# An open-source System Identification Package for multivariable processes

Giuseppe Armenise
*Department of Civil and Industrial Engineering*
*University of Pisa*
Pisa, Italy
giuseppe.armenise@outlook.com

Marco Vaccari
*Department of Civil and Industrial Engineering*
*University of Pisa*
Pisa, Italy
marco.vaccari@ing.unipi.it

Riccardo Bacci di Capaci
*Department of Civil and Industrial Engineering*
*University of Pisa*
Pisa, Italy
riccardo.bacci@ing.unipi.it

Gabriele Pannocchia
*Department of Civil and Industrial Engineering*
*University of Pisa*
Pisa, Italy
gabriele.pannocchia@unipi.it

***Abstract*—We present in this paper an open-source System Identification Package for PYthon (SIPPY[1]), which implements different methods to identify linear discrete-time multi-input multi-output systems, in input-output transfer function or state space form. For input-output transfer function models, identification is performed using least-squares regression (FIR and ARX models) or recursive least-squares (ARMAX model). For state space models, various subspace identification algorithms are implemented according to traditional methods (N4SID, MOESP, and CVA) and to parsimonious methods which enforce causal projections. When the model order is not known a priori, three different information criteria can help the user in the choice of the most appropriate order. Many identification and validation tests have been performed on simulation data collected both in open-loop and closed-loop mode. Results show effectiveness and computational efficiency of SIPPY also in comparison with state-of-art proprietary system identification software.**

***Index Terms*—System Identification, input-output models, state space models, subspace methods**

## I. Introduction

Advanced control techniques, i.e. those control algorithms specifically based on process models, are applied to different fields of research and technology, ranging from Aerospace and Automotive to Process Industries. Very often process models are not directly available or their representation derived from physical laws may be cumbersome for controller implementation due to nonlinearity or the presence of many unknown parameters. For such reasons, linear multi-variable models are used ubiquitously in applications, and the most common way of deriving them is to collect data of process inputs and outputs over specific time windows and to use suitable algorithms to obtain the process model parameters. The route that involves designing experiments, collecting data and building models is commonly referred to as *system identification*.

[1]SIPPY is freely available at GITHUB: *https://github.com/CPCLAB-UNIPI/SIPPY*

System identification algorithms can be classified into two main categories [9]: Prediction Error Methods (PEMs) and Subspace Identification Methods (SIMs). The former are aimed at identifying input-output transfer function models, whereas the latter have been developed to obtain state space models. Models identified using SIMs are often less accurate than those obtained from PEMs, but on the other hand, state space models offer a simpler parametrization for large multivariable systems and they are more convenient for model predictive controller (MPC) design.

Many identification software packages, which cover the above mentioned algorithms, are available from commercial vendors. The most famous and consolidated package is the System Identification Toolbox in MATLAB [11]. Other options are represented by the ISIAC software [22] and the NI LabVIEW System Identification Toolkit [6]. At present, there are available several open-source identification packages written on different programming languages, as the various MATLAB-based toolboxes: UNIT [12], CONTSID [4], CAPTAIN [24], and ITSIE [5]. Nevertheless, to the best of the authors' knowledge, there is no single package for Python to cover such a wide range of identification methods, as the one presented in this paper. Moreover, the possibility to identify multivariable systems is a further important feature which adds value to the present tool. Hence, the objective of this work is to provide an open-source package, developed in Python language [3], whose spread in scientific and technical environments is significantly increasing. It has to be noted that the present work is focused only on linear models, while nonlinear system identification is out of our current scope and it is implemented in other software, e.g. nonlinear ARX in [11] or Hammerstein-Wiener models in [12].

The remainder of the paper is organized as follows. In Section II different model structures with the various identification algorithms and options available for the user are described. Section III deals with several examples of numerical

simulation, while Section IV concerns with a comparison of the different algorithms. Finally, conclusions are drawn in Section V.

## II. Package description and usage

Both input-output models and state space models, identified with the proposed package, are described below. We assume to have collected input-output data for a general multivariable system with $m$ inputs and $l$ outputs, defined as follows:

$$\mathbf{u} = [u_0\ u_1\ u_2\ \dots\ u_{L-1}], \quad \mathbf{y} = [y_0\ y_1\ y_2\ \dots\ y_{L-1}] \quad (1)$$

where $L$ is the number of samples, $\mathbf{u} \in \mathbb{R}^{m\times L}$ is the input sequence, and $\mathbf{y} \in \mathbb{R}^{l\times L}$ is the output sequence, being $u_k = [u_k^{(1)},\ u_k^{(2)},\ \dots,\ u_k^{(m-1)},\ u_k^{(m)}]^T$ and $y_k = [y_k^{(1)},\ y_k^{(2)},\ \dots,\ y_k^{(l-1)},\ y_k^{(l)}]^T$ considered at the $k$-th sampling time with $k = 0, \dots, L-1$.

