PREPROCESSING OF EXPERIMENTAL DATA FOR USE IN MODEL BUILDING AND MODEL VALIDATION

Magnus Komperød[1], Tor Anders Hauge[2], Bernt Lie[3]

[1]Østfold University College, Faculty of Engineering
Halden, Norway

[2]Xstrata Nikkelverk
Kristiansand, Norway

[3]Telemark University College, Faculty of Technology,
Department of Electrical Engineering,
Information Technology, and Cybernetics
Porsgrunn, Norway

Bernt.Lie@hit.no (Bernt Lie)

Abstract

A method for outlier detection based on the residual of system identification is presented in [1]. An alternative approach to this method is derived in this paper; outliers may cause large innovation processes (absolute values). Hence, such outliers may be detected by searching for samples having large innovation processes. In [2] it is proved that the innovation process can be identified directly from system input and system output data, without relying on models. This method is therefore well suited for outlier detection. In [3] it is shown that the innovation process identified by the method of [2] is identical to the residual of an ARX identification. Hence, the method to be derived in this paper can mathematically be proved to be a special case of the method presented in [1]: If an ARX algorithm is chosen for the system identification step in the method of [1], then the method of [1] and the method to be presented in this paper become mathematically identical. The latter method is tested on experimental data from the copper refining process of Xstrata Nikkelverk, Kristiansand, Norway. The MATLAB command `delayest` is used to estimate time delays between system inputs and system outputs in a dataset. A simulation study of `delayest` shows that the command is sensitive to: (i) The specified model order. (ii) Stochastic elements in the dataset. (iii) Whether all time delays are estimated during one execution of `delayest`, or whether only one time delay is estimated for each execution. `delayest` is also tested on experimental data from Xstrata Nikkelverk. These tests confirm that `delayest` is sensitive to the factors listed above.

Keywords

Data preprocessing; Outlier detection; Time delay estimation; Time delay identification.

1 Introduction

Modeling of dynamic systems is a most important part of today's science and engineering. Dynamic models serve many purposes. For example: (i) Training of process operators, pilots, and astronauts. (ii) Exploring systems in a different time scale than physical time. (iii) Testing systems by simulations before they are manufactured. For example ships, airplanes, missiles, and sub-sea oil installations. (iv) Model-based control, such as LQG control, model-based predictive control (MPC), linear and nonlinear decouplers, Smith predictors, etc.

In those cases where the systems to be modeled already exist and it is possible to log data from the systems, it may be desirable to include such experimental data in the modeling work. For mechanistic (first principle) modeling, experimental data may be used for parameter estimation. For empirical modeling (black-box modeling), including system identification, the models are built directly from experimental data. Experimental data is also essential with respect to model validation.

Experimental data logged from real-life systems, such as process industry, often requires preprocessing before they can be used for model building and model validation. This paper considers two sorts of data preprocessing that are frequently required on experimental data:

1. Outlier detection: In [1], a method for outlier de-

tection is presented. This method is based on the residuals of system identification: Samples having particularly large residuals (one-step-ahead prediction errors) should be inspected further as they may be associated with outliers. An alternative approach to this method for outlier detection will be derived in this paper.

2. Time delay estimation: The MATLAB command `delayest` is included in the MATLAB System Identification Toolbox. This command is used to estimate time delays between system inputs and system outputs in a dataset. A simulation study of `delayest` is presented in this paper. A suggestion for improvement that will make `delayest` less sensitive to the specified model order is also presented.

Both with respect to outlier detection and with respect to time delay estimation, real-life examples from the copper refining process at Xstrata Nikkelverk, Kristiansand, Norway, will be presented. Three process inputs, u^1, u^2, and u^3, and one process output, y, are considered in the examples. u^1 is the mass flow from the roasting furnace to the copper leaching process. u^2 is a recirculation flow from after the electro winning back to the copper leaching process. u^3 is the flow of sulphuric acid (H_2SO_4) to the copper leaching process. y is the concentration of sulphuric acid before the electro winning.

2 Notation and Definitions

The inputs to a system are collected in the input column vector, $u \in \mathbb{R}^{r\times 1}$, where r is the number of inputs. The outputs from a system are collected in the output column vector, $y \in \mathbb{R}^{m\times 1}$, where m is the number of outputs. A sub-script to these vectors refers to the sampling number. A super-script refers to the input or output number. For example u_k^2 refers to input no. 2 at sample no. k.

The symbol τ refers to true time delay (number of samples). τ is in general unknown and is subject to estimation. A super-script to τ, for example τ^2, refers to the time delay from input u^2 to the output y. $\hat{\tau}$ refers to time delay estimates. $\bar{\tau}$ refers to a time delay given as argument to a system identification algorithm. $\bar{\tau}$ may or may not be equal to the actual time delay, τ.

Def. 1 (Innovation Process)**.** The system output, y_k, may be decomposed into two components: (i) The component of y_k that can be predicted from previous inputs, $u_{-\infty}, \ldots, u_{k-1}$, and previous outputs, $y_{-\infty}, \ldots, y_{k-1}$, assuming no model errors. This predictable component of y_k is referred to as $\bar{y}_k$. (ii) The complement of $\bar{y}_k$, i.e. the component of y_k that can *not* be predicted from previous inputs and previous outputs. This is referred to as the innovation process, ε_k. Hence, $\varepsilon_k = y_k - \bar{y}_k$.

For bi-proper systems, i.e. systems having direct feed-through from the input, u_k, to the output, y_k, the current input, u_k, may also be included in the prediction of y_k. For simplicity, only strictly proper systems, i.e. systems without direct feed-through from u_k to y_k, will be considered in this paper.

The symbol ε is used for the true innovation process, which in general is unknown. The symbol ϵ is used for the identified innovation process. The identified innovation process is in general not exactly identical to the true innovation process.

