5.7　モンテカルロシミュレーション ARCH Family(2)　ver.0.1

前章では ARCH、GRACH そして IGARCH の過程のシミュレーションとその推定について扱いましたが、ここではそれらの若干のバリエーションについてプログラムします。

GARCH-M(p,q)過程

GARCH のモデルに、λｈのタームが加わって、Ｙの平均がボラティリティーｈにも依存する形になっている。すなわち、

$$Y = X\beta + \lambda h + u$$

$$h_t = a_0 + a_1 u_{t-1}^2 + ... + a_q u_{t-q}^2 + b_1 h_{t-1} + ... + b_p h_{t-p}$$

$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

というぐあいに、ｈがもとの残差を求める式に入っている。

プログラム

```
new; cls;
a={0.2,0.1,0.1}; b={0.2,0.1,0.1,0.1}; a0=1; cutn=20; n=100; beta={0.5,2};
r=0.1;
{y,x}=garchmsim(a,b,a0,cutn,n,beta,r);
data=y~x;
print data;

proc(2)=garchmsim(a,b,a0,cutn,n,beta,r);
    local s2,q,p,u,h,max,t,x,y;
    if sumc(a)+sumc(b)>=1;
        errorlog "ERROR:Parameter sum must be less than 1.";
        retp(-1);
    endif;
    if cutn<maxc(rows(a)|rows(b));
        errorlog "ERROR:Initial cut must be at least greater than order p or q.";
        retp(-1);
    endif;
    q=rows(a); p=rows(b);
    s2=a0/(1-sumc(a)-sumc(b));
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    max=maxc(q|p);
```

```
    u[1:max]=sqrt(s2)*rndn(max,1); h[1:max]=s2*ones(max,1);
    t=max+1;
    do while t<=cutn+n;
        h[t]=a0+rev(a)'(u[t-q:t-1]^2)+rev(b)'h[t-p:t-1];
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    x=ones(n,1)~(10*rndu(n,1)-5); /* Set a constant and X of [-5,5] here. */
    y=x*beta+r*h+u;                    /* We now have r*h[t] term. */
    retp(y,x);
endp;
```

上のプログラムでは GARCH の一般形をγｈのタームを加えてシミュレーションしたｙとｘの系列を出力するプログラムになっている。この他にも GARCH にはいろいろなバリエーションが存在する。簡単化のため、以下では GARCH(1,1)についてのバリエーションに特化して見ていこう。

GARCH(1,1) with Leverage

ここでは、通常の GARCH では下の丸括弧の中が u_{t-1} だけなのだが、そこに新たにγの係数だけ増された u_{t-1} の Leverage 項が存在する。前の絶対値は、２乗されるのでもともとの u_{t-1}^2 と同じことなのだが、Leverage 項とは独立させるために絶対値がついている。

$$h_t = a_0 + a_1(|u_{t-1}| + \gamma u_{t-1})^2 + b_1 h_{t-1}$$

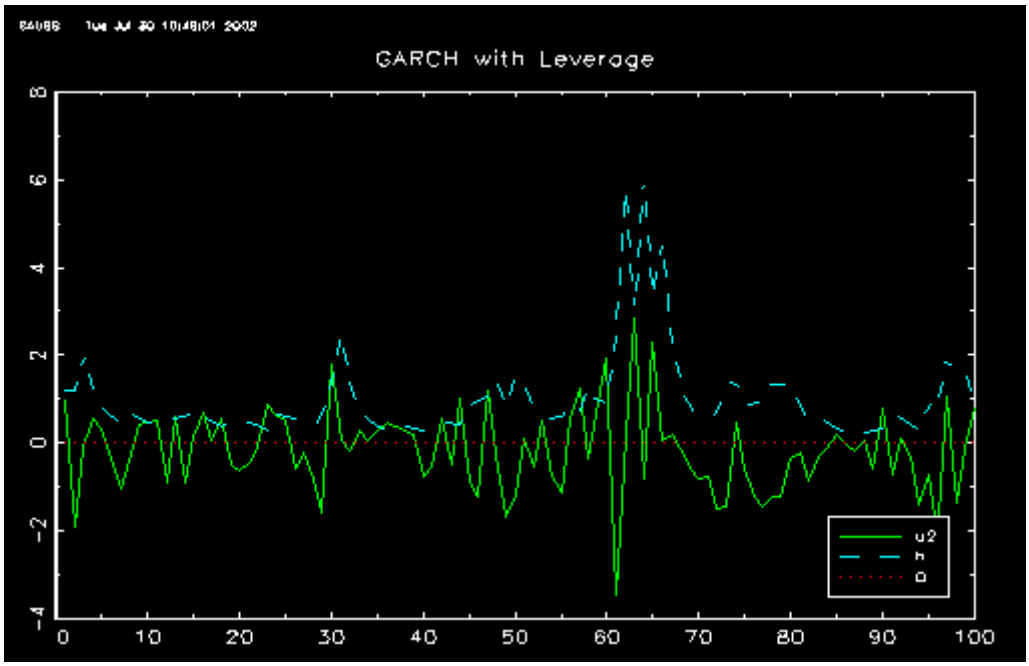
$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

