University of Coimbra

Doctoral Program in Information Science and Technology

Real Time Learning in Intelligent Systems

Assignment #1 - Parameter Estimation

Joaquim Pedro Bento Gonçalves Pratas Leitão - 2011150072

March 2, 2017

1 Introduction

The current assignment proposes a recursive identification of two distinct linear systems. The first system is described by an ARX model, while an $ARMAX$ model represents the second system.

The recursive identification task will be based on provided datasets, where inputs and corresponding outputs for the systems in question were collected and stored: for the first system, two distinct datasets were collected each containing an input and output signal over a period of time; in the second system only a single dataset was provided, containing an input signal and the corresponding response of the system.

Based on the provided datasets, the parameters of the models that describe each system can be estimated and, consequently, the corresponding linear systems can be identified. Section 2 covers the estimation procedure for the parameters of the first model, while section 3 is related with the equivalent procedure, applied for the second system.

2 ARX Estimation

The current section covers the estimation of the parameters of the first system considered in this work, described by an ARX model. Subsection 2.1 provides a theoretical introduction to this task, while subsection 2.2 details the implementation.

2.1 Theory

In a simplistic view, a system can be considered as an object in which variables of different natures interact and produce output signals. The system can also be affected by external stimuli, which can be of two kinds: *inputs*, if they can be manipulated by the observer; or *disturbances*, if they cannot. A model of a system describes and represents the system in question by detailing the existing relationship between its observed signals, disturbances and outputs.

A system described by an ARX model is assumed to be composed of two distinct components: A *deterministic* and a *non-deterministic* component. Figure 1 presents such a model. A system of this nature considers a noise filter transfer function, described by S_e, and a system transfer function described by S_d.

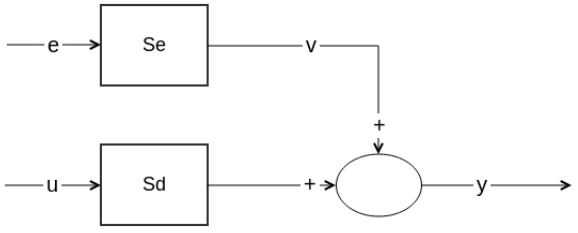

Figure 1: ARX Model.

In a system described by an ARX model, the transfer function S_d has the form $\frac{B(q)}{A(q)}$ and the noise filter has the form $\frac{1}{A(q)}$ where $A(q)$ and $B(q)$ are polynomials of the form:

$A(q) = 1 + q^{-1} \times a_1 + ... + q^{-na} \times a_{na}$ and $B(q) = q^{-1} \times b_1 + ... + q^{-nb} \times b_{nb}$.

Therefore, the system's output is described by:

$$y_k = \frac{B(q)}{A(q)} u_k + \frac{1}{A(q)} e_k$$

The process of estimating the parameters of a system described by an ARX model consists in deriving sounding estimates for the parameters $a_1, ..., a_{na}$ and $b_1, ..., b_{nb}$. A general and simplistic approach for computing these estimations comprises the following steps:

1. Obtain the number of parameters, n_a and n_b. If the values of n_a and n_b are unknown then they must be estimated, using expert knowledge of the system's behaviour and historical information of its inputs and outputs.
2. Compute the estimations of the parameters using a recursive algorithm:
   (a) Consider the disturbance, e_k, to be null (that is, equal to 0) for all time instances. The reason for this assumption has to due with the fact that the disturbance is assumed to be random, following a Gaussian distribution with mean equal to 0. As only input and output information is available, it is impossible to draw any assumption about e_k. As such, it is common to consider its most probable value, that is, 0.
   (b) Define a forgetting factor λ in the range $[0.98, 0.995]$, a parameter vector $\hat{\theta}(0) = [\hat{a_1}\ \hat{a_2}\ \cdots\ \hat{a_{na}}\ \hat{b_1}\ \hat{b_2} \cdots \hat{b_{nb}}]^T$ and a covariance matrix $P(0)$, initialized to a large value.
   (c) In each iteration of the algorithm compute:
   $h(N+1) = [-y_N\ \cdots\ y_{N-n_a}\ u_N \cdots u_{N-n_b}]^T$
   $\hat{y}_{N+1} = \hat{h}_{N+1} \times \hat{\theta}_N$
   $K(N+1) = P(N)\, h(N+1)\, [\hat{h}(N+1)\, P(N)\, h(N+1) + \lambda]^{-1}$
   $E_{N+1} = y_{N+1} - \hat{y}_{N+1}$
   $\hat{\theta}(N+1) = \hat{\theta}(N) + K(N+1)\, E(N+1)$

