Comments to Simple mixtures estimation

Introduction to discrete model description

Let us have two-dimensional binary explanation variable $x_t = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. The model $f(x_1, x_2)$ can be written in two possible ways:

- **Without coding** the vector variable x - this way is instructive and easily created, however, it holds only for two-dimensional variables. The model (for binary variables) has the form

$x_1 \backslash x_2$	1	2
1	p_{11}	p_{12}
2	p_{21}	p_{22}

(1)

  where p_{ij} are probabilities $P([x_1, x_2] = [i, j])$ such that $\sum_i \sum_j p_{ij} = 1$, (it is joint probability function)
  Remark: This way is not in accord with our introduced notation connected with the model $f(y|x)$ which should be a table where the values of y go vertically and codes of the vector variable x go horizontally.

- **With coding** of the vector variable x - which is more complex but is in line with our notation. The model is

$[x_1, x_2]$	$[1,1]$	$[1,2]$	$[2,1]$	$[2,2]$
x	1	2	3	4
$f(x_1, x_2) = q_x$	$q_1 = p_{11}$	$q_2 = p_{12}$	$q_3 = p_{21}$	$q_4 = p_{22}$

(2)

Remark: Here, to be able to determine probability of $[x_1 = i, x_2 = j]$ we need first ti determine the coded index

$$k = 2(i-1) + j$$

e.g. for $i = 2$ and $j = 1$ it is $k = 2(2-1) + 1 = 3$ and then we can determine the probability q_3 which is p_{21}
and vice versa: having q_k we must compute the values i and j; $[x_1 = i, x_2 = j]$ as follows

$r = \frac{q_k}{2}$; $\tilde{r} = fix(r)$;
if $r == \tilde{r}$
  $j = 2;\ i = r$
else
  $j = 1; i = \tilde{r} + 1;$
end

For general cases (more variables) the coding and decoding can be performed by the functions

$$\begin{array}{ll} \text{coding:} & \texttt{z=xt2col(x,b)} \\ \text{decoding:} & \texttt{x=col2xt(z,b)} \end{array} \tag{3}$$

where z is the coded variable, $x = [x_1, x_2, \cdots x_n]$ are the original variables and b is the base, i.e. a vector of maxims of individual variables (which must start with the value 1)

$$b = max\,(x, 1)\,.$$

Discrete generators

There are several ways how to generate data from discrete models.

- **One binary variable**

  The command

  ```
  x = (rand(1,nd,'u')>p)+1
  ```

  generates nd random numbers 1 and 2 with probability of 1 equal to p.
  Generally (for one value)

  ```
  x(t) = sum(cumsum([p,1-p])<rand(1,1,'u'))+1
  ```

- **Model of multivariate variable** $x = [x_1, x_2]$

  1. For the model (1)

$x_1 \backslash x_2$	1	2
1	p_{11}	p_{12}
2	p_{21}	p_{22}

  with two variables $x = [x_1, x_2]'$ we need to factorize the joint probability function according to the chain rule

  $$f\,(x_1, x_2) = f\,(x_1)\,f\,(x_2|x_1)$$

  For example

$x_1 \backslash x_2$	1	2
1	0.2	0.1
2	0.4	0.3

  we have marginal

  $$f\,(x_1) = [0.6,\ 0,4]$$

  and conditional

$f(x_2\|x_1)$	1	2
1	0.33	0.25
2	0.67	0.75

If we denote the marginal vector by p and the conditional matrix by T, then the generator is

```
x(1,t) = sum(cumsum(p<rand(1,1,'u'))+1
x(2,t) = sum(cumsum(T(:,x(1,:))<rand(1,1,'u'))+1
```

2. For the model (2)

$[x_1, x_2] = z$	1	2	3	4
$f(z)$	p_1	p_2	p_3	p_4

we first need to generate the coded variable z

```
z = sum(cumsum(p)<rand(1,1,'u'))+1
```

and then to decode it (to determine the values of x_1 and x_2)

using the function `x=col2xt(z,b)` - see (3).