プログラム

```
new; cls;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100; lev=0.1;
{u,h}=garch11leverage(a,b,a0,cutn,n,lev);
library pgraph;
graphset;
title("GARCH with Leverage");
_plegctl=1;
_plegstr="u2¥000h¥0000";
xy(seqa(1,1,n),u~h~zeros(n,1));
```

```
proc(2)=garch11leverage(a1,b1,a0,cutn,n,lev);
    local s2,u,h,t,e;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    e=zeros(cutn+n,1);
    u[1]=sqrt(s2)*rndn(1,1); h[1]=s2;
    t=2;
    do while t<=cutn+n;
        e[t-1]=abs(u[t-1])+lev*u[t-1];
        h[t]=a0+a1*e[t-1]^2+b1*h[t-1];
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    retp(u,h);
endp;
```

グラフ表示



なお Leverage のパラメータの値が０のときには、u のタームの方が Absolute Value の GARCH になる。ここで-M もこの with Leverage も一般の GARCH だけでなく、その変種

の各種のモデルにも適用される。その都度、名前の後に、-M または、with Leverage とつけることになる。

PGARCH(1,1)

下のように２乗の項にさらにｄが power の形でかかっているものを PGARCH と呼ぶ。

$$(h_t)^d = a_0 + a_1(u_{t-1}^2)^d + b_1(h_{t-1})^d$$

$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

プログラム

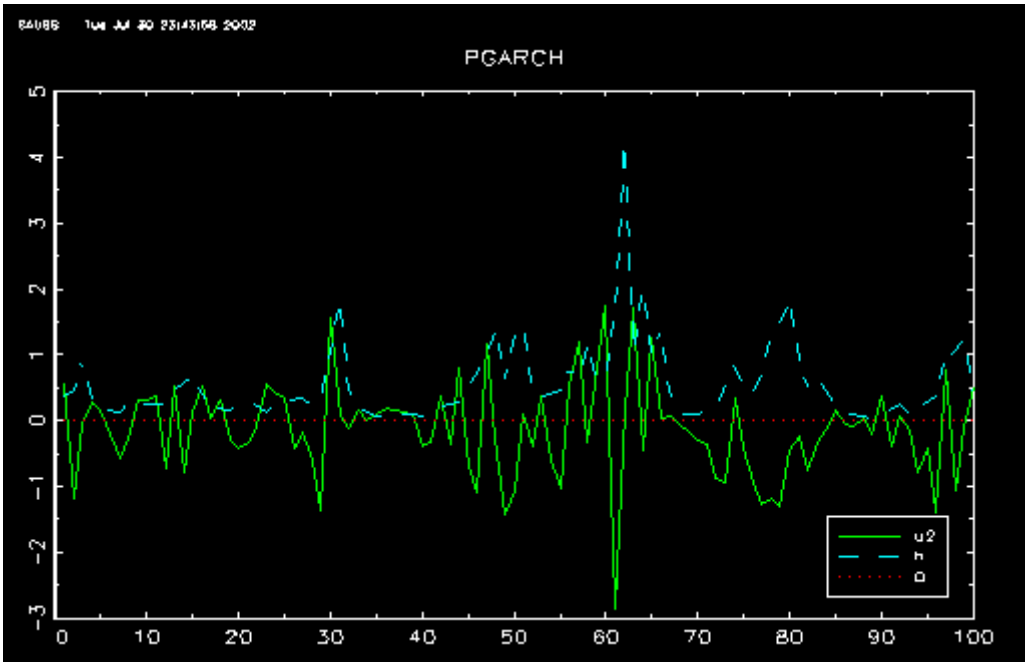
```
new; cls;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100; d=0.3;
{u,h}=pgarch11(a,b,a0,cutn,n,d);
library pgraph;
graphset;
title("PGARCH");
_plegctl=1;
_plegstr="u2¥000h¥0000";
xy(seqa(1,1,n),u~h~zeros(n,1));

proc(2)=pgarch11(a1,b1,a0,cutn,n,r);
    local s2,u,hd,h,t;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    hd=zeros(cutn+n,1);
    u[1]=s2^(1/(2*d))*rndn(1,1); hd[1]=s2;
    t=2;
    do while t<=cutn+n;
        hd[t]=a0+a1*abs(u[t-1]^2)^d+b1*hd[t-1];
        u[t]=hd[t]^(1/(2*d))*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=hd[cutn+1:cutn+n]^(1/d);
    retp(u,h);
```

endp;