Def. 2 (Orthogonal Projection)**.** The orthogonal projection of G onto H, G/H, is defined as in Eq. (1) [2].

$$G/H \overset{\text{def}}{=} GH^T(HH^T)^{\dagger}H \tag{1}$$

The super-script $\dagger$ refers to the Moore-Penrose pseudo-inverse.

Def. 3 (Complement of Orthogonal Projection)**.** The complement of the orthogonal projection of G onto H, $GH^{\perp}$, is defined by Eq. (2) [2].

$$GH^{\perp} \overset{\text{def}}{=} G - G/H \overset{\text{def}}{=} G - GH^T(HH^T)^{\dagger}H \tag{2}$$

Def. 4 (ARMAX Model Form)**.** Eq. (3) defines the general form of ARMAX models [1].

$$A(q)y_k = B(q)u_k + C(q)\varepsilon_k \tag{3}$$

In Eq. (3), q is the time-shift operator of the Z-transform, i.e. $q^{-1}\, y_k = y_{k-1}$. The symbol q is commonly used within the subject of system identification. The symbol z is used in many other contexts. $A(q)$, $B(q)$, and $C(q)$ are polynomials. n_A, n_B, and n_C are the number of coefficients in the polynomials that in general are different from 1. The $A(q)$ polynomial and the $C(q)$ polynomial are monic polynomials, i.e. the coefficient of their highest order term is 1.

Def. 5 (ARX Model Form)**.** Eq. (4) defines the general form of ARX models [1].

$$A(q)y_k = B(q)u_k + \varepsilon_k \tag{4}$$

$A(q)$ is a monic polynomial.

3 Outlier Detection

Outliers may be data that for one sample, or for a few samples, go to unlikely values, and then back to normal values. Such outliers can usually be seen by plotting the dataset. However, as shown in [1], outliers may also be data that are *not* unlikely values, but rather average values. These data can for example be outliers because the derivative of the data changes unlikely fast, i.e. the second derivative has unlikely large absolute value.

In [1, Example 14.1] the following method for outlier detection is suggested: (i) Identify an empirical model based on the dataset using a system identification algorithm. In [1, Example 14.1], a 3rd order ARMAX model is used. (ii) Run a one-step-ahead prediction simulation using the empirical model and the dataset. (iii) Plot the one-step-ahead prediction errors from the simulation to detect outliers: Large prediction errors (absolute values) may be caused by outliers.

3.1 An Alternative Approach to a Method for Outlier Detection Presented in [1]

This subsection derives an alternative approach to the method for outlier detection presented in [1]. This alternative approach is based on sub-space system identification, in particular the work presented in [2].

Most real-life systems do have some innovation process, ε. The innovation process is typically caused by unmeasured process disturbances and / or measurement noise. Assume that there is a measurement error in one of the variables used to compute $\bar{y}_k$ or in y_k that is significantly larger than the normal measurement noise. This is likely to cause a mismatch between y_k and $\bar{y}_k$ that is also significantly larger than normal. According to Def. 1: Assuming that the model used to compute $\bar{y}_k$ has no modeling error, then the mismatch between y_k and $\bar{y}_k$ is the innovation process, ε_k. Hence, measurement errors may be detected by looking for samples having unusually large innovation processes (absolute values).

In [2] it is proved that for linear time-invariant (LTI) systems, the innovation process, ε_k, can be identified directly from previous inputs, $u_{-\infty}, \ldots, u_{k-1}$, and previous outputs, $y_{-\infty}, \ldots, y_{k-1}$, without relying on models. As infinite number of preceding samples can not be used in any practical problems, only the J preceding samples are used, where J is a user-specified parameter. The identified innovation process, ϵ_k, is the complement of the orthogonal projection of current outputs, y_k, onto inputs and outputs from the J preceding samples, $u_{k-J}, \ldots, u_{k-1}$ and $y_{k-J}, \ldots, y_{k-1}$ [2]. This can mathematically be written as Eq. (5). The matrices of Eq. (5) are as presented in Eq. (6) to Eq. (8). K is the number of columns of the matrices of Eq. (6) to Eq. (8) [2, 4, 3]. (Eq. (5) to Eq. (8) are obtained by choosing $L = 1$ and $g = 0$ in the derivation of [2], as shown in [5, 4, 3].)

$$\epsilon_{J|1} = Y_{J|1} \begin{bmatrix} U_{0|J} \\ Y_{0|J} \end{bmatrix}^{\perp} \quad (5)$$

$$Y_{J|1} = \begin{bmatrix} y_J & y_{J+1} & \cdots & y_{J+K-1} \end{bmatrix} \in \mathbb{R}^{m \times K} \quad (6)$$

$$\epsilon_{J|1} = \begin{bmatrix} \epsilon_J & \epsilon_{J+1} & \cdots & \epsilon_{J+K-1} \end{bmatrix} \in \mathbb{R}^{m \times K} \quad (7)$$

$$\begin{bmatrix} U_{0|J} \\ Y_{0|J} \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & \cdots & u_{K-1} \\ u_1 & u_2 & \cdots & u_K \\ \vdots & \vdots & \ddots & \vdots \\ u_{J-1} & u_J & \cdots & u_{K+J-2} \\ y_0 & y_1 & \cdots & y_{K-1} \\ y_1 & y_2 & \cdots & y_K \\ \vdots & \vdots & \ddots & \vdots \\ y_{J-1} & y_J & \cdots & y_{K+J-2} \end{bmatrix} \in \mathbb{R}^{(r+m)J \times K} \quad (8)$$

The method for identifying the innovation process given by Eq. (5) to Eq. (8) is equivalent to the first step of the DSR E subspace system identification algorithm [5, 4]. The DSR E algorithm is presented in [5, 4]. DSR E should not be confused with the DSR algorithm, although DSR and DSR E are closely related.