2.2 Practice

As detailed in section 1, two datasets containing measured input and output signals from the same system were provided at this step. As such, two different sets of parameter estimations were obtained for the same system, described by an ARX model. The practical implementation of the parameter estimation comprises the following steps:

1. The provided datasets, for each estimation task, were divided into an *estimation* and *validation* dataset, with the first 70% of the data being used for estimation and the remaining 30% for validation.
2. As no knowledge regarding the order of the system was available, the number of parameters, n_a and n_b, and the delay of the system, n_k, were estimated from the provided estimation and validation data. To perform this estimation, the function *selstruct* [1] was used. The obtained estimation for the delay (n_k) was further confirmed using the function *delayest*[1].

[1] Available in *MATLAB's System Identification Toolbox.*

3. Using the estimations for the number of parameters obtained in the previous point, create a recursive *ARX* estimator, using the function *recursiveARX*[1]. In both estimation tasks, a forgetting factor of 0.99 was used.

4. Iteratively estimate the parameters of the *ARX* model, using the *step* function[1] and the estimation data. Compute the *Mean Squared Error (MSE)* and the fit in the estimation data.

5. Finally, validate the estimator, using the validation dataset. Two distinct approaches can be used at this point: Similarly to the previous point, the *step* function can be used to compute the output of the estimator, given a series of input values. By defining the value 0 to the property *"EnableAdaptation"*, no adaptation of the estimator's parameters is performed. Alternatively, an *offline* approach can be followed, using the *compare* function[1], which plots the expected and measured outputs, as well as the computed fit for the validation data.

2.2.1 Estimation #1

For the first system to estimate at this point, the *selstruct* function produced the following values for the estimations of n_a, n_b and n_k: 4, 5 and 1, respectively.

After the on-line learning step (described in the forth step of the previous enumeration), an estimation of the *ARX* model was computed. The following polynomials were computed for $A(q)$ and $B(q)$:

$$A(q) = 1 - 1.3101 \times q^{-1} + 0.4119 \times q^{-2} + 0.0986 \times q^{-3} - 0.0948 \times q^{-4}$$

$$B(q) = 0.996 \times q^{-1} - 1.2982 \times q^{-2} + 0.2242 \times q^{-3} + 0.1695 \times q^{-4} - 0.067 \times q^{-5}.$$

A *Mean Squared Error (MSE)* of 3.4019 was registered over the estimation data. A fit of 70.07% was also registered. Figure 2 plots the measured and estimated outputs of the system, as well as the quadratic error in the estimation process.

After the estimation process was completed, the developed estimator needed to be properly validated. In the iterative approach (using the *step* function) a *Mean Squared Error (MSE)* of 1.8455 and a fit of 74.96% were registered. Figure 3 plots the measured and estimated outputs of the system, as well as the quadratic error in the iterative validation process. In the *offline* approach, the *compare* function yield a fit of 45.5052%.

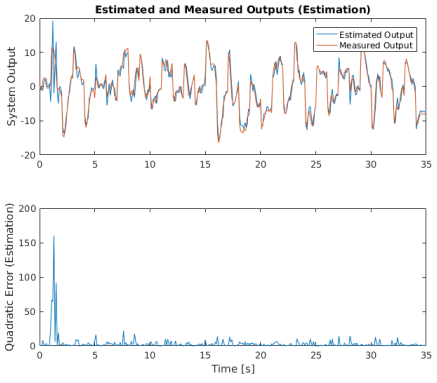

Figure 2: Performance in the estimation.

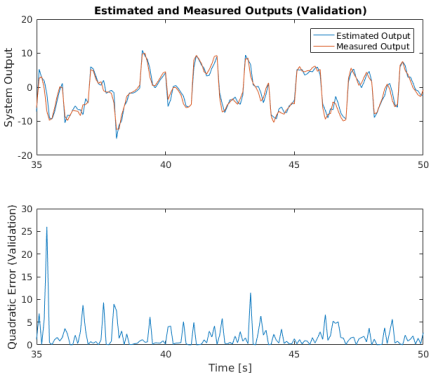

Figure 3: Performance in the validation.