プログラムでは、変数ベクトル hd をすでにｈに d の power がかかっているものとして、再帰計算をし、それを 1/2d の power をかけなおしてルートのｈをつくる。これを標準正規乱数にかけている。最後にｈの系列を求めるのに、hd に 1/d の power をかけなおしてｈの系列に変換して u とともにリターンとして返している。

画面表示



NGARCH(1,1)

　ここでは非線形の GARCH を扱う。一般形は BOX-COX 変換のようなλに依存した関数形が各タームにかかるのだが、ここはその簡略形のよく使われる下の形で推定をしよう。

$$h_t = a_0 + a_1 |u_{t-1}|^{\lambda} + b_1 h_{t-1}$$

$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

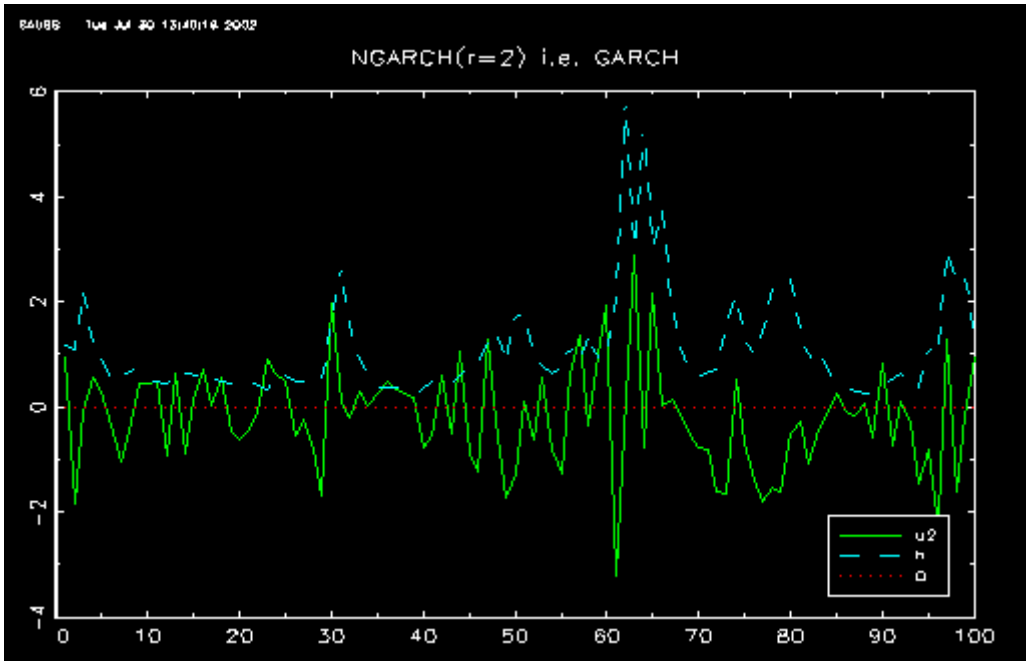
プログラム

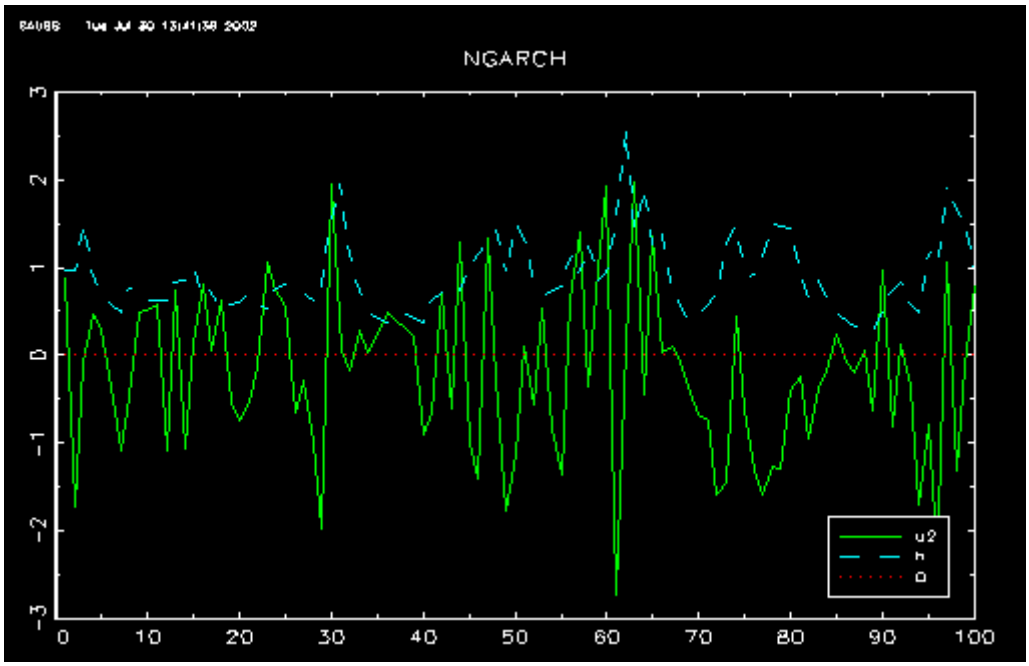
```
new; cls;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100; r=1.3;
{u,h}=ngarch11(a,b,a0,cutn,n,r);
library pgraph;
graphset;
title("NGARCH");
_plegctl=1;
```

```
_plegstr="u2¥000h¥0000";
xy(seqa(1,1,n),u~h~zeros(n,1));

proc(2)=ngarch11(a1,b1,a0,cutn,n,r);
    local s2,u,h,t;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    u[1]=sqrt(s2)*rndn(1,1); h[1]=s2;
    t=2;
    do while t<=cutn+n;
        h[t]=a0+a1*abs(u[t-1])^r+b1*h[t-1];
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    retp(u,h);
endp;
```

