Control Engineering Practice 101 (2020) 104488

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Robot control parameters auto-tuning in 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

ARTICLE INFO

Keywords:
Controller auto-tuning
Bayesian optimization
Trajectory tracking
Dynamics compensation
Industrial robots

ABSTRACT

Autonomy is increasingly demanded to industrial manipulators. Robots have to be capable to regulate their behavior to different operational conditions, adapting to the specific task to be executed without requiring high time/resource-consuming human intervention. Achieving an automated tuning of the control parameters of a manipulator is still a challenging task, which involves modeling/identification of the robot dynamics. This usually results in an onerous procedure, both in terms of experimental and data-processing time. This paper addresses the problem of automated tuning of the manipulator controller for trajectory tracking, optimizing control parameters based on the specific trajectory to be executed. A Bayesian optimization algorithm is proposed to tune both the low-level controller parameters (*i.e.*, the equivalent link-masses of the feedback linearizator and the feedforward controller) 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 error are also included. The performance of 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 less than 130 iterations, and comparable results with respect to the FRANKA Emika embedded position controller are achieved. In addition, the generalization capabilities of the proposed approach are shown exploiting the proper reference trajectory for the tuning of the control parameters.

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 which allows them to face unforeseen situations (Makridakis, 2017). Such capabilities are in particularly needed by industrial manipulators, which are expected to execute highly dynamic tasks (Thoben, Wiesner, & Wuest, 2017; Wong, Yang, Yan, & Gu, 2017). This requires 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. In fact, human intervention can be time- and resources-consuming and not always feasible (*e.g.*, for safety reasons) (Bruzzone, Massei, Di Matteo, & Kutej, 2018; Pichler et al., 2017; Starke, Hahn, Pedroza Yanez, & Ugalde Leal, 2016).

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.

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 & Gans, 2015; Swevers, Verdonck, & De Schutter, 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. Direct methods, such as Iterative feedback tuning (IFT) (Hjalmarsson, Gevers, Gunnarsson, & Lequin, 1998) and Virtual reference feedback tuning (VRFT) (Formentin, Piga, Tóth, & Savaresi, 2016; Lecchini, Campi, & Savaresi, 2002), could be used to design a controller based on open-loop data, without identifying a model of the system. However, these methodologies require to a-priori specify a model of the desired closed-loop behavior, which might not be achieved by the chosen controller structure. Although some techniques have been recently proposed in Novara, Formentin, Savaresi,

* Corresponding author.
E-mail addresses: loris.roveda@idsia.ch (L. Roveda), marco.forgione@idsia.ch (M. Forgione), dario@idsia.ch (D. Piga).

https://doi.org/10.1016/j.conengprac.2020.104488
Received 19 September 2019; Received in revised form 20 April 2020; Accepted 28 May 2020
Available online 8 June 2020
0967-0661/

and Milanese (2016), Piga, Formentin, and Bemporad (2018) and Selvi, Piga, and Bemporad (2018) to overcome these limitations, the a-priori chosen closed-loop reference model and the controller parameterization may still affect the final closed-loop performance. In addition, the final performance of the closed-loop system may be degraded by the noise corrupting the data used for direct controller design.

In recent years, *machine learning* is emerging as an alternative paradigm for data-driven robot control design (Antsaklis & Rahnama, 2018; Arulkumaran, Deisenroth, Brundage, & Bharath, 2017). Programming-by-demonstration approaches are proposed in Mollard, Munzer, Baisero, Toussaint, and Lopes (2015) and Rozo, Calinon, Caldwell, Jiménez, and Torras (2013), in which the human is physically teaching a specific task to the manipulator, that simultaneously learn dynamic behaviors and task execution. Additionally, autonomous learning of robot tasks are also deeply investigated (Pinto & Gupta, 2016).

Machine learning techniques are commonly used to tackle different control learning problems. On the one hand, data-driven modeling 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, Calandra, Xiao, Levine, & Tomiin, 2017; Calandra, Ivaldi, Deisenroth, Rueckert, & Peters, 2015; Finn, Goodfellow, & Levine, 2016; Piga, Forgione, Formentin, & Bemporad, 2019; Piga et al., 2018; 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, Ranatunga, Lewis, and Popa (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, Chotiprayanakul, and Limnararat (2016) investigates a Neural Network approach to self-tune the robot controller in object balancing applications; Roveda, Pallucca, Pedrocchi, Braghin, and Tosatti (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. For instance, Hernández-Alvarado, García-Valdovinos, Salgado-Jiménez, Gómez-Espinosa, and Fonseca-Navarro (2016) proposes an auto-tuned PID-like controller based on Neural Networks; Zhao, Han, Wang, and Yu (2016) presents a fuzzy PID controller optimized by genetic algorithms; Galicki (2016) discusses absolutely continuous Jacobian transpose robust controllers for finite-time trajectory tracking control in a task-space under dynamic uncertainties; Van Cuong and Nan (2016) proposes a neural-network controller for an n-link robot manipulator with robust compensator to achieve a high-precision position tracking; Hsiao and Huang (2017) presents an iterative learning controller enhanced by a disturbance observer to robustly linearize the dynamics of robot manipulators. In general, these approaches are commonly based on computational demanding algorithms, also requiring advanced programming skills. Furthermore, control tuning is in general too time- and cost-consuming to be affordable in industrial environments. In addition, safety constraints are not explicitly taken into account in the control design, so that the robot may show an unstable/unsafe behavior.

