Output Error Methods for Robot Identification

Mathieu Brunot, Alexandre Janot, Francisco Carrillo, Joono Cheong, Jean-Philippe Noël

Output error methods for robot identification

Mathieu Brunot
ONERA
Toulouse, France
Email: Mathieu.Brunot@onera.fr

Alexandre Janot
ONERA
Toulouse, France
Email: Alexandre.Janot@onera.fr

Francisco Carrillo
ENIT
Tarbes, France
Email: Francisco.Carrillo@enit.fr

Joono Cheong
Korea University
Sejong-City, Republic of Korea
Email: jncheong@korea.ac.kr

Jean-Philippe Noël
Aerospace and Mechanical Engineering Department
ULiège, Belgium
Email: jp.noel@uliege.be

Industrial robot identification is usually based on the Inverse Dynamic Identification Model (IDIM) that comes from the Newton's laws and has the advantage of being linear with respect to the parameters. Building the IDIM from the measurement signals allows the use of linear regression techniques like the Least-Squares (LS) or the Instrumental Variable (IV) for instance. Nonetheless, this involves a careful preprocessing to deal with sensor noise. An alternative in system identification is to consider an output error approach where the model's parameters are iteratively tuned in order to match the simulated models output and the measured systems output. This paper proposes an extensive comparison of three different output error approaches in the context of robot identification. One of the main outcomes of this work is to show that choosing the input torque as target identification signal instead of the output position may lead to a gain in robustness versus modeling errors and noise and in computational time. Theoretical developments are illustrated on a 6-degree-of-freedom rigid robot.

1 Introduction

During decades, Least-Squares (LS) optimization and estimation of the Inverse Dynamic Identification Model (IDIM) have been the two key elements of the most common method used for industrial robot identification: see e.g. [1]. With the IDIM, the input torque is expressed as a linear function of the physical parameters: see e.g. [2]. Nevertheless, it is not always robust to the measurement noise correlation that arises from the closed-loop structure required for robot operation. To overcome this issue, in [3], the authors suggested an Instrumental Variable (IV) optimization adapted to robot systems that was recently improved [4].

Both the IDIM-LS and the IDIM-IV methods require the construction of the IDIM from the measured signals. According to the authors of [5], they belong to the *equation error* class, also denoted as "using explicit mathematical relations" [6]. The idea of this class is to obtain an overdetermined set of equations linear with respect to the parameters that can be solved with a linear regression technique. An alternative class is the *output error* one, also denoted as "using a model-adjustment technique" [6]. With this class, the parameters are tuned in such a way that the model's output fits the measured signal. The tuning of the parameters is usually performed thanks to nonlinear optimization algorithms, whereas the prediction of the model's output can be simulated by solving the differential equations modeling the system.

The drawback of the *equation error* methods is that, if all the signals are not available to the user, they must be estimated prior to the identification with a specific preprocessing [7]. For rigid industrial robots, only the joint position is sensed and must therefore be differentiated to obtain the joint velocity and acceleration see e.g. [8–11]. This pre-processing can be complex and/or a potential source of error. This is why, in [1], the authors introduced an *output error* method called Direct and Inverse Dynamic Identification Models (DIDIM), based on the torque linearity with respect to the parameters and optimized with a Gauss-Newton algorithm.

Because the *output error* class can deal with outputs nonlinear with respect to the parameters, one may wonder if it would be interesting to consider the measured position as the identification signal as usually done in the automatic

control community see e.g. [12–14] . The resultant question is: would it be possible to consider another identification signal like the output position to improve the robustness of the estimation? Furthermore, since those methods rely on the simulation of the closed-loop system, an underlying issue is the robustness of those simulations. In other words, the question is to what extent those methods can reject modeling errors or a too large noise.

The aim of this paper is therefore to evaluate the robustness of *output error* methods for robot identification which has attracted some renewed interest over the past few years [15–20] among others. In pursuing this goal, this work shows that it is more interesting to consider the input signal rather than the output one due to the larger sensitivity. Furthermore, if the torque linearity with respect to the parameters is insured, the designer should favor the DIDIM method to save computation time. This paper also illustrates the limit of the underlying white noise assumption of the *equation error* methods in the case of rigid robots.

This paper is organized as follows. Section 2 is dedicated to the modeling of rigid industrial robots. Section 3 reviews the identification methodologies. The comparison between those methodologies is undertaken in Section 4 and validated with experimental results in Section 5. Finally, Section 6 provides concluding remarks.

Notations

General notations

a, b, c	=	Scalars
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$	=	Vectors
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$	=	Matrices

Conventions

$\boldsymbol{A}^{-1}$	=	Inverse of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{T}$	=	Transpose of matrix $\boldsymbol{A}$
$\|\boldsymbol{A}\|_2$	=	Euclidean norm of matrix $\boldsymbol{A}$
$\widehat{\boldsymbol{x}}$	=	Estimated vector
x_{k_i}	=	i^{th} component of vector $\boldsymbol{x}_k$
$\frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}}$	=	Jacobian matrix of function $\boldsymbol{g}$

2 Robot Modeling

2.1 Dynamic Models

If a robot with n moving links is considered, the vector $\boldsymbol{\tau}(t)$ contains the inputs of those links, which are the applied forces or torques. The signals $\boldsymbol{q}(t)$, $\dot{\boldsymbol{q}}(t)$ and $\ddot{\boldsymbol{q}}(t)$ are respectively the $(n \times 1)$ vectors of generalized joint positions, velocities and accelerations, [2] . With respect to the Newton's second law it comes out:

$$\boldsymbol{M}(\boldsymbol{q}(t))\,\ddot{\boldsymbol{q}}(t) = \boldsymbol{\tau}(t) - \boldsymbol{N}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) \tag{1}$$

where, $\boldsymbol{M}(\boldsymbol{q}(t))$ is the $(n \times n)$ inertia matrix of the robot, and $\boldsymbol{N}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))$ is the $(n \times 1)$ vector modeling the disturbances or perturbations. Those perturbations contain the friction forces, gravity, centrifugal and Coriolis effects. The mathematical expressions show that those disturbances are linear in the parameters, but not in the states see [2]. Therefore, it appears to be very convenient for the identification to consider the Inverse Dynamic Model (IDM). The IDM is described by $\boldsymbol{\tau}_{idm}(t) = \boldsymbol{\phi}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t))\,\boldsymbol{\theta}$, where: the input torque is the dependent (or observation) variable; $\boldsymbol{\phi}$ is the $(n \times b)$ matrix of regressors (or independent variables); $\boldsymbol{\theta}$ is the $(b \times 1)$ vector of base parameters to be estimated. Because of perturbations coming from measurement noise and modeling errors, the actual torque $\boldsymbol{\tau}$ differs from $\boldsymbol{\tau}_{idm}$ by an error $\boldsymbol{\nu}$. The Inverse Dynamic Identification Model (IDIM) is given by

$$\boldsymbol{\tau}(t) = \boldsymbol{\tau}_{idm}(t) + \boldsymbol{\nu}(t) = \boldsymbol{\phi}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t))\,\boldsymbol{\theta} + \boldsymbol{\nu}(t). \tag{2}$$

The last alternative is the Direct Dynamic Model (DDM) that expresses the joint accelerations as a function of the torques, the joint positions and velocities:

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{M}(\boldsymbol{q}(t))^{-1}\left(\boldsymbol{\tau}_{idm}(t) - \boldsymbol{N}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\right). \tag{3}$$

The DDM is not used directly for the identification because the joint accelerations are nonlinear with respect to the parameters and so they are more difficult to estimate. Nonetheless, it is more convenient for simulation purposes and is used as such in section 3.2.

2.2 Control Law

Due to their double integrator behavior, robots must be operated in closed-loop. In most cases, the control laws are simple Proportional Derivative (PD), Proportional Integral Derivative (PID), or computed torque and passive control [2, see Chapter 14]. In this paper, the controller is assumed to be linear; that each link is controlled separately from the others; and that there is one position sensor (i.e. an encoder or a resolver) for each link. According to [2], this is a typical configuration for an industrial robot and the integral action is usually weak, or even deactivated when the position error is too small, in order to avoid oscillations due to the Coulomb friction.

