```
// T61mixExpNor1.sce
// MIXTURE ESTIMATION (predictive normal)
// - scalar case of T62mixExpNor
// - dynamic normal componens
// - dynamic pointer
// - weight for initial information
// - scalar variables
//   y(t)=a1.y(t-1)+a2.y(t-2)+b.u(t)+e(t)
// Experiments: Change
// - simulated parameters thS
//   (including model order for both simulation and estimation)
// - initial parameters thI (through the statistics V, ka)
// - the course of input signal (smooth signal is bad for estimation)
// - type of prediction (either point or generated),
// - strength of initial information (through the statistics ka),
// - parameters of pointer model alS (the more is diagonal-like,
//   the longer are inervals between svitching)
// -------------------------------------------------------------------
exec("ScIntro.sce",-1),
getd(), mode(0)

nd=500;                                                                 // 1
ni=1;                                   // weight for initial inf.      // 2
```

```
// PARAMETERS                                                            // 3
aS=list();                                                               // 4
aS(1)=[.6 -.5]; bS(1)=.8; kS(1)=-1;      // regression coefficients      // 5
aS(2)=[.2 .4]; bS(2)=-1.8; kS(2)=1;      // regression coefficients      // 6
sd=.1;                                   // common and known noise std.  // 7
nc=length(aS);                           // number of components         // 8
alS=[.8 .1                               // parameter of pointer model   // 9
     .2 .9];                                                             // 10
y(1)=2; y(2)=-1; cS=ones(1,nc);          // initial conditions           // 11
u=signal(nd,1);                          // definition of input signal   // 12
                                                                         // 13
// INITIALIZATION                                                        // 14
V=list(); thI=list(); thE=list();                                        // 15
ka=ones(1,nc)*ni;                         // initial counter             // 16
for j=1:nc                                                               // 17
  thI(j)=[aS(j) bS(j) kS(j)]+1*rand(1,4); // initial parameters          // 18
  V(j)=eye(5,5);                         // initial sum statistics       // 19
  V(j)(1,2:5)=thI(j);                                                    // 20
  V(j)(2:5,1)=thI(j)';                                                   // 21
  V(j)=V(j)*ni;                                                          // 22
  thE(j)=v2thN(V(j)/ka(j),1);            //initial point estimates       // 23
  c(j).th(:,2)=thE(j)(:);                // remember for plot            // 24
end                                                                      // 25
```

```
ga=eye(nc,nc);                          // statistics for pointer model
alE=fnorm(ga,2);                        // point estimate
ws=fnorm(ones(nc,1));                   // old component weights

// 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]';           // regression vector
  for j=1:nc
    select 1           // 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());
    end
  end
  yp(:,t)=ypp;                          // prediction at time t

  // simulation
  jS=sampCat(alS,cS(t-1));              // pointer value
  cS(t)=jS;                             // remember
  y(t)=[aS(jS) bS(jS) kS(jS)]*[y(t-1) y(t-2) u(t) 1]'+sd*randn();
                                        // output generation
  // estimation
  for j=1:nc
    [nill,qp(j)]=GaussN(y(t),thE(j)'*ps,sd); // component proximity
  end
  q=exp(qp-max(qp)).*(alE*ws);          // overall proximity
  w=q/sum(q);                           // component weights
  wt(:,t)=w;                            // remember
  for j=1:nc
    Ps=[y(t) y(t-1) y(t-2) u(t)1]';     // extended regression vector
    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),1);         // pt estimate of components
    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,nc,j)
  plot(c(j).th(:,2:$)')
  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(:))

disp 'Simulated and estimated parameters - comp. 1'
disp([aS(1) bS(1) kS(1);thE(1)'])         // parameters of first comp.

disp 'Simulated and estimated parameters - comp. 2'
disp([aS(2) bS(2) kS(2);thE(2)'])         // parameters of second comp.
```

About the example

This example not only illustrates estimation of a mixture with scalar regression components but also introduces many new features. The most important is a possibility of prediction with estimated mixture. For it the mixture should have both dynamic model of pointer and dynamic models of components.

