Implementation of nonlinear ARX

BEJENARIU Adina
FERENȚ Sarah
MARIN Călin
Group 30331/1
Data set 20/9

Contents

1. Introduction

- What is the purpose of this presentation?
    - develop a **black-box** model for the given data set, using a nonlinear ARX model

.

- Why is this considered a black box model?
    - it is obtained only from the experimental data given on the system, everything else about it is entirely unknown.

1. Introduction

- What is a Linear ARX model?
    - A **Linear ARX Model** is a representation of a dynamic system in discrete time. It uses a generalized notion of transfer functions to express the relationship between the input, ***u*(*k*)**, the output ***y*(*k*)**, and the noise **e(*k*)**.

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) - \ldots - a_{na} y(k-na)$$
$$b_1 u(k-1) + b_2 u(k-2) + \ldots + b_{nb} u(k-nb) + e(k)$$

- What is a Nonlinear ARX model?
    - A **Nonlinear ARX Model** extends the linear ARX models to the nonlinear case. It consists of the model regressors and an output function.

$$y(k) = p(y(k-1), \ldots, y(k-na), u(k-1), \ldots, u(k-nb)) + e(k)$$

2. Methodology

$$y = \phi \cdot \theta$$

Regressors' matrix (ϕ)

For $m = 2$ and $n_a = n_b = 2$

$$\begin{pmatrix}
1 & y(0) & 0 & u(0) & 0 & y(0)^2 & y(0)\,0 & y(0)\,u(0) & y(0)\,0 & 0^2 & 0\,u(0) & 0\,0 & u(0)^2 & u(0)\,0 & 0^2 \\
1 & y(1) & y(0) & u(1) & u(0) & y(1)^2 & y(1)\,y(0) & y(1)\,u(1) & y(1)\,u(0) & y(0)^2 & y(0)\,u(1) & y(0)\,u(0) & u(1)^2 & u(1)\,u(0) & u(0)^2 \\
1 & y(2) & y(1) & u(2) & u(1) & y(2)^2 & y(2)\,y(1) & y(2)\,u(2) & y(2)\,u(1) & y(1)^2 & y(1)\,u(2) & y(1)\,u(1) & u(2)^2 & u(2)\,u(1) & u(1)^2 \\
 & & & & & & & & & & & \dots & & & \\
1 & y(k-1) & y(k-2) & u(k-1) & u(k-2) & y(k-1)^2 & y(k-1)\,y(k-2) & y(k-1)\,u(k-1) & y(k-1)\,u(k-2) & y(k-2)^2 & y(k-2)\,u(k-1) & y(k-2)\,u(k-2) & u(k-1)^2 & u(k-1)\,u(k-2) & u(k-2)^2
\end{pmatrix}$$

Simplified notation

$$\begin{aligned} y(k-1) &\rightarrow x_1 \\ y(k-2) &\rightarrow x_2 \\ u(k-1) &\rightarrow x_3 \\ u(k-2) &\rightarrow x_4 \end{aligned}$$

In code, executed using the **`buildXMatrix`** function

Example

Prediction for $m = 3$

General formula

for each row of the regressors' matrix

$$\left(1\ x_1\ x_2\ x_3\ {x_1}^2\ x_1 x_2\ x_1 x_3\ {x_2}^2\ x_2 x_3\ {x_3}^2\ {x_1}^3\ {x_1}^2 x_2\ {x_1}^2 x_3\ x_1 {x_2}^2\ x_1 x_2 x_3\ x_1 {x_3}^2\ {x_2}^3\ {x_2}^2 x_3\ x_2 {x_3}^2\ {x_3}^3\right)$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Starting point

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Repeat the parameters on the column

omitting duplicate values

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \color{red}{x_2} \\ \color{red}{x_3} \\ \color{blue}{x_3} \end{pmatrix}$$

Figuring out combinations specific to m = 2

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \color{red}{x_2} \\ \color{red}{x_3} \\ \color{blue}{x_3} \end{pmatrix} \xrightarrow[\text{to each section}]{\text{Prepend parameters}} \begin{pmatrix} \color{blue}{x_1} & x_1 \\ \color{blue}{x_1} & x_2 \\ \color{blue}{x_1} & x_3 \\ \color{red}{x_2} & x_2 \\ \color{red}{x_2} & x_3 \\ \color{purple}{x_3} & x_3 \end{pmatrix}$$

