UPPSALA UNIVERSITET
AVDELNINGEN FÖR SYSTEMTEKNIK
BC 9903, Last rev September 1, 2014

Computer exercise 5 (compulsory) for the course: Empirical Modelling

Recursive identification

Preparation exercises:
1. Read notes from lecture "Recursive identification"

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

Contents

1 Goals

In this computer lab we will investigate and compare some methods for recursive identification.

Recursive identification is not a part of the basic project work but included in the course (both in one lecture and in this lab exercises). The reason is that methods for recursive identification is useful in many practical applications, in particular when the dynamics of the system is time varying. Also note that Task 3 "Stability monitoring in a nuclear power plant", will be one topic in the last guest lecture.

2 Task 1 - Introduction

In this first task the aim is to briefly illustrate how to use the System Identification Toolbox (SIT) for recursive parameter estimation. Note that the graphical user interface (ident) can not be used for recursive techniques.

The number of methods and algorithms used in recursive identification is huge. This course covers only some of the most important techniques. In the SIT, the recursive version of an estimation algorithm is obtained by putting an r in front of the function name. For example, `rarx` is the recursive variant of `arx` (estimating an ARX model).

- Type `help rarx` and study how the function is used. Note that both forgetting factor and Kalman filter interpretation can be used (as well as some other techniques).
- Start the demo by typing `iddemo`. Select the demonstration *Adaptive/recursive methods*, and go through the example[1] Study the last example more carefully (where it is shown how to proceed in a real on-line application),

3 Task 2 - User choices

3.1 Effect of the initial values

We will here study how the choice of the initial P matrix influences the estimate, see the lecture notes. First generate some data with the file `data1`. A print out of this m-file is given below.

```
%data1
%Generating data from an ARX system
%
NN=100;
u = sign(randn(NN,1)); e = 0.2*randn(NN,1);
B=[0 1];A=[1 -0.8];
y=filter(B,A,u)+filter(1,A,e);
z = [y u];
```

[1] The algorithm *plr* has not been covered in the course, it is an approximation of the general recursive PEM, see Example 9.3 in Söderström and Stoica for details.

Next the system will be estimated with the recursive least squares (RLS) method using a first-order model

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

The P matrix is initialized by

$$P = \rho \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2}$$

Run the file `pinit` (a print out is given below) and try different values of ρ. How does ρ influence the result? Is it best to choose a high or a low value on ρ?

Answer:

Change the file `pinit` so that the initial guess of θ, (denoted th0 in the file) is close to the true value $\theta_o = [-0.8 \;\; 1]$. This means that we a priori know the system quite well. Run the file `pinit` again and try different values of ρ. Is it now best to choose a high or a low value on ρ?

Answer:

```
%pinit
% Test of initial values P(0)
%
%1. Initial guess of theta:
th0=[0 0];
th_s1(1,:)=th0;  %just for the plotting
%
lam=1;
%%%% Choice of P(0):
rho=input('Give scale factor to initial P matrix: ')
P0=rho*eye(2);
%
% Initialization of the RLS:
[th,yh,P,phi] = rarx(z(1,:),[1 1 1],'ff',lam,th0',P0);
%
% The loop starts:
for k = 2:NN
```

```
      [th,yh,P,phi] = rarx(z(k,:),[1 1 1],'ff',lam,th',P,phi);
   th_s1(k,:)=th;
 end
 plot([th_s1 ones(NN,1)*[-0.8 1]])
```

3.2 Effect of the forgetting factor

We will here study how the forgetting factor affects the estimate. In particular we will study the problem to track a time varying system. The file `data2` simulates a first order ARX system which makes an abrupt change at time 100. The b parameter changes then from 1 to 0.5. Run the file `data2`.

```
%data2
%Generating data from a first order ARX system
% where the B parameter changes in the middle of the interval.
%
NN=200;
u = sign(randn(NN,1)); e = 0.2*randn(NN,1);
B1=[0 1];A=[1 -0.8];
[y1,xx]=filter(B1,A,u(1:NN/2,1));
%
% Now B is changes:
B2=[0 0.5];
[y2]=filter(B2,A,u(NN/2+1:NN,1),xx);
y=[y1;y2]+filter(1,A,e);
z = [y u];
idplot(z)
```

Run the file `forg` using different forgetting factors. Describe the trade off which has to be made when choosing the forgetting factor. Study also if the estimated a parameter (the blue line) is negatively affected by a low forgetting factor.

Answer:

```
% forg
% Test of different forgetting factors
%
NN=200;
th0=[0 0];
lam=input('Give forgetting factor: ')
rho=1000
P0=rho*eye(2);
[th,yh,P,phi] = rarx(z(1,:),[1 1 1],'ff',lam,th0',P0);
 for k = 2:NN
```

```
      [th,yh,P,phi] = rarx(z(k,:),[1 1 1],'ff',lam,th',P,phi);
    th_s(k,:)=th;
  end

    % Plot theta and true values:
 plot([th_s [ones(NN,1)*(-0.8)] [ones(NN/2,1)*(1); ones(NN/2,1)*(0.5)]] )
grid
```

3.3 The kalman filter approach - Optional task which are mainly for those who had taken reglerteknik II

The most logical approach for tracking parameter changes is to assume a certain model for how the true parameter vector θ_o is varying. A typical choice is a random walk model:

$$\theta_o(t) = \theta_o(t-1) + w(t)$$

where $w(t)$ is assumed to be white Gaussian noise with covariance matrix

$$Ew(t)w^T(t) = R_1$$

In most cases R_1 can be regarded as a user choice. Normally, R_1 is chosen as a diagonal matrix where the diagonal elements are used to reflect how much the corresponding parameters are varying.