In order to develop an efficient and easily-applicable 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. In fact, BO is known to be efficient in terms of number of function evaluations for derivative-free optimization in comparison with other methods, such as Particle Swarm Optimization and Genetic Algorithms (Brochu, Cora, & De Freitas, 2010; Jones, Schonlau, & Welch, 1998; Letham, Karrer, Ottoni, Bakshy, et al., 2019; Pelikan, Goldberg, & Cantú-Paz, 1999; Snoek, Larochelle, & Adams, 2012). A simple PID trajectory control loop (probably the most applied control strategy in real applications Ang, Chong, & Li, 2005) 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 which guides the control parameters tuning. Moreover, a penalty is given to the control parameters leading to tracking errors exceeding a given threshold or to unstable/unsafe behaviors resulting in a safety stop of the experiment.

Two strategies are proposed for parameter tuning. In the first strategy, the equivalent link-mass parameters and the PID gains are jointly optimized according to the trajectory-tracking objective using Bayesian optimization. The second strategy is inspired by classical control design strategies for manipulators and consists of two stages. In the first stage, the link masses characterizing the feedback linearization and the feedforward action are tuned with the objective of decoupling the joint-level robot degree-of-freedom (DoFs) by compensating the gravitational and the Coriolis force. In the second stage, PID gains are tuned according to the original trajectory-tracking objective, while keeping link mass parameters fixed to the values optimized in the first stage. Both in the first and in the second stage, Bayesian optimization is used to tune the control parameters from experiments.

For the sake of completeness, it is worth mentioning other works applying Bayesian optimization in robotics applications. In particular, (Drieß, Englert, & Toussaint, 2017) employs constrained BO to select three force-controller parameters for combined position/interaction tasks; (Cully, Clune, Tarapore, & Mouret, 2015) describes a trial-and-error algorithm that allows robots to adapt their behavior in presence of damage; (Yuan, Chatzinikolaidis, & Li, 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, which relies on a simple feedback linerizator for dynamics compensation, a feedforward action, and PID controllers at the joint level. This increases the degrees of freedom in the control design, maintaining a gray-box tuning of the controller parameters. As described in Section 4, this also allows us to design the parameters characterizing the dynamics of the robot and the PID controllers sequentially, by splitting the control design problem into two simpler problems.

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 *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 less than 130 trials, and comparable performance with the FRANKA Emika embedded position controller is achieved. In addition, the generalization capabilities of the proposed approach are shown exploiting the proper reference trajectory for the tuning of the control parameters (*i.e.*, properly exciting the robot dynamics).

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

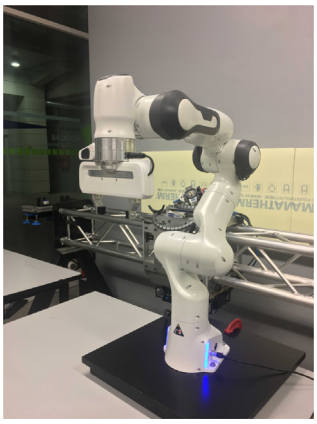


Fig. 1. The FRANKA Emika manipulator used as a test platform to evaluate the proposed procedures for control parameters auto-tuning.

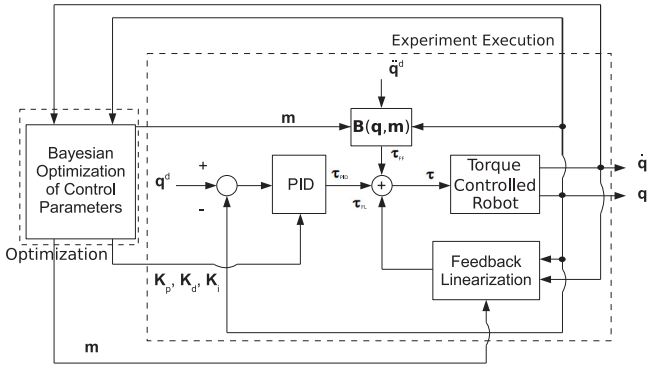


Fig. 2. Proposed architecture of self-tuning for trajectory tracking.

the details of the BO algorithm used to tune the control parameters. In Section 4 a two-stage version of the BO-based tuning algorithm is described. The experimental results obtained in controlling the FRANKA Emika manipulator are reported in Section 5. Conclusions and directions for future works are given in Section 6.

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 linearizator and feedforward controller are optimized through Bayesian optimization 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 & 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 $\mathbf{q}$ with respect to time.

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 supposed to be 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.

