Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Two-Stage Robot Controller Auto-Tuning Methodology for Trajectory Tracking Applications

Loris Roveda, * **Marco Forgione,** * **Dario Piga** *

* *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 (e-mail: loris.roveda@idsia.ch; marco.forgione@idsia.ch; dario.piga@idsia.ch).*

Abstract: Autonomy is increasingly demanded of industrial manipulators. Robots have to be capable of regulating 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 firstly the low-level controller parameters (*i.e.*, robot dynamics compensation), then the high-level controller parameters (*i.e.*, the joint PID gains), providing a two-stage robot controller auto-tuning methodology. In both the optimization phases, 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 4 robot dynamics parameters (*i.e.*, 4 link-mass parameters) are tuned in 40 iterations, while the robot control parameters (*i.e.*, 21 PID gains) are tuned in 90 iterations. Comparable trajectory tracking-errors results with respect to the FRANKA Emika embedded position controller are achieved.

Keywords: Industrial robots, robot control, robot dynamics, parameter optimization, parameter identification, system identification.

1. INTRODUCTION

1.1 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 (Makridakis, 2017). 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 (Bruzzone et al., 2018).

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

1.2 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 (Jin and Gans, 2015; Swevers et al., 2007). 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 (Formentin et al., 2016; Piga et al., 2018).

In recent years, *machine learning* is emerging as an alternative paradigm for data-driven robot control design (Antsaklis

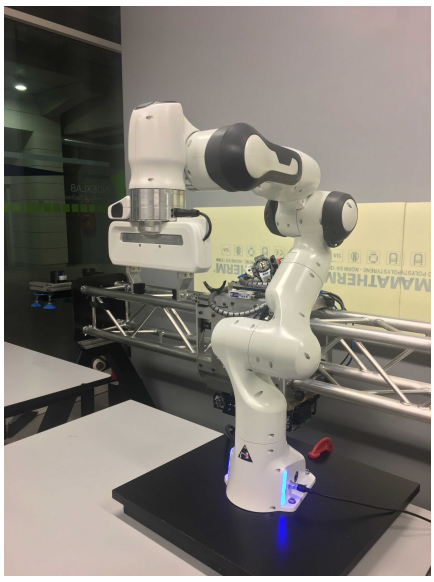

Fig. 1. The FRANKA Emika manipulator used as a test platform is shown.

and Rahnama, 2018; Arulkumaran et al., 2017). Programming-by-demonstration approaches are proposed in (Mollard et al., 2015; Rozo et al., 2013), in which the human is physically teaching a specific task to the manipulator. Additionally, autonomous learning of robot tasks are also deeply investigated (Pinto and Gupta, 2016).

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 (Bansal et al., 2017; Calandra et al., 2015; Finn et al., 2016; Piga et al., 2019; Venkatraman et al., 2016). On the other hand, machine learning techniques are also employed for auto-tuning of the robot control parameters. For example, (Modares et al., 2015) 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; (Jaisumroum et al., 2016) investigates a neural network approach to self-tune the robot controller in object balancing applications; (Roveda et al., 2018) presents an iterative reinforcement learning approach which combines dynamics compensation (*i.e.*, friction) and control parameters tuning for force tracking applications; (Balatti et al., 2018) proposes a sensor-based strategy to self-tune the impedance control for interaction tasks.

1.3 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 (Hernández-Alvarado et al., 2016), fuzzy PID controllers optimized by genetic algorithms (Zhao et al., 2016), Jacobian transpose robust controllers for finite-time trajectory tracking control in a task-space under dynamic uncertainties (Galicki, 2016), iterative learning controllers enhanced by a disturbance observer (Hsiao and Huang, 2017). 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 easily-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 BO algorithm is twofold: (*i*) choosing the equivalent link-mass parameters used by the feedback linearization 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. Firstly, the equivalent link-mass parameters are optimized in order to compensate for the robot dynamics. Then, the PID gains are optimized to maximize the trajectory tracking performance. While the first optimization can be referred to the system dynamics identification methodology, the second optimization can be referred to the data-driven control design methodology. As a test platform, the 7 DoFs *FRANKA Emika manipulator* shown in Fig. 1 has been used. High-performance control tuning is reached in less than 130 trials (40 iterations to tune the robot dynamics parameters - *i.e.*, 4 link-mass parameters - and 90 iterations to tune the robot control parameters - *i.e.*, 21 PID gains), and comparable performance with the FRANKA Emika embedded position controller is achieved.

For the sake of completeness, it is worth mentioning other works applying BO in robotics applications. In particular, (Drieß et al., 2017) employs constrained BO to select three force-controller parameters for combined position/interaction tasks; (Cully et al., 2015) describes a trial-and-error algorithm that allows robots to adapt their behaviour in presence of damage; (Yuan et al., 2019) 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 linerizator + feedforward action + PID).

1.4 Paper layout

The paper is structured as follows. Section 2 is devoted to the problem formulation, where the model of the robot dynamics and the considered control architecture are described. Section 3 provides the details of the BO algorithm for the control parameters tuning. The experimental results obtained in controlling the FRANKA Emika manipulator are reported in Section 4. Conclusions and future works are given in Section 5.