グラフ表示





最初のグラフは NGARCH の簡略形の $|u_{t-1}|$ タームにかかる係数λ＝2 のときのもので、これは GARCH そのものである。上のグラフはλ=1.3 の非線形のもの。なお、λ＝1 のときには、Leverage がある時のγ＝0 に相当する Absolute Value の GARCH と同じになる。

AGARCH(1,1)過程

非対称な形の GARCH を考えるため、以下のようなλの項を加えたモデルを考える。

$$h_t = a_0 + a_1(u_{t-1} + \lambda)^2 + b_1 h_{t-1}$$

$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

プログラム

```
new; cls;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100; r=0.1;
{u,h}=agarch11(a,b,a0,cutn,n,r);
library pgraph;
graphset;
title("AGARCH");
_plegctl=1;
_plegstr="u2¥000h¥0000";
xy(seqa(1,1,n),u~h~zeros(n,1));

proc(2)=agarch11(a1,b1,a0,cutn,n,r);
```

```
    local s2,u,h,t;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    u[1]=sqrt(s2)*rndn(1,1); h[1]=s2;
    t=2;
    do while t<=cutn+n;
        h[t]=a0+a1*(u[t-1]+r)^2+b1*h[t-1];
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    retp(u,h);
endp;
```

グラフは GARCH をのばしたような形になるので省略するが、λ＝0 のとき、GARCH そのものになる。この AGARCH は GJR や EGARCH などと同様に非対称モデルの１つである。インパクトカーブを作ると、この AGARCH だけは u_{t-1}＝－λに中心がずれる。