The identified innovation process, ϵ_k, is plotted for all timesteps, k (except for the first J timesteps, which are used for initialization purposes). The upper subplot of Fig. 1 shows an ϵ plot. Timesteps having particularly large innovation processes (absolute values) can easily be identified as peaks in the ϵ plot. These timesteps and the surrounding timesteps are candidates for being outliers and should be subject to further inspection.

In [3] it is proved that the innovation process identified by the method of [2], i.e. Eq. (5) to Eq. (8), is mathematically identical to the residual from identification of a strictly proper ARX model where $n_A = n_B = J$. This residual can also be expressed as the one-step-ahead prediction error when simulating the identified ARX model on its own training dataset. Hence, the method for outlier detection based on the identified innovation process, as presented above, is a special case of the method explained in [1, Example 14.1]: If choosing a strictly proper ARX model where $n_A = n_B = J$ in the system identification step of [1, Example 14.1], then the two methods become mathematically identical.

With respect to the method of [1] and the method de-

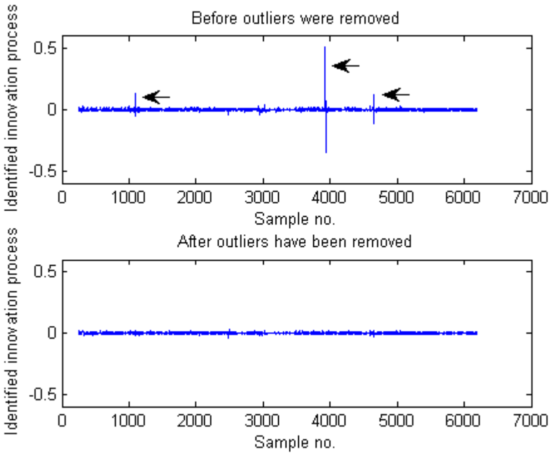

Fig. 1 The innovation process identified by Eq. (5) to Eq. (8) before (upper subplot) and after (lower subplot) outliers have been removed.

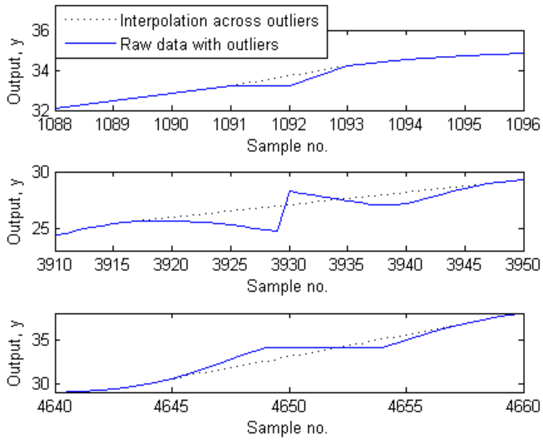

Fig. 2 The process output, y, zoomed in at the samples marked by arrows in Fig. 1.

rived above, it is most important to be aware that these methods identify samples having large identified innovation processes / prediction errors. One should *not* conclude that samples having large identified innovation processes in general are *equivalent* to samples associated with outliers: (i) Large innovation processes may be caused by other factors than outliers, for example large, sudden process disturbances. In other words; large innovation process at a sample indicates that this sample and the surrounding samples should be objects to further inspection. Whether it actually is an outlier, and if so, how to handle it, is a decision to be taken by the human data analyzer (or by other methods). (ii) The identified innovation process may be significantly different from the true innovation process. For example, the method of [1] and the method derived above assume linear time-invariant (LTI) systems (unless a nonlinear system identification method is used in the method of [1]). Strongly nonlinear systems may cause large deviations between the true innovation process and the identified innovation process. (iii) There may also be outliers that do not cause particularly large innovation process and are therefore not detected by these methods.

3.2 Outlier Detection on Industrial Data

This subsection presents a real-life industrial example of the method for outlier detection presented in Subsec. 3.1. The example is based on experimental data from the copper refining process at Xstrata Nikkelverk, Kristiansand, Norway. This process is briefly explained in Sec. 1.

The innovation process identified in a dataset from Xstrata Nikkelverk is shown in the upper subplot of Fig. 1. The innovation process was identified using Eq. (5) to Eq. (8). The three largest peaks of the ϵ plot, indicated by arrows, will be considered in this example. The exact sample numbers of these peaks must be identified. The zoom options of MATLAB figures may be used for this purpose.

The subplots of Fig. 2 show the output, y, zoomed in at the samples indicated by arrows in Fig. 1. The solid lines in Fig. 2 show y before outliers were removed. The samples indicated by arrows in Fig. 1 are all at samples where the derivative of y changes very fast, i.e. the second derivative has high absolute value. These data were classified as outliers and were removed by linear interpolations across them. These interpolations are shown by dotted lines in Fig. 2. After these outliers were removed, the innovation process was re-identified using Eq. (5) to Eq. (8). The new ϵ plot is shown in the lower subplot of Fig. 1. The three peaks marked by arrows in the upper subplot are no longer present. Hence, it is reasonable to conclude that these peaks were caused by the outliers shown in Fig. 2.

Consider criterion V defined by Eq. (9).

$$V \overset{\text{def}}{=} \frac{1}{N}\sum_{k=1}^{N} {\epsilon_k}^2 \tag{9}$$

In Eq. (9), N is the number of samples in the dataset (minus the number of samples used for initialization purposes and to compensate for time delays). Criterion V is a commonly used model fit criterion within the subject of system identification as ϵ is identical to the one-step-ahead prediction error. The value of V was reduced by more than 80% when removing the outliers. Hence, removing the outliers resulted in significantly better model fit.

4 Time Delay Estimation

When building models using system identification, time delays from system inputs to system outputs may cause problems if not handled properly. If the time delays are not compensated for, a large number of model states may be necessary to model the dynamics of the time delays. Too many model states are unfortunate because the identified models may be over-fitted. Over-fitted models may perform very well on their respective training datasets, but such models will in general perform poorly on independent validation datasets. If the time delays are known, they may be compensated for by shifting the input dataseries with respect to the output dataseries.

