Lectures on

Stochastic systems

Contents

1 Introduction, probability, system **4**

1.1 Revision of basic probability notions 4

1.2 System and its variables 8

1.3 Model of the system 8

2 Differential equations **10**

2.1 Differential equations 10

2.2 Discretization 11

3 Regression model **11**

3.1 Regression model in a state-space form 14

4 Discrete and logistic models **18**

4.1 Discrete model 18

4.2 Logistic model 23

5 Bayesian estimation **26**

5.1 Estimation with Bayes rule 26

6 Estimation of specific models **30**

6.1 Recursive estimation of normal regression model 30

6.2 Batch estimation of regression model 33

6.3 Categorical model 34

6.4 Model of logistic regression 39

7 Prediction of model output **40**

7.1 Output estimation (zero step prediction) . . . 40

7.2 One step prediction . . . 43

7.3 Multi-step prediction . . . 44

7.4 Prediction with discrete model . . . 46

8 State-space model, state estimation **50**

8.1 Model . . . 50

8.2 Estimation . . . 50

9 Nonlinear state estimation **54**

9.1 Nonlinear model . . . 54

9.2 Model with unknown parameters . . . 56

10 Control with regression model **58**

10.1 Derivation of the control in pdf . . . 58

10.2 Derivation for normal regression model . . . 60

11 Control with categorical model **65**

11.1 Optimization . . . 65

11.2 Application . . . 67

12 Adaptive control **68**

13 Appendix **68**

13.1 Elementary differential equations . . . 68

13.2 Elementary difference equations . . . 70

13.3 Discretization of a continuous model . . . 72

13.4 Point estimate with quadratic criterion . . . 74

13.5 Minimization of the control criterion criterion . . . 75

14 Introduction to Scilab **77**

15 Programs for exercises **82**

15.1 Simulation with regression model 83
15.2 Simulation with discrete model 86
15.3 Simulation with state-space model 88
15.4 Least squares estimation 90
15.5 Estimation with continuous model 93
15.6 Estimation with discrete model 101
15.7 Prediction with continuous model 104
15.8 Adaptive on-line prediction with continuous model 107
15.9 Prediction with discrete model 114
15.10 Adaptive prediction with discrete model 116
15.11 Adaptive on-line prediction with discrete model 119
15.12 State estimation 122
15.13 Noise filtration 125
15.14 Control with regression model 128
15.15 Adaptive control with regression model 131
15.16 Control with discrete model 135

16 Supporting subroutines **138**

16.1 Simulation of discrete data 138
16.2 Kalman filter 139
16.3 Coding of discrete variables 140

1 Introduction, probability, system

All necessary for study can be found on the web:

www.fd.cvut.cz/personal/nagyivan+ Mat. Models and Applications

Here you can find:

- For lectures
  - Text for study (both in English and Czech)
    It should be ready for self-study. If not, please contact me on mail ibp.nagy@gmail.com or by phone 739 081 050.
  - Schedule of lectures
    It is approximate and can be changed (e.g. according to your wish).
- For exercises
  - Basic programs
    They should be programmed during exercises and they cover the basic tasks of this course.
  - Other programs
    Here are programs to other tasks that could be interesting for you.
- Help to Scilab
  - Introduction to Probability (for repetition of what has been forgotten)
  - Introduction: brief (7 pages), detailed (87 pages)
  - Manual for beginners
- Other programs which can be directly used
  Here are more programs concerning the basic tasks as well as other ones relating the topic.
- Home Page of Scilab where it can be downloaded.
  This is the Home Page of the Scilab program. Here the latest version of Scilab can be downloaded for free. Other useful pages can also be reached in references.

1.1 Revision of basic probability notions

If necessary, the basic notions from Probability can be obtained from the webpage

https://people.smp.uq.edu.au/DirkKroese/asitp.pdf

or its copy from our webpage

https://www.fd.cvut.cz/personal/nagyivan/StochSyst/IntroToProbability.pdf.

- **Variable** is anything which can be assigned values. The values can be measured and they form measured data or they cannot be measured (e.g. due to nonexistence or too high price of some measuring device).

- **Random variable** is a variable for which even in the same conditions different values are measured.

  REMARK: Random variables are (i) ***continuous*** - with values form real axis, (ii) ***discrete*** - with finite or countably many different values.

- **Distribution** gives a full description of random variable.

  - Discrete ***probability function***: is given by the following formula

    $$f(x) \equiv P(X = x),$$

    i.e. it assigns probabilities to its values.
  - Continuous ***density function***: is given by the formula

    $$f(x) = \frac{dF(x)}{dx} \text{ or } F(x) = \int_{-\infty}^{x} f(t)\, dt$$

    where $F(x) = P(X \leq x)$ is distribution function.

- **Random vector** is a column vector of random variables

  $$x = [x_1, x_2, \cdots, x_n]'$$

  where $'$ denotes transposition.

- **Joint distribution** (for two variables x_1 and x_2) is $f(x_1, x_2)$ and it contains all information not only of x_1 and x_2 separately but also about their mutual relation.

- **Marginal distribution** is $f(x_1)$ or $f(x_2)$ and it contains information only about tj respective variable, the other one being unknown. It holds

  $$f(x_1) = \int_{-\infty}^{\infty} f(x_1, x_2)\, dx_2$$

  and similarly for x_2.

- **Conditional distribution** is distribution of one variable if we know the value of the other one. It holds

  $$f(x_2|x_1) = \frac{f(x_1, x_2)}{f(x_1)}$$

  and similarly for $x_1|x_2$.

- **Independence** of random variables x_1 and x_2 is given by

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

- **Expectation** of random variable
  – discrete

$$E[X] = \sum_{x_i \in X} x_i f(x_i)$$

  – continuous

$$E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

- **Variance**

$$D[X] = E\left[(X - E[X])^2\right]$$

- **Covariance**

$$C[X_1, X_2] = E[(X_1 - E[X_1])(X_2 - E[X_2])]$$

- **Expectation** of random vector

$$E[X] = \begin{bmatrix} E[x_1] \\ E[x_2] \end{bmatrix}, \quad C[X] = \begin{bmatrix} D[x_1] & \mathbf{cov}[x_1, x_2] \\ \mathbf{cov}[x_1, x_2] & D[x_2] \end{bmatrix}$$

- **Random process** is random variable indexed by time

time \ values	discrete	continuous
discrete	Markov chains	random sequences
continuous	queues	x

- **Categorical distribution**

x	1	2	$\cdots$	n
$f(x)$	p_1	p_2	$\cdots$	p_n

  where $p_1 \geq 0$, $\sum p_i = 1$. Each realization has its probability.

- **Normal distribution**

$$f(X) = \frac{1}{\sqrt{(2\pi)^n |R|}} \exp\left\{-\frac{1}{2}(x - \mu)' R^{-1} (x - \mu)\right\}$$

Example

For the distribution $f(x_1, x_2)$ determine marginal and conditional ones.

– *For discrete case*

$f(x_1, x_2)$

$x_1 \backslash x_2$	1	2	$f(x_1)$
1	0.1	0.3	0.4
2	0.4	0.2	0.6
$f(x_2)$	0.5	0.5	

$f(x_2\|x_1)$	
$\frac{1}{4}$	$\frac{3}{4}$
$\frac{2}{3}$	$\frac{1}{3}$

$f(x_1\|x_2)$	
$\frac{1}{5}$	$\frac{3}{5}$
$\frac{4}{5}$	$\frac{2}{5}$

$f(x_1)$	$f(x_2)$
0.2	0.2
0.3	0.3

Here, the joint distribution is introduced as the matrix $\begin{bmatrix} 0.1 & 0.3 \\ 0.4 & 0.2 \end{bmatrix}$ (sum of all entries must be 1). Below this matrix the marginal $f(x_2)$ is listed with entries as sums of the matrix in columns and right to the matrix is the marginal $f(x_1)$ as a sum of the matrix in rows.

Below-left is the conditional $f(x_1|x_2)$ for both $x_2 = 1$ and $x_2 = 2$ (in one matrix) and similarly right is the conditional $f(x_2|x_1)$, again for $x_1 = 1$ and $x_1 = 2$ in one matrix.

Down-right is the product of marginals. It is not equal to the joint probability, so the variables are not independent.

– *For continuous case*

For given joint density function, determine marginals, conditional distributions and decide, if the random variables are independent.

– joint density

$$f(x_1, x_2) = 6x_1^2 x_2, \;\; x_1, x_2 \in (0,\, 1)$$

– marginal

$$f(x_1) = \int_0^1 6x_1^2 x_2 dx_2 = 3x_1^2$$

$$f(x_2) = \int_0^1 6x_1^2 x_2 dx_1 = 2x_2$$

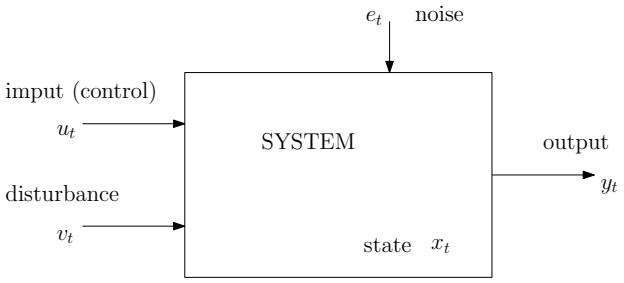
– conditional

$$f(x_1|x_2) = \frac{6x_1^2x_2}{2x_2} = 3x_1^2$$
$$f(x_2|x_1) = \frac{6x_1^2x_2}{3x_1^2} = 2x_2$$

As it is $f(x_1, x_2) = f(x_1) f(x_2)$ the variables are independent.

1.2 System and its variables

System is a part of reality we are interested in, on which we measure data and which we want to learn about to be able to predict its behavior or to influence it by control. It can be schematically demonstrated in the following picture



- Output - is the modeled variable. It is unknown at prior but after application of a control it can be measured.
- Input - is some variable that influences the output. It can be fully manipulated by us as a control variable).
- Disturbance - is another variable influencing the output. It can be measured, but we cannot influence it.
- State - is hidden variable. It is influenced by the input and it influences the output. It cannot be measured and mostly is estimated.
- Noise - is a disturbance that influences the system. It can be neither measured nor predicted

1.3 Model of the system

It is a mathematical equation based on the system variables that expresses the output (or state) by means of other variables and model parameters. The model with parameters which are not determined express the structure of the system. The values of the parameters are determined on the basis of measured data. With correct values of the parameters the output measured and that produced by th model (output prediction) should be similar (up to the deviance caused by th noise).

Examples

- **A controlled crossroads**.
  (state estimation)
  Input - the proportion of green in the signal lights.
  Disturbance - the traffic intensity of vehicles coming to the crossroads.
  State - the queue lengths in the crossroads arms.
  Output - the intensity of vehicles going out of the arms of the crossroads.
  Parameters - saturated flow in the crossroads arms, directional relations in the crossroads.

- **Automatic control of a vehicle.**
  (control)
  Input - angle of the gas pedal.
  Disturbance - wind, turnings, slope of the road etc.
  Output - speed of the vehicle.
  Parameters - express bounds between output and other variables. They could be deduced from th construction parameters of the vehicle or estimated from measured data.

- **Traffic accidents in some area or town**.
  (classification)
  Disturbance - characteristics of situation during the accident (light, weather, $\cdots$), the environment in the place of accident (type or surface of the road, view conditions $\cdots$) or behavior of the the vehicle (speed, power of engine $\cdots$).
  Output - seriousness of the accident (damage, injury, death)
  Parameters - estimated probabilities of the type of accident.

2 Differential equations

2.1 Differential equations

Dynamic process is described by a differential equation. Here we will repeat basic notions concerning **stationary** (constant coefficients) and **homogeneous** (zero right-hand side) equations of the first and second order.

First order

The equation with the initial condition is ($'$ denotes derivative)

$$y' + ay = 0, \;\; y(0) = y_0$$

- The solution by Laplace transformation

$$pY - y_0 + aY = 0$$

$$(p + a) Y = y_0$$

$$Y = y_0 \frac{1}{p + a} \rightarrow y(t) = y_0 \exp\{-at\}$$

- The solution in time domain using characteristic equation

$$y = \alpha \exp\{\lambda t\}$$

characteristic equation

$$\lambda + a = 0 \;\; \rightarrow \;\; \lambda = -a$$

substitution for λ

$$y = \alpha \exp\{-at\}$$

α according to initial condition

$$y(0) = \alpha = y_0$$

the solution is

$$y = y_0 \exp\{-at\}$$

Second order

The equation with initial conditions is

$$y'' + a_1 y' + a_0 y = 0, \quad y(0) = y_0, \;\; y'(0) = d_0$$

Characteristic equation

$$\lambda^2 + a_1 \lambda + a_0 = 0$$

Solution

1. two real roots - two exponentials
2. one double root - exponential and polynomial
3. two complex roots - exponentials and sine, cosine

Stability - real parts of the roots must be in the left half-plane.

2.2 Discretization

Approximate

It can be done by replacing the derivative by the difference.

The equation

$$y' + ay = 0$$

Replacing the derivative by the difference

$y'(t) \to \frac{y(t+T)-y(t)}{T} = \frac{y_{t+1}-y_t}{T}$ - where T is a step of discretization

$$\frac{y(t+T) - y(t)}{T} + ay(t) = 0$$

$$y_{t+1} - y_t + Tay_t = 0 \;\rightarrow\; y_{t+1} = \underbrace{(1 - Ta)}_{\tilde{A}} y_t$$

Precise

Here the generated discrete values are equal to the samples from the continuous solution

$$y_{t+1} = \exp\{-aT\} y_t = Ay_t$$

The derivation of this precise discretization can be found in the Appendix 13.3.

3 Regression model

The basic model for continuous data is the regression one. It has the form of difference equation with the random part. Its equation is

$$y_t = \psi_t' \Theta + e_t \tag{3.1}$$

- y_t modeled variable (output) at time t,
- ψ_t regression vector, containing values of variables influencing the output,
- Θ model parameters (regression coefficients θ and noise variance r),
- e_t noise, with zero expectation, constant variance, independent of variables in regression vector - sequence of independent and identically distributed (i.i.d.) random variables.

Here it is

$$\begin{aligned} \psi_t &= [u_t, y_{t-1}, u_{t-1} \cdots y_{t-n}, u_{t-n}, 1]' \\ \theta &= [b_0,\, a_1,\, b_1, \cdots\, a_n,\, b_n,\, k]', \end{aligned}$$

The model can be written in a more detailed form

$$y_t = b_0 u_t + a_1 y_{t-1} + b_1 y_{t-1} + \cdots + a_n y_{t-n} + b_n u_{t-n} + k + e_t$$

Comments

1. Number of delayed y and u can be different. Number of delayed y is called **model order.**
2. The term $\psi_t'\theta$ is at time t known constant. Model represents a transformation of e_t to y_t according to the model equation.
3. If ψ_t contains no delayed outputs, the model is static. Otherwise, it is dynamic.
4. $y_t = \psi_t'\theta$ is a difference difference equation.

Model as a pdf

A general description of the model as a tool, describing y_t as random variable is distribution

$$f\left(y_t|\psi_t, \Theta\right)$$

Moments of the model are

$$E\left[y_t|\psi_t, \Theta\right] = E\left[\psi_t'\theta + e_t\right] = \psi_t'\theta \equiv \hat{y}_t$$

$$D\left[y_t|\psi_t, \Theta\right] = D\left[\psi_t'\theta + e_t\right] = D\left[e_t\right] = r$$

Normal regression model as a pdf

Distribution of the normal noise is

$$f\left(e_t\right)=\frac{1}{\sqrt{2\pi r}}\exp\left\{-\frac{1}{2r}e_t^2\right\}$$

The transformation according to (3.1) is $y_t=\psi_t^{'}\theta+e_t\rightarrow e_t=y_t-\psi_t^{'}\theta$, with Jacobian equal to 1. The transformed pdf is the normal model and it has the form

$$f\left(y_t|\psi_t,\Theta\right)=\frac{1}{\sqrt{2\pi r}}\exp\left\{-\frac{1}{2r}\left(y_t-\psi_t^{'}\theta\right)^2\right\}$$

Example of generation

Generate values of y_t for $t=1,2,3$ from the regression model

$$y_t=u_t+0.4y_{t-1}-0.2u_{t-1}+e_t$$

with the initial condition $y_0=0.1$ and $u_0=-0.1$; $u_1=1.2$; $u_2=-0.6$; $u_3=0.1$.

The values of the noise e_t will be generated using the function `rand`. Let them be $e=[0.103,-0.211,0.125]$.

Solution

$$y_1=u_1+0.4y_0-0.2u_0+e_1=1.2+0.4\cdot0.1-0.2\cdot(-0.1)+0.103=1.363$$

$$y_2=u_2+0.4y_1-0.2u_1+e_2=-0.6+0.4\cdot1.363-0.2\cdot1.2-0.211=-0.5058$$

$$y_3=u_3+0.4y_2-0.2u_2+e_3=0.1+0.4\cdot(-0.5058)-0.2\cdot(-0.6)+0.125=0.14268$$

Program

```
// prgRegGen.sce
// Generation form a regression model
// ----------------------------------------
clc, clear, close, mode(0)

y(1)=.1;                              // initial y
u=[0 1.2 -.6 .1];                     // control
a=.4; b=[1 -.2];                      // model parameters
s=.1;                                 // std of noise

for t=2:4                             // shift of t due to zero indexes
```

```
  e(t)=s*rand(1,1,'n');
  y(t)=b(1)*u(t)+a*y(t-1)+b(2)*u(t-1)+e(t);  // generation of y
end

y, u                                  // print of results
// the results depend on actual values of rand. generators !
```

3.1 Regression model in a state-space form

The model describing the evolution of a state is

$$x_t = Mx_{t-1} + Nu_t + w_t.$$

The regression model introduced in (3.1) can be given this state form. We will demonstrate the transformation for the 2nd order regression model with constant term k

$$y_t = b_0u_t + a_1y_{t-1} + b_1u_{t-1} + a_2y_{t-2} + b_2u_{t-2} + k + e_t$$

We define the state as

$$x_t = [y_t, u_t, y_{t-1}, u_{t-1}, 1]' \rightarrow x_{t-1} = [y_{t-1}, u_{t-1}, y_{t-2}, u_{t-2}, 1]'$$

and substitute into the state model. Then, the first entry of the state is y_t and it is given by the regression model. The second entry is u_t and it is directly copied from the term Nu_t. The rest of the state entries express just shift of time: y_{t-1} was the first entry of x_{t-1} and now it is the third entry of x_t. Similarly for u_{t-1}. The constant k is not shifted, so it is plainly copied.

The resulting state model is

$$\begin{bmatrix} y_t \\ u_t \\ y_{t-1} \\ u_{t-1} \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & a_2 & b_2 & k \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{t-1} \\ u_{t-1} \\ y_{t-2} \\ u_{t-2} \\ 1 \end{bmatrix} + \begin{bmatrix} b_0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_t + \begin{bmatrix} e_t \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The advantage of the state-space model lies in recurrent computations. Its memory is only one. The computations have a closed form.

Example

For the regression model

$$y_t = b_0u_t + a_1y_{t-1} + b_1u_{t-1} + a_2y_{t-2} + b_2u_{t-2} + k$$

express y_3 with initial conditions y_0, y_{-1} and known u_t.

The solution using regression model

$$y_1 = b_0u_1 + a_1y_0 + b_1u_0 + a_2y_{-1} + b_2u_{-2} + k$$
$$y_2 = b_0u_2 + a_1y_1 + b_1u_1 + a_2y_0 + b_2u_0 + k$$
$$= b_0u_2 + a_1\left(b_0u_1 + a_1y_0 + b_1u_0 + a_2y_{-1} + b_2u_{-2} + k\right) +$$
$$+b_1u_1 + a_2y_0 + b_2u_0 + k$$
$$y_3 = \cdots \text{ horror}$$

If the regression model is transformed to the state form, we can write

$$x_1 = Mx_0 + Nu_1$$
$$x_2 = M\left(Mx_0 + Nu_1\right) + Nu_2 = M^2x_0 + MNu_1 + Nu_2$$
$$x_3 = M^3x_0 + M^2Nu_1 + MNu_2 + Nu_3$$

and

$$y_3 = x_{1;3}$$

where $x_{1;3}$ is the first entry of the state x_3.

We can even write a compact recurrent formula for arbitrary x_k.

$$x_k = M^kx_0 + \sum_{i=2}^{k} M^{k-i}u_i$$

Example of generation

For the regression model from the previous section

$$y_t = u_t + 0.4y_{t-1} - 0.2u_{t-1} + e_t$$

with the initial condition $y_0 = 0.1$ and $u_0 = -0.1$; $u_1 = 1.2$; $u_2 = -0.6$; $u_3 = 0.1$. and $e = [0.103, -0.211, 0.125]$ use the state form for generation of y_1, $t = 1, 2, 3$.

Solution

The equivalent regression model reads

$$\begin{bmatrix} y_t \\ u_t \\ 1 \end{bmatrix} = x_t = \begin{bmatrix} 0.4 & -0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_{t-1} + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} u_t + \begin{bmatrix} e_t \\ 0 \\ 0 \end{bmatrix}$$

So, for x_1 it is

$$x_1 = \begin{bmatrix} 0.4 & -0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ u_0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} e_1 \\ 0 \\ 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.4 & -0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} 1.2 + \begin{bmatrix} 0.103 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.363 \\ 0.1 \\ 1 \end{bmatrix}$$

where $x_1(1) = y_1 = 1.363$. For x_2

$$x_2 = \begin{bmatrix} 0.4 & -0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_1 + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} u_2 + \begin{bmatrix} e_2 \\ 0 \\ 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.4 & -0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1.363 \\ 0.1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} (-0.6) + \begin{bmatrix} -0.211 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5058 \\ -0.6 \\ 1 \end{bmatrix}$$

with $x_2\,(1) = y_2 = -0.5058$. And finally for x_3

$$x_3 = \begin{bmatrix} 0.4 & -0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_2 + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} u_3 + \begin{bmatrix} e_3 \\ 0 \\ 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.4 & -0.2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -0.5058 \\ -0.6 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} (0.1) + \begin{bmatrix} 0.125 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.14268 \\ 0.1 \\ 1 \end{bmatrix}$$

and $y_3 = x_3\,(1)$ =0.14268.

By comparison with the previous example with regression model we can see, that the results (with the same realization of noise) are exactly the same.

Program

```
// prgRegStGen.sce
// Generation form a logistic regression model
// ---------------------------------------
clc, clear, close, mode(0)

y=.1;
u=[-.1 1.2 -.6 .1];
x=[y(1) u(1) 0]';
M=[
```

```
.4 -.2 0
 0  0  0
 0  0  1
 ];
N=[1 1 0]';

for t=2:4
  w(:,t)=[.1*rand(1,1,'n') 0 0]';           // state noise
  x(:,t)=M*x(:,t-1)+N*u(t)+w(:,t);
  y(t)=x(1,t);                // generation of y
end

y                                  // print of results
// the results depend on actual values of rand. generators !
```

4 Discrete and logistic models

4.1 Discrete model

Discrete model can be used if all the variables entering model are discrete. Then there is a finite number of value configurations of data vector $\left[y_t', \psi_t'\right]'$. It enables us to assign a probability to each configuration separately and the model is

$$f\left(y_t|\psi_t, \Theta\right) = \Theta_{y_t|\psi_t} \tag{4.1}$$

y_t - output, ψ_t - regression vector, Θ parameter.

For two-valued variables and $\psi_t = \left[u_t', y_{t-1}'\right]'$ the parameters are $\Theta_{y_t|u_t,y_{t-1}}$. The model can be given a form of a table

$[u_t, y_{t-1}]$	$y_t = 1$	$y_t = 2$
1, 1	$\Theta_{1\|11}$	$\Theta_{2\|11}$
1, 2	$\Theta_{1\|12}$	$\Theta_{2\|12}$
2, 1	$\Theta_{1\|21}$	$\Theta_{2\|21}$
2, 2	$\Theta_{1\|22}$	$\Theta_{2\|22}$

In the left, there are all configurations of the regression vector. The entries of the table denote all configurations of the data vector, each of them is assigned its own parameter $\Theta_{i|jk}$ where $i \in \{1, 2\}$ is a value of y_t, $j \in \{1, 2\}$ is a value of u_t and $k \in \{1, 2\}$ is a value of y_{t-1}.

As all Θ are probabilities, it must hold:

$$\Theta_{i|jk} \geq 0, \ \sum_i \Theta_{i|jk} = 1, \ \forall jk$$

Remarks

1. The the model given by the given structure of is practically general. It is dynamic and possesses control variable. If there are more variables or more different values of th variables, only the dimension of Θ grows, but the model structure remains the same.

2. The number of all data configurations is always finite. However, with increasing number of variables and number of values of the variables, its dimension rapidly grows. Very soon it reaches such extend, that it can hardly be implemented in a standard computer.

Examples

1. *Model of an unfair coin*
   Tail is assigned by 1 and its probability is Θ_1, head is denoted by 2 and its probability is Θ_2. As there are no variables influencing the output y_t,the model is represented only by a vector and it has the form

$$f(y_t) = \Theta_{y_t}, \quad y_t \in \{1, 2\}$$

$y_t = 1$	$y_t = 2$
Θ_1	Θ_2

2. *Coin with a memory*
   In this case we expect, that the output y_t of the system (a coin) is influenced by the result of the previous result y_{t-1}. We can imagine, e.g. that the coin landed in a mud, got dirty and thus is unbalanced. Now, the model is represented by by the 2×2 matrix parameter as indicated below

$$f(y_t|y_{t-1}) = \Theta_{y_t|y_{t-1}}, \quad y \in \{1, 2\}$$

y_{t-1}	$y_t = 1$	$y_t = 2$
1	$\Theta_{1\|1}$	$\Theta_{2\|1}$
2	$\Theta_{1\|2}$	$\Theta_{2\|2}$

3. *Controlled coin*
   Here the output y_t is influenced again, but by the control variable. Otherwise, the situation is the same as for the coin with memory

$$f(y_t|u_t), \quad y, u \in \{1, 2\}$$

u_{t-1}	$y_t = 1$	$y_t = 2$
1	$\Theta_{1\|1}$	$\Theta_{2\|1}$
2	$\Theta_{1\|2}$	$\Theta_{2\|2}$

4. *Controlled coin with memory*
   In this model, both the influencing variables are present. As the regression vector ψ_t is really a vector, we must code the combinations of its values. If we write all these combinations so that the right position in the vector changes more rapidly (as indicated below), we can denote these combinations by the number of the row in the model matrix. That is 1,1→1, 1,2→2, 2,1→3, and 2,2→4. Notice! The code numbers correspond to the

numbers of the matrix row. For this simple case, of the regression vector $[u_t, y_{t-1}]$, the code numbers j can be determined by the formula

$$j = 2\,(u_t - 1) + y_{t-1.}$$

The model is

$$f\,(y_t|u_t, y_{t-1})\,,\ \ y, u \in \{1, 2\}$$

$[u_t, y_{t-1}]$	$y_t = 1$	$y_t = 2$
1, 1	0.8	0.2
1, 2	0.7	0.3
2, 1	0.25	0.75
2, 2	0.1	0.9

where y_t mostly obeys u_t

The meaning of the parameter

Uncertainty of the regression model is given by the noise variance. Here, it is given by Θ. If its entries are close to 0 or 1, the model is almost deterministic. If they are near to 0.5, the model is very uncertain. E.g.

$$\left[\begin{array}{cc} 0.1, & 0.9 \\ 0.9, & 0.1 \end{array}\right] \left[\begin{array}{cc} 0.4, & 0.6 \\ 0.6, & 0.4 \end{array}\right] \text{ or } \left[\begin{array}{cc} 1, & 0 \\ 0, & 1 \end{array}\right] \left[\begin{array}{cc} 0, & 1 \\ 1, & 0 \end{array}\right]$$

Example

For the model $f\,(y_t|u_t, y_{t-1})$, the model matrices represent deterministic models. The left one says: if $u_1 = 1$and $y_{t-1} = 1$, then $y_t = 0$;otherwise, $y_t = 1$. The second one, on the right, means: if $u_t = 0$ then $y_t = 1$; if $u_t = 1$, $y_t = 1$ (independently on y_{t-1})

$$\left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{array}\right] \left[\begin{array}{cc} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{array}\right]$$

Another possible verbal expression of the models is: Left - y_t is the bigger from u_t and y_{t-1}; right - y_t is the opposite to u_t.

Models

$$\left[\begin{array}{cc} 0.9 & 0.1 \\ 0.9 & 0.1 \\ 0.1 & 0.9 \\ 0.1 & 0.9 \end{array}\right] \left[\begin{array}{cc} 0.99 & 0.01 \\ 0.99 & 0.01 \\ 0.01 & 0.99 \\ 0.01 & 0.99 \end{array}\right]$$

express both the same rule and are uncertain. The right one is more deterministic.

The model

$$\begin{bmatrix} 0.51 & 0.49 \\ 0.47 & 0.53 \\ 0.52 & 0.48 \\ 0.49 & 0.51 \end{bmatrix}$$

is very uncertain (it practically carries on information).

Scilab generations

These ways of generation will be useful in our programs. You can or need not understand them. In the latter case, just use them.

