

以前にも繰り返しことわっているように、GPE2はサードパーティーのプログラム（こういうものはGAUSSXをはじめ数多くある）で、そのソースがコンパイルされて読めなくなっているProf.Linのアドオンパッケージ（機能を加えるもの）である。一部の読者は、彼の本を手に入れてプログラムをしている人もいると思われるので、ここでGAUSS本体およびMAXLINK／CMLライブラリとこのGPE２パッケージとの違いについて述べます。

GPE2もライブラリーと同じようにしてGAUSSのプログラム上で動きます。しかしながら、その根幹のところで、GAUSSの文法や基本構造とはまったく別の操作を必要とするので細心の注意が必要です。

GAUSS	GPE2
プログラムの冒頭は new;	プログラムの冒頭は 必ずuse gpe2; new;どこにも書いてはいけない

GAUSSを学んだことのなる人なら奇異に感じるかもしれませんが、new;文はGPE2を呼び出したGAUSSのプログラムでは使用してはいけません。結果が全く表示されなくなってしまいます。また、通常ライブラリを呼び出すときには、new; cls;の後にライブラリを呼び出せますが、GPE2を呼び出すのは、必ず１行目の一番最初でなければなりません。Prof.Linの本に、new;文がないのはそういうわけである。（おそらく、new;文がuse gpe2;とする際に実行されるものと思われるし、そう希望します。）なお、Output Window上の前回までの古い表示を消してから新しく表示するには、GAUSSのプログラム構造と同様に、cls;が使えます。また、GPE2では、推定や最大最小を求める場合には、すべてestimate文を用います。OLSもQNewtonもMAXLINKもCMLもすべてestimate文で一括して扱うことになります。GPE2とは、いわばマルチ関数estimateを扱うためのパッケージであり、それに各種専用グローバル変数を与えて、いろいろな推定をするものです。

GAUSS	GPE2
それぞれのリセット文; call それぞれの関数名; (例) maxset; call maxprt( );	call reset; call estimate( );

GAUSSでもリセット文は、それぞれのライブラリにあって、call文で書いても、直接書い

ても構いませんでしたから、その点はGAUSSでもGPE２でも同じ構造になっています。なお、GPE２においても、callなしのreset;で同じ動作をするものと思われます。

１）_nlopt=0（デフォルト）のケース

さて、ここで基本的な線形の最小２乗法のregressionを行なって違いを見てみます。基本的には、下のような関係になっています。GPE２の方には、estimate文の前にcall reset;がさらに必要になります。

GAUSS	GPE2
call ols(0,y,x);	call estimate(y,x);
または自作プログラム	

今、Dドライブにdatafile1.txtというｙとｘについてのデータがあるとします。

プログラム（GAUSS本体を用いた場合）

```
new; cls;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];
y=data[.,1]; x=ones(28,1)~data[.,2];
{b,se,t,shat,df,r2}=lse(y,x);
print "y-intercept:" b[1] "          slope for x:" b[2];
print "          se:" se';
print "           t:" t';
print "Std of est :" shat;
print /rz "          df=" df;
print /rz "          R2=" r2;

proc(6)=lse(y,x);
    local b,uhat,n,k,df,s2hat,shat,varb,se,t,r2;
    b=inv(x'x)*x'y;
    uhat=y-x*b;
    n=rows(x);
    k=cols(x);
    df=n-k;
```

```
        s2hat=uhat'uhat/df;
        shat=sqrt(s2hat);
        varb=s2hat*inv(x'x);
        se=diag(sqrt(varb));
        t=b./se;
        r2=1-uhat'uhat/(y'(eye(n)-1/n*ones(n,1)*ones(n,1)')*y);
        retp(b,se,t,shat,df,r2);
    endp;
```

画面表示

```
y-intercept:      77.795          slope for x:      52.010
         se:      22.037                            3.8317
          t:      3.5301                            13.573
Std of est :     52.331
         df=          26
         R2=     0.87633
```