Because there is one position sensor and one motor per link, there is usually one controller C_j for link j defined by

$$\nu_{\tau_j}(t) = C_j(p)(q_{r_j}(t) - q_{m_j}(t)), \tag{4}$$

where $p = d/dt$ is the differential operator, ν_{τ_j}, q_{r_j} and q_{m_j} are respectively the control signal, the reference trajectory and the measured position of link j. The robot controller is given by the transfer matrix $\boldsymbol{C} = diag(C_1, ..C_n)$. For convenience, the controller is modeled as a continuous-time system although, in practice, it is implemented in Discrete Time (DT) on the micro-controllers that are used to perform the control actions. The control signals, $\boldsymbol{\nu}_\tau$, serve as references to the inner current loops of the amplifiers that supply the motors. For link j, assuming that the current closed-loop has a bandwidth greater than 500 Hz, its transfer function is modeled as a static gain, g_{τ_j} that applies in the frequency range of the rigid robot dynamics ω_{dyn} (usually less than 10

Hz), [1]. In this article, this gain is assumed known. With $\boldsymbol{G}_\tau = diag(g_{\tau_1}, ..g_{\tau_n})$, the actual torque is then calculated by

$$\boldsymbol{\tau}(t) = \boldsymbol{G}_\tau \boldsymbol{\nu}_\tau(t). \tag{5}$$

3 Identification Methodologies

3.1 IDIM-LS Method

Prefiltering Process

To build the matrix of regressors, $\boldsymbol{\phi}$, the velocity and the acceleration must be estimated from the measured position. In most applications, the only available information is indeed the joint positions, $\boldsymbol{q}_m$. As described in [9] for instance, the differentiation is performed with a centralized finite difference. The drawback of this technique is the amplification of the noise. In practice, that phenomenon is limited by an adequate filtering of $\boldsymbol{q}_m$, prior to the differentiation, to obtain an estimate $\widehat{\boldsymbol{q}}$. The filter, which is usually a Butterworth one, is applied in both forward and reverse directions to eliminate the phase lag that is inherent in the forward-pass filtering operation. Its cut-off frequency, ω_{f_q}, is influenced by the sampling frequency, ω_s, usually chosen 100 times larger than the natural frequency of the highest mode to be modeled, ω_{dyn}, in order to satisfy the Nyquist rule. According to [9], the rule of thumb for the cut-off frequency is $\omega_{f_q} \geq 5\omega_{dyn}$. The combination of the two-pass Butterworth filter and central differencing is referred to as the *BandPass* (BP) filtering process. By selecting the cut-off frequency, the user must ensure that $\left(\widehat{\boldsymbol{q}}, \widehat{\dot{\boldsymbol{q}}}, \widehat{\ddot{\boldsymbol{q}}}\right) \approx (\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ in the range $[0, \omega_{f_q}]$.

In practice, unmodeled friction and flexibility effects disturb the torque. Although they are rejected by the controller during the operation of the robot, those perturbations must be removed prior to the identification with a parallel low-pass filtering at the cut-off frequency $\omega_{F_p} \geq 2\omega_{dyn}$ [9]. To be consistent, this filter is also applied to the independent variables. Thereafter, there is no more useful information beyond the cut-off frequency. A 'decimation' procedure is thus undertaken: i.e. re-sampling to keep one sample over $n_d = \omega_{nyq}/\omega_{Fp}$ [9]. After data acquisition and decimation, we obtain

$$\boldsymbol{\tau}_{F_p}(t) = F_p(z^{-1})\boldsymbol{\tau}(t) = \boldsymbol{\phi}_{F_p}\left(\widehat{\boldsymbol{q}}(t), \widehat{\dot{\boldsymbol{q}}}(t), \widehat{\ddot{\boldsymbol{q}}}(t)\right)\boldsymbol{\theta} + \boldsymbol{\nu}_{F_p}(t), \tag{6}$$

with F_p the parallel filter applied to each element of the observation matrix $\boldsymbol{\phi}_{F_p}\left(\widehat{\boldsymbol{q}}(t), \widehat{\dot{\boldsymbol{q}}}(t), \widehat{\ddot{\boldsymbol{q}}}(t)\right) = F_p(z^{-1})\boldsymbol{\phi}\left(\widehat{\boldsymbol{q}}(t), \widehat{\dot{\boldsymbol{q}}}(t), \widehat{\ddot{\boldsymbol{q}}}(t)\right)$, as well as the error vector $\boldsymbol{\nu}_{F_p}(t) = F_p(z^{-1})\boldsymbol{\nu}(t)$ and z^{-1} is the backward shift (delay) operator.

Least-Squares Estimation

The model described by (2) can be straightforwardly extended to a regrouped matrix formulation, which we may also be called *en-bloc* formulation. The IDIM is re-written

$$\boldsymbol{y}(\boldsymbol{\tau}) = \boldsymbol{X}\left(\widehat{\boldsymbol{q}}, \widehat{\dot{\boldsymbol{q}}}, \widehat{\ddot{\boldsymbol{q}}}\right)\boldsymbol{\theta} + \boldsymbol{\varepsilon}, \tag{7}$$

where

- $\boldsymbol{y}(\boldsymbol{\tau})$ is the $(r \times 1)$ measurements vector built from the filtered torques $\boldsymbol{\tau}_{F_p}$;
- $\boldsymbol{X}\left(\widehat{\boldsymbol{q}}, \widehat{\dot{\boldsymbol{q}}}, \widehat{\ddot{\boldsymbol{q}}}\right)$ is the $(r \times b)$ regrouped observation matrix;
- $\boldsymbol{\varepsilon}$ is the $(r \times 1)$ vector of errors terms;
- $r = n \cdot N$ is the number of rows in (7), where $N = n_m/n_d$ is the number of sampling points after decimation and n_m is the number of measurement points.

In $\boldsymbol{y}$ and $\boldsymbol{X}$, the equations of each joint j are regrouped together. Thus, $\boldsymbol{y}$ and $\boldsymbol{X}$ are partitioned so that

$$\boldsymbol{y}(\boldsymbol{\tau}) = \begin{bmatrix} \boldsymbol{y}^1 \\ \vdots \\ \boldsymbol{y}^n \end{bmatrix}, \quad \boldsymbol{X}\left(\widehat{\boldsymbol{q}}, \widehat{\dot{\boldsymbol{q}}}, \widehat{\ddot{\boldsymbol{q}}}\right) = \begin{bmatrix} \boldsymbol{X}^1 \\ \vdots \\ \boldsymbol{X}^n \end{bmatrix}, \tag{8}$$

where $\boldsymbol{y}^j = \begin{bmatrix} \tau^j_{F_p}(t_1) \\ \vdots \\ \tau^j_{F_p}(t_N) \end{bmatrix}$; $\boldsymbol{X}^j = \begin{bmatrix} \boldsymbol{\phi}^j_{F_p}\left(\widehat{\boldsymbol{q}}(t_1), \widehat{\dot{\boldsymbol{q}}}(t_1), \widehat{\ddot{\boldsymbol{q}}}(t_1)\right) \\ \vdots \\ \boldsymbol{\phi}^j_{F_p}\left(\widehat{\boldsymbol{q}}(t_N), \widehat{\dot{\boldsymbol{q}}}(t_N), \widehat{\ddot{\boldsymbol{q}}}(t_N)\right) \end{bmatrix}$;

and $\boldsymbol{\phi}^j_{F_p}\left(\widehat{\boldsymbol{q}}(t_k), \widehat{\dot{\boldsymbol{q}}}(t_k), \widehat{\ddot{\boldsymbol{q}}}(t_k)\right)$ is the j^{th} row of the $(n \times b)$ filtered observation matrix at time t_k (k between 1 and N).

With the *en-bloc* matrix formulation (7), the Ordinary LS (OLS) estimates are computed with

$$\widehat{\boldsymbol{\theta}}_{LS}(N) = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}(\boldsymbol{\tau}). \tag{9}$$

The solution exists if $\left(\boldsymbol{X}^T\boldsymbol{X}\right)$ is invertible. That is to say that $\boldsymbol{X}$ is full column rank. The covariance matrix of the LS estimates is

$$\boldsymbol{\Sigma}(\widehat{\boldsymbol{\theta}}_{LS}) = \left(\boldsymbol{X}^T\widehat{\boldsymbol{\Omega}}_\tau^{-1}\boldsymbol{X}\right)^{-1}. \tag{10}$$

$\widehat{\boldsymbol{\Omega}}_\tau$ is the estimate of $\boldsymbol{\Omega}_\tau$ defined such as:

$$\boldsymbol{\Omega}_\tau = diag\left(\sigma_1^2\boldsymbol{I}_N, \ldots, \sigma_j^2\boldsymbol{I}_N, \ldots, \sigma_n^2\boldsymbol{I}_N\right) \tag{11}$$

where $\boldsymbol{I}_N$ is the $(N \times N)$ identity matrix and σ_j^2 is the noise variance of link j. This matrix is constructed from the covariance matrix of the $\boldsymbol{\nu}_{F_p}$ defined by:

$$\boldsymbol{\Lambda}_\tau = diag\left(\sigma_1^2, \ldots, \sigma_j^2, \ldots, \sigma_n^2\right). \tag{12}$$

For each link j, the noise $\nu_{F_p j}$ is assumed to have zero mean, to be serially uncorrelated and to be homoskedastic; i.e. a white noise.

From a theoretical point of view, the LS estimates (9) are unbiased if the error has a zero mean and if the regressors are uncorrelated with the error, see relations (13), [21] and [7].

$$\mathsf{E}[\boldsymbol{\varepsilon}] = 0, \quad \mathsf{E}[\boldsymbol{X}^T\boldsymbol{\varepsilon}] = 0 \tag{13}$$

It is assumed that those two assumptions hold. However, systems considered in this article operate in closed-loop, since they are unstable in open-loop. In that case, the assumption given by (13) does not hold; see e.g. [22]. This partly explains why a tailor-made pre-filtering of the data is done in practice.

