```
// T63mixPreStat.sce
// MIXTURE ESTIMATION AND PREDICTION (predictive, state-space)
// Experiments
// - change: simulated parameters, initial parameters, input signal,
//           type of proximity, length of prediction
// -------------------------------------------------------------------
exec("ScIntro.sce",-1),
getd(), mode(0)

nd=500;                                                                 // 1
np=3;                                                                   // 2
// PARAMETERS                                                           // 3
c(1).mS=[-.5 .1                         // component parametrs          // 4
         .2 .7];                                                        // 5
c(2).mS=[.6 .2                                                          // 6
         .1 -.4];                                                       // 7
c(1).nS=[1 -1]';                                                        // 8
c(2).nS=[-1 1]';                                                        // 9
c(1).aS=[.9 1.5];                                                       // 10
c(2).aS=[1.2 .5];                                                       // 11
c(1).bS=10;                                                             // 12
c(2).bS=-1;                                                             // 13
c(1).swS=[.5 .1                         // component covariances        // 14
```

```
           0 .2]*.1;
c(2).swS=[.1 .5
           0 .2]*.1;
c(1).svS=.3;
c(2).svS=.5;
alS=[.95 .05                            // parameters of pointer model
     .05 .95];
nx=size(c(1).mS,1);
nc=length(c);                           // number of components
xS(:,1)=zeros(nx,1);                    // initail state
cS=1;                                   // initial pointer
u=.5*signal(nd,2,0,4);                  // input

// SIMULATION
for t=2:nd
  jS=sampCat(alS(:,cS(t-1)));           // pointer value
  cS(t)=jS;                             // stor pointer value
  xS(:,t)=c(jS).mS*xS(:,t-1)+c(jS).nS*u(t)+c(jS).swS*randn(2,1);
  y(t)=c(jS).aS*xS(:,t)+c(jS).bS*u(t)+c(jS).svS*randn();
end

// INITIALIZATION
ka=[1 1];                               // initial counter
```

```
c(1).x=randn(nx,1);                     // initial state x1
c(2).x=randn(nx,1);                     // initial state x2
c(1).R=10*eye(nx,nx);                   // state estimate covariance
c(2).R=10*eye(nx,nx);                   // state estimate covariance
c(1).rw=c(1).swS*c(1).swS';             // state model noise
c(2).rw=c(2).swS*c(2).swS';             // state model noise
c(1).rv=c(1).svS^2;                     // output model noise
c(2).rv=c(2).svS^2;                     // output model noise

// TIME LOOP
x=zeros(2,nd);
for t=2:(nd-np)
  //proximity
  for j=1:nc
    [nill,nill,py,ry] = Kalman(c(j).x(:,t-1),y(t),u(t),..
                          c(j).mS,c(j).nS,[],c(j).aS,c(j).bS,[],..
                          c(j).rw,c(j).rv,c(j).R); // prediction
     select 2            // select type of prox.: 1-quadratic, 2-pdf
     case 1, lq(j)=-2*log(abs(y(t)-py));     // quadratic proximity
     case 2, [nill,lq(j)]=GaussN(y(t),py,ry);// pdf proximity
     end
  end
  // weights
```

```
q=exp(lq-max(lq));
w=q/sum(q);                           // weights
if 0, w=dDel(w); end   // 1=point estimates of weight
wt(:,t)=w;                            // remember weights
// estimation and prediction
xP(:,t+np)=zeros(nx,1);
yP(t+np)=0;
for j=1:nc
  // estimation within j-th component
  [c(j).x(:,t),c(j).R,c(j).yp(t)] = Kalman(c(j).x(:,t-1),y(t),u(t),..
                           c(j).mS,c(j).nS,[],c(j).aS,c(j).bS,[],..
                           c(j).rw,c(j).rv,c(j).R);
   c(j).x(:,t)=limit(c(j).x(:,t),5);  // limiting the state estimate
  xp=c(j).x(:,t);
  yp=c(j).yp(t);
  // prediction within j-th component
  for i=1:np
    [nill1,nill2,yp,nill4,xp,nill5]=.. // prediction only
    Kalman(xp,0,u(t+i),c(j).mS,c(j).nS,[],..
    c(j).aS,c(j).bS,[],c(j).rw,c(j).rv,c(j).R);
  end
  // mixture of state and output estimates from comonents
  xP(:,t+np)=xP(:,t+np)+w(j)*xp;      // resulting state estimate
```

```
    yP(t+np)=yP(t+np)+w(j)*yp;          // resulting prediction
  end
end

// RESULTS
s=(np+1):(nd-np);
tx=[’b’;’r’;’g’];
set(scf(1),’position’,[600 10 600 800]) // evolution of est. state
  title ’Simulated and estimated state’
for i=1:nx
  subplot(nx,1,i)
  plot(s,xS(i,s),s,xP(i,s))
  xlabel(’x’+string(i))
  legend(’xS’,’xP’);
end

set(scf(2),’position’,[1200 10 600 400])// output prediction
plot(s,y(s),s,yP(s))
legend(’y’,’yp’);
title ’Output and its prediction’

[nill,cp]=max(wt,’r’);                  // accuracy of classification
disp ’Accuracy of classification’
```

```
ACC=acc(cS(s),cp(s))                                                 // 107
                                                                     // 108
disp 'Relative prediction error of y'   // relative prediction error    // 109
RPE1=rpe(y(s),yP(s))                                                 // 110
```

About the example

This example is practically identical with T49mixDesStat.sce, where the mixture of state-space components is estimated. Here, a prediction of state and output estimates for arbitrary many steps ahead is added and can be set by the choice n_p.

As the state-space model is of the first order (in difference with a general regression model) the prediction is relatively simple. Just after estimation (calling full Kalman filter) the estimates of the state and output are copied into another variables (as the prediction doe not enter the loop of estimation - it goes blindly ahead without using new data) and these variables are predicted (the first part of Kalman filter) without any closed loop. In the end the values are remembered as final predictions.

Description of the program

- Rows 77–81 introduce the prediction.
  The new variables entering the prediction are x_p and y_p. They are predicted in rows 78–80 by the specially accommodated Kalman filter (instead of the filtered values of the state, its predicted value is used. The value of y_p is derived from x_p at each step of prediction.

Remark

For comparison the program for estimation is here T49mixDesStat.sce.