プログラム（GAUSSのOLSコマンドを用いた場合）

```
    new; cls;
    load data[29,2]=d:datafile1.txt;
    data=data[2:29,.];
    y=data[.,1]; x=ones(28,1)~data[.,2];      /*   Or x=data[.,2];   */
    call ols(0,y,x);
```

画面表示

```
Valid cases:                28    Dependent variable:                Y
Missing cases:               0    Deletion method:                 None
Total SS:           575746.679    Degrees of freedom:                26
R-squared:               0.876    Rbar-squared:                   0.872
Residual SS:         71201.875    Std error of est:              52.331
F(1,26):               184.239    Probability of F:              0.000

                        Standard                 Prob   Standardized  Cor with
Variable     Estimate     Error       t-value     >|t|     Estimate    Dep Var
------------------------------------------------------------------------------
CONSTANT    77.795198   22.037400    3.530144    0.002        ---         ---
X1          52.009829    3.831727   13.573468    0.000    0.936126    0.936126
```

プログラム（GPE2を用いた場合）

```
use gpe2;
cls;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];
y=data[.,1]; x=data[.,2];
call reset;
call estimate(y,x);
```

画面表示

```
Least Squares Estimation
------------------------
Dependent Variable = Y
Estimation Range =  1          28
Number of Observations = 28
Mean of Dependent Variable = 345.11
Standard Error of Dependent Variable = 146.03

R-Square = 0.87633      R-Square Adjusted = 0.87157
Standard Error of the Estimate = 52.331
Log-Likelihood = -149.51
Log Ammemiya Prediction Criterion (APC) =  7.9842
Log Akaike Information Criterion (AIC) =  7.9839
Log Schwarz Bayesian Information Criterion (BIC) =  8.0791

Sum of Squares         SS           DF          MSS            F       Prob>F
Explained     5.0454E+005            1  5.0454E+005       184.24  2.6071E-013
Residual            71202.           26       2738.5
Total         5.7575E+005           27       21324.

Variable        Estimated     Standard      t-Ratio         Prob      Partial
Name          Coefficient        Error        26 DF         >|t|   Regression
X1                 52.010       3.8317       13.573  2.6071E-013      0.87633
CONSTANT           77.795       22.037       3.5301    0.0015710      0.32401
```

３つのOLSを行なうプログラムを結果とともに上に示しました。一番最初のプログラムは

再々掲になりますが、通常GAUSSでプログラムをやる際に最低限要求されるものです。通常、学部上級や大学院において、GAUSSでプログラムせよという場合には、コマンドを使わずに行列方式の計算を自分で組み立てていきます。必要な統計量の計算は自分で書いて組み入れます。また、コンスタントのところが必要な場合には、自分ですべての要素が１の列ベクトルを独立変数側にマージする必要があります。２番目のプログラムは、コマンドを用いたGAUSSの使い方ということで今まであえてとりあげなかったのですが、GPE２との比較ということで、ここに掲載してみました。文法は非常に簡単で、ols(0,y,x)をcall文で呼び出します。その第１要素の０が入っているところには、データセットが入るのですが、通常の使用では０または””としてnull stringをいれて無効にします。第２要素には従属変数のベクトルが、第３要素には独立変数のベクトルまたは行列はきます。なお、OLSを用いる場合には、コンスタントの１ばかりの列ベクトルを作成して独立変数側にマージしてもしなくても、OLS関数が自動判断してコンスタントをつけた計算をしてくれます。正式の文法は、

{ vnam,m,b,stb,vc,stderr,sigma,cx,rsq,resid,dwstat } =ols(dataset,depvar,indvars);

というように11個のリターンを返す関数になっています。なおコンスタント項がないOLSをするのには、GAUSSのOLSコマンドでは、ダブルの下線のグローバル変数

__CON = 0;　　（GAUSS OLSコマンドの場合）

としてやります。その次に３番目の、GPE2のプログラムでは、独立変数には決して１のコンスタントの列ベクトルをマージしてはいけません。Perfect Colinearityのエラーを出して止まってしまいます。なお、コンスタント項がないOLSをするのには、GPE２では、