Figuring out combinations specific to m = 2

$$\begin{pmatrix} x_1 & x_1 \\ x_1 & x_2 \\ x_1 & x_3 \\ x_2 & x_2 \\ x_2 & x_3 \\ x_3 & x_3 \end{pmatrix} \xrightarrow{\text{Collapse rows into a single value}} \begin{pmatrix} {\color{blue}x_1 \times x_1} \\ {\color{blue}x_1 \times x_2} \\ {\color{blue}x_1 \times x_3} \\ {\color{red}x_2 \times x_2} \\ {\color{red}x_2 \times x_3} \\ {\color{purple}x_3 \times x_3} \end{pmatrix}$$

Figuring out combinations specific to m = 2

Append values to the
row accumulator

$$\begin{pmatrix}1 & x_1 & x_2 & x_3 & {x_1}^2 & x_1 x_2 & x_1 x_3 & {x_2}^2 & x_2 x_3 & {x_3}^2\end{pmatrix}$$

Figuring out combinations specific to m = 2

Repeat for combinations specific to $m = 3$

$$\begin{pmatrix} x_1 & x_1 \\ x_1 & x_2 \\ x_1 & x_3 \\ x_2 & x_2 \\ x_2 & x_3 \\ x_3 & x_3 \end{pmatrix} \longrightarrow \left(\begin{array}{ccc} x_1 & x_1 & x_1 \\ x_1 & x_1 & x_2 \\ x_1 & x_1 & x_3 \\ x_1 & x_2 & x_2 \\ x_1 & x_2 & x_3 \\ x_1 & x_3 & x_3 \\ \hline x_2 & x_2 & x_2 \\ x_2 & x_2 & x_3 \\ x_2 & x_3 & x_3 \\ \hline x_3 & x_3 & x_3 \end{array}\right)$$

Figuring out combinations specific to m = 3

Append values to the
row accumulator

$$\begin{pmatrix}1 & x_1 & x_2 & x_3 & {x_1}^2 & x_1\,x_2 & x_1\,x_3 & {x_2}^2 & x_2\,x_3 & {x_3}^2 & {x_1}^3 & {x_1}^2\,x_2 & {x_1}^2\,x_3 & x_1\,{x_2}^2 & x_1\,x_2\,x_3 & x_1\,{x_3}^2 & {x_2}^3 & {x_2}^2\,x_3 & x_2\,{x_3}^2 & {x_3}^3\end{pmatrix}$$

Figuring out combinations specific to m = 3

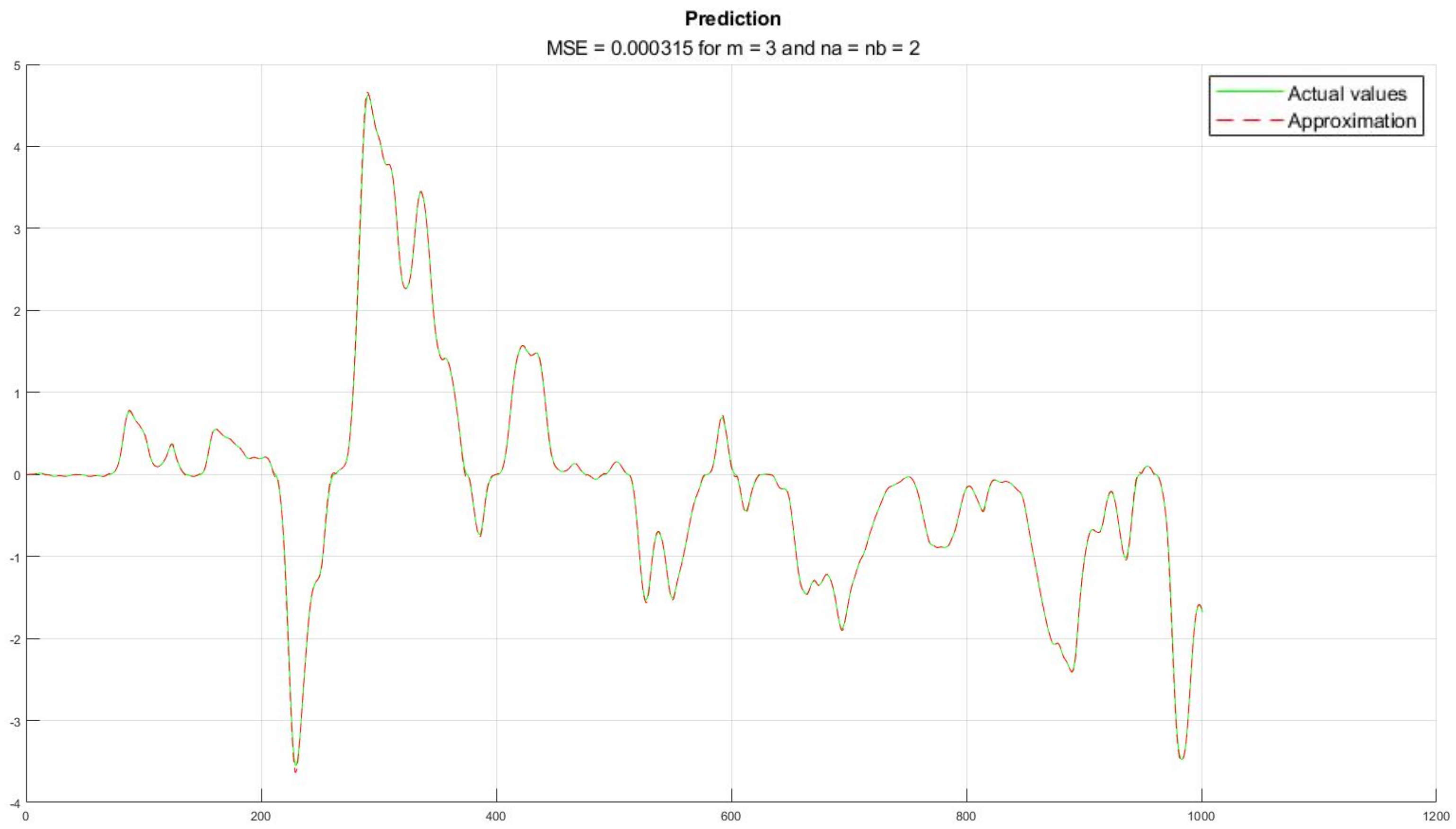
```matlab
function [y_hat, theta] = prediction(m, na, nb, id, val)
    phi_id = computePhi(id.InputData, id.OutputData, na, nb, m);

    theta = phi_id \ id.OutputData;

    phi_val = computePhi(val.InputData, val.OutputData, na, nb, m);

    y_hat = phi_val * theta;
end
```