- **Conditional model** $f(y|x)$

This model for y is set by a conditional table (for two variables x and all binary)

x_1	1	1	2	2
x_2	1	2	1	2
$y = 1$	$th_{1\|11}$	$th_{1\|12}$	$th_{1\|21}$	$th_{1\|22}$
$y = 2$	$th_{2\|11}$	$th_{2\|12}$	$th_{2\|21}$	$th_{2\|22}$

and for x (for simplicity independent) by two probability vectors

$$f(x_1) = [p_1, p_2] \quad \text{and} \quad f(x_2) = [q_1, q_2]$$

The the generator is

```
x(1,t) = sum(cumsum(p<rand(1,1,'u'))+1
x(2,t) = sum(cumsum(q<rand(1,1,'u'))+1
       k=2(x(1,t)-1)+x(2,t)
y(t) = sum(cumsum(th(:,k)<rand(1,1,'u'))+1
```

where k is the number of the column of th corresponding to the values of $x_{1;t}$ and $x_{2;t}$.

Mixtures with continuous components

Program: `E15_mix_est_Cont.sce`

Standard mixture estimation in the simplest possible form. Two variables $x = [x_1, x_2]$; two components $j = 1, 2$ (variables are denoted by i, components by j)

Components

$$x_t = k_j + e_t = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} k_{1;j} \\ k_{2;j} \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

for $j = 1, 2$.

Statistics

$$\kappa = [\kappa_1, \kappa_2], \quad \text{counter}$$

$$S = [S_1, S_2], \quad \text{sum}$$

Update for component 1

$$S_{1;t} = S_{1;t} + w_1 \begin{bmatrix} x_{1;t} \\ x_{2;t} \end{bmatrix}, \quad \kappa_{1;t} = \kappa_{1;t-1} + w_1$$

and for component 2

$$S_{2;t} = S_{2;t} + w_2 \begin{bmatrix} x_{1;t} \\ x_{2;t} \end{bmatrix}, \quad \kappa_{2;t} = \kappa_{2;t-1} + w_2$$

i.e. S is a 2×2 matrix $[S_{i,j}]$ with variables in rows and components in columns.

Parameter estimate

$$m_j = \begin{bmatrix} \hat{k_1} \\ \hat{k_2} \end{bmatrix} = \frac{S_j}{\kappa_j}, \quad \text{expectation}$$

for $j = 1, 2$.

Initiation for prior parameter m_0

$\cdots$ set κ_0 (strength of initial information) and then

$$S_0 = m_0 \kappa_0$$

Weights for measured $x_t = \xi$

$$w_j \propto f_j(x_t = \xi)$$

where f_j is model of the j-the component and x_t is the current data record.

Classification

$$cp_t = \arg\max_{w_j}(w)$$

```
// Classification based on mixture estimation
// - on-line estimation
// - normal models  y(t)=k+e(t)
// -------------------------------------
clc, clear, close(winsid()), mode(0)                                  // 1
getd();                                                               // 2
                                                                      // 3
nd=100;                                                               // 4
// SIMULATION                                                         // 5
pS=[.4 .6];                                                           // 6
// c=1 =2                              components in columns          // 7
kS=[-1 1       // x1                    variables in rows             // 8
    -2 3];     // x2                                                  // 9
for t=1:nd                                                            // 10
  c(t)=sum(cumsum(pS)<rand(1,1,'u'))+1;                               // 11
  x(:,t)=kS(:,c(t))+.1*rand(1,1,'n');                                 // 12
end                                                                   // 13
                                                                      // 14
// ESTIMATION                                                         // 15
// c=1 =2                                                             // 16
m=[-2 5        // x1                                                  // 17
   -1 3];      // x2                    // prior parameters           // 18
ka=[1 1]';                               // prior counter              // 19
for j=1:2                                                             // 20
  S(:,j)=m(:,j)*ka(j);                   // prior statistics           // 21
end                                                                   // 22
                                                                      // 23
// time loop                                                          // 24
for t=1:nd                                                            // 25
  for j=1:2                                                           // 26
    q(j)=GaussN(x(:,t),m(:,j),.1*eye(2,2)); // proximity              // 27
  end                                          //  = fj(x=xt)         // 28
  w=q/sum(q);                            // weights                   // 29
  wt(:,t)=w;                                                          // 30
  for j=1:2                                                           // 31
    S(:,j)=S(:,j)+w(j)*x(:,t);           // statistics                // 32
    ka(j)=ka(j)+w(j);                    //  update                   // 33
```

```
    m(:,j)=S(:,j)/ka(j);              // parameters                 // 34
  end                                                              // 35
  [nill,cp(t)]=max(w);                // CLASSIFICATION             // 36
end                                                                // 37
                                                                   // 38
// ACCURACY of classification                                      // 39
acc=sum(c==cp)/nd                                                  // 40
```

