```
// T62mixExpNor.sce
// MIXTURE ESTIMATION (predictive normal)
// - dynamic normal componens
// - weight for initial information
// - multinomial y and v
//   |y1|    |a11 a12||y1|   |b11 b12 b13||v1|
//   |  | =  |       ||  | + |           ||v2| + e
//   |y2|    |a21 a22||y2|   |b21 b22 b23||v3|
//    new             old                 actual
// Experiments
// - change simulated expectations thS
// - change initial expectations thI (through the statistics S, ka)
// ----------------------------------------------------------------------
exec("ScIntro.sce",-1),
getd(), mode(0)

nd=500;                                                                 // 1
ni=1;                                   // weight for initial info.     // 2
// PARAMETERS                                                           // 3
sd=.01;                                 // std of the noise             // 4
aS=list(); bS=list(); kS=list();                                        // 5
aS(1)=[.7 -.2 .3 -.2                    //regression coefficients       // 6
       .1 .5  .1 .4];                                                   // 7
```

```
aS(2)=[.2 .4  .6 .2
      -.1 .7  .2 .1];
aS(3)=[.3 -.3 .4 .1
      -.2 .6  .1 .4];
bS(1)=[1; 3];
bS(2)=[8; 4];
bS(3)=[6; 5];
kS(1)=[1; -1];
kS(2)=[-1; 1];
kS(3)=[1; 1];
alS=[.7  2 .1
     .2 .5 .3
     .1 .3 .6];                      // parameters of pointer model
alS=[.9  .05 .05
     .05 .9  .05
     .05 .05 .9];                    // parameters of pointer model
y=zeros(2,4);
u=signal(nd,1);                      // definition of input signal
nc=length(aS);                       // number of components

// INITIALIZATION
V=list(); thI=list(); thE=list();
ka=[1 1 1]*ni;                       // initial counter
```

```
for j=1:nc
  thI(j)=[aS(j) bS(j) kS(j)]+.1*rand(2,6); // initial comp. parameters
  V(j)=eye(8,8);                          // initial sum statistics
  V(j)(1:2,3:$)=thI(j);
  V(j)(3:$,1:2)=thI(j)';
  V(j)=V(j)*ni;
  thE(j)=v2thN(V(j)/ka(j),2);             // initial point estimates
  c(j).th=thE(j)(:);                      // remember
end
ga=eye(nc,nc);                            // statistics for pointer model
alE=fnorm(ga,2);                          // point estimate
ws=fnorm(ones(nc,1));

// TIME LOOP
for t=3:nd
  // prediction
  ypp=0;
  fc=fnorm(alE*ws);                       // f(c(t-1)|d(t))
  ps=[y(:,t-1)' y(:,t-2)' u(t) 1]';
  for j=1:nc
    select 2        // type of prediction: 1-point, 2-generated;
    case 1, ypp=ypp+fc(j)*thE(j)'*ps;
    case 2, ypp=ypp+fc(j)*(thE(j)'*ps+sd*randn(2,1));
```

```
  end
end
yp(:,t)=ypp;                            // remember value of prediction

// simulation
jS=sampCat(alS);                        // pointer value
cS(t)=jS;
par=[aS(jS) bS(jS)];                    // parameters
reg=[y(:,t-1)' y(:,t-2)' u(t)'];        // data
y(:,t)=par*reg'+sd*randn(2,1);          // output generation

// estimation
for j=1:nc
  ps=[y(:,t-1)' y(:,t-2)' u(t-1)' 1]';
  [nill,qp(j)]=GaussN(y(:,t),thE(j)'*ps,sd*eye(2,2));
                                        // component proximity
end
q=exp(qp-max(qp)).*(alE*ws);            // full proximity
w=q/sum(q);                             // weights
for j=1:nc
  Ps=[y(:,t)' y(:,t-1)' y(:,t-2)' u(t)' 1]';
  V(j)=V(j)+w(j)*Ps*Ps';                // update of sum statistics
  ka(j)=ka(j)+w(j);                     // update of counter
```

```
    thE(j)=v2thN(V(j)/ka(j),2);          // point estimates
    c(j).th(:,t)=thE(j)(:);              // remember for plot
  end

  ga=ga+w*ws';                           // stat. update of poiner mod.
  alE=fnorm(ga,2);                       // pt estimate of pointer model
  ws=w;                                  // w -> old w
  wt(:,t)=w;                             // remember
end

// RESULTS
set(scf(1),'position',[600 10 1200 400])
for j=1:nc
  subplot(1,3,j)
  plot(c(j).th')
  title('Evolution of estimated parametrs: comp. '+string(j))
end

set(scf(2),'position',[900 500 600 400])  // plot data and prediction
plot(1:nd,y,'ob',1:nd,yp,'xr','markersize',8)
title 'Data (b) and prediction (r)'

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

```
disp 'Accuracy of classification'
ACC=acc(cS,cp)

disp 'Relative prediction error'          // accuracy of prediction
RPE=rpe(y(:),yp(:))
```

About the example

This example is is practically identical with the previous one `T61mixPreNor1.sce` with the difference that this one is multivariate. It also introduces dynamic models for both components and pointer model.

The component models are

$$\begin{bmatrix} y_{1;t} \\ y_{2;t} \end{bmatrix} = a_1 \begin{bmatrix} y_{1;t-1} \\ y_{2;t-1} \end{bmatrix} + a_2 \begin{bmatrix} y_{1;t-2} \\ y_{2;t-2} \end{bmatrix} + bu_t + \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} + \begin{bmatrix} e_{1;t} \\ e_{2;t} \end{bmatrix}$$

where a_1 and a_2 are matrices 2×2. The models are actually indexed by the component labels.

The component model is an ordinary dynamic categorical model.

The estimation uses prior knowledge derived from the actual parameters from simulation with added disturbance. The prior knowledge is imbedded with a given strength.

Description of the program

- Here,the description will be a bit more concise. For better information see the previous program `T61mixPreNor1.sce`.

- Rows 3–23 prepare parameters for simulation of components and th pointer.
- Row 25 defines the input signal.
- Rows 30–42 set initial statistics and point estimates of parameters.
- Rows 45–84 perform the time loop in which the prediction, simulation (data measurement) and estimation are called.
  - Rows 47–56 are prediction (point prediction or generated one can be selected).
  - Rows 59–63 perform simulation of new data y_t.
  - Rows 65–84 compute proximities of new data to components, construct weights w_t, recompute the statistics and evaluate the point estimates of parameters