### A. Input-output models

Three input-output (I-O) models are considered in this work: FIR (Finite Impulse Response), ARX (AutoRegressive with eXogenous inputs) and ARMAX (AutoRegressive-Moving-Average with eXogenous inputs) models. They are described jointly since FIR and ARX models can be seen as particular cases of ARMAX models, although their identification can be done in a simpler dedicated way, as later explained.

The approach used to build multiple input - multiple output (MIMO) models is to write a series of multiple input - single output (MISO) relations, by identifying for each output $y^{(i)}$, with $i = 1, \dots, l$, the parameters of the following linear difference equation:

$$\begin{aligned} &a_0^{(i)} y_k^{(i)} + \dots + a_{n_a^{(i)}}^{(i)} y_{k-n_a^{(i)}}^{(i)} = b_{1,0}^{(i)} u_k^{(1)} + \dots + \\ &b_{1,n_{b1}^{(i)}}^{(i)} u_{k-n_{b1}^{(i)}}^{(1)} + \dots + b_{m,0}^{(i)} u_k^{(m)} + \dots + \\ &b_{m,n_{bm}^{(i)}}^{(i)} u_{k-n_{bm}^{(i)}}^{(m)} + e_k^{(i)} + c_1^{(i)} e_{k-1}^{(i)} + \dots + c_{n_c^{(i)}}^{(i)} e_{k-n_c^{(i)}}^{(i)} \end{aligned} \quad (2)$$

where $u_k^{(j)}$ is the $j$-th input, $y_k^{(i)}$ is the $i$-th output, $e_k^{(i)}$ represents the $i$-th disturbance; $n_a^{(i)}$ is the $i$-th output order, $n_{bj}^{(i)}$ is the $j$-th input order associated to the $i$-th output, $n_c^{(i)}$ is the $i$-th error order. We also assume $e_k^{(i)}$ is white noise.

*Assumption 1:* The following coefficients are fixed a priori:

$$a_0^{(i)} = 1,\ b_{j,0}^{(i)} = 0 \quad \forall\, i = 1, \dots, l,\ \forall\, j = 1, \dots, m \quad (3)$$

Using the backward shift operator $z^{-1}$ we can define:

$$A^{(i)}(z) = 1 + a_1^{(i)} z^{-1} + \dots + a_{n_a^{(i)}}^{(i)} z^{-n_a^{(i)}} \quad (4)$$

$$B_j^{(i)}(z) = b_{j,1}^{(i)} z^{-1} + \dots + b_{j,n_{bj}^{(i)}}^{(i)} z^{-n_{bj}^{(i)}} \quad (5)$$

$$C^{(i)}(z) = 1 + c_1^{(i)} z^{-1} + \dots + c_{n_c^{(i)}}^{(i)} z^{-n_c^{(i)}} \quad (6)$$

Hence, the system (2) can be rewritten as:

$$y_k^{(i)} = F^{(i)}(z, u_k, e_k) = \sum_{j=1}^{m} G_j^{(i)}(z) u_k^{(j)} + H^{(i)}(z) e_k^{(i)} \quad (7)$$

where:

$$G_j^{(i)}(z) = \frac{B_j^{(i)}(z)}{A^{(i)}(z)} \quad (8)$$

$$H^{(i)}(z) = \frac{C^{(i)}(z)}{A^{(i)}(z)} \quad (9)$$

As said before, the overall MIMO model is then built by grouping the various MISO relations (7) into a single transfer function matrix. Note that if $n_c = 0$, the ARMAX structure reduces to a ARX structure; if also $n_a = 0$, the finite impulse response (FIR) structure is obtained. The objective of system identification is to compute, for each output, $n_a^{(i)}$ output parameters $(a_1^{(i)}, \dots, a_{n_a^{(i)}}^{(i)})$, $m$ sets of $n_{bj}^{(i)}$ input parameters $(b_{j,1}^{(i)}, \dots, b_{j,n_{bj}^{(i)}}^{(i)})$, and $n_c^{(i)}$ error parameters $(c_1^{(i)}, \dots, c_{n_c^{(i)}}^{(i)})$.

