

## APPENDIX D

### MATLAB FUNCTIONS

#### D.1 SIMULATION OF DATA FROM THE SV MODELS

##### DATA A

```
function [y,x]=sim_model1(phi,Q,alpha,n)
% randsp : give random sample from the sv model
% y(t)=alpha+a(t)*x(t)+v(t)
% x(t)=phi*x(t-1)+w(t)
% w(t) Gaussian (0,Q)
% v(t) centered log of chisquares(1)
% n=1000; number of observation
```

```
w=randn(1,n+10000)*sqrt(Q);
v1=chi2rnd(1,1,n+10000);
cv=log(v1)+1.2749;
x0=randn(1);
x(1)=phi*x0+w(1);
for t=2:n+10000
    x(t)=phi*x(t-1)+w(t);
end
y=alpha+x+cv;
y=y(10001:n+10000);
x=x(10001:n+10000);

[y_A, x_A] = sim_model1(0.9, 1, -3, 1000);
```

##### DATA B

```
function [y,x]=sim_model2(phi,Q,alpha,mu1,R0,R1,p,n)
% sim_model2 : give stationary random sample from the sv model(2): normal mixtures
% y(t)=alpha+x(t)+v(t)
% x(t)=phi*x(t-1)+w(t)
% w(t) Gaussian (0,Q)
% v(t) mixtures of two normals N(mu1,R1),N(mu0,R0) mixing prob=p

w=randn(1,n+10000)*sqrt(Q);
```

```

Ind=(rand(1,n+10000)<p);
nor1=randn(1,n+10000);
v1=Ind.*(nor1*sqrt(R1)+mu1)+(1-Ind).*(nor1*sqrt(R0));

x0=randn(1);
x(1)=phi*x0+w(1);
for t=2:n+10000
    x(t)=phi*x(t-1)+w(t);
end
y=alpha+x+v1;
y=y(10001:n+10000);
x=x(10001:n+10000);

[y_B, x_B] = sim_model2(0.8, 1.5, -4, -3, 3, 5, 0.5, 1000);

```

## D.2 ALGORITHM A: PARAMETER ESTIMATION FOR CHAPTER 2

### Initial parameter selection

```

function Iparm=Iparm_c(y)

%parm=[phi, Q,alpha];
n=max(size(y));

parm(3)=mean(y);
y2=y(1:n-2);
y1=y(2:n-1);
y0=y(3:n);
A=cov(y0,y2);
B=cov(y1,y2);
parm(1)= min(A(1,2)/B(1,2),0.95);
parm(2)=max(mean((y0-mean(y0))-parm(1)*(y1-mean(y1))).^2) - 5 -5*parm(1)^2,0.1);

Iparm=parm

```

### Main function

```

function [Eparm,Vparm,Rellike,Tcal,Niter]=mainchis(y,a,Iparm,Tol,Miter,Npar,n,tt)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% main fuction for the estimation of parameters for regular sv model with missing values
% model: y(t)=alpha+a(t)x(t)+v(t) v(t):centered log of chisquares
%          x(t)=phi*x(t-1)+w(t) w(t):normal(0, Q) x(0):normal(mu0,Sig0)
%parameters: theta=[phi, Q, alpha]
%input values -- y: log(r^2) where r is returns,
%                a: missing indicator ( 1= observed, 0=missing),
%                Tol: tolerance, Miter=maximum iteration, Npar=number of particles.
%output values -- Eparm : History of parameter estimates [phi,Q, alpha],
%                Vparm: var-cov matrix of parameter estimates
%                Rellike: relative likelihood
%                : loglikelihood at iteration i- log likelihood at iteration i-1
%                Tcal: time for the whole calculation

```

```

%                               Niter: number of iteration until convergence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic %to get the time to elapse
% declaration of variables-----
Eparm=zeros(Miter, 3);
RelLike=zeros(Miter,1);
Tcal=0;

%estimation-----
for i=1:Miter
    y_full=y;
    %1.filtering step
    f=PfilterChis(Iparm, y_full, a, n, Npar);
    %2. smoothing step
    s=PsmootherChis(y_full, a, f, Iparm, n, Npar);
    %3. estimation step
    Rparm=EstChis(s, y_full, n, Npar)
    Eparm(i,:)=Rparm;

    %4. Relative likelihood
    if i>1,
        RelLike(i)=RelLikeChis(s,n,y,Eparm(i-1,:),Rparm,Npar);
    end
    if i==1
        RelLike(i)=RelLikeChis(s,n,y,Iparm,Rparm,Npar);
    end

    %5. Check convergence
    Con=0;
    if i>tt,
        Con=ConChis(RelLike,i ,Tol );
    end
    if Con==1, Niter=i, break, end
    Iparm=Rparm;

end

%6. Variance of parameter estimates
Vparmm=VarChis(s,y,n,Eparm(i-1,:),Npar)
Eparm=Eparm(1:i-1,:);
RelLike=RelLike(1:i-1,:);

Tcal=toc; %to get the time to elapse