NAGARCH(1,1)過程

今度は、N タイプのものと非対称項の A タイプのものとの組合せモデルを考える。

$$h_t = a_0 + a_1(u_{t-1} + \lambda\sqrt{h_{t-1}})^2 + b_1 h_{t-1}$$

$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

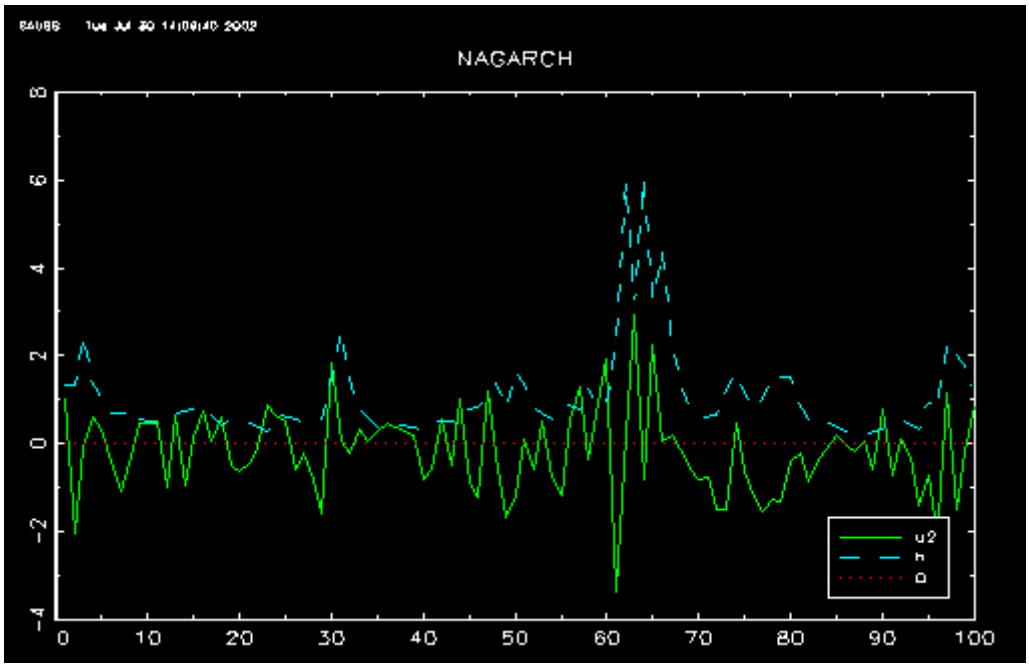
ルートのｈのタームで非線形を表し、λをかけたその項全体で非対称項にしている。

プログラム

```
new; cls;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100; r=0.1;
{u,h}=nagarch11(a,b,a0,cutn,n,r);
library pgraph;
graphset;
title("NAGARCH");
_plegctl=1;
_plegstr="u2¥000h¥0000";
xy(seqa(1,1,n),u~h~zeros(n,1));
```

```
proc(2)=nagarch11(a1,b1,a0,cutn,n,r);
    local s2,u,h,t;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    u[1]=sqrt(s2)*rndn(1,1); h[1]=s2;
    t=2;
    do while t<=cutn+n;
        h[t]=a0+a1*(u[t-1]+r*sqrt(h[t-1]))^2+b1*h[t-1];
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    retp(u,h);
endp;
```

グラフ表示



同様に、上の NAGARCH のときもλ＝０の時に GARCH そのものになる。ここでは AGARCH では定数項がλとしてついていたものに、これが係数となってルートのh_{t-1}の非線形項が加わっている。

VGARCH(1,1)過程

Vタイプの GARCH は以下のような特殊な形をとる。

$$h_t = a_0 + a_1\left(\frac{u_{t-1}}{\sqrt{h_{t-1}}} + \lambda\right)^2 + b_1 h_{t-1}$$

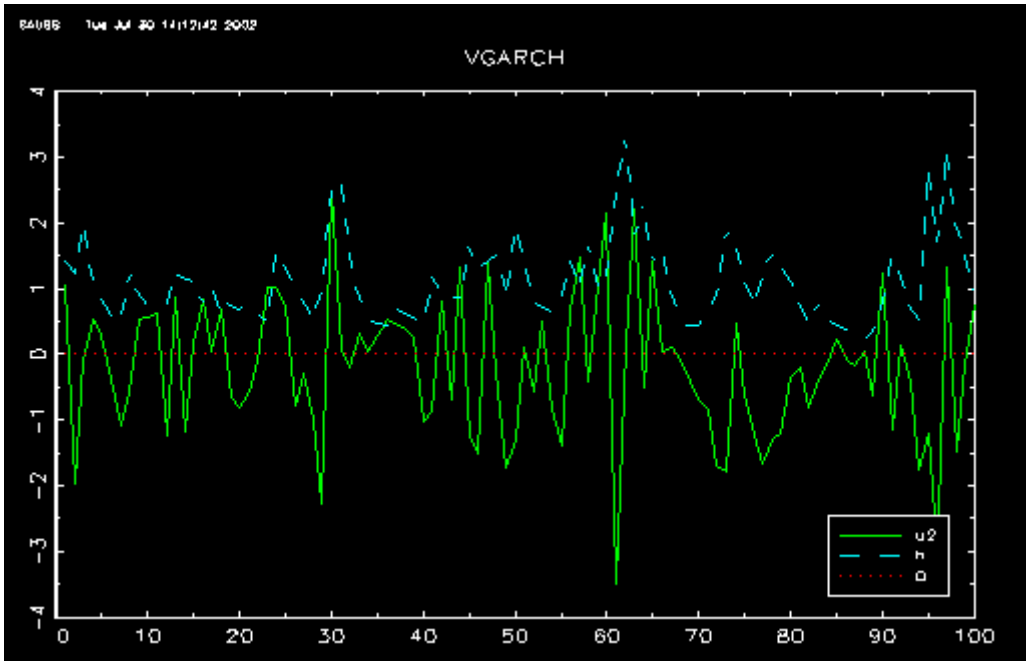
$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