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 therefore 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 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:

$$J = \sum_{i=1}^{n}\left(e_i^{\max} + \dot{e}_i^{\max} + e_i^{\text{mean}} + \dot{e}_i^{\text{mean}}\right) + L, \tag{4}$$

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

$$e_i^{\max} = \max_{t\in[0,\,T]} |\mathbf{e}_{q,i}(t)|, \tag{5}$$

$$e_i^{\text{mean}} = \frac{1}{T}\int_0^T |\mathbf{e}_{q,i}(t)|dt, \tag{6}$$

with $\mathbf{e}_{q,i}$ denoting the ith element of the vector $\mathbf{e}_q$, namely, the joint position error of the ith joint. The terms $\dot{e}_i^{\max}$ and $\dot{e}_i^{\text{mean}}$ in (4) are defined similarly for the velocity error, obtained by numerical differentiation of the joint position signals using a filter with cut-off frequency of 50 Hz.

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:

$$L = 100\, e^{-\xi/T} \;\text{ if }\; |\mathbf{e}_{q,i}(\xi)| > \bar{e}, \tag{7a}$$

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

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

3. Bayesian optimization for parameters tuning

The robot performance is quantified in terms of the cost function J in (4). Using the controller structure described in Section 2.2, the cost J depends on tunable *design parameters*, namely, the PID gains $\mathbf{K}_p$, $\mathbf{K}_d$, $\mathbf{K}_i$ and the link-mass parameters $\mathbf{m}$. After collecting all these design parameters in a vector $\boldsymbol{\theta}$, their tuning 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 J 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 in a noisy observations of the "true" cost function, namely

$$J_i = J(\boldsymbol{\theta}_i) + n_i, \tag{8}$$

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, *Bayesian optimization* (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*).

By using BO, the cost J is simultaneously learnt and optimized by sequentially performing experiments on the robot. Specifically, at each iteration i of the BO 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 $\mathcal{D}_i = \{(\boldsymbol{\theta}_1, J_1), (\boldsymbol{\theta}_2, J_2), \dots, (\boldsymbol{\theta}_i, J_i)\}$ 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. Such a probabilistic 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 $\mathcal{D}_i$ gathered up to iteration i is computed via Bayesian inference, *i.e.*,

$$p(J(\boldsymbol{\theta})|\mathcal{D}_i) = \frac{p(\mathcal{D}_i|J(\boldsymbol{\theta}))p(J(\boldsymbol{\theta}))}{p(\mathcal{D}_i)}. \tag{9}$$

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 & Williams, 2006), which is a flexible and non-parametric model allowing one to describe arbitrarily complex function classes. In a GP modeling 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 cost values $J(\boldsymbol{\theta}_1)$ and $J(\boldsymbol{\theta}_2)$, and implicitly models smoothness of J w.r.t. $\boldsymbol{\theta}$. A simple commonly used covariance function $\kappa(\cdot,\cdot)$ is the *Squared Exponential* defined as

$$\kappa(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \sigma_f^2 e^{-\frac{\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|^2}{2\lambda^2}}, \tag{10}$$

with hyper-parameters $\sigma_f, \lambda \in \mathbb{R}$.

Performing Bayesian inference on a GP model only requires closed-form matrix operations. Under the assumption that the prior mean $\mu(\cdot)$ is equal to 0 and n_i in (8) is a white zero-mean Gaussian noise with variance σ_n^2, the posterior distribution of $J(\boldsymbol{\theta})$ given evidence $\mathcal{D}_i$ is Gaussian with mean

$$\mu_i(\boldsymbol{\theta}) = \mathbf{k}_i(\boldsymbol{\theta})' \left(\mathbf{K}_i + \sigma_n^2 I\right)^{-1} J_{1:i} \tag{11a}$$

and variance

$$\sigma_i^2(\boldsymbol{\theta}) = \kappa(\boldsymbol{\theta}, \boldsymbol{\theta}) - \mathbf{k}_i(\boldsymbol{\theta})' \left(\mathbf{K}_i(\boldsymbol{\theta}) + \sigma_n^2 I\right)^{-1} \mathbf{k}_i + \sigma_n^2, \tag{11b}$$

where the jth component of the column vector $\mathbf{k}_i(\boldsymbol{\theta}) \in \mathbb{R}^i$ is $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_j)$ (for $j = 1, \dots, i$); the $[j,h]$-th entry of the Kernel matrix $\mathbf{K}_i \in \mathbb{R}^{i \times i}$ is $\kappa(\boldsymbol{\theta}_j, \boldsymbol{\theta}_h)$; the jth component of the column vector $J_{1:i} \in \mathbb{R}^i$ is the observation J_j; the symbol $'$ denotes the transpose operator; and I is an identity matrix of proper dimension.

