One-Stage Auto-Tuning Procedure of Robot Dynamics and Control Parameters for Trajectory Tracking Applications

Loris Roveda\textsuperscript{1}, Marco Forgione\textsuperscript{1} and Dario Piga\textsuperscript{1}

Abstract—Autonomy is increasingly demanded by industrial manipulators. Robots have to be capable to regulate their behavior to different operational conditions, without requiring high time/resource-consuming human intervention. Achieving an automated tuning of the control parameters of a manipulator is still a challenging task. This paper addresses the problem of automated tuning of the manipulator controller for trajectory tracking. A Bayesian optimization algorithm is proposed to tune both the low-level controller parameters (i.e., robot dynamics compensation) and the high-level controller parameters (i.e., the joint PID gains). The algorithm adapts the control parameters through a data-driven procedure, optimizing a user-defined trajectory-tracking cost. Safety constraints ensuring, e.g., closed-loop stability and bounds on the maximum joint position errors, are also included. The performance of the proposed approach is demonstrated on a torque-controlled 7-degree-of-freedom FRANKA Emika robot manipulator. The 25 robot control parameters (i.e., 4 link-mass parameters and 21 PID gains) are tuned in 125 iterations, and comparable results with respect to the FRANKA Emika embedded position controller are achieved.

I. INTRODUCTION

A. Context

Nowadays, robots are required to adapt to (partially) unknown situations, being able to optimize their behaviors through continuous interactions with the environment. In such a way, robots can achieve a high level of autonomy, allowing them to face unforeseen situations [1]. Such capabilities are also needed for industrial manipulators, requiring reconfigurability, adaptability and flexibility. Indeed, the manipulator has to autonomously adapt to new tasks and working conditions, avoiding as much as possible the human intervention (i.e., time- and resources-consuming tasks [2]).

In order to achieve this autonomy, the manipulator has to be capable to self-adapt for the task at hand. Machine learning techniques are extremely effective to tackle such a problem, and many approaches have been investigated in recent years to improve the level of autonomy of manipulators through auto-tuning methodologies.

The work has been developed within the project DAEDALUS - Distributed control and simulation platform to support an ecosystem of digital automation developers, funded by the European Commission under the grant agreement H2020-723248.

\textsuperscript{1} Loris Roveda, Marco Forgione and Dario Piga are with Istituto Dalle Molle di studi sull’Intelligenza Artificiale (IDSIA), Scuola Universitaria Professionale della Svizzera Italiana (SUPSI), Università della Svizzera italiana (USI), via Cantonale 2C- 6928, Manno, Switzerland
loris.roveda@idsia.ch, marco.forgione@idsia.ch, dario.piga@idsia.ch

B. State-of-the-art machine learning techniques for control tuning

Controller design and tuning is one of the most investigated topics in robotics. Standard model-based methodologies require identification of the robot dynamics, with data gathered from time-consuming ad-hoc experiments [3], [4]. Furthermore, when estimating a model of the manipulator, it is hard to determine a priori the model accuracy required to meet a given closed-loop performance specification [5], [6].

In recent years, 	extit{machine learning} is emerging as an alternative paradigm for data-driven robot control design [7], [8]. Programming-by-demonstration approaches are proposed in [9], [10], in which the human is physically teaching a specific task to the manipulator. Additionally, autonomous learning of robot tasks are also deeply investigated [11].

Machine learning techniques are commonly used to tackle different control learning problems. On the one hand, data-driven modelling of the robot dynamics are employed, and advanced controllers are designed based on the estimated model of the robot and of the interacting environment [6], [12]–[16]. On the other hand, machine learning techniques are also employed for auto-tuning of the robot control parameters. For example, [17] proposes a reinforcement learning approach to assists a human operator to perform a given task with minimum workload demands, while optimizing the overall human-robot performance; [18] investigates a neural network approach to self-tune the robot controller in object balancing applications; [19] presents an iterative reinforcement learning approach which combines dynamics

![Fig. 1. The FRANKA Emika manipulator used as a test platform to evaluate the proposed procedures for control parameters auto-tuning.](image-url)
compensation (i.e., friction) and control parameters tuning for force-tracking applications; [20] proposes a sensor-based strategy to self-tune the impedance control for interaction tasks.