Notation in the program:

- c - pointer
- x - data
- pS - switching parameters
- kS - parameters of the components (components in columns)
- m - parameter estimates (components in columns)
- ka - counter statistics
- S - sum statistics
- q - proximity
- w - weights
- wt - storing the weights for plot
- cp - pointer estimate

Description of the program:

- rows 3-9: parameters for simulation
- rows 10-13: simulation (fist pointer, then data from c(t)-th component)
- rows 16-22: initial parameters and statistics statistics
- rows 24-37: time loop
  - 26-28: proximities
  - 29: normalization giving weights
  - 32-33: update of statistics
  - 34: recomputation of parameter estimates
- row 36 - classification (argument of maxima of the weights)

Mixtures with discrete components: basic estimation

Program: `E16_mix_est_Disc.sce`

Standard mixture estimation with discrete categorical components.

Simulation of two-dimensional variable $x = [x_1, x_2]'$ with independent variables (just for simplicity) described by components $f(x_i|c)$ where c is the pointer

$x_i \backslash c$	1	2
1	$p_{1\|1}$	$p_{1\|2}$
2	$p_{2\|1}$	$p_{2\|2}$

, $i = 1, 2$

with the pointer model $f(c)$

c	1	2
$f(c)$	q_1	q_2

Estimation is performed with the coded form of components $f(x_1, x_2|c)$

$[x_1, x_2] \to z$	$c = 1$	$c = 2$
1	$p_{1\|1}$	$p_{1\|2}$
2	$p_{2\|1}$	$p_{2\|2}$
3	$p_{3\|1}$	$p_{3\|2}$
4	$p_{4\|1}$	$p_{4\|2}$

and uniform pointer model.[1]

The **statistics** have the same form as the component model and they are initialized in various options (rows 23-26 in the program):

1. random parameters,
2. true parameters from the simulation + noise,
3. true parameters (without noise)

Through the counter ka (row 27) we can set the strength of out prior belief.

```
// Classification based on mixture estimation
// - on-line estimation
// - discrete models  f(x1,x2|c)
```

[1] If the pointer model is set as uniform, it does not appear in the procedure of estimation. As the most information about classes comes from the components, the assumption of the pointer model uniformity is well acceptable.

```
// -------------------------------------
clc, clear, close(winsid()), mode(0)                                    // 1
getd();                                                                 // 2
                                                                        // 3
nd=100;                                                                 // 4
// SIMULATION                                                           // 5
pS=[.4 .6];                                                             // 6
//      c =1 =2                                                         // 7
thS(1).x=[.03 .98      // x1                                            // 8
          .97 .02];                                                     // 9
thS(2).x=[.99 .03      // x2                                            // 10
          .01 .97];                                                     // 11
for t=1:nd                                                              // 12
  c(t)=sum(cumsum(pS)<rand(1,1,'u'))+1;                                 // 13
  for i=1:2                                                             // 14
    x(i,t)=sum(cumsum(thS(i).x(:,c(t)))<rand(1,1,'u'))+1;               // 15
  end                                                                   // 16
end                                                                     // 17
                                                                        // 18
// ESTIMATION                                                           // 19
thI=[pmult(thS(1).x(:,1)',thS(2).x(:,1)')'..                            // 20
     pmult(thS(1).x(:,2)',thS(2).x(:,2)')'];                            // 21
select 2  // 1=random, 2=with noise, 3=precise  init. parameters       // 22
case 1, th=rand(4,2,'u');                                               // 23
case 2, th=fnorm(thI+.1*(rand(4,2)-.5),1);                              // 24
case 3, th=thI;                                                         // 25
end                                                                     // 26
ka=[1 1];                                 // prior counter              // 27
S=th.*(ones(4,1)*ka);                                                   // 28
                                                                        // 29
// time loop                                                            // 30
for t=1:nd                                                              // 31
  for j=1:2                                                             // 32
    k=2*(x(1,t)-1)+x(2,t);                                              // 33
    q(j)=th(k,j); // proximity = fj(x=xt)                               // 34
  end                                                                   // 35
  w=q/sum(q);                              // weights                  // 36
  wt(:,t)=w;                                                            // 37
  for j=1:2                                                             // 38
    k=2*(x(1,t)-1)+x(2,t);                                              // 39
```

```
    S(k,j)=S(k,j)+w(j);          // statistics                    // 40
    ka(j)=ka(j)+w(j);                    //  update               // 41
    th(:,j)=fnorm(S(:,j)/sum(S(:,j)));                            // 42
  end                                                             // 43
  th1(:,t)=th(:,1); th2(:,t)=th(:,2);                             // 44
  [nill,cp(t)]=max(w);                   // CLASSIFICATION        // 45
end                                                               // 46
                                                                  // 47
// ACCURACY of classification                                     // 48
disp 'Accuracy'                                                   // 49
acc=sum(c==cp)/nd                                                 // 50
disp 'Simulated parametrs'                                        // 51
thI                                                               // 52
disp 'Parameter estimates'                                        // 53
thE=fix(th*10000)/10000                                           // 54
                                                                  // 55
set(scf(),'position',[800 20 600 600]);                           // 56
subplot(211), plot(th1')                                          // 57
title 'Evolution of the estimates - first component'              // 58
subplot(212), plot(th2')                                          // 59
title 'Evolution of the estimates - second component'             // 60
```