This section considers the MATLAB command `delayest`, which is included in the MATLAB System Identification Toolbox [6]. This command serves the purpose of estimating time delays from the system inputs, u, to a system output, y, based on a dataset. The command can only handle single output systems, i.e. $y \in \mathbb{R}^{1\times 1}$. Quoting [6]:

> *The `delayest` command estimates the time delay in a dynamic system by estimating a low-order, discrete-time ARX model with a range of delays, and then choosing the delay that corresponding to the best fit.*

Subsec. 4.1 presents a simulation study of `delayest`. This subsection also includes a suggestion for improvement of `delayest`. In Subsec. 4.2, `delayest` is tested on experimental data from the copper refining process at Xstrata Nikkelverk; the estimates of `delayest` are compared to the estimates of [7], which are based on process knowledge.

The following properties of the time delay estimates made by `delayest` will be considered: (i) The accuracy of the estimates, i.e. how close the estimates are to the actual time delays (or to the estimates of [7]). (ii) Whether the time delay estimates are sensitive to the model order specified to `delayest`. (iii) Whether the time delays estimates are sensitive to stochastic elements in the dataset. (iv) Whether the time delay estimates are sensitive to whether the system is considered as one 3×1 system or as three 1×1 systems. In the 3×1 case, all three inputs, u^1, u^2 and u^3, are provided to `delayest` in one single execution of the command. `delayest` then estimates $\hat{\tau}^1$, $\hat{\tau}^2$ and $\hat{\tau}^3$ by making ARX models based on u^1, u^2 and u^3, and of course y. In the 1×1 case, only one input is provided to `delayest` at each execution. Hence, three executions are needed to estimate the three time delays. The 1×1 case is to be preferred with respect to computational efficiency [3].

4.1 Simulation Study of Time Delay Estimation

In the experiments presented in this subsection, two models are used: (i) A single input, single output (SISO) ARMAX model on the form of Eq. (10). This model is referred to as model no. 1. (ii) A multiple input, single output (MISO) ARMAX model on the form of Eq. (11). This model is referred to as model no. 2.

$$A(q)y_k = B^1(q)u_k^1 + C(q)\varepsilon_k \tag{10}$$

$$\begin{aligned} A(q)y_k &= B^1(q)u_k^1 + B^2(q)u_k^2 \\ &\quad + B^3(q)u_k^3 + C(q)\varepsilon_k \end{aligned} \tag{11}$$

For the polynomials of Eq. (10) and Eq. (11), $n = n_A = n_{B^1} = n_{B^2} = n_{B^3} = n_C = 6$. Both models are asymptotically stable.

The experiment to be presented in the following aims to explore the accuracy of `delayest` and which factors the command are sensitive to. `delayest` is tested on two datasets generated by simulations based on model no. 2. The input data used in the simulations, u^1, u^2, and u^3, are based on experimental data from Xstrata Nikkelverk. One dataset was generated as no innovation process, ε, was applied to the simulation. This dataset is referred to as the deterministic case. The other dataset was generated as a white noise sequence was applied for simulating the innovation process, ε. This dataset is referred to as the stochastic case. Except for the innovation process, the datasets were generated under identical condition. After the simulations, the input dataseries, u^1, u^2 and u^3, are shifted with respect to the output, y, so that $\tau^1 = 20$, $\tau^2 = 30$, and $\tau^3 = 50$. The deterministic and the stochastic datasets are then provided to `delayest` for estimation of the time delays, τ^1, τ^2, and τ^3.

`delayest` requires the user to specify intervals of possible values for $\hat{\tau}^1$, $\hat{\tau}^2$, and $\hat{\tau}^3$. `delayest` will only consider these intervals when estimating time delays. In the simulation study presented here, these intervals are set to ±10 the true time delays: $\hat{\tau}^1$ is specified to be in the interval $[10, 30]$, $\hat{\tau}^2$ is specified to be in the interval $[20, 40]$, and $\hat{\tau}^3$ is specified to be in the interval $[40, 60]$. `delayest` also requires the user to specify $\bar{n}_A$, $\bar{n}_{B^1}$, $\bar{n}_{B^2}$, and $\bar{n}_{B^3}$ of the ARX models to be identified during execution of the command. The bar symbol is here used to avoid confusion with the corresponding values of the ARMAX models, i.e. Eq. (10) and Eq. (11). In this simulation study, these values are always chosen so that $\bar{n}_A = \bar{n}_{B^1} = \bar{n}_{B^2} = \bar{n}_{B^3}$. This value will be referred to as $\bar{n}$.

Fig. 3 to Fig. 6 show the estimates of `delayest`, $\hat{\tau}^1$, $\hat{\tau}^2$, and $\hat{\tau}^3$, plotted as functions of $\bar{n}$. Fig. 3 shows the deterministic 3×1 case, Fig. 4 shows the stochastic 3×1 case, Fig. 5 shows the deterministic 1×1 case, and Fig. 6 shows the stochastic 1×1 case. These simulations show that the estimates of `delayest` are sensitive to:

1. The value of $\bar{n}$. If `delayest` was not sensitive to the value of $\bar{n}$, then the curves in Fig. 3 to Fig. 6 would have been horizontal lines.
2. Stochastic elements in the dataset. If `delayest` was not sensitive to stochastic elements, then the curves of Fig. 3 and Fig. 4 would have been identical. Likewise for Fig. 5 and Fig. 6.
3. Whether the 3×1 approach or the 1×1 approach is used. If `delayest` was not sensitive to this, the curves of Fig. 3 and Fig. 5 would have been identical. Likewise for Fig. 4 and Fig. 6.