2. 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 linearization and feedforward controller are optimized through BO in order to achieve high performance trajectory tracking.

2.1 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 (Siciliano and Villani, 2000):

$$\mathbf{B}(\mathbf{q},\mathbf{m})\ddot{\mathbf{q}}+\mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\mathbf{m})+\mathbf{g}(\mathbf{q},\mathbf{m})+\mathbf{h}_{f,q}(\dot{\mathbf{q}})=\boldsymbol{\tau}-\mathbf{J}(\mathbf{q})^T\mathbf{h}_{ext} \tag{1}$$

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

The inertia parameters of the manipulator (*i.e.*, the link-mass vector $\mathbf{m}$), affecting the inertia matrix $\mathbf{B}(\mathbf{q},\mathbf{m})$, the Coriolis vector $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\mathbf{m})$, and the gravitational vector $\mathbf{g}(\mathbf{q},\mathbf{m})$, are unknown. External wrench $\mathbf{h}_{ext}$ is null (*i.e.*, no interaction during trajectory tracking task) and friction $\mathbf{h}_{f,q}(\dot{\mathbf{q}})$ is not included in the feedback linearization.

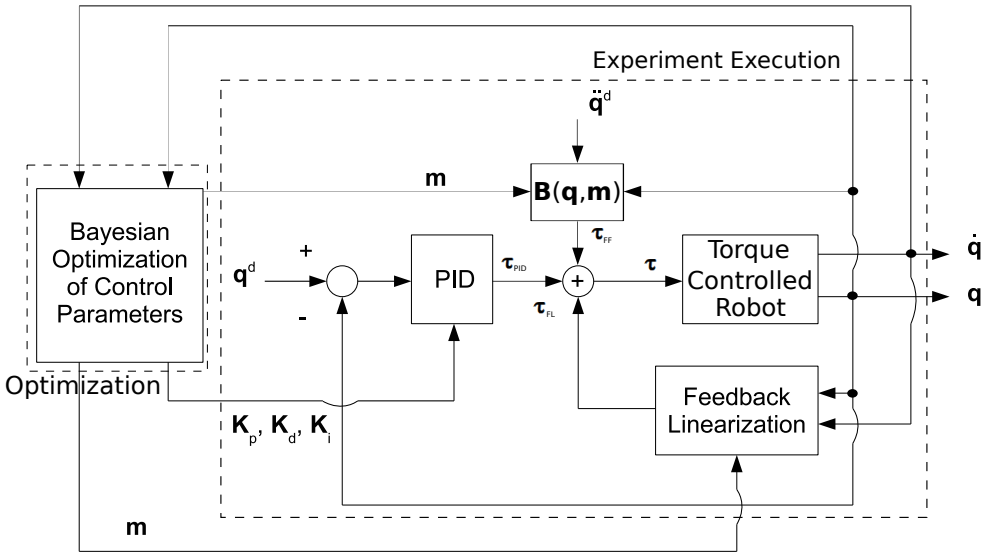

Fig. 2. Proposed architecture of control parameters self-tuning for the trajectory tracking application.

2.2 Control architecture

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

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{FF}} + \boldsymbol{\tau}_{\text{PID}} + \boldsymbol{\tau}_{\text{FL}}, \tag{2}$$

where $\boldsymbol{\tau}_{\text{FF}}$ and $\boldsymbol{\tau}_{\text{PID}}$ implement the feedforward and feedback control action, respectively, and $\boldsymbol{\tau}_{\text{FL}}$ is the action of the feedback linearization controller, namely

$$\boldsymbol{\tau}_{\text{FF}} = \mathbf{B}(\mathbf{q},\mathbf{m})\ddot{\mathbf{q}}^d, \tag{3a}$$

$$\boldsymbol{\tau}_{\text{PID}} = \mathbf{K}_p\mathbf{e}_q + \mathbf{K}_d\dot{\mathbf{e}}_q + \mathbf{K}_i\int \mathbf{e}_q, \tag{3b}$$

$$\boldsymbol{\tau}_{\text{FL}} = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\mathbf{m}) + \mathbf{g}(\mathbf{q},\mathbf{m}), \tag{3c}$$

with $\mathbf{q}^d$ being the joint position reference vector and $\mathbf{e}_q = \mathbf{q}^d - \mathbf{q}$ the joint position error vector.

Note that $\boldsymbol{\tau}_{\text{PID}}$ aims to improve the performance of the trajectory tracking in closed-loop, while $\boldsymbol{\tau}_{\text{FL}}$ compensates for the Coriolis and gravity terms in (1). In order to achieve high trajectory tracking performance, it is important to tune the PID control gains $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i$ and the link-mass parameters $\mathbf{m}$ in (3a) and (3c).