_const=0;　　（GPE2の場合）

の行を加えます。こちらは、シングルの下線です。また、変数ラベルを加える場合には、

_names={"Dep","Indep"};　　（GPE2の場合）

という具合に、従属変数名、独立変数名の順で引用符に包んでラベル名を設定します。２個以上独立変数がある場合は、さらにラベル名を書き加えていきます。なお、コンスタント項は、最初からCONSTANTとされているので、このラベル名の指定には加えません。前述したように、GPE2を用いるときには、call reset;の文が必要で、これらのグローバル変数を指定する前に置きます。このように、リニアの単純なケースでも、GAUSSの構造と

GPE2の構造は似て非なるというよりも、区別するために故意に根幹のところで変更が加えられているので注意が必要です。とは言うものの、GPE2はGAUSSの上に乗っているパッケージであってGAUSSのプログラムについてこれまで習得したすべてのことができます。最終局面でestimate文を実行するために、通常のGAUSS部分と区別するために、文法に大幅な変更が加えられているのです。

上のケースは、リニアのケースでしたが、estimate文はそれ１つで、こればかりでなく最大最小を求めるケースからノンリニアの推定のケースまで幅広く使えるようになっています。それぞれのケースにはそれぞれのケースのグローバル変数を設定することになります。その値によって、estimate関数の内部で場合分けがなされて、何も設定されない場合はリニアのOLS、そうでない場合には、最大最小を求めたり、ノンリニアのケースになったりするわけです。

次に、GAUSSのQNewtonで扱った最小値（または極小値）をもとめるプログラムとGPE２のプログラムを比較します。グローバルな極小値をもたない最適値問題(2)で扱ったと同じ関数$h(x_1,x_2)=x_1^3+x_2^3-9x_1x_2+27$を用いて極小値を求めてみます。QNewtonで扱ったのと同じプログラムを再掲します。

プログラム（GAUSSのQNewtonのケース）

```
new; cls;
qnewtonset;
fn h(x)=x[1]^3+x[2]^3-9*x[1]*x[2]+27;
x0={1,1};
call QNewton(&h,x0);
```

画面表示

```
return code =    0
normal convergence

Value of objective function          0.000000

Parameters    Estimates    Gradient
----------------------------------------
P01               3.0000         0.0000
P02               3.0000         0.0000

Number of iterations          9
Minutes to convergence      0.00467
```

プログラム（GPE２のケース）

```
use gpe2;
cls;
call reset;
fn h(data,x)=x[1]^3+x[2]^3-9*x[1]*x[2]+27;
_nlopt=0;
_b={1,1};
_method=6;          /* Or 3 or 5 */
_iter=100;
call estimate(&h,0);
```

上のように、まず１番目のQNewtonの場合には、１値ベクトルの関数を立ててから、それをQNewtonでポインタのしるし＆をつけて、スタートベクトルとともに呼び出しました。そのグローバル変数をリセットするのには、qnewtonset;を用います。２番目のGPE2のケースでは、まったく異なる文法体系になります。MAXLIKで呼び出す関数などと同じように、dataも含んだ２値関数を立てなくてはなりません。しかしながら、関数の単純な最小（極小）の場合には、全体のdataそのものが存在しませんから、一応からのままdataとしてやります。MAXLIKで呼び出される関数は、ll(b,data)のように、パラメータベクトル、そして全体のもとになるデータ行列の順でしたが、GPE２の場合には、この左右を逆にします。これは後で取り扱うノンリニアのMax Log-Likelihoodを求めるケースでも左右を逆にしなければなりません。（車の非関税障壁のハンドルの左右付け替えのようなことがGPE２パッケージではみられます。したがって、GPE2を初心者に学習させるのは将来GAUSSでプログラムする人にはふさわしくないと思われます。大きな混乱が生じます。）GPE２では、このような左右逆にした２値関数を、関数の最小（極小）を求める際にも使います。call reset;の後、_nlopt=0;によってどのようなOptimizationを行なうかを指定します。デフォルト０の場合は、最小化（極小化）問題を解きます。これが１の場合には、最大化（極大化）問題を解きます。２の場合には、MAXLIK/CMLと同じようなベクトル形式になったLog-Likelihoodの最大化を行ないます。ですから、_nlopt=0;はこの場合任意となります。スタートベクトルはグローバル変数_bとして必ず設定しなくてはなりません。直接変数として設定してはいけません。_nlopt=0;の場合の最小化極小化の場合にはこれをとりおこなうMethodはSteep-ascent/decentなのですが、上の問題はこれでは解けなくて、Methodを設定するグローバル変数_methodで６としています。なお、上の問題の場合には、３または５でも計算できます。ここで注意すべきことは、iterationの数の設定です。QNewton MAXLIK/CMLではデフォルトが決まっていて設定する必要がありませんでしたが、GPE2では必ず設定することが必要になります。上のプログラムでは、_iterというグローバル変数に100を代入することによって、最高100回のiterationをする設定にしています。そうでなければ、iterationは１回で止まります。最後に、call estimate(&h,0);でその第１