In the simulations presented in Fig. 3 to Fig. 6, `delayest` estimates τ^1, τ^2, *and* τ^3 exactly correct

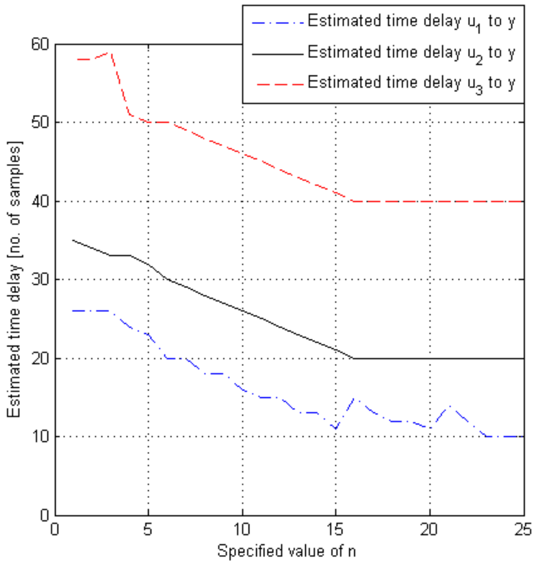

Fig. 3 Time delay estimates, $\hat{\tau}^1$, $\hat{\tau}^2$, and $\hat{\tau}^3$, plotted as functions of $\bar{n}$ for the deterministic 3×1 case.

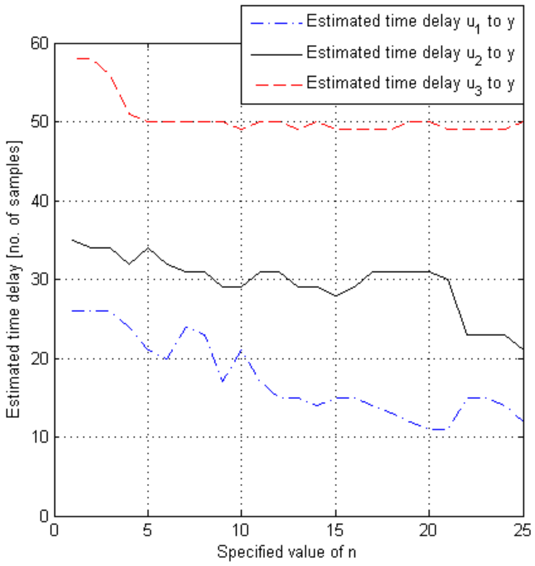

Fig. 5 Time delay estimates, $\hat{\tau}^1$, $\hat{\tau}^2$, and $\hat{\tau}^3$, plotted as functions of $\bar{n}$ for the deterministic 1×1 case.

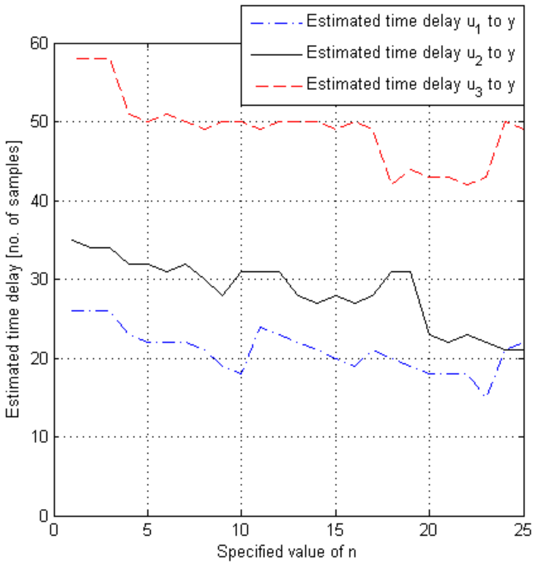

Fig. 4 Time delay estimates, $\hat{\tau}^1$, $\hat{\tau}^2$, and $\hat{\tau}^3$, plotted as functions of $\bar{n}$ for the stochastic 3×1 case.

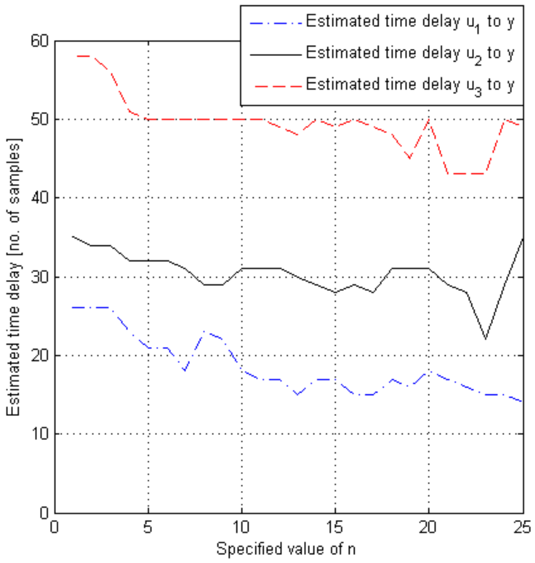

Fig. 6 Time delay estimates, $\hat{\tau}^1$, $\hat{\tau}^2$, and $\hat{\tau}^3$, plotted as functions of $\bar{n}$ for the stochastic 1×1 case.

only when the following three criteria are met: (i) The correct polynomial orders are specified, i.e. $\bar{n} = 6$. (ii) Stochastic elements are *not* present in the dataset, i.e. $\varepsilon = 0$. (iii) The 3×1 approach is used. In any other cases, `delayest` fails to make exactly correct estimates for all three time delays.

In particular the deterministic 3×1 case, i.e. Fig. 3, and to some extend also the other figures show that there is a *systematic* error present: The time delay estimates tend to decrease as $\bar{n}$ increases. In the deterministic 3×1 case, $\hat{\tau}^2$ and $\hat{\tau}^3$ decrease as $\bar{n}$ increase from $\bar{n} = 6$ until $\hat{\tau}^2$ and $\hat{\tau}^3$ saturate at the lower ends of their specified intervals, i.e. at $\bar{n} = 16$. Over the interval $\bar{n} \in [6, 16]$, $\bar{n}$ has increased by 10 and $\hat{\tau}^2$ and $\hat{\tau}^3$ have decreased by 10.

