```
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@
Simulated Density Paper
Jun 2005

Density Graphs: Figures 2 and 3 from the paper
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@*/
new; cls;
library cml, pgraph;
clearg Tg, ee_gmm, ee_sim;

output file = c:\fig_data.out reset;
outwidth 255; output on; format /MA1 /LD 6,4;

        "Data for Density Graphs: Figures 2 and 3 from the paper";
        "Geetesh Bhardwaj,Valentina Corradi, and Norman R. Swanson, June 2005, ";
        "A Simulation Based Specification Test for Diffusion Processes"; " ";

load xt[] = c:\ed1_w.txt;
xt=xt/100;

Tg=rows(xt);

x1=xt;
arg=seqa(minc(x1),(maxc(x1)-minc(x1))/199,200);
d=Lden(arg,xt);


{b1,s1}=est_cir(xt);
{a,d1,d2,d4,d12}=sim_Lden(xt,b1,1);
"Figure 2 Panel 1";
"Arg, True, 1-Step, 2-Step, 4-Step, 12-step";
a~d~d1~d2~d4~d12;
d=d~d1~d2~d4~d12;
xy(a,d);


{b1,s1}=est_ou(ln(xt));
{a,d1,d2,d4,d12}=sim_Lden(xt,b1,2);
"Figure 2 Panel 2";
"Arg, True, One-Step, 2-Step, 4-Step, 12_step";
a~d~d1~d2~d4~d12;
d=d~d1~d2~d4~d12;
xy(a,d);


xt=xt[992:rows(xt),.];
x1=xt;
arg=seqa(minc(x1),(maxc(x1)-minc(x1))/199,200);
d=Lden(arg,xt);


{b1,s1}=est_cir(xt);
{a,d1,d2,d4,d12}=sim_Lden(xt,b1,1);
"Figure 3 Panel 1";
"Arg, True, 1-Step, 2-Step, 4-Step, 12-step";
a~d~d1~d2~d4~d12;
d=d~d1~d2~d4~d12;
xy(a,d);


{b1,s1}=est_ou(ln(xt));
```

```
{a,d1,d2,d4,d12}=sim_Lden(xt,b1,2);
"Figure 3 Panel 2";
"Arg, True, One-Step, 2-Step, 4-Step, 12_step";
a~d~d1~d2~d4~d12;
d=d~d1~d2~d4~d12;
xy(a,d);




end;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*   sim_Lden: returns the Kernal density for 1,2,4, and 12 step ahead simulations
                                                   *
************************************************************************************
*****************************
* Inputs:         x      -   The time series
                                           *
*             b1     -   Estimated coefficients
                                           *
*             mod_ind-   Model indicator 1 for CIR, 2 for OU
                                           *
* Output:     arg    -   Points where density is evaluated
                                           *
*             den1   -   1 step ahead simulated density
                                           *
*             den2   -   2 step ahead simulated density
                                           *
*             den4   -   5 step ahead simulated density
                                           *
*             den12  -   12 step ahead simulated density
                                           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
proc (5)= sim_Lden(x1,b1,mod_ind);
local ii,x_sim,den1,den2,den4,den12,arg,count;
ii=0;den1={};den2={};den4={};den12={};count=0;
    do while ii<rows(x1)-12;ii=ii+1;
        if mod_ind==1;
            x_sim=dgp_cir(12,1/Tg,b1[1],b1[2],b1[3],x1[ii]);
        elseif mod_ind==2;
            x_sim=dgp_ou(12,1/Tg,b1[1],b1[2],b1[3],ln(x1[ii]));
            x_sim=exp(x_sim);
        endif;
        den1=den1|x_sim[1];
        den2=den2|x_sim[2];
        den4=den4|x_sim[4];
        den12=den12|x_sim[12];
        if ii==rows(x1)-12;
            ii=0;count=count+1;
        endif;
        if count==10;
            ii=rows(x1);
        endif;
    endo;
    arg=seqa(minc(x1),(maxc(x1)-minc(x1))/199,200);
    den1=Lden(arg,den1);    den2=Lden(arg,den2);
    den4=Lden(arg,den4);    den12=Lden(arg,den12);
retp(arg,den1,den2,den4,den12);
endp;
```

```
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
*   Lden: returns the Kernal density
           *