```

### 1. Filtering function

```

function f=PfilterChis(Iparm, y, a, n, Npar)
% returns filtered values
% Iparm=theta=[ phi, Q, alpha]
%assign parameters
phi=Iparm(1);
Q=Iparm(2);
alpha=Iparm(3);

```

```

mu0=0;
Sig0=Q/(1-phi^2);

f(:,1)=randn(Npar, 1)*sqrt(Sig0)+mu0;
wt=randn(Npar, n)*sqrt(Q);
for t=1:n
    p=phi*f(:,t)+wt(:,t);
    w=exp((y(t)-a(t)*p-1.2749-alpha)/2).*exp(-1*exp(y(t)-a(t)*p-1.2749-alpha)/2);
    f(:,t+1)=Rsample1(p,w,Npar);
end

```

## 2. Smoothing function

```

function s=PsmootherChis(y, a, f, Iparm, n, Npar)
%particle smoother :
%Iparm=theta=[ phi, Q, alpha]
phi=Iparm(1);
Q=Iparm(2);
alpha=Iparm(3);
mu0=0;
Sig0=Q/(1-phi^2);

%for t=n
st=unidrnd(Npar,1,Npar);
s(:,n+1)=f(st,n+1);

for j=1:Npar
    for t=n:-1:1
        w=exp(-1*(s(j,t+1)-phi*f(:,t)).^2/(2*Q));
        s(j,t)=RSample1(f(:,t),w,1);
    end
end
end

```

## 3. Estimation function

```

function Rparm=EstChis(s, y, n,Npar)

%theta=[ phi, Q, alpha]

phi=sum(mean(s(:,1:n)).*s(:,2:n+1)))/sum(mean(s(:,1:n).^2));
Q=mean(mean((s(:,2:n+1)-phi*s(:,1:n)).^2));

y_ext= repmat(y, Npar, 1);
alpha=log(mean(mean(exp(y_ext-s(:,2:n+1)-1.2749))));

Rparm=[phi,Q,alpha];

```

## 4. Relative likelihood

```

function Rel_Like=RelLikeChis(s,n,y,oldp,newp,m )

temp=0;

```

```

for j=1:m
    loglike1=comp_loglike(s(j,:),y,oldp,n);
    loglike2=comp_loglike(s(j,:),y,newp,n);
    m_log=(loglike1+loglike2)/2;
    temp=temp+exp(loglike1-m_log)/exp(loglike2-m_log);
end

Rel_Like=-1*log(temp/m)

```

### 5. Assessing convergence

```

function Con=ConChis(RelLike,i ,Tol )
Con=0;
if abs(RelLike(i))<Tol,
    Con=1;
end

```

### 6. Variance of parameter estimates

```

function [vparamm,vparmt,term1,term2,term2_trim,term3a]=VarChis_trim(s,y,theta,ptrim)
% theta=[phi, Q, alpha]
phi=theta(1);
Q=theta(2);
alpha=theta(3);
[Npar,n]=size(s);
n=n-1;

term1=zeros(3,3);
term2=zeros(3,3);
term3a=zeros(3,1);
ddl=zeros(3,3);
dl=zeros(3,1);
term2t=zeros(3,Npar);

for i=1:Npar
    x1=s(i,:);
    dl(1)=sum(x1(1:n).*(x1(2:n+1)-phi*x1(1:n)))/Q;
    dl(2)=-1*n/(2*Q)+sum(((x1(2:n+1)-phi*x1(1:n)).^2))/(2*Q^2);
    dl(3)=0.5*sum(exp(y-x1(2:n+1)-alpha-1.2749)-1);

    term2=term2+dl*dl'/Npar;
    term3a=term3a+dl/Npar;
    term2t(:,i)=dl;

    ddl(1,1)=-1*sum(x1(1:n).^2)/Q;
    ddl(1,2)=-1*sum(x1(1:n).*(x1(2:n+1)-phi*x1(1:n)))/Q^2;
    ddl(2,1)=ddl(1,2);
    ddl(2,2)=-1*sum((x1(2:n+1)-phi*x1(1:n)).^2)/Q^3 +n/(2*Q^2);
    ddl(3,3)=-0.5*sum(exp(y-x1(2:n+1)-alpha-1.2749));

    term1=term1+ddl/Npar;
end
term2_trim=zeros(3,3);

