System Identification – ARX method

Costandin Marius

1 Theory

Let $t_0, t_1, \ldots, t_N$ be some time moments in arithmetic progression, $t_{i+1} - t_i = T_s$, with the ratio being the sampling time. We denote the output of the process by the letter y and the input of the process by the letter u. Furthermore, for simplicity of notation, let $y_k = y(t_k)$ and $u_k = u(t_k)$, that is the output of the process which is measured at the time moment t_k respectively the input which is given to the process at the time moment t_k.

Then let

$$\mathcal{Y}(z) = \sum_{k\in\mathbb{Z}} y_k \cdot z^{-k} \qquad \mathcal{U}(z) = \sum_{k\in\mathbb{Z}} u_k \cdot z^{-k} \tag{1}$$

be the so called $\mathcal{Z}$ transforms of the sequences $(y_k)_{k\in\mathbb{Z}}$ and $(u_k)_{k\in\mathbb{Z}}$ with $y_k = 0$ for an integer $k \leq 0$ and $u_k = 0$ for an integer $k < 0$.

The ARX method considers the following input-output relation:

$$\mathcal{Y}(z) = \frac{B(z)}{A(z)} \cdot z^{-n_p} \cdot \mathcal{U}(z) + \frac{1}{A(z)} \cdot \mathcal{E}(z) \tag{2}$$

where

$$\begin{aligned} B(z) &= b_1 \cdot z^{-1} + \ldots + b_{n_b} \cdot z^{-n_b} \\ A(z) &= 1 + a_1 \cdot z^{-1} + \ldots + a_{n_a} \cdot z^{-n_a} \end{aligned} \tag{3}$$

with $n_a, n_b, n_p \in \mathbb{N} \cup \{0\}$. Multiplying both sides by $A(z)$ and applying the inverse $\mathcal{Z}$ transform, one obtains:

$$y_k + a_1 \cdot y_{k-1} + \ldots + a_{n_a} \cdot y_{k-n_a} = b_1 \cdot u_{k-n_p-1} + \ldots + b_{n_b} \cdot u_{k-n_p-n_b} + e_k \tag{4}$$

hence it is obtained:

$$y_k = \begin{bmatrix} -y_{k-1} & \ldots & -y_{k-n_a} & u_{k-1-n_p} & \ldots & u_{k-n_b-n_p} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_{n_a} \\ b_1 \\ \vdots \\ b_{n_b} \end{bmatrix} + e_k \tag{5}$$

Letting k successively take the values $1, 2, \ldots, N$ in (5), one obtains:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} -y_0 & \ldots & -y_{1-n_a} & u_{-n_p} & \ldots & u_{1-n_b-n_p} \\ -y_1 & \ldots & -y_{2-n_a} & u_{1-n_p} & \ldots & 2-n_b-n_p \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -y_{N-1} & \ldots & -y_{N-n_a} & u_{N-1-n_p} & \ldots & u_{N-n_b-n_p} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_{n_a} \\ b_1 \\ \vdots \\ b_{n_b} \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{bmatrix} \tag{6}$$

Let $Y = \begin{bmatrix} y_1 & \ldots & y_N \end{bmatrix}$ be the vector of measurements, Φ be the resolvent matrix in the above equation (6), $\theta = \begin{bmatrix} a_1 & \ldots & a_{n_a} & b_1 & \ldots & b_{n_b} \end{bmatrix}^T$ and $E = \begin{bmatrix} e_1 & \ldots & e_N \end{bmatrix}^T$. Then, (6) is rewritten as follows:

$$Y = \Phi \cdot \theta + E \iff E = Y - \Phi \cdot \theta \tag{7}$$

Let us then consider the quantity $J = E^T \cdot E$. It is easy to see that once the inputs and outputs are given, that is y_ks and u_ks, for different values of θ, there will be different values of J. We search for θ such that $J = E^T \cdot E$ is the smallest. One can observe that

$$J = e_1^2 + \ldots + e_N^2 \tag{8}$$

For this, consider the operator:

$$\frac{\partial}{\partial \theta} = \begin{bmatrix} \frac{\partial}{\partial a_1} & \cdots & \frac{\partial}{\partial b_{n_b}} \end{bmatrix}^T \tag{9}$$

that is a column vector of partial derivatives with respect to the components of θ. We require

$$\frac{\partial}{\partial \theta} J = \begin{bmatrix} 0 & \ldots & 0 \end{bmatrix}^T = 0_{(n_a+n_b)\times 1} \iff \begin{cases} \frac{\partial J}{\partial a_1} = 0 \\ \vdots \\ \frac{\partial J}{\partial b_{n_b}} = 0 \end{cases} \tag{10}$$

that is

$$\frac{\partial}{\partial \theta} J = \frac{\partial}{\partial \theta}(E^T \cdot E) = \begin{bmatrix} \frac{\partial J}{\partial a_1} \\ \vdots \\ \frac{\partial J}{\partial b_{n_b}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E^T \cdot E}{\partial a_1} \\ \vdots \\ \frac{\partial E^T \cdot E}{\partial b_{n_b}} \end{bmatrix} = 2 \cdot \frac{\partial E^T}{\partial \theta} \cdot E \tag{11}$$