C. Paper contribution

This paper addresses automated tuning of robot control parameters in trajectory tracking applications with unknown manipulator dynamics. This topic has attracted the attention of many researchers in the last years, proposing auto-tuned PID-like controllers based on neural networks [21], fuzzy PID controllers optimized by genetic algorithms [22], Jacobian transpose robust controllers for finite-time trajectory tracking control in a task-space under dynamic uncertainties [23], iterative learning controllers enhanced by a disturbance observer [24]. In general, these approaches are commonly based on computationally demanding algorithms. Furthermore, control tuning is in general too time- and cost-consuming to be affordable in industrial environments, and safety constraints are not explicitly taken into account.

In order to develop an easy-applicable and efficient approach that can be implemented in real industrial plants, this paper presents a Bayesian-optimization (BO) based algorithm for data-driven self-tuning of the robot control parameters. A simple PID trajectory control loop with feedback linearization (compensating for the manipulator dynamics) and feedforward action is implemented as a control strategy. The aim of the Bayesian optimization algorithm is twofold: (i) choosing the equivalent link-mass parameters used by the feedback linearizator and the feedforward action, and (ii) optimizing the joint-level PID control gains. Joint position and velocity errors over the trajectory are used to define a performance index guiding the parameters tuning. A penalty is given to the sets of parameters leading to tracking errors exceeding a given threshold or to unstable/unsafe behaviours, also resulting in a safety stop. The equivalent link-mass parameters and the PID gains are then jointly optimized according to the trajectory-tracking objective using Bayesian optimization.

For the sake of completeness, it is worth mentioning other works applying BO in robotics applications. In particular, [25] employs constrained BO to select three force-controller parameters for combined position/interaction tasks; [26] describes a trial-and-error algorithm that allows robots to adapt their behaviour in presence of damage; [27] proposes a methodology to achieve automatic tuning of optimal parameters for whole-body control algorithms, iteratively learning the parameters of sub-spaces from the whole high-dimensional parametric space through interactive trials. Our contribution differs from the works mentioned above since it is tailored to industrial manipulators with unknown dynamics and adopting a widely-used control architecture (feedback linearizator + feedforward action + PID).

The proposed approach has been implemented in C++ exploiting KDL and limbo libraries for the robot modeling and Bayesian optimization, respectively. As a test platform, the 7 DoFs FRANKA Emika manipulator shown in Fig. 1 has been used. The dynamical model of the manipulator is assumed to be unknown. High-performance control tuning is reached in 125 trials, and comparable performance with the FRANKA Emika embedded position controller is achieved.

II. PROBLEM SETTING

Fig. 2 shows the proposed control scheme and tuning strategy, where the parameters of the joint-level PID controller and the equivalent link-mass parameters used by the feedback linearizator and feedforward controller are optimized through Bayesian optimization in order to achieve high performance trajectory tracking.

A. Robot dynamics

To implement the PID trajectory tracking controller with feedback linearization in Fig. 2, the following differential equations describing the manipulator dynamics are used [28]:

\[
B(q, m)\ddot{q} + C(q, \dot{q}, m) + g(q, m) + h_{f/d}(\dot{q}) = \tau - J(q)^T h_{cut}
\]

where \(B(q, m)\) is the robot inertia matrix; \(\dot{q}\) is the robot joint position vector; \(m\) is the robot link-mass vector; \(C(q, \dot{q}, m)\) is the robot Coriolis vector; \(g(q, m)\) is the robot gravitational vector; \(h_{f/d}(\dot{q})\) is the robot joint friction vector; \(J(q)\) is the robot Jacobian matrix; \(h_{cut}\) is the robot external wrench vector; and \(\tau\) is the robot joint torque vector. Dot notation is used in this paper to indicate time derivatives, i.e., \(q\) and \(\dot{q}\) are the first- and second-order time derivatives of \(q\).