```

```

for i=1:Npar
    kk=term2t(:,i)*term2t(:,i)';
    for j=1:9
        forterm2(i,j)=kk(fix((j-1)/3)+1,rem(j-1,3)+1);
    end
end
term2_temp=trimmean(forterm2,ptrim)

for i=1:3
    for j=1:3
        term2_trim(i,j)=term2_temp((i-1)*3+j);
    end
end

obInfo=-1*term1 -term2 + (term3a)*(term3a)';
obsInfot=-1*term1-term2_trim+(term3a)*(term3a)';
vparmt=inv(obsInfot);
vparmm=inv(obInfo)

```

## Miscellaneous functions

### Resampling function

```

function Newdata=Rsample1(data,weight,NofSample)
n=max(size(data));
re_ind=rand(1,NofSample);
cmwt=cumsum(weight)/sum(weight);
for k=1:NofSample
    st=(re_ind(k)>cmwt(1:n-1));
    Newdata(k)=data(sum(st)+1);
end
Newdata=Newdata';

```

### Complete likelihood

```

function loglike=comp_loglike(x,y,theta,n)

phi=theta(1);
Q=theta(2);
alpha=theta(3);
mu0=0;
Sig0=Q/(1-phi^2);

loglike=-1*(log(Sig0)+(x(1)-mu0)^2/Sig0+n*log(Q)+sum((x(2:n+1)-...
    phi*x(1:n)).^2)/Q+sum(exp(y-x(2:n+1)-alpha-1.2749)-(y-x(2:n+1)-alpha-1.2749)))/2;

```

### Data completion function

```

function y_full=completey_c(y,a,init)
% complete y with missing by taking random sample

n=max(size(y));

```

```

for t=1:n
    if a(t)==0
        rdn=randn(1,1)
        y_full(t)=init(3)+log(rdn^2)-1.2749;
    else
        y_full(t)=y(t);
    end
end
end

```

## D.3 ALGORITHM B: PARAMETER ESTIMATION FOR CHAPTER 3 AND CHAPTER 4

### Initial parameter selection

```

function parm=Iparmnorm(y)
%parm=[phi, Q,m0,m1,R0,R1,pi];
%m0>m1;
k1=3;
k2=4;
k3=4;
k4=0.5;
n=max(size(y));
y2=y(1:n-2);
y1=y(2:n-1);
y0=y(3:n);
A=cov(y0,y2);
B=cov(y1,y2);
parm(1)= min(A(1,2)/B(1,2),0.95);
parm(3)=mean(y)+k4*k1;
parm(4)=parm(3)-k1;
parm(5)=k2;
parm(6)=k3;
parm(7)=k4;
varofv=k1^2*k4*(1-k4)+k3*k4+k2*(1-k4);
parm(2)=max(mean((y0-mean(y0))-parm(1)*(y1-mean(y1))).^2) -...
    varofv-varofv*parm(1)^2,0.1);

```

### Main function

```

function [Eparm,Vparmm,Vparmt,RelLike,Tcal,Niter]=...
    mainnorm(y,a,Iparm,Tol,Miter,Npar,n,tt)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% main fuction for the estimation of parameters for regular sv model      %
%                               with missing values with stationary assumption %
% use auxiliary PF                                                         %
% model: y(t)=alpha+a(t)x(t)+v(t)                                          %
%       v(t):two normal mixtures (1-I(t))*N(m0,R0)+I(t)*N(m1,R1),        %
%       I(t)~bernoulli(pi)                                                %
%       x(t)=phi*x(t-1)+w(t) w(t):normal(0, Q) x(0):normal(mu0,Sig0)      %
% From stationary assumption: mu0=0, Sig0=Q/(1-phi^2)                      %
%parameters: theta=[phi, Q,m0,m1,R0,R1,pi];                               %

