```
// 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
// - length of prediction np
// - 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
```

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

```
  c(j).th(:,2)=thE(j)(:);              // remember for plot            // 25
end                                                                   // 26
ga=eye(nc,nc);                         // statistics for pointer model // 27
alE=fnorm(ga,2);                       // point estimate               // 28
ws=fnorm(ones(nc,1));                  // old component weights        // 29
                                                                      // 30
// TIME LOOP                                                          // 31
for t=3:(nd-np)                                                       // 32
  // prediction                                                       // 33
  ypp=0;                                                              // 34
  fc=fnorm(alE^(np+1)*ws);             // f(c(t-1)|d(t))^(np+1)        // 35
  ps=[y(t-1) y(t-2) u(t) 1]';          // 1st regression vector        // 36
  for j=1:nc                                                          // 37
    yy=thE(j)'*ps;                     // 1st prediction (k=0)         // 38
    for k=1:np                         // loop for inner predictions   // 39
      ps=[yy; ps(1); u(t+k); 1];       // inner regression vector      // 40
      yy=thE(j)'*ps;                   // inner prediction             // 41
    end                                                               // 42
    ypp=ypp+fc(j)*yy;                  // mixture of final predictions // 43
  end                                                                 // 44
  yp(:,t)=ypp;                         // total prediction at time t   // 45
                                                                      // 46
  // simulation                                                       // 47
```

```
jS=sampCat(alS,cS(t-1));              // pointer value                // 48
cS(t)=jS;                             // remember                     // 49
y(t)=[aS(jS) bS(jS) kS(jS)]*[y(t-1) y(t-2) u(t) 1]'+sd*randn();       // 50
                                      // output generation            // 51
                                                                      // 52
// estimation                                                         // 53
for j=1:nc                                                            // 54
  [nill,qp(j)]=GaussN(y(t),thE(j)'*ps,sd); // component proximity     // 55
end                                                                   // 56
q=exp(qp-max(qp)).*(alE*ws);          // overall proximity            // 57
w=q/sum(q);                           // component weights            // 58
wt(:,t)=w;                            // remember                     // 59
for j=1:nc                                                            // 60
  Ps=[y(t) y(t-1) y(t-2) u(t)1]';     // extended regression vector   // 61
  V(j)=V(j)+w(j)*Ps*Ps';              // update of sum statistics     // 62
  ka(j)=ka(j)+w(j);                   // update of counter            // 63
  thE(j)=v2thN(V(j)/ka(j),1);         // pt estimate of components    // 64
  c(j).th(:,t)=thE(j)(:);             // remember for plot            // 65
end                                                                   // 66
ga=ga+w*ws';                          // stat. update of poiner mod.  // 67
alE=fnorm(ga,2);                      // pt estimate of pointer model // 68
ws=w;                                 // w -> old w                   // 69
wt(:,t)=w;                            // remember                     // 70
```

```
end                                                                    // 71
                                                                       // 72
// RESULTS                                                             // 73
set(scf(1),'position',[600 10 1200 400])                               // 74
for j=1:nc                                                             // 75
  subplot(1,nc,j)                                                      // 76
  plot(c(j).th(:,2:$)')                                                // 77
  title('Evolution of estimated parametrs: comp. '+string(j))          // 78
end                                                                    // 79
                                                                       // 80
set(scf(2),'position',[900 500 600 400])  // plot data and prediction   // 81
plot(1:(nd-np),y,'ob',1:(nd-np),yp,'xr','markersize',8)                // 82
title 'Data (b) and prediction (r)'                                    // 83
                                                                       // 84
[nill,cp]=max(wt,'r');                    // accuracy of classification // 85
disp 'Accuracy of classification'                                      // 86
ACC=acc(cS,cp)                                                         // 87
                                                                       // 88
disp 'Relative prediction error'          // accuracy of prediction    // 89
RPE=rpe(y(:),yp(:))                                                    // 90
                                                                       // 91
disp 'Simulated and estimated parameters - comp. 1'                    // 92
disp([aS(1) bS(1) kS(1);thE(1)'])         // parameters of first comp. // 93
```

```
                                                                    // 94
disp 'Simulated and estimated parameters - comp. 2'                 // 95
disp([aS(2) bS(2) kS(2);thE(2)'])         // parameters of second comp. // 96
```

About the example

This program is identical to that `T61mixPreNor1.sce` (scalar normal mixture) with the only difference that the prediction is multi-step. It length can be set by th parameters n_p. $n_p = 0$ means zero step prediction, etc.

Program description

Only the difference against `T61mixPreNor1.sce` is described.

- Row 2 defines th number of steps of the prediction.
- Row 35 computes the k-step prediction of the pointer (just the power of α)
- Rows 36 and 38 construct the first step of the multivariate prediction. This is based on real (measured data).
- Rows 39–42 represent the subsequent inner predictions the aim to the final one y_p
  - Row 40 represents the shifting regression vector corresponding to growing time approaching the final time of the prediction.

– Row 41 computes the inner predictions which are used in constructions of higher predictions instead of the data (which are nor at disposal at that time)
The procedure (for one component) goes like this

$$\hat{y_t} = a_1 y_{t-1} + a_2 y_{t-2} + u_t + k$$

$$\hat{y}_{t+1} = a_1 \hat{y}_t + a_2 y_{t-1} + u_{t+1} + k$$

$$\hat{y}_{t+2} = a_1 \hat{y}_{t+1} + a_2 \hat{y}_t + u_{t+2} + k$$

and then only on predictions