Course on Modelling and System Identication (MSI): Exercise 8 - Identification of a heating system: the Prediction Error Method

TASK 1

Write a Matlab function which given an input vector u and an output vector y identifies the coefficient vectors a and b of the ARX model

$$y(t) + a_1 * y(t-1) + a_2 * y(t-2) + ... + a_n a * y(t-na) = b_1 * u(t-1) + b_2 * u(t-2) + ... + b_n b * u(t-nb) + e(t)$$

using the prediction error method (PEM) explained in the lecture. The function should have the following syntax; **[a,b] = arx_pem(y, u, na, nb);**

```
function residuals = OE_f(theta,y,u,t,na,nb)
if ((length(y)~= length(u)) || (length(t)~= length(u)))
    error('The control data, measurement and/or time data have different size');
    return;
end
if ((na<0)||(nb<1))
    error('The size assigned to na and nb is not correct, please assign na > -1 and nb > 0');
    return;
end

% Extract parameters
a = theta(1:na);
b = theta(na+1:na+nb);
deltaT = t(2)-t(1);             % sample time
sysd = tf([0 b],[1 a], deltaT); % Define system
ysim = lsim(sysd,u,t);          % simulation system with input u
residuals = ysim-y;             % compute residuals
end
```

TASK 2

Load inside Matlab the data

```
clc;clear;close;
load ww.mat;       % Import data
whos               % List current variables
```

```
  Name          Size              Bytes  Class     Attributes

  t             1x10000           80000  double
  u         10000x1               80000  double
  y         10000x1               80000  double
```

TASK 3

Using the function created in Task 1 identify the model corresponding to data set. Set the order $na = nb = 2$ (the motivation for this is that we expect a second order model to describe the system well, because a state space model of the two rooms would have two states, the temperature in each room). Give the values a and b .

```
na=2;nb=2;
[a,b] = arx_pem(y,u,na,nb);
```

TASK 4

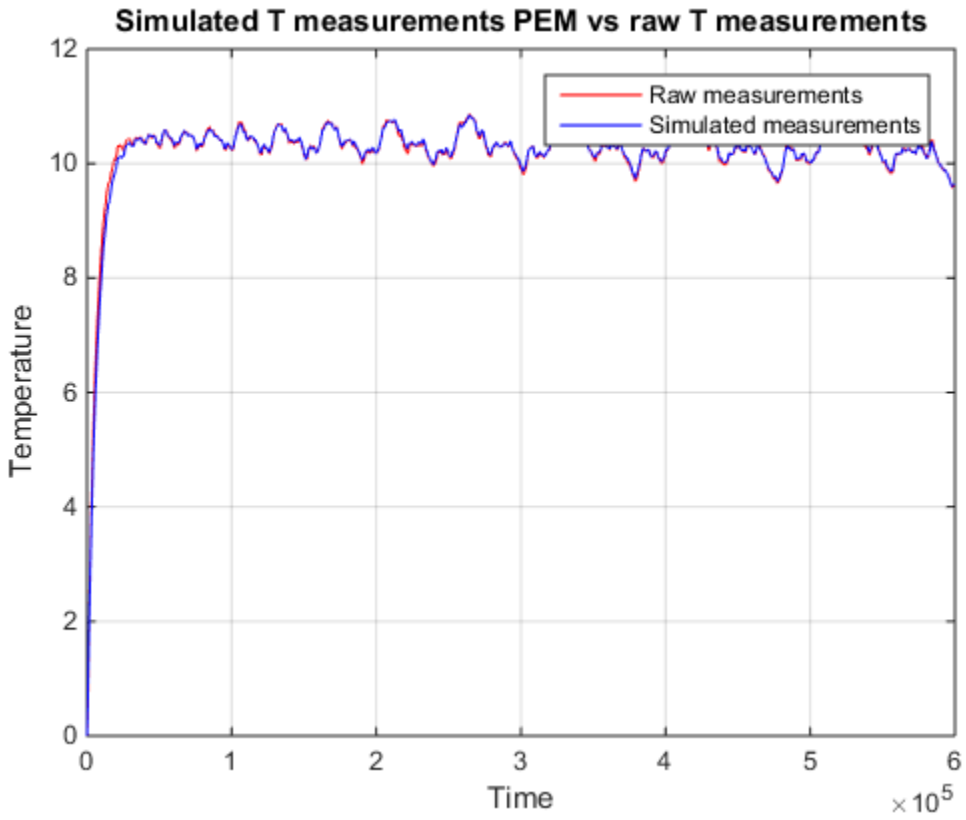
With the ARX coefficients calculated in the previous task define the corresponding discrete-time transfer function (use the command **tf** and find the sampling time using the vector t).

```
deltaT = t(2)-t(1);
sysd00 = tf([0 b'],[1 a'], deltaT);
```