2.3 Objective function

The proposed two-stage methodology aims to optimize the robot dynamics parameters and the robot control parameters separately, in order to have two optimization problems: the first related to the system dynamics identification problem, the second related to the data-driven control design problem. In such a way, the complexity of the optimization is split into two optimization problems, decreasing the number of parameters to be optimized. In fact, due to the high-complexity of the robot dynamic model (*i.e.*, high number of dynamics parameters) and the high number of the control parameters (considering the PID structure, an n joint manipulator is characterized by $3\,x\,n$ control gains), it is convenient to separate the two optimization problems from both conceptual and computational points of view.

In the first optimization stage, under the assumption that no external forces are applied to the robot and by neglecting friction effects at the joint-level, the robot dynamics in (1) reduces to:

$$\mathbf{B}(\mathbf{q},\mathbf{m})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\mathbf{m}) + \mathbf{g}(\mathbf{q},\mathbf{m}) = \boldsymbol{\tau}. \tag{4}$$

The controller is designed with feedback linearization and feedforward action (without PID term) in order to decouple the joint-level robot DoFs, reducing (2) to:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{FL}} + \boldsymbol{\tau}_{\text{FF}} = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}},\mathbf{m}) + \mathbf{g}(\mathbf{q},\mathbf{m}) + \mathbf{B}(\mathbf{q},\mathbf{m})\ddot{\mathbf{q}}^d. \tag{5}$$

Substituting (5) into (4) leads to:

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}^d. \tag{6}$$

Thus, in the ideal case, the actual joint accelerations are equal to the target feedforward accelerations. Based on (6), the performance of the feedback linearization with feedforward action is measured in terms of the following closed-loop objective function:

$$J_{\text{FL}}(\mathbf{m}) = \sum_{i=1}^{n} (\ddot{e}_i^{\max} + \ddot{e}_i^{\text{mean}}) + L, \tag{7}$$

where n is the number of robot DoFs, $\ddot{e}_i^{\max}$ and $\ddot{e}_i^{\text{mean}}$ are the maximum and the average, respectively, of the absolute value of the i-th joint acceleration error over the execution time T, *i.e.*,

$$\ddot{e}_i^{\max} = \max_{t\in[0,\,T]} |\ddot{\mathbf{e}}_{q,i}(t)|, \tag{8}$$

$$\ddot{e}_i^{\text{mean}} = \frac{1}{T}\int_0^T |\ddot{\mathbf{e}}_{q,i}(t)|dt, \tag{9}$$

where $\mathbf{e}_{q,i}$ denoting the i-th element of the vector $\mathbf{e}_q$, namely, the joint position error of the i-th joint, and $\ddot{\mathbf{e}}_{q,i}$ denoting its acceleration-level term.

The term L in (7) 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:

$$L = 100\,e^{-\xi/T} \text{ if } |\mathbf{e}_{q,i}(\xi)| > \bar{e}, \tag{10a}$$

$$L = 1000\,e^{-\bar{t}/T} \text{ if test interrupted}, \tag{10b}$$

where $\xi \leq T$ is the time when the constraint on the maximum joint position error is violated, and $\bar{t} \leq T$ 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).

The cost (7) is then minimized w.r.t. the link-mass parameters $\mathbf{m}$ through BO reviewed in Section 3.

Remark 1. Since the proposed feedback linearization structure (3c) only considers link-masses, the number of parameters to be tuned in the first stage is smaller than the number of PID gains. However, in general, the feedback linearization may include other terms modeling, for instance, friction, joint and link elasticity, *etc*. ■

In the second optimization stage, once the link-mass parameters $\mathbf{m}$ are computed, the trajectory tracking error is minimized. 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:

$$J_{\text{FB}} = \sum_{i=1}^{n} (e_i^{\max} + \dot{e}_i^{\max} + e_i^{\text{mean}} + \dot{e}_i^{\text{mean}}) + L, \tag{11}$$

where the penalty L is defined as in (10). Eq. (11) has the same structure as (7), but uses the position-level errors $e_i^{\max}$ and e_i^{mean} (the maximum and the average, respectively, of the absolute value of the i-th joint position error over the execution time T)

$$e_i^{\max} = \max_{t\in[0,\ T]} |\mathbf{e}_{q,i}(t)|, \tag{12}$$

$$e_i^{\text{mean}} = \frac{1}{T}\int_0^T |\mathbf{e}_{q,i}(t)|dt, \tag{13}$$

and velocity-level errors $\dot{e}_i^{\max}$ and $\dot{e}_i^{\text{mean}}$, computed similarly to position errors, are considered.

The cost (11) is then minimized w.r.t. the PID gains $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i$ through BO.

3. BAYESIAN OPTIMIZATION FOR PARAMETERS TUNING

Using the controller structure described in Section 2.2, the cost functions J_{FL} in (7) and J_{FB} in (11) depend on tunable *design parameters*, namely, the link-mass parameters $\mathbf{m}$ and the PID gains $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i$, respectively.