because, one obtains, for instance, for the first component:

$$\frac{\partial E^T \cdot E}{\partial a_1} = \frac{\partial E^T}{\partial a_1} \cdot E + E^T \cdot \frac{\partial E}{\partial a_1} = 2 \cdot \frac{\partial E^T}{\partial a_1} \cdot E \tag{12}$$

and even more:

$$\frac{\partial E^T}{\partial \theta} = \begin{bmatrix} \frac{\partial e_1}{\partial a_1} & \cdots & \frac{\partial e_N}{\partial a_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_1}{\partial b_{n_b}} & \cdots & \frac{\partial e_N}{\partial b_{n_b}} \end{bmatrix} = \begin{bmatrix} \frac{\partial E^T}{\partial a_1} \\ \vdots \\ \frac{\partial E^T}{\partial b_{n_b}} \end{bmatrix} \tag{13}$$

In equation (7), let $C_1, \ldots, C_{n_a+n_b}$ denote the columns of matrix Φ. Then

$$E^T = Y^T - \begin{bmatrix} a_1 & \ldots & a_{n_a} & b_1 & \ldots & b_{n_b} \end{bmatrix} \cdot \begin{bmatrix} C_1^T \\ \vdots \\ C_{n_a+n_b}^T \end{bmatrix} \tag{14}$$

therefore it is easy to see that

$$\frac{\partial E^T}{\partial a_1} = C_1^T \quad \ldots \quad \frac{\partial E^T}{\partial b_{n_b}} = C_{n_a+n_b}^T \quad \Rightarrow \quad \frac{\partial E^T}{\partial \theta} = \begin{bmatrix} C_1^T \\ \vdots \\ C_{n_a+n_b}^T \end{bmatrix} = \Phi^T \tag{15}$$

From (12) one obtains:

$$0_{n_a+n_b,1} = \frac{\partial E^T}{\partial \theta} \cdot E \iff \Phi^T \cdot (Y - \Phi \cdot \theta) = 0_{n_a+n_b,1} \tag{16}$$

therefore

$$\theta = \left(\Phi^T \cdot \Phi\right)^{-1} \cdot \Phi^T \cdot Y \tag{17}$$

Remark 1.1 *A further improvement can be made to (17) from the numeric point of view, by letting L_k denote the k'th line of matrix Φ. In this case $\Phi^T \cdot \Phi = \sum_{k=1}^N L_k^T \cdot L_k$ and $\Phi^T \cdot Y = \sum_{k=1}^N L_k^T \cdot y_k$ therefore (17) becomes*

$$\theta = \left(\sum_{k=1}^N L_k^T \cdot L_k\right)^{-1} \cdot \sum_{k=1}^N L_k^T \cdot y_k \tag{18}$$

2 Implementation

Assume the available data for identification is composed out of two arrays $Y = \begin{bmatrix} y_0 & y_1 & \dots & y_N \end{bmatrix}^T$ and $U = \begin{bmatrix} u_0 & u_1 & \dots & u_N \end{bmatrix}^T$. Next, one has to decide on the orders of the system, and choose n_a, n_b, n_p, see equations (2) and (3). Once this information is available, one will simply form the matrix Φ (using Y and U and equation (6)) and then apply the formula given in equation (17) or (18) to obtain θ. Next, the coefficients of the A, B polynomials are extracted from θ according to (5),(6).

In the following, a $MATLAB^{®}$ code is provided to implement just that:

```
% arx offline algorithm

% clear console, clear workspace, close figures
clc;
clear all
close all

% Load the data: data is in .mat format and in the working directory.
% The file is provided by the laboratory assistant.

data = load('lab6_2');
u = data.id.u;
y = data.id.y;

% plot the data, just for visualization
plot(u);
hold on
plot(y,'r');
title('initial data');
xlabel('time [s]')
legend('u','y');

% choose parameters: ... somehow ...
na = 1;
nb = 1;
np = 0;

% pad with zeros to account for negative indexes
y_ = [zeros(na+nb+np,1);y]';
u_ = [zeros(na+nb+np,1);u]';

% initilize the sums with zero matrices of appropriate size
S_1 = zeros(na+nb);
S_2 = zeros(na+nb,1);

% begin iterations
for i = na+nb+np+1:length(y_)
%   get a line in matrix Phi
    L = [-y_(i-1:-1:i-na) u_(i-1-np:-1:i-nb-np)];
%   update the sums
    S_1 = S_1 + L'*L;
```

```
    S_2 = S_2 + L'*y_(i);
end

% find theta
th = inv(S_1)*S_2;

% extract polynomials
A = [1 th(1:na)'];
B = [zeros(1,np), 0, th(na+1:end)'];

% validate:Ts is needed here
figure
sys_ = idpoly(A,B,1,1,1,0,data.val.ts);
compare(data.val,sys_);

% --------------------------------------
% ------- matlab solution
% --------------------------------------

figure
marx = arx(data.id,[na,nb,1]);
compare(marx,data.val);
```

Remark 2.1 *Sometimes, if the identification data does not meet the ARX assumpltions (about the error model), then higher orders are required for a good fit.*