TASK 5

Use the identified system and the command **lsim** to simulate the output trajectory *ysim* resulting from the given input trajectory *u* . Compare *ysim* with the actual measurements *y* in a plot.

```
ysim = lsim(sysd00,u,t);
figure(1);
plot(t,y,'r');grid on;hold on;
plot(t,ysim,'b');hold off;
title('Simulated T measurements PEM vs raw T measurements');
xlabel('Time');ylabel('Temperature');
legend ('Raw measurements','Simulated measurements');
```



TASK 6

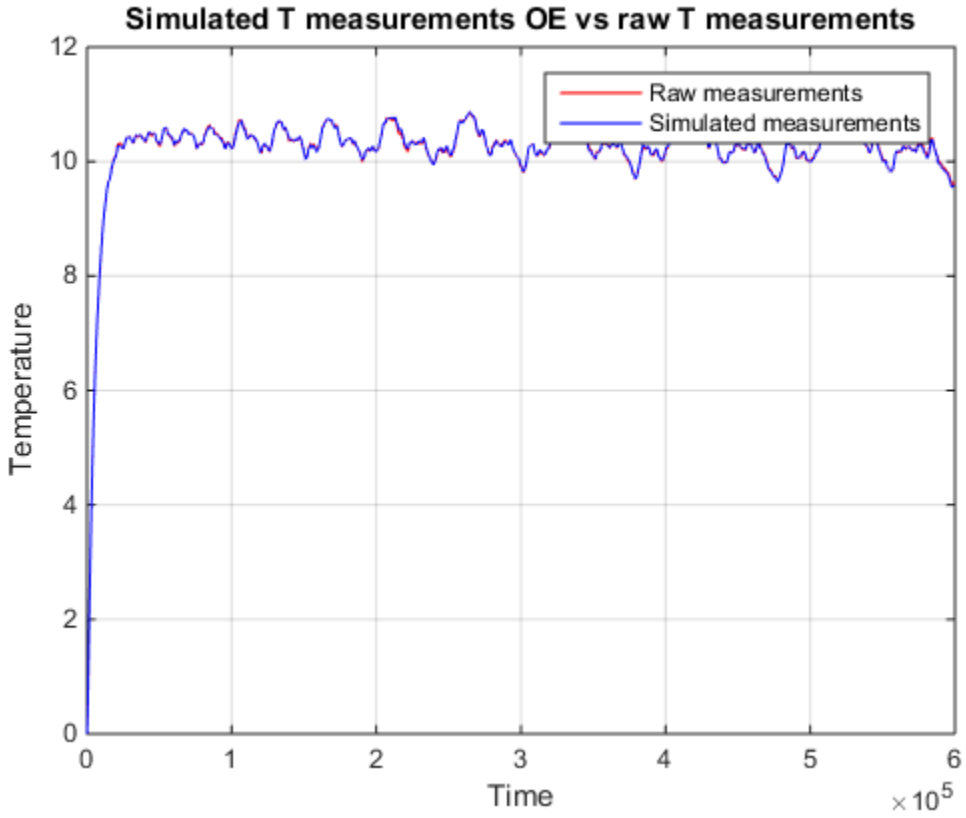
Instead of minimizing the equation errors (giving rise to linear least squares) one could also minimize the output errors, which gives rise to nonlinear least squares. For this aim, write a function that simulates the linear system for given parameters a, b and control trajectories (you can use **lsim** , as above), subtract the resulting output from the actual measurements, and minimize the sum of the squares of these residuals using **lsqnonlin**. For initialization, you can use the values obtained in Task 3.

```
theta0 = [a' b'];
theta00 = lsqnonlin(@(theta00) OE_f(theta00,y,u,t,na,nb),theta0);
a00 = theta00(1:na);
b00 = theta00(na+1:na+nb);
% Print value on console
formatSpec = '     %s* using the output error method is %f \n';
C = {'a1','a2','b1','b2';a00(1),a00(2),b00(1),b00(2)};
disp(sprintf(formatSpec,C{:}))
sysd00 = tf([0 b00],[1 a00], deltaT);
ysim00 = lsim(sysd00,u,t);
% Comparison ysim by means of output errors and y
figure(2);plot(t,y,'r');grid on;hold on;
plot(t,ysim00,'b');hold off;xlabel('Time');ylabel('Temperature');
title('Simulated T measurements OE vs raw T measurements');
legend ('Raw measurements','Simulated measurements');
```

```
Local minimum possible.

lsqnonlin stopped because the final change in the sum of squares relative to
its initial value is less than the default value of the function tolerance.
```

```
a1* using the output error method is -1.844875
a2* using the output error method is 0.846389
b1* using the output error method is 0.000003
b2* using the output error method is 0.000001
```