The inertia parameters of the manipulator (i.e., the link-mass vector \(m\)), affecting the inertia matrix \(B(q, m)\), the Coriolis vector \(C(q, \dot{q}, m)\), and the gravitational vector \(g(q, m)\), are supposed to be unknown. External wrench \(h_{cut}\) is null (i.e., no interaction during trajectory tracking task) and friction \(h_{f/d}(\dot{q})\) is not included in the feedback linearization.

B. Control architecture

According to Fig. 2, the overall control action is defined as

\[
\tau = \tau_{FF} + \tau_{PID} + \tau_{FL},
\]

where \(\tau_{FF}\) and \(\tau_{PID}\) implement the feedforward and feedback control action, respectively, and \(\tau_{FL}\) is the action of the feedback linearization controller, namely

\[
\tau_{FF} = B(q, m)q^d,
\]

\[
\tau_{PID} = K_p e_q + K_d e_q + K_i \int e_q,
\]

\[
\tau_{FL} = C(q, \dot{q}, m) + g(q, m),
\]

with \(q^d\) being the joint position reference vector and \(e_q = q^d - q\) the joint position error vector.

Note that \(\tau_{PID}\) aims to improve the performance of the trajectory tracking in closed-loop, while \(\tau_{FL}\) compensates for the Coriolis and gravity terms in (1). In order to achieve high trajectory tracking performance, it is therefore important to tune the PID control gains \(K_p, K_d, K_i\) and the link-mass parameters \(m\) in (3a) and (3c).
C. Objective function

The overall performance of the trajectory tracking problem can be defined by the user to reflect its goals and requirements. In this paper, the performance is measured in terms of joint position and velocity errors over the trajectory execution time \( T \), and it is defined by the following cost:

\[
J = \sum_{i=1}^{n} (e_i^{\text{max}} + e_i^{\text{mean}} + \bar{e}) + L,  \tag{4}
\]

where \( n \) is the number of robot DoFs, \( e_i^{\text{max}} \) and \( e_i^{\text{mean}} \) are the maximum and the average, respectively, of the absolute value of the \( i \)-th joint position error over the execution time \( T \), i.e.,

\[
e_i^{\text{max}} = \max_{t \in [0, T]} |e_{q,i}(t)|,  \tag{5}
\]

\[
e_i^{\text{mean}} = \frac{1}{T} \int_{0}^{T} |e_{q,i}(t)| \, dt,  \tag{6}
\]

with \( \Delta t \) the sampling time of the controller, and \( e_{q,i} \) denoting the \( i \)-th element of the vector \( e_q \), namely, the joint position error of the \( i \)-th joint. The terms \( e_i^{\text{max}} \) and \( e_i^{\text{mean}} \) in (4) are defined similarly for the velocity error.

The term \( L \) in (4) is introduced to penalize violations of signal constraints, which may reflect safety and hardware requirements. In this paper, the penalty term \( L \) is defined in terms of a (smooth) barrier function, which penalizes trajectories exceeding a maximum joint position error \( \bar{e} \) and unstable behaviors. In particular:

\[
L = 100 e^{-\xi/T} \text{ if } |e_{q,i}(\xi)| > \bar{e}, \tag{7a}
\]

\[
L = 1000 e^{-t/T} \text{ if test interrupted}, \tag{7b}
\]

where \( \xi \leq T \) is the time when the constraint on the maximum joint position error is violated, while \( t \leq T \) is the time when test is interrupted because of a user-defined safety stopping criterion. The penalty \( L \) allows us to take into account the time in which constraint violations or safety issues arise (earlier constraint violations and unsafe events are penalized more).