- generate $y \in \{1, 2\}$ so that $P(y = 1) = 0.3$

```
y=(rand(1,1,'u')>0.3)+1      (one value);

y=(rand(1,nd,'u')>0.3)+1     (nd values);
```

- generate $y \in \{1, 2, \cdots, n\}$ so that $P(y = i) = p_i$; $p = [p_1 \cdots p_n]$

```
pp=cumsum(p);

y=sum(rand(1,1,'u')>pp)+1;
```

- number of row i in the table for u_t, $y_{t-1} \in \{1, 2\}$

```
i=2*(u(t)-1)+y(t-1);
```

- generate output y_t from the model $f(y_t|u_t, y_{t-1})$

```
i=2*(u(t)-1)+y(t-1);

pp=cumsum(th(i,:));

y(t)=sum(rand(1,1,'u')>pp)+1;
```

Example of generation

Let us generate three values of the output variable y_t from the discrete model

$$f\left(y_t|u_t, y_{t-1}, \Theta\right)$$

where both y and u are binary variables with values from $\{1, 2\}$ and the parameter Θ is given by the matrix

$$\begin{bmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \\ 0.7 & 0.3 \\ 0.8 & 0.2 \end{bmatrix}$$

The initial condition is $y_0 = 1$ and the input variable is $u_t = [1, 2, 2]$.

Solution

The model with description is

$[u_t, y_{t-1}]$	$y_t = 1$	$y_t = 2$
1,1	0.2	0.8
1,2	0.9	0.1
2,1	0.7	0.3
2,2	0.8	0.2

Time 1: The regression vector $[u_t, y_{t-1}] = [u_1, y_0] = [1, 1]$ - so we are at the first row of the table. So we should to generate from categorical (here it is also Bernoulli) distribution with parameters $p_1 = P\left(y_t = 1\right) = 0.2$ and $p_2 = P\left(y_t = 2\right) = 0.8$. According to the above formulas, we generate random uniform number U(0, 1)=rand(1,1,'u') - let it be 0.7369. Then rand(1,1,'u') $> 0.2 = 0.7369$ is True and

$$y_1 = (\text{rand(1,1,'u')} > 0.2) + 1 = 2$$

because True $+1 = 2$.

Time 2: The regression vector $[u_t, y_{t-1}] = [u_2, y_1] = [2, 2]$ - it indicates the fourth row in the table. Let the random generator gives a value 0.3184. Then

$$y_2 = (\text{rand(1,1,'u')} > 0.8) + 1 = 1,$$

as $0.3184 > 0.8 =$ False and False $+ 1 = 0$.

Time 3: The regression vector $[u_t, y_{t-1}] = [u_3, y_2] = [2, 1]$ - third row of the table. Let rand(1,1,'u') $= 0.9417$. Then

$$y_3 = (\text{rand(1,1,'u')} > 0.7) + 1 = 2,$$

as $0.9417 > 0.7 =$ True and True $+ 1 = 2$.

So, the generated values are $y = [2, 1, 2]$.

Program

```
// prgDiscGen.sce
// Generation form a discrete model
// ----------------------------------------
clc, clear, close, mode(0)

y(1)=1;                              // initial y
thy=[                                // y-parameter
0.2 0.8
0.9 0.1
0.7 0.3
0.8 0.2
];
thu=[0.6 0.4];                       // u-parameter

for t=2:4                            // shift of t due to zero indexes
  u(t)=(rand(1,1,'u')>thu(1))+1;     // generation of u
  i=2*(u(t)-1)+y(t-1);               // row of the y-parameter
  y(t)=(rand(1,1,'u')>thy(i,1))+1; // generation of y
end

y, u                                 // print of results
// the results depend on actual values of rand. generators !
```

4.2 Logistic model

This model is not entirely in one line with the previous ones because, as we will see later, its estimation cannot be done recursively. However, it is important because it can describe a special case when the model output is discrete and regression vector contains at least one continuous variable.

Neither regression nor discrete model can be used in this situation because the output of a regression model will not necessary be integer and the matrix of a discrete model would have infinite dimension due to the continuous variable in the regression vector.

For $y_t \in \{0, 1\}$ the model is

$$\begin{aligned} f\left(y_t|\psi_t, \Theta\right) &= \frac{\exp\left\{y_t z_t\right\}}{1+\exp\left\{z_t\right\}} \\ z_t &= \psi_t^{'}\Theta + e_t \end{aligned} \tag{4.2}$$

ψ_t regression vector with continuous and possibly discrete variables and the function $\frac{\exp\{\cdot\}}{1+\exp\{\cdot\}} = Lg\left(\cdot\right)$ is the logistic function.

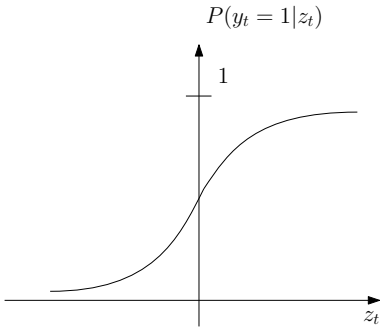
The model can be explained in two steps:

First, the regression $z_t = \psi_t' \Theta$ with the regression vector ψ_t is performed and scalar variable $z_t \in R$ is obtained. This is an ordinary regression with the output z_t.

Then, the real value z_t is transformed through the logistic function

$$Lg\,(z_t) = p_t$$

where p_t is the probability that the system output y_t is equal to one, on condition that the regression vector is ψ_t. The logistic function Lg has the graph



From this graph it can be seen that for arbitrary $z_t \in R$ it always holds that $p_t \in (0, 1)$. Moreover, if $z_t > 0$,then $p_t > 0.5$ and vice versa if $z_t > 0$, then $p_t < 0.5$.

It holds

$$\begin{array}{llll} \text{if } z_t > 0 & \text{then } P\,(y_t = 1|z_t) > 0.5 & \text{estimate: } y_t = 1 \\ \text{if } z_t < 0 & \text{then } P\,(y_t = 1|z_t) < 0.5 & \text{estimate: } y_t = 0 \end{array}$$

Example

Let he output y_t denotes "car accidents" with values: 0 - just damage, 1 - injury or death; the regression vector contains the variables: "light": 1 - full, 2 - gloom, 3 - dark; "weather": 1 dry, 2 - slippery; "speed": continuous variable with positive values.

Here the output is discrete - its values denote system modes (seriousness of the accident), regression vector are circumstances of the accident.

Remarks

1. The model (4.2) of logistic regression can be written in the form

$$P\,(y_t = 1|z_t) = \frac{\exp\{z_t\}}{1 + \exp\{z_t\}}$$

$$P\,(y_t = 0|z_t) = \frac{1}{1 + \exp\{z_t\}}$$

because for $y_t = 0$ it is $y_t z_t = 0$, too.

2. Notice, that the conditions for probabilities are satisfied

$$\frac{\exp\{z_t\}}{1+\exp\{z_t\}}+\frac{1}{1+\exp\{z_t\}}=1$$

and both the terms (probabilities) are nonnegative.

3. Other form of logistic model is

$$logit\,(p_t)=\psi_t'\Theta+e_t$$

where $logit\,(p)=\ln\frac{p}{1-P}$ is the **logistic function.**

4. For $y_t\in\{0,1,2,\cdots n\}$ the model can be extended

$$\ln\frac{p_1}{p_0}=\psi'\theta_1,\ \ln\frac{p_2}{p_0}=\psi'\theta_2,\cdots,\ln\frac{p_n}{p_0}=\psi'\theta_n$$

with the parameter $\Theta=[\theta_1,\theta_2,\cdots,\theta_n]$, where θ_i are columns. However, we will not follow this generalization.

Program

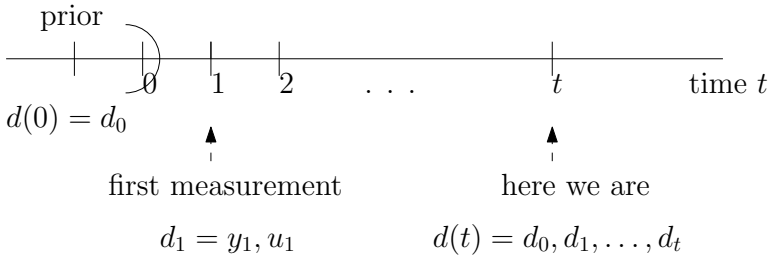
```
// prgLogrGen.sce
// Generation form a logistic regression model
// ---------------------------------------
clc, clear, close, mode(0)

th=[-3 5 8];                        // model parameters
x=[                                 // data (in rows)
.3 .2 -.1
.6 -.8 .5
-.7 .2 -.3
];

for t=1:3
  z(t)=x(t,:)*th';                  // regression
  p(t)=exp(z(t))/(1+exp(z(t)));     // logistic function
  y(t)=round(p(t));                 // generation of y
end

y                                   // print of results
// the results depend on actual values of rand. generators !
```

5 Bayesian estimation

Notation

To be able to express the data history, from which the estimation takes information, we introduce the following notation:

- y_t is the value of the output y at time t (we have already used it),
- $d_t = \{y_t, u_t\}$ is the couple of variables that are usually generated by (measured on) the system at time instant t,
- $d\,(t) = \{d_0, d_1, d_2, \cdots, d_t\}$ is a set of all values of the system data up to and including time instant t,
- $d\,(0) = d_0$ are the prior data, i.e. data measured before the estimation starts or data (information) provide by an expert.



5.1 Estimation with Bayes rule

Classical statistics define parameters as unknown constants. Their description is a number (point estimate) or a confidence interval within which a true value of the parameter lies with a given probability.

Bayesian statistics treats parameters as random variables - their description is a distribution.

Types of distributions

In estimation we distinguish two types of distributions

- model distribution
$$f\,(y_t|\psi_t, \Theta)$$
which is a distribution of the output,
- parameter distributions
$$f\,(\Theta|d\,(t-1))\,,\ f\,(\Theta|d\,(t))$$

which are distributions of the parameter; the first one based on old data with respect to the current time t, the second one using information from all available data at time t.

Bayes rule

The estimation runs according to the Bayes rule. It says how you can obtain new (posterior) parameter distribution $f(\Theta|d(\tau))$ from the old (prior) one $f(\Theta|d(\tau-1))$ by means of the system model $f(y_\tau|\psi_\tau,\Theta)$ for $\tau=1,2,\cdots,t$

$$f(\Theta|d(\tau)) \propto f(y_\tau|\psi_\tau,\Theta)\, f(\Theta|d(\tau-1))$$

It gives an evolution of the parameter pdf in time, while newly measured data are supported

$$f(\Theta|d(0)) \underbrace{\rightarrow}_{d_1=\{u_1,y_1\}} f(\Theta|d(1)) \underbrace{\rightarrow}_{d_2=\{u_2,y_2\}} \cdots \underbrace{\rightarrow}_{d_t=\{u_t,y_t\}} f(\Theta|d(t))$$

with the initial (prior) pdf $f(\Theta|d(0))$ constructed from prior data or specified by an expert.

Remarks

1. *Derivation* of the Bayes rule is very simple

$$\begin{aligned} f(A,B|C) &= f(A|B,C)\, f(B|C) \\ &= f(B|A,C)\, f(A|C) \end{aligned}$$

$$\rightarrow \qquad f(A|B,C) = \frac{f(B|A,C)f(A|C)}{f(B|C)}$$

   For estimation of parameters, we lay

$$A \rightarrow \Theta,\ \ B \rightarrow d_t,\ \ C = d(t-1)$$

   and $\{B,C\} = \{d_t,\, d(t-1)\} = d(t)$.

2. **Natural conditions of control**
   In derivation of the Bayes rule for model with control variable, we assume so called Natural conditions of control (NCC)

   $f(\Theta|u_t,d(t-1)) = f(\Theta|d(t-1))$ and conversely

   $f(u_t|d(t-1),\Theta) = f(u_t|d(t-1))$.

   Remark

   NCC can be explained in the following way: We assume that the person that estimates also controls. In construction of the control u_t he uses only information from $d(t-1)$. That is why (in the former condition) that u_t in condition cannot bring any other information than that which is already in $d(t-1)$ - thus it can be omitted in condition.
   The latter condition is derived from the former one using Bayes rule.

   Thus, in estimation with a model with control, it applies

$$f(\Theta|d(t)) \propto f(y_t|\psi_t,\Theta)\, f(u_t|d(t-1),\Theta)\, f(\Theta|d(t-1))$$

which means that in recursion for Θ the pdf $f(u_t|\cdots)$ is only a constant term. In the recursion it disappears as for the Bayes rule hols only proportionality $\propto$.

3. *Batch estimation*

   From the Bayes rule if follows

$$f(\Theta|d(t)) \propto \underbrace{\left[\prod_{\tau=1}^{t} f(y_\tau|\psi_t\Theta)\right]}_{L_t(\Theta)} f(\Theta|d(0)) = L_t(\Theta) f(\Theta|d(0))$$

   where $L_t(\Theta) = \prod_{\tau=1}^{t} f(y_\tau|\psi_\tau, \Theta)$ is likelihood and $f(\Theta|d(0))$ is the very prior pdf.

4. *Self reproducing form* of the Bayes rule
   Bayes rule is a recursive for functions (distributions). To be able to work with functions it is necessary to parameterize them - e.g. normal distribution is given just by two numbers expectation and variance. Recursivity requires so that the structure form of prior pdf after multiplication by the model is reproduced in the posterior pdf. E.g. if the prior pdf is normal, we need so that the posterior pdf would be normal, too. Only the statistics of the posterior (which are numbers or vectors) are recomputed.
   An example of recursive recomputation is here

   Example (recursive)
   For the exponential distribution the model is

$$f(y_t|a) = a\exp\{-ay_t\}$$

   The form of prior and posterior pdfs can be guessed for th model products (Likelihood)

$$L = f(a|y_1) f(a|y_2) f(a|y_3)\cdots = a\exp\{-ay_1\} a\exp\{-ay_2\} a\exp\{-ay_3\}\cdots =$$

$$= a^3\exp\{-a(y_1+y_2+y_3)\}$$

   We denote (statistics): $\kappa_3 = 3$ and $S_3 = y_1 + y_2 + y_3$ and we can write the product

$$L_3 = a^{\kappa_3}\exp\{-aS_3\}$$

   The statistics κ and S is simply evolved in time as follows

$$\begin{aligned} \kappa_\tau &= \kappa_{\tau-1} + 1 \\ S_\tau &= S_{\tau-1} + y_\tau \end{aligned}$$

   for any $\tau = 1, 2, \cdots$, with initial statistics κ_0 and S_0.

   Remark

   The meaning of the initial statistics is

- κ_0 *is a number of prior data samples, from which the prior statistics is constructed.*
- $S_0 = \sum_{i=1}^{\kappa_0} y_i$ *is a sum of prior data.*

If we do not have any prior data but only a prior guess about data average $\bar{y}$ *provided by some expert we can construct the prior statistics* κ_0 *and* S_0 *according to the formula* $\bar{y} = \frac{S_0}{\kappa_0}$. *The belief in the prior information is given by the number* κ_0 *(as if number of data used) and for* S_0 *we have* $S_0 = \bar{y}\kappa_0$.
In practice, the larger κ_0 *and* S_0 *are the greater is their effect.*

Example (not recursive)
The following models are examples of those, that do not lead to recursive estimation

$$f\left(y_t|a\right) = \frac{a}{1+a}\left(\frac{1}{y_t^2} + \exp\left\{-ay_t\right\}\right)$$

or

$$f\left(y_t|a\right) = \frac{1}{2+\pi a}\left(\sin\left(y_t\right) + a\right)$$

When the product of such models is computed the number of different terms grows and the form of this product (Likelihood or posterior pdf) is more and more complex until the computations are unfeasible.

- *Results* of estimation

  (i) **Posterior** pdf $f\left(\Theta|d\left(t\right)\right)$ is the immediate result following from Bayesian estimation. It brings all the information available about the estimated parameters. Sometimes it can be used as it is - e.g. in output prediction

$$f\left(y_t|d\left(t-1\right)\right) = \int_{\Theta^*} f\left(y_t, \Theta|d\left(t-1\right)\right) d\Theta = \int_{\Theta^*} f\left(y_t|\psi_t, \Theta\right) \underbrace{f\left(\Theta|d\left(t-1\right)\right)}_{\text{posterior for time t-1}} d\Theta$$

  (ii) **Point estimates** $\hat{\Theta}$ or $\hat{y}$ constitute representation of the values of estimated parameters or unknown future output. They can be computed using posterior pdf as follows

$$\hat{\Theta}_t = E\left[\Theta|d\left(t\right)\right] = \int_{\Theta^*} \Theta \underbrace{f\left(\Theta|d\left(t\right)\right)}_{\text{posterior}} d\Theta$$

$$\hat{y}_t = E\left[y_t|d\left(t-1\right)\right] = \int_{y^*} y_t\, f\left(y_t|d\left(t-1\right)\right) dy_t =$$

$$\int_{y^*} y_t \left[\int_{\Theta^*} f\left(y_t|\psi_t, \Theta\right) \underbrace{f\left(\Theta|d\left(t-1\right)\right)}_{\text{posterior for time t-1}} d\Theta\right] dy_t$$

6 Estimation of specific models

6.1 Recursive estimation of normal regression model

It is the most frequently used model. However, the way of its estimation is a bit tricky to obtain statistics in a nice compact form. The trick concerns a special form of the model into which it is transformed.

Model

$$f\left(y_t|\psi_t,\Theta\right)=\frac{1}{\sqrt{2\pi}}r^{-0.5}\exp\left\{-\frac{1}{2r}\left(y_t-\psi_t'\theta\right)^2\right\}$$

For estimation, this model is given a special form. The term under the square it holds

$$y_t-\psi_t'\theta=\left[y_t,\psi_t'\right]\left[\begin{array}{c}-1\\ \theta\end{array}\right]=y_t-\theta'\psi_t=\left[1,\,-\theta'\right]\left[\begin{array}{c}y_t\\ \psi_t\end{array}\right]$$

Then the square in the exponent of the model can be written in the following way

$$\left(y_t-\psi_t'\theta\right)^2=\left(y_t-\theta'\psi_t\right)\times\left(y_t-\psi_t'\theta\right)=$$

$$=(-1)\left[-1,\,\theta'\right]\left[\begin{array}{c}y_t\\ \psi_t\end{array}\right]\times(-1)\left[y_t,\,\psi_t'\right]\left[\begin{array}{c}-1\\ \theta\end{array}\right]=$$

$$=\left[-1,\,\theta'\right]\underbrace{\left[\begin{array}{c}y_t\\ \psi_t\end{array}\right]\left[y_t,\,\psi_t'\right]}_{D_t}\left[\begin{array}{c}-1\\ \theta\end{array}\right]$$

where D_t is so called data matrix.

Model in modified form

$$f\left(y_t|\psi_t,\Theta\right)\propto r^{-0.5}\exp\left\{\left[-1,\,\theta'\right]D_t\left[\begin{array}{c}-1\\ \theta\end{array}\right]\right\}$$

Prior pdf

In the same form as model we express the prior pdf - it is Gauss-inverse-Wishart (GiW) distribution

$$f\left(\Theta|d\left(0\right)\right)\propto r^{-0.5\kappa_0}\exp\left\{\left[-1,\,\theta'\right]V_0\left[\begin{array}{c}-1\\ \theta\end{array}\right]\right\}$$

where V_0 and κ_0 are prior statistics.

The recursion for evolution of the statistics can be derived if we substitute the model and prior pdf into the Bayes rule and get the first posterior $f\left(\Theta|d\left(1\right)\right)$

Bayes

$$f\left(\Theta|d\left(1\right)\right) \propto r^{-0.5} \exp\left\{\left[-1,\, \theta'\right] D_1 \left[\begin{array}{c} -1 \\ \theta \end{array}\right]\right\} r^{-0.5\kappa_0} \exp\left\{\left[-1,\, \theta'\right] V_0 \left[\begin{array}{c} -1 \\ \theta \end{array}\right]\right\} =$$

$$= r^{-0.5(\kappa_0+1)} \exp\left\{\left[-1,\, \theta'\right] \left(D_1 + V_0\right) \left[\begin{array}{c} -1 \\ \theta \end{array}\right]\right\} =$$

$$= r^{-0.5\kappa_1} \exp\left\{\left[-1,\, \theta'\right] V_1 \left[\begin{array}{c} -1 \\ \theta \end{array}\right]\right\} \tag{6.1}$$

Posterior

The general form of the posterior at time t is (the form of the posterior is fixed, only the statistics are indexed by t)

$$f\left(\Theta|d\left(t\right)\right) \propto r^{-0.5\kappa_t} \exp\left\{\left[-1,\, \theta'\right] V_t \left[\begin{array}{c} -1 \\ \theta \end{array}\right]\right\}$$

Recursion

By comparison of the second and third part of (6.1) the general form of the recursion for statistics can be easily derived

$$\kappa_t = \kappa_{t-1} + 1 \tag{6.2}$$
$$V_t = V_{t-1} + D_t \tag{6.3}$$

with κ_0 and V_0 as prior statistics.

Result

(a) As a result we can take the derived posterior pdf. It is given by the GiW with the computed statistics κ_t and V_t inserted.

(b) If needed, the point estimates of parameters can be determined in the following way. Firs we must divided information matrix as indicted

$$V_t = \left[\begin{array}{cc} V_y & V_{y\psi} \\ V_{y\psi} & V_\psi \end{array}\right] \cdots \left[\begin{array}{cc} \bullet & -- \\ | & \square \end{array}\right]$$

Then it holds: the ***estimate of regression coefficients*** is

$$\hat{\theta}_t = V_\psi^{-1} V_{y\psi} \tag{6.4}$$

and the ***estimate of noise covariance*** is

$$\hat{r}_t = \frac{V_y - V'_{y\psi} V_\psi^{-1} V_{y\psi}}{\kappa_t}$$

Point estimate of the output = prediction

$$\hat{y}_t = \psi_t \hat{\theta}_{t-1} \qquad (\theta \to \hat{\theta}_{t-1},\ e_t \to 0)$$

which means that in the model, we substitute the parameter estimates $\hat{\theta}_t$ for the regression parameter θ and lay $e_t = 0$.

Remark

To be able to see in prediction not only the position (expectation)of the output y_t but also its uncertainty, it is advantageous to make co called simulated prediction. It is

$$\hat{y}_t = \psi_t \hat{\theta}_{t-1} + \sqrt{\hat{r}_t} N(0,1)$$

where $\hat{r}_t$ is the point estimate of the noise variance and $N(0,1)$ is generator of standard normal distribution.
If $\hat{r}_t$ is a matrix (covariance matrix), instead of the square root, we must perform so called LD decomposition. It can be done with the function uut.sci from our package of Scilab functions.

Program

```
// prgRegEst.sce
// Estimation of regression model
//    y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)+k+e(t)
// ---------------------------------------
clc, clear, close, mode(0)

y=[.5 -.5 .1 .8 .3];                  // output
u=[-.1 1.2 -.6 .1 .9];                // input

V=zeros(5,5);                        // initial statistics
for t=2:5                            // loop of statistics update
  Ps=[y(t) u(t) y(t-1) u(t-1) 1]'; // extended regression vector
  V=V+Ps*Ps';                        // update
end
Vyp=V(2:5,1); Vp=V(2:5,2:5);         // partitioning of statistics
th=inv(Vp)*Vyp;                      // point estimates of parametrs

b0=th(1), a1=th(2), b1=th(3), k=th(4) // results of estimation
```

6.2 Batch estimation of regression model

Sometimes, when the dataset is already collected and at disposal, it is advantageous to use batch estimation. It is very easy and handy. The principle of the estimation is as follows:

The model is

$$y_t = b_0 u_t + \cdots a_n y_{t-n} + b_n u_{t-n} + k + e_t$$

for $t = 1, 2, \cdots, N$

Write the model for each time instant of measurements

$$\begin{aligned} y_1 &= b_0 u_1 + \cdots a_n y_{1-n} + b_n u_{1-n} + k + e_1 \\ y_2 &= b_0 u_2 + \cdots a_n y_{2-n} + b_n u_{2-n} + k + e_2 \\ &\cdots \\ y_N &= b_0 u_N + \cdots a_n y_{N-n} + b_n u_{N-n} + k + e_N \end{aligned}$$

In this way you obtain the following matrix form of the model with all measured data

$$Y = X\theta + E$$

The optimization minimizes the following criterion (least squares)

$$J = \sum e_i^2 = E'E = (Y - X\theta)'(Y - X\theta) = Y'Y - 2\theta' X'Y + \theta' X'X\theta$$

$$\frac{\partial}{\partial \theta} J = -2X'Y + 2X'X\theta = 0$$

$$X'X\theta = X'Y \quad \rightarrow \quad \underbrace{\hat{\theta}_t = \left(X'X\right)^{-1} X'Y}_{\text{the result}}$$

where

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}, \quad X = \begin{bmatrix} u_1 & y_0 & u_0 & \cdots & y_{1-n} & u_{1-n} & 1 \\ u_2 & y_1 & u_1 & \cdots & y_{2-n} & u_{2-n} & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ u_N & y_{N-1} & u_{N-1} & \cdots & y_{N-n} & u_{N-n} & 1 \end{bmatrix}$$

Remarks

1. *In the derivation we used matrix derivative. If you differentiate according to the transposed parameter θ' then the rules are the same as for a scalar derivative.*
2. *The dataset is usually obtained in Excel (or other database program) from which both the vector Y and the matrix X can be directly copied.*

Program

```
// prgRegBEst.sce
// Batch estimation of regression model
//    y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)+k+e(t)
// with data
// y=[.5  -.5  .1 .8 .3];                  // output
// u=[-.1 1.2 -.6 .1 .9];                // input
// t   0   1   2   3
// ----------------------------------------
clc, clear, close, mode(0)

Y=[
-.5
.1
.8
.3
];

X=[
1.2  .5 -.1 1
-.6 -.5 1.2 1
 .1  .1 -.6 1
 .9  .8  .1 1
];

th=inv(X'*X)*X'*Y;       // point estomates

b0=th(1), a1=th(2), b1=th(3), k=th(4) // results of estimation
```

6.3 Categorical model

Estimation of categorical model basically can be explained on an experiment of flipping a coin. If we want to verify that the selected coin is fair (the same probability of both sides), we proceed as follows: We toss the coin many times and count number of heads and tails. Then divide these numbers by the total number of tosses and you obtain the desired probabilities.

Formally, to obtain procedure of estimation a discrete (categorical) model according to the Bayes rule, we proceed in the following way.

Product form of the model

We formally express the model in a product form[1]

[1] Here $y|\psi$ is a vector index - instead of an integer for a standard index, here it is a vector of indexes.

$$f\left(y_t|\psi_t,\Theta\right)=\Theta_{y_t|\psi_t}=\prod_{y|\psi}\Theta_{y|\psi}^{\delta(y|\psi;\,y_t|\psi_t)}$$

i.e. product over all possible configurations of $y|\psi$; but due to the Kronecker function $\delta\left(y|\psi;\,y_t|\psi_t\right)=1$ for $y=y_t$ and $\psi=\psi_t$ and equal to zero otherwise, only $y_t|\psi_t$ is chosen.

Posterior pdf

Posterior pdf is introduced with the same form as the product form of the model

$$f\left(\Theta|d\left(t\right)\right)\propto\prod_{y|\psi}\Theta_{y|\psi}^{\nu_{y|\psi;t}}$$

where $\nu_{y|\psi;t}$ for all configurations of $y|\psi$ is statistics; $\nu_{y|\psi;0}$ is the prior one.

Statistics update

Similarly as for the regression model 6.1 we substitute model and prior (i.e. posterior for time $t-1$) to the Bayes rule we and obtain

$$\nu_{y|\psi;t}=\nu_{y|\psi;t-1}+\delta\left(y|\psi;\,y_t|\psi_t\right) \tag{6.5}$$

for all configurations of $y|\psi$ (or $\nu_{y_t|\psi_t;t}=\nu_{y_t|\psi_t;t-1}+1$ for actual data)

Remark

In practice, it means: Choose the statistics entry corresponding to $y_t|\psi_t$ and increment it by one. It resembled estimation of the coin - if you obtain "head", take number of heads and increment it by 1.

Point estimate

Similarly as for the coin, the point estimates are computed as follows

$$\hat{\theta}_{y|\psi;t}=\frac{\nu_{y|\psi;t}}{\sum_i\nu_{i|\psi;t}} \tag{6.6}$$

It is nothing more than normalization of the statistic matrix so that the sum of its row entries is equal to one.

Remark

The dynamic model with control variable introduced in 4.1 by the table below can be viewed as four plain models (each given by one row of the table) indexed by the vector index ψ. Thus the statistics update runs as follows: Look at the regression vector and find the corresponding row (model). Then update this model, i.e. according to the value of y increment the corresponding entry by adding 1.

Example

Estimation of a plain coin

Model

$$f(y_t|p) = p_{y_t}, \;\; y = 1, 2\,;\, p = [p_1,\, p_2]'$$

where y_t is the result of a toss at time t, p_1, p_2 are probabilities, $p_1 + p_2 = 1$.