Two identification algorithms have been implemented:

1) ARX procedure: for each output, the problem is reduced to a direct least-squares solution (in this case $n_c = 0$, i.e. $C^{(i)}(z) = 1\ \forall\, i$).
2) ARMAX procedure: for each output, first an ARX identification is performed; then, output residuals vector is computed, defined as $\hat{e}_k^{(i)} = y_k^{(i)} - \hat{y}_k^{(i)}$, being $\hat{y}_k^{(i)}$ the output of the estimated model; finally, model parameters are updated, by using a recursive least-squares method combined with a line-search algorithm. The procedure is iterative and stops when a minimum of the variance of each residuals vector is obtained.

The user has to specify the order $n_a^{(i)}$ for each output $y^{(i)}$, the input orders and, for the ARMAX case, also the error order for each $y^{(i)}$. In addition, the input time-delays $\theta^{(i)}$ for each transfer function $F^{(i)}$ could be specified, by modifying (5):

$$B_j^{(i)}(z) = b_{j,1}^{(i)} z^{-1-\theta_j^{(i)}} + \dots + b_{j,n_{bj}^{(i)}}^{(i)} z^{-n_{bj}^{(i)}-\theta_j^{(i)}} \quad (10)$$

where $\theta_j^{(i)}$ is the $j$-th input time-delay associated to the $i$-th output. Finally, for the ARMAX model, in order to ensure the procedure stops, a maximum number of iterations is enforced (by default, which the user can override).

Note that for the single input - single output (SISO) case, three information criteria are available to help the user in the choice of model orders. The information criteria are not implemented in the MIMO case, due to the large number of combinations of orders that should be tested, leading to excessive computational load, especially for the ARMAX algorithm. These criteria are explained in details further on in Section II-C.

### B. State space models

Subspace identification methods (SIMs) [19] identify state space models, that can be represented in the *process* form:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ y_k = Cx_k + Du_k + v_k \end{cases} \quad (11)$$

where $x_k \in \mathbb{R}^n$, $w_k \in \mathbb{R}^n$ and $v_k \in \mathbb{R}^l$ are the system state, state noise, and output noise, respectively; $A \in \mathbb{R}^{n\times n}$,

$B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{l\times n}$, $D \in \mathbb{R}^{l\times m}$ are the system matrices. Two other common ways to represent MIMO systems are the so-called *innovation* form:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + Ke_k \\ y_k = Cx_k + Du_k + e_k \end{cases} \tag{12}$$

with $K \in \mathbb{R}^{n\times l}$, usually referred to as the Kalman predictor gain, and also the *predictor* form:

$$\begin{cases} x_{k+1} = A_K x_k + B_K u_k + Ky_k \\ y_k = Cx_k + Du_k + e_k \end{cases} \tag{13}$$

where the following relations hold:

$$A_K = A - KC, \quad B_K = B - KD \tag{14}$$

The objective of SIMs is to estimate the system matrices $(A, B, C, D, K)$ from the collected data.

The implemented SIMs can be classified into two categories:

1) *traditional* methods: N4SID, MOESP and CVA;
2) *parsimonious* methods: PARSIM-P, PARSIM-S and PARSIM-K.

The three *traditional* methods, N4SID [13], MOESP [23] and CVA [8], have been later grouped into a unifying theorem as discussed in [14]. Based on this approach, the three methods can be seen as a singular value decomposition (SVD) of a weighted matrix. The algorithm implemented in the package here presented is based on the so-called "*combined algorithm 2*" proposed in [15]. We shall use $f$ as the future horizon and $p$ as the past horizon, with the following notation:

$$\begin{aligned} N &= L - f - p + 1, \\ X_f &= [x_f \; x_{f+1} \; \dots \; x_{f+N-1}] \in \mathbb{R}^{n\times N}, \\ Y_f &= [y_f \; y_{f+1} \; \dots \; y_{f+N-1}] \in \mathbb{R}^{l\times N}, \\ U_f &= [u_f \; u_{f+1} \; \dots \; u_{f+N-1}] \in \mathbb{R}^{m\times N} \end{aligned} \tag{15}$$

The algorithm proposed in [15] determines two state sequences ($X_f$ and $X_{f+1}$), and then system matrices are obtained by solving, in a least-squares sense, the following system:

$$\begin{pmatrix} X_{f+1} \\ Y_f \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} X_f \\ U_f \end{pmatrix} \tag{16}$$

In the implemented procedure, instead, after the evaluation of the state sequence $X_f$, system matrices are obtained by solving the following system in a least-squares sense:

$$\begin{pmatrix} X_+ \\ Y_- \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} X_- \\ U_- \end{pmatrix} \tag{17}$$

where $X_+$ is $X_f$ without the first column, $X_-, Y_-$ and $U_-$ are $X_f, Y_f$ and $U_f$ without the last column, respectively. According to extensive simulation studies, using (17) instead of (16) has resulted in superior performance in terms of the variance of the residuals. Moreover, the implemented algorithm has a lower computation load given that only a single state sequence needs to be identified. For *traditional* methods, the user can set only the future horizon $f$, as the past horizon $p$ has to be equal to $f$; by default, $f = 20$.