2.2.2 Estimation #2

For the second system to estimate at this point, the *selstruct* function produced the following values for the estimations of n_a, n_b and n_k: 2, 3 and 1, respectively.

After the on-line learning step (described in the forth step of the previous enumeration), an estimation of the ARX model was computed. The following polynomials were computed for $A(q)$ and $B(q)$:

$A(q) = 1 - 1.3059 \times q^{-1} + 0.4511 \times q^{-2}$

$B(q) = 0.9686 \times q^{-1} - 1.2622 \times q^{-2} + 0.2578 \times q^{-3}$.

A *Mean Squared Error (MSE)* of 2.5223 was registered over the estimation data. A fit of 65.70% was also registered. Figure 4 plots the measured and estimated outputs of the system, as well as the quadratic error in the estimation process.

After the estimation process was completed, the developed estimator needed to be properly validated. In the iterative approach (using the *step* function) a *Mean Squared Error (MSE)* of 1.8668 and a fit of 69.62% were registered. Figure 5 plots the measured and estimated outputs of the system, as well as the quadratic error in the iterative validation process. In the *offline* approach, the *compare* function yield a fit of 32.1336%.

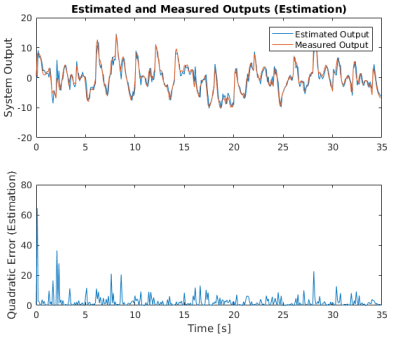

Figure 4: Performance in the estimation.

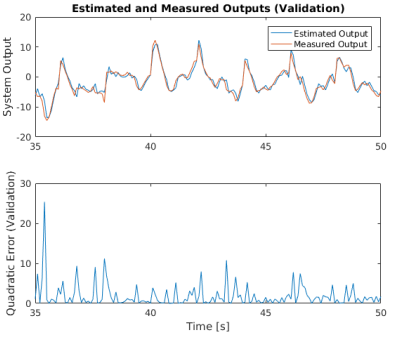

Figure 5: Performance in the validation.

2.2.3 Remarks

In general, the presented results suggest that the computed estimators have a reduced generalization capacity. This statement is supported by the low fit values registered during the validation, namely in the offline approach.

In a posterior stage of the analysis, the obtained estimators were validated with the opposite validation dataset - that is, the estimator obtained in 2.2.1 was validated with the validation dataset of 2.2.2 and vice-versa.

Regarding the first estimator (obtained in 2.2.1), a fit of 65.23% and a *Mean Squared Error (MSE)* of 2.4457 were registered using the iterative approach, while a fit of 30.7064% was registered in the *offline* approach. Regarding the second estimator (obtained in 2.2.2), a fit of 65.42% and a *Mean Squared Error (MSE)* of 3.5215 were registered using the iterative approach, while a fit of 22.4408% was registered in the *offline* approach.

Indeed, from the iterative approach, very similar results were registered with respect to the fit; however, in the offline approach, the first estimator registered an higher fitness on the validation data of the second estimator. Regardless,as already stated, both fitnesses are very small at this point.

These results suggest the need to collect a larger volume of data, which may increase the generalization capacity of the classifier. Regardless, the obtained results also suggest a low signal/noise relationship for the system, making its approximation considerably more difficult, as the signal and the noise are very similar.

3 ARMAX Estimation

The current section covers the estimation of the parameters of the first system considered in this work, described by an $ARMAX$ model. Subsection 3.1 provides a theoretical introduction to this task, while subsection 3.2 details the implementation.

3.1 Theory

A system described by an $ARMAX$ model has a similar structure to the ARX model presented in Figure 1. The difference between the ARX and $ARMAX$ models is in the system's output, which in the case of $ARMAX$ is described by:

$$y_k = \frac{B(q)}{A(q)}u_k + \frac{C(q)}{A(q)}e_k$$

where $C(q) = q^{-1} \times c_1 + ... + q^{-nc} \times c_{nc}$.