III. BAYESIAN OPTIMIZATION FOR PARAMETERS TUNING

Using the controller structure described in Section II-B, the cost function \( J \) in (4) depends on tunable design parameters, namely, the PID gains \( K_p, K_d, K_i \), and the link-mass parameters \( m \). By collecting all these design parameters in a vector \( \theta \), the tuning task reduces to the minimization of the cost \( J(\theta) \) with respect to \( \theta \), within a space of admissible values \( \Theta \). However, a closed-form expression of the cost \( J \) as a function of the design parameter vector \( \theta \) is not available. Furthermore, this cost cannot be evaluated through numerical simulations as the robot dynamics are assumed to be partially unknown. Instead, it is possible to perform experiments on the robot and measure the cost \( J_i \) achieved for a given controller parameter vector \( \theta_i \), and thus run an optimization algorithm driven by measurements of \( J \). Nonetheless, the peculiar nature of the optimization problem at hand restricts the class of applicable optimization algorithms. Indeed,

(i) the measured cost \( J_i \) consists in a noisy observations of the “true” cost function, namely \( J_i = J(\theta_i) + n_i \), with \( n_i \) denoting measurement noise and possibly intrinsic process variability;

(ii) no derivative information is available;

(iii) there is no guarantee that the function \( J(\theta) \) is convex;

(iv) function evaluations may require possibly costly and time-consuming experiments on the robot.

Features (i), (ii) and (iii) rule out classical gradient-based algorithms and restrict us to the class of gradient-free, global optimization algorithms. Within this class of algorithms, Bayesian optimization (BO) is generally the most efficient in terms of number of function evaluations [29], [30] and it is thus the most promising approach to deal with (iv).

In BO, the cost \( J \) is simultaneously learnt and optimized by sequentially performing experiments on the robot. Specifically, at each iteration \( i \) of the algorithm, an experiment is
performed for a given controller parameter $\theta$, and the corresponding cost $J_i$ is measured. Then, all the past parameter-cost observations $D_i = \{(\theta_1, J_1), (\theta_2, J_2), \ldots, (\theta_i, J_i)\}$ are processed and a new parameter $\theta_{i+1}$ to be tested at the next experiment is computed according to the approach discussed in the following.

A. Surrogate model

The underlying idea behind BO is to construct a surrogate probabilistic model describing the relation between the input parameter vector $\theta$ and the corresponding cost $J$. The model is a stochastic process defined over the feasible parameter $\Theta$.

For each feasible parameter $\theta \in \Theta$, a prior model provides the probabilistic distribution $p(J(\theta))$ of the cost $J(\theta)$. Then, the posterior distribution $p(J(\theta)|D_i)$ given the available dataset $D_i$ is computed via Bayesian inference, i.e.,

$$p(J(\theta)|D_i) = \frac{p(D_i|J(\theta))p(J(\theta))}{p(D_i)}.$$  \hfill (8)

Such a probabilistic representation describes the experimenter’s uncertainty for configurations that have not yet been tested. This is the key feature distinguishing BO from classic response surface algorithms, that fit a deterministic surrogate of the cost function to the observations [30]. Intuitively, the probabilistic model $p(J(\theta)|D_i)$ should assign low variance for parameters $\theta$ that are “close” to some previously observed parameters, with a mean value close to the corresponding observations. Conversely, it should assign high variance for parameters that are “far” from all observations.

The most widely used probabilistic model used for BO is the Gaussian Process (GP) [31], which is a flexible and non-parametric model allowing one to describe arbitrarily complex functions. In a GP modelling framework, the prior model for cost $J(\cdot)$ is a Gaussian Process with mean $\mu(\cdot): \Theta \rightarrow \mathbb{R}$ and covariance function $\kappa(\cdot, \cdot): \Theta \times \Theta \rightarrow \mathbb{R}$. The latter describes prior assumptions on the correlation between two values $J(\theta_1)$ and $J(\theta_2)$, and implicitly models smoothness of $J$ w.r.t. $\theta$.