プログラム

```
new; cls;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100; r=0.1;
{u,h}=vgarch11(a,b,a0,cutn,n,r);
library pgraph;
graphset;
title("VGARCH");
_plegctl=1;
_plegstr="u2¥000h¥0000";
xy(seqa(1,1,n),u~h~zeros(n,1));

proc(2)=vgarch11(a1,b1,a0,cutn,n,r);
    local s2,u,h,t;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    u[1]=sqrt(s2)*rndn(1,1); h[1]=s2;
    t=2;
    do while t<=cutn+n;
        h[t]=a0+a1*(u[t-1]/sqrt(h[t-1])+r)^2+b1*h[t-1];
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    retp(u,h);
endp;
```

グラフ表示



GJR-GARCH(1,1)過程

ここでは、bad news(前期の残差がマイナス)があった後に、当期のボラティリティーｈが増すようなモデルは次のようになる。

$$h_t = a_0 + a_1 u_{t-1}^2 + b_1 h_{t-1} + \gamma D_{t-1} u_{t-1}^2$$

$$D_{t-1} = \begin{cases} 1, & if \quad u_{t-1} < 0 \\ 0, & if \quad u_{t-1} \geq 0 \end{cases}$$

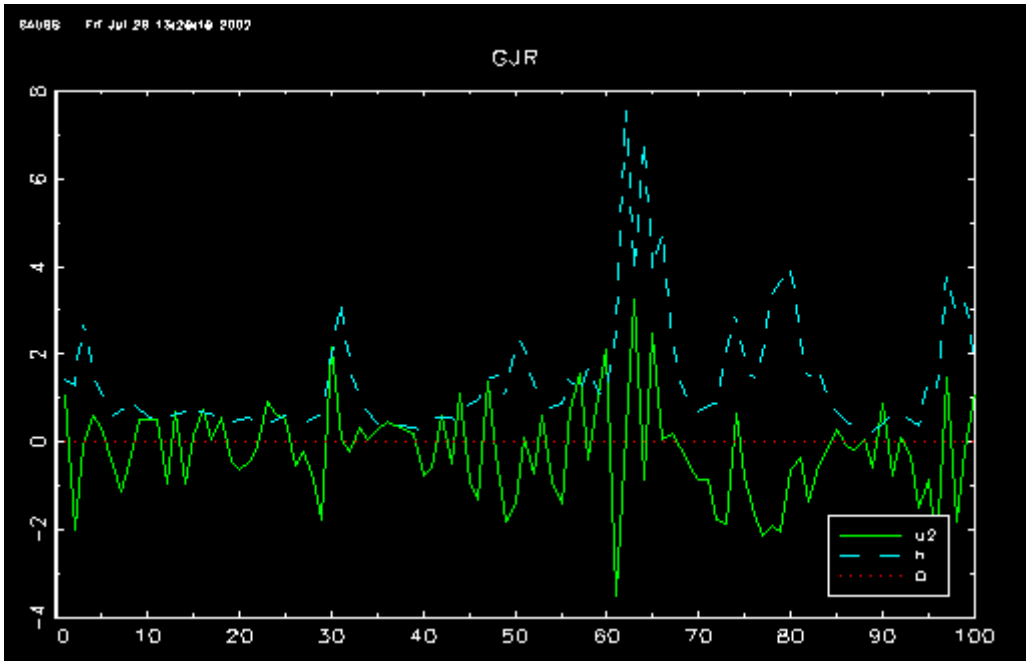
$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

プログラム

```
new; cls;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100; g=0.05;
{u,h}=gjrgarch11(a,b,a0,cutn,n,g);
library pgraph;
graphset;
title("GJR");
_plegctl=1;
_plegstr="u2¥000h¥0000";
xy(seqa(1,1,n),u~h~zeros(n,1));
```

```
proc(2)=gjrgarch11(a1,b1,a0,cutn,n,g);
    local s2,u,h,t;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    u[1]=sqrt(s2)*rndn(1,1); h[1]=s2;
    t=2;
    do while t<=cutn+n;
        if u[t-1]<0;
            h[t]=a0+a1*u[t-1]^2+b1*h[t-1]+g*u[t-1]^2;
        else;
            h[t]=a0+a1*u[t-1]^2+b1*h[t-1];
        endif;
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    retp(u,h);
endp;
```