The unknown hyper-parameters σ_f and λ defining the Squared Exponential in (10), as well as the noise variance σ_n^2, can be estimated by maximizing (with respect to σ_f, λ, and σ_n) the marginal log-likelihood of the observations (Rasmussen & Williams, 2006)

$$\begin{aligned}\log p(J_{1:i}|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_i, \sigma_f, \lambda, \sigma_n) = \\ = -\frac{1}{2}\log\det\left(\mathbf{K}_i + \sigma_n^2 I\right) - \frac{1}{2} J_{1:i}' \left(\mathbf{K}_i + \sigma_n^2 I\right)^{-1} J_{1:i} - \frac{i}{2}\log(2\pi).\end{aligned} \tag{12}$$

3.2. Acquisition function

Bayesian optimization exploits 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 the so-called *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 ith iteration. Next point $\boldsymbol{\theta}_{i+1}$ is thus chosen as

$$\boldsymbol{\theta}_{i+1} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{A}(\boldsymbol{\theta}). \tag{13}$$

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

$$\mathcal{A}(\boldsymbol{\theta}) = \mathbf{EI}(\boldsymbol{\theta}) = \mathbb{E}\left[\max\{0, J_i^- - J(\boldsymbol{\theta})\}\right], \tag{14}$$

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

$$J_i^- = \min_{j=1,\dots,i} J_j, \tag{15}$$

and the expectation in (14) 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 improvement of the objective value (with respect to the best in the

dataset $\mathcal{D}_i$) that could be achieved for the point θ. 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 a closed-form expression for a GP model in terms of the posterior mean and variance (Jones et al., 1998):

$$\mathbf{EI}(\theta) = \begin{cases} (J_i^- - \mu_i(\theta))\Phi(Z) + \sigma_i(\theta)\phi(Z) & \text{if } \sigma_i(\theta) > 0 \\ 0 & \text{if } \sigma_i(\theta) = 0 \end{cases} \tag{16}$$

where

$$Z = \frac{J_i^- - \mu_i(\theta)}{\sigma_i(\theta)}, \tag{17}$$

and Φ and ϕ are the *cumulative distribution function* and the *probability density function* of the standard normal distribution, respectively. Thanks to the available closed-form expression for the acquisition function, the latter can be maximized using, e.g., standard gradient-based optimization techniques.

3.3. Algorithm outline

The overall Bayesian optimization procedure for autotuning of the controller parameters θ is outlined in Algorithm 1.

At Step 1, an initial dataset of observations $\mathcal{D}_{n_{\text{in}}}$ is constructed by performing n_{in} initial experiments with different parameters $\theta_j \in \Theta$ and measuring the corresponding performance J_j. The initial parameters may be just randomly chosen withing in the admissible range Θ or they may be determined by some other experiment design techniques such as Latin Hypercube (Montgomery, 2017) in order to explore the parameter space as much as possible.

The algorithm is then iterated. At each iteration i the posterior distribution $p(J(\theta)|\mathcal{D}_i)$ is updated with the available dataset $\mathcal{D}_i$ (Step 2.1). Then, the acquisition function $\mathcal{A}(\theta)$ is computed based on $p(J(\theta)|\mathcal{D}_i)$ and optimized to determine the next point θ_{i+1} to be tested (Step 2.2). Note that this inner optimization step does not require additional experiments.

An experiment is performed with parameter θ_{i+1} and the corresponding performance index J_{i+1} is measured (Step 2.3). Finally, the dataset $\mathcal{D}_{i+1}$ is constructed by augmenting $\mathcal{D}_i$ with the latest measurements (Step 2.4). 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.

Algorithm 1 Bayesian optimization for controller autotuning

1. **generate** an initial dataset $\mathcal{D}_{n_{\text{in}}}$ by performing n_{in} experiments with a set of arbitrarily chosen parameters $\theta_j \in \Theta$, $j = 1 \dots n_{\text{in}}$:

   $\mathcal{D}_{n_{\text{in}}} = \{(\theta_1, J_1), (\theta_2, J_2), \dots, (\theta_{n_{\text{in}}}, J_{n_{\text{in}}})\}$

2. **for** $i = n_{\text{in}}, n_{\text{in}}+1, \dots$ **do**

   (a) **train** the probabilistic model $p(J(\theta)|\mathcal{D}_i)$ with the available dataset $\mathcal{D}_i$;

   (b) **optimize** the acquisition function to determine θ_{i+1}

   $\theta_{i+1} \leftarrow \arg\max_{\Theta} \mathcal{A}(\theta)$;

   (c) **execute** an experiment using parameters θ_{i+1} and **measure** J_{i+1};

   (d) **augment** the dataset:

   $\mathcal{D}_{i+1} \leftarrow \mathcal{D}_i \cup \{(\theta_{i+1}, J_{i+1})\}$;

3. **exit** when termination criterion is met;
4. **extract** the optimal controller parameters $\theta_{i^\star}$, with

   $i^\star = \arg\min_i J_i$;

Output: Optimal controller parameters $\theta_{i^\star}$

4. Decoupling parameter tuning

The Bayesian optimization approach described in the previous section simultaneously optimizes the complete set of design parameters θ, namely the PID gains $\mathbf{K}_p$, $\mathbf{K}_d$, $\mathbf{K}_i$ and the equivalent link-masses $\mathbf{m}$ characterizing the feedback linearizator (3c) and the feedforward control action (3a). A possible drawback of this one-stage tuning algorithm is that it may require many iterations (thus, closed-loop experiments) to converge in case of a high-dimensional optimization space Θ. Nevertheless, the role of the feedback linearizator in the classical robot control design suggests a way to overcome this limitation by splitting the original optimization w.r.t. the whole design parameter θ into two optimization problems of smaller size, according to the two-stage tuning visualized in Fig. 3 (right panel) and described in the following.