B. Acquisition function

Bayesian optimization exploits model’s uncertainty information on the posterior distribution $p(J(\theta)|D_i)$ to determine the most promising point to be tested in the next iteration. Specifically, next point $\theta_{i+1}$ is chosen taking into account the trade-off between exploitation (choosing points where the value of the performance index $J(\theta)$ is expected to be optimal) and exploration (choosing points in unexplored regions of the feasible set $\Theta$ where the variance of $J(\theta)$ is high) [29].

Such a trade-off criterion is formalized by an acquisition function $A: \Theta \rightarrow \mathbb{R}$ that should take high values for parameters $\theta$ which are expected to improve the best objective function observed up to the $i$-th iteration. Next point $\theta_{i+1}$ is chosen as

$$\theta_{i+1} = \arg\max_{\theta \in \Theta} A(\theta).$$  \hfill (9)

A popular choice for the acquisition function is the expected improvement, defined as

$$A(\theta) = EI(\theta) = \mathbb{E} \left[ \max \left\{ 0, J_i - J(\theta) \right\} \right],$$

where $J_i$ is the best value of the performance cost achieved up to iteration $i$-th, i.e.,

$$J_i = \min_{j=1 \ldots i} J_j,$$

and the expectation in (10) is taken over the posterior distribution $p(J(\theta)|D_i)$.

The acquisition function $EI(\theta)$ is thus the expected value of the objective value improvement (with respect to the best in the dataset $D_i$) that could be achieved for the point $\theta$. Besides having this intuitive interpretation, expected improvement was found to perform well in practice, providing a good balance between exploration and exploitation [29]. Furthermore, it has an analytic expression in terms of the posterior mean and variance of the GP process [30] and can thus be optimized using standard gradient-based techniques.

C. Algorithm outline

The overall Bayesian optimization procedure for autotuning of the controller parameters $\theta$ is summarized in the following. First, an initial dataset of observations $D_{in}$ is constructed by performing $n_{in}$ initial experiments with different parameters $\theta \in \Theta$ and measuring the corresponding performance $J_j$, for $j = 1 \ldots i$. The initial parameters may be randomly chosen within in the admissible range $\Theta$.

The algorithm is then iterated. At each iteration $i$ the posterior distribution $p(J(\theta)|D_i)$ is updated with the available dataset $D_i$. Then, the acquisition function $A(\theta)$ is computed based on $p(J(\theta)|D_i)$ and optimized to determine the next point $\theta_{i+1}$ to be tested. Note that this inner optimization step does not require additional experiments.

An experiment is performed with parameter $\theta_{i+1}$ and the corresponding performance index $J_{i+1}$ is measured. Finally, the dataset $D_{i+1}$ is constructed by augmenting $D_i$ with the latest measurements. This sequence continues until a termination criterion is met. The criterion could be simply a maximum number of iterations or, for instance, be based on the performance achieved in the best experiment.

IV. EXPERIMENTAL RESULTS

A. Setup description

The FRANKA Emika manipulator has been used as test platform, implementing a 1-kHz torque control. The number of link-mass parameters $m$ characterizing the feedback $\tau_{FB}$ and the feedforward $\tau_{FF}$ actions is equal to 4. These parameters are related to links 1, 2 and 5, and to the end-effector. The number of PID gains is equal to 21 (namely, 3 parameters per joint). Thus, in total, 25 parameters have to be tuned. Robot dynamics has been computed using the C++ KDL library [32]. Bayesian optimization has been performed exploiting the C++ limbo [33] and the NLopt [34] libraries.
B. Use case trajectories

The following three trajectories are considered to assess the performance of the proposed approach:

- joint variable-frequency sinusoidal motion (Traj. #1)
  \[ q_i(t) = q_i^0 + A_i \cos(2\pi f_i(t) t + \phi_i), \]
  where \( i = 1, \ldots, 7 \) identifies the joint, \( f_1 = [0.05, 0.075, 0.05, 0.075, 0.1, 0.1, 0.1] \) Hz, \( f_2 = [0.015, 0.015, 0.015, 0.015, 0.015, 0.015] \) Hz, \( A = [40, 20, 30, 20, 20, 30, 20] \), and \( \phi \) is randomly selected. 