Product form

$$f(y_t|p) = p_1^{\delta(y_t,1)} p_2^{\delta(y_t,2)}$$

Posterior

$$f(p|y(t)) \propto p_1^{\nu_{1;t}} p_2^{\nu_{2;t}}$$

Statistics

$$\nu_t = [\nu_{1;t},\, \nu_{2;t}]$$

Update

– for $y = 1$

$$\nu_{1;t} = \nu_{1;t-1} + 1$$

– for $y = 2$

$$\nu_{2;t} = \nu_{2;t-1} + 1$$

For the data

t	1	2	3
y_t	1	1	2

and zero initial statistics the, the progress of estimation is indicated here

t	0	1	2	3
ν_1	0	1	2	2
ν_2	0	0	0	1
p_1	x	1	1	$\frac{2}{3}$
p_2	x	0	0	$\frac{1}{3}$

If we choose the initial statistics 10 for both initial statistics, the progress is different

t	0	1	2	3
ν_1	10	11	12	12
ν_2	10	10	10	11
p_1	x	0.524	0.546	0.522
p_2	x	0.476	0.454	0.478

The magnitude of ν entries of the initial statistics expresses our belief in our prior information. The information in both case is that the coin is fair. In the latter case the belief in our prior information is higher.

Output estimate

From estimated model, the output prediction can be constructed through the predictive probability

$$f\left(y_t|d\left(t-1\right)\right)=f\left(y_t|\psi_t,\Theta=\hat{\Theta}_{t-1}\right)$$

y_t	1	2	3	$\cdots$	n
$f\left(y_t\|\psi_t\hat{\Theta}_{t-1}\right)$	$P\left(y_t=1\right)$	$P\left(y_t=2\right)$	$P\left(y_t=3\right)$		$P\left(y_t=n\right)$

Point estimate $\hat{y}_t$ can be obtained either as the argument of the largest probability $\hat{y}_t=i$ for the greatest $P\left(y_t=i\right)$ or, better, as the expectation

$$\hat{y}_t=\sum_j jP\left(y_t=j\right).$$

This latter way does not lead to prediction within the values of y_t.

Example

For data $y=[1\ 2\ 1\ 1\ 2\ 1\ 2]$ and $u=[1\ 1\ 2\ 2\ 1\ 1\ 2]$ estimate the categorical model

$$f\left(y_t|u_t,y_{t-1}\right)$$

where $y\left(1\right)$ and $u\left(1\right)$ are initial conditions.

Solution

We define statistics

$$V_0=\begin{bmatrix}0&0\\0&0\\0&0\\0&0\end{bmatrix}$$

and we update it with the data

time 1: $j=2\left(u_1-1\right)+y_0=2\left(1-1\right)+1=1$

$$V(j,y_1)=V\left(1,2\right)=V(1,2)+1$$

i.e.

$$V_1=\begin{bmatrix}0&1\\0&0\\0&0\\0&0\end{bmatrix}$$

time 2: $j = 2\,(u_2 - 1) + y_1 = 2\,(2-1) + 2 = 4$

$$V(4,1) = V\,(4,1) + 1$$

$$V_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

and similarly for other time instants. See the Program below. In the end we obtain statistic

$$V_6 = \begin{bmatrix} 0 & 2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

The point estimates are obtained by normalization of the rows to sum equal to one

$$\Theta = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0.5 & 0.5 \\ 1 & 0 \end{bmatrix}$$

Remark

In the normalization we divide the rows of the model by their sums. If some row stays all zero, the program ends with error. To this end, it is advantageous to start with the statistics with small numbers instead with zeros.

Program

```
// prgDiscEst.sce
// Batch estimation of categorical model
//    y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)+k+e(t)
// with given data.
// ----------------------------------------
clc, clear, close, mode(0)

y = [1 2 1 1 2 1 2];
u = [1 1 2 2 1 1 2];
//t= 0 1 2 3 4 5 6

V=zeros(4,2);
for t=2:7
```

```
  j=2*(u(t)-1)+y(t-1);
  V(j,y(t))=V(j,y(t))+1;
end

for i=1:4
  th(i,:)=V(i,:)/sum(V(i,:));
end
th
```

6.4 Model of logistic regression

For estimation, numerical maximization of log-likelihood is used.

For $y_t \in \{0, 1\}$ the **model** is

$$f\left(y_t|z_t\right) = \frac{\exp\left\{y_t z_t\right\}}{1+\exp\left\{z_t\right\}}, \quad z_t = \psi_t^{'}\Theta + e_t$$

Likelihood

$$L_t = \prod_{\tau=1}^{t} f\left(y_\tau|z_\tau\right) = \prod_{\tau=1}^{t} \frac{\exp\left\{y_\tau z_\tau\right\}}{1+\exp\left\{z_\tau\right\}}$$

$$\ln L_t = \sum_{\tau=1}^{t} \left[y_t z_t - \ln\left(1+\exp\left\{z_t\right\}\right)\right], \quad z_t = \psi_t^{'}\Theta$$

$$\hat{\Theta}_t = \arg\min_{\Theta} \ln L_t$$

for minimization, Newton method can be used.

Output estimation

Substitute ψ_t into the model with parameter estimates. The value with the biggest probability can be selected.

Classification

The space of all possible ψ is divided into two subsets - one with $\hat{y} = 0$, the other with $\hat{y} = 1$.

7 Prediction of model output

Prediction means estimation of the future output. We assume, we are at time instant t but the output y_t has not been measured, yet. Its estimation is called **zero step prediction**. It serves mainly for validation of the model when the prediction is compared to the output after its measurement. The estimation of the more remote output, say y_{t+k} is called **k-step prediction** where k is number of the steps ahead.

The k-steps prediction is governed by the predictive pdf

$$f\left(y_{t+k}|y\left(t-1\right),u\left(t+k\right)\right)$$

i.e. probability of values of the output k-steps ahead from the current time t, conditioned by values of the variables already measured at the present time t, when y_t is not known, yet.

Here, the control variable is somehow unhandy, so further on we will consider only models without control.

7.1 Output estimation (zero step prediction)

Predictive pdf

Te task is: for a regression model without control $f\left(y_t|y\left(t-1\right),\Theta\right)$[2] determine the predictive pdf $f\left(y_t|y\left(t-1\right)\right)$. This pdf is similar to the model, however, the parameters are missing and without the parameters, the model is useless. So, we must supply the parameters into the model. It can be done in the following way (we add the parameters and immediately integrate them out; then we use the chain rule)

$$f\left(y_t|y\left(t-1\right)\right)=\int_{\Theta^*}f\left(y_t,\Theta|y\left(t-1\right)\right)d\Theta=$$

$$=\int_{\Theta^*}f\left(y_t|y_{t-1},\Theta\right)f\left(\Theta|y\left(t-1\right)\right)d\Theta \tag{7.1}$$

where $f\left(y_t|y_{t-1},\Theta\right)$ is model and $f\left(\Theta|y\left(t-1\right)\right)$ is the posterior pdf conditioned on data actually at disposal (we are at time t, y_t has not been measured, yet, so at disposal are data $y\left(t-1\right)$.

Remark

The pdf $f\left(\Theta|y\left(t-1\right)\right)$ can be viewed as prior for time t or, better, as posterior form th last step $t-1$.

[2]The model as we introduced it is $f\left(y_t|\psi_t,\Theta\right)$. Without control variable the regression vector is $\psi_t=\left[y_{t-1},y_{t-2},\cdots,y_{t-n}\right]$. Here, for simplicity and lucidity we replaced ψ_t by $y\left(t-1\right)$. The variables y_{t-n-1},y_{t-n-2}, $\cdots$ are independent of y_t and thus, in reality, they would be omitted in the condition.

The derived formula for output prediction requires integration. If we want to get rid of it, we can rely on point estimates of parameters $\hat{\Theta}_{t-1}$ (again based on past data). To this end we express the posterior i the following way (something like "deterministic" distribution)

$$f\left(\Theta|y\left(t-1\right)\right) \doteq \delta\left(\Theta, \hat{\Theta}_{t-1}\right)$$

We substitute into (7.1) and obtain

$$f\left(y_t|y\left(t-1\right)\right) = \int_{\Theta^*} f\left(y_t|y_{t-1}, \Theta\right) f\left(\Theta|y\left(t-1\right)\right) d\Theta \doteq$$

$$\doteq \int_{\Theta^*} f\left(y_t|y_{t-1}, \Theta\right) \delta\left(\Theta, \hat{\Theta}_{t-1}\right) d\Theta = f\left(y_t|y_{t-1}, \hat{\Theta}_{t-1}\right)$$

where $\hat{\Theta}_{t-1} = E\left[\Theta|y\left(t-1\right)\right] = \int_{\Theta^*} \Theta f\left(\Theta|y\left(t-1\right)\right) d\Theta$ is the point estimate of Θ based on the data $y\left(t-1\right)$.

Point prediction

Point prediction can be determined as expectation of the output, i.e.

$$\hat{y}_t = E\left[y_t|y\left(t-1\right)\right] = \int_{y^*} y_t f\left(y_t|y\left(t-1\right)\right) dy$$

The point prediction based on point estimates of the parameters can be evolved directly using the model equation with point estimates of parameters inserted and without the noise term. The procedure is very natural. E.g. for model

$$y_t = bu_t + ay_{t-1} + e_t$$

with a, b being either known parameters or their point estimates, we can write

$$\begin{aligned} \hat{y}_1 &= bu_1 + ay_0 \\ \hat{y}_2 &= bu_2 + a\hat{y}_1 \\ \hat{y}_3 &= bu_3 + a\hat{y}_2 \end{aligned} \tag{7.2}$$

etc.

Program

Zero step point prediction with 2nd order regression model with known parameters:

```
// prg0stPre.sce
// Zero step prediction with the model
//      y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)  for y(t)=0
// ---------------------------------------
clc, clear, close, mode(0);

nd=120;                              // length of data
b0=.8; a1=.8; b1=1; sd=.2;           // model parameters
y(1)=0; yp(1)=0;                     // initial conditions
u=.5+.1*rand(1,nd,'n');              // generation of control for all t

for t=2:nd                           // loop simulation and prediction
  // prediction
  yp(t)=b0*u(t)+a1*yp(t-1)+b1*u(t-1);
  // simulation
  y(t)=b0*u(t)+a1*y(t-1)+b1*u(t-1)+sd*rand(1,1,'n');
end

// Rsults
plot(1:nd,y,1:nd,yp)
legend('y','yp');
title('Zero step prediction','fontsize',5)
```

The same program as the previous but with model with unknown parameters:

```
// Prg0stPreEst.sce
// Zero step prediction with the model with unknown parameters
//       y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)  for y(t)=0
// Prediction is without knowledge of y(t), then we measure y(t)
//   and use new data for estimation. First step uses priors.
// ---------------------------------------
clc, clear, close, mode(0);

nd=120;                                  // length of data
b0=.8; a1=.8; b1=1; sd=.2;               // model parameters
y(1)=0; yp(1)=0; u(1)=0;                 // initial conditions
u=.5+.1*rand(1,nd,'n');                  // generation of input for all t
V=.0001*eye(4,4);                        // information matrix
b0E=rand(1,1,'n'); a1E=rand(1,1,'n'); b1E=rand(1,1,'n'); // priors

for t=2:nd                               // time loop
  // prediction
  yp(t)=b0E*u(t)+a1E*yp(t-1)+b1E*u(t-1); // prediction
```

```
  // simulation
  y(t)=b0*u(t)+a1*y(t-1)+b1*u(t-1)+sd*rand(1,1,'n'); // simulation
  // estimation
  Ps=[y(t) u(t) y(t-1) u(t-1)]';        // ext. regression vector
  V=V+Ps*Ps';                           // update of statistics
  Vyp=V(2:4,1); Vp=V(2:4,2:4); th=inv(Vp)*Vyp; // estimation
  b0E=th(1); a1E=th(2); b1E=th(3);      // estimated parameters
end

// Rsults
plot(1:nd,y,1:nd,yp)
legend('y','yp');
title('Zero step prediction','fontsize',5)
```

7.2 One step prediction

It means: We are at time t, have data $y(t-1)$ at disposal, y_t is not measured, yet, and we want to estimate the output y_{t+1} at time $t+1$. The procedure is the same as for the unknown parameters but now for both the parameters Θ and the output y_t

$$f\left(y_{t+1}|y\left(t-1\right)\right)=\int_{\Theta^{*}}\int_{y_{t}^{*}}f\left(y_{t+1},y_{t},\Theta|y\left(t-1\right)\right)dy_{t}d\Theta=$$

$$=\int_{\Theta^{*}}\int_{y_{t}^{*}}\underbrace{f\left(y_{t+1}|y\left(t\right),\Theta\right)}_{\text{model at }t+1}\underbrace{f\left(y_{t}|y_{t-1},\Theta\right)}_{\text{model at }t}\underbrace{f\left(\Theta|y\left(t-1\right)\right)}_{\text{posterior for }t-1}dy_{t}d\Theta$$

Again, for point estimates, we get

$$f\left(y_{t+1}|y\left(t-1\right)\right)=f\left(y_{t+1}|\hat{y}_{t},y\left(t-1\right),\hat{\Theta}_{t-1}\right)$$

where we laid $f\left(\Theta|y\left(t-1\right)\right)\doteq\delta\left(\Theta,\hat{\Theta}_{t-1}\right)$ and $f\left(y_{t}|y\left(t-1\right)\right)\doteq\delta\left(y_{t},\hat{y}_{t}\right)$ with $\hat{\Theta}_{t-1}$ and $\hat{y}_{t}$ being point estimates.

Remark

- *Here, both Θ and y_t are missing. We must supply both.*
- *The result for point estimates is very natural. We use model with substituted point estimates for the unknown variables Θ and y_t*
- *Comparing both the results (for pdfs and point estimates) we can see the basic principle of Bayesian estimation. In the former approach (with pdfs) all values of the missing*

unknown variable (Θ and y_t) are substituted and they are weighted by its probability (e.g. for Θ: $\int \underbrace{\Theta}_{value} \underbrace{f\left(\Theta|y\left(t-1\right)\right)}_{probability} d\Theta$). In the variant with point estimates, first the point estimates are computed somewhere aside and then they are substituted for the unknown variables.

7.3 Multi-step prediction

Here, we estimate y_{t+k} being at time t and with the knowledge of $y\left(t-1\right)$. The procedure is still the same, only the complexity (too many integrals) arise.

However, if we use points estimates for the unknown variables and want to determine only point estimate of the future output, we come to simple formulas as in (7.2). We can start at the present time and perform one step prediction until we reach the time, for which we want the final multi-step prediction of the output.

Regression model with known parameters and point estimation

For a 1st order regression model $y_t = ay_{t-1} + bu_t + e_t$ with known parameters and point prediction we have

$$\begin{aligned}
y_t &= ay_{t-1} + bu_t + e_t \\
\hat{y}_t &= ay_{t-1} + bu_t \\
\hat{y}_{t+1} &= a\hat{y}_t + bu_{t+1} = a\left(ay_{t-1} + bu_t\right) + bu_{t+1} = \\
&= a^2 y_{t-1} + abu_t + bu_{t+1} \\
\hat{y}_{t+2} &= a\hat{y}_{t+1} + bu_{t+2} = \\
&= a^3 y_{t-1} + a^2 bu_t + abu_{t+1} + bu_{t+2} \\
&etc.
\end{aligned}$$

The point prediction can be achieved by a simple repetitive substitution of the model. For simulation, directly last estimates can be used.

Program

One step prediction with known model parameters. First prediction is based on measured actual value of y, the second one uses the prediction of y from the previous step.

```
// prg1stPre.sce
// One step prediction with the model
//     y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)  for y(t)=0
// Prediction is computed before the actual output is measured!
```

```
// ----------------------------------------
clc, clear, close, mode(0);

nd=80;                              // length of data
b0=.8; a1=.5; b1=1; sd=.05;         // model parameters
y(1)=0; yy=0;                       // initial conditions
u=.5+.1*rand(1,nd+1,'n');           // generation of control for all t

for t=2:nd                          // loop simulation and prediction
  // prediction
  yy=b0*u(t)+a1*y(t-1)+b1*u(t-1);  // zero step prediction
  yy=b0*u(t+1)+a1*yy+b1*u(t);      // one step prediction (repeated)
  yp(t+1)=yy;
   // simulation
  y(t)=b0*u(t)+a1*y(t-1)+b1*u(t-1)+sd*rand(1,1,'n');
end

// Results
plot(1:nd,y(1:nd),1:nd,yp(1:nd))
legend('y','yp');
title('One step prediction','fontsize',5)
```

Full prediction with regression model under condition of normality

If the model of the system is linear and normal, we can simply obtain even the full (probabilistic) prediction using the following trick based on the fact that the normality is preserved (the composition of two normal models is also normal).

We perform the one step prediction as in the previous case, but we preserve also the noise term. This term is during the prediction somehow cummulated. Then, with the final model of the prediction, we compute expectation and variance. With those characteristics we can express the whole predictive pdf as we know that it is normal. For he two step prediction we have

$$\begin{aligned} y_t &= ay_{t-1} + bu_t + e_t \\ y_{t+1} &= ay_t + bu_{t+1} + e_{t+1} = \\ &= a\left(ay_{t-1} + bu_t + e_t\right) + bu_{t+1} + e_{t+1} = \\ &= a^2 y_{t-1} + abu_t + bu_{t+1} + ae_t + e_{t+1} \\ y_{t+2} &= ay_{t+1} + bu_{t+2} + e_{t+2} = \\ &= a^3 y_{t-1} + a^2 bu_t + abu_{t+1} + bu_{t+2} + a^2 e_t + ae_{t+1} + e_{t+2} \end{aligned}$$

$\rightarrow$

$E\left[y_{t+2}|y\left(t-1\right)\right]=a^{3}y_{t-1}+a^{2}bu_{t}+abu_{t+1}+bu_{t+2}$

$D\left[y_{t+2}|y\left(t-1\right)\right]=D\left[a^{2}e_{t}+ae_{t+1}+e_{t+2}\right]=\left(a^{4}+a^{2}+1\right)r$

Predictive pdf

$$f\left(y_{t+2}|y\left(t-1\right)\right)=N_{y_{t+2}}\left(E\left[y_{t+2}|y\left(t-1\right)\right],D\left[y_{t+2}|y\left(t-1\right)\right]\right)$$

(Normal distribution is determined by its expectation and variance.)

7.4 Prediction with discrete model

Similarly as for regression model, the prediction with control variable is a bit complex. So again, we will consider the model without it in the form

$$f\left(y_{t}|y_{t-1},\Theta\right)=\Theta_{y_{t}|y_{t-1}}$$

y_{t-1}	$y_t=1$	$y_t=2$
1	$\Theta_{1\|1}$	$\Theta_{2\|1}$
2	$\Theta_{1\|2}$	$\Theta_{2\|2}$

with known parameter Θ.

Zero step prediction

It is done directly by the model

$$f\left(y_{t}|y_{t-1},\Theta\right)=\Theta_{y_{t}|y_{t-1}}$$

It means for $y_{t-1}=1$ we have

$$f\left(y_{t}|y_{t-1}=1\right)=\left[\Theta_{1|1},\,\Theta_{2|1}\right]$$

and for $y_{t-1}=2$ it is

$$f\left(y_{t}|y_{t-1}=2\right)=\left[\Theta_{2|1},\,\Theta_{2|2}\right]$$

Again, th prediction can be either the argument of the maximum parameter or the expectation

$$\hat{y}_{t}=1\Theta_{1|y_{t-1}}+2\Theta_{2|y_{t-1}}$$

for $y_{t-1}=1,\,2$.

Multi-steps prediction

Repeating the above procedure, we come to the following result for k-step prediction

$$f\left(y_{t+k}|y\left(t-1\right)\right)=\left(\Theta^{k+1}\right)_{y_{t+k}|y_{t-1}}$$

where Θ^{k+1} is the $(k+1)$th power of the matrix parameter Θ. The derivation is indicated in the following example.

Program

An example of a one step prediction is in the following program.

```
// prg1stDPre.sce
// One step prediction with a discrete model
//    f( y(t)|u(t),y(t-1) )  - known parameters
// Again, first prediction and then simulation (measurement of y(t))
// ----------------------------------------
clc, clear, close, mode(0);

nd=30;                              // length of data
th=[.99 .01                         // model parameters
    .98 .02
    .02 .98
    .01 .99];
y(1)=1; yy=1;                       // initial conditions

u=(rand(1,nd,'u')>.5)+1;            // generation of control
for t=2:nd                          // loop simulation and prediction
  // prediction
  j=2*(u(t)-1)+y(t-1);
  yy=sum(rand(1,1,'u')>cumsum(th(j,:)))+1;    // 0-step prediction
  j=2*(u(t)-1)+yy;
  yy=sum(rand(1,1,'u')>cumsum(th(j,:)))+1;    // 1-step prediction
  yp(t+1)=yy;
  // simulation
  j=2*(u(t)-1)+y(t-1);
  y(t)=sum(rand(1,1,'u')>cumsum(th(j,:)))+1;
end

// Results
plot(3:nd,y(3:nd),'x:',3:nd,yp(3:nd),'o:','markersize',8)
legend('y','yp');
```

```
title('One step prediction','fontsize',5)
set(gca(),'data_bounds',[2 nd+1 0.8 2.2])
```

Example

Two steps prediction

$$f\left(y_{t+2}|y\left(t-1\right)\right)=\sum_{y_{t+1}}\sum_{y_t}f\left(y_{t+2}|y_{t-1}\right)f\left(y_{t+1}|y_t\right)f\left(y_t|y_{t-1}\right)=$$

$$=\sum_{y_{t+1}}\Theta_{y_{t+2}|y_{t+1}}\sum_{y_t}\Theta_{y_{t+1}|y_t}\Theta_{y_t|y_{t-1}}=\left(\Theta^3\right)_{y_{t+2}|y_{t-1}}$$

For

$$\Theta=\left[\begin{array}{cc}0.4, & 0.6\\ 0.8, & 0.2\end{array}\right]$$

$$f\left(y_{t+2}|y\left(t-1\right)\right)=\left[\begin{array}{cc}0.4, & 0.6\\ 0.8, & 0.2\end{array}\right]^3=\left[\begin{array}{cc}0.544, & 0.456\\ 0.608, & 0.392\end{array}\right]$$

That is:

for $y_{t-1}=1$ we have $f\left(y_{t+2}|1\right)=[0.544,\ 0.456]$

for $y_{t-1}=2$ we have $f\left(y_{t+2}|2\right)=[0.608,\ 0.392]$

Point prediction based on argument maxima will be

$$\begin{aligned}\hat{y}_t&=1,\ \text{for } y_{t-1}=1\\ &=1,\ \text{for } y_{t-1}=2\end{aligned}$$

Point prediction as the expectation is

$$\begin{aligned}\hat{y}_t&=1\cdot 0.544+2\cdot 0.456=1.456,\ \text{for } y_{t-1}=1\\ &=1\cdot 0.608+2\cdot 0.392=1.392,\ \text{for } y_{t-1}=2.\end{aligned}$$

Program

Scilab program to this example in here

```
// prgNstDPre.sce
// np-step prediction with a discrete model
//    f( y(t)|y(t-1) )  - known parameters
// ---------------------------------------
clc, clear, close, mode(0);
```

```
nd=30;                              // length of data
np=3;                               // length of prediction
th=[.4 .6                           // model parameters
    .8 .2];
thN=th^np;
y(1:np)=1;                          // initial conditions
yp(1:np+1)=1;

for t=2:nd                          // loop simulation and prediction
  y(t)=sum(rand(1,1,'u')>cumsum(th(y(t-1),:)))+1;      // simulation
  yp(t+np)=sum(rand(1,1,'u')>cumsum(thN(y(t-1),:)))+1; // np-step prediction
end

// Results
plot(np+1:nd,y(np+1:nd),'x:',np+1:nd,yp(np+1:nd),'o:','markersize',8)
legend('y','yp');
title('One step prediction','fontsize',5)
set(gca(),'data_bounds',[2 nd+1 0.8 2.2])
```

8 State-space model, state estimation

We have already met the state equation of the state-space model connected with the state representation of regression model. In that case, the model was constructed form the measured values of the input and output. However, the real state variable an immeasurable one which must be estimated based on measured variables (input and output). The assumption made on the state is, that its present value depends only on control and its previous value. I.e. it holds

$$f\left(x_t|u\left(t\right),y\left(t-1\right),x\left(t-1\right)\right)=f\left(x_t|u_t,x_{t-1}\right)$$

which means, that its model is of the first order.

Similarly, knowing the value of x_t the output variable is modeled with only u_t not any older variables.

8.1 Model

The full state-space model consists of tho equations: state and output ones

$$\begin{array}{cc} f\left(x_t|x_{t-1},u_{t-1}\right) & \text{model of the state} \\ f\left(y_t|x_t,u_t\right) & \text{model of the output} \end{array}$$

is generated by the equations

$$\begin{aligned} x_t &= Mx_{t-1}+Nu_{t-1}+w_t \\ y_t &= Ax_t+Bu_t+v_t \end{aligned}$$

where M, N, A, B are matrices, w_t and v_t white noises with covariance matrices r_w and r_v.

Remark

The first equation form the state-space model gives evolution of the state itself, without any information from measurements. It is also called states prediction. The second equation serves as predictor of the output y_t in dependence on the state estimate x_t. If the prediction is good, the estimate is also good and vice versa. Thus it introduces measured data. It is called state filtration. This takes part in the state estimation to which the next paragraph is devoted.

8.2 Estimation

The state at each time is instant is a random variable. That is why, its description is its density function conditioned by measured data. However, this density function contains two time indexes - one says to what time the state belongs the other which data were used for its estimation. So, e.g. $f\left(x_t|d\left(t-1\right)\right)$ describes the statex_t (at time instant t) for which the information pf the data $d\left(t-1\right)$ (data from the beginning up to time $t-1$, i.e. old data)

have been used. The state description is evolved successively: first, using the state equation of the state-space model, the state is predicted one step ahead, i.e. from x_{t-1} to x_t. Then, in the filtration step using the output model, the data index is increased including measured output y_t updating $f\left(x_t|d\left(t-1\right)\right)$ to $f\left(x_t|d_t\right)$. The whole update is illustrated in the following scheme

$$f\left(x_{t-1}|d\left(t-1\right)\right) \underbrace{\rightarrow}_{\text{prediction}} f\left(x_t|d\left(t-1\right)\right) \underbrace{\rightarrow}_{\text{filtration}} f\left(x_t|d\left(t\right)\right)$$

Evolution

As we have said, the evolution of the state description is performed using the state-space model.

Prediction

The prediction of the state is performed in the standard way for predicting unknown variable according to (7.1) where the output has been predicted

$$f\left(x_t|d\left(t-1\right)\right) = \int_{x_{t-1}^*} f\left(x_t|x_{t-1},u_{t-1}\right) f\left(x_{t-1}|d\left(t-1\right)\right) \tag{8.1}$$

Filtration

It consists in application of the Bayes rule, similarly as for estimation of unknown parameter Θ (as indicated in the formula below)

$$f\left(\underbrace{x_t}_{\Theta}|d\left(t\right)\right) \propto \underbrace{f\left(y_t|x_t,u_t\right)}_{\text{model}} f\left(\underbrace{x_t}_{\Theta}|d\left(t-1\right)\right) \tag{8.2}$$

Remark

In the above derivation Natural Conditions of Control are used.

Kalman filter

The previous considerations were only theoretical. The formula derived holds for the whole distributions as functions. It cannot be practically used. Its conversion to numbers is performed further on.

For normal model and normal prior of the state distribution the normality is preserved. Functional recursion becomes algebraic one for expectations and covariance matrices.

Notation for normal model is

$$\begin{aligned} f\left(x_t|x_{t-1},u_t\right) &= N_{x_t}\left(Mx_{t-1}+Nu_t,\, r_w\right) \\ f\left(y_t|x_t,u_t\right) &= N_{y_t}\left(Ax_t+Bu_t,\, r_v\right) \end{aligned}$$

and for pdfs from the evolution

$$\begin{aligned} f\left(x_{t-1}|d\left(t-1\right)\right) &= N_{x_{t-1}}\left(x_{t-1|t-1},\, R_{t-1|t-1}\right) \\ f\left(x_t|d\left(t-1\right)\right) &= N_{x_t}\left(x_{t|t},\, R_{t|t}\right) \\ f\left(x_t|d\left(t\right)\right) &= N_{x_t}\left(x_{t|t},\, R_{t|t}\right) \end{aligned}$$

Substitution into the evolution equations (8.1) and (8.2) gives Kalman filter (KF)

Kalman filter

$x_{t|t-1} = Mx_{t-1|t-1} + Nu_t$ — state prediction

$R_{t|t-1} = r_x + MR_{t-1|t-1}M'$

$y_p = Ax_{t|t-1} + Bu_t$ — output prediction

$R_p = r_y + AR_{t|t-1}A'$

$R_{t|t} = R_{t|t-1} - R_{t|t-1}A'R_p^{-1}AR_{t|t-1}$

$K = R_{t|t}A'r_y^{-1}$ — Kalman gain

$x_{t|t} = x_{t|t-1} + K\left(y_t - y_p\right)$ — state correction

The filter starts with prior $x_{0|0}$ and $R_{0|0}$, uses data y_t, u_t, $t = 1, 2, \cdots, N$ and currently computes $x_{t|t}$ and $R_{t|t}$. The result is either point state estimate $x_{t|t}$ or the full distribution of the state $f\left(x_t|u_t, d\left(t\right)\right) = N_{x_t}\left(x_{t|t},\, R_t\right)$.