*Parsimonious* methods remove non-causal terms in the linear projections by enforcing causal models. The oldest methods are PARSIM-S [17] and PARSIM-P [18]. Nevertheless, these algorithms are based on the assumption that the input is not correlated with the output noise, hence they may fail for data collected in closed-loop mode. For this reason, another parsimonious method specifically consistent with closed-loop data has been implemented: the PARSIM-K algorithm [16]. For *parsimonious* methods, there is the possibility to set both the past and future horizon; by default, $f = p = 20$.

For both *traditional* and *parsimonious* algorithms, the model order $n$ has to be specified by the user and it cannot be larger than the future horizon. A truncated SVD retaining up to the $n-$th singular value is performed. Note that SVD is performed on a matrix depending on the selected algorithm. Alternatively, one can specify a threshold value with the maximum order allowed; otherwise, one of the information criteria described in Section II-C can be employed. If a threshold value is selected, $n$ is chosen as the largest integer $n \leq f$ such that the ratio between the $n$-th singular value and the largest one is greater than the selected threshold. By default, this threshold value is equal to $0.1$, while the maximum order is set equal to $f$.

Moreover, it has to be noted that the user can also identify a system with a direct input-output relation, i.e. matrix $D$ is not zero. By default, $D$ is filled with zeros. In addition, for *traditional* methods, the user can impose the stability of matrix $A$, i.e. eigenvalues are located within the unit circle in the complex plane. In this case, once the system is identified, the matrix $A$ is analyzed with a stability test. If the result is negative, a guaranteed stable matrix $A$ is then obtained on the basis of an extended observability matrix [10], and matrix $B$ is recomputed accordingly in the least-squares sense. Note that forcing the stability of matrix $A$ may lead to significant errors in the identified model.

### C. The information criteria

As introduced in the previous sections, a suitable method to help the user choosing the model orders is an information criterion (IC). For both input-output and state space models, three information criteria have been implemented: Akaike Information Criterion (AIC) [1], correction of Akaike Information Criterion (AICc) [21], and Bayesian Information Criterion (BIC) [20]. The basic idea of an information criterion is to find the most appropriate model order to balance between a complex model, which implies higher orders and longer computational times, and a simpler model, which allows the controller to compute a faster response and to be more robust to errors.

Each information criterion finds the minimum of a specific function including a likelihood function and a penalty term on the number of model parameters to be identified. By denoting the maximized likelihood function as $\hat{\mathcal{L}}$ and the number of independent model parameters as $K_{IC}$, the considered IC are

formulated as follows:

$$AIC = -2\log(\hat{\mathcal{L}}) + 2K_{IC} \tag{18}$$

$$AICc = AIC + \frac{2K_{IC}(K_{IC}+1)}{N_{IC}-K_{IC}-1} \tag{19}$$

$$BIC = -2\log(\hat{\mathcal{L}}) + K_{IC}\log(N_{IC}) \tag{20}$$

where $N_{IC}$ is the number of data points used to compute the variance of the model residuals $\hat{\sigma}^2$. The likelihood function can be written as [2]:

$$\log\hat{\mathcal{L}} = -\frac{N_{IC}}{2}\log\hat{\sigma}^2 \tag{21}$$

where the variance of the model residuals for multi-output cases is:

$$\hat{\sigma}^2 = \frac{\sum_{k=1}^{N_{IC}} \hat{\epsilon}_k^T \hat{\epsilon}_k}{l\, N_{IC}} \tag{22}$$

being $\hat{\epsilon}_k = y_k - \hat{y}_k \in \mathbb{R}^l$ the output residuals at the $k$-th sampling time. By substituting (21) in (18) and (20), the following functions are obtained:

$$AIC = N_{IC}\log(\hat{\sigma}^2) + 2K_{IC} \tag{23}$$

$$BIC = N_{IC}\log(\hat{\sigma}^2) + K_{IC}\log(N_{IC}) \tag{24}$$

In a SISO ARX model $K_{IC} = n_a + n_b$, while in SISO ARMAX $K_{IC} = n_a + n_b + n_c$. For the state space models, in the case of *parsimonious* methods, the number of independent parameters is $K_{IC} = 2\,n \times l + n \times m$ (see Corollary 4A.1 in [9]). As a matter of fact, the matrices $A$, $B$, and the Kalman filter gain $K$ have $n \times l$, $n \times m$, and $n \times l$ independent parameters, respectively, while the matrix $C$ has $l \times n$ non-independent parameters. For *traditional* SIMs, $K_{IC} = n \times l + n \times m$, since only $A$, $B$, and $C$ matrices are identified in the process form, while $K$ is obtained by solving the Riccati equation. Finally, if the matrix $D$ is required, other $l \times m$ independent parameters have to be estimated (see section 1.3 in [7]).