In the first 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

$$\mathbf{B}(\mathbf{q}, \mathbf{m})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{m}) + \mathbf{g}(\mathbf{q}, \mathbf{m}) = \boldsymbol{\tau}. \tag{18}$$

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

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{FL}} + \boldsymbol{\tau}_{\text{FF}} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{m}) + \mathbf{g}(\mathbf{q}, \mathbf{m}) + \mathbf{B}(\mathbf{q}, \mathbf{m})\ddot{\mathbf{q}}^d. \tag{19}$$

Substituting (19) into (18) leads to:

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}^d. \tag{20}$$

Thus, in the ideal case, the actual joint accelerations are equal to the target feedforward accelerations. Based on (20), the performance of the feedback linearizator with feedforward action is measured in terms of the following closed-loop objective function:

$$J_{\text{FL}}(\mathbf{m}) = \sum_{i=1}^{n} \left(\ddot{e}_i^{\max} + \ddot{e}_i^{\text{mean}} \right) + L, \tag{21}$$

where the penalty L is defined as in (7). Eq. (21) has the same structure as (4), but the acceleration-level errors are considered (joint accelerations are obtained by numerical differentiation of the joint velocity signals using a filter with cut-off frequency of 50 Hz). The cost (21) is then minimized w.r.t. the link-mass parameters $\mathbf{m}$ through Bayesian optimization. In the second stage, once the link-mass parameters $\mathbf{m}$ are computed, only the PID gains are tuned in a subsequent BO minimizing the original trajectory tracking performance index (4).

Remark 1. Since in the proposed feedback linearizator structure (3c) only link-masses are considered, the number of parameters to be tuned in the first stage is smaller than the number of PID gains. Therefore, the two-stage approach is not expected to significantly outperform the one-shot approach in terms of required closed-loop experiments. However, in general, the feedback linearizator may include other terms modeling, for instance, friction, joint and link elasticity, *etc*. In such cases, the two-stage optimization could provide a significant improvement with respect to the one-stage tuning approach. ■

5. Experimental results

This section presents experimental results on the application of the proposed auto-tuning procedure to the FRANKA Emika 7-DoFs manipulator (Fig. 1). A comparison with the performance achieved by the embedded robot position controller is also provided.

The results described in this section are also presented in the video available at https://youtu.be/W_MOijrIzMU.

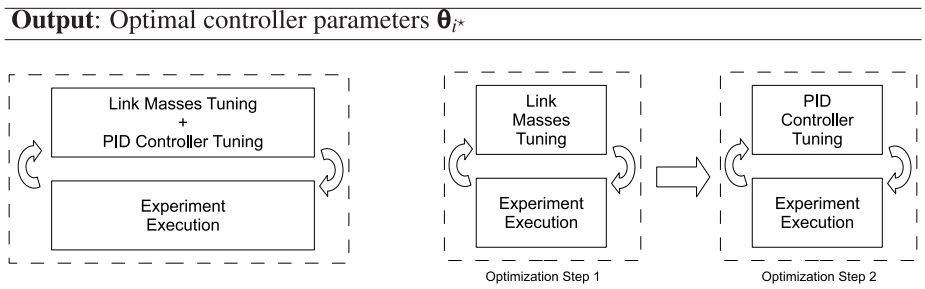


Fig. 3. Bayesian optimization for robot control parameter design: one-stage tuning (left panel) and two-stage tuning (right panel).

5.1. Setup description

The FRANKA Emika manipulator implements 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.

For a given value of the mass vector $\mathbf{m}$, the manipulator inertia matrix $\mathbf{B}(\mathbf{q}, \mathbf{m})$, the gravity $\mathbf{g}(\mathbf{q}, \mathbf{m})$, and the Coriolis $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{m})$ vectors are computed using the C++ *KDL* library (Smits, Bruyninckx, & Aertbeliën, 2013). Bayesian optimization is performed exploiting the C++ *limbo* (Cully, Chatzilygeroudis, Allocati, & Mouret, 2016) and the *NLopt* (Johnson) libraries.