The program of the Kalman filter is here

```
function [xt,Rx,yp]=Kalman(xt,yt,ut,M,N,F,A,B,G,Rw,Rv,Rx)
  // Kalman filter for state estimtion with the model
  //                              xt = M*xt + N*ut + F + w
  //                              yt = A*xt + B*ut + G + v
  // xt    state
  // Rx    state estimate covariance matrix
  // yp    output prediction
```

```
  // yt    output
  // ut    input
  // M,N,F state model parameters
  // A,B,G output model parameters
  // Rw    state model covariance
  // Rv    output model covariance

  nx=size(M,1);
  ny=size(A,1);
  if isempty(F), F=zeros(nx,1); end
  if isempty(G), G=zeros(ny,1); end

  xt=M*xt+N*ut+F;                      // time update of the state
  Rx=Rw+M*Rx*M';                       // time updt. of state covariance

  yp=A*xt+B*ut+G;                      // output prediction
  Ry=Rv+A*Rx*A';                       // noise covariance update
  Rx=Rx-Rx*A'*inv(Ry)*A*Rx;            // state est. coariance update
  ey=yt-yp;                            // prediction error
  KG=Rx*A'*inv(Rv);                    // Kalman gain
  xt=xt+KG*ey;                         // data update of the state
endfunction
```

A program with the state estimation task is in T46statEst_KF.sce and Kalman.sci.

9 Nonlinear state estimation

The Kalman filter from the previous chapter holds only for linear state-space model with normal distribution of noises. If the model is nonlinear, first it must be linearized. Then the Kalman filter can be used.

9.1 Nonlinear model

Nonlinear model is given by nonlinear functions g and h, as follows

$$\begin{aligned} x_t &= g\left(x_{t-1}, u_t\right) + w_t \\ y_t &= h\left(x_t, u_t\right) + v_t \end{aligned}$$

Linearization

It is done using first two terms of Taylor expansion of the nonlinear functions (here g_1) at the point of the last point estimate of the state. For the state equation it is $\hat{x}_{t-1}$ and for the output equation it is $\hat{x}_t$.

For a general value x the expansion around the point $\hat{x}_{t-1}$ (last point estimate) reads

$$g\left(x, u_t\right) \doteq g\left(\hat{x}_{t-1}, u_t\right) + g'\left(\hat{x}_{t-1}, u_t\right)\left(x - \hat{x}_{t-1}\right)$$

$$h\left(x, u_t\right) \doteq h\left(\hat{x}_t, u_t\right) + h'\left(\hat{x}_t, u_t\right)\left(x - \hat{x}_t\right)$$

where g' and h' denote derivative of g and h.

Remarks

1. x_t and x_{t-1} are random variables. x is their general value, $\hat{x}_t$ and $\hat{x}_{t-1}$ are special values: $\hat{x}_t$ is the point estimate of x_t and $\hat{x}_{t-1}$ is point estimate of x_{t-1}.
2. *Linearization can be applied only to nonlinear parts of the model. The linear parts can stay as they are.*

The derivatives of generally vector functions g' and h' according to vector argument x are

$$g'\left(\hat{x}_{t-1}, u_t\right) = \left[\begin{array}{cccc} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ & & \cdots & \\ \frac{\partial g_n}{\partial x_1} & & \cdots & \frac{\partial g_n}{\partial x_n} \end{array}\right]_{|x=\hat{x}_{t-1}}, \quad h'\left(\hat{x}_t, u_t\right) = \left[\begin{array}{cccc} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ & & \cdots & \\ \frac{\partial h_m}{\partial x_1} & & \cdots & \frac{\partial h_m}{\partial x_n} \end{array}\right]_{|x=\hat{x}_t}$$

After substitution of the linearized functions g and h into the model, we have

$$\begin{aligned} x_t &\doteq g\left(\hat{x}_{t-1}, u_t\right) + g'\left(\hat{x}_{t-1}, u_t\right)\left(x - \hat{x}_{t-1}\right) + w_t \\ y_t &\doteq h\left(\hat{x}_t, u_t\right) + h'\left(\hat{x}_t, u_t\right)\left(x - \hat{x}_t\right) + v_t \end{aligned}$$

and for $x = x_{t-1}$ in the case of the state equation and $x = x_t$ for output equation we obtain the linearized model

$$\begin{aligned} x_t &= \bar{M} x_{t-1} + F + w_t \\ y_t &= \bar{A} x_t + G + v_t \end{aligned}$$

where

$$\bar{M} = g'\left(\hat{x}_{t-1}, u_t\right), \quad F = g\left(\hat{x}_{t-1}, u_t\right) - g'\left(\hat{x}_{t-1}, u_t\right)\hat{x}_{t-1},$$

$$\bar{A} = h'\left(\hat{x}_t, u_t\right), \quad G = h\left(\hat{x}_t, u_t\right) - h'\left(\hat{x}_t, u_t\right)\hat{x}_t.$$

Example

For two dimensional state x and scalar control u and output y

$$x_t = \left[\begin{array}{c} x_1 \\ x_2 \end{array}\right]_t, \;\; u_t, \;\; y_t$$

let us have the model in the form

$$\begin{aligned} x_{1;t} &= \exp\left\{-x_{1;t-1} - x_{2;t-1}\right\} + u_t + w_t \\ x_{2;t} &= x_{1;t-1} - 0.3u_t + w_{2;t} \\ y_t &= x_{2;t} + v_t \end{aligned}$$

Here

$$g = \left[\begin{array}{c} g_1 \\ g_2 \end{array}\right] = \left[\begin{array}{c} \exp\left\{-x_{1;t-1} - x_{2;t-1}\right\} + u_t \\ x_{1;t-1} - 0.3u_t \end{array}\right]$$

$$h = x_{2;t}$$

Only the first function g_1 is nonlinear and is necessary to be linearized

$$g_1\left(x, u_t\right) = \exp\left\{-x_1 - x_2\right\} + u_t$$

$$g_1'\left(x, u_t\right) = \left[\frac{\partial g_1}{\partial x_1}, \frac{\partial g_1}{\partial x_2}\right] = \left[-\exp\left\{-x_1 - x_2\right\}, -\exp\left\{-x_1 - x_2\right\}\right]$$

Fully linearized model is

$$\begin{aligned} x_{1;t} &= g_1^{'}(\hat{x}_{t-1}, u_t)\, x_{t-1} + g_1(\hat{x}_{t-1}, u_t) - g_1^{'}(\hat{x}_{t-1}, u_t)\, \hat{x}_{t-1} + w_t \\ x_{2;t} &= [1,\, 0]\, x_{t-1} - 0.3u_t + w_{2;t} \\ y_t &= [0,\, 1]\, x_t + v_t \end{aligned}$$

where

$$\bar{M} = \begin{bmatrix} g_1^{'}(\hat{x}_{t-1}, u_t) \\ [1,0] \end{bmatrix}, \quad F = \begin{bmatrix} g_1(\hat{x}_{t-1}, u_t) - g_1^{'}(\hat{x}_{t-1}, u_t)\, \hat{x}_{t-1} \\ -0.3u_t \end{bmatrix},$$

$$N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \bar{A} = [0,\, 1], \quad G = 0, \quad B = 0.$$

With this, we can use subroutine Kalman with in the following form

[xt,Rx,yp]=Kalman(xt,yt,ut,$\bar{M}$,N,F,$\bar{A}$,B,G,Rw,Rv,Rx)

To run the estimation, it is necessary to choose initial values of recursively recomputed variables x_t and R_x. $x_{t|t=0} = x_0$ ca be chosen e.g. as zero vector; the matrix R_x (initial covariance matrix of the state estimate) is usually chosen as diagonal matrix with big numbers on the diagonal.

A problem is with the choice of matrices R_w (covariance of the state noise) and R_v (covariance of the output noise). They should reflect the variability of the state and the output.

9.2 Model with unknown parameters

As we have indicated, for the state estimation we assume the parameters of th state-space model as known (including noise covariances). Here we are going to show, what can be done if some of these parameters are unknown.

The unknown parameters of the model are added to the state a and estimated together. However, the model becomes nonlinear - model matrices contain state entries and they are multiplied by state. So, the technique of linearization must be used, again.

Example

Model

$$\begin{aligned} x_t &= \exp\{-ax_{t-1}\} + bu_t + w_t \\ y_t &= x_t + v_t, \end{aligned}$$

where a and b are unknown.

We define new state

$$X_t = \left[x_t^{'},\, a,\, b\right]^{'},\;\; X_{t-1} = \left[x_{t-1}^{'},\, a,\, b\right]^{'}$$

and obtain new model

$$\begin{aligned} X_t &= \left[\begin{array}{c} \exp\left\{-X_{2;t-1}X_{1;t-1}\right\} + X_{3;t-1}u_t \\ X_{2;t-1} \\ X_{3;t-1} \end{array}\right] + \underbrace{\left[\begin{array}{c} w_t \\ \epsilon_{2;t} \\ \epsilon_{3;t} \end{array}\right]}_{W_t} \\ y_t &= [1,\, 0,\, 0]\, X_t + v_t \end{aligned}$$

Only the first equation is nonlinear, however, we will treat the whole model as nonlinear (it is well possible)

$$g = \left[\begin{array}{c} \exp\left\{-X_{2;t-1}X_{1;t-1}\right\} + X_{3;t-1}u_t \\ X_{2;t-1} \\ X_{3;t-1} \end{array}\right]_{X_{t-1}=\hat{X}_{t-1}}$$

$$g' = \left[\begin{array}{ccc} -X_{2;t-1}\exp\left\{-X_{2;t-1}X_{1;t-1}\right\}, & -X_{1;t-1}\exp\left\{-X_{2;t-1}X_{1;t-1}\right\}, & u_t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]_{X_{t-1}=\hat{X}_{t-1}}$$

Approximated model is

$$\begin{aligned} X_t &\doteq \underbrace{g'}_{\bar{M}} X_{t-1} + \underbrace{g - g'\hat{X}_{t-1}}_{F} + W_t \\ y_t &\doteq \underbrace{[1,\, 0,\, 0]}_{\bar{A}} X_t + v_t \end{aligned}$$

and $N = [0,\, 0,\, 0]^{'}$, $B = 0$, $G = 0$.

For the estimation of the linearized model the Kalman filter can be used in the form

$$[\mathsf{x},\, \mathsf{Rx},\, \mathsf{yp}] = \mathsf{Kalman}(\mathsf{x},\, \mathsf{y},\, \mathsf{u},\, \bar{M},\, \mathsf{N},\, \mathsf{F},\, \bar{A},\, \mathsf{B},\, \mathsf{G},\, \mathsf{Rw},\, \mathsf{Rv},\, \mathsf{Rx})$$

10 Control with regression model

The previous tasks (prediction and state estimation) concerned learning about the reality of interest. This task (control) is about influencing the reality so that it would behave according to out wishes. The control, we are going to speak about is to be an optimal one. So, a criterion evaluating the control process must be defined. The control variable then is constructed so that the value of this criterion would be minimal. The control is designed for N steps ahead for so called control interval.

10.1 Derivation of the control in pdf

Criterion

W will use the criterion in the form of a sum

$$J = \sum_{t=1}^{N} J_t \tag{10.1}$$

where J_t is a penalization for time t. Mostly it is $J_t = y_t^2 + \omega u_t^2$.

We want to construct $u_t,\ t = 1, 2, \cdots, N$ that minimizes J. But, J is a random variable, due to the output y_t. As a random variable it can take many different values. That is why it is not possible to speak about its direct minimization. So, we must minimize its estimate (which is expectation). So the minimized criterion is

$$E\left[J|d\left(0\right)\right] = E\left[\sum_{t=1}^{N} J_t|d\left(0\right)\right]$$

where in condition of the expectation is our preliminary knowledge (from prior data or an expert).

Remark

For $N = 1$ we obtain one-step control. Here, we optimize control only for the next output. This control is dangerous, because the controller does not take into account future evolution of the system and to act best way in one step it can generate too big outputs. This can excite the system so much that it is not possible even to stabilize it in the future steps and the control fails.

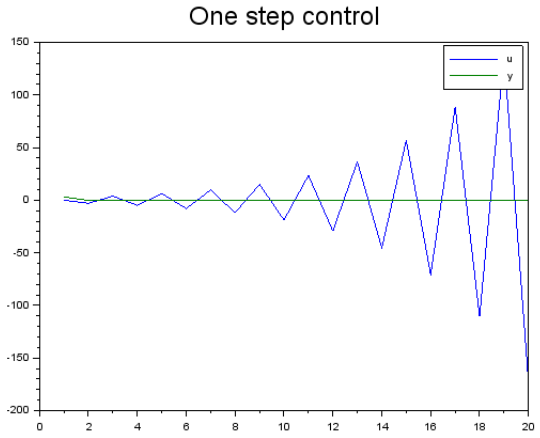
The program for this control is very simple - for the model

$$y_t = b_0 u_t + a_1 y_{t-1} + b_1 u_{t-1} + e_t$$

it is here

```
// prg1stCon.sce
// One step control demonstrating danger of this strategy
// y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)  for y(t)=0
// must be  u(t)=-1/b0.( a1.y(t-1)+b1.u(t-1) )
// ----------------------------------------

nd=20;
b0=.8; a1=.8; b1=1;
y(1)=3; u(1)=0;

for t=2:nd
  u(t)=-1/b0*(a1*y(t-1)+b1*u(t-1));
  y(t)=b0*u(t)+a1*y(t-1)+b1*u(t-1)+.1*rand(1,1,'n');
end

plot(1:nd,u,1:nd,y)
legend('u','y');
title('One step control','fontsize',5)
```

with the result



Here, value of the control grows, output still remains at zero. However, the magnitude of the control grows to infinity and due to roundup errors the output will pass away, too.

Indication of criterion minimization

The procedure is recurrent. It starts at the end of the control interval N and subsequently

$$\min_{u_{1:N}} E\left[\varphi^*_{N+1} + \sum_{t=1}^{N} J_t | d(0)\right] =$$

$$= \min_{u_{1:(N-1)}} E\left[\min_{u_N} \underbrace{E\left[\varphi^*_{N+1} + J_N | u_N, d(N-1)\right]}_{\varphi^*_N} + \sum_{t=1}^{N-1} J_t \Big| d(0)\right] =$$

$$= \min_{u_{1:(N-1)}} E\left[\min_{u_N} \varphi_N + \sum_{t=1}^{N-1} J_t | d(0)\right] = \min_{u_{1:N}} E\left[\varphi^*_N + \sum_{t=1}^{N-1} J_t | d(0)\right]$$

which reproduces the initial form, only with $N \to N-1$ and where (due to the reproduction in general form)

Bellman equations

$$\varphi_t = E\left[\varphi^*_{t+1} + J_t | u_t, d(t-1)\right] \quad \text{expectation} \tag{10.2}$$

$$\varphi^*_t = \min_{u_t} \varphi_t \quad \text{minimization} \tag{10.3}$$

for $t = N, N-1, N-2, \cdots, 1$. Each minimization gives the formula for optimal control - it is $u_t = \arg\min \varphi_t (d(t-1))$. However, ti cannot be used immediately, because the data $d(t-1)$ is not known, yet. Only at time $t = 1$ we need data $d(0)$ and the control can start to be generated.

Remark

The operator form of expectation we have used for criterion minimization is elegant but not very lucid. In the Appendix 13.5 we show the minimization in an integral form. It is longer but better for understanding.

10.2 Derivation for normal regression model

Regression model can be converted to the state form (see Section3.1). The resulting model is

$$x_t = Mx_{t-1} + Nu_t + w_t$$

where $x_t = [y_t, u_t, y_{t-1}, u_{t-1}, \cdots y_{t-n+1}, u_{t-n+1}]'$.

Taking into account the form of the state, the penalty in the criterion (10.1) can be written as

$$J_t = y_t^2 + \omega u_t^2 = x_t' \Omega x_t \tag{10.4}$$

where Ω is a diagonal matrix

$$\Omega = \begin{bmatrix} 1 & & & & \\ & \omega & & & \\ & & 0 & & \\ & & & \cdots & \\ & & & & 0 \end{bmatrix}$$

Now the model and criterion is used in general Bellman equations, where we guess the form of $\varphi^*_{t+1} = x_t^{'} R_{t+1} x_t$

$$E\left[x_t^{'} R_{t+1} x_t + x_t^{'} \Omega x_t | u_t, d\left(t-1\right)\right] = E\left[x_t^{'} U x_t\right] =$$

where $U = R_{t+1} + \Omega$

$$= \left(M x_{t-1} + N u_t\right)^{\prime} U \left(M x_{t-1} + N u_t\right) + \rho =$$

where the expectation is performed by
substitution th model for x_t and ρ involves
variance of noise

$$= x_{t-1}^{'} \underbrace{M^{\prime} U M}_{C} x_{t-1} + 2 u_t^{'} \underbrace{N^{\prime} U M}_{B} x_{t-1} + u_t^{'} \underbrace{N^{\prime} U N}_{A} u_t + \rho =$$

this is a quadratic form that will
be completed to square to determine
minimum and the rest after minimization

$$= u_t^{'} A u_t + 2 u_t^{'} A \underbrace{A^{-1} B}_{S_t} x_{t-1} + x_{t-1}^{'} S_t^{'} A S_t x_{t-1} +$$
$$+ \underbrace{x_{t-1}^{'} C x_{t-1} - x_{t-1}^{'} S_t^{'} A S_t x_{t-1}}_{x_{t-1} R_t x_{t-1}} + \rho =$$

this is the completion to square

$$= \left(u_t + S_t x_{t-1}\right)^{\prime} A \left(u_t + S_t x_{t-1}\right) + x_{t-1}^{'} R_t x_{t-1} + \rho$$

and this is the result of completion:
square and the reminder

Optimal control is $u_t = S_t x_{t-1}$.

Recursion for the control synthesis

$R_{N+1} = 0$

for $t = N, N-1, \cdots, 1$

$U = R_{t+1} + \Omega$
$A = N'UN$
$B = N'UM$
$C = M'UM$
$S_t = A^{-1}B$
$R_t = C - S_t' A S_t$
$u_t = S_t x_{t-1}.$

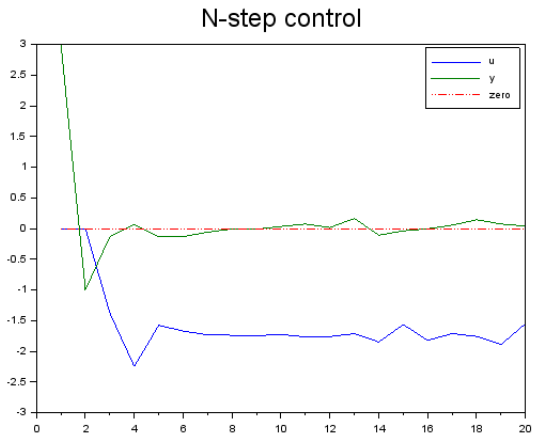
end

The **program** realizing this control is here

```
// PrgNstCon.sce
// N step control with regression model of the second order
// y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)+a2.y(t-2)+b2.u(t-2)+k+e(t)
// ---------------------------------------------------------------
clc, clear, close, mode(0)

nd=20;                              // length of control interval
b0=1; a1=.8; b1=.6; a2=-.3; b2=.1; k=3; sd=.1; // parameters
y(1)=3; y(2)=-1; u(1)=0; u(2)=0;    // initial data
om=.1;                              // penaluzation of u

M= [a1 b1 a2 b2 k                   // state model parameter
    0  0  0  0  0
    1  0  0  0  0
    0  1  0  0  0
    0  0  0  0  1];
N= [b0 1  0  0  0]';                // output model parameter
Om=[1  0  0  0  0                   // state penalization
    0  om 0  0  0
    0  0  0  0  0
    0  0  0  0  0
    0  0  0  0  0];
R=zeros(5,5);                       // initialization of optimization

S=list();
for t=nd:-1:2                       // loop of optimization
  U=R+Om;
  A=N'*U*N;
```

```
  B=N'*U*M;
  C=M'*U*M;
  S(t)=inv(A)*B;                    // control law
  R=C-S(t)'*A*S(t);                 // remainder (withou noise cov.)
end

for t=3:nd                          // loop of control
  x=[y(t-1) u(t-1) y(t-2) u(t-2) 1]'; // state x(t-1)
  u(t)=-S(t)*x;                     // control
  e(t)=sd*rand(1,1,'n');            // noise
  y(t)=b0*u(t)+a1*y(t-1)+b1*u(t-1)+a2*y(t-2)+b2*u(t-2)+k+e(t); // output
end

// Results
plot(1:nd,u,1:nd,y,[1 nd],[0 0],':')
legend('u','y','zero');
title('N-step control','fontsize',5)
```

with the result



Remark

The penalty function (10.4) *can be very easily extended to the following form*

$$(y_t - s_t)^2 + \omega u_t^2 + \lambda (u_t - u_{t-1})^2$$

where the first term leads to the following the output y_t *the prescribed set-point* s_t *and the last term introduces penalization of increments of the control variable. Penalizing the control*

increments calms control behavior and at the same time it does not result to steady-state deviation of the output and the set-point as it is when penalizing the whole control variable.

The solution how to introduce the above requirements for the control lies in construction of the penalization matrix as follows

$$\Omega = \begin{bmatrix} 1 & & & & & & -1 \\ & \omega+\lambda & & -\lambda & & & \\ & & 0 & & & & \\ & -\lambda & & \lambda & & & \\ & & & & \cdots & & \\ & & & & & 0 & \\ -1 & & & & & & 1 \end{bmatrix}$$

which is evident if we take into account that the criterion is

$$x_t^{'} \Omega x_t$$

and $x_t = [y_t,\, u_t,\, y_{t-1},\, u_{t-1},\, \cdots, 1]$.

11 Control with categorical model

We will show the synthesis for the controlled coin with memory.

$$\begin{aligned} &\text{model} \quad f\left(y_t|u_t, y_{t-1}\right) \\ &\text{penalty} \quad J_{y_t|u_t,y_{t-1}} \end{aligned}$$

for three steps control, i.e. for $t = 1, 2, 3$ and the following model and penalization

model (Θ)

u_3, y_2	$y_3 = 1$	$y_3 = 2$
1, 1	0.7	0.3
1, 2	0.2	0.8
2, 1	0.9	0.1
2, 2	0.4	0.6

penalty (J)

u_3, y_2	$y_3 = 1$	$y_3 = 2$
1, 1	0	1
1, 2	1	0
2, 1	1	2
2, 2	2	1

11.1 Optimization

Step for $t = 3$: $\varphi_4^* = 0$

Expectation

$$\varphi_3 = E\left[J|u_3, d\left(2\right)\right] = \sum_{y_3=1}^{2} J_{y_3|u_3,y_2}\Theta_{y_3|u_2,y_2} =$$

$$= \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} .* \begin{bmatrix} 0.7 \\ 0.2 \\ 0.9 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix} .* \begin{bmatrix} 0.3 \\ 0.8 \\ 0.1 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.2 \\ 1.1 \\ 1.4 \end{bmatrix} \begin{matrix} \cdots & u_3 = 1,\, y_2 = 1 \\ \cdots & u_3 = 1,\, y_2 = 2 \\ \cdots & u_3 = 2,\, y_2 = 1 \\ \cdots & u_3 = 2,\, y_2 = 2 \end{matrix}$$

Minimization

$$\begin{aligned} &\text{for}: \quad y_2 = 1 \rightarrow \min\left\{0.3,\, 1.1\right\} = 0.3 \;\; \text{for}\, u_3 = 1 \\ &\text{for}: \quad y_2 = 2 \rightarrow \min\left\{0.2,\, 1.4\right\} = 0.2 \;\; \text{for}\, u_3 = 1 \end{aligned}$$

$\rightarrow$

$$u_3 = \begin{cases} 1 & \text{for } y_2 = 1 \\ 1 & \text{for } y_2 = 2 \end{cases}$$

and reminder after minimization

$$\begin{array}{cc} y_2=1 & y_2=2 \\ \hline 0.3 & 0.2 \end{array} \forall u_2, y_1 \quad \rightarrow \quad \begin{array}{c|cc} u_2,\, y_1 & y_2=1 & y_2=2 \\ \hline 1,\,1 & 0.3 & 0.2 \\ 1,\,2 & 0.3 & 0.2 \\ 2,\,1 & 0.3 & 0.2 \\ 2,\,2 & 0.3 & 0.2 \end{array} = \varphi_3^*$$

Step for $t = 2$:

Expectation

$$\varphi_2 = E\left[J + \varphi_3^* | u_2, d\left(1\right)\right] = \sum_{y_2=1}^{2} \left(J_{y_2|u_2,y_1} + \varphi^*_{3;y_2|u_2,y_1}\right) \Theta_{y_2|u_2,y_1} =$$

$$= \left(\left[\begin{array}{c} 0 \\ 1 \\ 1 \\ 2 \end{array} \right] + \left[\begin{array}{c} 0.3 \\ 0.3 \\ 0.3 \\ 0.3 \end{array} \right] \right) . \varphi_2^* = 0 * \left[\begin{array}{c} 0.7 \\ 0.2 \\ 0.9 \\ 0.4 \end{array} \right] + \left(\left[\begin{array}{c} 1 \\ 0 \\ 2 \\ 1 \end{array} \right] + \left[\begin{array}{c} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{array} \right] \right) . * \left[\begin{array}{c} 0.3 \\ 0.8 \\ 0.1 \\ 0.6 \end{array} \right] =$$

$$= \left[\begin{array}{c} 0.8 \\ 0.7 \\ 1.6 \\ 1.9 \end{array} \right] \begin{array}{ll} \cdots & u_2 = 1,\, y_1 = 1 \\ \cdots & u_2 = 1,\, y_1 = 2 \\ \cdots & u_2 = 2,\, y_1 = 1 \\ \cdots & u_2 = 2,\, y_1 = 2 \end{array}$$

Minimization

$$\begin{array}{ll} \text{for}: & y_1 = 1 \rightarrow \min\left\{0.8,\, 1.6\right\} = 0.8 \;\; \text{for}\, u_2 = 1 \\ \text{for}: & y_1 = 2 \rightarrow \min\left\{0.7,\, 1.9\right\} = 0.7 \;\; \text{for}\, u_2 = 1 \end{array}$$

$\rightarrow$

$$u_2 = \begin{cases} 1 & \text{for } y_1 = 1 \\ 1 & \text{for } y_1 = 2 \end{cases}$$

and reminder after minimization

$$\begin{array}{cc} y_1=1 & y_1=2 \\ \hline 0.8 & 0.7 \end{array} \forall u_1, y_0 \quad \rightarrow \quad \begin{array}{c|cc} u_1,\, y_0 & y_1=1 & y_1=2 \\ \hline 1,\,1 & 0.8 & 0.7 \\ 1,\,2 & 0.8 & 0.7 \\ 2,\,1 & 0.8 & 0.7 \\ 2,\,2 & 0.8 & 0.7 \end{array} = \varphi_2^*$$

Step for $t = 1$:

Expectation

$$\varphi_1 = E\left[J + \varphi_2^* | u_1, d\left(0\right)\right] = \sum_{y_1=1}^{2} \left(J_{y_1|u_1,y_0} + \varphi^*_{2;y_1|u_1,y_0}\right) \Theta_{y_1|u_1,y_0} =$$

$$= \left(\begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.8 \\ 0.8 \\ 0.8 \\ 0.8 \end{bmatrix} \right) .* \begin{bmatrix} 0.7 \\ 0.2 \\ 0.9 \\ 0.4 \end{bmatrix} + \left(\begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.7 \\ 0.7 \\ 0.7 \\ 0.7 \end{bmatrix} \right) .* \begin{bmatrix} 0.3 \\ 0.8 \\ 0.1 \\ 0.6 \end{bmatrix} =$$

$$= \begin{bmatrix} 1.8 \\ 1.7 \\ 2.6 \\ 2.9 \end{bmatrix} \begin{array}{ll} \cdots & u_1 = 1,\, y_0 = 1 \\ \cdots & u_1 = 1,\, y_0 = 2 \\ \cdots & u_1 = 2,\, y_0 = 1 \\ \cdots & u_1 = 2,\, y_0 = 2 \end{array}$$

Minimization

$$\text{for}: \quad y_0 = 1 \rightarrow \min\{1.8,\, 2.6\} = 1.8 \;\; \text{for}\, u_1 = 1$$

$$\text{for}: \quad y_1 = 2 \rightarrow \min\{1.7,\, 2.9\} = 1.7 \;\; \text{for}\, u_1 = 1$$

$\rightarrow$

$$u_1 = \begin{cases} 1 & \text{for } y_0 = 1 \\ 1 & \text{for } y_0 = 2 \end{cases}$$

and reminder after minimization

$y_0 = 1$	$y_0 = 2$
1.8	1.7

11.2 Application

For $t = 0$ let us have $y_0 = 2$.