The "best" model order is obtained by the information criterion finding the minimum of the corresponding function. The comparison between models is made by using the same identification algorithm, i.e. once chosen the algorithm, the IC selects the "best" order by comparing identified models with different orders.

## III. Simulation examples

In order to verify the performance of the implemented identification package, a large set of tests has been performed; only few of them are reported below for the sake of brevity. The input and output data are generated in Python by using the package `control 0.7.0`.

### A. Example 1: input-output models

The plant data are here generated by using the MIMO ARMAX system ($l = 3, m = 4$) showed in Table I. The $l$ orders for the output and the error, $l \times m$ input orders and time-delays are:

$$n_a = [3\ 1\ 2],\ n_b = \begin{bmatrix} 2 & 1 & 3 & 2 \\ 3 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 \end{bmatrix},\ n_c = [2\ 0\ 3],\ \Theta = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}$$

Four independent pseudo-random binary sequences (PRBS) with a switch probability equal to $3\%$ are used as input to the system. Corresponding ranges of PRBSs from input 1 to input 4 are $[-0.3, 0.1]$, $[-1.0, 1.0]$, $[2.3, 5.7]$, and $[8.0, 11.5]$. ARX and ARMAX models are tested, by assuming system orders and time-delays as known. Five levels of white noise $e^{(i)}$ with different variances are considered: $\sigma_1^2 = [25, 50, 0.5]$, $\sigma_2^2 = [50, 100, 1]$, $\sigma_3^2 = [100, 200, 2]$, $\sigma_4^2 = [200, 400, 4]$, and $\sigma_5^2 = [400, 800, 8]$. For each noise level, a set of Monte Carlo simulations with $L = 400$ data points are considered: 500 simulations for the identification stage and 500 for the validation stage. Identification performance are evaluated by using the explained variance ($EV$), defined as:

$$EV^{(i)} = 1 - \frac{\sum_k (\hat{\epsilon}_k^{(i)})^2}{\sum_k (y_k^{(i)} - \overline{y}^{(i)})^2} \tag{25}$$

where $\overline{y}^{(i)}$ is the mean value of the $i$-th output. When a model returns $EV^{(i)} < 0$ for an output, the explained variance is considered equal to zero. The results, in terms of average explained variance $\overline{EV}$, for the identification and the validation data sets, are shown in Table II.

The ARX algorithm shows superior performance, but it has to be observed that the ARMAX algorithm returns a negative $EV$ for several simulations, which make values of $\overline{EV}$ decrease. Nevertheless, the single $EV$ returned by ARMAX procedure is usually greater than the one obtained by ARX, whether the identification is successful.

### B. Example 2: state space models

To generate the plant data, the following MIMO system ($l = 2, m = 2, n = 3$) is used:

$$\begin{aligned} A &= \begin{bmatrix} 0.95 & 0 & 0 \\ 0 & -0.97 & 0 \\ 0 & 0 & 0.73 \end{bmatrix},\ B = \begin{bmatrix} 0.043 & 0.196 \\ 0.906 & 0.406 \\ 0.761 & 0.095 \end{bmatrix}, \\ C &= \begin{bmatrix} 0.874 & -0.97 & 0.781 \\ -0.885 & -0.909 & -0.326 \end{bmatrix},\ D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \\ K &= \begin{bmatrix} 0.002 & -0.01 \\ -0.008 & 0.05 \\ -0.093 & -0.024 \end{bmatrix} \end{aligned} \tag{26}$$

Four different levels of white noise, with the same variance for both outputs, are considered: $\sigma_1^2 = 10^{-4}$, $\sigma_2^2 = 10^{-3}$, $\sigma_3^2 = 10^{-2}$, and $\sigma_4^2 = 10^{-1}$. The closed-loop input is calculated by using the following proportional control law:

$$u_k = K_c(r_k - y_k), \quad K_c = \begin{bmatrix} -0.0316 & -0.1400 \\ -0.0954 & -0.3578 \end{bmatrix} \tag{27}$$

The reference vector $r$ is comprised by two independent PRBSs with a switch probability of $2\%$, in a range $[-0.5, 0.5]$ for both the outputs. On the closed-loop data collected $(\mathbf{u}, \mathbf{y})$, with known model order, two identification methods are tested: N4SID and PARSIM-K. We set $f = 20$, $p = 20$.

