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

System Identification

Computer exercise 3

Different Identification Methods and Model Validation

Preparation exercises:

1. Study Chapters 7, 8 and 11 of System Identification.
   Also, read this instruction carefully, and browse the provided 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 investigate and compare different methods for system identification (the least squares method, the prediction error method and the instrumental variable method). We will also examine some various approaches for model validation.

2 Theory

2.1 Identification Methods

The three identification methods, the least squares method, the prediction error method and the instrumental variable method, are thoroughly described in the course book system identification. Nevertheless, in this section we will shortly recapitulate some of the important features of these methods.

1. **The least squares method.** The LS method is applicable to models of the form

$$A(q^{-1})y(t) = B(q^{-1})u(t) + \varepsilon(t) \tag{2.1}$$

where $\varepsilon(t)$ denotes an equation error, or equivalently expressed as the linear regression model

$$y(t) = \varphi^T(t)\theta + \varepsilon(t) \tag{2.2}$$

where

$$\varphi^T(t) = [-y(t-1)\cdots - y(t-n_a) \quad u(t-1)\cdots u(t-n_b)] \tag{2.3}$$

$$\theta = [a_1 \cdots a_{n_a} \quad b_1 \cdots b_{n_b}] \tag{2.4}$$

The LS estimate, *i.e.*, the estimate that minimizes the sum of squared equation errors, is readily obtained as

$$\hat{\theta} = \big[\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\big]^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)y(t) \tag{2.5}$$

Assuming that the true system dynamic is described as

$$A_0(q^{-1})y(t) = B_0(q^{-1})u(t) + v(t) \tag{2.6}$$

where $v(t)$ is a stationary stochastic process that is independent of the input signal, it turns out that this estimate is consistent if

$$E\{\varphi(t)\varphi^T(t)\} \quad \text{is nonsingular} \tag{2.7}$$

$$E\{\varphi(t)v(t)\} = 0 \tag{2.8}$$

where the latter condition is quite restrictive. In fact, we must require $v(t)$ to be an uncorrelated disturbance (*e.g.* white noise) for the condition to hold.

2. **The instrumental variable method.** This method can be seen as a generalization of the least squares method. The main idea is to modify the LS estimate, so it is consistent for an arbitrary disturbance. We are still considering ARX structures (2.1), but we modify the LS estimate (2.5), into

$$\hat{\theta} = \big[\frac{1}{N}\sum_{t=1}^{N}z(t)\varphi^T(t)\big]^{-1}\frac{1}{N}\sum_{t=1}^{N}z(t)y(t) \tag{2.9}$$

where $z(t)$ is a vector of instrumental variables that can be chosen in different ways subject to certain conditions guaranteeing consistency. We will assume that the data for the true system (2.6) is stationary. If we chose $z(t)$ to be uncorrelated with the disturbance $v(t)$, and $R \triangleq E\{z(t)\varphi^T(t)\}$ has full rank, then it holds that the estimate (2.9) is consistent. The rank condition is satisfied for almost any, but not all, systems. A common way of chasing the instruments is as follows

$$z(t) = [-\eta(t-1)\cdots\eta(t-n_a) \quad u(t-1)\cdots u(t-n_b)]^T \tag{2.10}$$

where the signal $\eta(t)$ is obtained by filtering the input

$$C(q^{-1})\eta(t) = D(q^{-1})u(t) \tag{2.11}$$

The polynomials C and D can be chosen in many ways. One special choice is to let C and D be *a priori* estimates of A and B, respectively (obtained, for instance, by using the LS methods; this is implemented in the MATLAB file `iv4`). Another special case is $C(q^{-1}) = 1$ and $D(q^{-1}) = -q^{-n_b}$.

3. **Prediction error methods** This methods can be applied to the general linear parametric model

$$y(t) = G(q^{-1},\theta)u(t) + H(q^{-1},\theta)e(t) \tag{2.12}$$

$$E\{e(t)e^T(s)\} = \Lambda(\theta)\delta_{t,s} \tag{2.13}$$

The main idea is to determine the model parameter vector θ, so that the prediction error

$$\varepsilon(t,\theta) = y(t) - \hat{y}(t|t-1,\theta) \tag{2.14}$$

is small. Here $\hat{y}(t|t-1,\theta)$ denotes the mean-square optimal prediction of $y(t)$.

To define a prediction error method the user has to make the following choices: Chose a model structure, *i.e.*, parameterize G, H and Λ, as a function of θ. Chose a criterion function, *i.e.*, decide upon a scalar-valued function of all the prediction errors; this criterion is to be minimized with respect to θ, in order to find the desired estimate.

The prediction error methods can be shown to (under weak conditions) be consistent, and statistically efficient. However, the methods require the computation of the prediction errors, and they (often) rely on a numerical minimization, with the possibility of getting stuck in local minima.