For $y_0 = 2$ we have $u_1 = 1$; $\rightarrow$ [1, 2] $\Theta_{1,2}$ =[0.2, 0.8] let us obtain $y_1 = 2$

For $y_1 = 2$ we have $u_2 = 1$; $\rightarrow$ [1, 2] $\Theta_{1,2}$ =[0.2, 0.8] let us obtain $y_2 = 1$

For $y_2 = 1$ we have $u_3 = 1$; $\rightarrow$ [1, 1] $\Theta_{1,1}$ =[0.7, 0.3] let us obtain $y_3 = 2$

The final value of criterion is $J_{2|12} + J_{1|12} + J_{2|11} = 0 + 1 + 1 = 2$.

12 Adaptive control

In the previous two Chapters we dealt with control algorithms based on model with known parameters. These algorithms were optima with respect to the chosen criterion.

If the model used in control has unknown parameters, the situation is much more difficult. The reason is that the design of the control law runs from th end of the control interval against the time while the algorithm for estimation of unknown parameters goes from the beginning in the direction of time. This leads to very complex optimal algorithm that even cannot be implemented in a computer. We say that it is unfeasible.

That is why some suboptimal algorithm instead of the optimal one must be used. Most frequently, the **algorithm with receding horizon** is used. This algorithm runs as follows:

At the beginning we have prior data and prior estimates of parameters. Then at time t

1. Design the control for some control interval $(t, t+T)$ using the current estimates of parameters.
2. Use the control law for the current time and compute u_t.
3. Apply this control u_t and measure the output y_t.
4. Use the new data $d(t)$ for recomputation of the parameters.
5. Shift the time $t \to t+1$ and go to 1

The program is in th Paragraph 15.15.

13 Appendix

13.1 Elementary differential equations

First order equations

The first order homogeneous differential equation with constant coefficients has the form

$$y' + a_0 y = 0, \qquad y(0) = \tilde{y}_0 \tag{13.1}$$

Characteristic equation is linear with unique solution

$$\lambda + a_0 = 0 \quad \to \quad \lambda = -a_0 \tag{13.2}$$

The solution to the differential equation (13.1) is

$$y(t) = \tilde{y}_0 \mathrm{e}^{\lambda t} \tag{13.3}$$

Second order equations

The second order homogeneous differential equation with constant coefficients has the form

$$y'' + a_1 y' + a_0 y = 0, \qquad y(0) = \tilde{y}_0,\ y'(0) = \tilde{y}_1 \tag{13.4}$$

Characteristic equation is quadratic

$$\lambda^2 + a_1 \lambda + a_0 = 0 \tag{13.5}$$

with the following types of solution

1. **Two different real roots** λ_1 and λ_2

   The equation (13.4) is

   $$y'' - (\lambda_1 + \lambda_2) y' + \lambda_1 \lambda_2 y = 0$$

   The solution is

   $$y(t) = c_1 \mathrm{e}^{\lambda_1 t} + c_2 \mathrm{e}^{\lambda_2 t}, \tag{13.6}$$

   where the coefficients c can be obtained as a solution of the set of linear equations

   $$\begin{aligned} c_1 + c_2 &= \tilde{y}_0 \\ \lambda_1 c_1 + \lambda_2 c_2 &= \tilde{y}_1 \end{aligned}$$

   which gives the solution

   $$c_1 = (\lambda_2 y_0 - y_0')/(\lambda_2 - \lambda_1) \ \text{ and } \ c_2 = (\lambda_1 y_0 - y_0')/(\lambda_1 - \lambda_2)$$

2. **One real double root** λ

   The equation (13.4) is

   $$y'' - 2\lambda y' + \lambda^2 y = 0$$

   The solution is

   $$y(t) = c_1 \mathrm{e}^{\lambda t} + c_2 t \mathrm{e}^{\lambda t}, \tag{13.7}$$

   where the coefficients c can be obtained as a solution of the set of linear equations

   $$\begin{aligned} c_1 &= \tilde{y}_0 \\ \lambda c_1 + c_2 &= \tilde{y}_1 \end{aligned}$$

   which gives the solution

   $$c_1 = \tilde{y}_0 \ \text{ and } \ c_2 = \tilde{y}_1 - \lambda \tilde{y}_0$$

3. **Two complex roots** $\lambda_1 = \rho + \omega\mathrm{i}$ and $\lambda_2 = \rho - \omega\mathrm{i}$

   The equation (13.4) is

   $$y'' - 2\rho y' + \rho^2 + \omega^2 = 0$$

   The solution is

   $$y(t) = c_1 \mathrm{e}^{\rho t} \cos(\omega t) + c_2 \mathrm{e}^{\rho t} \sin(\omega t), \tag{13.8}$$

   where the coefficients c can be obtained as a solution of the set of linear equations

   $$\begin{aligned} c_1 &= \tilde{y}_0 \\ \rho c_1 + \omega c_2 &= \tilde{y}_1 \end{aligned}$$

   which gives the solution

   $$c_1 = \tilde{y}_0 \quad \text{and} \quad c_2 = (\tilde{y}_1 - \rho\tilde{y}_0)/\omega$$

13.2 Elementary difference equations

Here, we will consider discrete time k for which it holds $t = kT$, where t is continuous time and T is a fix period of sampling.

First order equations

The first order homogeneous difference equation with constant coefficients has the form

$$y_{k+1} + a_0 y_k = 0, \qquad y_0 = \tilde{y}_0 \tag{13.9}$$

Characteristic equation is linear with unique solution

$$\lambda + a_0 = 0 \quad \rightarrow \quad \lambda = -a_0 \tag{13.10}$$

The solution to the differential equation (13.9) is

$$y_k = \tilde{y}_0 . \lambda^k \tag{13.11}$$

Second order equations

The second order homogeneous difference equation with constant coefficients has the form

$$y_{k+2} + a_1 y_{k+1} + a_0 y = 0, \qquad y_0 = \tilde{y}_0, \ = y_1 = \tilde{y}_1 \tag{13.12}$$

Characteristic equation is quadratic

$$\lambda^2 + a_1 \lambda + a_0 = 0 \tag{13.13}$$

with the following types of solution

1. **Two different real roots** λ_1 and λ_2

   The equation (13.12) is

$$y_{k+2} - (\lambda_1 + \lambda_2)y_{k+1} + \lambda_1\lambda_2 y = 0$$

   The solution is

$$y_k = c_1\lambda_1^k + c_2\lambda_2^k, \tag{13.14}$$

   where the coefficients c can be obtained as a solution of the set of linear equations

$$\begin{aligned} c_1 + c_2 &= \tilde{y}_0 \\ \lambda_1 c_1 + \lambda_2 c_2 &= \tilde{y}_1 \end{aligned}$$

   which gives the solution

$$c_1 = (\lambda_2\tilde{y}_0 - \tilde{y}_1)/(1 - \lambda_1) \quad \text{and} \quad c_2 = (\lambda_1\tilde{y}_0 - \tilde{y}_1)/(1 - \lambda_2)$$

2. **One real double root** λ

   The equation (13.12) is

$$y_{k+2} - 2\lambda y_{k+1} + \lambda^2 y = 0$$

   The solution is

$$y_k = c_1\lambda^k + c_2 k\lambda^k, \tag{13.15}$$

   where the coefficients c can be obtained as a solution of the set of linear equations

$$\begin{aligned} c_1 &= \tilde{y}_0 \\ \lambda c_1 + \lambda c_2 &= \tilde{y}_1 \end{aligned}$$

   which gives the solution

$$c_1 = \tilde{y}_0 \quad \text{and} \quad c_2 = \tilde{y}_1/\lambda - \tilde{y}_0$$

3. **Two complex roots** $\lambda_1 = \rho + \omega\mathrm{i}$ and $\lambda_2 = \rho - \omega\mathrm{i}$

   The equation (13.12) is

$$y_{k+2} - 2\rho y_{k+1} + \rho^2 + \omega^2 = 0$$

   The solution is

$$y_k = |c|^k[c_1\cos(\omega k) + c_2\sin(\omega k)], \tag{13.16}$$

   where the coefficients c can be obtained as a solution of the set of linear equations

$$\begin{aligned} c_1 &= \tilde{y}_0 \\ c_1|Re\lambda| + c_2|Im\lambda| &= \tilde{y}_1 \end{aligned}$$

   which gives the solution

$$c_1 = y_0 \quad \text{and} \quad c_2 = (\tilde{y}_1 - \tilde{y}_0|Re\lambda|)/|Im\lambda|$$

13.3 Discretization of a continuous model

Our aim is to construct a discrete regression model whose output is a sampled output of the corresponding continuous one - homogeneous differential equation of 1st or 2nd order with constant coefficients. We will call this task **discretization**.

Let us denote the continuous time by t and the discrete time by k. It holds

$$t = kT, \qquad T \text{ is a period of sampling.}$$

First order equation

Consider a homogeneous differential equation with constant coefficient

$$y' + a_0 y = 0, \qquad y_0 = \tilde{y}_0. \tag{13.17}$$

Then the equivalent difference equation (whose response is the sampled response to the differential one) is

$$y_{k+1} = A_0 y_k, \qquad \text{where} \quad A_0 = \mathrm{e}^{-a_0 T}. \tag{13.18}$$

Solution: The solution to the differential equation is

$$y_t = \tilde{y}_0 \mathrm{e}^{-a_0 t}.$$

To get the discrete version of the solution, we set $t = k$ for actual sample and $t + T = k + 1$ for the shifted one. So, for the actual sample, the solution the same but the substitution k for t and for the shifted sample it holds

$$y_{k+1} = y_{t+T} = \tilde{y}_0 \mathrm{e}^{-a_0(t+T)} = \tilde{y}_0 \mathrm{e}^{-a_0 t} \mathrm{e}^{-a_0 T} = \mathrm{e}^{-a_0 T} y_k$$

which proves (13.18). ◁

Second order equation

- ***Two distinct real roots***

  Let us consider a homogeneous differential equation with constant coefficients

  $$y'' + a_1 y' + a_0 y = 0, \qquad y_0 = \tilde{y}_0,\ y_0' = \tilde{y}_1 \tag{13.19}$$

  whose characteristic equation $\lambda^2 + a_1 \lambda + a_0 = 0$ has two different real roots λ_1 and λ_2.

  Then the equivalent difference equation (whose response is the sampled response to the differential one) is

  $$y_{k+2} = A_1 y_{k+1} + A_0 y_k, \tag{13.20}$$

  where

  $$A_1 = \mathrm{e}^{\lambda_1 T} + \mathrm{e}^{\lambda_2 T}, \qquad A_0 = -\mathrm{e}^{(\lambda_1 + \lambda_2) T} \tag{13.21}$$

Solution: A response to the considered continuous model is

$$y_t = c_1 \mathrm{e}^{\lambda_1 t} + c_2 \mathrm{e}^{\lambda_2 t}.$$

Sampling with $t = kT$ and the denotation $y_k = y_{kT}$ gives

$$y_k = c_1 \mathrm{e}^{\lambda_1 kT} + c_2 \mathrm{e}^{\lambda_2 kT}$$

This sampled response must obey the difference equation

$$y_{k+2} = A_1 y_{k+1} + A_0 y_k.$$

We express still the shifted responses

$$\begin{aligned} y_{k+1} &= c_1 \mathrm{e}^{\lambda_1 kT} \mathrm{e}^{\lambda_1 T} + c_2 \mathrm{e}^{\lambda_2 kT} \mathrm{e}^{\lambda_2 T} \\ y_{k+2} &= c_1 \mathrm{e}^{\lambda_1 kT} \mathrm{e}^{\lambda_1 2T} + c_2 \mathrm{e}^{\lambda_2 kT} \mathrm{e}^{\lambda_2 2T} \end{aligned}$$

and notice that they all are expressed in the basis with items $\mathrm{e}^{\lambda_1 kT}$ and $\mathrm{e}^{\lambda_2 kT}$. Thus we substitute into the difference equation and obtain

$$c_1 \mathrm{e}^{\lambda_1 kT} \mathrm{e}^{\lambda_1 2T} + c_2 \mathrm{e}^{\lambda_2 kT} \mathrm{e}^{\lambda_2 2T} = A_1 (c_1 \mathrm{e}^{\lambda_1 kT} \mathrm{e}^{\lambda_1 T} + c_2 \mathrm{e}^{\lambda_2 kT} \mathrm{e}^{\lambda_2 T}) + A_0 (c_1 \mathrm{e}^{\lambda_1 kT} + c_2 \mathrm{e}^{\lambda_2 kT}).$$

The coefficients B and A will be obtained by the comparison of items with the same basis element. We obtain the following system of equations

$$\begin{aligned} c_1 \mathrm{e}^{\lambda_1 2T} &= A_1 c_1 \mathrm{e}^{\lambda_1 T} + A_0 c_1 \\ c_2 \mathrm{e}^{\lambda_2 2T} &= A_1 c_2 \mathrm{e}^{\lambda_1 T} + A_0 c_2. \end{aligned}$$

The coefficients c get canceled (what is important) and the solution to this system is just what we want to prove. ◁

- ***One double root***

  Let the characteristic equation of (13.19) has one two-fold solution $\lambda = \lambda_1$.

  Then the equivalent difference equation (13.19) has the coefficients

$$A_1 = 2\mathrm{e}^{\lambda_1 T}, \qquad A_0 = -\mathrm{e}^{2\lambda_1 T} \tag{13.22}$$

  Solution: The proof is formally the same as for the two distinct roots, only with basis elements $\mathrm{e}^{\lambda_1 kT}$ and $kT\mathrm{e}^{\lambda_1 kT}$.

  The response to the continuous system is

$$y_k = c_1 \mathrm{e}^{\lambda_1 t} + c_2 t \mathrm{e}^{\lambda_1 t}$$

  After expressing the sampled response and its shifted variants, substituting into (13.20) and comparing the terms at the individual basis items, we obtain the following set of equations

$$\begin{aligned} c_1 \mathrm{e}^{2\lambda_1 T} + 2c_2 T \mathrm{e}^{2\lambda_1 T} &= A_1 c_1 \mathrm{e}^{\lambda_1 T} + A_1 T c_2 \mathrm{e}^{\lambda_1 T} + A_0 c_1 \\ c_2 \mathrm{e}^{2\lambda_1 T} &= A_1 c_2 \mathrm{e}^{\lambda_1 T} + A_0 c_2 \end{aligned}$$

  Again, the solution is just what we wanted to proof. ◁

- ***Two complex roots***

  Let the characteristic equation of (13.19) has two complex roots $\lambda_1 = \rho + \omega\mathrm{i}$ and $\lambda_2 = \rho - \omega\mathrm{i}$.

  Then the equivalent difference equation (13.19) has the coefficients

  $$A_1 = 2\mathrm{e}^{\rho T}\cos(\omega T), \qquad A_0 = -\mathrm{e}^{2\rho T} \tag{13.23}$$

  Solution: Again, the proof is formally the same as for the previous cases, only with the basis elements $\mathrm{e}^{\rho kT\sin(\omega kT)}$ and $\mathrm{e}^{\rho kT}\cos(\omega kT)$.

  The response of the continuous system is

  $$y_k = c_1\mathrm{e}^{\rho t}\cos(\omega t) + c_2\mathrm{e}^{\rho t}\sin(\omega t)$$

  After expressing the sampled response and its shifted variants, substituting into (13.20) and comparing the terms at the individual basis items, we obtain the following set of equations

  $$\begin{aligned} -c_1\mathrm{e}^{2\rho T}\sin(2\omega T) + c_2\mathrm{e}^{2\rho T}\cos(2\omega T) &= -A_1c_1\mathrm{e}^{\rho T}\sin(\omega T) + A_1c_2\mathrm{e}^{\rho T}\cos(\omega T) + A_0c_2 \\ c_1\mathrm{e}^{2\rho T}\cos(2\omega T) + c_2\mathrm{e}^{2\rho T}\sin(2\omega T) &= A_1c_1\mathrm{e}^{\rho T}\cos(\omega T) + A_1c_2\mathrm{e}^{\rho T}\sin(\omega T) + A_0c_1 \end{aligned}$$

  Once more, the solution is just what we wanted to proof. ◁

13.4 Point estimate with quadratic criterion

Here we show that for quadratic criterion of optimality, the optimal point estimate of unknown parameters Θ, based on data $d(t)$, is the conditional expectation $\hat{\Theta} = \hat{E}[\Theta|d]$.

Proof

The criterion to be minimized is

$$J = E\left[\left(\hat{\Theta} - \Theta\right)^2 |d(t)\right] \to \min$$

We derive (completion to square)

$$\begin{aligned} \min_{\hat{\Theta}} E\left[\left(\hat{\Theta} - \Theta\right)^2 |d\right] &= \min_{\hat{\Theta}} E\left[\hat{\Theta}^2 - 2\hat{\Theta}\Theta + \Theta^2|d\right] = \\ &= \min_{\hat{\Theta}}\left\{\hat{\Theta}^2 - 2\hat{\Theta}E[\Theta|d] + E\left[\Theta^2|d\right]\right\} = \\ = \min_{\hat{\Theta}}\left\{\hat{\Theta}^2 - 2\hat{\Theta}E[\Theta|d] + E[\Theta|d]^2 \underbrace{-E[\Theta|d]^2 + E\left[\Theta^2|d\right]}_{D[\Theta]}\right\} &= \\ = \min_{\hat{\Theta}}\left\{\hat{\Theta}^2 - 2\hat{\Theta}E[\Theta|d] + E[\Theta|d]^2\right\} + D[\Theta] &= \\ = \min_{\hat{\Theta}}\left\{\left(\hat{\Theta} - E[\Theta|d]\right)\right\} + D[\Theta] & \end{aligned}$$

$\to \hat{\Theta} = E[\Theta|d]$.

13.5 Minimization of the control criterion criterion

We start at the end on the control interval, i.e. at time $t = N$. In this step, only penalization for $t = N$ should be minimized, however, we know, that after each step of minimization a remainder (not depending on control in this step) is obtained and it goes to the next step. So, we include the term φ^*_{N+1} even if we know that in reality it is zero. It is only a formal step that helps us to catch the recurrence.

$$\min_{u_{1:N}} E\left[\varphi^*_{N+1} + \sum_{t=1}^{N} J_t | d\left(0\right)\right] =$$

we will express the criterion in an integral form

$$= \min_{u_{1:N}} \int \cdots \int \left(\varphi^*_{N+1} + \sum_{t=1}^{N} J_t\right) f\left(y\left(N\right), u\left(N\right) | d\left(0\right)\right) dy\left(N\right) du\left(N\right) =$$

we use chain rule for y_N and u_N
and put together reminder and last penalization

$$= \min_{u_{1:N}} \int \cdots \int \int \int \left(\left[\varphi^*_{N+1} + J_N\right] + \sum_{t=1}^{N-1} J_t\right) f\left(y_N | u_N, d\left(N-1\right)\right) f\left(u_N | d\left(N-1\right)\right) \times$$
$$\times f\left(y\left(N-1\right), u\left(N-1\right) | d\left(0\right)\right) dy\left(N\right) du\left(N\right) =$$

we integrate over y_n (expectation)
and denote the integral by φ_N

$$= \min_{u_{1:(N-1)}} \left\{ \int \cdots \int \min_{u_N} \int \underbrace{\int \left(\varphi^*_{N+1} + J_N\right) f\left(y_N | u_N, d\left(N-1\right)\right) dy_N}_{\varphi_N(u_N, d(N-1))} f\left(u_N | d\left(N-1\right)\right) du_N + \right.$$
$$\left. \sum_{t=1}^{N-1} J_t f\left(y\left(N-1\right), u\left(N-1\right) | d\left(0\right)\right) dy\left(N-1\right) du\left(N-1\right) \right\}$$

now we are going to minimize over u_N

$$\min_{u_N} \int \underbrace{\int \left(\varphi^*_{N+1} + J_t\right) f\left(y_N | u_N, d\left(N-1\right)\right) dy_N}_{\varphi_N(u_N, d(N-1))} f\left(u_N | d\left(N-1\right)\right) du_N =$$

$$= \min_{u_N} \int \varphi_N\left(u_N, d\left(N-1\right)\right) f\left(u_N | d\left(N-1\right)\right) du_N$$

The minimum is achieved if we give the control u_N a deterministic form (its pdf is Dirac function) and place it into the minimum of the integral φ_N. So, we take:

$u^*_N = \arg\min_{u_N} \varphi_N$ and $f\left(u_N | d\left(N-1\right)\right) = \delta\left(u_N, u^*_N\right)$ - all u_t is concentrated into one point u^*_N.

The final form of the criterion has the same form as in the beginning, only time indexes are shifted by one down $N \to N-1$. So, we can exchange index N for the current index t and we are coming to the recursion (10.2) and (10.3), i.e.

$$\varphi_t = E\left[\varphi^*_{t+1} + J_t | u_t, d\left(t-1\right)\right] \quad \text{expectation}$$

$$\varphi^*_t = \min_{u_t} \varphi_t \quad \text{minimization.}$$

14 Introduction to Scilab

Remarks

1. All variables are matrices. Scalar is matrix a(1,1). Vector in first row is a(1,:), in first column a(:,1). The sign : means "all".

2. Semi-column ; means: no response. If there is comma or nothing in the end of a command, its value is printed on the screen.
   Remark: the command mode(0) must be called at the beginning.

3. help „object" gives help on "object".
   Icon **?** calls the main help.

4. Comment begins with //.

Variables and operations

There are the following main typed of variables:

- **čísla (matice)**

  Definition:
  - scalar a=5;
  - row vector a=[3 5 1];
  - column vector a=[3; 5; 1], which is the same as a=[3 5 1]'
  - matrix a=[2 3 4; 8 7 6];
  - command a=5:8 creates the vector [5 6 7 8]; 5:2:13 = [5 7 9 11 13]
  - command a=zeros(2,3) creates matrix 2×3 from zeros
  - command a=ones(2,3) creates matrix 2×3 from ones
  - transposition is performed by ' (apostroph)
  - matrix b (3×3) can be composed like this: b=[a; 2*a; 5*a];

  Operations:
  - product of matrices * division / power ^ or ** square root sqrt()
  - dot operations .* ./ .^ are performed entry by entry
  - in operation * the rules of matrix product hold
  - operation a/b means multiplication of a by inversion of b
  (inversion itself is inv(b))

- **text**: a='hello'. It is a vector of letters can be concatinated:
  a='hello '; b='boys' a c=a+b, then c='hello boys'.
  Conversion: s=string(a) gives value of variable a as a string

- **logical variables -** their valies are „true" (=1) a „false" (=0).
  Logical operations: == ~= < <= > >= & (and) | (or) ~ (not)

Examples

Set:

a=[1 2 3] b=[8; 9] c=[11 12 13; 21 22 23; 31 32 33];

Try and justify:

x1=a*a' x2=a'*a y=[[a;5*a] b] c(2,:).*a c(1,2:3)*b

c(3,:).^c(1,:) c(3,:)**2 d1=c(:) dd=c'; d2=dd(:) d2(3:2:7)

Set:

u='first' v='attempt' x=%t (setting of "true") y=5==5 z=5>5

Try and justify:

u+' '+v x & y x & z x | y x | z

Work with variables

- Command who_user(); gives information about defined variables.
- [m,n]=size(a), m=size(a,1), n=size(a,2) give dimensions of the matrix a, resp. number of rows, number of columns. Instead of 1 a 2 one can use 'r' a 'c'.
- n=length(a) number of elements of a.
- n=max(size(a)) length of a vector
- clc clears screen
- clear clears variables
- xdel(winsid()) clears all graphs (close clears the last one)

Programming commands

- **Condition if**

```
if b>c,
   a=5;
else
   a=0;
end
```

If `b>c` is true, it is preformed `a=5;` otherwise `a=0;`.

Example

```
// Determine c as bigger from  a, b
a=rand(1,1,'n'); b=rand(1,1,'n');
if a>b, c=a;
else c=b;
end
printf('a = %g, b = %g, c = %g\n',a,b,c)
```

- **Branching of program**

```
select i,
   case 1, prikaz_A;
   case 2, prikaz_B;
   else prikaz_D
end
```

According to `i` the respective command is performed.

Example

```
// According to  i  perform
// set the vectors
a=[1 3 5]; b=[2 4 6];
// 1 - addition
// 2 - scalar product
// 3 - tenzor product
// set the operation
```

cont.

```
i=2;
select i
case 1, d=a+b;
case 2, d=a*b';
case 3, d=a'*b;
end
disp(d,'result')
```

- **Cycle for**

```
for i=1:5
   a(i)=2*i;
end
```

For `i=1,2,3,4,5` the command `a(i)=2*i;` is performed. :Result is `a=[2, 4, 6, 8, 10]`.

Example 1

```
// Determine weighted sum
x=[1 2 3 4 5 6]; // numbers
p=[.1 .3 .2 .1 .2 .1]; // weights
n=length(x);
s=0;
for i=1:n
s=s+x(i)*p(i);
end
disp(s,'the average is')
```

Example 2

```
// Order numbers according to magnitude
n=10; // how many numbers
a=fix(100*rand(1,n,'u')); // čísla
disp(a,'original numbers')
b=[];
for i=1:n
[x,j]=min(a);
b=[b x];
a(j)=%nan;
end
disp(b,'ordered numbers')
end
```

- **Control of the program**

  pause stops the program.
  resume resumes the program after pause
  abort stops the program definitely.

- **Calling of subprogram**
  exec('my_program',-1) runs the program my_program (-1 suppresses response)

- **Loading functions to memorry**
  getd('my_address') loads all subroutines in the address moje_adresa
  (Scilab does not have path. It knows only the loaded functions)

Printing

Commands disp and fprintf .

- disp(a) shows value of a.
- disp(a,'text') gives value and the text
- printf('entry %d of vector a is %g\n',i,a(i));
  gives e.g.: *entry 5 of vector a is 4.12*

Graphical output

Two-dimensional graph can be constructed by plot.

Examples:

- plot(y) draws values of y.

- plot(x,y) draws values of y against of x (so called xy-graf).
- plot(a) draws columns of matrix a.

Formatting of a graph:

Line	**Points**	**Color**
- (full)	. (point)	r (red)
: (dotted)	+ (plus)	g (green)
-. (dot dashed)	o (ring)	b (blue)
– (dashed)	x (cross)	w (white)

For more details, call: help plot or go to Scilab help: Scilab Help >> Graphics > GlobalProperty

Examples:

- plot(x,'or') draws x using red crosses.
- plot(x,y,'r-+',u,v,'b-x') draws two curves (x,y) a (u,v); the first one is red by full line with pluses, the second one by blue line with crosses.

15 Programs for exercises

Basic advises for running programs

Each Scilab program should start with commands clearing screen, memory and closing figures from previous run: clc, clear, close. These commands together with other useful ones are collected in the program SciIntro.sce

```
// General introduction of a SciLab run

clc, clear, clearglobal()     // clear screen, variables, global vars
xdel(winsid());               // clear all existing figures
funcprot(0);                  // renaming functions without echo
warning('off');               // suppress warnings

printf('Running ...\n\n')     // echo of runing program
```

Here, instead of clearing only one latest figure, all existing figures are cleared by the command xdel(winsid());. Then warnings are suppressed. In the end, an echo of running program is displayed.

This command should be called at the beginning of each program by

exec("SciIntro.sce",-1)

However, for the program to be found, the working directory containing this program and other functions if used must be set.

So, the procedure is as follows: put all your programs into your working directory (named howsoever by you). Then open Scilab and check if in the File Browser you can see your programs. If not, then at the top of this window you can set your working directory.

Remark

If the File Browser is not visible, put the cursor into the Console Window, click at Applications and on File Browser down in the menu).

Hint

If you want so that the working directory is automatically set to that from you run your program, put the following command directly before the command exec(···). It will do the work itself.

[u,t,n]=file(); chdir(dirname(n(2))); clear u t n;

Other useful commands

- If you want so that the values of variables appear automatically of the screen when program is running, you must insert the command mode(0) after the command exec($\cdots$) ... not before it.
- If some functions are to be executed during the program run, the command getd() must be inserted (again after exec). It will load all functions in the current directory. If the functions are in some other directory, you must call getd('direrory') where directory is the address where you have your functions.

15.1 Simulation with regression model

In the program, first length of simulation and parameters of the model are set.

As the model is of the second order, initial conditions of y for times -1 and 0 are set. Scilab uses indexes 1, 2, $\cdots$ (not less than one). So times 1 and 2 belong to initial times -1 and 0. That is why, the simulation must start at time $t = 3$.

Control variable is defined beforehand as a noisy sinusoidal curve.

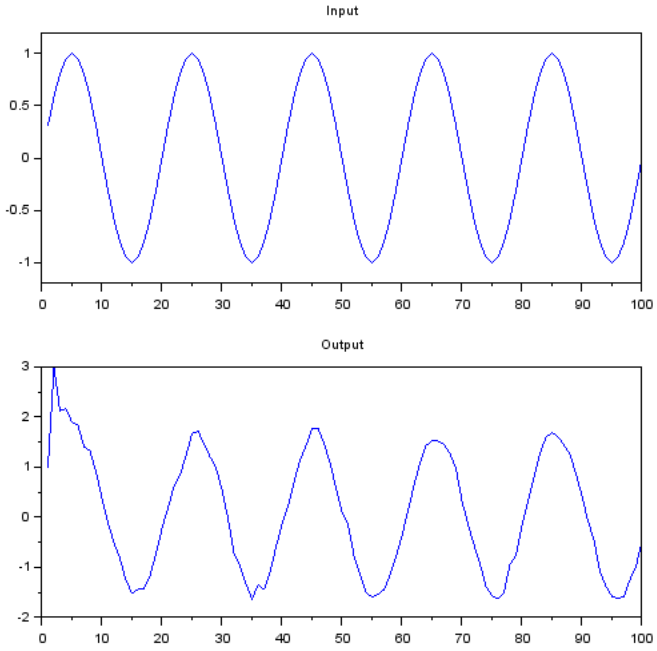
The simulation itself is realized in the for loop t = 3:nd. First the regression vector ps is constructed. It is then multiplied with the vector of parameters with the noise term added. The result is simulated output in a current time t.