3.2 Output Error Methods

OEM Principles

The purpose of the Output Error Methods (OEM) is to find the best parametric model with respect to a specific criterion. The criterion is a function of the error between the noisy measured output and the simulated model output. As explained in [23], many criteria may be used. In this article, we focus on the quadratic criterion given by

$$V_N(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y}(t_i) - \boldsymbol{y}_s(t_i,\boldsymbol{\theta})\right)\Big)^2, \tag{14}$$

where $\boldsymbol{y}$ is the $(n\times 1)$ vector of measured output and $\boldsymbol{y}_s$ is the $(n\times 1)$ vector simulated output. As explained in chapter 7 of [24] for instance, with such a criterion, output error models assume that the additive noise on the output is white. According to the same reference, the estimator can be consistent although the noise model is not adequate. To minimize that quadratic error, the unknown system parameters are tuned iteratively so that the simulated model output fits the measured system output, with $\widehat{\boldsymbol{\theta}}^{it+1} = \widehat{\boldsymbol{\theta}}^{it} + \widehat{\boldsymbol{\Delta\theta}}^{it}$, where $\widehat{\boldsymbol{\Delta\theta}}^{it}$ is the innovation vector at iteration it. The innovation is calculated differently depending on the applied nonlinear optimization algorithm. The criterion minimization is usually solved thanks to nonlinear optimization algorithms based on a first- or second-order Taylor series expansion like the gradient, the Gauss-Newton (GN) and the Levenberg-Marquardt methods. For the GN method, the parameter innovation is given by

$$\widehat{\boldsymbol{\Delta\theta}}^{it} = -\left[V_N''(\widehat{\boldsymbol{\theta}}^{it})\right]^{-1} V_N'(\widehat{\boldsymbol{\theta}}^{it}), \tag{15}$$

where $V_N'(\widehat{\boldsymbol{\theta}}^{it})$ and $V_N''(\widehat{\boldsymbol{\theta}}^{it})$ are the gradient vector and the Hessian matrix of the criterion V_N. That innovation vector requires the computation of the criterion derivatives with respect to the parameters. In some cases those derivatives can be exactly known. For example, in [14], the authors developed an exact formulation of the first derivative for CT-LTI systems. In [25], the authors derived the same for nonlinear systems. Those derivatives of the criterion are function of the derivatives of the system's outputs with respect to the parameters, which are called *sensitivities*.

To simulate the continuous-time system and obtain a simulated output, the differential equations must be solved. Many numerical solvers exist in the literature like the well-known Runge-Kutta method, for further examples see [26]. In this article, they will be referred as "integration solvers" to avoid confusion with the "optimization solvers" introduced in the previous paragraph. In practice, the integration solver needs the same input as the real system and a set of values for the parameters to identify. The choice of the integration solver is decisive. For each model, the practitioner must find the integration solver which suits to the system properties. For instance, if the system presents two dynamics with characteristic times that greatly differ, a stiff solver should be employed. If the integration solver is not appropriate, it may lead to a biased identification.

The initial values is a crucial point for OEM. With a bad initialization the optimization solver may lead to local minimum (if it is a local optimizer) or even diverge [23]. The integration solver may also diverge if the parameters are not suitable. Depending on the application, different techniques may be used to initialize correctly the method. If the problem is linear with respect to the parameters and if all the states are available, a LS estimation can be employed. As shown in [1], in the field of robotics the Computer-Aided Design (CAD) values of the inertia are accurate enough to initialize.

Closed-Loop Output Error

By applying directly the OEM to a robot model, it seems natural to take the joint position vector as the identification signal. The output error vector is defined by:

$$\boldsymbol{\varepsilon}_{CLOE}(t,\boldsymbol{\theta}) = \boldsymbol{q}_m(t) - \boldsymbol{q}_s(t,\boldsymbol{\theta}), \tag{16}$$

where $\boldsymbol{q}_s$ is the $(n\times 1)$ vector of simulated joint positions. Since the robots are unstable in open-loop, they are identified in closed-loop and the dedicated identification method is called the Closed-Loop Output Error (CLOE) method. The simulated output $\boldsymbol{q}_s$ is generated with the reference signal $\boldsymbol{q}_r$ that is perfectly known and consequently noise-free. Therefore, there is not bias induced by a noise correlation and the estimation is consistent, assuming that there is no modeling error and that the optimization solver has converged to the global minimum.

As explained in the previous section, OEM problem is usually solved thanks to nonlinear optimization algorithms. In this part, we focus on the GN method, which is based on a second order Taylor series expansion of $\boldsymbol{q}_s$, at current estimates $\boldsymbol{\theta}_{CLOE}$; see (15). After data sampling, the following over-determined system is obtained at iteration it:

$$\boldsymbol{\Delta y}(\boldsymbol{q}) = \boldsymbol{\Psi}^{it}_{CLOE}\boldsymbol{\Delta\theta}^{it}_{CLOE} + \boldsymbol{e}_{CLOE} \tag{17}$$

where

- $\boldsymbol{\Delta y}(\boldsymbol{q})$ is the $(r'\times 1)$ vector built from the sampling of $\boldsymbol{\varepsilon}_{CLOE}(t,\boldsymbol{\theta})$, similarly to (8);
- $\boldsymbol{\Psi}^{it}_{CLOE}$ is the $(r'\times b)$ matrix built from the n matrices ${\boldsymbol{\Psi}^{it}_{CLOE}}^{j} = \begin{bmatrix}\boldsymbol{\Delta}^j_{\boldsymbol{q}_s}(t_1)\\ \vdots \\ \boldsymbol{\Delta}^j_{\boldsymbol{q}_s}(t_{n_m})\end{bmatrix}$, where $\boldsymbol{\Delta}^j_{\boldsymbol{q}_s}(\cdot)$ is the j^{th} row of the $(n\times b)$ Jacobian matrix $\boldsymbol{\Delta}_{\boldsymbol{q}_s} = \left.\frac{\partial \boldsymbol{q}_s}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}^{it}_{CLOE}}$;

- $\boldsymbol{e}_{CLOE}$ is the $(r' \times 1)$ vector built from the sampling of $\boldsymbol{\varepsilon}_{CLOE}$ and the residuals of the Taylor series expansion
- $r' = n \cdot n_m$ is the number of equation considered, without any decimation.

$\boldsymbol{\Delta\widehat{\theta}}^{it}_{CLOE}$ is the $(b \times 1)$ vector of estimated parameters increments and is the LS solution of (17). Each element of the Jacobian $\boldsymbol{\Delta}_{\boldsymbol{q}_s}$ is an output sensitivity function which defines the variation of the output position with respect to the parameters. Usually, those sensitivity functions are not exactly known and approximated with finite differences.

The construction of the *en-bloc* formulation is really similar to the one of the IDIM-LS method (7). The difference is the use of r' instead of r. This is due to the fact that we are in an output error framework. Hence, the additive noise is assumed white and does not need any prefiltering such as the decimate filter. It thus begins to emerge here a limitation of OEM for robot identification. In addition, as we shall see later the decimation process presents also an advantage for the conditioning number because the number of sampling points to treat is indeed reduced for the optimization solver.

Closed-Loop Input Error

As we have seen in section 2.1, it is common to use the input torque/force for robot identification. Therefore, a variant of the CLOE method based on the input signal can be considered. This technique, termed as Closed-Loop Input Error (CLIE) method in [27], relies on the input error vector

$$\boldsymbol{\varepsilon}_{CLIE}(t,\boldsymbol{\theta}) = \boldsymbol{\tau}(t) - \boldsymbol{\tau}_s(t,\boldsymbol{\theta}). \tag{18}$$

If the problem is solved with the GN algorithm, $\boldsymbol{\Delta\theta}^{it}_{CLIE}$ is the LS solution of

$$\boldsymbol{\Delta y}(\boldsymbol{\tau}) = \boldsymbol{\Psi}^{it}_{CLIE}\boldsymbol{\Delta\theta}^{it}_{CLIE} + \boldsymbol{e}_{CLIE} \tag{19}$$

where

- $\boldsymbol{\Delta y}(\boldsymbol{\tau})$ is the $(r' \times 1)$ vector built from the sampling of $\boldsymbol{\varepsilon}_{CLIE}(t,\boldsymbol{\theta})$, similarly to (8);
- $\boldsymbol{\Psi}^{it}_{CLIE}$ is the $(r' \times b)$ matrix built from the n matrices ${\boldsymbol{\Psi}^{it}_{CLIE}}^{j} = \begin{bmatrix} \boldsymbol{\Delta}^{j}_{\boldsymbol{\tau}_s}(t_1) \\ \vdots \\ \boldsymbol{\Delta}^{j}_{\boldsymbol{\tau}_s}(t_{n_m}) \end{bmatrix}$, where $\boldsymbol{\Delta}^{j}_{\boldsymbol{\tau}_s}(\cdot)$ is the j^{th} row of the $(n \times b)$ Jacobian matrix $\boldsymbol{\Delta}_{\boldsymbol{\tau}_s} = \left.\frac{\partial \boldsymbol{\tau}_s}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}^{it}_{CLIE}}$;
- $\boldsymbol{e}_{CLIE}$ is the $(r' \times 1)$ vector built from the sampling of $\boldsymbol{\varepsilon}_{CLIE}$ and the residuals of the Taylor series expansion.

The CLOE and CLIE methods are iterative and can share the initialization and the convergence criterion. In section 4, we investigate their properties.

