Uppsala University
Information Technology
Dept. of Systems and Control
Torsten Söderström 1992
Revised JS
Revised EKL 0101

System Identification

Computer exercise 4

Recursive Identification and some Practical Aspects

Preparation exercises:
1. Study Chapters 9, 10 and 12 of System Identification.
   Also, read the instruction carefully and browse the MATLAB code.
2. Solve preparation exercise 1.

Name		Assistant's comments
Program	Year of reg.	
Date		
Passed prep. ex.	Sign	
Passed comp. ex.	Sign	

1 Goals

In this computer laboratory we will study some features of recursive identification, as well as some practical aspects of system identification, including:

- System identification as a way of model approximation, when the model structure is not rich enough to describe the true dynamics.
- Estimation of physical parameters.

2 Recursive Identification

In recursive identification methods, measured input-output data are processed recursively (sequentially) as they become available, *i.e.*, the model is based on observations up to the current time. Recursive identification, also referred to as on-line or adaptive identification, is used in various areas, such as: adaptive systems, fault detection and parameter-tracking. Most off-line (batch) algorithms can be converted (exactly or approximately) into a recursive counterpart in a straightforward manner. Moreover, in many cases there is a need to modify the algorithms so that time-varying dynamics can be tracked. Two approaches for such modifications are:

- Change the loss function to be minimized. For instance, for the least squares method we can modify the loss function according to

$$V_t(\theta) = \sum_{s=1}^{t} \lambda^{t-s}\varepsilon^2(s) \tag{2.1}$$

  where λ is known as a forgetting factor. This means, as an example, that measurements that are older than $T_0 = 1/(1-\lambda)$ are included in the criterion with a weight that is $\approx 36\%$ of that of the most recent measurement (T_0 is called the memory time constant).

- Model the parameter variations as a state space model (*e.g.* a random walk), and apply Kalman filtering techniques.

The first approach will be illustrated in this lab, while the second technique is covered in one of the homework assignments.

2.1 General Comparison

The following system is to be simulated:

$$(1 - 0.7q^{-1})y(t) = 0.8q^{-1}u(t) + (1 - 0.7q^{-1})e(t)$$

where $u(t)$ and $e(t)$ are uncorrelated white noise sequences with zero mean and unit variance. Identify the system using the following estimation methods: recursive least squares (RLS), recursive instrumental variable (RIV), recursive pseudo linear regression (RPLR) and recursive prediction error methods (RPEM). For RLS and RIV the model structure is

$$y(t) + ay(t-1) = bu(t-1) + \varepsilon(t)$$
$$\theta = (a \;\; b)^T$$

For RPLR and RPEM the model structure is

$$y(t) + ay(t-1) = bu(t-1) + e(t) + ce(t-1)$$

$$\theta = (a \ b \ c)^T$$

The task can be carried out using the MATLAB function `lab4a`. For a print-out see Appendix A.

What can you say about the performance of the methods? Especially, give comments about consistency and convergence for the different methods.

Answer:

2.2 Effect of the initial values

We will here study how the choice of the initial P matrix influences the estimate. The following system is to be simulated

$$y(t) - 0.9y(t-1) = 0.5u(t-1) + e(t) \tag{2.7}$$

where $u(t)$ is a white binary sequence uncorrelated with the white noise sequence $e(t)$, which has zero mean and variance 1. Identify the system using RLS and a first-order model

$$y(t) + ay(t-1) = bu(t-1) + \varepsilon(t) \tag{2.8}$$

The P matrix is initialized by

$$P = \rho \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.9}$$

Run the file `lab4b` (a print-out is given in the appendix). How does ρ influence the result?