************************************************************************************
*************
* Inputs:        arg    -   the values at which the density is to evaluated
               *
*             v      -   The realised values of the series for which density is to
be estimates *
* Output:     p      -   density
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc  Lden(arg,v);
local Narg,r,N,h,P,j,d,t1;
    Narg = rows(arg);
    r=1/(4+cols(v));
    N=rows(v);
    h=(1/(N^r))*stdc(v)';
    P=zeros(Narg,1);
    j=0;
    do while j < Narg;j=j+1;
        d=(arg[j,.]-V)./h;
        t1=pdfn(d)./h;
        t1=prodc(t1');
        P[j]=meanc(t1);
    endo;
retp(P);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
*   est_OU: returns the SGMM estimaes and standard errors for OU process
           *
************************************************************************************
*************
* Inputs:        x1      -   time series
               *
* Output:     b_OU   -   estimates
           *
*             se_OU  -   GMM standard errors
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc (2)= est_OU(x);
local b_OU,f,g,cov,retcode,se_OU;
    b_OU=start_ou(x);
    cmlset;
    __output=0;_cml_Algorithm = 1;_cml_LineSearch = 2;_cml_GradMethod =
1;_cml_DirTol = 1e-5;_cml_DFTol = 1e-5;
    _cml_C = {  1 0 0,
                0 0 1};
    _cml_D = {  0,
                0};
    ee_gmm=sqrt(1/Tg)*rndn(10*Tg*Tg,1);
    {b_ou,f,g,cov,retcode} = CML(x,0,&SGMM_OU,b_OU);
    SE_OU=OU_SE(b_ou,x);
    SE_OU=SQRT(DIAG(SE_OU));
retp(b_ou,se_ou);
endp;
```

```
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
*   est_cir: returns the SGMM estimaes and standard errors for CIR process
           *
*************************************************************************************
*************
* Inputs:         x1      -   time series s
               *
* Output:     b_cir   -   estimates
           *
*             se_cir  -   GMM standard errors
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc (2)= est_cir(x);
local b_cir,f,g,cov,retcode,se_cir;
    b_cir=start_cir(x);
    cmlset;
    __output=0;_cml_Algorithm = 1;_cml_LineSearch = 2;_cml_GradMethod =
1;_cml_DirTol = 1e-5;_cml_DFTol = 1e-5;
    _cml_C = {  1 0 0,
                0 1 0,
                0 0 1};
    _cml_D = {  0,
                0,
                0};
    _cml_IneqProc = &ineq_CIR;
    ee_gmm=sqrt(1/Tg)*rndn(10*Tg*Tg,1);
    {b_CIR,f,g,cov,retcode} = CML(X,0,&SGMM_CIR,b_CIR);
    SE_CIR=CIR_SE(b_CIR,X);
    SE_CIR=SQRT(DIAG(SE_CIR));
retp(b_cir,se_cir);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
*   SGMM_OU: Returns the obj func
           *
*   dp(t)=phi*(p_bar-p(t))*dt+(sig2)*dW(t)
           *
*************************************************************************************
*************
* Inputs:       b1      -   starting values
               *
*                   x1      -   time series
                   *
* Output:           -   the objective function to be minimised
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc SGMM_OU(b1,x1);
local XS, g_prime, g, q, meanf, f, w0, invW, v, W, auto,s;
        s=rows(x1)*10;
    xs=dgp_OUS(s,1/Tg,b1[1],b1[2],b1[3],b1[2]);
    g_prime= meanc(X1) - meanc(Xs);
        g_prime=
g_prime~(((meanc((x1[1:rows(x1)-1]-meanc(x1)).*(x1[2:rows(x1)]-meanc(x1))))) -
((meanc((xs[1:rows(xs)-1]-meanc(xs)).*(xs[2:rows(xs)]-meanc(xs))))));
        g_prime= g_prime~(vcx(x1) - vcx(xs));
    g=g_prime';
    q=int(rows(X1)^(1/6));

f=(X1[2:rows(x1)])~(((x1[1:rows(x1)-1]-meanc(x1)).*(x1[2:rows(x1)]-meanc(x1))))~((x1
```

```
[2:rows(x1)]-meanc(x1))^2);
        f=f';
        meanf=meanc(f');