In the following, a suggestion for improvement of `delayest` is derived. This improvement makes `delayest` less sensitive for the value of $\bar{n}$. The improvement works only for deterministic, linear time-invariant (LTI) systems. The derivation presented here is based on a single input, single output (SISO) system. In [3] it is shown that this improvement can easily be extended to multiple input, single output (MISO) systems.

Based on model no. 1, a deterministic dataset was generated using an input sequence, u^1, from a dataset of Xstrata Nikkelverk. No innovation process, ε, was applied to the simulation. The input dataserie, u^1, was shifted with respect to the output dataserie, y, so that $\tau^1 = 20$.

Please note the difference between the symbols ε and ϵ. ε is the innovation process generated by a random generator. In this derivation $\varepsilon = 0$. ϵ is the one-step-ahead prediction error. In the following, ϵ refers to the one-step-ahead prediction error as a model is simulated on its own training dataset.

For a given dataset and a given system identification algorithm, in this case ARX, the one-step-ahead prediction error, ϵ, and inherently the model fit criterion, V, as defined in Eq. (9), are functions of the parameters specified to the ARX system identification algorithm: (i) The specified time delay, $\bar{\tau}^1$, and (ii) the specified values of $\bar{n}_A$ and $\bar{n}_{B^1}$. In this derivation, $\bar{n}_A$ and $\bar{n}_{B^1}$ are always chosen so that $\bar{n}_A = \bar{n}_{B^1}$. This value will be referred to as $\bar{n}$. Hence, $\epsilon = \epsilon\left(\bar{\tau}^1, \bar{n}\right)$ and $V = V\left(\bar{\tau}^1, \bar{n}\right)$.

Fig. 7 shows V plotted as function of $\bar{\tau}^1$ for $\bar{n} = 5$, for $\bar{n} = 6$, and for $\bar{n} = 12$. Please note that: (i) For $\bar{n} = 5$, V does not reach zero for any value of $\bar{\tau}^1$. (ii) For $\bar{n} = 6$, i.e. the same value as for model no. 1: V does reach zero ($< 10^{-28}$) for $\bar{\tau}^1 = \tau^1 = 20$, i.e. for the correct time delay, but not for any $\bar{\tau}^1 \neq \tau^1$. (iii) For $\bar{n} = 12$, V is zero ($< 10^{-28}$) for $\bar{\tau}^1 \in [14, 20]$. Values below 10^{-28} are considered as zero as it is assumed that these values differ from zero only due to numerical reasons.

Further simulations were run, which are *not* presented in this paper. These simulations indicate that the observations presented in Fig. 7 can be generalized to the following two statements; Statement (i): For $\bar{n} < n$, V does not reach 0 for any $\bar{\tau}^1$. Statement (ii): For $\bar{n} \geq n$, $V = 0$ for any $\bar{\tau}^1 \in [\tau^1 - (\bar{n} - n), \tau^1]$.

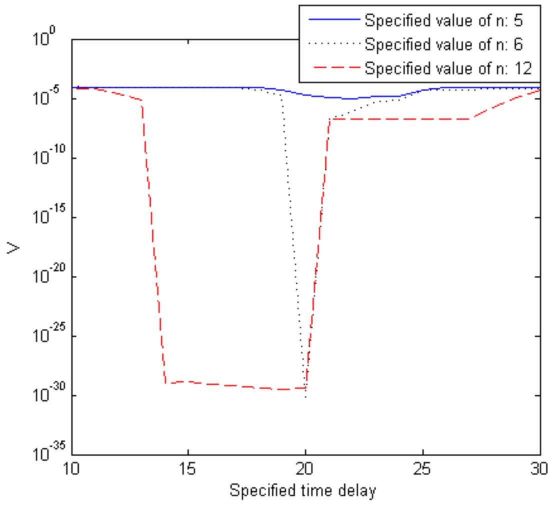

Fig. 7 V plotted as function of $\bar{\tau}^1$ for $\bar{n} = 5$, for $\bar{n} = 6$, and for $\bar{n} = 12$. The Y axis is logarithmic.

Statement (i) has a trivial explanation: As the polynomials of the ARX model have lower order than the polynomials of model no. 1, which was used to generate the dataset, the ARX model does not have enough poles nor enough zeros to perfectly explain the dynamics of model no. 1.

Statement (ii) can be explained by considering the ARX regression matrix. For simplicity, a deterministic SISO system were $n = n_A = n_B = 2$ is used for illustration. The time delay of the system is $\tau = 20$ samples. Hence, the system is on the form of Eq. (12).

$$y_k = -a_1 y_{k-1} - a_2 y_{k-2} + b_1 u_{k-20} + b_2 u_{k-21} \quad (12)$$

Now assume that a dataset is generated by applying an input signal, u, to the system of Eq. (12). Further assume that u is *not* a periodic signal. The generated dataset is now to be used for system identification of the system presented in Eq. (12). If the value of n is known, but the time delay, τ, is unknown, the ARX regression problem is on the form of Eq. (13). For simplicity, only one row of the regression matrix is shown.

$$y_k = \quad (13)$$

$$\begin{bmatrix} -y_{k-1} & -y_{k-2} & u_{k-\bar{\tau}} & u_{k-\bar{\tau}-1} \end{bmatrix} \begin{bmatrix} \bar{a}_1 \\ \bar{a}_2 \\ \bar{b}_1 \\ \bar{b}_2 \end{bmatrix}$$

In Eq. (13), $\bar{\tau}$ is a time delay specified to the ARX system identification algorithm. For this ARX regression problem to have no residual, the columns u_{k-20} and u_{k-21} must be included in the regression matrix. This is true if, and only if, $\bar{\tau}$ is chosen identical to the true time delay, i.e. 20 samples. For any $\bar{\tau} \neq 20$, there will be residuals. Hence, the true time delay can be identified by considering the residual for various values of $\bar{\tau}$.