Remark 2. The link-mass parameters $\mathbf{m}$ are considered to model the link properties. In fact, since the link-inertia parameters $\mathbf{I}$ have a limited effect on the robot dynamics (compared to the link-mass parameters), such parameters can be neglected in the tuning procedure. Thus, 3 parameters per joint related to the inertia tensor of the link are omitted in the optimization procedure (Roveda et al., 2018). In addition, the link-mass parameters and their link center of mass parameters $\mathbf{CoM}$ appear multiplied into the dynamic equation of the manipulator (Siciliano & Villani, 2000). While the link center of mass position vector is composed by three terms, the main effect is related to the one in the link direction, allowing to neglect the effect of the other two. Therefore, by keeping the link-center of mass parameters constant in the middle of each link, the link-mass parameters tuning will adapt the link-mass values to optimize the product of this two parameters. The tuned link-mass parameters will not be the real link-mass values, while their physical meaning is still equivalent to a mass. This tuning will guarantee that the product between link-mass parameters and the link-center of mass parameters is optimized, limiting the number of optimization variables in the proposed procedure. ■

5.2. Use case trajectories

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

- variable-frequency sinusoidal joint trajectory;
- circular Cartesian trajectory;
- sinusoidal Cartesian trajectory.

More details for each trajectory are described in the next paragraphs.

5.2.1. Variable-frequency sinusoidal joint trajectory

This trajectory performs a variable-frequency sinusoidal motion at joint level according to

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

where $i = 1, \dots, 7$ identifies the joint,

$\mathbf{f}_1 = [0.05 \;\; 0.075 \;\; 0.05 \;\; 0.075 \;\; 0.1 \;\; 0.1 \;\; 0.1]$ Hz,

$\mathbf{f}_2 = [0.015 \;\; 0.015 \;\; 0.015 \;\; 0.015 \;\; 0.015 \;\; 0.015 \;\; 0.015]$ Hz,

and

$\mathbf{A} = [40 \;\; 20 \;\; 30 \;\; 20 \;\; 20 \;\; 20 \;\; 30]°$,

while $\boldsymbol{\phi}$ is randomly selected.

The execution time for the trajectory is $T = 30$ s, which corresponds to a complete excitation cycle (*i.e.*, trajectory start and end point are the same, with zero velocity and acceleration).

Remark 3. The chosen trajectory is commonly used for robot dynamics identification (Roveda et al., 2018). In fact, it excites both slow and fast robot dynamics, and explores a wide area of the robot work-space. In this way, the inertia parameters (*i.e.*, the link-mass parameters) and friction-related parameters (*i.e.*, PID gains to compensate the friction effect) are excited at the same time. This trajectory is chosen in order to evaluate the generalization capabilities of the proposed approach (*i.e.*, use the parameters learned on this trajectory against other trajectories). ■

5.2.2. Circular Cartesian trajectory

This trajectory performs a circular-translational motion at Cartesian level in the $y-z$ plane (keeping x translation and rotations fixed) given by

$$y(t) = y(t - dt) + r\,(1 - \cos(\phi(t))),$$
$$z(t) = z(t - dt) + r\,\sin(\phi(t)),$$

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. The sampling time Δt is equal to 1 ms and the execution time of the trajectory is $T = 7$ s.

5.2.3. Sinusoidal Cartesian trajectory

This trajectory performs a sinusoidal translational-motion at Cartesian level (keeping rotations fixed) according to

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

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.

5.3. Performance evaluation

This section describes the results achieved by the proposed one-stage and the-stages BO procedures described in Sections 3 and 4, respectively. In our experiments, the BO is executed for 125 and 130 iterations for the one-stage and the two-stages approaches, respectively. In practice, it is possible to interrupt the tuning procedure as an acceptable performance is achieved. Tests are performed 3 times for