Now that we have ϕ, we can do the prediction.



Prediction
MSE = 0.000315 for m = 3 and na = nb = 2
Actual values
Approximation
5
4
3
2
1
0
-1
-2
-3
-4
0
200
400
600
800
1000
1200

Now for the simulation

Simulation

$$\hat{y}(1) = \begin{pmatrix} 0 & \dots & 0 \end{pmatrix} \cdot \theta$$

$$\hat{y}(2) = \begin{pmatrix} \hat{y}(1) & 0 & \dots & 0 & u(1) & \dots & 0 \end{pmatrix} \cdot \theta$$

$$\hat{y}(3) = \begin{pmatrix} \hat{y}(2) & \hat{y}(1) & 0 & \dots & 0 & u(2) & u(1) & \dots & 0 \end{pmatrix} \cdot \theta$$

$$\dots$$

The output values are determined incrementally, and is then built using the same algorithm as the prediction

Simulation

```
function y_hat_sim = simulation(m, na, nb, val, theta)
    y_hat_sim = zeros(length(val.OutputData), 1);
    phi_sim = zeros(length(val.OutputData), height(theta));

    phi_sim(1, :) = [1 zeros(1, height(theta) - 1)];
    y_hat_sim(1) = phi_sim(1, :) * theta;
    for k = 2 : length(y_hat_sim)
        phi_sim(k, :) =  buildRow(m, buildXMatrix(y_hat_sim, val.InputData, k, na, nb));
        y_hat_sim(k) = phi_sim(k, :) * theta;
    end
end
```

Code snippet for a function that performs the simulation

Simulation

```
function y_hat_sim = simulation(m, na, nb, val, theta)
    y_hat_sim = zeros(length(val.OutputData), 1);
    phi_sim = zeros(length(val.OutputData), height(theta));

    phi_sim(1, :) = [1 zeros(1, height(theta) - 1)];
    y_hat_sim(1) = phi_sim(1, :) * theta;
    for k = 2 : length(y_hat_sim)
        phi_sim(k, :) =  buildRow(m, buildXMatrix(y_hat_sim, val.InputData, k, na, nb));
        y_hat_sim(k) = phi_sim(k, :) * theta;
    end
end
```