要素でポインタのしるしの＆をつけて関数を呼び出し、第２要素には、この場合データは空ですから０を代入して空であることをしめしています。この第２要素はスタートベクトルではありませんので注意が必要です。なお、ここに空の０ではなくて、全体のdataがくると、もし定義されている関数がベクトル形式のresidual functionであれば、SSE最小化を行なうことになります。というわけで、ここの２番目の０はデータが空であるという意味です。第１番目のポインタがついた関数ではなくて従属変数名がくるとリニアのOLSを自動的に判断してやることになるのは、一番最初にやったとおりです。Methodオプションをまとめておくと、

_method=0（デフォルト）	Steep-ascent/decent
1	BFGS法
2	DEP法
3	Greenstad法
4	ニュートン＝ラプソン法
5	Quadratic Hill-Climbing法
6	修正Quadratic Hill-Climbing

なお、デフォルトは、ノンリニアLSの場合にはガウス＝ニュートン法が、ML推定の場合には、BHHHがデフォルトになり変化します。

GAUSS	GPE2
fn h(b)=… ; １値関数	fn h(data,b)=… ; ２値関数
b={ };	_b={ }; ←グローバルとして
	_iter=100;←必須
call QNewton(&h,b);	call estimate(&h,0);

次にこれと同じ最小値を求める計算を利用してSSEを最小にするパラメータを求める計算を引き続きdatafile1.txtを用いて行ないます。今度はdataがある場合についてです。

プログラム（GAUSSのQNewtonのケース）

```
new; cls;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];

b={1,1};
```

```
    qnewtonset;
    _qn_ParNames={"CONST","X"};
    call QNewton(&sse,b);

    proc sse(b);
        local y,x;
        y=data[.,1];
        x=ones(28,1)~data[.,2];
        retp((y-x*b)'(y-x*b));
    endp;
```

画面表示

```
return code =    0
normal convergence

Value of objective function    71201.875467

Parameters    Estimates    Gradient
----------------------------------------
CONST           77.7952       -0.0016
X               52.0098       -0.0079

Number of iterations       18
Minutes to convergence     0.00083
```

プログラム（GPE2のケース）

```
    use gpe2;
    cls;
    load data[29,2]=d:datafile1.txt;
    data=data[2:29,.];

    call reset;
    _b={1,1};
    _method=6;
    _iter=100;
    _names={"CONST","X"};
    call estimate(&sse,data);
```

```
    proc sse(data,b);
        local y,x;
        y=data[.,1];
        x=ones(28,1)~data[.,2];
        retp(y-x*b);                    /* (y-x*b)'(y-x*b) is also allowed. */
    endp;
```

画面表示

```
Final Result:
Iterations = 66          Evaluations = 39900
Sum of Squares =       71202.
Parameters =       77.795       52.010
Gradient Vector = -5.6919e-005 -7.1731e-005

                          Asymptotic   Asymptotic
               Parameter   Std. Error      t-Ratio
CONST             77.795       21.236       3.6634
X                 52.010       3.6923       14.086
```