3.3 Direct and Indirect Dynamic Identification Method

Recently in [1], the authors have introduced a *pseudo* OEM dedicated to robots based on the DDM, called DIDIM for Direct and Inverse Identification Model. Similarly to the CLIE method, the observation variable is the torque. This method may be regarded as *pseudo* OEM because it includes the parallel filter. Hence, the additive noise, $\boldsymbol{\nu}$, is not necessary considered as white. The second specificity of the DIDIM method is that the dependence of $\boldsymbol{\phi}$ in $\boldsymbol{\theta}$ is neglected. In the field of system identification, such an assumption is called Pseudo-Linear Regression (PLR), see Eq. (7.112) in [28]. According to the same reference, PLR is derived from [29]. Thanks to the PLR, the GN algorithm becomes equivalent to the linear LS, as shown in section 4.3.3.3 of [23]. The input sensitivity is written:

$$\begin{aligned} \boldsymbol{\Delta}_{\boldsymbol{\tau}_s}(t) &= \left.\frac{\partial \boldsymbol{\tau}_s(t)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}^{it}_{DIDIM}} \\ &= \boldsymbol{\phi}\left(\boldsymbol{q}_s(t,\widehat{\boldsymbol{\theta}}^{it}_{DIDIM}), \dot{\boldsymbol{q}}_s(t,\widehat{\boldsymbol{\theta}}^{it}_{DIDIM}), \ddot{\boldsymbol{q}}_s(t,\widehat{\boldsymbol{\theta}}^{it}_{DIDIM})\right). \end{aligned} \tag{20}$$

Thanks to this relation, the input sensitivity can be calculated with only one simulation of the closed-loop system. In the opposite, with finite differences, $b+1$ simulations are needed to evaluate the sensitivity; considering a forward or a backward first order and one-sided difference scheme. As we shall see later, the gain in computing time is therefore not negligible.

4 Comparison of Output Error Methods

4.1 Sensitivity Relation

For the CLIE method, at iteration it, the input sensitivity is defined such as

$$\boldsymbol{\Delta}_{\boldsymbol{\tau}_s}(t) = \left.\frac{\partial \boldsymbol{\tau}_s(t)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}^{it}_{CLIE}}. \tag{21}$$

By using the controller definition (5), it comes:

$$\boldsymbol{\Delta}_{\boldsymbol{\tau}_s}(t) = \left.\frac{\partial \boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)\left(\boldsymbol{q}_r(t) - \boldsymbol{q}_s(t,\widehat{\boldsymbol{\theta}}^{it}_{CLIE})\right)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}^{it}_{CLIE}}, \tag{22}$$

and assuming that the controller is known, or at least not identified at the same time:

$$\begin{aligned} \boldsymbol{\Delta}_{\boldsymbol{\tau}_s}(t) &= -\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)\left.\frac{\partial\left(\boldsymbol{q}_s(t,\widehat{\boldsymbol{\theta}}^{it}_{CLIE})\right)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}^{it}_{CLIE}} \\ &= -\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)\boldsymbol{\Delta}_{\boldsymbol{q}_s}(t). \end{aligned} \tag{23}$$

Eq. (23) is the key relation to compare the CLOE and CLIE methods. Loosely speaking, the CLIE method is a frequency weighting of the CLOE method by the controller. In practice, the input sensitivity functions can be obtained from the filtering of the output ones, if they are available.

Because the robots are controlled in position and assuming that the optimization solver is adequately initialized, the following relations are expected:

$$\boldsymbol{q}_m(t) \approx \boldsymbol{q}_r(t), \tag{24}$$
$$\boldsymbol{q}_s(t) \approx \boldsymbol{q}_r(t), \tag{25}$$

at each iteration *it* of the algorithm. Because of (24) and (25), the controller may be assumed to operate in low frequencies range and, then, to be a $(n \times n)$ constant matrix $\boldsymbol{C}_0$. If the controller is a PD control law then this result is straightforward. If the controller contains an integral term, this result can be still considered as valid. In fact, as explained in section 2.2, to avoid oscillations due to the Coulomb friction, the integral action is deactivated when the position error is too small and this implies that PID controller reduces to PD controller. Thus, (23) may be re-written

$$\boldsymbol{\Delta}_{\boldsymbol{\tau}_s}(t) = -\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}_0\boldsymbol{\Delta}_{\boldsymbol{q}_s}(t). \tag{26}$$

4.2 Equivalence of the CLOE and CLIE estimations

Considering relation (26), the CLIE method can be seen as a weighted CLOE method (WCLOE). That can be seen with the LS solution of (19):

$$\begin{aligned}\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLIE} &= \left[\left(\boldsymbol{\Psi}^{it}_{CLIE}\right)^T\boldsymbol{\Psi}^{it}_{CLIE}\right]^{-1}\left(\boldsymbol{\Psi}^{it}_{CLIE}\right)^T\boldsymbol{\Delta y}(\boldsymbol{\tau}) \\ &\approx \left[\left(\boldsymbol{\Psi}^{it}_{CLOE}\right)^T\bar{\boldsymbol{C}}_0^T\bar{\boldsymbol{C}}_0\boldsymbol{\Psi}^{it}_{CLOE}\right]^{-1}\left(\boldsymbol{\Psi}^{it}_{CLOE}\right)^T\bar{\boldsymbol{C}}_0^T\bar{\boldsymbol{C}}_0\boldsymbol{\Delta y}(\boldsymbol{q}) \\ &= \left[\left(\boldsymbol{\Psi}^{it}_{CLOE}\right)^T\boldsymbol{W}\boldsymbol{\Psi}^{it}_{CLOE}\right]^{-1}\left(\boldsymbol{\Psi}^{it}_{CLOE}\right)^T\boldsymbol{W}\boldsymbol{\Delta y}(\boldsymbol{q}),\end{aligned} \tag{27}$$

where

- $\bar{\boldsymbol{C}}_0$ is $(r' \times r')$ matrix built from the constant controller matrix $\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}_0$;
- the relation between both errors is $\boldsymbol{\Delta y}(\boldsymbol{\tau}) \approx -\bar{\boldsymbol{C}}_0\boldsymbol{\Delta y}(\boldsymbol{q})$, because it is built from

$$\begin{aligned}\boldsymbol{\tau}(t) - \boldsymbol{\tau}_s(t) &= \boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)\left(\boldsymbol{q}_r(t) - \boldsymbol{q}_m(t)\right) \\ &\quad - \boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)\left(\boldsymbol{q}_r(t) - \boldsymbol{q}_s(t)\right) \\ &= -\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)\left(\boldsymbol{q}_m(t) - \boldsymbol{q}_s(t)\right) \\ &\approx -\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}_0\left(\boldsymbol{q}_m(t) - \boldsymbol{q}_s(t)\right).\end{aligned}$$

Relation (27) is clearly a weighted least-squares solution. If n_m is sufficiently large and assuming that the trajectories are exciting enough, according to the theory of statistics [21], one has

$$\mathsf{E}[\widehat{\boldsymbol{\theta}}^{it}_{CLIE}] \to \mathsf{E}[\widehat{\boldsymbol{\theta}}^{it}_{CLOE}].$$

It comes out that the CLIE and CLOE methods asymptotically provide the same estimates. The resulting question is: is there a dominant estimator?

Because we are in an output error framework, the output position can be written:

$$\boldsymbol{q}_m(t) = \boldsymbol{q}(t) + \boldsymbol{n}_q(t), \tag{28}$$

where $\boldsymbol{n}_q$ is a $(n \times 1)$ vector of independent white noises with a $(n \times n)$ covariance matrix $\boldsymbol{\Lambda}_q$. With the closed-loop, the input is given by:

$$\begin{aligned}\boldsymbol{\tau}(t) &= \boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)(\boldsymbol{q}_r(t) - \boldsymbol{q}_m(t)) \\ &= \boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)(\boldsymbol{q}_r(t) - \boldsymbol{q}(t)) - \boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)\boldsymbol{n}_q(t).\end{aligned} \tag{29}$$

The reference trajectory being noise-free, the noise seen by the input torque is $\boldsymbol{v}(t) = -\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}(p)\boldsymbol{n}_q(t)$; see (2). With no further assumption, it appears that if the output error assumption is made on the output position, it is not validated for the input torque. Nonetheless, we assume that the closed-loop system operates in its bandwidth and consequently $\boldsymbol{C}(p) \approx \boldsymbol{C}_0$. The noises relation is given by:

$$\boldsymbol{v}(t) = -\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}_0\boldsymbol{n}_q(t). \tag{30}$$

Consequently, $\mathsf{E}[\boldsymbol{v}] = -\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}_0\mathsf{E}[\boldsymbol{n}_q] = 0$ and the covariances are linked such as:

$$\begin{aligned}\boldsymbol{\Lambda}_v &= \mathsf{E}\left[\left(\boldsymbol{v} - \mathsf{E}[\boldsymbol{v}]\right)\left(\boldsymbol{v} - \mathsf{E}[\boldsymbol{v}]\right)^T\right] = \mathsf{E}\left[\boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}_0\boldsymbol{n}_q\boldsymbol{n}_q^T\boldsymbol{C}_0^T\boldsymbol{G}_{\boldsymbol{\tau}}^T\right] \\ &= \boldsymbol{G}_{\boldsymbol{\tau}}\boldsymbol{C}_0\boldsymbol{\Lambda}_{n_q}\boldsymbol{C}_0^T\boldsymbol{G}_{\boldsymbol{\tau}}^T.\end{aligned} \tag{31}$$