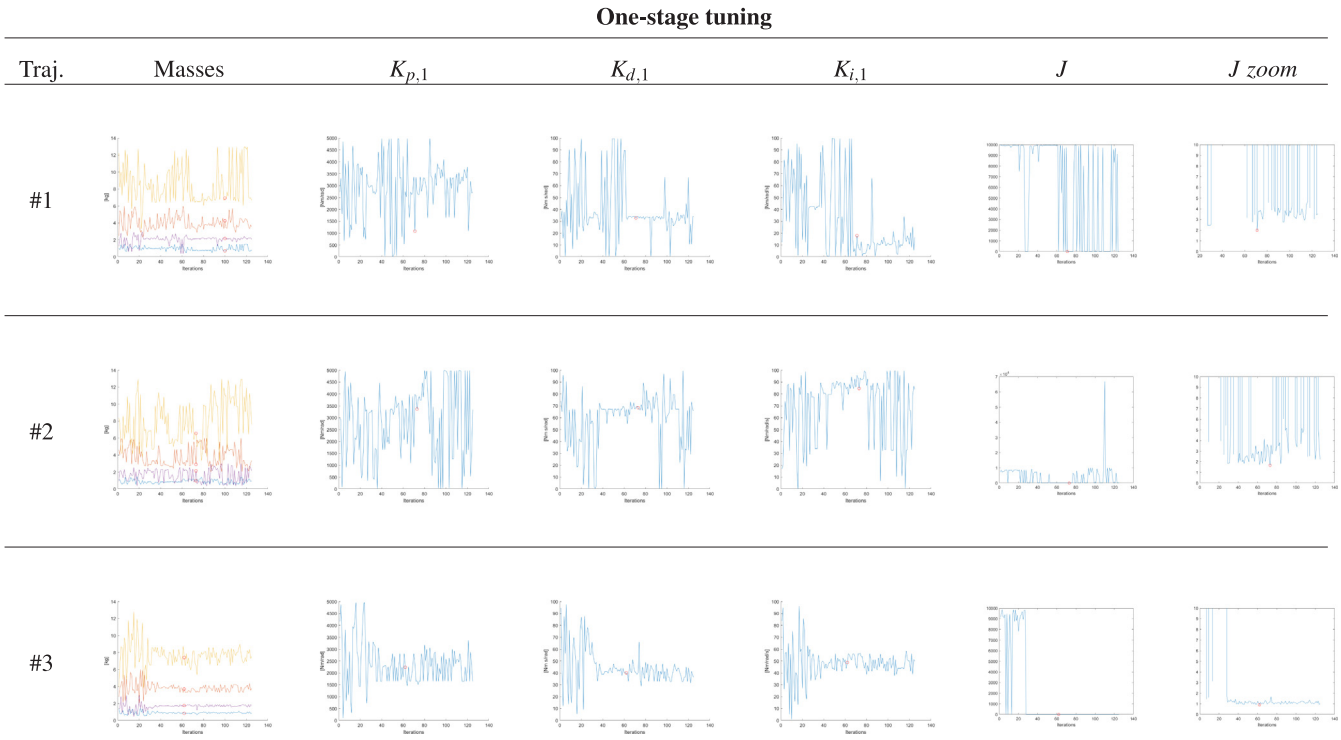


Fig. 4. Achieved results: number of the tested trajectory (first column); link-mass parameters **m** (second column, m_1 blue line, m_2 red line, m_3 yellow line, m_4 purple line); PID gains $\mathbf{K}_{p,1}$, $\mathbf{K}_{d,1}$ and $\mathbf{K}_{i,1}$ (third, fourth and fifth column); and objective 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. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

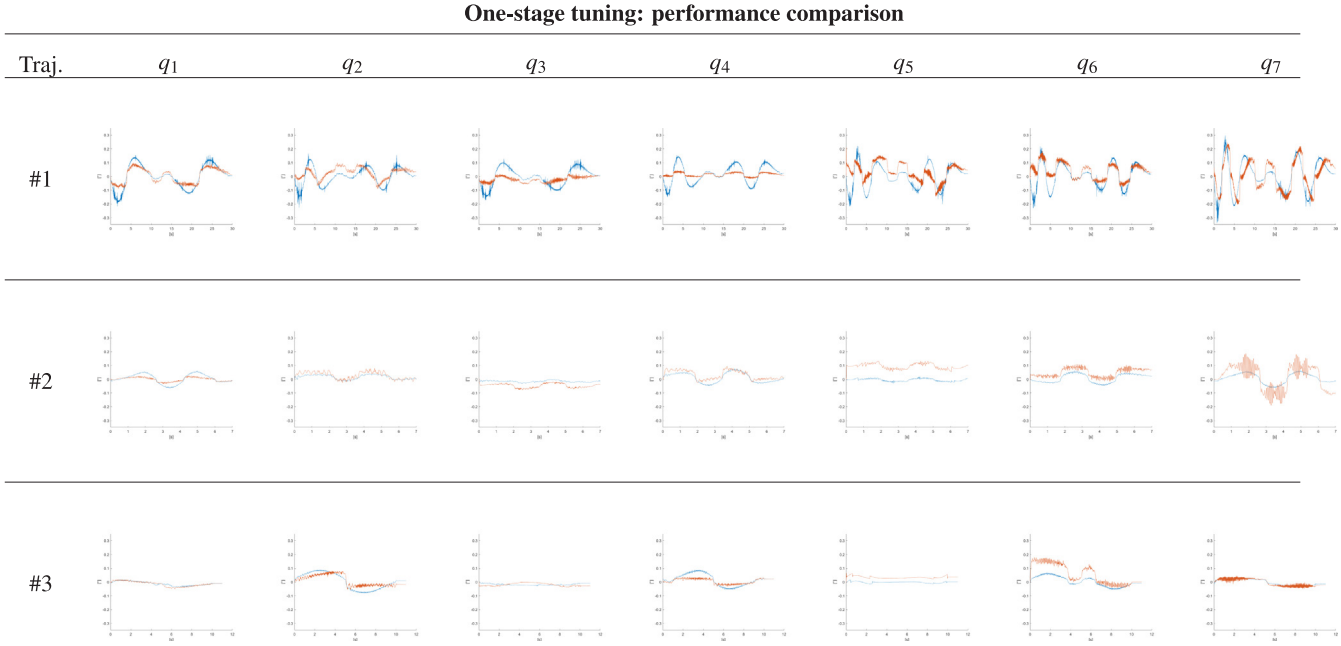


Fig. 5. Trajectory tracking errors for each robot joint achieved by the FRANKA Emika controller (blue line) and by the one-stage tuning procedure (red line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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;
- the 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 joint 1 to 7.