上のように、まず１番目のQNewtonを用いた方法では、１値関数のSSE(b)を作成したうえで、qnewtonset;でグローバル変数をリセットしてから、QNewtonでポインタ&をともなってその関数を、スタートベクトルとともに呼び出して最小化します。パラメータにラベルをつける場合には、グローバル変数_qn_ParNamesを設定します。一方、２番目のGPE2を用いた方法では、SSE(data,b)というふうに１値関数ではなくて２値関数を作成したうえcall reset;でグローバル変数をリセットしてから、グローバル変数_bとしてスタートベクトルを設定し、_namesでラベルを設定します。この際、最大iteration数は必ず設定しないと動きませんので、_iter=100;としています。SSE(data,b)のところは、GPE２の場合、通常のGAUSSの計算と同じようにSSEをe'eを計算することによって用いるほかに、直前に述べたように、eのままのresidual functionをリターンにして、これを呼び出してもGPE2の内部で判断して、SSEを最小にするパラメータを計算してくれます。

２）_nlopt=1のケース

これまでのケースは_nlopt=0のリニアなケースの最小２乗法および最小値をもとめる計算でした。今度の_nlopt=1というのは、Log-Likelihoodを直接最大にするオプションです。ベクトル形ではなくて、すでに足し合わせたLog-Likelihoodそのものを最大化します。MAXLIKであえて扱いませんでしたが、ベクトル形だけでなく、下のようにLog-Likelihood

そのものも最大化も可能です。
プログラム

```
new; cls;
library maxlik;
maxset;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];

start={0,0,1};
_max_Algorithm = 2;
_max_ParNames = {"CONST","X","s"};
call maxprt(maxlik(data,0,&ll,start));

proc ll(b,data);
    local beta, s, y, x, e,n;
    y=data[.,1];
    x=ones(28,1)~data[.,2];
    beta=b[1:2];
    s=b[3];
    e=y-x*beta;
    n=rows(data);
    retp(-1/2*n*ln(2*pi)-1/2*n*ln(s^2)-1/2*e'e/s^2);
endp;
```

画面表示

```
return code =    0
normal convergence

Mean log-likelihood          -5.33947
Number of cases      28

Covariance matrix of the parameters computed by the following method:
Inverse of computed Hessian

Parameters    Estimates     Std. err.  Est./s.e.  Prob.    Gradient
-------------------------------------------------------------------
```

CONST	77.7952	21.2356	3.663	0.0001	-0.0000
X	52.0098	3.6923	14.086	0.0000	-0.0000
s	50.4274	6.7387	7.483	0.0000	0.0000

Correlation matrix of the parameters

1.000	-0.894	-0.000
-0.894	1.000	0.000
-0.000	0.000	1.000

Number of iterations 91
Minutes to convergence 0.11250

プログラム（GPE2のケース）

```
use gpe2;
cls;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];



call reset;
_nlopt=1;
_method=5;
_iter=10000;
_b={0,0,1};
_names = {"CONST","X","s"};
call estimate(&ll,data);



proc ll(data,b);
    local beta, s, y, x, e,n;
    y=data[.,1];
    x=ones(28,1)~data[.,2];
    beta=b[1:2];
    s=b[3];
    e=y-x*beta;
    n=rows(data);
```

```
    retp(-1/2*n*ln(2*pi)-1/2*n*ln(s^2)-1/2*e'e/s^2);
endp;
```

画面表示

```
Final Result:
Iterations = 83          Evaluations = 2227
Function Value =      -149.51
Parameters =       77.795       52.010       50.427
Gradient Vector =     0.00000      0.00000  2.6490e-006

                          Asymptotic   Asymptotic
               Parameter   Std. Error      t-Ratio
CONST             77.795       21.236       3.6634
X                 52.010       3.6923       14.086
S                 50.427       6.7389       7.4831
```

上の２つの違いは、ll関数のインプットの順序が逆になることです。MAXLIKでは、パラメータベクトル、そしてデータ行列の順です。GPE2ではこれが左右逆転します。ll関数は、-1/2*n*ln(2*pi)-1/2*n*ln(s^2)-1/2*e'e/s^2というふうに、すでに足し合わされた形をどちらも使っています。