Starting case

With all values from ϕ_{sim} being 0

Code snippet for a function that performs the simulation

Simulation

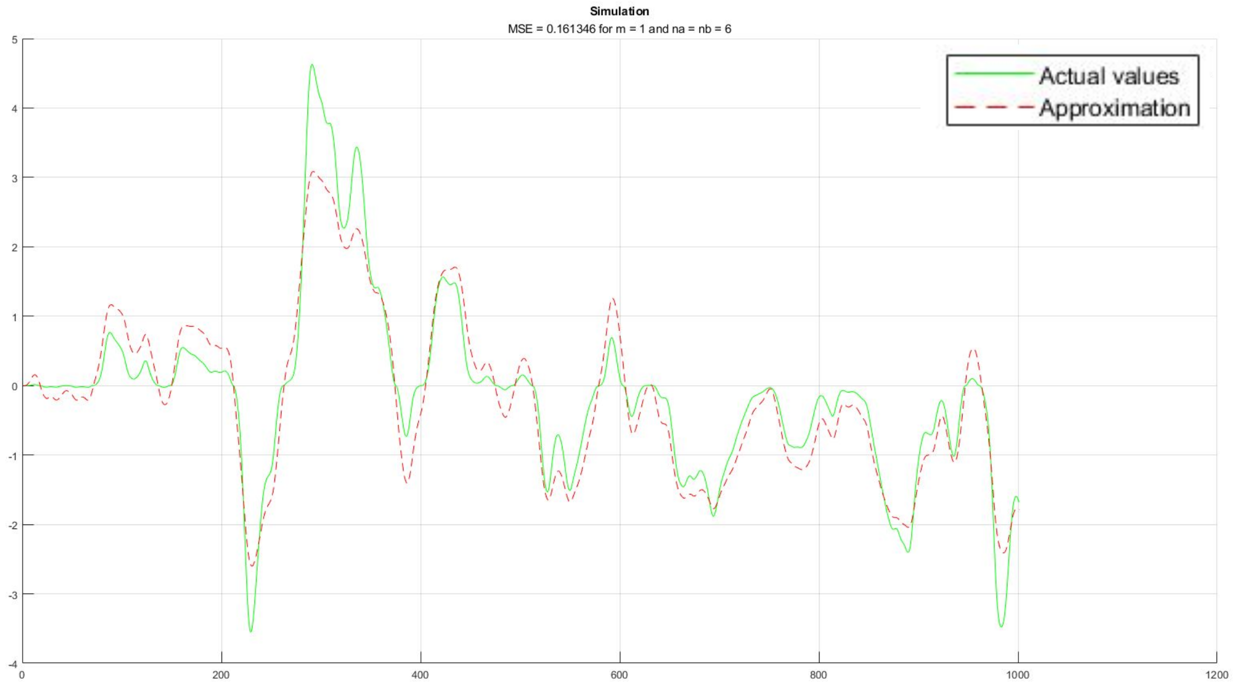
```
function y_hat_sim = simulation(m, na, nb, val, theta)
    y_hat_sim = zeros(length(val.OutputData), 1);
    phi_sim = zeros(length(val.OutputData), height(theta));

    phi_sim(1, :) = [1 zeros(1, height(theta) - 1)];
    y_hat_sim(1) = phi_sim(1, :) * theta;
    for k = 2 : length(y_hat_sim)
        phi_sim(k, :) =  buildRow(m, buildXMatrix(y_hat_sim, val.InputData, k, na, nb));
        y_hat_sim(k) = phi_sim(k, :) * theta;
    end
end
```

Building ϕ_{sim} and y
incrementally, using values determined so far

Code snippet for a function that performs the simulation

Simulation



$\text{MSE} = 0.161346$

3. Tuning results

After generating all the MSE for each $m \in \{1, 2, 3\}$ and $n_a = n_b \in \{1, 2, ..., 15\}$, the minimum value of MSE and the corresponding values for m, n_a, n_b and n_k are saved.

The MSE taken into consideration is the one computed on the validation data set.

The smallest value of MSE represents the best fit. This is met for $y_{prediction}$ when $m=1$ and $na = nb = 5$ and $nk = 1$ (imposed).

MSEuri_pred_val

3x15 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.0046	1.6596e-04	2.5436e-05	1.5452e-05	1.5126e-05	1.5864e-05	2.1980e-05	1.7692e-05	2.0405e-05	2.2168e-05	2.2694e-05	2.1742e-05	2.127
2	0.0046	1.6659e-04	2.5523e-05	2.2660e-05	2.6622e-05	5.1113e-05	1.2198e-04	1.3180e-04	0.0033	0.8932	357.6132	7.3648e+05	2.6685
3	0.0055	2.9552e-04	8.4118e-05	4.4281e-04	0.0012	0.0028	0.0483	1.1343	558.7302	4.5790e+03	1.3080e+03	609.0453	355
4													

We will only take into consideration the first table

(i.e the MSE computed on the validation data set)

