System Identification – IV 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 $k < 0$.

Althought in this laboratory we study the IV (Instrumental Variable) method, let us recall the ARX method:

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

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:

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

and then:

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

$$Y = \Phi \cdot \theta + E \tag{5}$$

where $Y = \begin{bmatrix} y_1 & \ldots & y_N \end{bmatrix}$ be the vector of measurements, Φ be the resolvent matrix in the above equation (5), $\theta = \begin{bmatrix} a_1 & \ldots & a_{n_a} & b_1 & \ldots & b_{n_b} \end{bmatrix}^T$ and $E = \begin{bmatrix} e_1 & \ldots & e_N \end{bmatrix}^T$.

Let us consider the quantity

$$J = e_1^2 + \ldots + e_N^2 == E^T \cdot E \tag{6}$$

We found, in ARX dedicated laboratory, that θ which minimizes J is:

$$\theta = \left(\Phi^T \cdot \Phi\right)^{-1} \cdot \Phi^T \cdot Y \tag{7}$$

Remark 1.1 *A further improvement can be made to (7) 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 (7) becomes*

$$\theta = \left(\sum_{k=1}^{N} L_k^T \cdot L_k\right)^{-1} \cdot \sum_{k=1}^{N} L_k^T \cdot y_k \tag{8}$$

Assume, we are given the measured data in form of vectors Y and U. Then it is easy to obtain Φ and produce the solution (7). But how accurate is it? Let $\theta^\star$ be the true parameter vector. Then we are interested in the difference:

$$\begin{aligned}\theta^\star - \theta &= \left(\sum_{k=1}^{N} L_k^T \cdot L_k\right)^{-1} \left(\sum_{k=1}^{N} L_k^T \cdot L_k\right)\theta^\star - \left(\sum_{k=1}^{N} L_k^T \cdot L_k\right)^{-1} \cdot \sum_{k=1}^{N} L_k^T \cdot y_k \\ &= \left(\sum_{k=1}^{N} L_k^T \cdot L_k\right)^{-1} \cdot \sum_{k=1}^{N} L_k^T \left(L_k \cdot \theta^\star - y_k\right) \end{aligned}\tag{9}$$

This difference is the null vector if $\sum_{k=1}^N L_k^T(L_k \cdot \theta^\star - y_k)$ is also the null vector. Therefore, in the following we take a closer look at this quantity:

$$Q = \frac{1}{N}\sum_{k=1}^{N} L_k^T(L_k \cdot \theta^\star - y_k) = \begin{bmatrix} \frac{1}{N}\sum_{k=1}^N y_{k-1}(L_k \cdot \theta^\star - y_k) \\ \vdots \\ \frac{1}{N}\sum_{k=1}^N y_{k-n_a}(L_k \cdot \theta^\star - y_k) \\ \frac{1}{N}\sum_{k=1}^N u_{k-1-n_p}(L_k \cdot \theta^\star - y_k) \\ \vdots \\ \frac{1}{N}\sum_{k=1}^N u_{k-n_b-n_p}(L_k \cdot \theta^\star - y_k) \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{k=1}^N y_{k-1}v_k \\ \vdots \\ \frac{1}{N}\sum_{k=1}^N y_{k-n_a}v_k \\ \frac{1}{N}\sum_{k=1}^N u_{k-1-n_p}v_k \\ \vdots \\ \frac{1}{N}\sum_{k=1}^N u_{k-n_b-n_p}v_k \end{bmatrix} \tag{10}$$

where $v_k = L_k \cdot \theta^\star - y_k$.

If f, g are stochastic processes jointly wide-sense stationary, then one approximates the cross-covariance of these two as:

$$R_{fg}(\tau) = \frac{1}{N}\sum_{k=1}^{N}\left(f(k) - \bar{f}\right) \cdot (g(k-\tau) - \bar{g}) = \ldots = \sum_{k=1}^{N} f(k) \cdot g(k-\tau) - \bar{f} \cdot \bar{g} \tag{11}$$

where $\bar{f} = \frac{1}{N} \cdot \sum_{k=1}^N f(k)$ and $\bar{g} = \frac{1}{N} \cdot \sum_{k=1}^N g(k)$ and we assumed that also $\bar{g} = \frac{1}{N}\sum_{k=1}^N g(k-\tau)$ However, if $\bar{v} = 0$ follows:

$$Q = \begin{bmatrix} R_{vy}(-1) \\ \vdots \\ R_{vy}(-n_a) \\ R_{vu}(-1-n_p) \\ \vdots \\ R_{vu}(-n_b - n_p) \end{bmatrix} \tag{12}$$

Because v_k (measurement and/or process noise) and the input u_k are independent (which means their covariance function is zero), for $Q = 0_{n_a+n_b}$ is sufficient to have the independence of v_k and $y_{k-\tau}$ for ($\tau > 0$). But for instance, take $\tau = 1$, hence y_{k-1} contains v_{k-1}. Therefore, we are essentially requiring v_k to be uncorrelated with its past values. This along with the fact that its mean in zero qualifies v_k as white noise. It is concluded that the equation (8) gives unbiased estimation of the parameter vector only in the case when prediction error is white noise, ARX models.