Let us consider the generic cost function J and its related design parameters. By collecting all these design parameters in a vector $\boldsymbol{\theta}$, the tuning task reduces to the minimization of the cost $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, within a space of admissible values $\boldsymbol{\Theta}$. However, a closed-form expression of the cost J as a function of the design parameter vector $\boldsymbol{\theta}$ 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 $\boldsymbol{\theta}_i$, 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 of noisy observations of the "true" cost function, namely $J_i = J(\boldsymbol{\theta}_i) + n_i$, 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 $J(\boldsymbol{\theta})$ 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, BO is generally the most efficient in terms of number of function evaluations (Brochu et al., 2010; Jones et al., 1998) 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 $\boldsymbol{\theta}_i$ and the corresponding cost J_i is measured. Then, all the past parameter-cost observations $\mathcal{D}_i = \{(\boldsymbol{\theta}_1, J_1), (\boldsymbol{\theta}_2, J_2), \ldots, (\boldsymbol{\theta}_i, J_i)\}$ are processed and a new parameter $\boldsymbol{\theta}_{i+1}$ to be tested at the next experiment is computed according to the approach discussed in the following.

3.1 Surrogate model

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

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

$$p(J(\boldsymbol{\theta})|\mathcal{D}_i) = \frac{p(\mathcal{D}_i|J(\boldsymbol{\theta}))p(J(\boldsymbol{\theta}))}{p(\mathcal{D}_i)}. \tag{14}$$

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 (Jones et al., 1998). Intuitively, the probabilistic model $p(J(\boldsymbol{\theta})|\mathcal{D}_i)$ should assign low variance for parameters $\boldsymbol{\theta}$ 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) (Rasmussen and Williams, 2006), 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 $\mu(\cdot) : \boldsymbol{\Theta} \to \mathbb{R}$ and covariance function $\kappa(\cdot,\cdot) : \boldsymbol{\Theta} \times \boldsymbol{\Theta} \to \mathbb{R}$. The latter describes prior assumptions on the correlation between two values $J(\boldsymbol{\theta}_1)$ and $J(\boldsymbol{\theta}_2)$, and implicitly models smoothness of J w.r.t. $\boldsymbol{\theta}$.

3.2 Acquisition function

BO exploits the model's uncertainty information on the posterior distribution $p(J(\boldsymbol{\theta})|\mathcal{D}_i)$ to determine the most promising point to be tested in the next iteration. Specifically, next point $\boldsymbol{\theta}_{i+1}$ is chosen taking into account the trade off between *exploitation*, choosing points where the value of the performance index $J(\boldsymbol{\theta})$ is expected to be optimal, and *exploration*, choosing

points in unexplored regions of the feasible set $\boldsymbol{\Theta}$ where the variance of $J(\boldsymbol{\theta})$ is high (Brochu et al., 2010).

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

$$\boldsymbol{\theta}_{i+1} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathcal{A}(\boldsymbol{\theta}). \tag{15}$$

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

$$\mathcal{A}(\boldsymbol{\theta}) = \mathbf{EI}(\boldsymbol{\theta}) = \mathbb{E}\left[\max\{0, J_i^- - J(\boldsymbol{\theta})\}\right], \tag{16}$$

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

$$J_i^- = \min_{j=1,\dots,i} J_j,$$

and the expectation in (16) is taken over the posterior distribution $p(J(\boldsymbol{\theta})|\mathcal{D}_i)$.

The acquisition function $\mathbf{EI}(\boldsymbol{\theta})$ is thus the expected value of the objective value improvement (with respect to the best in the dataset $\mathcal{D}_i$) that could be achieved for the point $\boldsymbol{\theta}$. Besides having this intuitive interpretation, expected improvement was found to perform well in practice, providing a good balance between exploration and exploitation (Brochu et al., 2010). Furthermore, it has an analytic expression in terms of the posterior mean and variance of the GP process (Jones et al., 1998) and can thus be optimized using standard gradient-based techniques.

3.3 Algorithm outline

The overall BO procedure for autotuning of the controller parameters $\boldsymbol{\theta}$ is summarized in the following. First, an initial dataset of observations $\mathcal{D}_{n_{\text{in}}}$ is constructed by performing n_{in} initial experiments with different parameters $\boldsymbol{\theta}_j \in \boldsymbol{\Theta}$ and measuring the corresponding performance J_j, for $j = 1, \dots, i$. The initial parameters may be just randomly chosen within in the admissible range $\boldsymbol{\Theta}$.

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

An experiment is performed with parameter $\boldsymbol{\theta}_{i+1}$ and the corresponding performance index J_{i+1} is measured. Finally, the dataset $\mathcal{D}_{i+1}$ is constructed by augmenting $\mathcal{D}_i$ 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.

4. EXPERIMENTAL RESULTS

4.1 Setup description

The FRANKA Emika manipulator has been used as test platform, implementing a 1-kHz torque control. The number of link-mass parameters $\mathbf{m}$ characterizing the feedback $\boldsymbol{\tau}_{\text{FL}}$ and the feedforward $\boldsymbol{\tau}_{\text{FF}}$ 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: link-mass parameters are related to the first optimization stage, PID gains are related to the second optimization stage. Robot dynamics has been computed using the C++ *KDL* library (Smits et al., 2013). BO has been performed using the C++ *limbo* (Cully et al., 2016) and the *NLopt* (Johnson, 2019) libraries.