For each noise level, a set of Monte Carlo simulations with $L = 500$ are considered: 500 simulations for the identification stage and 500 for the validation stage. The performance are evaluated by using the explained variance (25). When a model

TABLE I
MIMO ARMAX SYSTEM FOR EXAMPLE 1

| | Output 1 | Output 2 | Output 3 |
|---|---|---|---|
| Input 1 | $g_{11} = \frac{4z^3+3.3z^2}{z^5-0.3z^4-0.25z^3-0.021z^2}$ | $g_{21} = \frac{-85z^2-57.5z-27.7}{z^4-0.4z^3}$ | $g_{31} = \frac{0.2z^3}{z^4-0.1z^3-0.3z^2}$ |
| Input 2 | $g_{12} = \frac{10z^2}{z^5-0.3z^4-0.25z^3-0.021z^2}$ | $g_{22} = \frac{71z+12.3}{z^4-0.4z^3}$ | $g_{32} = \frac{0.821z^2+0.432z}{z^4-0.1z^3-0.3z^2}$ |
| Input 3 | $g_{13} = \frac{7z^2+5.5z+2.2}{z^5-0.3z^4-0.25z^3-0.021z^2}$ | $g_{23} = \frac{-0.1z^3}{z^4-0.4z^3}$ | $g_{33} = \frac{0.1z^3}{z^4-0.1z^3-0.3z^2}$ |
| Input 4 | $g_{14} = \frac{-0.9z^3-0.11z^2}{z^5-0.3z^4-0.25z^3-0.021z^2}$ | $g_{24} = \frac{0.994z^3}{z^4-0.4z^3}$ | $g_{34} = \frac{0.891z+0.223}{z^4-0.1z^3-0.3z^2}$ |
| Error model | $h_1 = \frac{z^5+0.85z^4+0.32z^3}{z^5-0.3z^4-0.25z^3-0.021z^2}$ | $h_2 = \frac{z^4}{z^4-0.4z^3}$ | $h_3 = \frac{z^4+0.7z^3+0.485z^2+0.22z}{z^4-0.1z^3-0.3z^2}$ |

TABLE II
EXAMPLE 1: AVERAGE EXPLAINED VARIANCE FOR IDENTIFICATION (ID) AND VALIDATION (VA) DATA SETS

| | $\sigma_1^2$ | $\sigma_2^2$ | $\sigma_3^2$ | $\sigma_4^2$ | $\sigma_5^2$ |
|---|---|---|---|---|---|
| ARMAX (ID) | 0.9423 | 0.9585 | 0.9513 | 0.9386 | 0.9293 |
| ARX (ID) | 0.9955 | 0.9918 | 0.9858 | 0.9769 | 0.9672 |
| ARMAX (VA) | 0.9417 | 0.9565 | 0.9474 | 0.9341 | 0.9230 |
| ARX (VA) | 0.9940 | 0.9890 | 0.9819 | 0.9712 | 0.9622 |

TABLE III
EXAMPLE 2: AVERAGE EXPLAINED VARIANCE FOR IDENTIFICATION (ID) AND VALIDATION (VA) DATA SETS

| | $\sigma_1^2$ | $\sigma_2^2$ | $\sigma_3^2$ | $\sigma_4^2$ |
|---|---|---|---|---|
| N4SID (ID) | 0.9988 | 0.9859 | 0.8793 | 0.4256 |
| PARSIM-K (ID) | 0.9989 | 0.9886 | 0.8825 | 0.4089 |
| N4SID (VA) | 0.9988 | 0.9871 | 0.8766 | 0.4181 |
| PARSIM-K (VA) | 0.9988 | 0.9881 | 0.8758 | 0.3972 |

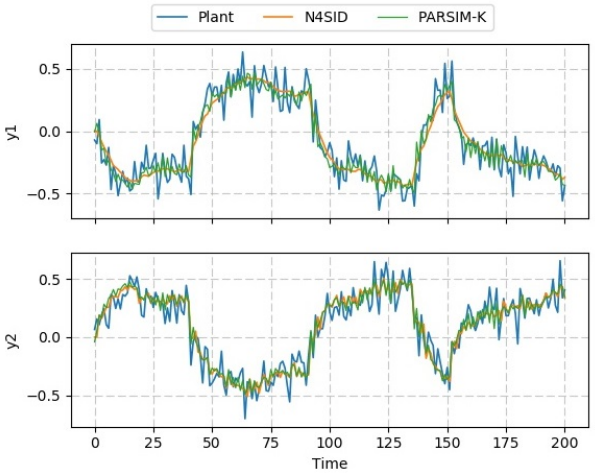

Fig. 1. Example 2, validation stage.

returns $EV < 0$ for an output, the variance is considered null. The overall results for the identification and the validation data sets are shown in Table III. It can be observed that the PARSIM-K algorithm shows better performance in the identification stage, but it shows a slightly lower robustness to noise in the validation stage. A generic simulation example, with noise variance $\sigma_3^2$, has been chosen. For the sake of brevity, only time trends of validation stage are reported in Figure 1. It has to be noted that, despite being a hard case of closed-loop data, both PARSIM-K and N4SID give excellent performance, by properly fitting the original output.