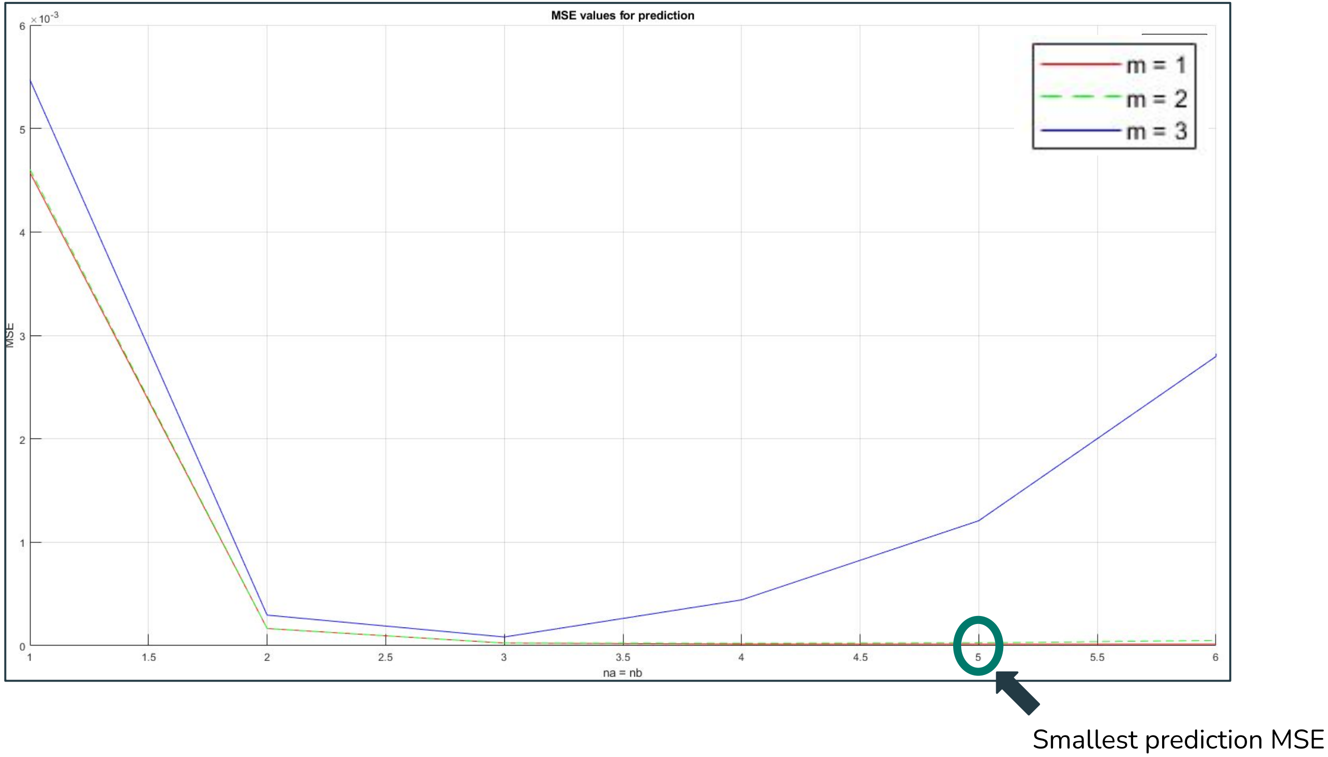
It is notable that the values increase drastically after $n_a = n_b = 6$