In order to compute the parameter estimations, an approach similar to the one presented in 2.1 must be conducted, adjusting the dimensions of the vectors and matrices whenever needed to take into account $C(q)$.

3.2 Practice

The practical implementation of the parameter estimation for the $ARMAX$ model comprises the following steps:

1. The provided datasets, for each estimation task, were divided into an *estimation* and *validation* dataset, with the first 70% of the data being used for estimation and the remaining 30% for validation.

2. As no knowledge regarding the order of the system was available, the number of parameters - n_a, n_b and n_c - and the delay of the system, n_k, were estimated from the provided estimation and validation data. To perform this estimation, a model was estimated for all combinations of the values of n_a, n_b, n_c and n_k, using the $armax$[2] function. At this step, the following intervals were considered: n_a, n_b, $n_c \in [1; 10]$ and $n_k \in [1; 3]$.

[2]Available in *MATLAB's System Identification Toolbox*.

The model which registered the higher fit on the validation dataset was stored. In this work, this was the case for a model with $[n_a, n_b, n_c, n_k] = [7, 7, 10, 1]$.

3. In the next step, a recursive parameter estimation was conducted, using the *recursiveARMAX* and *step* functions[2], similarly to what was performed for the *ARX* estimation. The following polynomials were computed for $A(q)$, $B(q)$ and $C(q)$:

   $A(q) = 1 - 0.6594q^{-1} - 0.7420q^{-2} - 0.0457q^{-3} + 0.6375q^{-4} + 0.5912q^{-5} - 0.9068q^{-6} + 0.1648q^{-7}$

   $B(q) = 0.9032q^{-1} - 0.6362q^{-2} + 0.9096q^{-3} - 0.0934q^{-4} + 0.7044q^{-5} + 0.7503q^{-6} - 0.8708q^{-7}$.

   $C(q) = 1 + 0.3564q^{-1} - 0.7181q^{-2} - 0.5632q^{-3} + 0.4154q^{-4} + 0.7232q^{-5} - 0.1983q^{-6} - 0.1687q^{-7} + 0.0836q^{-8} + 0.0836q^{-9} + 0.0139q^{-10}$

   A *Mean Squared Error (MSE)* of 3.2293 and a fit of 68.25% were registered. Figure 6 plots the measured and estimated outputs of the system, as well as the quadratic error in the iterative estimation process.

4. Finally, the estimator was validated using the validation data. Again, both iterative and offline approaches were performed. In the iterative approach a MSE of 1.2521 and a fit of 79.12% were registered. Figure 7 plots the measured and estimated outputs of the system, as well as the quadratic error in the iterative validation process. In the offline approach a fit of 64.7406% was registered.

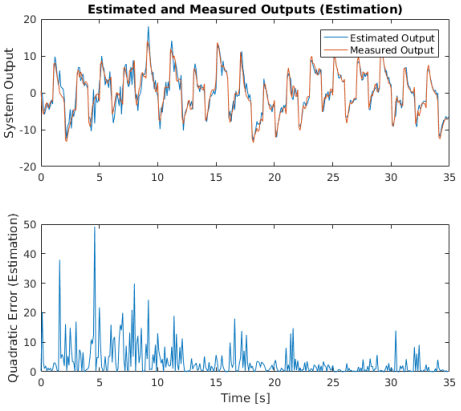

Figure 6: Performance in the estimation.

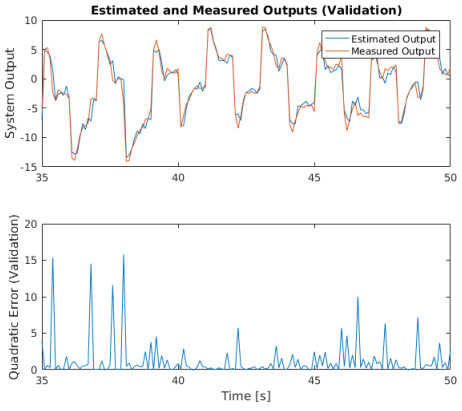

Figure 7: Performance in the validation.