        w0=(1/rows(X1))*((f-meanf)*(f-meanf)');
        invW=w0;
        if q > 0;
                v=0;
                do while v < q; v=v+1;

auto=(1/rows(x1))*(((f[.,v+1:rows(f')]-meanf)*(f[.,1:rows(f')-v]-meanf)')+((f[.,1:ro
ws(f')-v]-meanf)*(f[.,v+1:rows(f')]-meanf)'));
                        invW=invW+(1-(v/(q+1)))*auto;
                endo;
        endif;
        W=inv(invW);
retp(-g_prime*W*g);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
*   SGMM_CIR: Returns the obj func
           *
*   dp(t)=phi*(p_bar-p(t))*dt+(sig1*sqrt(p(t)))*dW(t)
           *
*************************************************************************************
*************
* Inputs:       b1      -   starting values
               *
*                   x1      -   time series
                   *
* Output:           -   the objective function to be minimised
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc SGMM_CIR(b1,x1);
local XS, g_prime, g, q, meanf, f, w0, invW, v, W, auto,s;
        s=rows(x1)*10;
    xs=dgp_cirS(s,1/Tg,b1[1],b1[2],b1[3],b1[2]);
    g_prime= meanc(X1) - meanc(Xs);
        g_prime=
g_prime~(((meanc((x1[1:rows(x1)-1]-meanc(x1)).*(x1[2:rows(x1)]-meanc(x1))))) -
((meanc((xs[1:rows(xs)-1]-meanc(xs)).*(xs[2:rows(xs)]-meanc(xs))))));
        g_prime= g_prime~(vcx(x1) - vcx(xs));
    g=g_prime';
    q=int(rows(X1)^(1/6));

f=(X1[2:rows(x1)])~(((x1[1:rows(x1)-1]-meanc(x1)).*(x1[2:rows(x1)]-meanc(x1))))~((x1
[2:rows(x1)]-meanc(x1))^2);
        f=f';
        meanf=meanc(f');
        w0=(1/rows(X1))*((f-meanf)*(f-meanf)');
        invW=w0;
        if q > 0;
                v=0;
                do while v < q; v=v+1;

auto=(1/rows(x1))*(((f[.,v+1:rows(f')]-meanf)*(f[.,1:rows(f')-v]-meanf)')+((f[.,1:ro
ws(f')-v]-meanf)*(f[.,v+1:rows(f')]-meanf)'));
                        invW=invW+(1-(v/(q+1)))*auto;
                endo;
        endif;
        W=inv(invW);
retp(-g_prime*W*g);
```

```
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
* ineq_CIR: Single Factor Square Root Process Nonlinear Inequality constraints
           *
************************************************************************************
*************
*   dp(t)=phi*(p_bar-p(t))*dt+(sig1*sqrt(p(t)))*dW(t)
           *
*   Nonlinear Inequality constraints
           *
*   2*phi*p_bar >= sig1^2
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc ineq_CIR(b);
retp(2*b[1]*b[2]-b[3]^2);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
*   OU_SE: Returns the gmm var-cov matrix for OU Process
           *
************************************************************************************
*************
* Inputs:       b1      -   estimated values
               *
*                   x1      -   time series
                   *
* Output:           -   gmm var-cov matrix
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/

proc OU_SE(b1,x1);
local se, XS, g_prime, g, q, meanf, f, w0, invW, v, W, auto,s;
    W=SE_W(b1,x1);
    q=GRADp(&grad_f21,b1);
    g=meanc(q);
    q=GRADp(&grad_f22,b1);
    g=g~meanc(q);
    q=GRADp(&grad_f23,b1);
    g=g~meanc(q);
    se=inv(g*W*g');
retp(se/rows(x1));
endp;
proc grad_f21(b);
local g,s,xs;
        s=Tg*10;
        Xs=dgp_OUS(s,1/TG,b[1],b[2],b[3],b[2]);
    g= (Xs);
retp(g);
endp;
proc grad_f22(b);
local g,s,xs;
        s=Tg*10;
        Xs=dgp_OUS(s,1/TG,b[1],b[2],b[3],b[2]);
        g= (xs[1:rows(xs)-1]-meanc(xs)).*(xs[2:rows(xs)]-meanc(xs));
retp(g);
endp;
```

```
proc grad_f23(b);
local g,s,xs;
        s=Tg*10;
        Xs=dgp_OUS(s,1/TG,b[1],b[2],b[3],b[2]);
        g= (xs-meanc(xs))^2;
retp(g);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