MSEuri_pred_id

3x15 double

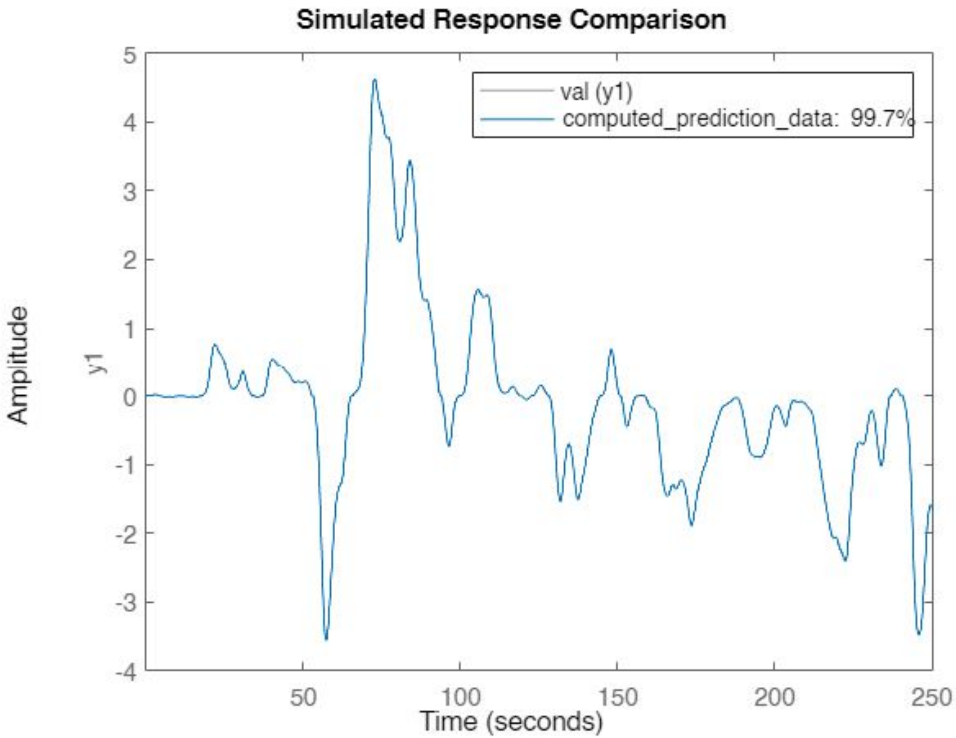
	1	2	3	4	5	6	7	8	9	10	11	12	13
1	3.1551	3.1842	3.1801	3.1804	3.1802	3.1801	3.1799	3.1800	3.1799	3.1800	3.1799	3.1800	3
2	3.1549	3.1838	3.1801	3.1806	3.1808	3.1805	3.1804	3.1779	3.1805	4.1046	360.7542	7.3642e+05	2.6685
3	3.1633	3.1980	3.1749	3.1996	3.1478	3.1566	3.6360	3.0696	558.8713	4.6261e+03	1.3364e+03	628.4665	359

3.1 MSE values for prediction



The prediction MSE, as a function of $n_a = n_b$ and of m.