The three considered information criteria are tested on both algorithms, by using the same set of simulations of the identification stage. The number of times that the IC select the correct order are counted, being $n_{IC} \in [1, 5]$ the range for system order. The overall results are shown in Table IV. It can be observed that when the measurements are affected by low level of noise, the ICs show excellent performance; in particular, the BIC selects always the right order for $\sigma_1^2$ and $\sigma_2^2$ when applied to PARSIM-K algorithm. For higher noise variances, performance decreases for all three information criteria, but BIC proves to be much more sensitive.

TABLE IV
NUMBERS OF CORRECT SELECTION OF MODEL ORDER OUT OF 500 MONTE CARLO SIMULATIONS

| | PARSIM-K | | | | N4SID | | | |
|---|---|---|---|---|---|---|---|---|
| | $\sigma_1^2$ | $\sigma_2^2$ | $\sigma_3^2$ | $\sigma_4^2$ | $\sigma_1^2$ | $\sigma_2^2$ | $\sigma_3^2$ | $\sigma_4^2$ |
| AIC | 495 | 497 | 449 | 126 | 375 | 258 | 35 | 277 |
| AICc | 497 | 498 | 450 | 108 | 393 | 267 | 42 | 290 |
| BIC | 500 | 500 | 257 | 6 | 486 | 397 | 59 | 352 |

## IV. COMPARISON WITH MATLAB

An extensive comparisons with the System Identification Toolbox in MATLAB R2017a [11] is here proposed. The various computational times ($T_c$) are reported, being performed on an Intel® Core™i7-7500U Processor with Windows 10 Home as operating system.

### A. Comparison 1

For the sake of brevity, the same set of Monte Carlo simulations of Table II, with $\sigma_5^2$ as variance level, is here considered. The proposed ARX and ARMAX algorithms are compared with the `armax` function of MATLAB Toolbox. It has to be observed that the MATLAB function has higher identification performance ($\overline{EV} = 0.9999$), since several efficient line search algorithms, e.g. Gauss-Newton and Levenberg-Marquardt methods, are available (see [9, Ch. 10]). The `SearchMethod` option is here set as `'auto'` (the default option), i.e. a combination of the available algorithms is tried in sequence at each iteration. However, the MATLAB function needs much more time to perform the various identifications

TABLE V
COMPARISON 2: PERFORMANCE OF THE VARIOUS SUBSPACE IDENTIFICATION ALGORITHMS.

| | $\overline{EV}$ | # of unstable system | $T_c[s]$ |
|---|---|---|---|
| 'SSARX' MATLAB | 0.8283 | 48 | 90 |
| 'MOESP' MATLAB | 0.8622 | 11 | 87 |
| 'CVA' MATLAB | 0.8706 | 10 | 87 |
| N4SID | 0.8793 | 1 | 13 |
| MOESP | 0.8785 | 1 | 13 |
| CVA | 0.8817 | 2 | 15 |
| PARSIM-K | 0.8825 | 13 | 49 |
| PARSIM-S | 0.7281 | 119 | 36 |
| PARSIM-P | 0.8914 | 2 | 90 |

($T_c = 670s$), while the implemented algorithms, both ARX and ARMAX model, show good results and save high computational times: $T_c = 4s$ and $42s$, respectively.

### B. Comparison 2

A comparison with the `n4sid` function of MATLAB Toolbox is here presented. The MIMO state space system of (26) is studied. The same set of Monte Carlo simulations of Table III with $\sigma_3^2$ as noise level is considered. Matrix $D$ is not identified, future and past horizons are equal to 20, and the order of the identified systems is known and hence set to $n = 3$. As an index of the consistency of the various methods, the number of unstable identified systems is computed. The overall results are shown in Table V. It is worth noting that the MATLAB functions have lower global performance, both in terms of $\overline{EV}$ and computational times. Among the proposed algorithms, both parsimonious and traditional methods show good results; in particular, all three traditional approaches are very efficient, and, among parsimonious methods, PARSIM-K allows the best balance between accuracy and velocity. Therefore, the proposed Python package can yield comparable results with the MATLAB toolbox both in terms of identification accuracy and computational times, but with the great advantage of being fully open-source.