３）_nlopt=2のケース

最後に、足し合わせる前のベクトル形のLog-Likelihoodを最大にする方法をGAUSSのMAXLIKによる従来の方法とGPE2による方法の２つを示して比較します。

プログラム（GAUSSのMAXLIKを用いたケース）

```
new; cls;
library maxlik;
maxset;

load data[29,2]=d:datafile1.txt;
data=data[2:29,.];
start={0,0,1};
_max_Algorithm = 2;
_max_ParNames = {"CONST","X","s"};
call maxprt(maxlik(data,0,&ll,start));
```

```
    proc ll(b,data);
        local beta, s, y, x, e;
        y=data[.,1];
        x=ones(28,1)~data[.,2];
        beta=b[1:2];
        s=b[3];
        e=y-x*beta;
        retp(-1/2*ln(2*pi)-1/2*ln(s^2)-1/2*e^2/s^2);
    endp;
```

画面表示

```
return code =    0
normal convergence

Mean log-likelihood         -5.33947
Number of cases      28

Covariance matrix of the parameters computed by the following method:
Inverse of computed Hessian

Parameters    Estimates     Std. err.  Est./s.e.  Prob.    Gradient
-------------------------------------------------------------------
CONST           77.7945       21.2356    3.663   0.0001     -0.0000
X               52.0099        3.6923   14.086   0.0000     -0.0000
s               50.4275        6.7388    7.483   0.0000      0.0000

Correlation matrix of the parameters
   1.000  -0.894  -0.000
  -0.894   1.000   0.000
  -0.000   0.000   1.000

Number of iterations    91
Minutes to convergence     0.11800
```

プログラム（GPE2のケース）

```
    use gpe2;
```

```
cls;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];


call reset;
_nlopt=2;
_method=5;
_iter=10000;
_b={0,0,1};
_names = {"CONST","X","s"};
call estimate(&ll,data);


proc ll(data,b);
    local beta, s, y, x, e;
    y=data[.,1];
    x=ones(28,1)~data[.,2];
    beta=b[1:2];
    s=b[3];
    e=y-x*beta;
    retp(-1/2*ln(2*pi)-1/2*ln(s^2)-1/2*e^2/s^2);
endp;
```

画面表示

```
Final Result:
Iterations = 83          Evaluations = 62356
Log Likelihood =      -149.51
Parameters =       77.795       52.010       50.427
Gradient Vector = -2.2834e-009 -1.7077e-009  2.6966e-006

                          Asymptotic   Asymptotic
               Parameter   Std. Error      t-Ratio
CONST             77.795       21.236       3.6634
X                 52.010       3.6923       14.086
S                 50.427       6.7389       7.4831
```

今度は、-1/2*ln(2*pi)-1/2*ln(s^2)-1/2*e^2/s^2という具合に、ll関数を足し合わせる前のベクトルの形でリターンさせたものを呼び出します。そのほかの部分は、オプション２のときと基本的に同じプログラムです。習慣的に、また経験的に、このベクトルの形の方が収束する確率が高いとされており、この方法がLog-Likelihoodを直接最大化するよりも用いられる傾向にあります。そういった意味で、MAXLIKのところでは、直接最大化する方法は示していませんでした。

GAUSS	GPE2
library maxlik;	use gpe2;
maxset;	call reset;
b={ };	_b={ };
	_nlopt=2;または_nlopt=3;
call maxprt(maxlik(data,0,&ll,start));	call estimate(&ll,data);
proc ll(b,data);	proc ll(data,b);

今度は、最大化をする機能ではなくて、リニアの制約を加えて、F分布に関するテストをするオプションを考えます。いま、Dドライブにdatafile10.txtという30×３のデータがあるとします(Judge[1988]Ch.12より)。１列目はL、２列目はK、最終３列目にQとします。これをトランスログ関数として、２次項にすべて０の同時制約をかけてコブダグラス型にできるかどうかテストします。その次に、このトランスログ型関数が成立しているものとして、１次同次を満たすかどうかを係数の線形制約からテストします。モデルは

$$\ln Q = \alpha + \beta_1 \ln L + \beta_2 \ln K + 0.5\beta_3 (\ln L)^2 + \beta_4 (\ln L)(\ln K) + 0.5\beta_5 (\ln K)^2$$