3.1.1 Fitting for prediction



99.7%

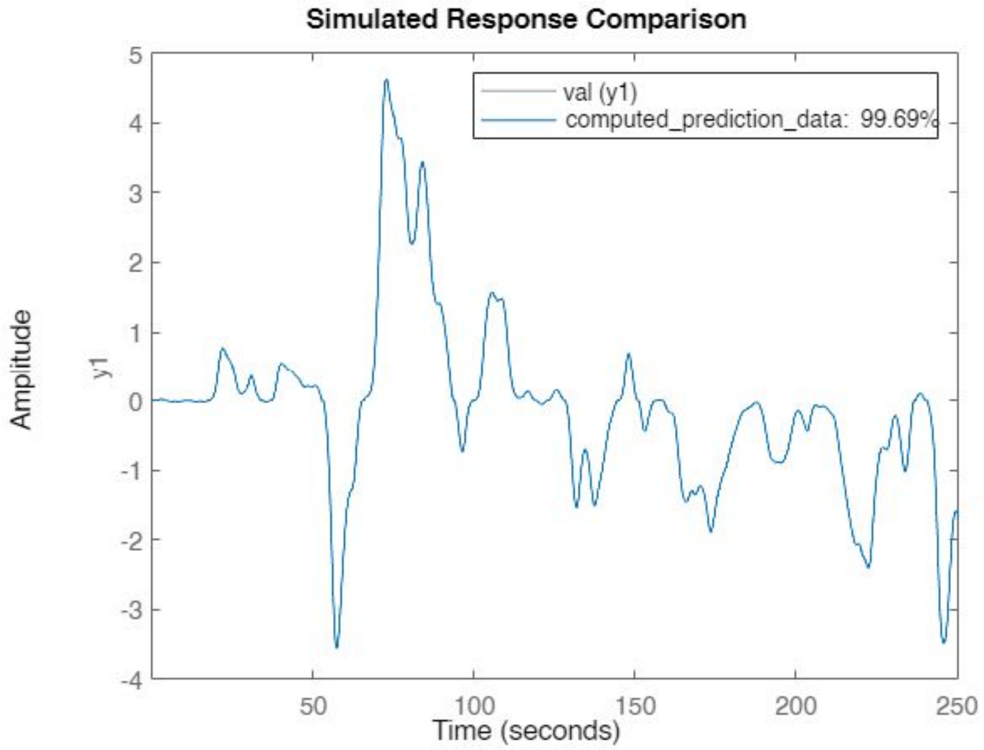
Best fit

$MSE = 0.000015$

$m = 1$

$n_a = n_b = 5$

3.1.1 Fitting for prediction



99.69%

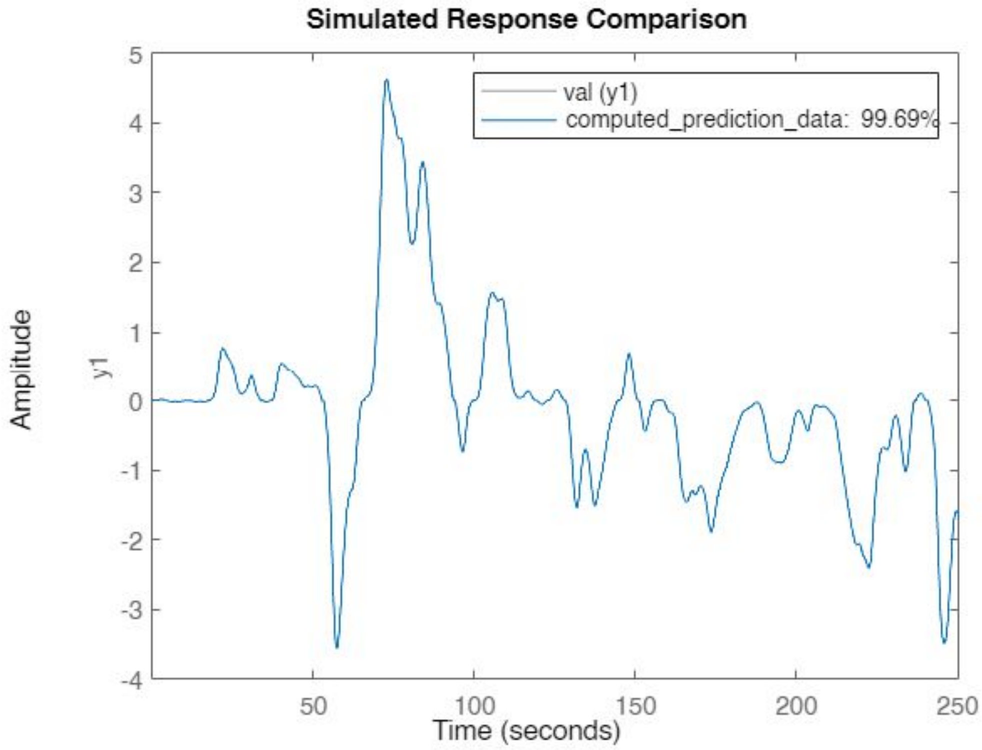
Good fit

MSE = 0.000015

m = 1

$n_a = n_b =$ 4

3.1.1 Fitting for prediction



99.69%

Good fit

MSE = 0.000016

m = 1

$n_a = n_b =$ 6

MSEuri_sim_val

3x15 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.5175	0.2485	0.1799	0.1831	0.1640	0.1613	0.2492	0.2173	0.1984	0.2352	0.2338	0.2502
2	0.5414	0.2411	0.1913	0.4515	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4												

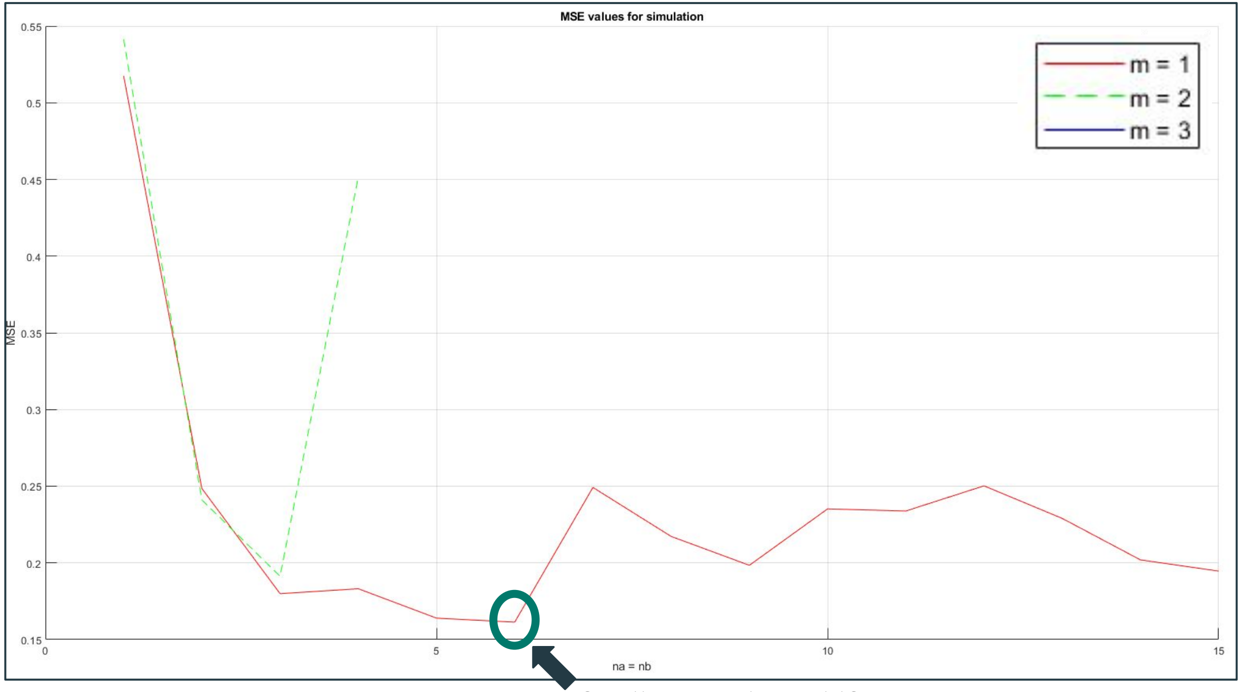
We will take into consideration only the first table, the MSE computed on the validation data set. The approximation becomes unstable after m=2 and na=nb greater than 4.