Answer:

2.3 Effect of the forgetting factor

Next, we will study how the forgetting factor affects the estimate. In particular we will study the problem of tracking a time varying system. Consider a first order ARX system which makes an abrupt change at time $t = 100$.

$$y(t) - 0.8y(t-1) = b_0 u(t) + e(t) \qquad b_0 = \begin{cases} 1.5 & t \leq 100 \\ 0.5 & t > 100 \end{cases} \tag{2.10}$$

Run the MATLAB function `lab4c` to identify the system using a RARX method with different forgetting factors. Describe the trade off which has to be made when choosing the forgetting factor. Study also if the estimated a parameter is negatively affected by a low forgetting factor.

Answer:

3 Model Approximation

In this task we will examine how system identification can be viewed as a form of model approximation, when the system dynamics is too complex to belong the model structure considered. To simplify the study we consider a noise–free situation. Consider the following system, which has two distinct resonances.

$$G_0(q^{-1}) = \frac{1.0q^{-2} - 1.3q^{-3} + 0.8q^{-4}}{(1 - 1.5q^{-1} + 0.9q^{-2})(1 + 0.9q^{-2})}$$

Simulate the system using the input $u(t)$ as white binary noise of zero mean and unit amplitude. Use the function `gendata2`, listed in the Appendix, to generate 100 data points. Next, we will estimate the system above as a second order ARMAX model (denoted by $G(q^{-1}, \theta)$), by means of a prediction error method using prefiltered data

$$u^F(t) = F(q^{-1})u(t) \tag{3.1}$$

$$y^F(t) = F(q^{-1})y(t) \tag{3.2}$$

The filtering has the effect that we give emphasis to certain frequency ranges, depending on the choice of the filter. In fact, one can show that the parameter vector θ is determined as the minimizing element of

$$V(\theta) = \int |F(e^{i\omega})|^2 |G_0(e^{i\omega}) - G(e^{i\omega}, \theta)|^2 \Phi_u(\omega)\, d\omega \tag{3.3}$$

where $\Phi_u(\omega)$ is the spectral density of the input signal. This means that $|F(e^{i\omega})|^2\Phi_u(\omega)$ weights in what frequency region the deviation $|G_0(e^{i\omega}) - G(e^{i\omega,\theta})|$ will be penalized. Hence, by adjusting the prefilter, the user can directly influence the model fit in different frequency regions. The following tasks is to be considered:

1. Identify the system for $F(q^{-1}) = 1$. This task can be carried out by running the MATLAB function `lab4d`. The result is evaluated in the frequency domain by drawing Bode plots of the model, the filter and the true dynamics.
2. Let $F(q^{-1})$ be a sharp bandpass filter around one of the resonance frequencies of the system. Use a 5'th order Butterworth filter. The MATLAB function `filtdes` can be useful when designing the filter. Once the filter is designed (numerator and denominator stored in *nn* and *dd*, respectively. This is automatically done by `filtdes`) run `lab4d` to perform the estimation procedure.
3. Repeat the previous task but for a filter with emphasize on the other resonance frequency.
4. Repeat the previous task but let F be a low pass filter.

Summarize your findings below

Answer:

4 Extra: Estimation of Physical parameters

Note: Do this exercise if you have the time. It might be useful in your future career to know how to estimate physical parameters.

In this section, we shall now see how system identification can be used to estimate physical parameters. As an example we will consider a DC motor. Its (continuous-time) transfer function from voltage to angular position, is given by

$$G(s) = \frac{K}{s(1+sT)}$$

where K and T are the parameters to be determined. By choosing the angular position and velocity as state variables, we can represent the motor in state space form as

$$\begin{aligned} \dot{x} &= \begin{pmatrix} 0 & 1 \\ 0 & -1/T \end{pmatrix} x + \begin{pmatrix} 0 \\ K/T \end{pmatrix} u \\ y &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x \end{aligned}$$

We are interested in estimating the parameters K and T from discrete-time measurements of the velocity and the position. This is done is the MATLAB demo number 6. Run the demo, and see how this quite complex problem can be solved by using MATLAB and

theory covered in the SI course. To run the demo, just type `iddemo` at the MATLAB prompt and choose number 6, *Building structured and user-defined models*. Focus on the first part of the demo (1 Free Parameters). The second part (2 Coupled parameters) is, however, also of interest.