2.2 Model Validation

The parameter estimation procedure picks out the "best" model within the chosen model structure. The crucial question then is whether, or not, this model is "good enough". This is the problem of model validation. In this computer laboratory we will consider the following model validation techniques:

1. **Residual analysis.** The "leftovers" from the modeling process, *i.e.*, the part of the data that the model could not reproduce are the residuals

$$\varepsilon(t) = y(t) - \hat{y}(t|\hat{\theta}_N) \tag{2.15}$$

It is clear that these bear information about the quality of the model. Ideally, the residuals should be an uncorrelated sequence. However, to decide whether they are correlated, or not, can be difficult by just looking at them. Moreover, one can gain much more insight by combining the residuals with other signals. Therefore, it is

more common to do some testing based on the sample covariance functions $\hat{r}_\varepsilon$ and $\hat{r}_{\varepsilon u}$, where $u(t)$ is the input signal. In the ideal case, these covariance functions should be zero (the obvious exception is, of course, the covariance element $r_\varepsilon(0)$). If there are correlation between past inputs and the residuals, then there is a part of $y(t)$ that originates from the past input and that has not been properly picked up by the model, *e.g.*, a clear peak at lag k indicates that the effect from input $u(t-k)$ on $y(t)$ is not properly described by the model. A rule of thumb is that a slowly varying cross correlation function outside the confidence region is an indication of too few poles, while sharper peaks indicates too few zeros or wrong delay. Similarly, a large $\hat{r}_\varepsilon$ means that $y(t)$ could have been better predicted, which again is a sign of deficiency of the model.

2. **Fit between simulated and measured output.** Compare the "measured" output $y(t)$ and the model output $y_m(t)$. For a good model they should, of course, resemble each other (To what degree?). This comparison should be done using both the estimation data set and a validation data set. Use of the latter data set gives an indication if the model is capable of describing new data, *i.e.*, in some sense it gives information about the robustness of the model.

3. **Using various test criteria.** Check the numerical values of the loss function $V_N(\hat{\theta}_N)$, as well as the validation criteria AIC and FPE, as functions of the model order n. As your final value choose the value op n that minimizes these criteria. However, in practice one should use these tests with care . For example, the loss function is a monotonically decreasing function with respect to n. Hence, this test will generically indicate that a very "large" model order (usually too large) should be selected. In the AIC- and FPE tests some cure of this plausible overparameterization is provided by introducing a complexity term that increases with n; still, neither the AIC, nor the FPE estimates of the order are consistent. Nevertheless, instead of searching for the global minimum of the above mentioned functions it is usually a better idea to choose the n, for which the functions make a "large" drop, as your final model order.

4. **Pole–zero cancellations.** Compute and plot the poles and the zeros of the models, with confidence regions. If pole–zero cancellations (almost) occurs, it is a sign that the model order has been chosen unnecessarily high. In particular, if it turns out that the order of ARX models has to be increased to get a good fit, but that pole–zero cancellations are indicated, then the extra poles are just introduced to describe the noise. Then try another model structure *e.g.* ARMAX.

Preparation exercise 1. This exercise aims at illustrating the *parsimony principle*, which is a rather useful rule when determining an appropriate model order. This principle says that out of two or more competing models, which all explains the data well, the model with the smallest number of independent parameters should be chosen.
Consider the following stable AR(1) process

$$\mathcal{S}: \quad y(t) + a_0 y(t-1) = e(t) \tag{2.16}$$

where $e(t)$ is white noise with zero mean and variance λ^2. Also, consider the following two models of $\mathcal{S}$:

$$\mathcal{M}_1: \quad y(t) + a y(t-1) = \varepsilon(t) \tag{2.17}$$
$$\mathcal{M}_2: \quad y(t) + a_1 y(t-1) + a_2 y(t-2) = \varepsilon(t) \tag{2.18}$$

Let $\hat{a}$ and $\hat{a}_1$ denote the LS estimate of a in $\mathcal{M}_1$ and a_1 in $\mathcal{M}_2$. Determine the asymptotic estimation error variances for $\hat{a}$ and $\hat{a}_1$ ($\mathrm{var}(\hat{a})$ and $\mathrm{var}(\hat{a}_1)$), and show that $\mathrm{var}(\hat{a}) <$

$\mathrm{var}(\hat{a}_1)$. Hint: For a linear regression

$$y(t) = \varphi^T(t)\theta + \varepsilon(t) \tag{2.19}$$

it asymptotically holds that

$$\mathrm{cov}\,(\hat{\theta}) = \lambda^2 \big[E\,\varphi(t)\varphi^T(t)\big]^{-1} \tag{2.20}$$

Further, for the system $\mathcal{S}$ the covariance function $r(\tau) = E\,y(t+\tau)y(t)$ fulfills $r(\tau) = (-a_0)^\tau r_0 \lambda^2$ with $r_0 \triangleq 1/(1-a_0^2)$.

Answer:

3 Tasks

3.1 The System

The system that we will identify is given by

$$(1 - 1.5q^{-1} + 0.7q^{-2})y(t) = (1.0q^{-1} + 0.5q^{-2})u(t) + (1 - 1.0q^{-1} + 0.2q^{-2})e(t)$$

where the input signal is binary white noise ($u(t) = \pm 1$), independent of the white noise $e(t)$, which has zero mean and variance $\lambda^2 = 1$. Simulate the system using $N = 250$ data points. This can be done using the MATLAB function `gendata`. Actually the file generates two sets of data; one set used for estimation and one set intended for validation. For an adequate comparison between the different estimation schemes it is important that the same set of data is used all the time.