In the end of the program, the results are displayed.

```
// T11simCont.sce
// SIMULATION OF THE SECOND ORDER REGRESSION MODEL
// Experiments
// - change parameters of the model
// - change the input signal
// - try to increase the model order to 3
// ---------------------------------------------------------------
exec("ScIntro.sce",-1)

// PARAMETERS OF THE SIMULATION
nd=100;                         // length of data
a=[.4 .2];                      // parameters at yt
b=[1 .2 -.5];                   // parameters at ut
k=0;                            // constant (model absolute term)
s=.1;                           // noise variance

yt(1)=1; yt(2)=3;               // initial conditions for output
ut=sin(10*%pi*(1:nd)'/nd)+.001*rand(nd,1,'n'); // input
```

```
// TIME LOOP OF THE SIMULATION
th=[a b k]';                    // vector of parameters
for t=3:nd
  // regression vector
  ps=[yt(t-1) yt(t-2) ut(t) ut(t-1) ut(t-2) 1]';
  // regression model
  yt(t)=th'*ps+s*rand(1,1,'norm');
end

// RESULTS OF THE SIMULATION
set(gcf(),"position",[700 100 600 500])
subplot(211),plot(1:nd,ut),title('Input')
subplot(212),plot(1:nd,yt),title('Output')
```

The result is



The system is excited by a sinusoidal signal (upper picture). The response is also sinusoidal, corrupted by noise. The dynamic of the system is relatively weak and in the output mainly amplification is apparent.

To see greater distortion of the output, set a=[.77 .2];

Remarks

1. *The amplification of the system in the steady state can be obtained by setting* $y_t = y_{t-1} =$

y_{t-2} *and* $u_t = 1$. *From this it follows*

$$y_\infty = \frac{b_0 + b_1 + b_2}{1 - a_1 - a_2}.$$

From this formula we can see, e.g. that if $a_1 + a_2 = 1$ *ten the system is on the border of stability as* $y_\infty \to \infty$.

2. *If* $a_1 + a_2 < 1$ *but* $a_1 + a_2 \to 1$ *the system is slow. New output nearly equals to to the average of the two previous. Thus, it changes very little.*

3. *If we want to see the dynamic of the system, set* `ut = ones(1,nd)` *and perhaps* `sd` *very small. The you obtain so called* ***step response*** *of the system that* shows the dynamic better.

4. If we set the control variable all equal to zero and set initial condition $y_{-1}=$ `y(1)` to one, then so called pulse response is generated. It also speaks about the dynamic of the system.

15.2 Simulation with discrete model

The beginning of the program is standard - initialization of Scilab (ScIntro), definition of parameters, length of data, input generation and initialization of output.

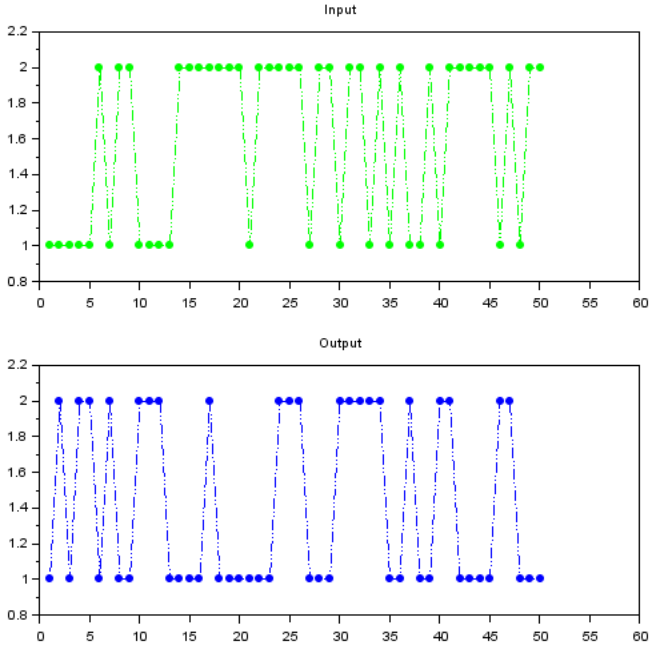
In the loop of generation, first, the row of the model matrix is computed in dependence on the input and last output: i=2*(ut(t)-1)+yt(t-1); and then from the selected row the output is generated using the probabilities from the row. The formula for generation is given in the Lectures in Paragraph 4.1.

In the end of the program, the results are displayed.

```
// T13simDisc.sce
// SIMULATION OF DISCRETE MODEL
//  (multinomial model - controlled coin with memmory)
//         f( yt(t) | ut(t),yt(t-1) ),  yt,ut=1,2
// Experiments
// - set the parameters to obtain a deterministic model
// - try to extend the model to f(y(t)|u(t),y(t-1),u(t-1))
//   and values 1,2,3.
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1)

// PARAMETERS OF THE SIMULATION
// model parameter (conditional probabilities)
//yt(t)=1 =2  //  ut(t) yt(t-1)
// ---------------------------
th= [.2 .8    //  1    1
     .6 .4    //  1    2
     .9 .1    //  2    1
     .3 .7];  //  2    2
nd=50;                          // number of steps
ut=(rand(1,nd,'u')>.3)+1;       // control variable P(ut=1)=.3, P(ut=2)=.7
yt(1)=1;                        // initial condition for output

// TIME LOOP OF THE SIMULATION
for t=2:nd
  i=2*(ut(t)-1)+yt(t-1);        // row in the model parameter
  yt(t)=(rand(1,1,'u')>th(i,1))+1;  // generation of the output
end

// RESULTS OF THE SIMULATION
subplot(211),plot(1:nd,ut,'g:.')
set(gcf(),"position",[700 100 600 500])
title('Input')
```

```
set(gca(),'data_bounds',[0 nd+1 .9 2.1])
subplot(212),plot(1:nd,yt,'b:.')
title('Output')
set(gca(),'data_bounds',[0 nd+1 .9 2.1])
```

The result is



Both, the input variable and the output are discrete, so they have finite number of different values (here only two).

The model is set as considerably uncertain (the parameters are far from being zeros or ones). To see a deterministic case, set the parameters equal or close to 0 or 1. E.g. the output of the model with the parameter

$$\Theta = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

will follow the input.

The output of this model

$$\Theta = \begin{bmatrix} .9 & .1 \\ .9 & 1 \\ .1 & .9 \\ .1 & .9 \end{bmatrix}$$

will mostly follow the input.

15.3 Simulation with state-space model

Here is a program generation data from regression model which is realized in the state form - see Paragraph 3.1.

After a standard beginning of the program, we come to the first loop, where a standard regression simulation is performed.

Then, a transformation of regression model to the state one is performed and the second loop follows where the same simulation is programmed and realized.

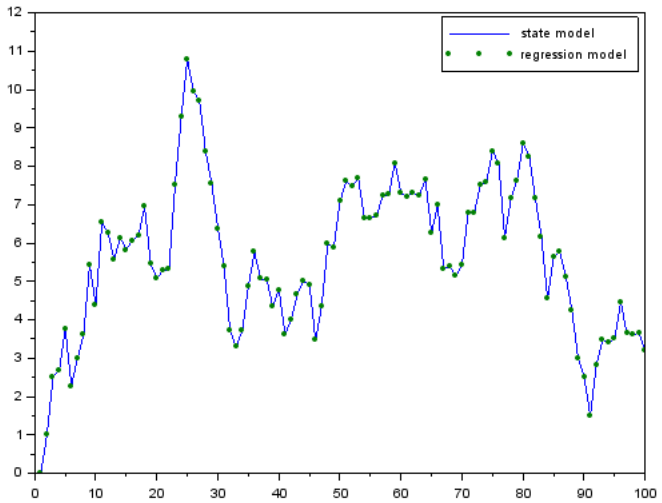
The output of the regression model is denoted by yr, that of the state realization is yt.

In the results both the simulations are compared. The results should be exactly the same.

```
// T15simState.sce
// SIMULATION WITH RM IN A STATE-SPACE FORM
// Experiments
// - extend to the 3-rd order model
//   y(t)=b0.u(t)+a1.y(t-1)+b1.u(t-1)+a2.y(t-2)+b2.u(t-2)+
//        +a3.y(t-3)+b3.u(t-3)+k+e(t)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1)
rand('seed',0)

// PARAMETERS OF THE SIMULATION
nd=100;                         // number of data

et=rand(1,nd,'n');
ut=rand(1,nd,'n');              // input
a=[.6 .1]; b0=.8; b=[.3 .2]; k=2; cv=.1; // model parameterers

// REGRESSION REALIZATION
yr(1)=0; yr(2)=1;
for t=3:nd
  er=sqrt(cv)*et(t);
  yr(t)=a*[yr(t-1) yr(t-2)]'+b0*ut(t)+b*[ut(t-1) ut(t-2)]'+k+er;
end

// STATE-SPACE REALIZATION
M=[a(1) b(1) a(2) b(2) k
   0    0    0    0    0
   1    0    0    0    0
   0    1    0    0    0
   0    0    0    0    1];
N=[b0 1 zeros(1,3)]'
```

```
A=[1 zeros(1,4)];
B=0;

yt(1)=0; yt(2)=1;
xt(:,2)=[yt(2) ut(2) yt(1) ut(1) 1]';  // initial conditions for state

// time loop of simulation
for t=3:nd
  es=[sqrt(cv)*et(t) zeros(1,4)]';
  xt(:,t)=M*xt(:,t-1)+N*ut(t)+es;
  yt(t)=A*xt(:,t);
end

// RESULTS OF SIMULATION
scf(1); plot(1:nd,yt,1:nd,yr,'.','markersize',4)
legend('state model','regression model');
```

The result is to show that really the outputs generated by regression model and the equivalent state model are identical.



Remark

So that the outputs are really the same, also the initial conditions must be identical. Here: xt(:,2)=[yt(2) ut(2) yt(1) ut(1) 1] (see construction of state in Paragraph 3.1).

15.4 Least squares estimation

Here, a very brief and useful procedure of estimation via least squares. We construct vector

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix} \text{ and } X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & 1 \\ x_{21} & x_{22} & \cdots & x_{2n} & 1 \\ \cdots & \cdots & \cdots & \cdots & 1 \\ x_{N1} & x_{N2} & \cdots & x_{Nn} & 1 \end{bmatrix}$$

where $x_1, x_2, \cdots, x_n$ are explanatory variables (e.g. $[u_t, y_{t-1}, u_{t-1}]$) and ones stay for model constant. The presentation of the method can be found in Paragraph 3.1.

Remark

This procedure is suitable especially when the data are obtained e.g. in Excel in the form of a matrix $\begin{bmatrix} y_1 & x_{11} & x_{12} & \cdots \\ y_2 & x_{21} & x_{22} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$ *where the data are practically prepared for construction of* Y *and* X*. It happens frequently in real applications when the output variable* y *is assumed to depend on several explanatory variables* x_1,x_2,$\cdots$,x_n *at the same time instant and we want to learn, which of them and how, if at all, influences the output.*

Here, the program demonstrates estimation of regression model with regression vector $\psi = x = [u_t, y_{t-1}, u_{t-1}, \cdots]$ giving the parameters $\theta = [b_0, a_1, b_1, \cdots]$.

In the first loop, the data are simulated from regression model.

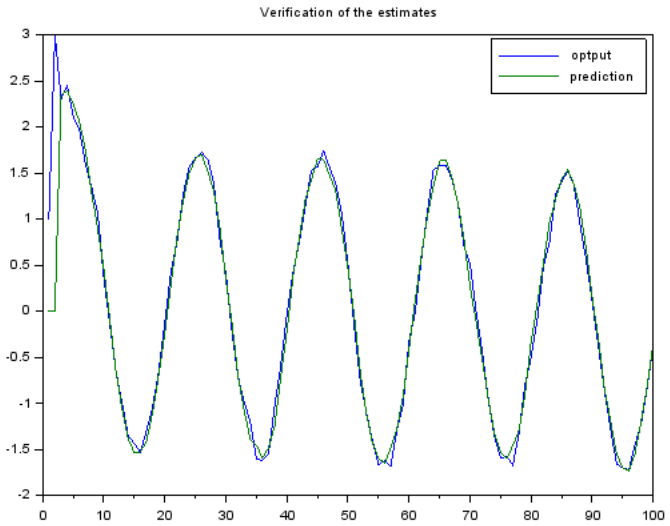
The second loop consists of (i) construction Y and X from outputs and regression vectors. After it, one-shot estimation of parameters is performed.

In the end, the estimated parameters are printed as well as the simulated data are plotted.

```
// T21estCont_LS.sce
// ESTIMATION OF 2ND ORDER REGRESSION MODEL
//   least squares estimation (off-line)
// Experiments
// - increase the model order
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1), mode(0)

// SIMULATION
// parameters
nd=100;                         // length of data
a=[.4 .2];                      // parameters at yt
b=[1 .2 -.5];                   // parameters at ut
k=0;                            // constant (model absolute term)
s=.1;                           // noise variance
```

```
yt(1)=1; yt(2)=3;              // initial conditions for output
ut=sin(10*%pi*(1:nd)/nd)'+.001*rand(nd,1,'n'); // input

// time loop
th=[a b k]';                   // vector of parameters
for t=3:nd
  // regression vector
  ps=[yt(t-1) yt(t-2) ut(t) ut(t-1) ut(t-2) 1]';
  // regression model
  yt(t)=th'*ps+s*rand(1,1,'norm');
end

// ESTIMATION
for t=3:nd
  Y(t,1)=yt(t);
  X(t,:)=[yt(t-1) yt(t-2) ut(t) ut(t-1) ut(t-2) 1];
end
th=inv(X'*X)*X'*Y;             // estimation of regression coefficients
yp=X*th;                       // prediction (for verification)
r=variance(yt-yp);             // noise variance

// Results
disp('Parameter estimates')
th,r

scf(1);                        // comparison od output and prediction
plot(1:nd,yt,1:nd,yp)
legend('optput','prediction');
title('Verification of the estimates')
```

The result is



Remark

The regression vector, in the program, is constructed in a bit different way as usually (firs outputs and then inputs)

$$\psi = [y_{t-1}, y_{t-2}, u_t, u_{t-1}, u_{t-2}, 1]\,.$$

It is OK, the order does not matter. However, we must keep in mind, that the regression coefficients in the parameter θ will have the corresponding order

$$\theta = [a_1, a_2, b_0, b_1, b_2, k]\,.$$

15.5 Estimation with continuous model

Here is a standard estimation with regression model presented. Standard beginning with definition of data length, model parameters and initial conditions. Also input signal is generated beforehand for all data length.

First loop generates values of y.

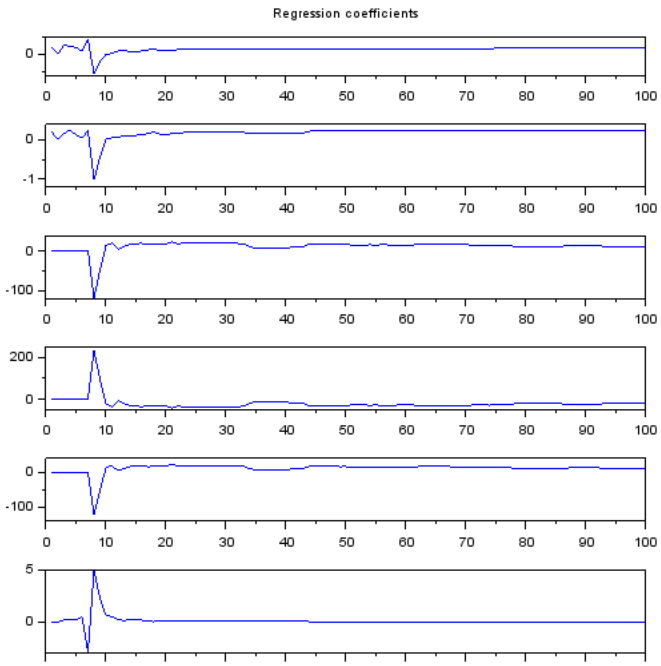
Second loop performs estimation. At each time step, extended regression vector is constructed (together with the actual output y_t) and the information matrix V is updated - see Paragraph 6.1 formula 6.3. Then the information matrix is partitioned and point estimates are computed - see Paragraph 6.1 below.

The end of the program shows the results.

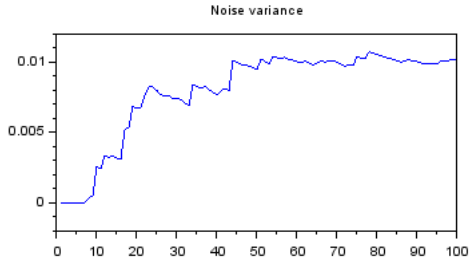
```
// T22estCont_B.sce
// ESTIMATION OF 2ND ORDER REGRESSION MODEL
// - Bayesian on-line estimation with statistic update
// Experiments
// - rewrite the program to a single time loop (on-line estimation)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1)

// SIMULATION
// parameters
nd=100;                         // length of data
a=[.4 .2];                      // parameters at yt
b=[1 .2 -.5];                   // parameters at ut
k=0;                            // constant (model absolute term)
s=.1;                           // noise variance

yt(1)=1; yt(2)=3;               // initial conditions for output
ut=sin(10*%pi*(1:nd)/nd)'+.001*rand(nd,1,'n'); // input

// time loop
th=[a b k]';                    // vector of parameters
for t=3:nd
  // regression vector
  ps=[yt(t-1) yt(t-2) ut(t) ut(t-1) ut(t-2) 1]';
  // regression model
  yt(t)=th'*ps+s*rand(1,1,'norm');
end

// ESTIMATION
V=1e-8*eye(7,7);                // initial information matrix
```

```
for t=3:nd
  psi=[yt(t:-1:t-2); ut(t:-1:t-2); 1];       // reg. vector
  V=V+psi*psi';                              // updt of information matrix
  Vy=V(1,1); Vyp=V(2:$,1); Vp=V(2:$,2:$);    // divisioning of inf. matrix
  thE(:,t)=inv(Vp+1e-8*eye(Vp))*Vyp;         // pt estimates of reg. coefficints
  r(t)=(Vy-Vyp'*inv(Vp+1e-8*eye(Vp))*Vyp)/t; // pt estimates of noise variance
end

// Results
set(scf(1),'position',[500 60 600 500])
for i=1:6
  subplot(6,1,i),plot(thE(i,:))
  if i==1, title('Regression coefficients'), end
end
set(scf(2),'position',[50 60 400 200])
plot(r)
title('Noise variance')
```

The results are: evolution of point estimates of regression coefficients during estimation (what is important is that they stabilize - even we can see that 30 samples of data wold be sufficient for good estimation)



and evolution of point estimate of noise variance. From the picture it is evident that estimation of the variance is more data demanding than it is for regression coefficients - this holds generally.



Estimation under model structure mismatch

In the above example we used the same structure of both models for simulation and estimation. I.e. we simulated data from the regression model with given parameters and for estimation we used the same regression model with unknown parameters, indeed. In this case, agreement on the values of the corresponding parameters can be expected for a good estimate. However, the reality is much more difficult. To approach it we will model structure mismatch in the three following examples. First two with simulated data then one with real ones.

• *Simulation of 3^{rd} order model, estimation with 1^{st} order one*

Here, a third-order model is used for simulation. The input signal can be chosen by setting the option `inp` as constant, sin function, jumps or random noise. It can be shown, that the first three signals are not very convenient as with them the excitation of the system is not good.

The success of estimation cannot be checked by agreement in simulated and estimated parameters! However, it can be judged by monitoring the evolution of parameter estimates. They should be stabilized as soon as possible and then stay constant.

```
// T22estCont_B2.sce
// ESTIMATION OF 2ND ORDER REGRESSION MODEL
// The model for simulation differs from that for estimation
// i.e. MODEL STRUCTURE MISMATCH is tackled.
// - Bayesian on-line estimation with statistic update
// Experiments
// - try various input signals - set inp = 1,2,3,4
// - set different noise variance r=0,.1,1,10
// - define other regression coefficients a,b,k
// - you can also change the orders of models for simulation
//   as well as for estimation (adjust the beginning of loop)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1)

// SIMULATION
// parameters
```

```
nd=100;                          // length of data
a=[.1 -.8 .3];                   // parameters at yt
b=[1 .2 -.5 -.8];                // parameters at ut
k=0;                             // constant (model absolute term)
s=.01;                           // noise variance
inp=1;                           // selection of input

yt(1)=0; yt(2)=0; yt(3)=0;       // initial conditions for output
                                                 // inputs at disposal
select inp
case 1, ut=ones(nd,1);                           // one jump
case 2, ut=sin(10*%pi*(1:nd)/nd)';               // several jumps
case 3, ut=sign(10*sin(10*%pi*(1:nd)/nd))';      // sine wave
case 4, ut=.1*rand(nd,1,'n');                    // white noise
end

// time loop
th=[a b k]';                     // vector of parameters
for t=4:nd
  // regression vector
  ps=[yt(t-1) yt(t-2) yt(t-3) ut(t) ut(t-1) ut(t-2) ut(t-3) 1]';
  // regression model
  yt(t)=th'*ps+s*rand(1,1,'norm');
end

// ESTIMATION
V=1e-8*eye(5,5);                 // initial information matrix
for t=3:nd
  psi=[yt(t:-1:t-1); ut(t:-1:t-1); 1];       // reg. vector
  V=V+psi*psi';                              // updt of information matrix
  Vy=V(1,1); Vyp=V(2:$,1); Vp=V(2:$,2:$);    // divisioning of inf. matrix
  thE(:,t)=inv(Vp+1e-8*eye(Vp))*Vyp;         // pt estimates of reg. coefficints
  r(t)=(Vy-Vyp'*inv(Vp+1e-8*eye(Vp))*Vyp)/t; // pt estimates of noise variance
end

// Results
set(scf(1),'position',[500 60 600 500])
for i=1:4
  subplot(6,1,i),plot(thE(i,:))
  if i==1, title('Regression coefficients'), end
end
set(scf(2),'position',[50 60 400 200])
plot(r)
title('Noise variance')
```

```
set(scf(3),'position',[50 360 400 200])
plot(1:nd,yt,1:nd,ut)
legend('output','input');
title 'Dataset for estimation'

disp(th','Simulated parameters')
disp(thE(:,$)','Easimated parameters')
```

• *Simulation of 1^{st} order model, estimation with the order that can be set*

Here, the situation is the same with the difference, that the model for simulation is of the first-order and that for estimation can be set with arbitrary order using the option ord which is pre-set to 3. It means again model mismatch.

```
// T22estCont_B3.sce
// ESTIMATION OF 2ND ORDER REGRESSION MODEL
// The model for simulation differs from that for estimation
// i.e. MODEL STRUCTURE MISMATCH is tackled.
// - Bayesian on-line estimation with statistic update
// Experiments
// - try various input signals - set inp = 1,2,3,4
// - set different noise variance r=0,.1,1,10
// - define other regression coefficients a,b,k
// - you can also change the orders of models for simulation
//   as well as for estimation (adjust the beginning of loop)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1)

// SIMULATION
// parameters
nd=100;                         // length of data
a=[.1 ];                        // parameters at yt
b=[1 .2];                       // parameters at ut
k=0;                            // constant (model absolute term)
s=.01;                          // noise variance
inp=1;                          // selection of input
ord=3;                          // order of the estimated model

yt(1)=0; yt(2)=0; yt(3)=0;      // initial conditions for output
                                              // inputs at disposal
select inp
case 1, ut=ones(nd,1);                        // one jump
case 2, ut=sin(10*%pi*(1:nd)/nd)';            // several jumps
case 3, ut=sign(10*sin(10*%pi*(1:nd)/nd))';   // sine wave
```

```
case 4, ut=.1*rand(nd,1,'n');                // white noise
end

// time loop
th=[a b k]';                  // vector of parameters
for t=2:nd
  // regression vector
  ps=[yt(t-1) ut(t) ut(t-1) 1]';
  // regression model
  yt(t)=th'*ps+s*rand(1,1,'norm');
end

// ESTIMATION
nth=2*(ord+1)+1;
V=1e-8*eye(nth,nth);             // initial information matrix
for t=(ord+1):nd
  psi=[yt(t:-1:t-ord); ut(t:-1:t-ord); 1];// reg. vector
  V=V+psi*psi';                              // updt of information matrix
  Vy=V(1,1); Vyp=V(2:$,1); Vp=V(2:$,2:$);    // divisioning of inf. matrix
  thE(:,t)=inv(Vp+1e-8*eye(Vp))*Vyp;         // pt estimates of reg. coefficints
  r(t)=(Vy-Vyp'*inv(Vp+1e-8*eye(Vp))*Vyp)/t; // pt estimates of noise variance
end

// Results
set(scf(1),'position',[500 60 600 500])
for i=1:nth-1
  subplot(nth,1,i),plot(thE(i,:))
  if i==1, title('Regression coefficients'), end
end
set(scf(2),'position',[50 60 400 200])
plot(r)
title('Noise variance')
set(scf(3),'position',[50 360 400 200])
plot(1:nd,yt,1:nd,ut)
legend('output','input');
title 'Dataset for estimation'

disp(th','Simulated parameters')
disp(thE(:,$)','Easimated parameters')
```

• *Estimation of real data*

Here, real dataset of intensities measured in the Strahov tunnel is used for estimation. The dataset contains measurements with 5 minutes period of sapling for approximately nine weeks.

One day contains 288 samples. Here we have selected third day and we start here with the 50^{th} sample (it is sometimes in the morning - approximately at 4 o'clock). The end is at midnight.

Again, the order of the estimated model (there is no other one) has the order that can be set by the option ord.

Here, again, the verification of the estimation success is problematic. The best way of verification is making a prediction and comparing the real data with this prediction. We shall demonstrate this case in the Paragraph 15.8.

```
// T22estCont_B4.sce
// ESTIMATION OF 2ND ORDER REGRESSION MODEL
// Estimation with REAL DATA (intensities of traffic in Strahov tunnel)
// i.e. MODEL STRUCTURE MISMATCH is tackled.
// - Bayesian on-line estimation with statistic update
// - cannot be done without estimation !!!
// Experiments
// - change the order of model for estimation
// Result: estimated parameters (better with prediction - T32preCont_Adapt3.sce)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1)

ord=5;                          // order of the estimated model
// selection of dataset
k=3;                            // selected day
nz=50;                          // beginning of the day
nd=288;                         // length of the whole one day data

// DATA
dtAll=csvRead('STRAHOV.csv',';');
dt=dtAll((k-1)*288+(nz+1:nd),1);
yt=dt(:,1);

// ESTIMATION
nth=ord+2;
V=1e-8*eye(nth,nth);              // initial information matrix
for t=(ord+1):(nd-nz)
  psi=[yt(t:-1:t-ord); 1];// reg. vector
  V=V+psi*psi';                                 // updt of information matrix
  Vy=V(1,1); Vyp=V(2:$,1); Vp=V(2:$,2:$);     // divisioning of inf. matrix
  thE(:,t)=inv(Vp+1e-8*eye(Vp))*Vyp;          // pt estimates of reg. coefficints
  r(t)=(Vy-Vyp'*inv(Vp+1e-8*eye(Vp))*Vyp)/t; // pt estimates of noise variance
end

// Results
```

```
set(scf(1),'position',[500 60 600 500])
for i=1:nth-1
  subplot(nth,1,i),plot(thE(i,:))
  if i==1, title('Regression coefficients'), end
end
set(scf(2),'position',[50 60 400 200])
plot(r)
title('Noise variance')
set(scf(3),'position',[50 360 400 200])
plot(1:nd-nz,yt)
legend('output');
title 'Dataset for estimation'

disp(thE(:,$)','Easimated parameters')
```

15.6 Estimation with discrete model

Program performs current estimation of model parameters. It begins in a standard way.

The first loop is simulation. First, determination of the row in model matrix and then generation of the output from the row see Paragraph 4.1.

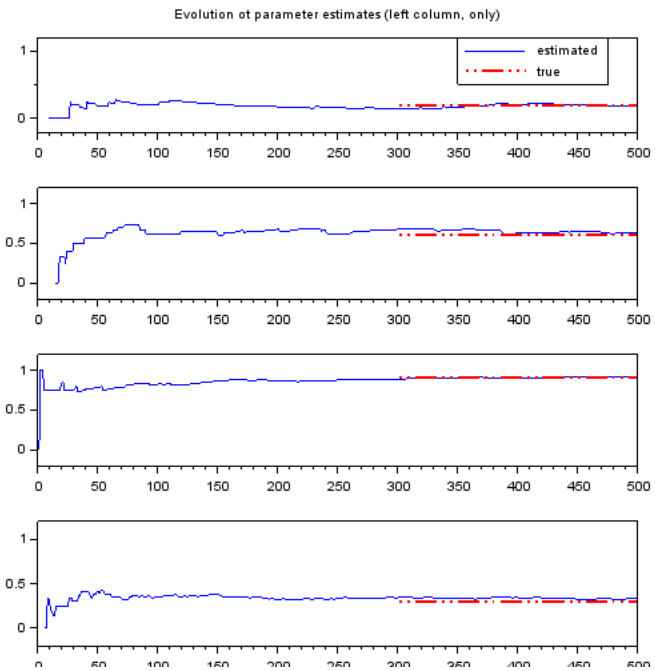
The second loop is estimation. Determination of the row, update of the statistics V (see (6.5)) and computation of point estimates of the parameters by normalization of the statistics (see (6.6)).