This is identical to the curve representing $\bar{n} = 6$ shown in Fig. 7.

If also the value of n is unknown, an arbitrary $\bar{n} \geq n$ is chosen. Assume $\bar{n} = 4$ is chosen. The ARX regression problem is now on the form of Eq. (14).

$$y_k = \begin{bmatrix} -y_{k-1} & -y_{k-2} & -y_{k-3} & -y_{k-4} & u_{k-\bar{\tau}} & u_{k-\bar{\tau}-1} & u_{k-\bar{\tau}-2} & u_{k-\bar{\tau}-3} \end{bmatrix} \begin{bmatrix} \bar{a}_1 \\ \bar{a}_2 \\ \bar{a}_3 \\ \bar{a}_4 \\ \bar{b}_1 \\ \bar{b}_2 \\ \bar{b}_3 \\ \bar{b}_4 \end{bmatrix} \tag{14}$$

Also in the case of Eq. (14), the columns u_{k-20} and u_{k-21} must be included in the regression matrix to avoid residuals. However, in this case, this is achieved for three different values of $\bar{\tau}$:

$$\bar{\tau} = 20 \quad \Rightarrow \quad [u_{k-\bar{\tau}}, u_{k-\bar{\tau}-1}] = [u_{k-20}, u_{k-21}]$$

$$\bar{\tau} = 19 \quad \Rightarrow \quad [u_{k-\bar{\tau}-1}, u_{k-\bar{\tau}-2}] = [u_{k-20}, u_{k-21}]$$

$$\bar{\tau} = 18 \quad \Rightarrow \quad [u_{k-\bar{\tau}-2}, u_{k-\bar{\tau}-3}] = [u_{k-20}, u_{k-21}]$$

This is identical to the results presented in Fig. 7: The residual is zero for $\bar{\tau} \in [\tau - (\bar{n} - n), \tau]$. In the case of Eq. (14) the residual is zero for $\bar{\tau} \in [20 - (4 - 2), 20]$, i.e. $\bar{\tau} \in [18, 20]$. The highest value of $\bar{\tau}$ that gives no residual is the correct time delay.

Statement (ii) can be used to identify the value of τ^1: As $V = 0$ for $\bar{\tau}^1 \in [\tau^1 - (\bar{n} - n), \tau^1]$, τ^1 is given by the highest values of $\bar{\tau}^1$ giving $V = 0$. This value can be read from Fig. 7: For both $\bar{n} = 6$ and for $\bar{n} = 12$, the highest value of $\bar{\tau}$ giving $V = 0$ is $\bar{\tau} = 20$, which is the correct time delay. Statement (ii) can not be used for $\bar{n} = 5 < n$ because the statement assumes $\bar{n} \geq n$.

Statement (ii) can also be used to identify the value of n: Define $\bar{\tau}^1_{min}(\bar{n})$ as the lowest $\bar{\tau}^1$ giving $V = 0$. According to statement (ii), $\bar{\tau}^1_{min}(\bar{n})$ is given by $\bar{\tau}^1_{min}(\bar{n}) = \tau^1 - (\bar{n} - n)$. $\bar{\tau}^1_{min}(\bar{n})$ can be read from the plot shown in Fig. 7. Hence, as τ^1, $\bar{\tau}^1_{min}(\bar{n})$, and $\bar{n}$ are known, n can be calculated by $n = \bar{\tau}^1_{min}(\bar{n}) - \tau^1 + \bar{n}$.

For the curve representing $\bar{n} = 6$ in Fig. 7: $\tau^1 = 20$ (see two paragraphs above), $\bar{\tau}^1_{min}(\bar{n}) = 20$ (read from Fig. 7), and $\bar{n} = 6$ (specified). Hence, $n = \bar{\tau}^1_{min}(\bar{n}) - \tau^1 + \bar{n} = 20 - 20 + 6 = 6$, which is the correct value.

For the curve representing $\bar{n} = 12$ in Fig. 7: $\tau^1 = 20$ (see three paragraphs above), $\bar{\tau}^1_{min}(\bar{n}) = 14$ (read from Fig. 7), and $\bar{n} = 12$ (specified). Hence, $n = \bar{\tau}^1_{min}(\bar{n}) - \tau^1 + \bar{n} = 14 - 20 + 12 = 6$, which is the correct value.

In the derivation presented above, it has been assumed that $n_A = n_B$. The method presented above actually identifies n_B, i.e. if the assumption $n_A = n_B$ is violated, the identified value of n is equal to n_B and different from n_A. This can be seen from the explanation given by Eq. (12) to Eq. (14): The value of $\bar{n}_A$ does not influence for which value of $\bar{\tau}$ there is no residual. Of course $\bar{n}_A$ must be chosen so that $\bar{n}_A \geq n_A$, otherwise there will always be residuals regardless of the value of $\bar{\tau}$.

4.2 Time Delay Estimation on Industrial Data

The `delayest` command has been tested on a dataset from the copper refining process at Xstrata Nikkelverk. This process is briefly explained in Sec. 1. As the actual system order is unknown, `delayest` has been tested for four different values of $\bar{n}$: $\bar{n} = 3$, $\bar{n} = 5$, $\bar{n} = 10$, and $\bar{n} = 20$, where $\bar{n} = \bar{n}_A = \bar{n}_{B^1} = \bar{n}_{B^2} = \bar{n}_{B^3}$. For each of the $\bar{n}$ values, both the 3×1 case and the 1×1 case were tested. The time delays, τ^1, τ^2, and τ^3, were all specified to be within the interval $[15, 110]$.

In [7] it is provided rough time delay estimates based on process knowledge. In Fig. 8, the estimates of `delayest` are compared to the estimates of [7]. The figure shows that: (i) There are large variations within the estimates of `delayest`. This confirms that `delayest` is sensitive to the specified values of $\bar{n}$ and to whether the 3×1 approach or the 1×1 approach is used. (ii) There are in general large deviations between the estimates of [7] and the estimates of `delayest`.