Notation

- c - pointer
- x - data
- pS - switching parameters
- thS - parameters of the components (components in columns)
- thI - collection of parameters from simulation (in the coded form)
- th - parameter estimates (components in columns)
- ka - counter statistics
- S - sum statistics
- q - proximity
- w - weights
- wt - storing the weights for plot

- cp - pointer estimate

Description

- row 6: simulation parameter for component switching
- rows 8-10: simulation parameters of components
- rows 12-17: simulation
- rows 30-47: time loop
  - 33-34: proximity
  - 36: weights
  - 39-41: statistics update (in the coded form)
  - 42: point estimates of component parameters
- row 45: classification (argument of maxima of the weights)

Mixtures with discrete components: initialization of estimation

In this program, the estimation is initialized by preliminary measured initial data. Two ways of initialization are offered:

1. With the knowledge of only initial data and no knowledge about component switching (the pointer values),
2. With the full initial knowledge of both the data and the pointer values.

Program `E16_mix_est_Disc2.sce`

The program is practically identical with the previous one. It differs only in the initialization part (rows 22-41 in the program).

The first option uses only prior data from which the frequency table is constructed. In the example solved, it is the matrix 2×2 with the frequencies of the combinations of values of $[x_1, x_2] = 11, 12, 21, 22$. Here, we determine the maximum frequency and use it as dominant with other randomly generated small frequencies as the first component. Then we do the same with the second maximum frequency.

The second option (which gives much better results, however, it needs not only prior data but also prior values of the pointer). Here, the initial data are divided into groups according o the pointer value and data from each group are used for estimation of the corresponding component.