*   CIR_SE: Returns the gmm var-cov matrix for CIR Process
           *
*************************************************************************************
*************
* Inputs:        b1      -   estimated values
               *
*                   x1      -   time series
                   *
* Output:            -   gmm var-cov matrix
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/

proc CIR_SE(b1,x1);
local se, XS, g_prime, g, q, meanf, f, w0, invW, v, W, auto,s;
    W=SE_W(b1,x1);
    q=GRADp(&grad_f11,b1);
    g=meanc(q);
    q=GRADp(&grad_f12,b1);
    g=g~meanc(q);
    q=GRADp(&grad_f13,b1);
    g=g~meanc(q);
    se=inv(g*W*g');
retp(se/rows(x1));
endp;
proc SE_W(b1,x1);
local se, XS, g_prime, g, q, meanf, f, w0, invW, v, W, auto,s;
    q=int(rows(X1)^(1/6));

f=(X1[2:rows(x1)])~(((x1[1:rows(x1)-1]-meanc(x1)).*(x1[2:rows(x1)]-meanc(x1))))~((x1
[2:rows(x1)]-meanc(x1))^2);
        f=f';
        meanf=meanc(f');
        w0=(1/rows(X1))*((f-meanf)*(f-meanf)');
        invW=w0;
        if q > 0;
                v=0;
                do while v < q; v=v+1;

auto=(1/rows(x1))*(((f[.,v+1:rows(f')]-meanf)*(f[.,1:rows(f')-v]-meanf)')+((f[.,1:ro
ws(f')-v]-meanf)*(f[.,v+1:rows(f')]-meanf)'));
                        invW=invW+(1-(v/(q+1)))*auto;
                endo;
        endif;
        W=inv(invW);
retp(W);
endp;
proc grad_f11(b);
local g,s,xs;
        s=Tg*10;
        Xs=dgp_cirS(s,1/TG,b[1],b[2],b[3],b[2]);
    g= (Xs);
retp(g);
```

```
endp;
proc grad_f12(b);
local g,s,xs;
        s=Tg*10;
        Xs=dgp_cirS(s,1/TG,b[1],b[2],b[3],b[2]);
        g= (xs[1:rows(xs)-1]-meanc(xs)).*(xs[2:rows(xs)]-meanc(xs));
retp(g);
endp;
proc grad_f13(b);
local g,s,xs;
        s=Tg*10;
        Xs=dgp_cirS(s,1/TG,b[1],b[2],b[3],b[2]);
        g= (xs-meanc(xs))^2;
retp(g);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
* dgp_ou:   Data Generation as OU process
           *
*           dp(t)=phi*(p_bar-p(t))*dt+sig*dW(t)
           *
**********************************************************************************
*************
* Inputs:       T       -   Length of time series
               *
*                   h       -   discretization interval
                   *
*                   phi     -   mean reversion parameter Process
                   *
*                   p_bar   -   mean level
                   *
*                   sig1    -   variance term
                   *
*                   start   -   starting value of the process
                   *
* Output:   dat     -   time series
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc dgp_ou(T,h,phi,p_bar,sig1,start);
local TN,dat,Pt,i,X1,ee;
        TN=round(T/h);
    ee=sqrt(h)*rndn(T/h,1);
        dat={};
        Pt=start;
        i=0;
        do while i < TN;
                i=i+1;
                X1=Pt+phi*(p_bar-Pt)*h+sig1*ee[i];
                if i%(h^-1)==0;
                        dat=dat|X1;
                endif;
                Pt=X1;
        endo;
retp(dat);
endp;
/*same as above but for sgmm with errors that do not change over optimization*/
proc dgp_ous(T,h,phi,p_bar,sig1,start);
local TN,dat,Pt,i,X1,ee;
        TN=round(T/h);
    ee=ee_gmm;
        dat={};
```

```
        Pt=start;
        i=0;
        do while i < TN;
                i=i+1;
                X1=Pt+phi*(p_bar-Pt)*h+sig1*ee[i];
                if i%(h^-1)==0;
                        dat=dat|X1;
                endif;
                Pt=X1;
        endo;
retp(dat);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
* start_ou: Returns the starting values for the OU process
           *
*           dp(t)=phi*(p_bar-p(t))*dt+sig*dW(t)
           *