```

```



```

```

end

%6. Check convergence
Con=0;
if i>tt,
[Con,neg]=ConNorm(RelLike,i ,Tol );
end
if Con==1, Niter=i, break, end
Iparm=Rparm;
end

%7. variance calculation
[Vparmm,Vparmt,term1,term2,term2_trim,term3a]=v2_trim(s_x,s_I,Eparm(i-1,:),y_full,5)

Eparm=Eparm(1:i-1,:);
RelLike=RelLike(1:i-1,:);

Tcal=toc; %to get the time to elapse

```

## 1. Data completion function

```

function y_full=DataCompChis(y,a,Iparm,n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Data completion step for 2 normal mixtures sv model
% parm=[ phi,Q, m0,m1,R0,R1,pi],
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

phi=Iparm(1);
Q=Iparm(2);
m0=Iparm(3);
m1=Iparm(4);
R0=Iparm(5);
R1=Iparm(6);
pi=Iparm(7);

Ind=(rand(1,n)<pi);
rdn=randn(1,n);
y_miss=Ind.*(rdn*sqrt(R1)+m1)+(1-Ind).*(rdn*sqrt(R0)+m0);
y(a==0)=0;
y_full=a.*y+(1-a).*y_miss;

```

## 2. Filtering function

```

function [f_x,f_I]=PfilterNormaux(Iparm, y, a, n, Npar)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PfilterNormaux returns filtered values based on auxiliary PF: fully adapted.
%Iparm=theta=[phi, Q, m0,m1,R0,R1,pi]
%ref:Pitt and Shephard (2001); Sequential Monte Carlo methods in Practice,
% Edt'd Doucet et al
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%assign parameters---
phi=Iparm(1);
Q=Iparm(2);

```

```

m0=Iparm(3);
m1=Iparm(4);
R0=Iparm(5);
R1=Iparm(6);
prob=Iparm(7);
mu0=0;
Sig0=Q/(1-phi^2);
%-----

%for t=0
f_x(:,1)=randn(Npar, 1)*sqrt(Sig0)+mu0;
f_I(:,1)=(rand(Npar,1)<prob);
    tttt= repmat(1,1,Npar);
    temp_mat=[tttt,1-tttt;1-tttt,tttt];
    temp_pi=[1-prob,prob];
    temp_pi=temp_pi*temp_mat;
    temp_R=[R0,R1];
    temp_R=temp_R*temp_mat;
    temp_m=[m0,m1];
    temp_m=temp_m*temp_mat;
for t=1:n
    %draw auxiliary variable and normal index(k,j)
    temp_x=repmat(f_x(:,t)',1,2);
    weight1=temp_pi.*exp(-1*(y(t)-temp_m-phi*temp_x).^2./...
        (2*(temp_R+Q)))./sqrt(temp_R+Q);
    index_temp=1:Npar*2;
    index_sampled=Rsample1(index_temp,weight1,Npar);
    aux_sampled=temp_x(index_sampled);
    I_sampled=(index_sampled>Npar)';
    R_f=R0*(1-I_sampled)+R1*I_sampled;
    R_star=(R_f*Q)./(R_f+Q);
    m_star=R_star.*((y(t)-m0*(1-I_sampled)-m1*I_sampled)./R_f+...
        phi*aux_sampled/Q);
    f_x(:,t+1)=(randn(1,Npar).*sqrt(R_star)+m_star)';
    f_I(:,t+1)=I_sampled';
end

```

### 3. Smoothing function

```

function [s_x,s_I]=PsmootherNorm(y_full, a, f_x,f_I, Iparm, n, Npar)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%particle smoother : returns particle smoother
%Iparm=theta=[phi, Q,m0,m1,R0,R1,pi];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
phi=Iparm(1);
Q=Iparm(2);
m0=Iparm(3);
m1=Iparm(4);
R0=Iparm(5);
R1=Iparm(6);
pi=Iparm(7);
%-----
%for t=n
st=unidrnd(Npar,1,Npar);

```

```

s_x(:,n+1)=f_x(st,n+1);
s_I(:,n+1)=f_I(st,n+1);

for j=1:Npar
    for t=n:-1:1
        w=exp(-1*(s_x(j,t+1)-phi*f_x(:,t)).^2/(2*Q)).*(pi*s_I(j,t+1)+...
                (1-pi)*(1-s_I(j,t+1)));
        [s_x(j,t),s_I(j,t)]=RSample2(f_x(:,t),f_I(:,t),w,1);
    end
end
end