In practice, the matrices $\boldsymbol{G}_{\boldsymbol{\tau}}$ and $\boldsymbol{C}_0$ can be assumed diagonal, which is justified by the technology: one controller by link as presented in section 2.2. Attention is drawn to the fact that the covariance $\boldsymbol{\Lambda}_v$ is the one of $\boldsymbol{v}(t)$ whereas $\boldsymbol{\Lambda}_{\boldsymbol{\tau}}$ is the one of $\boldsymbol{v}_{F_p}(t)$; see (12). Assuming no modeling error and that the optimization solver has converged to true parameters values, the estimated parameters covariances are then given by:

$$\begin{aligned}\boldsymbol{\Sigma}\left(\widehat{\boldsymbol{\theta}}^{it}_{CLIE}\right) &= \left[\left(\boldsymbol{\Psi}^{it}_{CLIE}\right)^T\boldsymbol{\Omega}_v^{-1}\boldsymbol{\Psi}^{it}_{CLIE}\right]^{-1} \\ &= \left[\left(\boldsymbol{\Psi}^{it}_{CLOE}\right)^T\bar{\boldsymbol{C}}_0^T\left(\bar{\boldsymbol{C}}_0\boldsymbol{\Omega}_q\bar{\boldsymbol{C}}_0^T\right)^{-1}\bar{\boldsymbol{C}}_0\boldsymbol{\Psi}^{it}_{CLOE}\right]^{-1} \\ &= \left[\left(\boldsymbol{\Psi}^{it}_{CLOE}\right)^T\boldsymbol{\Omega}_q^{-1}\boldsymbol{\Psi}^{it}_{CLOE}\right]^{-1} = \boldsymbol{\Sigma}\left(\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right),\end{aligned} \tag{32}$$

where $\boldsymbol{\Omega}_v$ and $\boldsymbol{\Omega}_q$ are the $(r' \times r')$ covariances matrices respectively built from $\boldsymbol{\Lambda}_v$ and $\boldsymbol{\Lambda}_q$ like in (11). It comes out that the CLIE and the CLOE methods provide the same estimates with the same variances.

To summarize, the users have the choice between $\boldsymbol{q}_m$ and $\boldsymbol{\tau}$, but the CLIE and the CLOE methods asymptotically provide the same results. The question is: what is the best choice to identify continuous-time systems operating in closed loop? The following subsection presents the main difference between the CLOE and CLIE methods.

4.3 Robustness of the CLIE method to errors

With the good tracking assumption (25), $\boldsymbol{q}_s$ has a little dependence on parameters' variations and the output sensitivity matrix $\boldsymbol{\Delta}_{\boldsymbol{q}_s}$ contains little information. This implies that the singular values of $\boldsymbol{\Delta}_{\boldsymbol{q}_s}$ are small whereas its conditioning number denoted as $cond_2^{CLOE}$ may be very good i.e. $cond_2^{CLOE} \approx 1$; see e.g. [30]. To show that, from (17), the OLS solution can be written

$$\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE} = \left[\left(\boldsymbol{\Psi}^{it}_{CLOE}\right)^T \boldsymbol{\Psi}^{it}_{CLOE}\right]^{-1} \left(\boldsymbol{\Psi}^{it}_{CLOE}\right)^T \boldsymbol{\Delta y}(\boldsymbol{q}) \tag{33}$$

Then, the relative variation of the solution $\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}$ denoted as $d\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}$ is expressed by two upper bounds given by

$$\frac{\left\|d\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right\|_2}{\left\|\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right\|_2} \leq cond_2^{CLOE} \frac{\|d\boldsymbol{\Delta y}(\boldsymbol{q})\|_2}{\|\boldsymbol{\Delta y}(\boldsymbol{q})\|_2}, \tag{34}$$

$$\frac{\left\|d\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right\|_2}{\left\|\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE} + d\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right\|_2} \leq cond_2^{CLOE} \frac{\left\|d\boldsymbol{\Psi}^{it}_{CLOE}\right\|_2}{\left\|\boldsymbol{\Psi}^{it}_{CLOE}\right\|_2}, \tag{35}$$

where $d\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}$, $d\boldsymbol{\Delta y}(\boldsymbol{q})$ and $d\boldsymbol{\Psi}^{it}_{CLOE}$ are small variations of $\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}$, $\boldsymbol{\Delta y}(\boldsymbol{q})$ and $\boldsymbol{\Psi}^{it}_{CLOE}$ respectively. $\|\cdot\|_2$ is the 2-norm of a vector or a matrix.

Let μ^{CLOE}_{min} and μ^{CLOE}_{max} be the smallest and the greatest singular values of $\boldsymbol{\Psi}^{it}_{CLOE}$ respectively. With, $cond_2^{CLOE} = \mu^{CLOE}_{max}/\mu^{CLOE}_{min}$ and $\left\|\boldsymbol{\Psi}^{it}_{CLOE}\right\|_2 = \mu^{CLOE}_{max}$, one obtains

$$\frac{\left\|d\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right\|_2}{\left\|\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right\|_2} \leq \frac{\mu^{CLOE}_{max}}{\mu^{CLOE}_{min}} \frac{\|d\boldsymbol{\Delta y}(\boldsymbol{q})\|_2}{\|\boldsymbol{\Delta y}(\boldsymbol{q})\|_2}, \tag{36}$$

$$\frac{\left\|d\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right\|_2}{\left\|\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE} + d\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}\right\|_2} \leq \frac{\left\|d\boldsymbol{\Psi}^{it}_{CLOE}\right\|_2}{\mu^{CLOE}_{min}}. \tag{37}$$

Assuming that μ^{CLOE}_{min} and μ^{CLOE}_{max} are very small with $\mu^{CLOE}_{min} \approx \mu^{CLOE}_{max}$, one has $cond_2^{CLOE} \approx 1$ and $1/\mu^{CLOE}_{min}$ very large.

With (36), it appears the interest of having a conditioning number as close as possible to one in order to minimize the relative norm error on the solution vector. That property was pointed out in [31] where a new criterion of exciting trajectory was developed based on this relation. Equation (37) pinpoints the role of μ^{CLOE}_{min} as we shall see below. In practice, two cases must be considered for the CLOE method.

- The first case is when (24) and (25) are fulfilled. That is to say that the controller was designed to provide an excellent tracking, while rejecting perturbations and model mismatches. In this case, there is $\boldsymbol{\varepsilon}_{CLOE}(t,\boldsymbol{\theta}) \approx 0$, for each t, or equivalently $\boldsymbol{\Delta y}(\boldsymbol{q}) \approx 0$. Then, from (33), the innovation is approximately zero. It comes out that the CLOE method may be totally insensitive to modeling errors and/or to measurement noise, if the control law is effective enough. In practice, the optimization solver would not move from the initialization point.
- The second case to be considered is when (24) and (25) are not totally fulfilled. In other words, the controller presents relatively poor tracking performances. Regarding the upper bound (37), $\Delta\widehat{\boldsymbol{\theta}}^{it}_{CLOE}$ may not be robust against a small variation $d\boldsymbol{\Psi}^{it}_{CLOE}$. In fact, $\left\|d\boldsymbol{\Psi}^{it}_{CLOE}\right\|_2$ is amplified by $1/\mu^{CLOE}_{min}$. It can be thought that $d\boldsymbol{\Psi}^{it}_{CLOE}$ is mainly due to modeling errors and a poor initialization, since the simulated output is not affected by the measurement noise. This leads to unpredictable results because those errors can be interpreted as an information by the optimization solver. It comes out that the CLOE method may be sensitive to small modeling errors, if the control law is not effective enough.

As a result, the objective is to obtain a conditioning number as close as possible to one while having the smallest singular value as large as possible. Contrary to the CLOE method, the CLIE one will not suffer from small singular values because it is the CLOE method weighted by the control; see (26). As regards the industrial robots, the gains of the controller are usually high and this implies $\boldsymbol{C}_0 \gg \boldsymbol{I}_n$. Then, it follows that μ^{CLIE}_{min} and μ^{CLIE}_{max} the smallest and the greatest singular values of $\boldsymbol{\Psi}^{it}_{CLIE}$ are far greater than μ^{CLOE}_{min} and μ^{CLOE}_{max}. It comes out that the CLIE method is more robust against a small modeling error and is more sensitive to parameters' variations than the CLOE method.

5 Experimentations

5.1 Experimental Setup

In this part, we illustrate the performances of the CLIE, CLOE and DIDIM methods with the Stäubli TX40 [32], which is a serial manipulator composed of six rotational joints. There is a coupling between the joints 5 and 6 that adds two parameters: fv_{m6} and fc_{m6}, which are, respectively, the viscous and dry friction coefficient of the motor 6. The SYMORO+ software is used to automatically calculate the customized symbolic expressions of models; see [2]. The robot has 60 base dynamic parameters and from these 60 base parameters, only 28 are well identified with good relative standard deviations. These 28 parameters define a set of essential parameters that are sufficient to describe the dynamic behavior of the robot. This set was validated with a F-statistic, as shown in [3] and only the estimation of these parameters is considered here.

The reference trajectories are trapezoidal velocities (also called smoothed bang-bang accelerations). With $cond(\boldsymbol{\phi}_{F_p}) = 200$, these reference trajectories provide sufficient excitation for the estimation of the base parameters according to [33]. The joint positions and control signals are stored with a measurement frequency $\omega_s = 5$ kHz. For the IDIM-LS method, the filter cut-off frequencies are tuned according to [1]: i.e. $\omega_{f_q} = 5\omega_{dyn} = 50$ Hz and $\omega_{F_p} = 2\omega_{dyn} =$

20 Hz for the Butterworth and the decimation filters, respectively. The maximum bandwidth for joint 6 is $\omega_{dyn} = 10$ Hz.