The components here are scalar regression models of the second order

$$y_t = a_{1j} y_{t-1} + a_{2j} y_{t-2} + b_j u_t + k_j + e_t$$

where j denotes a particular component.

The pointer model is $f\left(c_t | c_{t-1}, \alpha\right) = \alpha_{c_t | c_{t-1}}$ defined by the following table

$c_t \backslash c_{t-1}$	1	2
1	$\alpha_{1\|1}$	$\alpha_{1\|2}$
2	$\alpha_{2\|1}$	$\alpha_{2\|2}$

where $\alpha_{i|j}$ is a stationary probability of jumping into the ith component on condition that the system was in the component j.

A detailed explanation of how the prediction is constructed is given in Appendix XXX (predic).

Another feature used here is the construction of the initial statistics so that they would correspond to out prior knowledge of the estimated parameters. Also a parameter expressing the strength of out prior information is introduced. More detailed information about this topic is given in Appendix XXX (init)

The time loop of the program has several sections. They are prediction, simulation and estimation. They must be called in this order because at time t we predict output without knowledge of its real value, then we measure it (it is simulation) and with the new data we recompute the parameter estimates in estimation.

Description of the program

- Row 1–2 defines number of data and strength of prior information expressed as a number of initial data from which it has been constructed.
- Rows 4–12 concerns with simulation. The regression coefficient and noise standard deviation are defined, number of components n_c is determined, parameters of the pointer model is introduced, initial conditions are set and the input signal u is given for the whole interval of simulation.
- Rows 14–28 are devoted to initialization
  - At first, the statistics κ is defined. It determines the strength of initial knowledge through the parameter n_i.
  - Then, in the loop, the statistics V is constructed.
    - The initial guess of the parameters thI (row 18) is given as the true parameters aS, bS, kS disturbed by a noise. The magnitude of this noise simulates the error in our guess (and the necessity of the estimation algorithm to search for correct values of the parameters).
    - Rows 19–22 construct the statistics V according to the rules given in Appendix XXX
    - Row 23 constructs the point estimates of the parameters (regression coefficients).
      Remark: This row could be omitted because the statistics was constructed so that the parameters are are such as we defined is row 18.

  - Row 26 defines the initial statistics for the pointer model.
  - Row 27 constructs the point estimates of the pointer model parameters (normalization).
  - Row 28 sets the initial value of the weights which are in this case evolved during the simulation.

- Rows 31–68 form the actual substance of the estimation - the time loop in which prediction, simulation and estimation are repeatedly called for individual times of the whole task.

  - Rows 33–42 perform zero step prediction (see Appendix XXX).
    - Row 34 constructs stationary probability function of the pointer as a product of the pointer model and the old weighting vector w_{t-1} (this is a stationary prediction of the weights without a usage of the output y_t - it formally resembles the step of prediction in Kalman filtering).
    - Row 35 collects the regression vector for regression model.
    - Rows 38–39 give a choice of selection the type of prediction: either point prediction or simulated one.
    - Row 42 remembers the predicted value at time t.
  - Rows 45–47 perform simulation of the output value according to the pointer value generated at the row 45.
  - Rows 51–68 deal with estimation.
    - Rows 51–53 compute proximities of the measured data to individual components. As Gaussian model are very steep, the values of the components could be all practically zero. That is why they are computed in logarithm and only after a pre-normalization (subtraction of the maximal value) they are converted to their real values by exponentiation - rows 54–56. The they are normalized to the sum equal to one - row 55.

- The rest of this section performs a standard weighted update of regression models (components) statistics and computation of parameter estimates.

  Rows 64–65 do the same for the pointer model.
- Row 66 does the time shift: the current weights are renamed to old weights.