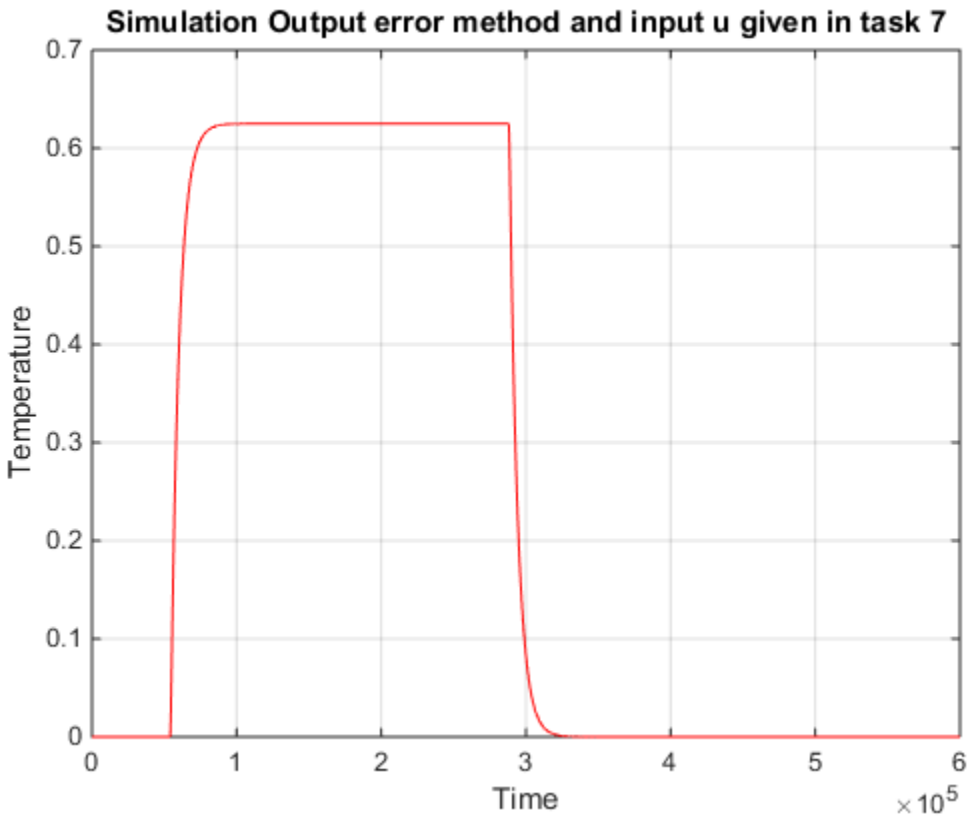


TASK 7

Simulate and plot the response of the identified system model to the following input trajectory:

```
index_15h = 15*3600 /deltaT+1;
index_80h = 80*3600 /deltaT+1;
u7 = zeros(length(t),1);
u7(index_15h:index_80h)=200;
ysimT7 = lsim(sysdOO,u7,t);
% Plot:Comparison ysim by means of output errors and y
figure(3);plot(t,ysimT7,'r');grid on;
```

```
title('Simulation Output error method and input u given in task 7')
xlabel('Time');ylabel('Temperature');
```



Published with MATLAB® R2014b