The DDM is simulated thanks to a Simulink© model integrated with the *ode3* integration solver: Bogacki-Shampine. The gains of the simulated controller are not updated to keep the bandwidth constant as it could be done with the DIDIM method, see e.g. [1]. For the CLIE and CLOE methods, the optimization is done with the Levenberg-Marquardt (LM) algorithm implemented in the lsqnonlin function of the MatLab© Optimization Toolbox. The three methods are initialized with acceptable CAD values: all base parameters equal to 0 except $ia_j = 1$ for $j \neq 5$ and $ia_5 = 2$ because of the coupling effect.

5.2 Experimental Results

This section provides the results of the comparison between the IDIM-LS, CLIE, CLOE and DIDIM methods. In a first time, the CLIE and CLOE methods are studied in a strict output error framework. In a second time, the three methods are compared in more general framework by taking into account the decimate filter. In addition of the estimated parameters, the relative standard deviations, the prediction errors, the singular values, the conditioning numbers and the computing times are used as metrics.

Strict Output Error Framework

Table 1 summarizes the essential estimated parameters and relative standard deviations. The first method illustrates the results obtained with an appropriate setting of the pre-filters. The LS estimate can therefore be considered as reference values. The CLIE and DIDIM estimated parameters are close to those of the IDIM-LS method and lie in the 3σ confidence intervals of the LS estimates. The CLOE estimates are not so far except for the Coulomb parameters fc_j and other parameters like mx_4, which are highlighted in bold in the table. This can be explained by the lack of sensitivity of this method. As explained in section 4.3, the CLIE method is in fact a CLOE method weighted by the controller. Due to large controller gains, the sensitivity with respect to the parameters is more important at the input than at the output. That is confirmed by the singular values of the Jacobian matrices in Table 2. The interest of considering the input torque for the identification is also visible in Table 3 that gives the relative errors averaged over the axes. The CLOE method indeed provides larger simulation errors on the torques than the CLIE method, whereas both have equivalent errors on the positions. The relative standard deviations of the estimated parameters may seem promising, however there is an issue with the computation of the variances. The additive noises are indeed assumed to be serially uncorrelated which is not verified with the residuals autocorrelations; see Figures 1 and 2 for the CLOE and CLIE methods respectively. Consequently, the CLOE, CLIE and DIDIM methods cannot be applied in a strict output error framework to industrial robots, although the CLIE and DIDIM methods provide consistent estimates. Since the CLIE and DIDIM results are really close, only the correlation of the CLIE residuals is presented. In fact, the CLIE and DIDIM estimators appear to be consistent but inefficient. The interest of the DIDIM method compared with the CLIE one appears with the computing time in Table 2. The DIDIM method has the advantage of calling only one time the simulator at each optimization step thanks to the PLR assumption. In the opposite, the CLIE method must call the simulator $b+1$ times the simulator to estimate the Jacobian matrix with finite differences. This explains the lower computing time of the DIDIM method in Table 2.

Table 1. CLIE, CLOE and DIDIM estimates - No decimate filter

	$\widehat{\boldsymbol{\theta}}_{LS}$	$\widehat{\boldsymbol{\theta}}^{7}_{CLIE}$	$\widehat{\boldsymbol{\theta}}^{7}_{CLOE}$	$\widehat{\boldsymbol{\theta}}^{7}_{DIDIM}$
$zz1_r$	1.24 (1.45 %)	1.25 (0.32 %)	1.22 (0.24 %)	1.25 (0.32 %)
fv_1	8.00 (0.91 %)	7.96 (0.28 %)	8.18 (0.12 %)	7.99 (0.28 %)
fc_1	7.34 (2.66 %)	7.21 (0.87 %)	6.34 (0.37 %)	7.11 (0.89 %)
$xx2_r$	-0.48 (3.28 %)	-0.47 (0.97 %)	-0.42 (0.55 %)	-0.48 (0.98 %)
$xz2_r$	-0.16 (5.40 %)	-0.16 (1.85 %)	-0.17 (0.72 %)	-0.15 (1.89 %)
$zz2_r$	1.09 (1.29 %)	1.09 (0.29 %)	1.08 (0.18 %)	1.09 (0.30 %)
$mx2_r$	2.21 (3.01 %)	2.24 (0.94 %)	2.43 (0.73 %)	2.22 (0.95 %)
fv_2	5.50 (1.45 %)	5.47 (0.35 %)	5.68 (0.22 %)	5.45 (0.35 %)
fc_2	8.24 (2.26 %)	8.35 (0.53 %)	7.40 (0.28 %)	8.35 (0.53 %)
$xx3_r$	0.13 (10.6 %)	0.13 (2.07 %)	0.18 (1.31 %)	0.13 (2.10 %)
$zz3_r$	0.11 (9.57 %)	0.12 (1.81 %)	0.11 (1.59 %)	0.11 (1.89 %)
$my3_r$	-0.60 (2.52 %)	-0.59 (0.44 %)	-0.53 (0.60 %)	-0.59 (0.44 %)
ia_3	0.09 (9.39 %)	0.09 (1.94 %)	0.09 (1.62 %)	0.09 (1.94 %)
fv_3	1.93 (2.05 %)	1.94 (0.31 %)	1.93 (0.46 %)	1.94 (0.31 %)
fc_3	6.48 (2.06 %)	6.47 (0.31 %)	5.80 (0.31 %)	6.47 (0.32 %)
mx_4	-0.02 (35.1 %)	-0.03 (3.40 %)	**0.01** (12.4 %)	-0.03 (3.50 %)
ia_4	0.03 (9.10 %)	0.03 (1.08 %)	0.03 (1.38 %)	0.03 (1.09 %)
fv_4	1.09 (1.62 %)	1.11 (0.19 %)	1.15 (0.28 %)	1.11 (0.19 %)
fc_4	2.57 (2.46 %)	2.44 (0.31 %)	2.17 (0.54 %)	2.42 (0.31 %)
$my5_r$	-0.03 (15.2 %)	-0.04 (1.81 %)	-0.06 (1.31 %)	-0.03 (1.85 %)
ia_5	0.04 (10.6 %)	0.04 (1.35 %)	0.05 (1.29 %)	0.04 (1.38 %)
fv_5	1.79 (2.33 %)	1.82 (0.24 %)	1.92 (0.34 %)	1.82 (0.25 %)
fc_5	3.07 (3.59 %)	3.01 (0.39 %)	2.26 (0.48 %)	2.99 (0.39 %)
ia_6	0.01 (13.3 %)	0.01 (2.06 %)	0.01 (1.78 %)	0.01 (2.08 %)
fv_6	0.65 (1.85 %)	0.66 (0.20 %)	0.67 (0.23 %)	0.65 (0.21 %)
fc_6	0.30 (32.9 %)	0.18 (5.92 %)	**0.04** (24.4 %)	0.25 (4.18 %)
fv_{m6}	0.61 (1.97 %)	0.60 (0.22 %)	0.57 (0.46 %)	0.60 (0.22 %)
fc_{m6}	1.90 (4.40 %)	1.94 (0.46 %)	1.95 (0.78 %)	1.92 (0.46 %)

Table 2. CLIE, CLOE and DIDIM optimization parameters - No decimate filter

	CLIE	CLOE	DIDIM
μ_{max}	$2.26\ 10^4$	9.85	$2.23\ 10^4$
μ_{min}	19.3	$2.6\ 10^{-3}$	19.1
Conditioning number	1172	3845	1166
Computing time	23min	23min	1min

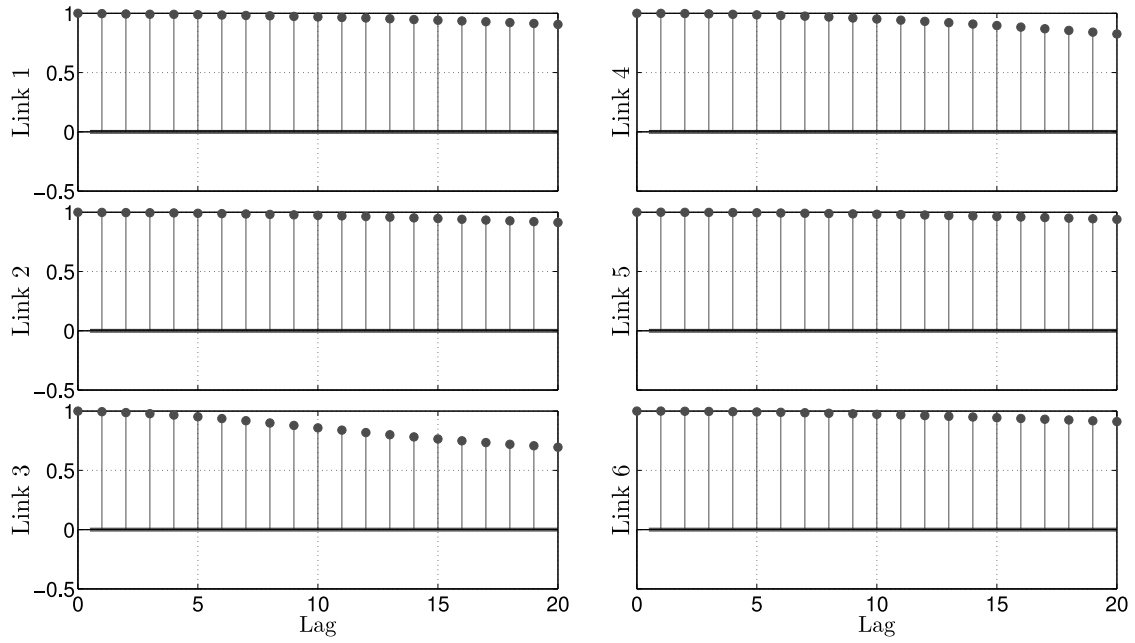

Fig. 1. CLOE residuals autocorrelations (red dots) and 2σ confidence intervals (blue lines) - No decimate filter