- circular Cartesian trajectory (Traj. #2) in the \( y-z \) plane
  \[ \begin{align*}
  y(t) &= y(t - \Delta t) + r (1 - \cos(\phi(t))) \\
  z(t) &= z(t - \Delta t) + r \sin(\phi(t))
  \end{align*} \]
  where \( r = 0.05 \) m, \( v = 0.1 \) m/s, and \( \phi(t) = \frac{\pi}{2} + \phi(t - \Delta t) \Delta t \). The sampling time \( \Delta t \) is equal to 1 ms and the execution time of the trajectory is \( T = 7 \) s.

- sinusoidal Cartesian trajectory (Traj. #3)
  \[ \begin{align*}
  x(t) &= x^0 + A_x (1 - \cos(2\pi f_x(t))) \\
  y(t) &= y^0 + A_y (1 - \cos(2\pi f_y(t))) \\
  z(t) &= z^0 + A_z (1 - \cos(2\pi f_z(t)))
  \end{align*} \]
  with \( f_x = f_y = f_z = 0.1 \) Hz, \( A_x = 0.2 \) m, \( A_y = 0.05 \) m, and \( A_z = -0.15 \) m. The execution time is \( T = 10 \) s.

C. Performance evaluation

Each optimization has been performed 3 times for each case-study trajectory, obtaining comparable results. For the sake of exposition, only one experimental test for each trajectory will be described.

In the Bayesian optimization algorithm, the following constraints are imposed:

- link-mass parameters \( m \) belong to the intervals \( m_1 \in [0.5, 1.5] \) kg, \( m_2 \in [2, 6] \) kg, \( m_3 \in [3.25, 13] \) kg, \( m_4 \in [0.375, 3] \) kg;
- PID gains \( K_{p,r}, K_{d,r} \) and \( K_i \) belong to the intervals \( K_{p,j} \in [0, 5000] \) Nm/rad with \( j \) from joint 1 to 4; \( K_{p,j} \in [0, 3000] \) Nm/rad with \( j \) from joint 5 to 6; \( K_{p,7} \in [0, 2000] \) Nm/rad, \( K_{d,7} \in [0, 100] \) Nms/rad with \( j \) from joint 1 to 4; \( K_{i,j} \in [0, 20] \) Nms/rad with \( j \) from joint 5 to 7, \( K_{i,j} \in [0, 100] \) Nm/rad/s with \( j \) from 1 to 7.

The equivalent link-mass parameters \( m \) and the PID gains \( K_{p,r}, K_{d,r} \) and \( K_i \) are tuned through the one-shot procedure described in Section III.

The Bayesian optimization is initialized with \( n_{in} = 25 \) initial experiments with randomly generated parameters, and terminated after 100 additional iterations. The performance index \( J \) in (4) is minimized over the trajectory execution time \( T \). Tests are interrupted if the joint position error \( e_\theta_j(t) \) is larger than \( \tilde{e} = 1.5^\circ \) and a penalty term \( L \) is added to the cost function \( J \) according to barrier function in (7a).

The obtained results are summarized in Fig. 3, which shows the evolution of the link-mass parameters, the P gains, D gains and I gains for joint 1 (similar evolution for other joints), and the performance cost \( J \) over the Bayesian optimization iterations for the considered case-study trajectories.

Because of the penalty terms \( L \) in (7), a high performance cost \( J \) is achieved in experiments where the task is interrupted due to an unstable behaviour of the robot or because the maximum joint position error \( \tilde{e} \) is reached. As expected, this happens mainly in the first initial iterations, where control parameters are chosen randomly. In total, the experiments are stopped 79 times (out of 125) for the sinusoidal joint trajectory; 55 times for the circular Cartesian; and 24 times for the sinusoidal Cartesian trajectory.

D. Comparison with embedded position controller

The performance resulting by the controller designed through the procedure presented in this paper are compared with the performance obtained with the FRANKA Emika embedded position controller.