## V. CONCLUSIONS

In this paper an open-source package for Python with various system identification methods has been presented. The most relevant user options in terms of identification algorithms and parameters settings to search for the best model have been analyzed. The underlying code is also intended to be quite simple to be used by beginners with default settings.

Two main structures of input-output models – ARX and ARMAX models – have been implemented, both for SISO systems, for which the information criteria are available, and for MIMO systems. For the state space model structure, six different algorithms are available: three use a standard approach (N4SID, MOESP, and CVA), the others enforce causal models (PARSIM-S, PARSIM-P, PARSIM-K). When a model order is not known a priori, three different information criteria can help the user in the choice of a suitable order.

The various methods show good results, as reported in the extensive numerical examples. The state-of-art MATLAB toolbox [11] has been taken as reference to test the Python package performance and accuracy. All the methods produce results comparable with the MATLAB counterpart underling a good efficiency in terms of computational times; in particular, SIMs algorithms returns more accurate and reliable model estimates than input-output ones.

## REFERENCES


[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, Dec 1974.

[2] K. P. Burnham and D. R. Anderson. Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research*, 33(2):261–304, 2004.

[3] Python Software Foundation. Python 2.7, `https://www.python.org/`, 2018.

[4] H. Garnier, M. Gilson, V. Laurain, and B. Ni. Developments for the CONTSID toolbox. *IFAC Proceedings Volumes*, 45(16):1553 – 1558, 2012. 16th IFAC Symposium on System Identification.

[5] J.L. Guzmn, D.E. Rivera, S. Dormido, and M. Berenguel. An interactive software tool for system identification. *Advances in Engineering Software*, 45(1):115 – 123, 2012.

[6] National Instruments. NI LabVIEW System Identification Toolkit, `http://sine.ni.com`, 2018.

[7] T. Katayama. *Subspace methods for system identification*. Springer, 2005.

[8] W. E. Larimore. Canonical variate analysis in identification, filtering, and adaptive control. In *29th IEEE Conference on Decision and Control*, pages 596–604 vol.2, 1990.

[9] L. Ljung. *System Identification: Theory for the User*. PTR Prentice Hall, Upper Saddle River, New Jersey 07458, 1999.

[10] J.M. Maciejowski. Guaranteed stability with subspace methods. *Systems & Control Letters*, 26(2):153–156, 1995.

[11] MathWorks. MATLAB System Identification Toolbox, `https://it.mathworks.com`, 2018.

[12] B. Ninness, A. Wills, and A. Mills. UNIT: A freely available system identification toolbox. *Control Engineering Practice*, 21(5):631–644, 2013.

[13] P. Van Overschee and B. De Moor. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, 1994.

[14] P. Van Overschee and B. De Moor. A unifying theorem for three subspace system identification algorithms. *Automatica*, 31(12):1853–1864, 1995.

[15] P. Van Overschee and B. De Moor. *Subspace identification for linear systems*. Kluwer Academic Publishers, 1996.

[16] G. Pannocchia and M. Calosi. A predictor form PARSIMonious algorithm for closed-loop subspace identification. *Journal of Process Control*, 20(4):517–524, 2010.

[17] S. J. Qin, W. Lin, and L. Ljung. A novel subspace identification approach with enforced causal models. *Automatica*, 41(12):2043–2053, 2005.

[18] S. J. Qin and L. Ljung. Parallel QR implementation of subspace identification with parsimonious models. *IFAC Proceedings Volumes*, 36(16):1591–1596, 2003. 13th IFAC Symposium on System Identification (SYSID 2003), Rotterdam, The Netherlands, 27-29 August, 2003.

[19] S.J. Qin. An overview of subspace identification. *Computers & Chemical Engineering*, 30(10):1502–1513, 2006.

[20] G. Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 03 1978.

[21] N. Sugiura. Further analysts of the data by Akaike' s information criterion and the finite corrections. *Communications in Statistics - Theory and Methods*, 7(1):13–26, 1978.

[22] P. Tona and J.M. Bader. Efficient system identification for model predictive control with the ISIAC software. In *Informatics in Control, Automation and Robotics I*, pages 225–232, Dordrecht, 2006. Springer Netherlands.

[23] M. Verhaegen and P. Dewilde. Subspace model identification part 1. The output-error state-space model identification class of algorithms. *International Journal of Control*, 56(5):1187–1210, 1992.

[24] P.C. Young and C.J. Taylor. Recent developments in the CAPTAIN toolbox for Matlab. *IFAC Proceedings Volumes*, 45(16):1838 – 1843, 2012. 16th IFAC Symposium on System Identification.