Then, the prior parameters are recomputed to the coded form, using the function `pmult()`which does nothing but multiplication of binomials in the indexes: i.e. entries of the matrix a_{ij} recomputes to he vector $[a_{11}, a_{12}, a_{21}, a_{22}]$. See the transition between models (1) and (2).

```
// Classification based on mixture estimation
// - on-line estimation
// - discrete models  f(x1,x2|c)
// - initial data
// -------------------------------------
clc, clear, close(winsid()), mode(0)                                  // 1
getd();                                                               // 2
                                                                      // 3
nd=300; ni=100;                                                       // 4
// SIMULATION                                                         // 5
pS=[.4 .6];                                                           // 6
//      c =1 =2                                                       // 7
thS(1).x=[.03 .98      // x1                                          // 8
          .97 .02];                                                   // 9
```

```
thS(2).x=[.99 .03      // x2                                          // 10
          .01 .97];                                                  // 11
for t=1:(ni+nd)                                                      // 12
  c(t)=sum(cumsum(pS)<rand(1,1,'u'))+1;                              // 13
  for i=1:2                                                          // 14
    x(i,t)=sum(cumsum(thS(i).x(:,c(t)))<rand(1,1,'u'))+1;            // 15
  end                                                                // 16
end                                                                  // 17
                                                                     // 18
// ESTIMATION                                                        // 19
xi=x(:,(nd+1):(nd+ni));                                              // 20
ci=c((nd+1):(nd+ni));                                                // 21
select 1       // 1=without, 2=with  the knowledge of c              // 22
case 1                                                               // 23
  T=tab(xi);                                                         // 24
  [nill,n1]=max(T);                                                  // 25
  T(n1(1),n1(2))=0;                                                  // 26
  t1=rand(2,2,'u');                                                  // 27
  t1(n1(1),n1(2))=15;                                                // 28
  t1=fnorm(t1,1);                                                    // 29
  [nill,n2]=max(T);                                                  // 30
  t2=rand(2,2,'u');                                                  // 31
  t2(n2(1),n2(2))=15;                                                // 32
  t2=fnorm(t2,1);                                                    // 33
case 2                                                               // 34
  t1=zeros(2,2);                                                     // 35
  s1=find(ci==1);                                                    // 36
  t1=fnorm(tab(xi(:,s1),2,2)+1e-6,1);                                // 37
  t2=zeros(2,2);                                                     // 38
  s2=find(ci==2);                                                    // 39
  t2=fnorm(tab(xi(:,s2),2,2)+1e-6,1);                                // 40
end                                                                  // 41
thI=[pmult(t1(:,1)',t2(:,1)')'..                                     // 42
    pmult(t1(:,2)',t2(:,2)')'];                                      // 43
th=thI;                                                              // 44
ka=[1 1];                                    // prior counter        // 45
S=th.*(ones(4,1)*ka);                                                // 46
                                                                     // 47
// time loop                                                         // 48
for t=1:nd                                                           // 49
```

```
  for j=1:2                                                    // 50
    k=2*(x(1,t)-1)+x(2,t);                                     // 51
    q(j)=th(k,j); // proximity = fj(x=xt)                      // 52
  end                                                          // 53
  w=q/sum(q);                                 // weights       // 54
  wt(:,t)=w;                                                   // 55
  for j=1:2                                                    // 56
    k=2*(x(1,t)-1)+x(2,t);                                     // 57
    S(k,j)=S(k,j)+w(j);                       // statistics    // 58
    ka(j)=ka(j)+w(j);                         //  update       // 59
    th(:,j)=fnorm(S(:,j)/sum(S(:,j)));                         // 60
  end                                                          // 61
  th1(:,t)=th(:,1); th2(:,t)=th(:,2);                          // 62
  [nill,cp(t)]=max(w);                        // CLASSIFICATION // 63
end                                                            // 64
                                                               // 65
// ACCURACY of classification                                  // 66
disp 'Accuracy'                                                // 67
s=1:nd;                                                        // 68
c=c(s); cp=cp(s);                                              // 69
h=c2c(c,cp);                                  // reordering of comps // 70
acc=sum(c==h(cp))/nd                                           // 71
disp 'Simulated parametrs'                                     // 72
thI                                                            // 73
disp 'Parameter estimates'                                     // 74
thE=fix(th*10000)/10000                                        // 75
                                                               // 76
set(scf(),'position',[800 20 600 600]);                        // 77
subplot(211), plot(th1')                                       // 78
title 'Evolution of the estimates - first component'           // 79
set(gca(),'data_bounds',[1 nd -.1 1.1])                        // 80
subplot(212), plot(th2')                                       // 81
title 'Evolution of the estimates - second component'          // 82
set(gca(),'data_bounds',[1 nd -.1 1.1])                        // 83
```

Mixtures with discrete components: independent explanatory variables

The categorical models generally suffer from extremely high dimension (for more variables with more values). A great relief can be reached, if the explanatory variables are treated as indepen-

dent. The simplification is done with the formula

$$f(x_1, x_2) \underbrace{=}_{\text{indep.}} f(x_1) f(x_2)$$

E.e. for ten variables each with ten values the full model has dimension 10^{10} while for independent variables it is only $10 \times 10 = 100$.

The program is similar to the previous ones with the only differences:

1. Parameters and statistics are defined for each variable separately - each variable has its own independent univariate model. See rows 23-40 of the program.
2. The proximities are also evaluated for each scalar variable and the multivariate one is given as their product. Rows 46-51.
3. Update of the statistics and construction of the point estimates are also performed for each variable separately. Rows 54-62.

```
// Classification based on mixture estimation
// - on-line estimation
// - discrete models
// - independent expl. vars f(x1,x2|c)=f(x1|c)f(x2|c)
// - initial data
// -------------------------------------
clc, clear, close(winsid()), mode(0)                                    // 1
getd();                                                                 // 2
                                                                        // 3
ni=100; nd=300;     // munb. of initial and measured data               // 4
// SIMULATION                                                           // 5
pS=[.4 .6];                                                             // 6
//      c =1 =2                                                         // 7
thS(1).x=[.03 .98      // x1                                            // 8
          .97 .02];                                                     // 9
thS(2).x=[.99 .03      // x2                                            // 10
          .01 .97];                                                     // 11
for t=1:(ni+nd)                                                         // 12
  c(t)=sum(cumsum(pS)<rand(1,1,'u'))+1;                                 // 13
  for i=1:2                                                             // 14
    x(i,t)=sum(cumsum(thS(i).x(:,c(t)))<rand(1,1,'u'))+1;               // 15
  end                                                                   // 16
end                                                                     // 17
```

```
                                                                    // 18
// ESTIMATION                                                       // 19
xi=x(:,(nd+1):(nd+ni));                                             // 20
ci=c((nd+1):(nd+ni));                                               // 21
fx=list(); S=list();                                                // 22
select 1       // 1=without, 2=with  the knowledge of c             // 23
case 1                                                              // 24
  T=tab(xi);                                                        // 25
  [nill,n1]=max(T);                                                 // 26
  T(n1(1),n1(2))=0;                                                 // 27
  fx(1)=rand(2,2,'u'); fx(1)(n1(1),n1(2))=15;                       // 28
  fx(1)=fnorm(fx(1),1);                                             // 29
  [nill,n2]=max(T);                                                 // 30
  fx(2)=rand(2,2,'u'); fx(2)(n2(1),n2(2))=15;                       // 31
  fx(2)=fnorm(fx(2),1);                                             // 32
case 2                                                              // 33
  fx(1)=zeros(2,2);                                                 // 34
  s1=find(ci==1);                                                   // 35
  fx(1)=fnorm(tab(xi(:,s1),2,2)+1e-6,1);                            // 36
  fx(2)=zeros(2,2);                                                 // 37
  s2=find(ci==2);                                                   // 38
  fx(2)=fnorm(tab(xi(:,s2),2,2)+1e-6,1);                            // 39
end                                                                 // 40
ka=[1 1];                                          // prior counter  // 41
S(1)=fx(1)*ka(1); S(2)=fx(2)*ka(2);                                 // 42
                                                                    // 43
// time loop                                                        // 44
for t=1:nd                                                          // 45
  for j=1:2                                                         // 46
    q(j)=1;                                                         // 47
    for i=1:2                                                       // 48
      q(j)=q(j)*fx(i)(x(i,t),j); // proximity = fj(xi=xti)          // 49
    end                                                             // 50
  end                                                               // 51
  w=q/sum(q);                                      // weights      // 52
  wt(:,t)=w;                                                        // 53
  for j=1:2                                                         // 54
    for i=1:2                                                       // 55
      S(i)(x(i,t),j)=S(i)(x(i,t),j)+w(j);          // statistics    // 56
      ka(j)=ka(j)+w(j);                            //  update       // 57
```

```
      fx(i)(:,j)=fnorm(S(i)(:,j)/sum(S(i)(:,j))); // scalar estimates // 58
    end                                                              // 59
  end                                                                // 60
  fx1t(:,t)=fx(1)(:,1);                          // joint             // 61
  fx2t(:,t)=fx(2)(:,1);                          //   estimates       // 62
  [nill,cp(t)]=max(w);   // CLASSIFICATION                          // 63
end                                                                  // 64
                                                                     // 65
// ACCURACY of classification                                        // 66
disp 'Accuracy'                                                      // 67
s=1:nd;                                                              // 68
c=c(s); cp=cp(s);                                                    // 69
h=c2c(c,cp);                                 // reordering of comps  // 70
acc=sum(c==h(cp))/nd                                                 // 71
                                                                     // 72
disp 'Simulated parametrs'                                           // 73
thS(1).x, thS(2).x                                                   // 74
disp 'Parameter estimates'                                           // 75
fix(fx(1)*100)/100                                                   // 76
fix(fx(2)*100)/100                                                   // 77
                                                                     // 78
set(scf(),'position',[800 20 600 600]);                              // 79
subplot(211), plot(fx1t')                                            // 80
title 'Evolution of the estimates - first component'                 // 81
set(gca(),'data_bounds',[1 nd -.1 1.1])                              // 82
subplot(212), plot(fx2t')                                            // 83
title 'Evolution of the estimates - second component'                // 84
set(gca(),'data_bounds',[1 nd -.1 1.1])                              // 85
```