3.2 The Least Squares method

First we will try to fit an ARX model of order n to the simulated data using the least squares method. We will try different values of n (say $n = 1, \ldots, 6$). Validate the models using the different techniques 1-5, described in Section 2.2.
The task can be carried out by running the MATLAB code `lab3a`, described in Appendix A.

What are your findings?

Answer:

3.3 The Prediction Error Method.

Repeat the previous task but now use the prediction error method applied with an ARMAX model of order n. Use the same data set as before. Try some values of n, say $n = 1, \ldots, 6$. Apply the same validation procedures as before.
The MATLAB file `lab3b` solves the task.

What are your findings?

Answer:

3.4 The IV Method.

Repeat the previous task but now use an instrumental variable method applied with an ARX model of order n. Use the same data set as before. Try some different values of n, say $n = 1, \ldots, 6$. Apply the same validation procedures as before. However, in this case, the criteria V, AIC, FPE are not theoretically supported, hence, use these tests with caution. Replace otherwise the residuals in the validation tests by the equation errors. The MATLAB file `lab3c` solves the task. Notice that this function uses the MATLAB

macro `iv4`, shortly described in section 2.1, to estimate the parameters.

What are your findings?

Answer:

A MATLAB code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Gendata                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
np=250
a0=[1 -1.5 0.7], b0=[0 1 .5], c0=[1 -1.0 0.2],
s2=1, l2=1
u=sign(randn(2*np,1))*sqrt(s2); e=randn(2*np,1);

th=idpoly(a0,b0,c0,1,1,l2);
y=idsim([u e],th);
uv=u(np+1:end);ev=e(np+1:end);yv=y(np+1:end);
u=u(1:np);e=e(1:np);y=y(1:np);
ze=iddata(y,u);zv=iddata(yv,uv);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% file lab3a.m
% Use of LSM (ARX model)
%Computer Laboratory 3 in System Identification
%TS 920315, rev TS 950405, rev KB 000211, rev EKL 010110
%rev EKL 040525
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clf
format compact

nlag=10;resl=[]; plsm=[]; zlsm=[];
np=length(y);
nmax=6;
for n=1:nmax,
  that1=arx(ze,[n n 1]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Change the above command to use prediction error method %
% or instrumental variable method according to            %
%                                                          %
% that1=armax(ze,[n n n 1]);                               %
% that1=iv4(ze,[n n 1]);                                   %
% (or use that1=iv(ze,[n n 1],1,[zeros(1,n),1]);)          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 eval(['TH' num2str(n) '=that1;']); % to save that1 for n=1:nmax
 %present(that1);
 [comp1(n),fit1(n)]=compare(ze,that1);
 [comp2(n),fit2(n)]=compare(zv,that1);
 V(n)=that1.EstimationInfo.LossFcn;
 FPE(n)=that1.EstimationInfo.FPE;
 AIC(n)=log(V(n))*np+2*2*n;
 % Modify the AIC for ARMAX because we have n more free variables!
 % i.e. AIC(n)=log(V(n))*np+3*2*n;
```

```
 format short
end;


echo on
%b) Testing based on the covariance functions of the residuals
echo off
for n=1:nmax
 eval(['resid(TH' num2str(n) ',ze);']);
 subplot(211)
 txt=['autocorrelation of residual e, n=',num2str(n)];
 title(txt)
 subplot(212)
 txt=['crosscorrelation between e and u, n=',num2str(n)];
 title(txt)
 pause
end;

echo on
%c)  measured output and model output
%    for the estimation data set.
echo off
mm=min([100 np]);
for n=1:nmax
  subplot(320+n);
  plot(1:mm,[comp1{n}.y(1:mm),y(1:mm)]);
  title(['n = ',num2str(n),' Fit :',num2str(fit1(n))]);
  legend('y_m','y',0)
end;
pause
echo on
%   measured output and model output
%   for the validation data set.
echo off
for n=1:nmax
  subplot(320+n);
  plot(1:mm,[comp2{n}.y(1:mm) ,yv(1:mm)]);
  title(['n = ',num2str(n),' Fit :',num2str(fit2(n))]);
  legend('y_m','y',0)
end;
pause
echo on

%d) loss function, AIC and FPE
echo off
clf
subplot(311)
plot(1:nmax,V), title('loss function'), xlabel('n')
subplot(312)
plot(1:nmax,AIC), title('AIC'), xlabel('n')
subplot(313)
plot(1:nmax,FPE), title('FPE'), xlabel('n')
```

```
pause

clf
echo on
%e) pole-zero plot with confidens regions
%   corresponding to 3 standard deviations.
echo off
for n=1:nmax
  eval(['zpplot(TH' num2str(n) ',3);']);
  title(['Poles and zeros ARX, n = ',num2str(n)])
   pause
end
```