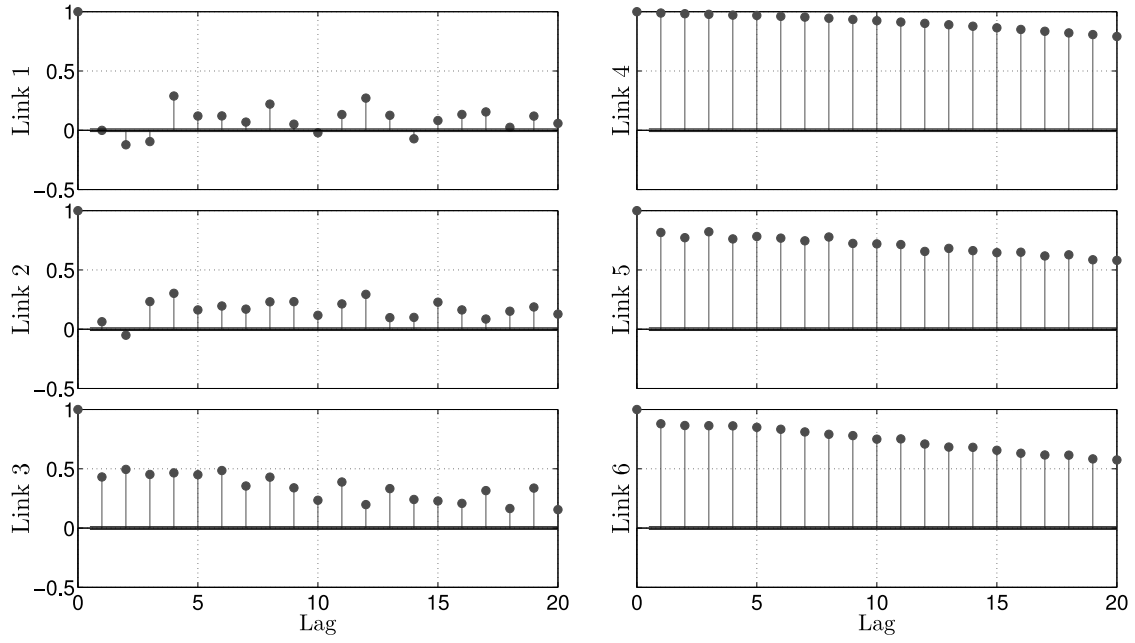

Fig. 2. CLIE residuals autocorrelations (red dots) and 2σ confidence intervals (blue lines) - No decimate filter

Output Error with the Decimation

In this part, we consider the CLIE and CLOE methods with the decimate filter as well as the DIDIM method. The estimated parameters provided in Table 4 are close to the previous estimations. There is still a lack of sensitivity for the CLOE method. Concerning the relative standard deviations of the CLIE, CLOE and DIDIM methods, they proved to be equivalent. The CLIE and DIDIM results are still really close and, regarding the correlation, the CLOE residuals present the same behaviour as the DIDIM ones. Thus, Figure 3 depicts the autocorrelations of the DIDIM residuals, representative of the three methods. The effect of the parallel filter is clear since the estimated autocorrelations coefficients are included in the confidence intervals indicated

Table 3. Direct comparison - Mean relative errors over axes - CLIE, CLOE and DIDIM - No decimate filter

	$\left\|\boldsymbol{q}_{m_j} - \boldsymbol{q}_{s_j}\right\| / \left\|\boldsymbol{q}_{m_j}\right\|$	$\left\|\boldsymbol{\tau}_j - \boldsymbol{\tau}_{s_j}\right\| / \left\|\boldsymbol{\tau}_j\right\|$
CLIE	0.017 %	6.97 %
CLOE	0.017 %	10.7 %
DIDIM	0.013 %	6.95 %

by the blue lines. These estimated autocorrelations prove that the DIDIM residuals can be considered as almost serially uncorrelated, and so are the CLIE and CLOE ones. The differences between the intervals sizes compared with Figures 1 and 2 are due to the number of samples considered for each method. If the methods without decimate filter take into account $n_m = 34500$ sampling points for the estimation, the others consider $N = n_m/n_d = 276$ sampling points due to the down-sampling. Finally, the careful reader can wonder why there is no significant time reduction between Table 2 and Table 5 since the amount of data is reduced. That comes from the fact that the most time consuming operation is the integration of the Ordinary Differential Equations (ODEs) and not the matrix inversion for the optimization. In practice, for a precision purpose, the ODEs integration is performed at the sampling frequency and not at the decimate one. Furthermore, at each call of the Simulink with a new set of parameters, a new compilation is required. This method is certainly not the optimal one from a computational perspective but is a practical way and the overall computing time is still acceptable, especially for the DIDIM method.

The experimental results show the advantages of considering the input (i.e. torque) signal for the identification. If the PLR assumption is admissible, the DIDIM method should be preferred to save computation time.

6 Conclusion

To overcome some difficulties of the *equation error* methods for robot identification, we considered the alternative of the *output error* methods. The idea is to simulate the closed-loop model of the robot and compare its *identification signal* with the one measured on the system. Two variants for *identification signal* have been studied and compared: the CLOE and CLIE methods that are respectively based on the output position and the input torque. In addition, a modification of the CLIE method named DIDIM has been considered in the case where the input torque is linear with respect to the parameters.

Experimental validation of the theoretical comparison has been carried out on a 6 DoF industrial robot and it is concluded that:

- For industrial robot with an effective control law, it is more interesting to consider the input torque as identification signal than the output position;
- If the Pseudo-Linear Regression assumption is admissible, the practitioner should consider DIDIM method to save computation time;

Table 4. CLIE, CLOE and DIDIM estimates - Decimate filter

	$\widehat{\boldsymbol{\theta}}^7_{CLIE}$	$\widehat{\boldsymbol{\theta}}^7_{CLOE}$	$\widehat{\boldsymbol{\theta}}^7_{DIDIM}$
$zz1_r$	1.25 (1.52 %)	1.22 (2.69 %)	1.25 (1.23 %)
fv_1	7.96 (0.91 %)	8.18 (1.39 %)	7.99 (0.75 %)
fc_1	7.21 (2.84 %)	6.34 (4.22 %)	7.11 (2.36 %)
$xx2_r$	-0.47 (3.72 %)	-0.42 (5.95 %)	-0.48 (3.06 %)
$xz2_r$	-0.16 (6.82 %)	-0.17 (7.92 %)	-0.15 (6.01 %)
$zz2_r$	1.09 (1.37 %)	1.08 (2.05 %)	1.09 (1.12 %)
$mx2_r$	2.24 (4.05 %)	2.43 (8.57 %)	2.22 (3.32 %)
fv_2	5.47 (1.55 %)	5.68 (2.66 %)	5.45 (1.24 %)
fc_2	8.35 (2.36 %)	7.40 (3.38 %)	8.35 (1.85 %)
$xx3_r$	0.13 (11.1 %)	0.18 (14.5 %)	0.13 (8.97 %)
$zz3_r$	0.12 (10.3 %)	0.11 (17.8 %)	0.11 (8.47 %)
$my3_r$	-0.59 (2.70 %)	-0.53 (6.38 %)	-0.59 (2.11 %)
ia_3	0.09 (11.1 %)	0.09 (17.7 %)	0.09 (8.61 %)
fv_3	1.94 (2.14 %)	1.93 (4.51 %)	1.94 (1.58 %)
fc_3	6.47 (2.19 %)	5.79 (3.22 %)	6.47 (1.59 %)
mx_4	-0.03 (26.9 %)	**0.01** (124 %)	-0.03 (18.8 %)
ia_4	0.03 (9.33 %)	0.03 (12.8 %)	0.03 (7.78 %)
fv_4	1.11 (1.81 %)	1.15 (2.52 %)	1.11 (1.48 %)
fc_4	2.44 (2.97 %)	2.17 (5.00 %)	2.42 (2.43 %)
$my5_r$	-0.04 (14.5 %)	-0.06 (13.7 %)	-0.03 (11.0 %)
ia_5	0.04 (12.1 %)	0.05 (15.4 %)	0.04 (9.78 %)
fv_5	1.82 (2.33 %)	1.92 (4.20 %)	1.82 (1.84 %)
fc_5	3.01 (3.72 %)	2.26 (6.00 %)	2.99 (2.92 %)
ia_6	0.01 (20.2 %)	0.01 (20.9 %)	0.01 (18.7 %)
fv_6	0.66 (2.07 %)	0.67 (2.85 %)	0.65 (1.59 %)
fc_6	0.18 (19.5 %)	**0.042** (315 %)	0.25 (11.6 %)
fv_{m6}	0.60 (2.19 %)	0.57 (5.66 %)	0.60 (1.67 %)
fc_{m6}	1.94 (4.50 %)	1.96 (9.84 %)	1.92 (3.44 %)

- A strict output error framework does not suit to robot identification. A careful filtering strategy must be employed to insure white residuals and thus a correct estimation of the error covariance.

Research and development are ongoing to compare the IDIM-IV and DIDIM methodologies, concentrating on their robustness and statistical properties. The work could also be extended to parallel robots that represent an important topical issue [34]. In this regard, [35] and [11] have applied the standard IDIM-LS method to parallel robots; while in [36], the authors have applied a standard LS procedure that is equivalent to the IDIM-LS approach.