Fig. 4 shows the joint trajectory errors achieved by the proposed controller design methodology for all the joints. The maximum tracking error is less then 0.3°. The same figure also show the error achieved with the FRANKA Emika embedded joint position controller. The advantage of the proposed approach is in its limited number of experiments required to obtain high-performance control. In addition, considering trajectory #1 (the most complex), the proposed approach improves the trajectory tracking performance. Smoothness of the trajectory can be still improved for instance, by properly quantifying this requirement in the performance index \( J \) or by implementing more advanced control strategies with, e.g., friction compensation.

V. Conclusions

A Bayesian-optimization based approach for auto-tuning of low-level robot trajectory tracking controller with unknown dynamics has been presented. The employed control architecture, consisting in a feedback linearizator, a feedforward action, and PID controllers at the joint level,
One-stage tuning

<table>
<thead>
<tr>
<th>Traj.</th>
<th>Masses</th>
<th>$K_{p,1}$</th>
<th>$K_{d,1}$</th>
<th>$K_{i,1}$</th>
<th>$J$</th>
<th>$J$ zoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td><img src="image1" alt="Graph" /></td>
<td><img src="image2" alt="Graph" /></td>
<td><img src="image3" alt="Graph" /></td>
<td><img src="image4" alt="Graph" /></td>
<td><img src="image5" alt="Graph" /></td>
<td><img src="image6" alt="Graph" /></td>
</tr>
<tr>
<td>#2</td>
<td><img src="image7" alt="Graph" /></td>
<td><img src="image8" alt="Graph" /></td>
<td><img src="image9" alt="Graph" /></td>
<td><img src="image10" alt="Graph" /></td>
<td><img src="image11" alt="Graph" /></td>
<td><img src="image12" alt="Graph" /></td>
</tr>
<tr>
<td>#3</td>
<td><img src="image13" alt="Graph" /></td>
<td><img src="image14" alt="Graph" /></td>
<td><img src="image15" alt="Graph" /></td>
<td><img src="image16" alt="Graph" /></td>
<td><img src="image17" alt="Graph" /></td>
<td><img src="image18" alt="Graph" /></td>
</tr>
</tbody>
</table>

Fig. 3. Achieved results: number of the tested trajectory (first column); link-mass parameters $m$ (second column, $m_1$ blue line, $m_2$ red line, $m_3$ yellow line, $m_4$ purple line); PID gains $K_{p,1}$, $K_{d,1}$ and $K_{i,1}$ (third, forth and fifth column); and objective function $J$ (sixth column) vs Bayesian optimization iterations. Zoomed evolution of the objective function $J$ around its minimum (seventh column). Optimal iteration is highlighted by a red circle.
Fig. 4. Trajectory tracking errors for each robot joint achieved by the FRANKA Emika controller (blue line) and by the one-stage tuning procedure (red line).
is widely used on industrial manipulators, therefore, easily implementable. Both the robot dynamic parameters and the PID gains are tuned in order to optimize the trajectory tracking performance. Safety constraints and maximum joint position errors are also taken into account.

The proposed methodology is evaluated in real experiments, using a torque-controlled FRANKA Emika manipulator. The 25 parameters defining the overall controller (i.e., 4 link-mass parameters and 21 PID gains) are optimized and comparable performance with respect to the FRANKA Emika embedded controller are achieved. The proposed procedure allows to optimize the control parameters in a limited number of iterations (i.e., 125 iterations), resulting in an efficient and effective auto-tuning methodology.

The main limitation of the presented work is the lack of knowledge transfer from trajectory to trajectory. Current and future works are devoted to fill this gap, by learning the relationship between the optimal controller parameters and the trajectory to be tracked. In addition, more advanced controllers will be considered (e.g., including friction compensation). Furthermore, a two-stage procedure is also under development, in order to split the overall design problem into two sub-problems of smaller complexity (i.e., optimizing separately dynamics parameters and trajectory tracking control parameters).

REFERENCES