という２次形式までの近似式とします。ここではMulticollinearityを無視して、上の関係を直接このまま推定するものとします。

プログラム

```
use gpe2;
cls;
load data[30,3]=d:datafile10.txt;
l=data[.,1];k=data[.,2];q=data[.,3];
y=ln(q);x=ln(l)~ln(k)~(0.5*(ln(l)^2))~(ln(l).*ln(k))~(0.5*(ln(k)^2));

_restr={0 0 1 0 0 0,
        0 0 0 1 0 0,
        0 0 0 0 1 0};
```

```
call estimate(y,x);

_restr={1 1 0 0 0 1,
        0 0 1 1 0 0,
        0 0 0 1 1 0};
call estimate(y,x);
```

画面表示

```
Least Squares Estimation
------------------------
Dependent Variable = Y
Estimation Range =  1          30
Number of Observations = 30
Mean of Dependent Variable = -1.5327
Standard Error of Dependent Variable = 1.4538

WARNING: Linear Restrictions Imposed.
R-Square, AOV, SE, and t may not be reliable!
Wald F-Test for Linear Restrictions
F(   3,  24)       Prob>F
      12.995  3.0389E-005

R-Square = 0.91833      R-Square Adjusted = 0.91229
Standard Error of the Estimate = 0.43056
Log-Likelihood = -15.708
Log Ammemiya Prediction Criterion (APC) =  -1.5030
Log Akaike Information Criterion (AIC) =  -1.3907
Log Schwarz Bayesian Information Criterion (BIC) =  -1.1105

Sum of Squares         SS           DF          MSS            F       Prob>F
Explained           56.285            2       28.142       151.81  2.0535E-015
Residual            5.0053           27      0.18538
Total               61.290           29       2.1135

Variable        Estimated     Standard      t-Ratio         Prob      Partial
Name          Coefficient        Error        27 DF         >|t|   Regression
X1                0.73581     0.065797       11.183  1.2237E-011      0.82244
```

X2	0.94899	0.062907	15.086	1.1213E-014	0.89394
X3	0.00000	0.00000	0.00000	0.00000	0.00000
X4	2.6645E-015	0.00000	0.00000	0.00000	0.00000
X5	-8.8818E-016	0.00000	0.00000	0.00000	0.00000
CONSTANT	0.42480	0.13781	3.0825	0.0046895	0.26031

Least Squares Estimation

Dependent Variable = Y
Estimation Range = 1 30
Number of Observations = 30
Mean of Dependent Variable = -1.5327
Standard Error of Dependent Variable = 1.4538

WARNING: Linear Restrictions Imposed.
R-Square, AOV, SE, and t may not be reliable!
Wald F-Test for Linear Restrictions

F(3, 24)	Prob>F
2.1255	0.12345

R-Square = 0.96061 R-Square Adjusted = 0.95770
Standard Error of the Estimate = 0.29901
Log-Likelihood = -4.7691
Log Ammemiya Prediction Criterion (APC) = -2.2323
Log Akaike Information Criterion (AIC) = -2.1199
Log Schwarz Bayesian Information Criterion (BIC) = -1.8397

Sum of Squares	SS	DF	MSS	F	Prob>F
Explained	57.304	2	28.652	320.47	1.5469E-019
Residual	2.4139	27	0.089405		
Total	61.290	29	2.1135		

Variable Name	Estimated Coefficient	Standard Error	t-Ratio 27 DF	Prob >|t|	Partial Regression
X1	0.43043	0.029075	14.804	1.7678E-014	0.89032

X2	0.56957	0.029075	19.590	1.7209E-017	0.93427
X3	-0.18945	0.016672	-11.364	8.5432E-012	0.82707
X4	0.18945	0.016672	11.364	8.5432E-012	0.82707
X5	-0.18945	0.016672	-11.364	8.5432E-012	0.82707
CONSTANT	-0.024217	0.062320	-0.38859	0.70063	0.0055615