Finally, plot of the results - time evolution of the estimated parameters.

```
// T23estDisc.sce
// ESTIMATION OF DISCRETE MODEL
//    f(y(t)|u(t),y(t-1)) with y,u from {0,1}
// Experiments
// - change number of values of individual variables
// - increase the model order
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1)

// SIMULATION
nd=500;                         // number of steps
// parameters of simulation
thS= [.2 .8
      .6 .4
      .9 .1
      .3 .7];
ut=(rand(1,nd,'u')>.3)+1;       // control variable P(ut=1)=.3, P(ut=2)=.7
yt(1)=1;                        // initial condition for output

// time loop of simulation
for t=2:nd
  i=2*(ut(t)-1)+yt(t-1);         // row in the model parameter
  yt(t)=(rand(1,1,'u')>thS(i,1))+1;  // generation of the output
end

// ESTIMATION
V=zeros(4,2);                   // initial statistics
for t=2:nd
  i=2*(ut(t)-1)+yt(t-1);        // row of model matrix
  V(i,yt(t))=V(i,yt(t))+1;      // updt of statistics
  thp=V./(sum(V,2)*ones(1,2)); // pt estimates of parameters
  thE(:,t)=thp(:,1);
```

```
end

// Results
set(scf(1),'position',[600 60 600 500])
for i=1:4
  subplot(4,1,i)
  plot(thE(i,:))                                    // estimated
  plot((nd-199:nd),ones(1,200)*thS(i,1),':r','linewidth',2)     // true
  set(gca(),'data_bounds',[0 nd -.1 1.1])
  if i==1,
    title('Evolution ot parameter estimates (left column, only)')
    legend('estimated','true',[350 1.2]);
  end
end

disp(thS,'Simulated parameter')
disp(thp,'Estimated parameter')
```

The result is



where the evolution of estimated parameters (probabilities for $y_1 = 1$) are displayed (blue curves) and compared to the true values of these parameters (red piece of line). Again, as in the previous program we can see that some 100 data would be enough for good estimation.

Remark

Only probabilities of $y_t = 1$ are treated. Those for y_t =2 are complementary to one.

15.7 Prediction with continuous model

The program performs general np-nstep prediction with the regression model. The prediction gives point estimate of the future output and in th regression vector it uses also point estimates of the unknown output. The program runs as if in real time. It has only one time loop in which both simulation and subsequently prediction runs.

The prediction is simple - it consists in repetitive use on the model always shifted one step ahead. However, its realization is a bit tricky. We must distinguish the first step of the np-nstep prediction when the regression vector get the really measured values of the output. For the subsequent steps the output is unknown and is replaced by its prediction from the previous time instants. So, in the part of prediction, the first step is performed with the regression vector ps filled by measured values of the inputs and outputs. Then a local loop begins with time tj which is a virtual time of future steps of the prediction tj = t+1, t+2, ..., t+np. For these steps the regression vector is constructed from the old one, so that it takes new value of input, then last predicted output, then the old regression vector with the last three steps committed (too old data) and with 1 at its end.

Example

If e.g. the old regression vector was $\psi_3 = [u_3, y_2, u_2, y_1, u_1, 1]$ *then the new (shifted) one, using prediction for* $y_3 = \hat{y}_3$ *will be* $\psi_4 = [u_4, \hat{y}_3, u_3, y_2, u_2, 1]$ *where the part* u_3, y_2 *is taken from* ψ_3.

```
// T31preCont.sce
// NP-STEP PREDICTION WITH CONTINUOUS MODEL (KNOWN PARAMETERS)
// Experiments
// Change: - np = number of steps of prediction
//         - r  = noise variance
//         - th = model parametrs
//         - u  = input signal
// ---------------------------------------------------------------
exec("ScIntro.sce",-1),mode(0)

nd=100;                              // number of data
np=5;                                // length of prediction (np>=1)
//  b0 a1 b1  a2 b2 k
th=[1 .4 -.3 -.5 .1 1]';             // regression coefficients
r=.02;                               // noise variance
u=sin(4*%pi*(1:nd)/nd)+rand(1,nd,'norm'); // input
y(1)=1; y(2)=-1;                     // prior data


// TIME LOOP
for t=3:(nd-np)                      // time loop (on-line tasks)
  // prediction
```

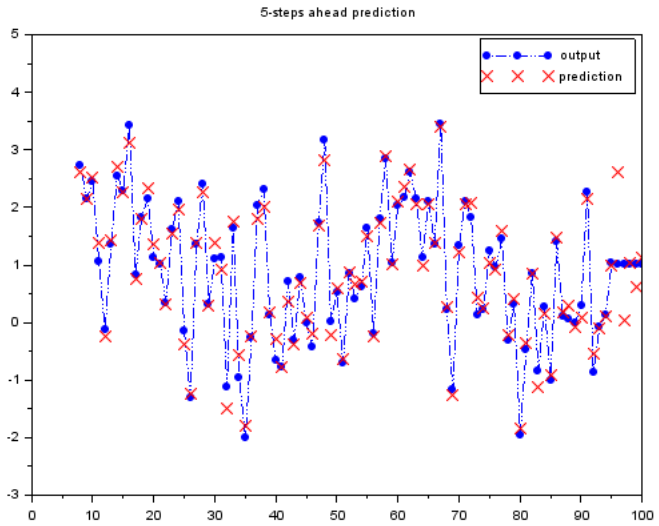
```
  ps=[u(t) y(t-1) u(t-1) y(t-2) u(t-2) 1]'; // first reg. vec for prediction
  yy=ps'*th;                        // zero prediction for time = t (np=0)
  for j=1:np                        // loop of predictions for t+1,...,t+np
    tj=t+j;                         // future times for prediction
    ps=[u(tj); yy; ps(1:$-3); 1];  // reg.vecs with predicted outputs
    yy=ps'*th;                      // new prediction (partial)
  end
  yp(t+np)=yy;                      // final prediction for time t+np

  // simulation
  ps=[u(t) y(t-1) u(t-1) y(t-2) u(t-2) 1]';  // regression vector for sim.
  y(t)=ps'*th+sqrt(r)*rand(1,1,'norm');      // output generation
end

// Results
s=(np+3):(nd-np);
scf(1);
plot(s,y(s),'.:',s,yp(s),'rx')
set(gca(),"data_bounds",[1 nd -3 5])
legend('output','prediction');
title(string(np)+'-steps ahead prediction')

RPE=variance(y(s)-yp(s))/variance(y)    // relative prediction error
```

After the local loop for `j=2:np` the final predicted value for time `t+np` is remembered at `yp(t+np)`.

In the end the results are plotted - here the generated and predicted outputs are compared.

The result is

Relative prediction error RPE = 0.07



The 5-step prediction is demonstrated, here. The differences between output (blue points) and their predictions (red crosses) are small for 5 steps. This fact is also expressed by the Relative prediction error

$$RPE = \frac{\text{var}\,(y - yp)}{\text{var}\,(y)}$$

which says how great is the prediction error relative to the output noise. The reason is: (i) the prediction knows precisely the model parameters, (ii) the same regression model is used for both simulation and prediction, (iii) the variance of the model noise is small ($r = 0.02$).

Prediction under model structure mismatch

Prediction without estimation in case of models mismatch does not make any sense. We will show individual cases of this task in the following Paragraph 15.8 where prediction combined with estimation is treated.

15.8 Adaptive on-line prediction with continuous model

Here, the prediction based on the model with unknown parameters is realized. The program is a combination of those performing estimation and prediction.

It starts in a standard way: length of data `nd` and prediction `np`, definition of regression model parameters (for simulation - at prediction they are as if unknown) and initialization of both estimation and prediction.

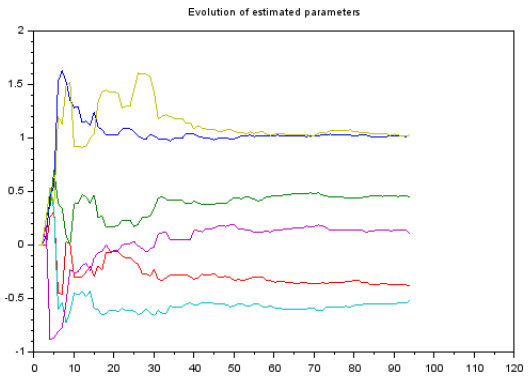
The time loop performs three subsequent steps. First simulation then, with the newly generated data, estimation and finally prediction. All these steps are described earlier. Namely at Paragraphs 15.1, 15.5 and 15.7.

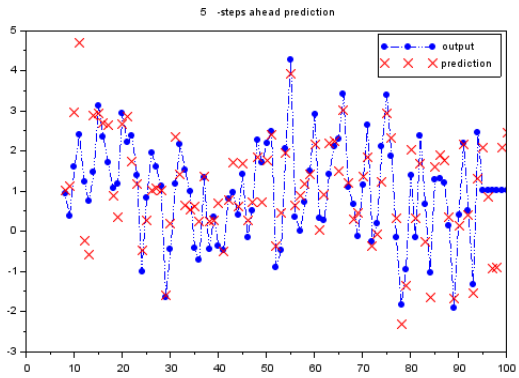
In the end of the program the results are printed and plotted. The parameter for simulation are compared to those coming from estimation, the evolution of estimated parameters are shown and the output and predicted output are compared.

```
// T32preCont_Adapt.sce
// N-STEP PREDICTION WITH CONTINUOUS MODEL (WITH ESTIMATION)
// Experiments
// Change: - np = number of steps of prediction
//         - r  = noise variance (effect on estimation)
//         - th = model parametrs
//         - u  = input signal (effect on estimation)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1),mode(0)

nd=100;                              // number of data
np=5;                                // length of prediction (np>=1)
nz=3;                                // starting time (ord+1)
//  b0 a1 b1  a2 b2 k
th=[1 .4 -.3 -.5 .1 1]';             // regression coefficients
r=.2;                                // noise variance
u=sin(4*%pi*(1:nd)/nd)+rand(1,nd,'norm'); // input
y(1)=1; y(2)=-1;                     // prior data
Eth=rand(6,1,'n');                   // prior parametrs

nu=zeros(4,2);
for t=nz:(nd-np)                     // time loop (on-line tasks)
  // prediction
  ps=[u(t) y(t-1) u(t-1) y(t-2) u(t-2) 1]'; // regression vector
  yy=ps'*Eth;                              // first prediction at t+1
  for j=1:np                               // loop of predictions for t+2,..,t+np
    tj=t+j;                                // future times for prediction
    ps=[u(tj); yy; ps(1:$-3); 1];          // reg.vecs with predicted outputs
```

```
    yy=ps'*Eth;                          // new prediction (partial)
  end
  yp(t+np)=yy;                            // final prediction for time t+np

  // simulation
  ps=[u(t) y(t-1) u(t-1) y(t-2) u(t-2) 1]';  // regression vector for sim.
  y(t)=ps'*th+sqrt(r)*rand(1,1,'norm');      // output generation

  // estimation
  Ps=[y(t) u(t) y(t-1) u(t-1) y(t-2) u(t-2) 1]';  // reg.vect. for estim.
  if t==nz, V=1e-8*eye(length(Ps)); end // initial information matrix
  V=V+Ps*Ps';                          // update of statistics
  Vp=V(2:$,2:$);
  Vyp=V(2:$,1);
  Eth=inv(Vp+1e-8*eye(Vp))*Vyp;        // point estimates
  Et(:,t)=Eth(:,1);                    // stor est. parameters
end

// Results
disp(' Simulated parameters')
disp(th)
disp(' Estimated parameters')
disp(Eth)

set(scf(1),'position',[100 100 1200 400]);
subplot(121),plot(Et')
set(gca(),"data_bounds",[0 nd+1 -1 2])
title('Evolution of estimated parameters')
subplot(122)
s=(np+3):(nd-np);
plot(s,y(s),'.:',s,yp(s),'rx')
set(gca(),"data_bounds",[1 nd -3 5])
legend('output','prediction');
title([string(np),'-steps ahead prediction'])
```

Results of the program are here





The left picture shows evolution of parameters point estimates (you can see that they satisfactorily stabilize) and the right one presents comparison of outputs and their predictions.

The values of the simulation parameters and their estimates is here

	b_0	a_1	b_1	a_2	b_2	k
simulation	1	0.4	-0.3	-0.5	0.1	1
estimation	1.026	0.441	-0.386	00.521	0.093	1.023

As the structures of both models is the same, the parameters should correspond. And they do.

Prediction under model structure mismatch

Here, no parameters that could be inserted into the model exist. So, we must estimate (mostly continuously point estimates) and use them in the task of prediction. The theoretical basis for prediction is given in(7.2) and point estimation is according to (6.4).

- *Prediction with 3^{rd} order model in simulation and 2^{nd} order one used for estimation*

Here, the dataset is simulated with the model of the third order and a model of the second order is used for estimation. It is a first step of approaching reality, where no parameters that can be used in model exist.

The task resembles an on-line real application where no data are known beforehand (up to some prior ones). So, there is only one loop simulating time progress. Here, (*i*) the simulation produces new data (input u_t is given for the whole task time interval). Then (*ii*) the estimation is performed giving actual point estimates of parameters. Last (*iii*) the prediction is computed based on point estimates of parameters and providing point estimates of the future output.

The excitation of the system is in the optimal form - a sine function with the added noise. Other types of excitation can be also tested - see programs with estimation with model mismatch.

```
// T32preCont_Adapt2.sce
// N-STEP PREDICTION WITH CONTINUOUS MODEL (WITh ESTIMATION)
// The model for simulation differs from that for estimation
// i.e. MODEL STRUCTURE MISMATCH is tackled.
// Experiments
// Change: - np = number of steps of prediction
//         - r  = noise variance (effect on estimation)
//         - th = model parametrs
//         - u  = input signal (effect on estimation)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1),mode(0)

nd=100;                              // number of data
np=3;                                // length of prediction (np>=1)
//  b0 a1 b1  a2 b2 a3 b3 k
th=[1 .4 -.3 -.5 .1 .6 .1 1]';       // regression coefficients
r=.2;                                // noise variance
u=sin(4*%pi*(1:nd)/nd)+rand(1,nd,'norm'); // input
y(1)=1; y(2)=-1; y(3)=0;             // prior data
Eth=rand(8,1,'n');                   // prior parametrs

nu=zeros(4,2);
yp=ones(1,nd);
nz=4;
for t=nz:(nd-np)                     // time loop (on-line tasks)
  // prediction
  ps=[u(t) y(t-1) u(t-1) y(t-2) u(t-2) y(t-3) u(t-3) 1]';  // regression vector
  yy=ps'*Eth;                        // first prediction at t+1
  for j=1:np                         // loop of predictions for t+2,...,t+np
    tj=t+j;                          // future times for prediction
    ps=[u(tj); yy; ps(1:$-3); 1];    // reg.vecs with predicted outputs
    yy=ps'*Eth;                      // new prediction (partial)
  end
  yp(t+np)=yy;                       // final prediction for time t+np

  // simulation
  ps=[u(t) y(t-1) u(t-1) y(t-2) u(t-2) y(t-3) u(t-3) 1]';  // regression vector for sim.
  y(t)=ps'*th+sqrt(r)*rand(1,1,'norm'); // output generation

  // estimation
  Ps=[y(t) u(t) y(t-1) u(t-1) y(t-2) u(t-2) y(t-3) u(t-3) 1]';  // reg.vect. for estim.
  if t==nz, V=1e-8*eye(length(Ps)); end // initial information matrix
  V=V+Ps*Ps';                          // update of statistics
  Vp=V(2:$,2:$);
```

```
  Vyp=V(2:$,1);
  Eth=inv(Vp+1e-8*eye(Vp))*Vyp;         // point estimates
  Et(:,t)=Eth(:,1);                     // stor est. parameters
end

// Results
disp(' Simulated parameters')
disp(th)
disp(' Estimated parameters')
disp(Eth)

set(scf(1),'position',[100 100 1200 400]);
subplot(121),plot(Et')
set(gca(),"data_bounds",[0 nd+1 -1 2])
title('Evolution of estimated parameters')
legend('b0E','a1E','b1E','a2E','b2E','kE');
subplot(122)
s=(np+nz):(nd-np);
plot(s,y(s),'.:',s,yp(s),'rx')
set(gca(),"data_bounds",[1 nd -6 10])
legend('output','prediction');
title([string(np),'- step ahead prediction'])
```

• *Prediction with real data and model of the optional order* ord *pre-set for 5*

Here, the prediction with real data is demonstrated.

The results can be evaluated by the quality of prediction (with bad estimation we cannot have good prediction) either visually in graphs or numerically by Relative Prediction Error RPE which shows the ratio of noise in the prediction with respect to the noise in the output.

```
// T32preCont_Adapt3.sce
// np-STEP PREDICTION WITH CONTINUOUS MODEL (WITh ESTIMATION)
// REAL DATA (intensity) from Strahov tunnel are used
// i.e. MODEL STRUCTURE MISMATCH is tackled.
// Experiments
// Change: - np  = number of steps of prediction
//         - ord = order of the model for estimation
// Result: - visual comparison of yt and yp
//         - RPE = relative prediction error
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1),mode(0)

np=5;                                // length of prediction (np>=1)
ord=2;                               // order of the estimated model
```

```
// data selection
k=3;                                // which day is selected
nz=50;                              // beginning of the day
nd=288*2;                           // length of the whole one day data

// DATA
dtAll=csvRead('STRAHOV.csv',';');
dt=dtAll((k-1)*288+(nz+1:nd),1);
nth=ord+2;                          // size of V
V=1e-8*eye(nth,nth);                // initial information matrix
thE=rand(nth-1,1,'n');              // prior parametrs
yt=dt(1:ord);                       // prior data

for t=ord+1:(nd-np-nz)              // time loop (on-line tasks)
//  PREDICTION
  ps=[yt(t-1:-1:t-ord); 1];         // regression vector
  yy=ps'*thE;                       // first prediction at t+1
  for j=1:np                        // loop of predictions for t+2,...,t+np
    tj=t+j;                         // future times for prediction
    ps=[yy; ps(1:(ord-1)); 1];      // reg. vector
    yy=ps'*thE;                     // new prediction (partial)
  end
  yp(t+np,1)=yy;                    // final prediction for time t+np

  // DATA MEASUREMENT (as if)
  yt(t)=dt(t);                      // measuring of output

  // ESTIMATION
  psi=[yt(t:-1:t-ord); 1];                  // reg. vector
  V=V+psi*psi';                             // updt of information matrix
  Vy=V(1,1); Vyp=V(2:$,1); Vp=V(2:$,2:$);   // divisioning of inf. matrix
  thE(:,1)=inv(Vp+1e-8*eye(Vp))*Vyp;        // pt estimates of reg. coefficints
  r(t)=(Vy-Vyp'*inv(Vp+1e-8*eye(Vp))*Vyp)/t; // pt estimates of noise variance
  Et(:,t)=thE;                              // stor est. parameters
end

// Results
// evolution of parametrs
set(scf(1),'position',[100 100 1200 400]);
subplot(121),plot(Et')
set(gca(),"data_bounds",[0 nd+1 -1 5])
title('Evolution of estimated parameters')
legend('b0E','a1E','b1E','a2E','b2E','kE');
// comparison of yt and yp
```

```
subplot(122)
s=1:length(yt);
plot(s,yt(s),'.:',s,yp(s),'rx')
set(gca(),"data_bounds",[1 nd -4 max(yt)+5])
legend('output','prediction');
title([string(np),'- step ahead prediction'])

RPE=variance(yt(s)-yp(s))/variance(yt(s))
```

15.9 Prediction with discrete model

Here, we are going to make a prediction with discrete (categorical) model with data simulated off-line, i.e. the whole dataset is generated beforehand. Indeed, in prediction we do as if the present and future outputs are unknown and measured continuously.

The program starts as always. Then goes the loop for simulation and the loop for prediction follows. The simulation is according to Paragraph 15.2.

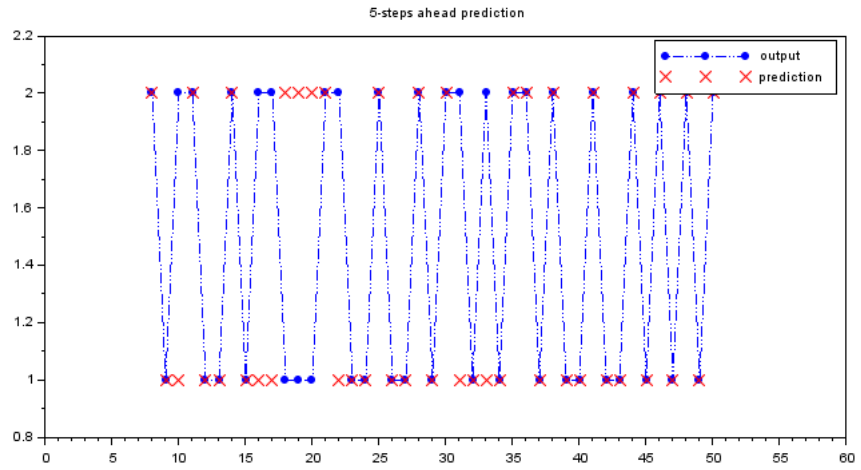
The loop for prediction, that is the core of this program, follows the prediction with regression model - see Paragraph 15.7 with only the difference, that for both simulation and prediction is used discrete model. The simulation is discussed in Paragraph 15.2 and the same command is used also in prediction.

As a result, output and predicted output are compared in graph.

```
// T33preCat_Off.sce
// PREDICTION WITH DISCRETE MODEL (OFF-LINE)
// Experiments
// Change: - np  = number of steps of prediction
//         - th1 = model parametrs
//         - u   = input signal (effect on estimation)
//         - uncertainty of the system (effect on estimation)
// ---------------------------------------------------------------
exec("ScIntro.sce",-1),mode(0)

nd=50;                              // length of data sample
np=0;                               // length of prediction (np>=1)
th1=[0.98 0.01 0.04 0.97]';         // parameters for simulation (for y=1)
th=[th1 1-th1];                     // all parameters
u=(rand(1,nd)>.3)+1;                // input
y(1)=1;

// SIMULATION
for t=2:nd
  i=2*(u(t)-1)+y(t-1);              // row of the table
  y(t)=(rand(1,1,'u')>th(i,1))+1;   // output generation
end

// PREDICTION
yy=ones(1,nd);                      // fictitious predicted output
for t=2:(nd-np)
  i=2*(u(t)-1)+y(t-1);              // row of the table
  yy=(rand(1,1,'u')>th(i,1))+1;     // prediction generation
  for j=1:np
```

```
    i=2*(u(t+j)-1)+yy;              // row of the table
    yy=(rand(1,1,'u')>th(i,1))+1;   // prediction generation
  end
  yp(t+np)=yy;                      // np-step predction
end

// Results
disp(th,' Model parameters'), disp(' ')

s=(np+3):nd;
plot(s,y(s),'.:',s,yp(s),'rx')
set(gcf(),'position',[600 100 800 400])
set(gca(),"data_bounds",[0 nd+1 .9 2.1])
legend('output','prediction');
title(string(np)+'-steps ahead prediction')

Wrong=sum(y(:)~=yp(:)), From=nd
```

The result of the program is in the following figure



The model for simulation is near to deterministic one, s even if the prediction is five steps ahead it is not bad. Only 16 out of 50 predictions is wrong.

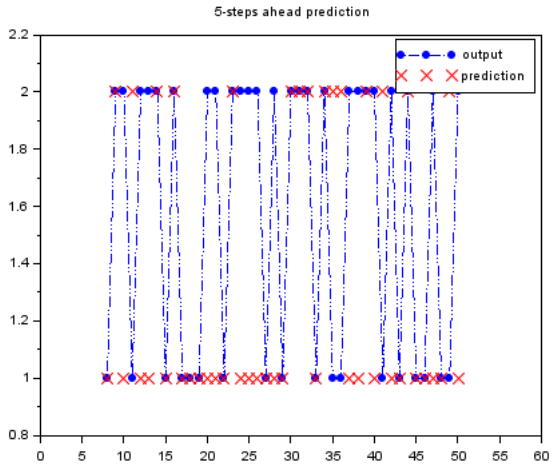
15.10 Adaptive prediction with discrete model

Here, a prediction based on a discrete model with unknown parameters is demonstrated. The program is practically identical with the previous one, only estimation is inserted between simulation and prediction. The combination of estimation and prediction can be seen also in Paragraph 15.8.

```
// T34preCat_OffEst.sce
// PREDICTION WITH DISCRETE MODEL (OFF-LINE)
// Experiments
// Change: - length of prediction
//         - uncertainty of the simulated model
//         - imput signal (effect on estimation)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1),mode(0)

nd=50;                              // number of data
np=5;                               // length of prediction
th1=[0.8 0.1 0.4 0.7]';             // parameters for simulation (for y=1)
th=[th1 1-th1];                     // all parameters
u=(rand(1,nd)>.3)+1;                // input
y=ones(1,nd);

// SIMULATION
for t=2:nd
  i=2*(u(t)-1)+y(t-1);              // row of the table
  y(t)=(rand(1,1,'u')>th(i,1))+1;   // output generation
end

// ESTIMATION
nu=zeros(4,2);
for t=2:nd
  i=2*(u(t)-1)+y(t-1);              // row of the table
  nu(i,y(t))=nu(i,y(t))+1;          // statistics update
end
Eth=nu./(sum(nu,2)*ones(1,2));      // estimate of parameters

// PREDICTION
yy(1)=1;                            // fictitious predicted output
for t=2:(nd-np)
  i=2*(u(t)-1)+y(t-1);              // row of the table
  yy=(rand(1,1,'u')>th(i,1))+1;     // prediction generation
  for j=1:np
    i=2*(u(t+j)-1)+yy;              // row of the table
```

```
    yy=(rand(1,1,'u')>th(i,1))+1;  // prediction generation
  end
  yp(t+np)=yy;                     // np-step predction
end

// Results
disp(' Simulated parameters')
disp(th)
disp(' Estimated parameters')
disp(Eth)

s=(np+3):nd;
plot(s,y(s),'.:',s,yp(s),'rx')
set(gcf(),'position',[300 100 500 400])
set(gca(),"data_bounds",[0 nd+1 .9 2.1])
legend('output','prediction');
title(string(np)+'-steps ahead prediction')

Wrong=sum(y(:)~=yp(:))
From=nd
```

The results are here



where again with 5-steps ahead predictions is used and 27 out of 50 of them are wrong. The simulated and estimated parameters are:

	Simulation			Estimation	
u_t, y_{t-1}	$y_t = 1$	$y_t = 2$		$y_t = 1$	$y_t = 2$
1 1	0.8	0.2		1	0
1 2	0.1	0.9		0.083	0.917
2 1	0.4	0.5		0.211	0.789
2 2	0.7	0.3		0.765	0.235

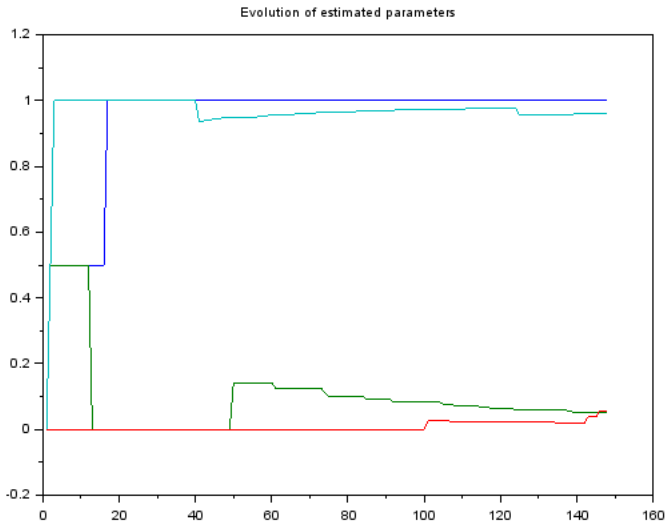
15.11 Adaptive on-line prediction with discrete model

Again the same case as in the previous program is presented here with the difference that all the tasks (simulation, estimation and prediction) are performed in one time loop. It means, the situation that occurs in a practical task, when the data are measured and used continuously.

```
// T35preCat_OnEst.sce
// PREDICTION WITH DISCRETE MODEL (ON-LINE)
// Change: - length of prediction
//         - uncertainty of the simulated model
//         - imput signal
//         - study the beginning when estimation is not finished
//           how can we secure quicker transient phase of estimation?
// Remark: another way og generation is
//            y(t)=sum(rand(1,1,'u')>cumsum(th(i,:)))+1;
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1),mode(0)

nd=150;                             // number of data
np=2;                               // length of prediction
th1=[0.98 0.01 0.04 0.97]';         // parameters (for y=1)
th=[th1 1-th1];                     // all parameters
u=(rand(1,nd+np,'u')>.3)+1;         // input
y(1)=1;

// TIME LOOP
nu=1e-8*ones(4,2);
Et=zeros(4,nd-np);
for t=2:nd                          // time loop
  // prediction
  i=2*(u(t)-1)+y(t-1);              // row of the table
  yy=(rand(1,1,'u')>th(i,1))+1;     // prediction generation
  for j=1:np
    i=2*(u(t+j)-1)+yy;              // row of the table
    yy=(rand(1,1,'u')>th(i,1))+1;   // prediction generation
  end
  yp(t+np)=yy;                      // np-step predction

  // simulation
  i=2*(u(t)-1)+y(t-1);
  y(t)=(rand(1,1,'u')>th(i,1))+1;

  // estimation
  i=2*(u(t)-1)+y(t-1);              // row of model matrix
```

```
  nu(i,y(t))=nu(i,y(t))+1;          // statistics update
  Eth=nu./(sum(nu,2)*ones(1,2));    // pt estimates
  Et(:,t)=Eth(:,1);
end

// Results
disp(' Simulated parameters')
disp(th)
disp(' Estimated parameters')
disp(Eth)

s=np+2:np+51;
set(scf(),'position',[100 100 1000 400])
subplot(121),plot(Et')
title('Evolution of estimated parameters')
set(gca(),"data_bounds",[0 nd-np+1 -.1 1.1])
subplot(122),plot(s,y(s),s,yp(s),'.:')
title('First 50 outputs and their prediction')
set(gca(),"data_bounds",[s(1) s($) .9 2.1])

s=np+2:nd;
Wrong=sum(y(s)~=yp(s))
From=nd-np
```

The results of the program are



where the picture shows time evolution of parameter point estimates (only left column of the model matrix is shown - the other one is a complement to one) The accuracy of the prediction (i.e. the ratio of the good predictions relative to the data length) is 0.162 which is due to only 2-steps ahead prediction and the mode near to deterministic one.