3 Results

Upon running the above code the following figures are obtained:

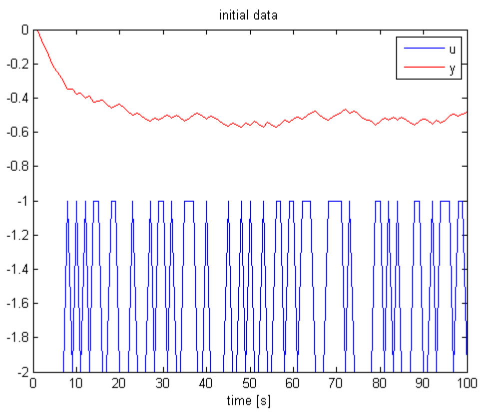

Figure 1: Initial Data

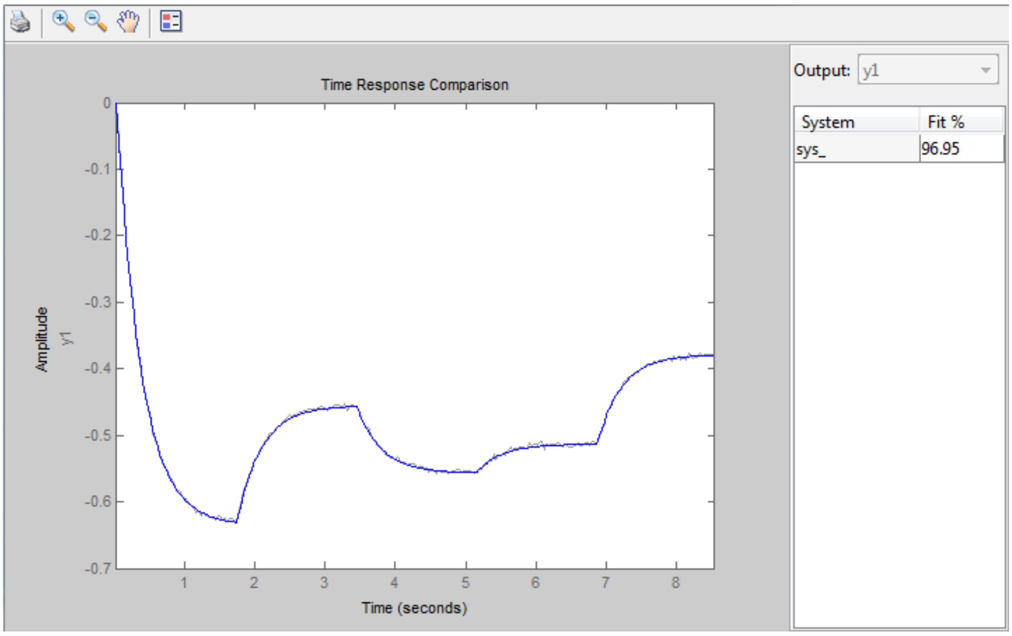

Figure 2: Compare Results: the proposed code

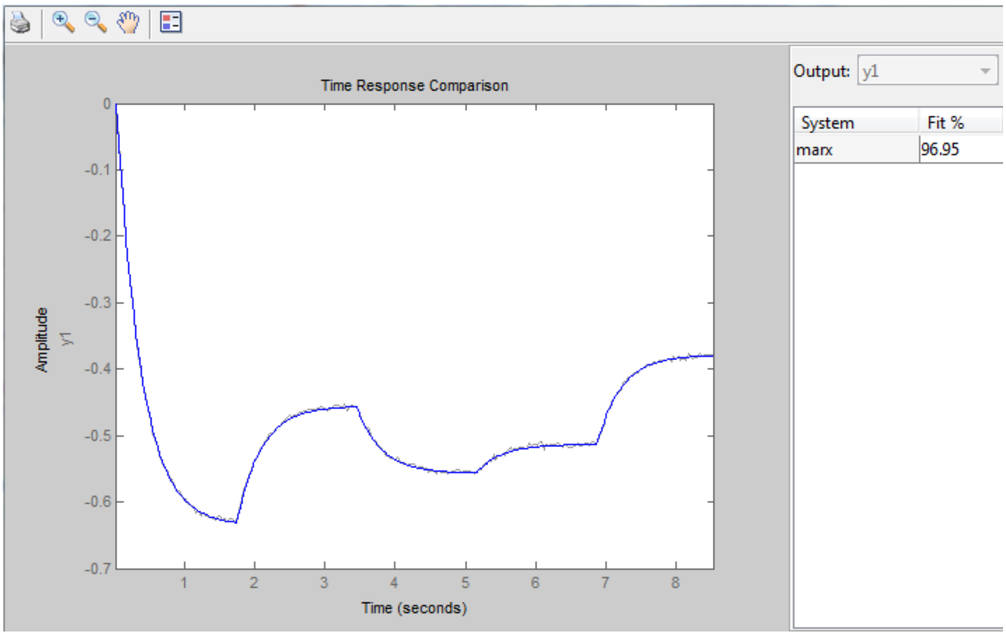

Figure 3: Compare Results: matlab solution