MSEuri_sim_id

3x15 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	2.8663	3.2608	2.7790	2.9529	2.8411	2.8297	3.0872	2.8440	2.7162	2.6974	2.7177	2.7329
2	2.8498	3.2125	2.8150	2.4275	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4												

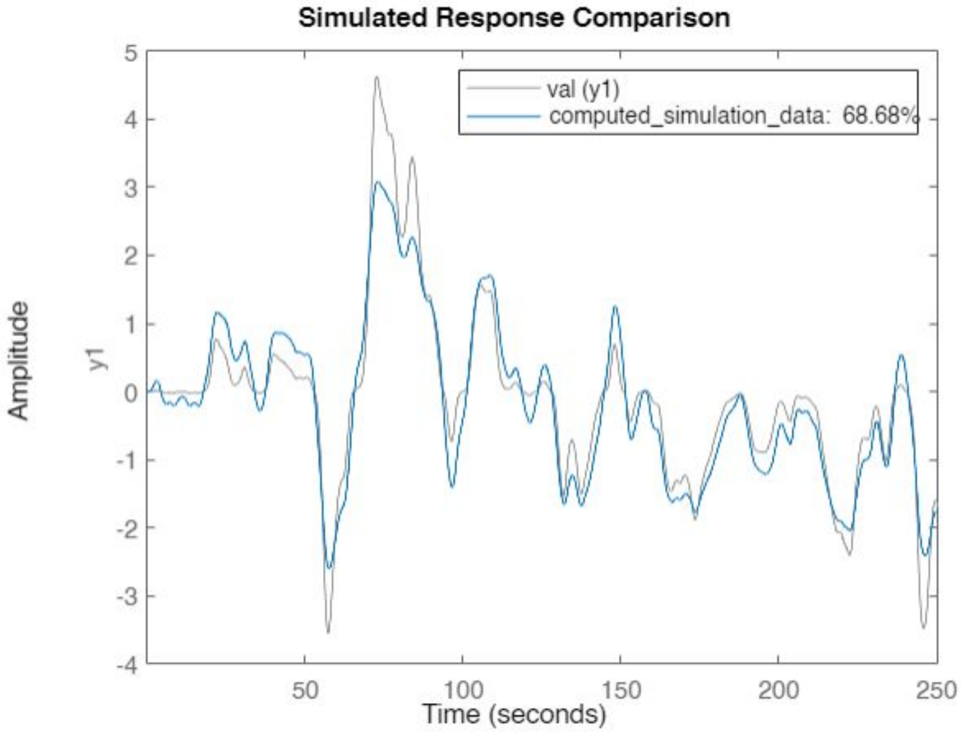
3.2 MSE values for simulation



Smallest simulation MSE

The simulation MSE, as a function of $n_a = n_b$ and of m.

3.1.1 Fitting for simulation



68,68%

Bad fit

But the best we've got

$$MSE = 0.161346$$

$$m = 1$$

$$n_a = n_b = 6$$

4. Conclusions

- $y_{prediction}$ is more accurate than $y_{simulation}$
- The simulation error is greater than the one step ahead prediction error
- For $n_a = n_b > 6$, the approximation struggles to find a good fit for prediction and simulation.