15.12 State estimation

This program deals with an unknown variable, called state, whose values must be estimated on the basis of information gained from the measured input and output. The estimation uses Kalman filter.

At the beginning of the program, parameters of the state-space model for simulation, input signal and initialization of simulation are performed.

Then loop for simulation follows.

After the simulation, all what is necessary for Kalman filter is prepared. They are covariance matrices and values of the initial state. As we have said in the theory (see Paragraph 8.2) the covariance of the state estimate Rx is set as a diagonal one with large diagonal (here 1000). The covariances Rw of the state noise and Rv of the output noise are problematic. Here, the situation is simple, as we work with simulated data and thus we know these covariances from simulation. However, in practice we must sometimes experiment with their setting.

The estimation itself is very simple and it consists in recursive call of the Kalman filter function. It recomputes the state expectation xp and covariance Rx. What we are mostly interested is continuous point estimate of the state.

Remarks

1. *The square brackets* [] *in the argument of Kalman filter stay for constants of the space state model - here they are not used. The full state-space model has the form*

$$x_t = Mx_{t-1} + Nu_{t-1} + F + w_t$$
$$y_t = Ax_t + Bu_t + G + v_t$$

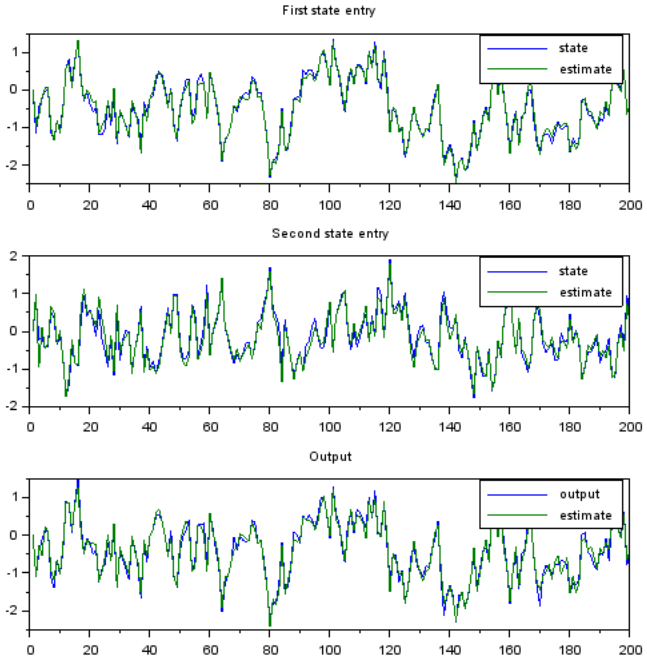
   where F *and* G *are constants of the model.*

2. *Do not forget to use the command* getd() *at he beginning of the program to load the Kalman filter at memory.*

```
// T46statEst_KF.sce
// STATE ESTIMATION (KALMAN FILTER)
// Experiments
// - change model parameters M,N,A,B
// - set different system and model covariances rw,rv and Rw,Rv
// - try lower stat-estimate covariance Rx
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1), getd()

nd=200;                 // number of data
// SIMULATION
// parameters of simulation
```

```
M=[.8 .1
   .3 .6];
N=[.5 -.5]';
A=[.9 -.2];
B=0;
rw=.1*eye(2,2);
rv=.1;
xt(:,1)=[0 0]';
ut=rand(1:nd,'n');
// time loop of simulation
for t=2:nd
  xt(:,t)=M*xt(:,t-1)+N*ut(t)+rw*rand(2,1,'n');
  yt(t)  =A*xt(:,t)+B*ut(t)+rv*rand(1,1,'n');
end

// ESTIMATION
// initialization of estimation
Rw=.1*eye(2,2);                     // state noise covariance
Rv=.1;                             // output noise vovariance
Rx=1000*eye(2,2);                  // estimated state covariance
xp(:,1)=zeros(2,1);                // initial state
// loop for state estimation
for t=2:nd
  [xp(:,t),Rx,yp(t)]=Kalman(xp(:,t-1),yt(t),ut(t),M,N,[],A,B,[],Rw,Rv,Rx);
end

// RESULTS
subplot(311),plot(1:nd,xt(1,:),1:nd,xp(1,:))
set(gcf(),"position",[700 100 600 500])
title('First state entry')
legend('state','estimate')
subplot(312),plot(1:nd,xt(2,:),1:nd,xp(2,:))
title('Second state entry')
legend('state','estimate')
subplot(313),plot(1:nd,yt,1:nd,yp')
title('Output')
legend('output','estimate')
```

The result is



Here, two dimensional state and scalar output are considered. The first two graphs in the picture show two components of the state in comparison with their estimates. The last graph shows comparison of the output and its prediction.

The data are simulated, so, the exact covariances of state and output are known and used. That is why the convergence of the algorithm is so fast. Also the precision is very good.

15.13 Noise filtration

Kalman filter is often used as a noise filter. Here, we will show how it can be done.

Let us have a true signal g_t that is corrupted by noise v_t. We can measure only the corrupted signal $y_t = g_t + v_t$. We introduce the pure unknown variable as a state $x_t = g_t$. Then the state-space model takes the form

$$x_t = x_{t-1} + w_t$$
$$y_t = x_t + w_t$$

where we say: the unknown signal is smooth (it changes only by small increments -values of the noise w_t with relatively small variance). What we measure is the pure signal with noise.

Now, variance of the noise w_t determines how big changes in the unknown signal are admitted; variance of the noise v_t says how big changes of the noise v_t are expected. I.e. which changes are to be assigned to signal changes and which are to be covered by noise.

After a definition of the state-space model and setting the values to R_w and R_v,standard Kalman filter i called.

As a result, we plot both the noisy and estimated signals. Here, we can judge quality of the estimate and possibly change the covariances.

Remark

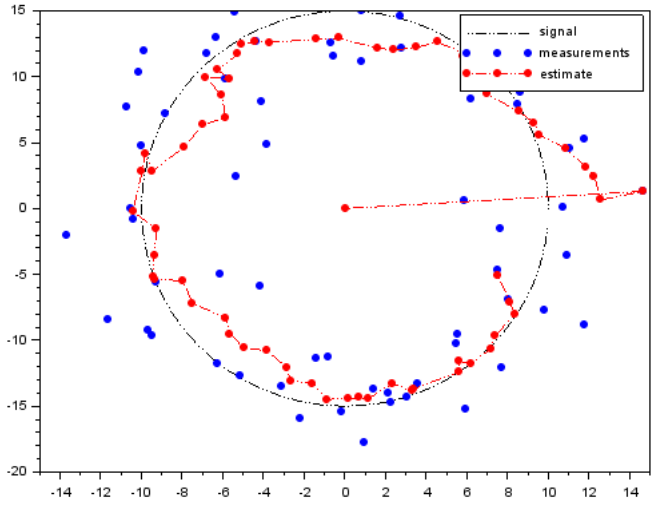
If the estimated signal is too noisy, we should enlarge the covariance of the output model.

If the estimated signal does not follow the average of the measured variable, we should enlarge the covariance of the state noise.

```
// T47statEst_Noise.sce
// KALMAN AS A NOISE FILTER
// Experiments
// - change Rw and Rv to catch properly the signal
// - Rw ... changes of the signal
// - Rv ... changes of the noise
// ---------------------------------------------------------------------
exec SCIHOME/ScIntro.sce, mode(0), getd()

// SIMULATION
tt=0:.1:(2*%pi);
nt=length(tt);
sd=2;                                // simulation noise
e=[sd*rand(1,nt,'n'); sd*rand(1,nt,'n')];
g=[10*cos(tt); 15*sin(tt)];          // pure signal (ellipse)
x=g+e;                               // measured noisy signal
```

```
// FILTRATION
Rz=1e6*eye(2,2);                    // state-estimate cov. matrix
Rw=.01*eye(2,2);                    // state-model cov. matrix
Rv=.1*eye(2,2);                     // output-model cov. matrix
M=[1 0                              // state-model matrices
   0 1];
A=[1 0
   0 1];
N=[0 0]';
F=[0 0]';
B=0;
G=0;
zt(:,1)=[0 0]';                          // initial state

for t=2:nt
  [zt(:,t),Rz,yp]=Kalman(zt(:,t-1),x(:,t),0,M,N,F,A,B,G,Rw,Rv,Rz);
end

// Results
plot(g(1,:),g(2,:),'k:')
plot(x(1,:),x(2,:),'b.')
plot(zt(1,:),zt(2,:),'r.:')
legend('signal','measurements','estimate');
```

The result is here



The pure signal forms the circle. The output (pure signal + noise) is denoted by the blue dots. The estimate of the pure signal is the red curve. The goal is to obtain as smooth estimate as possible, however, following the pure signal. To this end the covariances of the state and output should be tuned.

Remark

In the simulated case, as this, we can see both the pure signal and its estimate. In a real case, we cannot see the pure signal. Then we demand the estimate to be smooth enough and to approximate the measured output well.

15.14 Control with regression model

This program demonstrates optimal control on a finite interval with a regression model with known parameters. The derivation of the control synthesis is based on the state-space model (see Paragraph 3.1) and uses the Bellman equations (see Paragraph 10.2). The penalty must be accommodated for the state-space approach.

The program starts with definition of regression model parameters and the penalization matrix Om (for its constructions see Paragraph 10.2). Also the variables S and R are introduced as list() (they will be vectors of matrices).

Then the control optimization follows in the loop for t=nd:-1:2 here the matrix S(t) of the optimal control is constructed against the direction of time.

After it, the value of the control variable u(t) is computed in the direction of time using the corresponding entry of the list S: u(t)=-Sx(t-1) where the last state is x(t-1)=[y(t-1) u(t-1) y(t-2) u(t-2) 1]. After computing of u(t), it is immediately used for generation of new value of the output y(t).

As a result the controlled output is plotted.

Remark

The program listed uses a control of the output to the defined setpoint s_t. This can be done, if the control penalization is introduced in the form

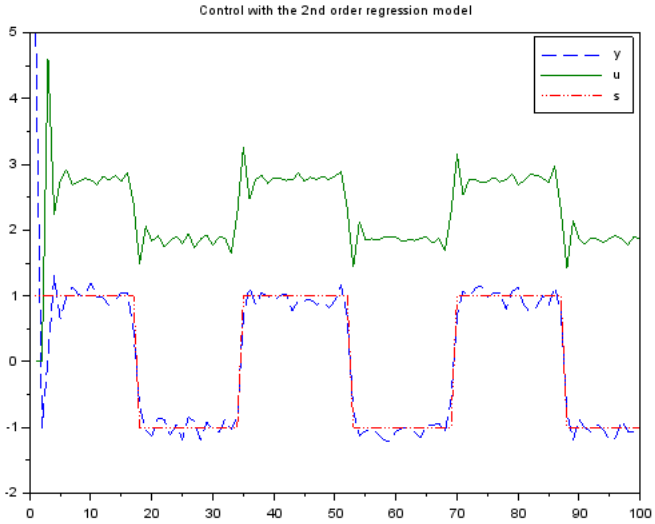
$$(y_t - s_t)^2 + \omega u_t^2.$$

The criterion aims at achieving $y_t - s_t \to 0$ and thus $y_t \to s_t$. The setpoint s_t must be defined and then it enters the program only in the penalization matrix in the loop of control optimization.

```
// T53ctrlX.sce
// Control with scalar 2nd order regression model
// - simulated data
//   y(t)=b0*u(t)+a1*y(t-1)+b1*u(t)+a2*y(t-2)+b2*u(t-2)+k+e(t);
// - state realization of the model for synthesis
// - control on a single control interval with the length nd
// - following a settpoint s(t)
// Experiments
// - change penalizations of input(om) and input increments (la)
// - set new setpoint s
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1), mode(0)

nd=100;                              // length of control interval
a1=.6; a2=-.2; b0=1; b1=.4; b2=-.1; k=-3; sd=.1; // regression model parameters
om=0; la=.1;                        // penalization (input, increment)
```

```
s=sign(10*sin(18*(1:nd)/nd));     // setpoint generation
// conversion to state-space model
M=[a1 b1 a2 b2 k
   0  0  0  0  0
   1  0  0  0  0
   0  1  0  0  0
   0  0  0  0  1];           // state matrix with set-point
N=[b0 1  0  0  0]';
Om=diag([1 om+la 0 la 0]);                 // matrix penalization
Om(2,4)=-la; Om(4,2)=-la;

S=list();
R=list();
R(nd+1)=zeros(Om);                 // initial condition for dyn. progr.

// CONTROL
// computation of control-law
for t=nd:-1:2
  Om(1,$)=-s(t); Om($,1)=-s(t); Om($,$)=s(t)**2;
  T=R(t+1)+Om;
  A=N'*T*N;
  B=N'*T*M;
  C=M'*T*M;
  S(t)=inv(A)*B;
  R(t)=C-S(t)'*A*S(t);
end

// control-law realization
y(1)=5; y(2)=-1;
u(1)=0; u(2)=0;
for t=3:nd
  u(t)=-S(t)*[y(t-1) u(t-1) y(t-2) u(t-2) 1]';     // optimaal control
  y(t)=a1*y(t-1)+a2*y(t-2)+b0*u(t)+b1*u(t-1)+b2*u(t-2)+k+sd*rand(1,1,'n');   // simulation
end

// RESULTS
x=1:nd;
plot(x,y(x),'--',x,u(x),x,s(x),':')
title 'Control with the 2nd order regression model'
legend('y','u','s');
```

The result of the control is in the following picture



The output follow the setpoint well, especially in the jumps. That is because the setpoint is known beforehand and the controller can get prepared for it.

15.15 Adaptive control with regression model

Adaptive control is suboptimal one. It uses so called receding horizon method. It consists in the following scheme (let us be at time t consider 3 steps control interval):

1. For existing point estimates of model parameters it designs control law for the control interval (from the end $t+3$ to the beginning t). From the designed control, we use only the first control (at time t), apply it and measure new output y_t.
2. With the newly measured data u_t and y_t we recompute estimation and obtain new point estimates of parameters.
3. We shift the control interval one step, i.e. for times $t+1$ up to $t+3+1$.

These three steps are repeated with the parameters estimates continuously improved. With improved parameters also the control should improve.

The program starts with definitions of various constants including model parameters for simulation.

Then pre-estimation runs - it is estimation with prior data still without control. This phase is almost necessity. If we would enter the full adaptive control with accidentally values of parameters then the control would be very bad and the worse the response of the system after application this control. Thus the system could be over-excited and and very probably the control would fail. After this pre-estimation we hope the parameters will be good enough to obtain a stabilized control process.

After pre-estimation the setpoint `s(t)` is defined, initial condition set and penalization matrix and matrices of the state-space model constructed.

Now, we have several loops. The above one is the loop for running time. Then the optimization loop follows running against the time. Here the first control is computed and immediately applied into the simulation. Ten, with the new data, the estimation goes, providing recomputed point estimates of parameters.

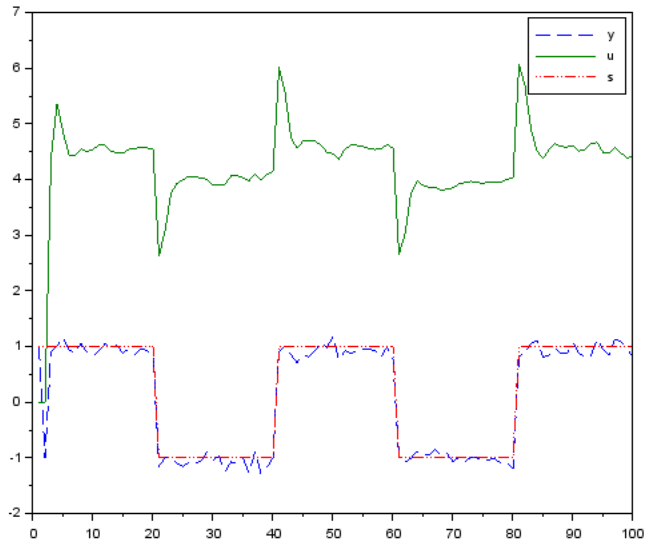
In the end, the results are shown.

```
// T54ctrlXEst.sce
// Control with scalar 1st order regression model
// - simulated data  y(t)=a*y(t-1)+b*u(t)+k+e(t);
// - state realization of the model for synthesis
// - control on a receding control interval with the length nh
// - following a settpoint s(t)
// Experiments
// - change setpoint and system parameters (slow - quick system)
// - penalization of input variable
// - initial condition for estimation (better or worse initial param.)
```

```
// - length of control interval nh
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1), mode(0)

nd=100;                              // number of data to be controlled
ni=20;                               // length of pre-etimation
nh=15;                               // length of control interval
a1=.6;  a2=.2;  b0=1;   b1=-.4;  b2=.1; k=-3;  // parameters for simulation
sd=.1;                               // stdev for simulation
om=.01; la=.001;                     // penalization of input / increments

// PRE-ESTIMATION
V=1e-8*eye(7,7);                     // initial information matrix
ui(1:2)=zeros(1,2); yi(1:2)=zeros(1,2);
for t=3:ni
  ui(t)=rand(1,1,'n');
  yi(t)=a1*yi(t-1)+a2*yi(t-2)+b0*ui(t)+b1*ui(t-1)+b2*ui(t-2)+k+.01*rand(1,1,'n');
  Ps=[yi(t) yi(t-1) yi(t-2) ui(t) ui(t-1) ui(t-2) 1]';
  V=V+Ps*Ps';
end
Vyp=V(2:$,1); Vp=V(2:$,2:$); thi=inv(Vp)*Vyp; // point estimates
a1E=thi(1); a2E=thi(2); b0E=thi(3); b1E=thi(4); b2E=thi(5); kE=thi(6);
thi=[a1E a2E b0E b1E b2E kE];

s=sign(100*sin(18*(1:nd+nh)/(nd+nh)));  // set-point
y(1)=1; y(2)=-1;                     // initial output
u(1)=0; u(2)=0;                      // initial control

Om=diag([1 om+la 0 la 0]);           // matrix penalization
Om(2,4)=-la; Om(4,2)=-la;
M=[a1E b1E a2E b2E kE
   0   0   0   0   0
   1   0   0   0   0
   0   1   0   0   0
   0   0   0   0   1];               // state-space model
N=[b0E 1   0   0   0]';               // state-space model

// COMPUTATION OF CONTROL-LAW
for t=3:nd          // loop for control
  R=0;                               // initial condition for dyn.prog.
  for i=nh:-1:1     // loop for receding horizon
    Om(1,$)=-s(t+i-1);
    Om($,1)=-s(t+i-1);
    Om($,$)=s(t+i-1)**2;
```

```
    T=R+Om;                          // dynamic
    A=N'*T*N;                        //    programming
    B=N'*T*M;
    C=M'*T*M;
    S=inv(A)*B;
    R=C-S'*A*S;
  end

// CONTROL REALIZATION (simulation)
  u(t)=-S*[y(t-1) u(t-1) y(t-2) u(t-2) 1]';             // optimal control value
  y(t)=a1*y(t-1)+a2*y(t-2)+b0*u(t)+b1*u(t-1)+b2*u(t-2)+k+sd*rand(1,1,'n');   // simulation

// ESTIMATION
  Ps=[y(t) y(t-1) y(t-2) u(t) u(t-1) u(t-2) 1]';
  V=V+Ps*Ps';
  Vyp=V(2:$,1);
  Vp=V(2:$,2:$);
  th=inv(Vp)*Vyp;                  // point estimates
  a1E=th(1);                       // regression
  a2E=th(2);                       //    coefficients
  b0E=th(3);
  b1E=th(4);
  b2E=th(5);
  kE=th(6);
end            // of loop for control

// RESULTS
th=[a1 a2 b0 b1 b2 k];             // simulated parameters
thE=[a1E a2E b0E b1E b2E kE];      // estimated parameters
z=1:nd;
set(scf(),'position',[900 50 600 500])
plot(z,y(z),'--',z,u(z),z,s(z),':')
legend('y','u','s');

disp(th,'simulated parametrs')
disp(thi,'initial parametrs')
disp(thE,'estimated parametrs')
```

The result of th program is here



In th figure we can see the setpoint (blue), the controlled output (red) as well as the control signal (green). We can see that the estimation is very fast - practically from the very beginning the output follows the setpoint satisfactorily.

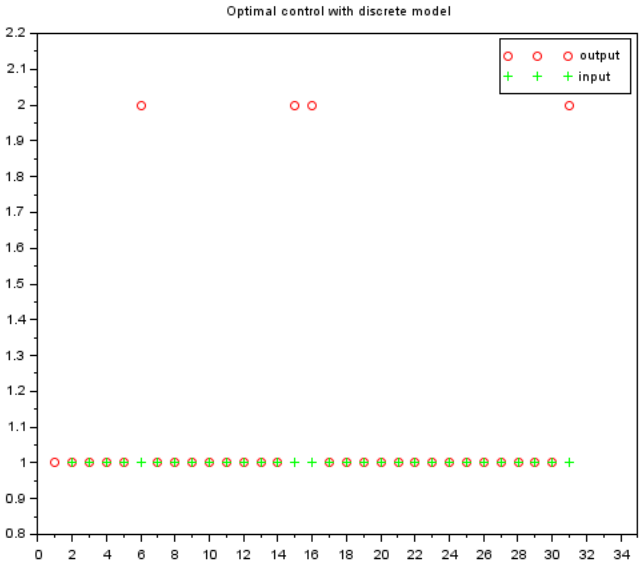
15.16 Control with discrete model

This control runs similarly as those with continuous model according to the Bellman equations (see Paragraph 11.1). However, its realization is not easy and we let the reader to look at the program, if he is interested. In any case, the program should run and it is possible to experiment with it.

```
// T52ctrlDisc.sce
// CONTROL WITH DISCTRETE DYNAMIC MODEL
// Experiments
// Change: - criterion om
//         - uncertainty of the system
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1),getd()

// VARIABLES TO BE SET
nh=30;                          // length of control interval
y0=1;                           // initial condition for output

//y 1 2        u y1  =  criterion
om=[1 2     // 1 1
    2 3     // 1 2
    2 3     // 2 1
    3 4];   // 2 2  ... preference of lower indexes

//y  1  2      u y1  =  system model
th=[.9 .1   // 1 1
    .4 .6   // 1 2
    .8 .2   // 2 1
    .1 .9]; // 2 2   ... rather uncertain system

// computed variables and initializations
fs=zeros(1,2);

// CONTROL LAW COMPUTATION
for t=nh:-1:1
  fp=om+ones(4,1)*fs;           // penalty + reminder from last step
  // expectation
  f=sum((fp.*th),'c');            // expectation over y
  // minimization
  if f(1)<f(3),                 // for y(t-1)=1
    us(t,1)=1; fs(1)=f(1);      // optimal control, minimum of criterion
  else
    us(t,1)=2; fs(1)=f(3);      // optimal control, minimum of criterion
```

```
  end
  if f(2)<f(4),                // for y(t-1)=2
    us(t,2)=1; fs(2)=f(2);     // optimal control, minimum of criterion
  else
    us(t,2)=2; fs(2)=f(4);     // optimal control, minimum of criterion
  end
end
J=fs(y0);                      // final value of criterion

// CONTROL APPLICATION
y(1)=y0;
for t=1:nh
  u(t+1)=us(t,y(t));           // optimal control
  y(t+1)=dsim(u(t+1),y(t),th); // simulation
end

// RESULTS
plot(1:nh+1,y,'ro',1:nh+1,u,'g+')
set(gcf(),"position",[700 100 600 500])
legend('output','input')
set(gca(),'data_bounds',[.8 nh+.2, .8 2.2])
title('Optimal control with discrete model')

printf('\n Minimal value of the expectation of criterion: %g\n',J)
```

The result is here



The penalization gives preference to lower numbers for both input and output. We can see this preference in the figure. The errors are caused by uncertainty in th model.

16 Supporting subroutines

16.1 Simulation of discrete data

This function performs simulation of the output from the discrete model

$$f\left(y_t|u_t, y_{t-1}, \Theta\right) = \Theta_{y_t|u_t,y_{t-1}}$$

We call the case simulation of a controlled coin with memory. It is because it contains control variable u_t and remembers the last output y_{t-1}. The model is given by the matrix of conditional probabilities for individual $y_t, u_t, y_{t-1} \in \{1, 2\}$ see Paragraph 4.1.

```
function y=dsim(u,y1,th)
  // y=dsim(u,y1,th)    Simulation of a disctrete system
  // y   new output
  // u   input
  // y1  old nput

  i=psi2row([u y1]); // index of conditional regressor
  yy=rand(1,1,'unif')<th(i,1);   // probability of conditional regressor
  y=2-yy;            // output (with values 1, 2)
endfunction
```

16.2 Kalman filter

This procedure performs the algorithm of Kalman filtering. It was discussed in Paragraph 15.12

```
function [xt,Rx,yp]=Kalman(xt,yt,ut,M,N,F,A,B,G,Rw,Rv,Rx)
  // Kalman filter for state estimtion with the model
  //                              xt = M*xt + N*ut + F + w
  //                              yt = A*xt + B*ut + G + v
  // xt    state
  // Rx    state estimate covariance matrix
  // yp    output prediction
  // yt    output
  // ut    input
  // M,N,F state model parameters
  // A,B,G output model parameters
  // Rw    state model covariance
  // Rv    output model covariance

  nx=size(M,1);
  ny=size(A,1);
  if isempty(F), F=zeros(nx,1); end
  if isempty(G), G=zeros(ny,1); end

  xt=M*xt+N*ut+F;                      // time update of the state
  Rx=Rw+M*Rx*M';                       // time updt. of state covariance

  yp=A*xt+B*ut+G;                      // output prediction
  Ry=Rv+A*Rx*A';                       // noise covariance update
  Rx=Rx-Rx*A'*inv(Ry)*A*Rx;            // state est. coariance update
  ey=yt-yp;                            // prediction error
  KG=Rx*A'*inv(Rv);                    // Kalman gain
  xt=xt+KG*ey;                         // data update of the state
endfunction
```

16.3 Coding of discrete variables

This procedure performs coding of several discrete variables into a single one with more values. The resulting code is equal to the order of the row of matrix, where individual combinations of the values of the entering variables are listed. The combinations are generated so that the index of the rightmost variable changes most rapidly. An example for three variables $a \in \{1,2\}, b \in \{1,2,3\}$ and $c \in \{1,2\}$ is here

row	a	b	c
1	1	1	1
2	1	1	2
3	1	2	1
4	1	2	2
5	1	3	1
6	1	3	2
7	2	1	1
etc. for $a = 2$			

The values of the variables must start by 1.

```
function i=psi2row(x,b)
  // i=psi2row(x,b)  i is row number of a model table with
  //                 the regression vector x with the base b;
  //                 elements of x(i) are 1,2,...,nb(i)
  // it is based on the relation
  //     i=b(n-1)b(n-2)...b(1)(x(n)-1)+...+b(1)(x(2)-1)+x(1)

  n=length(x);
  if argn(2)<2, b=2*ones(1,n); end
  bb=b(2:n);
  bb=bb(:)';
  b=[bb 1];
  i=0;
  for j=1:n
    i=(i+x(j)-1)*b(j);
  end
  i=i+1;
endfunction
```