************************************************************************************
*************
* Inputs:   y   -   time series
           *
* Output:   b   -   1x3 vector of starting values for phi p_bar and sig
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc start_ou(y);
local mu,rho,sig,phi,test;
    mu=meanc(y);
    rho=meanc((y[1:rows(y)-1]-meanc(y)).*(y[2:rows(y)]-meanc(y)))/vcx(y);
    phi=-ln(abs(rho));
    sig=(vcx(y)*2*phi)^.5;

retp(phi~mu~sig);
endp;


/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
* dgp_cir:   Data Generation as CIR process
           *
*           dp(t)=phi*(p_bar-p(t))*dt+sig*sqrt(p(t)*dW(t)
           *
************************************************************************************
*************
* Inputs:       T       -   Length of time series
               *
*                   h       -   discretization interval
                   *
*                   phi     -   mean reversion parameter Process
                   *
*                   p_bar   -   mean level
                   *
*                   sig1    -   variance term
                   *
*                   start   -   starting value of the process
                   *
* Output:   dat     -   time series
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc dgp_cir(T,h,phi,p_bar,sig1,start);
```

```
local TN,dat,Pt,i,X1,ee;
        TN=round(T/h);
    ee=sqrt(h)*rndn(T/h,1);
        dat={};
        Pt=start;
        i=0;
        do while i < TN;
                i=i+1;
                X1=Pt+phi*(p_bar-Pt)*h+sig1*sqrt(pt)*ee[i]
            -0.5*(sig1*sqrt(pt))*(0.5*sig1/sqrt(pt))*h
            +0.5*(sig1*sqrt(pt))*(0.5*sig1/sqrt(pt))*(ee[i]^2);
        /*Theory implies that this condition will never be reached as when process
approaches
        zero the volatility is switched off, that result depends on h going to zero
here h is
        very small indeed I am including the condition as a final safe guard, here I
am switching
        off the volatility manually something that should happens assymptotically*/
        if X1 < 0;
            X1=Pt+phi*(p_bar-Pt)*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|X1;
                endif;
                Pt=X1;
        endo;
retp(dat);
endp;
/*same as above but for sgmm with errors that do not change over optimization*/
proc dgp_cirS(T,h,phi,p_bar,sig1,start);
local TN,dat,Pt,i,X1,ee;
        TN=round(T/h);
    ee=EE_GMM;
        dat={};
        Pt=start;
        i=0;
        do while i < TN;
                i=i+1;
                X1=Pt+phi*(p_bar-Pt)*h+sig1*sqrt(pt)*ee[i]
            -0.5*(sig1*sqrt(pt))*(0.5*sig1/sqrt(pt))*h
            +0.5*(sig1*sqrt(pt))*(0.5*sig1/sqrt(pt))*(ee[i]^2);
        if X1 < 0;
            X1=Pt+phi*(p_bar-Pt)*h;
        endif;
                if i%(h^-1)==0;
                        dat=dat|X1;
                endif;
                Pt=X1;
        endo;
retp(dat);
endp;

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@
* start_cir: Returns the starting values for the CIR process
           *
*          dp(t)=phi*(p_bar-p(t))*dt+sig*sqrt(p(t)*dW(t)
           *
*************************************************************************************
*************
* Inputs:   y   -   time series
           *
* Output:   b   -   1x3 vector of starting values for phi p_bar and sig
```

```
           *
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@*/
proc start_cir(y);
local b,mu,rho,sig,phi,test;
    mu=meanc(y);
    rho=meanc((y[1:rows(y)-1]-meanc(y)).*(y[2:rows(y)]-meanc(y)))/vcx(y);
    phi=-ln(abs(rho));
    sig=(vcx(y)*2*phi/mu)^.5;
    b=phi~mu~sig;
retp(b);
endp;
```