A MATLAB-Code

```
%** General comparison.
%**********************
clear;
N=input('Number of samples N (RETURN gives N=250): ');
if (isempty(N)), N=250;end
u=randn(N,1); e=randn(N,1);clf;
a=-0.7; b=0.8; A=[1 a]; B=[0 b];
%av=a*ones(N,1);bv=b*ones(N,1);
y=filter(B,A,u)+e; z=[y u];

%** RLS.
%*******
nn=[1 1 1];
thm=rarx(z,nn,'ff',1);
a_vec=-0.7*ones(N,1); b_vec=0.8*ones(N,1);
subplot(2,2,1);
plot([1:N],thm(:,1),'y',[1:N],thm(:,2),'m',[1:N],a_vec,'r',[1:N],b_vec,'r');
axis([0 N -2 2]); grid; title('RLS');

%** RIV.
%*******
P=10000*eye(2);
th=[0; 0];
thmat=th;
for i=3:N
        phi=[-y(i-1); u(i-1,1)];
        Z=[u(i-1,1); u(i-2,1)];
        P=P-P*Z*phi'*P/(1+phi'*P*Z);
        ee=y(i)-phi'*th;
        th=th+P*Z*ee;
        thmat=[thmat th];
end
subplot(2,2,2);
plot([1:N-1],thmat(1,:),'y',[1:N-1],thmat(2,:),'m',[1:N],a_vec,'r',[1:N],b_vec,'r'
axis([0 N -2 2]); grid; title('RIV');

%** RPLR.
%********
nn1=[1 1 1 0 0 1];
thm1=rplr(z,nn1,'ff',1);
subplot(2,2,3);
plot([1:N],thm1(:,1),'y',[1:N],thm1(:,2),'m',[1:N],thm1(:,3),'c',[1:N],a_vec,'r',[
axis([0 N -2 2]); grid; title('RPLR');
```

```
%** RPEM.
%********
thm2=rpem(z,nn1,'ff',1);
subplot(2,2,4);
plot([1:N],thm2(:,1),'y',[1:N],thm2(:,2),'m',[1:N],thm2(:,3),'c',[1:N],a_vec,'r',[
axis([0 N -2 2]); grid; title('RPEM');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%** lab4b.m: Effect of the initial values.
%************************************
clear; clf
N=250; e=randn(N,1);

%** Generate input
%************************
u=sign(randn(N,1));

%** Simulate system.
%*******************
a=-0.9; b=0.5; A=[1 a]; B=[0 b];
y=filter(B,A,u)+filter(1,A,e);

%** Using RLS with different initial values of rho.
nn=[1 1 1];
z=[y u]; th0=[0 0];
a=-0.9*ones(N,1); b=0.5*ones(N,1);

%** rho=0.01.
%************
P01=0.01*eye(2);
[thm1,yhat,P]=rarx(z,nn,'ff',1,th0,P01);
subplot(2,2,1);
plot([1:N],thm1,[1:N],a,'k',[1:N],b,'k');
axis([0 N -1 1]); title('rho=0.01');

%** rho=0.1.
%***********
P02=0.1*eye(2);
[thm2,yhat,P]=rarx(z,nn,'ff',1,th0,P02);
subplot(2,2,2);
plot([1:N],thm2,[1:N],a,'k',[1:N],b,'k');
axis([0 N -1 1]); title('rho=0.1');

%** rho=1.
%*********
P03=1*eye(2);
[thm3,yhat,P]=rarx(z,nn,'ff',1,th0,P03);
subplot(2,2,3);
plot([1:N],thm3,[1:N],a,'k',[1:N],b,'k');
axis([0 N -1 1]); title('rho=1');

%** rho=10.
```

```
%**********
P04=10*eye(2);
[thm4,yhat,P]=rarx(z,nn,'ff',1,th0,P04);
subplot(2,2,4);
plot([1:N],thm4,[1:N],a,'k',[1:N],b,'k');
axis([0 N -1 1]); title('rho=10');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%** lab4c: effect of the forgetting factor.
%**************************************
clear;
N=200;