4.2 Use case trajectories

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

- sinusoidal joint trajectory (Traj. #1)

  $$\mathbf{q}_i(t) = \mathbf{q}_i^0 + \mathbf{A}_i\left(1 - \cos(2\pi \mathbf{f}_i t)\right),$$

  where $\mathbf{f} = [0.05 \;\; 0.075 \;\; 0.075 \;\; 0.1 \;\; 0.1 \;\; 0.1 \;\; 0.1]$ Hz, $\mathbf{A} = [30 \;\; 10 \;\; 10 \;\; 10 \;\; 10 \;\; 10 \;\; 10]^\circ$, and $i = 1, \dots, 7$ identifies the joint. The execution time for the trajectory is $T = 20$ s;
- circular Cartesian trajectory (Traj. #2) in the $y-z$ plane

  $$\begin{aligned} y(t) &= y(t-dt) + r\left(1 - \cos(\phi(t))\right), \\ z(t) &= z(t-dt) + r\sin(\phi(t)), \end{aligned}$$

  where $r = 0.05$ m, $v = 0.1$ m/s, and $\phi(t) = \frac{v}{r} + \phi(t-\Delta t)\,\Delta t$ are the radius, the tangential velocity, and the angular position of the circular path, respectively. Other Cartesian DoFs are kept constant in the reference Cartesian trajectory. Franka embedded inverse kinematics is used to compute the reference joint trajectory. The sampling time Δt is equal to 1 ms and the execution time of the trajectory is $T = 7$ s;
- sinusoidal Cartesian trajectory (Traj. #3)

  $$\begin{aligned} x(t) &= x^0 + A_x\left(1 - \cos(2\pi f_x t)\right), \\ y(t) &= y^0 + A_y\left(1 - \cos(2\pi f_y t)\right), \\ z(t) &= z^0 + A_z\left(1 - \cos(2\pi f_z t)\right), \end{aligned}$$

  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.

4.3 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 BO algorithm, the following constraints are imposed:

- link-mass parameters $\mathbf{m}$ 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 $\mathbf{K}_p$, $\mathbf{K}_d$ and $\mathbf{K}_i$ 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, $K_{d,j} \in [0,\ 100]$ 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 $\mathbf{m}$ and the PID gains $\mathbf{K}_p$, $\mathbf{K}_d$ and $\mathbf{K}_i$ are tuned through the two-stage procedure described in Section 2.3.

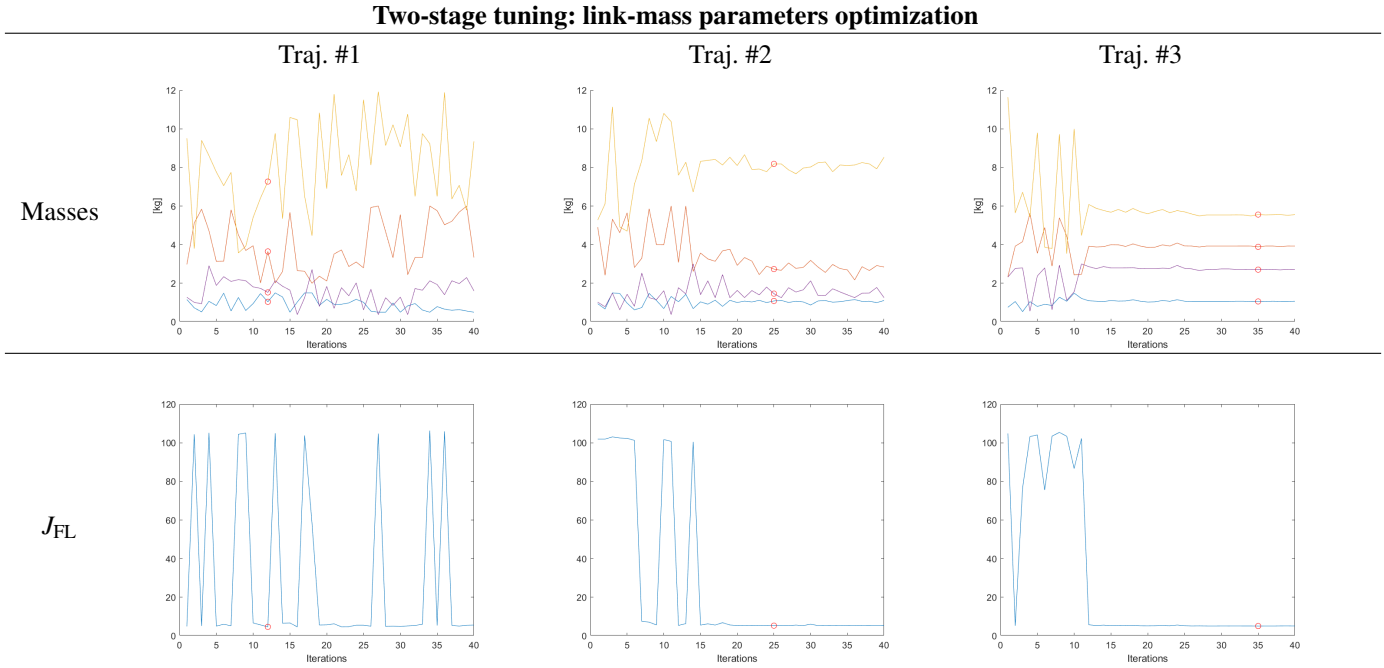

Fig. 3. Achieved results for the three trajectories: link-mass parameters $\mathbf{m}$ (first row column, m_1 blue line, m_2 red line, m_3 yellow line, m_4 purple line); and objective function J_{FL} (second row) *vs* BO iterations. Optimal iteration is highlighted by a red circle.