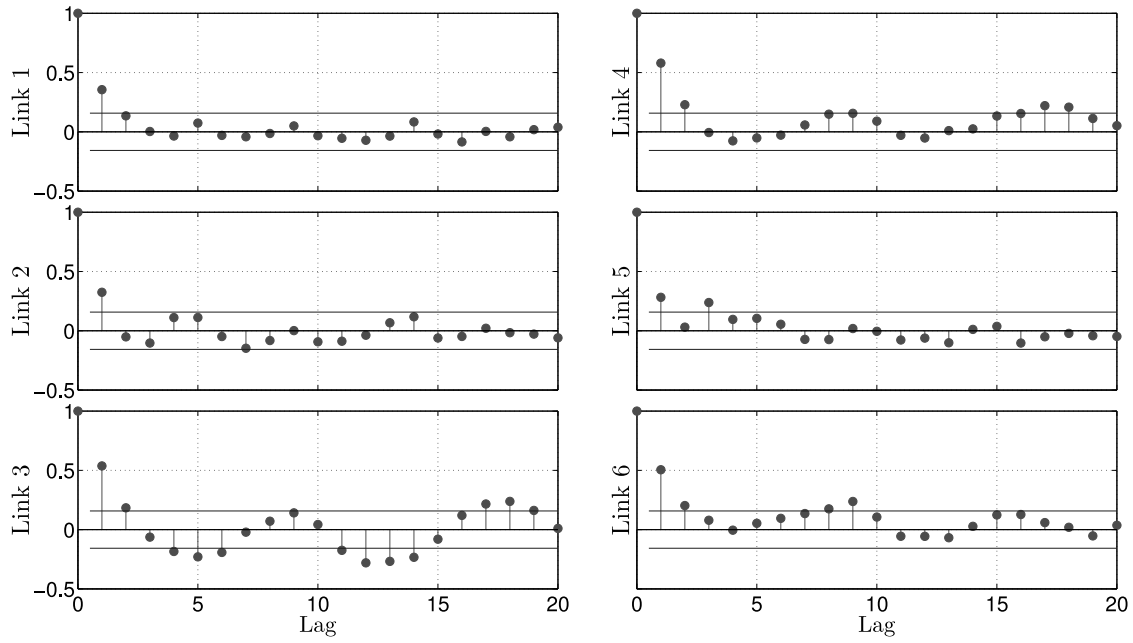

Fig. 3. DIDIM residuals autocorrelations (red dots) and 2σ confidence intervals (blue lines) - Decimate filter

Table 5. CLIE, CLOE and DIDIM optimization parameters

	CLIE	CLOE	DIDIM
μ_{max}	1589	1.34	1571
μ_{min}	1.3	1.74 10^{-4}	1.3
Conditioning number	1178	7622	1165
Computing time	23min	23min	1min

References

[1] Gautier, M., Janot, A., and Vandanjon, P.-O., 2013. "A new closed-loop output error method for parameter identification of robot dynamics". *IEEE Transactions on Control Systems Technology,* **21**(2), pp. 428–444.

[2] Khalil, W., and Dombre, E., 2004. *Modeling, identification and control of robots*. Butterworth-Heinemann.

[3] Janot, A., Vandanjon, P.-O., and Gautier, M., 2014. "A generic instrumental variable approach for industrial robot identification". *IEEE Transactions on Control Systems Technology,* **22**(1), pp. 132–145.

[4] Brunot, M., Janot, A., Young, P., and Carrillo, F., 2018. "An improved instrumental variable method for industrial robot model identification". *Control Engineering Practice,* **74**, pp. 107–117.

[5] Tomita, Y., Damen, A., and Van den Hof, P., 1992. "Equation error versus output error methods". *Ergonomics,* **35**(5-6), pp. 551–564.

[6] Eykhoff, P., 1974. *System Identification : Parameter and State Estimation*. Wiley-Interscience London.

[7] Young, P., 2011. *Recursive Estimation and Time-Series Analysis: An Introduction for the Student and Practitioner*. Springer Verlag Berlin.

[8] Bélanger, P., Dobrovolny, P., Helmy, A., and Zhang, X., 1998. "Estimation of angular velocity and acceleration from shaft-encoder measurements". *The International Journal of Robotics Research,* **17**(11), pp. 1225–1233.

[9] Gautier, M., 1997. "Dynamic identification of robots with power model". In Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on, Vol. 3, IEEE, pp. 1922–1927.

[10] Swevers, J., Verdonck, W., and De Schutter, J., 2007. "Dynamic model identification for industrial robots". *IEEE Control Systems,* **27**(5), pp. 58–71.

[11] Wu, J., Wang, J., and You, Z., 2010. "An overview of dynamic parameter identification of robots". *Robotics and computer-integrated manufacturing,* **26**(5), pp. 414–419.

[12] Landau, I. D., and Karimi, A., 1997. "An output error recursive algorithm for unbiased identification in closed loop". *Automatica,* **33**(5), pp. 933–938.

[13] Landau, I., Anderson, B., and De Bruyne, F., 1999. "Closed-loop output error identification algorithms for nonlinear plants". In Decision and Control, 1999. Proceedings of the 38th IEEE Conference on, Vol. 1, IEEE, pp. 606–611.

[14] Carrillo, F., Baysse, A., and Habbadi, A., 2009. "Output error identification algorithms for continuous-time systems operating in closed-loop". *IFAC Proceedings Volumes,* **42**(10), pp. 408–413.

[15] Urrea, C., and Pascal, J., 2016. "Design, simulation, comparison and evaluation of parameter identification methods for an industrial robot". *Computers and Electrical Engineering,* **In press**, pp. 1–16.

[16] Dolinský, K., and Čelikovský, S., 2017. "Application of the method of maximum likelihood to identification of bipedal walking robots". *IEEE Transactions on Control Systems Technology,* **PP**(99), pp. 1–8.

[17] Montazeri, A., West, C., Monk, S. D., and Taylor, C. J., 2017. "Dynamic modelling and parameter estimation of a hydraulic robot manipulator using a multi-objective genetic algorithm". *International Journal of Control,* **90**(4), pp. 661–683.

[18] Wensing, P. M., Kim, S., and Slotine, J. J. E., 2018. "Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution". *IEEE Robotics and Automation Letters,* **3**(1), Jan, pp. 60–67.

[19] Miranda-Colorado, R., and Moreno-Valenzuela, J., 2018. "Experimental parameter identification of flexible joint robot manipulators". *Robotica,* **36**(3), pp. 313–332.

[20] Jung, D., Cheong, J., Park, D., and Park, C., 2018. "Backward sequential approach for dynamic parameter identification of robot manipulators". *International Journal of Advanced Robotic Systems,* **15**(1), pp. 1–10.

[21] Davidson, R., and MacKinnon, J., 1993. *Estimation and Inference in Econometrics*. Oxford University Press.

[22] Van den Hof, P., 1998. "Closed-loop issues in system identification". *Annual reviews in control,* **22**, pp. 173–186.

[23] Walter, É., and Pronzato, L., 1994. *Identification de modèles paramétriques à partir de données expérimentales*. Masson.

[24] Söderström, T., and Stoica, P., 1988. *System identification*. Prentice-Hall, Inc.

[25] Landau, I., Anderson, B. D., and De Bruyne, F., 2001. "Recursive identification algorithms for continuous-time nonlinear plants operating in closed loop". *Automatica,* **37**(3), pp. 469–475.

[26] Hairer, E., Nørsett, S. P., and Wanner, G., 1993. *Solving Ordinary Differential Equations I*, 2 ed., Vol. 8. Springer-Verlag, Berlin.

[27] Garrido, R., and Miranda, R., 2012. "Dc servomechanism parameter identification: A closed loop input error approach". *ISA transactions,* **51**(1), pp. 42–49.

[28] Ljung, L., 1999. "System identification: theory for the user". *PTR Prentice Hall, Upper Saddle River, NJ*.

[29] Solo, V., 1979. "Time series recursions and stochastic approximation". *Bulletin of the Australian Mathematical Society,* **20**(01), pp. 159–160.

[30] Lawson, C., and Hanson, R., 1974. *Solving least squares problems*. Englewood Cliffs: Prentice-Hall.

[31] Pressé, C., and Gautier, M., 1993. "New criteria of exciting trajectories for robot identification". In Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on, IEEE, pp. 907–912.

[32] Stäubli Favergues, 2015. *Arm - TX series 40 family*. Stäubli, 02.

[33] Gautier, M., and Khalil, W., 1991. "Exciting trajectories for the identification of base inertial parameters of robots". In Proceedings of the 30th IEEE Conference on Decision and Control, pp. 494–499.

[34] Wu, J., Zhang, B., and Wang, L., 2016. "Optimum design and performance comparison of a redundantly actuated solar tracker and its nonredundant counterpart". *Solar Energy,* **127**, pp. 36–47.

[35] Briot, S., and Gautier, M., 2015. "Global identification of joint drive gains and dynamic parameters of parallel robots". *Multibody System Dynamics,* **33**(1), pp. 3–26.

[36] Wu, J., Wang, J., and Wang, L., 2008. "Identification of dynamic parameter of a 3dof parallel manipulator with actuation redundancy". *Journal of Manufacturing Science and Engineering,* **130**(4), p. 041012.

List of Tables

List of Figures