Lets assume that we know that only the b-parameter is varying, see the previous subsection.
How should the knowledge be used for selecting the R_1 matrix?

Answer:

In the file `kalmf` a Kalman filter is used for estimating the parameters. Run the file and try different values on r22. See the file print out for how the covariance matrix R_1 is defined based on r22! Tune r22 so that it takes approximately 25 samples to find the new value of the b parameter. (That is, at time 125 the estimate should be reasonable close to the true value. Note that since the estimated b-parameter is affected by noise, it is not possible to give a strict specification).

Answer:

Is the estimated a parameter negatively affected by r22? Compare with the use of a forgetting factor!

Answer:

```
%kalmf
% Illustration of the Kalman filter approach
NN=200;
th0=[0 0];
r22=input('Give element 2,2 of covariance matrix R1: ')
R1=[0 0;0 r22];
rho=1000
P0=rho*eye(2);
[th,yh,P,phi] = rarx(z(1,:),[1 1 1],'kf',R1,th0',P0);
 for k = 2:NN
       [th,yh,P,phi] = rarx(z(k,:),[1 1 1],'kf',R1,th',P,phi);
    th_s(k,:)=th;
  end
  plot([th_s [ones(NN,1)*(-0.8)] [ones(NN/2,1)*(1); ones(NN/2,1)*(0.5)]] )
grid
```

4 Task 3 - Stability monitoring in a nuclear power plant

When a nuclear power plant is running with low core circulation flow and high power, there is an increased risk that the reactor become unstable. This gives an oscillating neutron flux which may lead to a shut down of the reactor and/or fuel damages.

In order to surveillance the reactor stability, many nuclear power plants are using stability monitors. The idea is as follows. A time series model is recursively estimated from the data. Typically an AR model is used:

$$A(q^{-1})y(t) = e(t)$$

The stability margin can be determined by using a measure how close the model is to instability. The standard approach is to use the *decay ratio* (DR), which is defined as the damping between two peaks in the impulse response. DR=1 then corresponds to instability. Here we will study a simplified method, namely to inspect the radius of the pole closest to the unit circle.

In this exercise we will study (real!) data from a Swedish nuclear power plant. Load the data by the command `load zdec`. Plot the data (`plot(zdec)`) What you see is the reactor power (prefiltered!), sampled at a sampling rate of 3 Hz.

This data is know to have an instability (self oscillating) around $t \approx 800$. Hence a good model (order) should pick up this instability and have a pole with radius close

to one (a pole on the unit circle) for some time interval around $t \approx 800$.

In the file `stabtest`, a 2'nd order AR model is recursively estimated and the largest pole radius is calculated and plotted. Run the file `stabtest` and study the result. As you see, the pole radius is well within the unit circle.
However look at the spectra[2] of the signal:

```
g=spa(zdec);
bodeplot(g);
```

Is a second order model feasible?

Answer:

Try different model orders in `stabtest` and inspect the results. You may also want to try different forgetting factors. Is it possible to detect any instability using a higher order model?

Answer:

```
%stabtest
% It is assumed that the data is stored with name zdec!
%
%Default forgetting factor:
lam=0.999;
%
% An AR model of order norder is to be recursively estimated
norder=2;
th=rarx(zdec,norder,'ff',lam);
%
% Look at the estimate
% plot(th)
```

[2]Note also that the spectra gives the mean spectra of the whole time series and possible periods with instability is smoothed out.

```
%
% Find the largest root of the estimated model
for j=1:length(th);rr(j,:)=max(abs(roots([1 th(j,:)])))';end
%
% Plot the pole radii (skip first 100 samples)
plot(rr(100:length(th)))
```

A Appendix - Implementation of RLS

In case you want to use the RLS algorithm it may be advantageous to implement the algorithm by yourselves.
Below is a typical matlab code for the RLS, in the case of a an AR model (for ARX models the updating of the `phi` must be modified).

```
%** RLS algorithm for AR-model
% Input needed:
% lam=forgetting factor
% n=model order
% P= initial P matrix
% y=data
%*****************
th=zeros(n,1);  %
thmat=th;       % Just a vector for storing all estimates
%
%The loop starts:
for k=(n+1):N
%
%First the phi vector is defined (must be changed for other models)
        for i=1:n
                phi(i,1)=-y(k-i,1);
        end
% Implementation of (9.12) in TS-PS:
        b=P*phi;
        a=lam+phi'*b;
        K=b/a;
        P=( P-K*b' )/lam;
        y_hat(k,1)=phi'*th;
        ee(k,1)=y(k,1)-y_hat(k,1);
        th=th+K*ee(k);
        thmat=[thmat th];
end
```