```

#### 4. Estimation function

```

function Rparam=EstNorm(s_x,s_I, y, n, Npar)
    %%%%%%%%%%%
    % Parameter estimation
    %theta=[ phi, Q, m0,m1,R0,R1,pi]
    %%%%%%%%%%%

    phi=sum(mean(s_x(:,1:n).*s_x(:,2:n+1)))/sum(mean(s_x(:,1:n).^2));
    Q=mean(mean((s_x(:,2:n+1)-phi*s_x(:,1:n)).^2));

    y_ext= repmat(y, Npar, 1);
    m0=sum(mean((1-s_I(:,2:n+1)).*(y_ext-s_x(:,2:n+1))))/(n-sum(mean(s_I(:,2:n+1))));
    m1=sum(mean(s_I(:,2:n+1)).*(y_ext-s_x(:,2:n+1))))/(sum(mean(s_I(:,2:n+1))));

    R0=sum(mean((1-s_I(:,2:n+1)).*((y_ext-s_x(:,2:n+1)-m0).^2)))/(n-...
            sum(mean(s_I(:,2:n+1))));
    R1=sum(mean(s_I(:,2:n+1)).*((y_ext-s_x(:,2:n+1)-m1).^2))/...
            (sum(mean(s_I(:,2:n+1))));

    pi=mean(mean(s_I(:,2:n+1)));

    Rparam=[phi,Q,m0,m1,R0,R1,pi];

```

#### 5. Relative likelihood

```

function Rel_Like=RelLikeNorm(s_x,s_I,n,y,oldp,newp,m )

temp=0;
for j=1:m
    loglike1=comp_logliken(s_x(j,:),s_I(j,:),y,oldp,n);
    loglike2=comp_logliken(s_x(j,:),s_I(j,:),y,newp,n);
    m_log=(loglike1+loglike2)/2;
    temp=temp+exp(loglike1-m_log)/exp(loglike2-m_log);
end

Rel_Like=-1*log(temp/m)

```

#### 6. Assessing convergence

```

function [Con,neg]=ConNorm(RelLike,i ,Tol )
Con=0;
if RelLike(i)<Tol,

```

```

    Con=1;
end
neg=0;
if RelLike(i)<0, disp('RELLIKE NEGATIVE'), neg=1
end

```

## 7. Variace estimates

```

function [vparmm,vparmt,term1,term2,term2_trim,term3a]=v2_trim(sx,sI,theta,y,ptrim)
phi=theta(1);
Q=theta(2);
m0=theta(3);
m1=theta(4);
R0=theta(5);
R1=theta(6);
prob=theta(7);
[Npar,n]=size(sx);
n=n-1;

term1=zeros(7,7);
term2=zeros(7,7);
term3a=zeros(7,1);
ddl=zeros(7,7);
dl=zeros(7,1);
term2t=zeros(7,Npar);

for i=1:Npar
    x1=sx(i,:);
    I1=sI(i,2:n+1);
    sumI=sum(I1);
    dl(1)=sum(x1(1:n).*(x1(2:n+1)-phi*x1(1:n)))/Q;
    dl(2)=-0.5*(n/Q - sum((x1(2:n+1)-phi*x1(1:n)).^2)/Q^2);
    dl(3)=sumI/prob-(n-sumI)/(1-prob);
    dl(4)=sum( (1-I1).*(y-x1(2:n+1)-m0))/R0;
    dl(5)=sum(I1.*(y-x1(2:n+1)-m1))/R1;
    dl(6)=-0.5*((n-sumI)/R0 - sum((1-I1).*(y-x1(2:n+1)-m0).^2)/R0^2);
    dl(7)=-0.5*(sumI/R1-sum(I1.*(y-x1(2:n+1)-m1).^2)/R1^2);

    term2=term2+dl*dl'/Npar;
    term3a=term3a+dl/Npar;
    term2t(:,i)=dl;

    ddl(1,1)=-sum(x1(1:n).^2)/Q;
    ddl(1,2)=-1*dl(1)/Q;
    ddl(2,1)=ddl(1,2);
    ddl(2,2)=n/(2*Q^2)-sum((x1(2:n+1)-phi*x1(1:n)).^2)/Q^3;
    ddl(3,3)=-1*sumI/prob^2-(n-sumI)/(1-prob)^2;
    ddl(4,4)=-1*(n-sumI)/R0;
    ddl(5,5)=-1*sumI/R1;
    ddl(6,6)=(n-sumI)/(2*R0^2)-sum((1-I1).*((y-x1(2:n+1)-m0).^2))/R0^3;
    ddl(7,7)=sumI/(2*R1^2)-sum(I1.*((y-x1(2:n+1)-m1).^2))/R1^3;
    ddl(4,6)=-1*dl(4)/R0;
    ddl(6,4)=ddl(4,6);
    ddl(5,7)=-1*dl(5)/R1;