プログラムの前半部分は、30行３列のデータをdataという行列に読みこんで、１行目をL、２行目をK、３行目をQに割り当てます。従属変数ｙにln(q)を、独立変数に、lnL、lnK、$0.5(\ln L)^2$、lnLlnK、$0.5(\ln K)^2$の５変数と定数項（つけなければ自動的につきます）とします。これを一番最初にやったcall estimateをリニアのケースに使う方法で推定をします。その際に、estimateの第１要素は従属変数で、第２変数は独立変数を水平方向に～のしるしでマージしたものになります。その際に、リニアの制約を加えます。最初の制約は、

　　β3=0　かつ　β4＝0　かつ　β5＝0

ですから、Ｒβ＝ｑ方式で、<u>定数項を除いた形で</u>、

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

ですから

3×5　　5×1　3×1

GAUSSでは、GPE2のグローバル変数_restrのパラメータを、Rとｑを[Ｒ　ｑ]というふうに水平方向にマージした３行６列の行列に設定します。すなわち、ここでは

```
_restr={0 0 1 0 0 0,
        0 0 0 1 0 0,
        0 0 0 0 1 0};
```

となります（定数項の係数は含まれませんので注意を）。F分布はパラメータ制約が３本（すなわち、Rまたはqの行数が３）、N-Kの自由度は30－６＝24になります（６は定数項も含むXの行列の列数に相当）。F(3,24)が約13で十分に大きくP値もほとんど０で、F分布のかなり右のテイルにきますから、

H0：β3=0　かつ　β4＝0　かつ　β5＝0　（すなわちコブダグラス型）
H1：H0 is NOT true.

でH0の帰無仮定は棄却されますから、この場合、二者選択なら、コブダグラス型ではなくてトランスログ型が選択されることになります。同様に、これがトランスログだとした上で、今度は１次同次であるかどうかをテストします。１次同次のこの場合の条件は、

β1＋β2＝1
β3＋β4＝0
β4＋β5＝0

を同時に満たすことですから、Rβ＝q方式で、

$$\begin{bmatrix}1&1&0&0&0\\0&0&1&1&0\\0&0&0&1&1\end{bmatrix}\begin{bmatrix}\beta_1\\\beta_2\\\beta_3\\\beta_4\\\beta_5\end{bmatrix}=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

ですから

3×5　　5×1　3×1

GAUSSでは、Rとqを水平方向にマージして３×６の行列のパラメータ

```
_restr={1 1 0 0 0 1,
        0 0 1 1 0 0,
        0 0 0 1 1 0};
```

と設定します。この場合も制約式は３本で、N－Kは24でかわりませんから、F(3,24)が2.1255で、Ｐ値も0.1を超えていますから、10%に対しても５%に対しても

H0：β1＋β2＝1かつβ3＋β4＝0かつβ4＋β5＝0　（すなわち１次同次）
H1：H0 is NOT true.

で帰無仮説H0は棄却されないので、この場合、１次同次と言えます。要するに、リニアの制約を与えるパラメータ_restrにはRとqの両方をマージした（定数項を含まない）行列を使うということです。なお、GPE2の回帰分析の結果には定数項は一番最後にくるようにデザインされています。また、定数項が自動に付け加えられるのはデフォルト設定です。

- -

GAUSS(3.9章自作プログラム)　　GPE2

- -

定数項を含むRとqを用いる	_restrは定数項を除いた[R q] （水平方向のマージ）

- -

　この他に、さらに諸テストに関するオプション、ラグやARおよびMAに関するオプション､それにForecastに関するオプションなどがグローバル関数で与えられていろいろな計算がプログラムをすることなくできるようになったのがGPE２です。冒頭に述べたように、最初の行の最初にuse gpe2;としなくてはいけないこととnew;としてはいけないのを除くと、ほぼすべてのGAUSSの文法と関数が使えます。また、MAXLIKなしにLog-Likelihoodの最大化ができます。