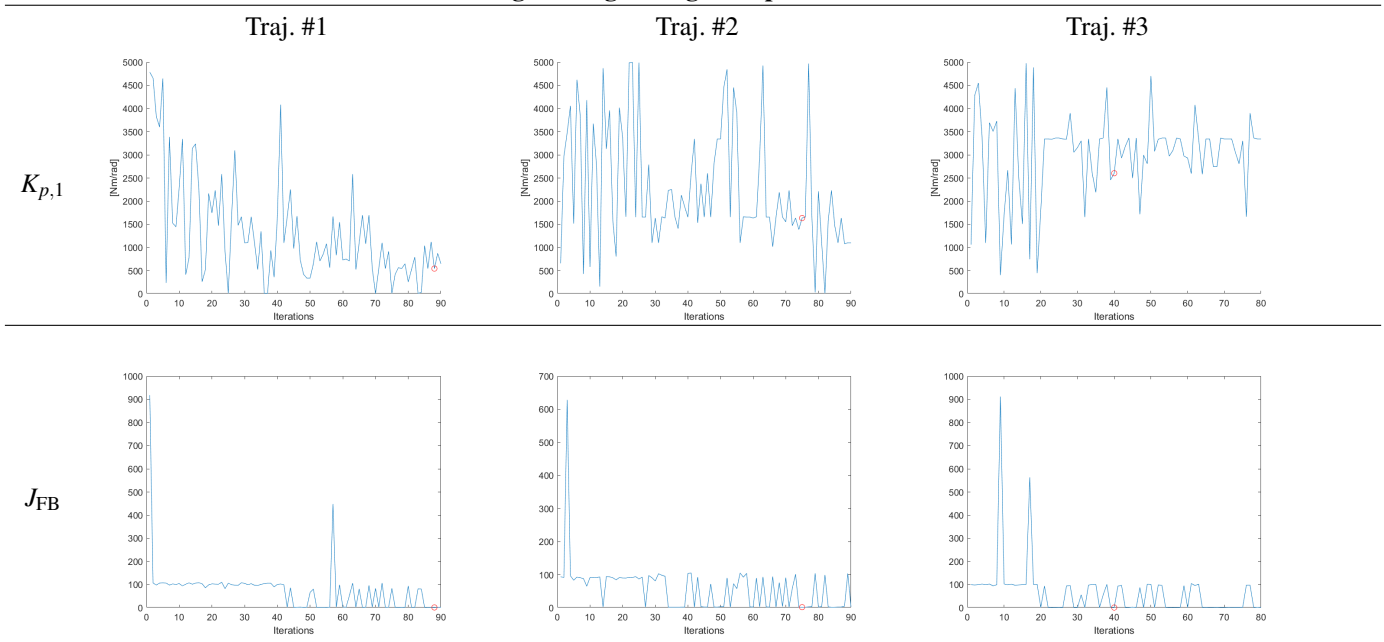

Fig. 4. Achieved results for the three trajectories: PID gains $K_{p,1}$ (first row); and objective function J_{FB} (second row) *vs* BO iterations. Optimal iteration is highlighted by a red circle.