```

```

    ddl(7,5)=ddl(5,7);

    term1=term1+ddl/Npar;
end

term2_trim=zeros(7,7);
for i=1:Npar
    kk=term2t(:,i)*term2t(:,i)';
    for j=1:49
        forterm2(i,j)=kk(fix((j-1)/7)+1,rem(j-1,7)+1);
    end
end
term2_temp=trimmean(forterm2,ptrim)
for i=1:7
    for j=1:7
        term2_trim(i,j)=term2_temp((i-1)*7+j);
    end
end

obInfo=-1*term1 -term2 + (term3a)*(term3a)';
obsInfot=-1*term1-term2_trim+(term3a)*(term3a)';
vparmt=inv(obsInfot);
vparmm=inv(obInfo)

```

## 8. Miscellaneous functions

### Random sampling (1)

```

function Newdata=Rsample1(data,weight,NofSample)
n=max(size(data));
re_ind=rand(1,NofSample);
cmwt=cumsum(weight)/sum(weight);
for k=1:NofSample
    st=(re_ind(k)>cmwt(1:n-1));
    Newdata(k)=data(sum(st)+1);
end
Newdata=Newdata';

```

### Random sampling (2)

```

function [Newdata1,Newdata2]=Rsample2(data1,data2, weight,NofSample)
n=max(size(data1));
re_ind=rand(1,NofSample);
cmwt=cumsum(weight)/sum(weight);
for k=1:NofSample
    st=(re_ind(k)>cmwt(1:n-1));
    Newdata1(k)=data1(sum(st)+1);
    Newdata2(k)=data2(sum(st)+1);
end
Newdata1=Newdata1';
Newdata2=Newdata2';

```

### Complete likelihood

```
function loglike=comp_loglik(x,I,y,theta,n)

phi=theta(1);
Q=theta(2);
m0=theta(3);
m1=theta(4);
R0=theta(5);
R1=theta(6);
pi=theta(7);
mu0=0;
Sig0=Q/(1-phi^2);

loglike=-0.5*(log(Sig0)+(x(1)-mu0)^2/Sig0) -0.5*(n*log(Q)+...
    (sum(x(2:n+1).^2)+(phi^2)*sum(x(1:n).^2) -2*phi*sum(x(1:n).*x(2:n+1)))/Q) +...
    sum(I(2:n+1))*log(pi)+(n-sum(I(2:n+1)))*log(1-pi) -...
    0.5* sum( I(2:n+1).* (log(R1)+((y-x(2:n+1)-m1).^2)/R1)+...
    (1-I(2:n+1)).*(log(R0)+((y-x(2:n+1)-m0).^2)/R0));
```

### Sample code for Data B : 10 % missing

```
size_y=size(y);
if size_y(1)>size_y(2)
    y=y';
end

% -----a and n
n=max(size(y)); %n:length of the data
a=~isnan(y);

%-----set initial parameter; Use mme or give some numbers
y_I=y(a>0);
Iparm=Iparmnorm(y_I)

Tol=0.1;
Miter=30;
Npar=500;

[Eparm1a,Vparm1a,RelLike1a,Tcal1a,Niter1a]=mainnormaux(y,a,Iparm,Tol,Miter,Npar,n,5);

Tol=0.05;
Miter=30;
Npar=500;

Iparm2a=Eparm1a(end,:);
[Eparm2a,Vparm2a,RelLike2a,Tcal2a,Niter2a]=mainnormaux(y,a,Iparm2a,Tol,Miter,Npar,n,5);

Tol=0.01;
Miter=20;
Npar=500;
Iparm3a=Eparm2a(end,:);
```

```
[Eparm3a,Vparm3a,RelLike3a,Tcal3a,Niter3a]=mainnormaux(y,a,Iparm3a,Tol,Miter,Npar,n,3);  
  
Tol=0.01;  
Miter=25;  
Npar=2000;  
  
Iparm4a=Eparm3a(end,:)  
[Eparm4a,Vparm4a,RelLike4a,Tcal4a,Niter4a]=mainnormaux(y,a,Iparm4a,Tol,Miter,Npar,n,2);
```