グラフ表示



グラフから緑色の残差がマイナスに下がった次の期にｈが上昇しているのがよくわかる。

EGARCH(1,1)過程

ここでは、exponential をとった形のボラティリティーを考えます。すなわち、

$$h_t = \exp(a_0 + a_1 \frac{|u_{t-1}|}{\sqrt{h_{t-1}}} + b_1 \ln h_{t-1})$$

$$u_t = \sqrt{h_t}\varepsilon_t \text{ここで} \varepsilon_t \sim NID(0,1)$$

となります。ここでも、弱定常性の条件から各パラメータは０よりも大きく、a1+b1 は１よりも小さくなくてはなりません。そうでなくては、発散してしまうか、この場合、グラフの下が計算不能になり切れてしまいます。また、初期の s2 の値を３とか４とか以下の小さい正の値に抑えておかないと、ここでは、exponential が計算に絡んでいるの、ｈが非現実的に u から上方に乖離してしまいます。下のプログラムでは、その点のエラーチェックは行なわずに最低限のものを示しています。

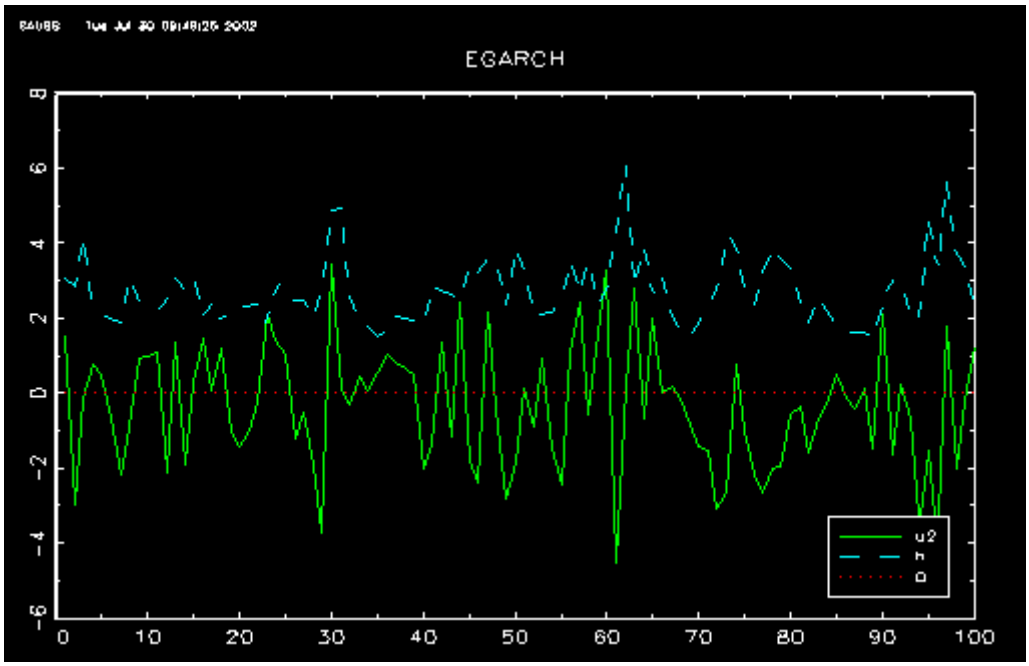
プログラム

```
new; cls;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100;
{u,h}=egarch(a,b,a0,cutn,n);
library pgraph;
graphset;
title("EGARCH");
_plegctl=1;
_plegstr="u2¥000h¥0000";
xy(seqa(1,1,n),u~h~zeros(n,1));

proc(2)=egarch(a1,b1,a0,cutn,n);
    local s2,u,h,t;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    u[1]=exp(ln(s2)/2)*rndn(1,1); h[1]=exp(s2);
    t=2;
    do while t<=cutn+n;
        h[t]=exp(a0+a1*abs(u[t-1])/sqrt(h[t-1])+b1*ln(h[t-1]));
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
```

```
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    retp(u,h);
endp;
```

画面表示



なお、上のプログラムでｈの計算として exponential をあらかじめとった計算を再帰的に繰り返していますが、

```
h[t]=exp(a0+a1*abs(u[t-1])/sqrt(h[t-1])+b1*ln(h[t-1]));
u[t]=sqrt(h[t])*rndn(1,1);
```

を、１つの変数 lnh をあらかじめ領域確保して作っておき、exp をはずした形で、

```
lnh[t]= a0+a1*abs(u[t-1])/sqrt(h[t-1])+b1*lnh[t-1];
u[t]=exp(lnh(t)/2)*rndn(1,1);
```

としても同じことです。U[t]のところは両辺に ln をとって、rndn(1,1)のところを考えずに計算すると、結局は 2*lnu[t]=lnh[t]となり、lnu[t]^2=lnh[t]ですから、lnh[t]は ln(h[t])を１つの変数に置いていましたから、u[t]^2＝h[t]となり、u[t]=sqrt(h[t])と同じです。

本章では一貫して NID(0,1)の乱数過程を採用しているが、プログラム中の rndn(1,1)と置き換えてやって、テールのより分厚い t 分布による過程や GED の過程にしてやることもできる。推定する場合、それに対応する Log-Likelihood の計算は GARCH のものとは若干異なってくる。