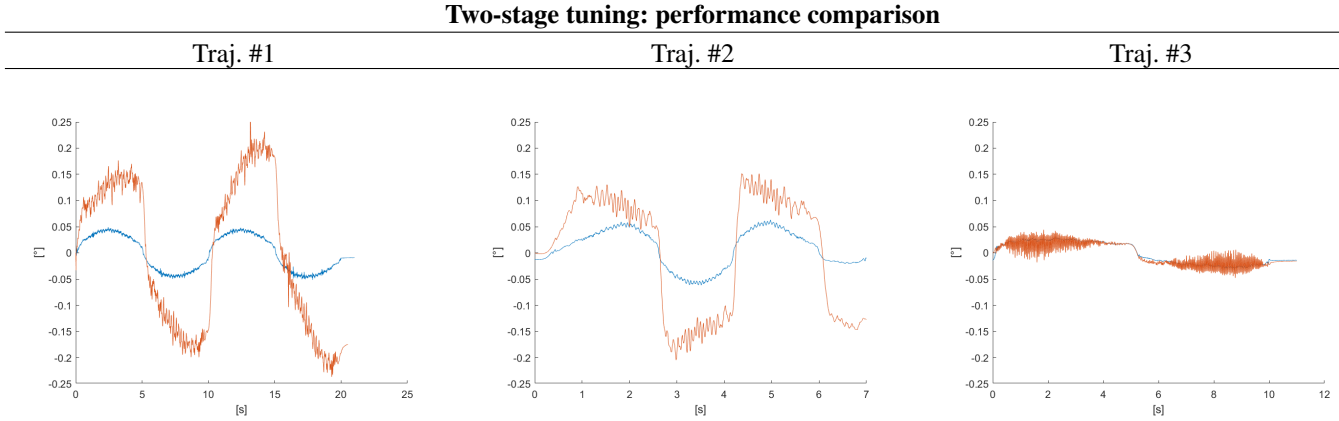

Fig. 5. Trajectory tracking errors for joint 7 (highest tracking errors) achieved by the FRANKA Emika controller (blue line) and by the two-stage tuning procedure (red line).

At the first stage, the equivalent link-mass parameters $\mathbf{m}$ characterizing the feedback linearization and the feedforward action are tuned by minimizing the performance cost J_{FL} in (7). The PID gains are not optimized at this stage. In particular, PID derivative and integral gains are set to zero, while the proportional gains $K_{p,j}$ are set to low values (specifically, 75 Nm/rad for all joints $j = 1,\dots,7$) just to avoid drifts due to over/under compensations of the robot dynamics.

The BO is initialized with $n_{\text{in}} = 10$ initial experiments with randomly generated parameters $\mathbf{m}$, and terminated after 40 iterations. Tests are interrupted if the joint position error $\mathbf{e}_{q,i}(t)$ is larger than $\bar{e} = 5^\circ$ or safety constraints are violated. The achieved results are summarized in Fig. 3, which shows the evolution of the link-mass parameters $\mathbf{m}$ characterizing the feedback and feedforward actions, as well as the performance index J_{FL} *vs* the BO iterations.

The tests are stopped 9 times (out of 40) for the sinusoidal joint trajectory; 8 times for the circular Cartesian; and 23 times for the sinusoidal Cartesian trajectory. As it can be also inferred from the cost J_{FL} in Fig. 3, the experiments are stopped mainly in the first 10 initial iterations.

At the second stage, once the link-mass parameters are optimized, the PID controllers are designed. The BO is initialized with $n_{\text{in}} = 20$ initial experiments with randomly generated PID parameters, and terminated after 90 iterations. Tests are interrupted if the joint position error $\mathbf{e}_{q,i}(t)$ is larger than $\bar{e} = 1.5^\circ$ or safety constraints are violated. The achieved results are summarized in Fig. 4. For the sake of visualization, only the proportional gains for joint 1 are plotted, together with the performance cost J_{FB} over the algorithm's iterations.

The tests are stopped 58 times (out of 90) for the sinusoidal joint trajectory, 48 times for the circular Cartesian, and 26 times for the sinusoidal Cartesian trajectory. As it can be also inferred from the cost J_{FB} in Fig. 4, the experiments are stopped mainly in the first iterations.

4.4 Comparison with embedded position controller

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

Fig. 5 shows the joint trajectory errors achieved by the proposed controller design methodology for joint 7 is shown, *i.e.*, the most critical; similar tracking errors are obtained for other joints. The maximum tracking error is less then 0.25°. The same figure also shows the error achieved with the FRANKA Emika embedded joint position controller. Although the FRANKA Emika controller achieves slightly better performance, the advantage of the proposed approach is in its limited number of experiments required to auto-tune a high-performance control. Trajectory tracking performance of the proposed algorithm can be improved including advanced dynamics compensation (*e.g.*, friction or joint elasticity terms) into the control law and/or designing more advanced trajectory tracking controllers (*e.g.*, optimal control).

5. CONCLUSIONS

A two-stage BO based approach tailored for auto-tuning of low-level robot trajectory tracking controller with unknown dynamics has been presented. The employed control architecture, consisting in a feedback linearization, a feedforward action, and PID controllers at the joint level, is widely used on industrial manipulators, therefore, easily implementable. Firstly, the robot dynamic parameters are tuned to compensate for the robot dynamics. Then, 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 in the two stafe approach 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.*, 40 iterations for the first stage, 90 iterations for the second stage), 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, together with the use of additional dynamics parameters in the first optimization stage (*e.g.*, friction parameters).

REFERENCES

Antsaklis, P.J. and Rahnama, A. (2018). Control and machine intelligence for system autonomy. *Journal of Intelligent & Robotic Systems*, 91(1), 23–34.

Arulkumaran, K., Deisenroth, M.P., Brundage, M., and Bharath, A.A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6), 26–38.

Balatti, P., Kanoulas, D., Rigano, G.F., Muratorc, L., Tsagarakis, N.G., and Ajoudani, A. (2018). A self-tuning impedance controller for autonomous robotic manipulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5885–5891.

Bansal, S., Calandra, R., Xiao, T., Levine, S., and Tomlin, C.J. (2017). Goal-driven dynamics learning via bayesian optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 5168–5173.