u=sign(randn(N,1));e=randn(N,1);
B1=[0 1.5];A=[1 -.8];
[y1,xx]=filter(B1,A,u(1:N/2));
B2=[0 .5];
y=[y1;filter(B2,A,u(N/2+1:N),xx)]+filter(1,A,e);
z1=[y u];
%idplot(z1);

nn2=[1 1 1];
th0=[0 0]; P0=1*eye(2);
c_vec=[1.5*ones(N/2,1); .5*ones(N/2,1)]; d_vec=-.8*ones(N,1);
lam=input('Give lambda [l1 l2 l3] (RETURN = default): ');
if(isempty(lam)), lam=[1 .95 .85];
elseif (length(lam)~=3)
  error(' Wrong input');
end

%** lambda=1.
%************
[thm3,yhat,P]=rarx(z1,[1 1 1],'ff',lam(1),th0,P0);
subplot(3,1,1);
plot([1:N],thm3,[1:N],c_vec,'r',[1:N],d_vec,'r');
text(210,1.2,'b')
text(210,-1.3,'a')
axis([0 N -2 2]); grid; title(['lambda=',num2str(lam(1))]);

%** lambda=0.97.
%***************
[thm4,yhat,P]=rarx(z1,nn2,'ff',lam(2),th0,P0);
subplot(3,1,2);
plot([1:N],thm4,[1:N],c_vec,'r',[1:N],d_vec,'r');
text(210,1.2,'b')
text(210,-1.3,'a')
axis([0 N -2 2]); grid; title(['lambda=',num2str(lam(2))]);

%** lambda=0.90.
%***************
[thm5,yhat,P]=rarx(z1,nn2,'ff',lam(3),th0,P0);
subplot(3,1,3);
plot([1:N],thm5,[1:N],c_vec,'r',[1:N],d_vec,'r');
```

```
text(210,1.2,'b')
text(210,-1.3,'a')
axis([0 N -2 2]); grid; title(['lambda=',num2str(lam(3))]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%** gendata2
%****************************************
num=[0 0 1 -1.3 .8];
den=conv([1 -1.5 .9],[1 0 .9]);
[msys,psys,w]=dbode(num,den,1);
u=sign(randn(100,1));
y=filter(num,den,u);
z=[y u];
nn=1;dd=1;

%file lab4d.m
%Computer Laboratory 3
%System Identification
%TS last rev 950405

% Model approximation -- frequency domain effects
clf

echo on
% In order to run the file, the following variables must exist:
%
% The input (u) and output (y) and the magnitude
% (msys) and phase (psys) of
% the system transfer function for the frequencies (w).
% All given by the file gendata3.
%
% Filternumerator (nn) and filerdenominator (dd), which can
% be provided by the function filtdes.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
%Estimation based on filtered data
filtnum=nn;%input('Give filtnumerator with [] ')
filtden=dd;%input('Give filtdenominator with [] ')

yf=filter(filtnum,filtden,y);
uf=filter(filtnum,filtden,u);
that1=armax([yf uf],[2 2 2 1]);

[m1,p1]=bode(that1,w);m1=squeeze(m1);p1=squeeze(p1);
[mf,pf]=dbode(filtnum,filtden,1,w);

format short
subplot(121)
loglog(w,m1,w,msys,'--',w,mf,':')
title('Frequency functions');
xlabel('Angular frequency'); ylabel('Amplitude');
```

```
mmax=max(max(m1),max(max(msys),max(mf)))+10;
axis([0.1 10 0.1 mmax])
legend('Est','Sys','Filt',0)

subplot(122);
semilogx(w,p1,w,psys,'--'); hold off;
title('Frequency functions');
ylabel('Phase'),xlabel('Angular frequency')
```