EGARCH(1,1)の推定

GARCH と同じ Log-Likelihood のベクトル形

$$LL = -\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(h) - \frac{e^2}{2h} \qquad where \quad e = y - x\beta$$

を用いて、そのなかのｈのところに EGARCH の過程を再帰的に代入していく。なお、GARCH と同様に u_{t-1} のタームには、データの残差を直接用いて、初期値 h[1]は直接は計算できないので、unconditional variance の s2 を計算してそれを exponential をとった形で与えることにする。ここは、最初から回らないようであれば、ほかのタームを０として a0 の exponential をとった形を与えてもかまわないであろう。推定では、第１期をウエイトを０にして推定している。なお、パラメータには a0＞０，０≦a1,b1≦１および a1+b1＜１の制約を加えてやって GARCH と同様に推定している。乱数設定は次のように

n	β_0	β_1	X_1	a0	a1	b1
100	0.5	2	[-5,5]	0.1	0.45	0.5

GARCH と同じような定数項ありの１説明変数のモデルを推定するものとする。

プログラム

```
new; cls;
library cml;
cmlset;
rndseed 111;
a=0.45; b=0.5; a0=0.1; cutn=20; n=100; beta={0.5,2};
{y,x}=egarchsim(a,b,a0,cutn,n,beta);
data=y~x;
_cml_Bounds={ -1e256     1e256,
              -1e256     1e256,
              0.001      1e256,
              0                1,
              0                1};
_cml_c={0 0 0 -1 -1}; _cml_d=-0.9999;
_cml_ParNames = {"const","beta","a0","a1","b1"};
_cml_Algorithm=4;
start={0,0,1,0.1,0.1};
__weight=(rows(data)/(rows(data)-1))*ones(rows(data),1); __weight[1]=zeros(1,1);
{x,f,g,cov,retcode}=cml(data,0,&ll,start);
cl=cmltlimits(x,cov);
```

```
call cmlclprt(x,f,g,cl,retcode);
rndseed 111;
proc ll(b,data);
    local y,x,e2,a0,a1,b1,h,t;
    y=data[.,1]; x=data[.,2:3]; beta=b[1:2];
    e2=(y-x*beta)^2;
    a0=b[3]; a1=b[4]; b1=b[5];
    h=zeros(rows(data),1);
    h[1]=exp(a0/(1-a1-b1));
    t=2;
    do while t<=rows(data);
        h[t]=exp(a0+a1*sqrt(e2[t-1])/sqrt(h[t-1])+b1*ln(h[t-1]));
        t=t+1;
    endo;
    retp(-1/2*ln(2*pi)-1/2*ln(h)-1/2*e2./h);
endp;

proc(2)=egarchsim(a1,b1,a0,cutn,n,beta);
    local s2,u,h,t,x,y;
    s2=a0/(1-a1-b1);
    u=zeros(cutn+n,1);
    h=zeros(cutn+n,1);
    u[1]=exp(ln(s2)/2)*rndn(1,1); h[1]=exp(s2);
    t=2;
    do while t<=cutn+n;
        h[t]=exp(a0+a1*abs(u[t-1])/sqrt(h[t-1])+b1*ln(h[t-1]));
        u[t]=sqrt(h[t])*rndn(1,1);
        t=t+1;
    endo;
    u=u[cutn+1:cutn+n];
    h=h[cutn+1:cutn+n];
    x=ones(n,1)~(10*rndu(n,1)-5); /* Set a constant and X of [-5,5] here. */
    y=x*beta+u;
    retp(y,x);
endp;
```

画面表示

Mean log-likelihood -1.93900

Number of cases 100

		0.95 confidence limits		
Parameters	Estimates	Lower Limit	Upper Limit	Gradient
const	0.6106	0.2978	0.9234	0.0000
beta	2.0541	1.9480	2.1602	-0.0000
a0	0.1373	-0.2593	0.5340	-0.0000
a1	0.4214	-0.0295	0.8724	-0.0000
b1	0.5127	-0.0260	1.0513	-0.0000

Number of iterations 35

Minutes to convergence 0.10250

乱数項がNID(0,1)である限りは、上と同じLog-Likelihoodで推定できる。GARCHの変種のモデルによってｈに行くまでの計算との初期値の設定が若干かわってくるだけである。当然のことながら、非対称な形になると収束し難くなり若干の工夫を要することもある。なお、乱数部がｔ分布によるものやGEDによる幅広のものは、通常のLog-Likelihoodよりも複雑な形となり、また複雑な定数項が出てくることなる。