Brochu, E., Cora, V.M., and De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint:1012.2599*.

Bruzzone, A.G., Massei, M., Di Matteo, R., and Kutej, L. (2018). Introducing intelligence and autonomy into industrial robots to address operations into dangerous area. In *International Conference on Modelling and Simulation for Autonomous Systesm*, 433–444. Springer.

Calandra, R., Ivaldi, S., Deisenroth, M.P., Rueckert, E., and Peters, J. (2015). Learning inverse dynamics models with contacts. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 3186–3191.

Cully, A., Chatzilygeroudis, K., Allocati, F., and Mouret, J.B. (2016). Limbo: A fast and flexible library for bayesian optimization. *arXiv preprint:1611.07343*.

Cully, A., Clune, J., Tarapore, D., and Mouret, J.B. (2015). Robots that can adapt like animals. *Nature*, 521(7553), 503.

Drieß, D., Englert, P., and Toussaint, M. (2017). Constrained bayesian optimization of combined interaction force/task space controllers for manipulations. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 902–907.

Finn, C., Goodfellow, I., and Levine, S. (2016). Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, 64–72.

Formentin, S., Piga, D., Tóth, R., and Saveresi, S.M. (2016). Direct learning of LPV controllers from data. *Automatica*, 65, 98–110.

Galicki, M. (2016). Finite-time trajectory tracking control in a task space of robotic manipulators. *Automatica*, 67, 165–170.

Hernández-Alvarado, R., García-Valdovinos, L., Salgado-Jiménez, T., Gómez-Espinosa, A., and Fonseca-Navarro, F. (2016). Neural network-based self-tuning PID control for underwater vehicles. *Sensors*, 16(9), 1429.

Hsiao, T. and Huang, P.H. (2017). Iterative learning control for trajectory tracking of robot manipulators. *International Journal of Automation and Smart Technology*, 7(3), 133–139.

Jaisumroum, N., Chotiprayanakul, P., and Limnararat, S. (2016). Self-tuning control with neural network for robot manipulator. In *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, 1073–1076.

Jin, J. and Gans, N. (2015). Parameter identification for industrial robots with a fast and robust trajectory design approach. *Robotics and Computer-Integrated Manufacturing*, 31, 21–29.

Johnson, S.G. (2019). The nlopt nonlinear-optimization package. `http://github.com/stevengj/nlopt`.

Jones, D.R., Schonlau, M., and Welch, W.J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4), 455–492.

Makridakis, S. (2017). The forthcoming artificial intelligence (AI) revolution: Its impact on society and firms. *Futures*, 90, 46–60.

Modares, H., Ranatunga, I., Lewis, F.L., and Popa, D.O. (2015). Optimized assistive human–robot interaction using reinforcement learning. *IEEE transactions on cybernetics*, 46(3), 655–667.

Mollard, Y., Munzer, T., Baisero, A., Toussaint, M., and Lopes, M. (2015). Robot programming from demonstration, feedback and transfer. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1825–1831.

Piga, D., Forgione, M., Formentin, S., and Bemporad, A. (2019). Performance-oriented model learning for data-driven MPC design. *IEEE Control Systems Letters*, 3(3), 577–582. doi:10.1109/LCSYS.2019.2913347.

Piga, D., Formentin, S., and Bemporad, A. (2018). Direct data-driven control of constrained systems. *IEEE Transactions on Control Systems Technology*, 25(4), 331–351.

Pinto, L. and Gupta, A. (2016). Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, 3406–3413.

Rasmussen, C.E. and Williams, C.K. (2006). *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA.

Roveda, L., Pallucca, G., Pedrocchi, N., Braghin, F., and Tosatti, L.M. (2018). Iterative learning procedure with reinforcement for high-accuracy force tracking in robotized tasks. *IEEE Transactions on Industrial Informatics*, 14(4), 1753–1763.

Rozo, L.D., Calinon, S., Caldwell, D., Jiménez, P., and Torras, C. (2013). Learning collaborative impedance-based robot behaviors. In *27th AAAI Conference on Artificial Intelligence*.

Siciliano, B. and Villani, L. (2000). *Robot Force Control*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition.

Smits, R., Bruyninckx, H., and Aertbeliën, E. (2013). Kdl: Kinematics and dynamics library (2001).

Swevers, J., Verdonck, W., and De Schutter, J. (2007). Dynamic model identification for industrial robots. *IEEE control systems magazine*, 27(5), 58–71.

Venkatraman, A., Capobianco, R., Pinto, L., Hebert, M., Nardi, D., and Bagnell, J.A. (2016). Improved learning of dynamics models for control. In *International Symposium on Experimental Robotics*, 703–713. Springer.

Yuan, K., Chatzinikolaidis, I., and Li, Z. (2019). Bayesian optimization for whole-body control of high degrees of freedom robots through reduction of dimensionality. *IEEE Robotics and Automation Letters*.

Zhao, J., Han, L., Wang, L., and Yu, Z. (2016). The fuzzy pid control optimized by genetic algorithm for trajectory tracking of robot arm. In *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, 556–559.