In order to remove this limitation, IV (instrumental variable) method proposes the following estimator:

$$\theta = \left(\sum_{k=1}^{N} Z_k^T \cdot L_k \right)^{-1} \cdot \sum_{k=1}^{N} Z_k^T \cdot y_k \tag{13}$$

In this situation one obtains:

$$\begin{aligned} \theta^\star - \theta = \ldots &= \left(\sum_{k=1}^{N} Z_k^T \cdot L_k \right)^{-1} \cdot \sum_{k=1}^{N} Z_k^T (L_k^T \cdot \theta^\star - y_k) \\ &= \left(\sum_{k=1}^{N} Z_k^T \cdot L_k \right)^{-1} \cdot \sum_{k=1}^{N} Z_k^T v_k \end{aligned} \tag{14}$$

Now, if the components of Z_k and v_k are independent, one can assure that the covariance function of the components of Z_k and v_k is zero, hence the estimation is unbiased.

In this laboratory we populate Z_k by modifying L_k to make it independent of v_k: the positions containing u_k are left unchanged, while the positions containing y_k are populated with some estimates of y_k which do not contain measurement of process noise. These estimates are made using some rough initial guess of the parameter vector (the one we are trying to find right now). This guess is done using the ARX algorithm. Of course it will not be perfect, but it will, hopefully, help us to obtain a better solution.

2 Implementation

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

```
% offline iv method

% arx offline algorithm

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

% load the data
data = load('lab9_4');
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 = data.n;
nb = data.n;
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);

% use arx algo to get an initial estimate of theta
m_arx = arx(data.id,[na,nb,np+1]);
theta_ = [m_arx.A(2:end) m_arx.B(np+2:end)]';

% init the simulated output
y_sim = zeros(1,na+nb+np);

% 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)];
%   get Z
    Z = [-y_sim(i-1:-1:i-na) u_(i-1-np:-1:i-nb-np)];
%   compute y_sim using theta_
    y_sim(i) = Z*theta_;
%   update the sums
    S_1 = S_1 + Z'*L;
    S_2 = S_2 + Z'*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
miv = iv(data.id,[nb,na,np+1],m_arx.A,m_arx.B);
compare(miv,data.val);
```

```
figure
compare(m_arx, data.val);
```

Remark 2.1 *This algorithm does not always converge!*

3 Results

Upon running the above code the following figures are obtained:

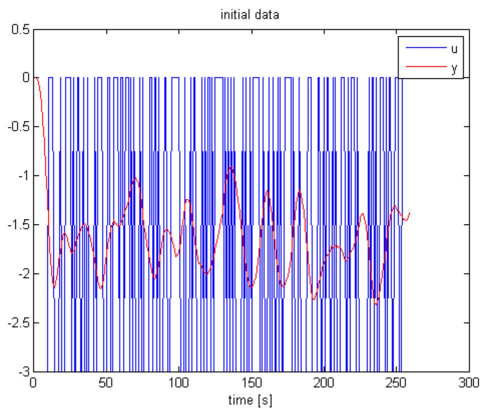
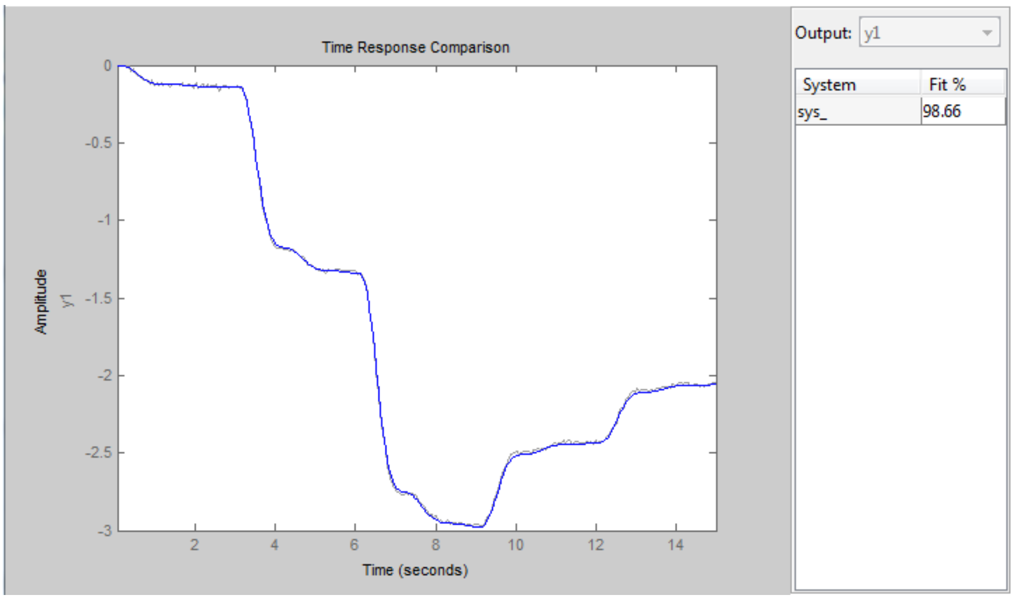
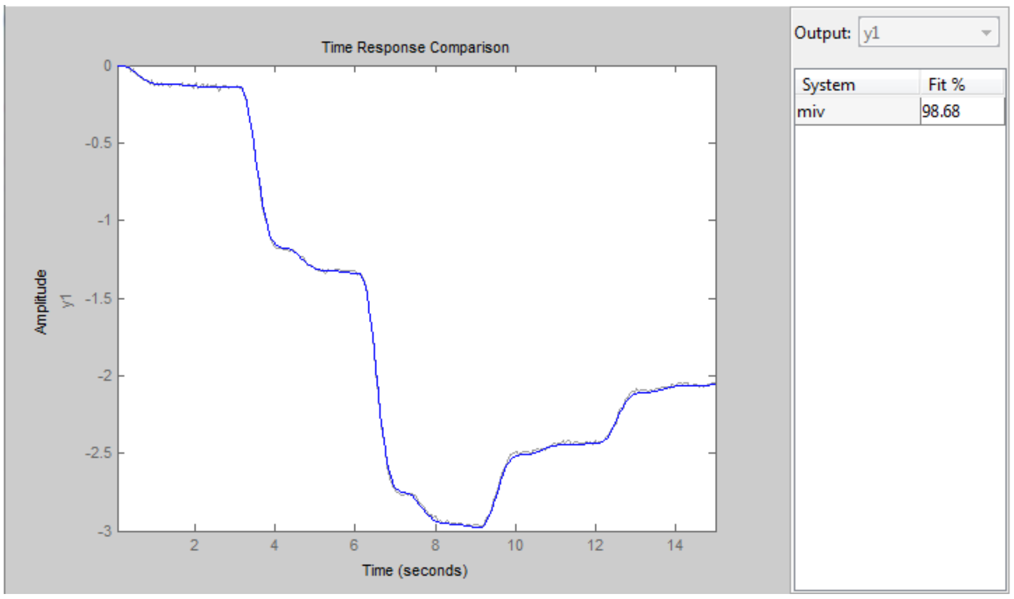


Figure 1: Initial Data



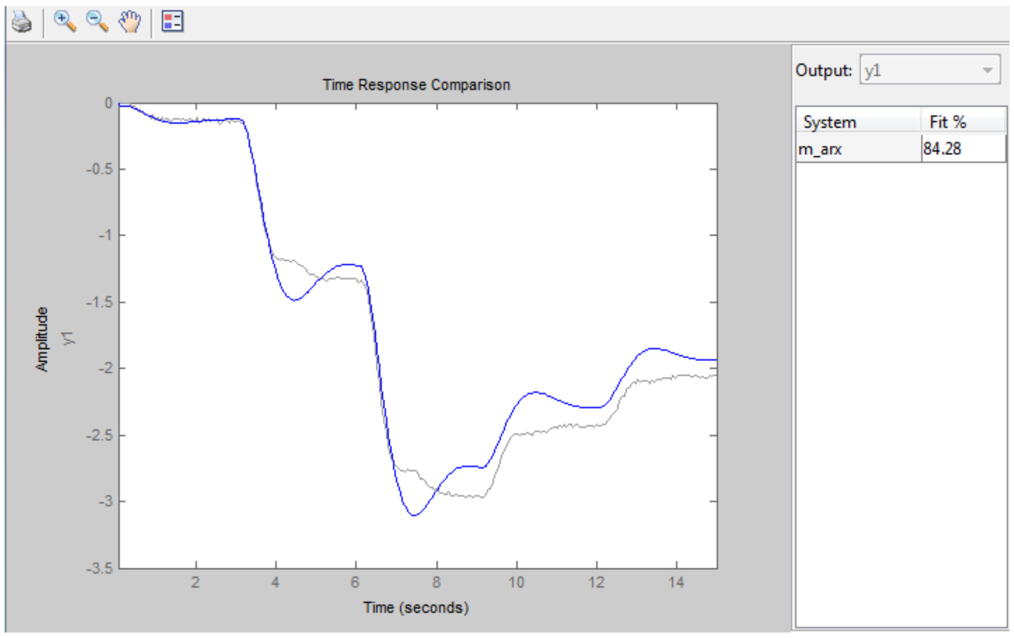
System	Fit %
sys_	98.66

Figure 2: Compare Results: the proposed code



System	Fit %
miv	98.68

Figure 3: Compare Results: matlab solution



System	Fit %
m_arx	84.28

Figure 4: Compare Results: Initial ARX identification