5 Conclusions

Outliers in a dataset may cause innovation processes, which are significantly larger (absolute value) than usual. Hence, searching for samples having large innovation processes may be used as a method for outlier detection. In [2] it is proved that for linear time-invariant (LTI) systems the innovation process can be identified directly from input and output data, without relying on models. This method is therefore well suited for outlier detection. In [3] it is shown that the innovation process

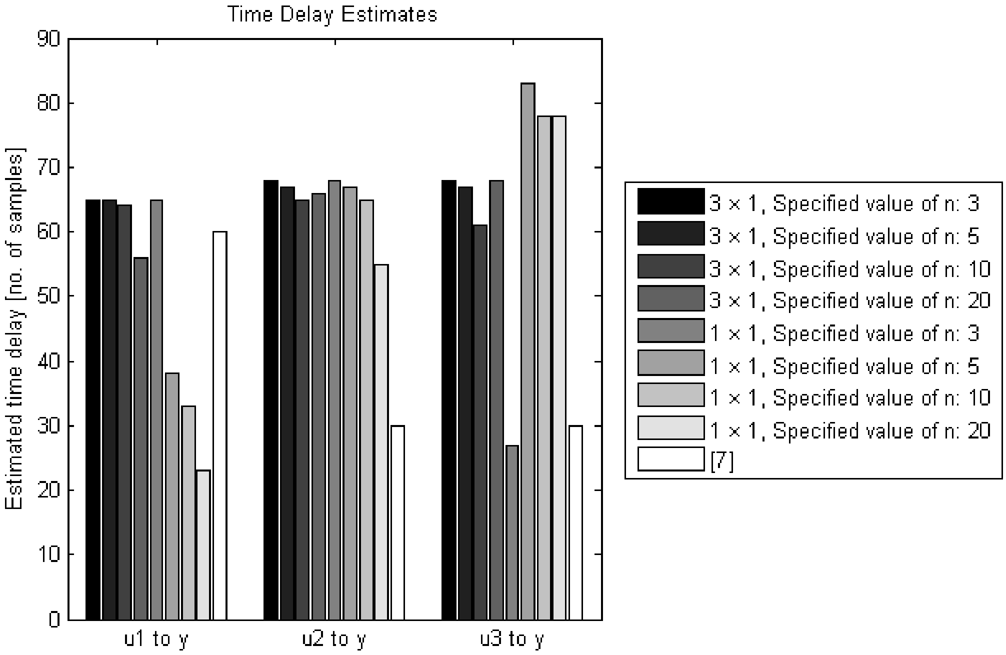

Fig. 8 Time delay estimates of `delayest` and of [7].

identified by the method of [2] is mathematically identical to the residual of an ARX identification of which $n_A = n_B = J$. It then follows that using the method of [2] for outlier detection is a special case of a method for outlier detection presented in [1, Example 14.1].

The method for outlier detection based on [2] has been tested on industrial data from Xstrata Nikkelverk, Kristiansand, Norway. The method detected three outliers that would not have been detected by plotting the raw data. The detected data are outliers because the derivative of the output changes unlikely fast, i.e. the second derivative of the output has unlikely large absolute values.

The MATLAB command `delayest` is used to estimate time delays between system inputs and system outputs in datasets. The command has been tested in a simulation study. The results from this study show that `delayest` is sensitive to: (i) The specified model order. (ii) Stochastic elements in the dataset. (iii) Whether all time delays are estimated during one execution of `delayest`, or whether only one time delay is estimated for each execution. The simulation study also indicates that `delayest` has a *systematic* error: The time delay estimates tend to decrease as the system order increases. In simulation of a 3 input, 1 output system, `delayest` estimated the time delays from the three inputs to the output exactly correct only when the following conditions were met: (i) The correct system order was specified. (ii) No stochastic elements were present. (iii) All three time delays were estimated during one execution of `delayest`.

`delayest` was also tested on industrial data from Xstrata Nikkelverk, Kristiansand, Norway. These tests confirmed that `delayest` is sensitive to: (i) The specified model order. (ii) Whether all time delays are estimated during one execution of `delayest`, or whether only one time delay is estimated for each execution. Whether `delayest` is sensitive to stochastic elements in the dataset could not be tested on the industrial data. In general, the estimates of `delayest` deviated significantly from estimates based on process knowledge.

In this paper, a suggestion to improvement of `delayest` is presented. This improvement makes `delayest` less sensitive for unknown system order. This improvement works only for deterministic LTI systems.

6 Acknowledgments

The authors are most grateful to Xstrata Nikkelverk, Kristiansand, Norway, for providing the datasets from the copper refining process and for allowing these data to be used in this paper.

7 References

[1] L. Ljung. *System Identification - Theory for the User, 2nd Edition.* Prentice-Hall, Upper Saddle River, New Jersey, 1999.

[2] D. Di Ruscio. Subspace system identification of the Kalman filter. *Modeling, Identification and Control*, 24:125–157, 2003.

[3] M. Komperød. *System Identification: Topics in Data Preprocessing and Model Realization, with Real-Life Applications from the Process Industry,*

Master Thesis. Telemark University College, Porsgrunn, Norway, 2008.

[4] GW. Nilsen. *Topics in Open and Closed Loop Subspace System Identification: Finite Data-Based Methods, Ph.D. Thesis*. Norwegian University of Science and Technology / Telemark University College, Trondheim / Porsgrunn, Norway, 2006.

[5] D. Di Ruscio. Subspace system identification of the Kalman filter: Open and closed loop systems. In *The 6th international conference on Computing, Communication and Control Technologies: CCCT 2008*, Orlando, Florida, June 29th - July 2rd 2008.

[6] The MathWorks, Inc. *The System Identification Toolbox*, 2008.

[7] TA. Hauge. E-mail communication, dated Mars 13, 2008.