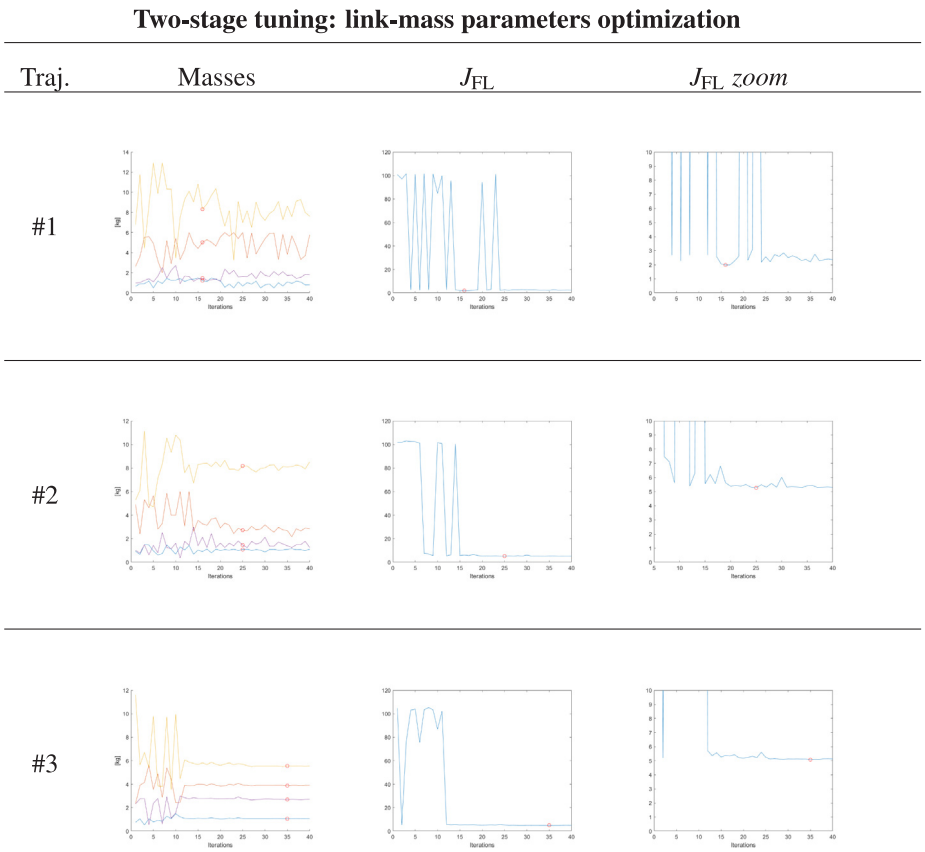


Fig. 6. Achieved results: number of the tested trajectory (first column); link-mass parameters $\mathbf{m}$ (second column, m_1 blue line, m_2 red line, m_3 yellow line, m_4 purple line); and objective function J_{FL} (third column) *vs* Bayesian optimization iterations. Zoomed evolution of the objective function J_{FL} around its minimum (fourth column). Optimal iteration is highlighted by a red circle. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.3.1. One-stage tuning

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 one-shot procedure described in Section 3.

The Bayesian optimization is initialized with $n_{\text{in}} = 25$ initial experiments with randomly generated parameters, followed by additional 100 iterations. The performance index J in (4) is minimized over the trajectory execution time T. Tests are interrupted if the joint position error $\mathbf{e}_{q,i}(t)$ is larger than $\bar{e} = 1.5°$ and a penalty term L is added to the cost function J according to the barrier function in (7a).

The obtained results are summarized in Fig. 4, which shows the evolution of the link-mass parameters, the PID gains and the performance cost J over the Bayesian optimization iterations and for the considered case-study trajectories identified in the figure as #1 (variable-frequency sinusoidal joint trajectory); #2 (circular Cartesian trajectory); and #3 (sinusoidal Cartesian trajectory). For the sake of visualization, only the PID gains for joint 1 are plotted.

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 behavior of the robot or because the maximum joint position error $\bar{e}$ is reached. As expected, this happens mainly in the first 25 initial iterations, where control parameters are chosen randomly. In total, the experiments are stopped 79 times (out of 125) for the variable-frequency sinusoidal joint trajectory; 55 times for the circular Cartesian; and 24 times for the sinusoidal Cartesian trajectory.

As it can be observed from the achieved tracking errors in Fig. 5, 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.

5.3.2. Two-stage tuning

In the second experiment campaign, the two-stage approach described in Section 4 is used. At the first stage, the equivalent link-mass parameters $\mathbf{m}$ characterizing the feedback linearizator and the feedforward action are tuned by minimizing the performance cost J_{FL} in (21). The PID gains are not optimized at this stage. In particular, derivative gains and integral gains are set to zero, while the proportional gains $K_{p,j}$ are set to low values (75 Nm/rad for all joints $j = 1, \dots, 7$) just to avoid drifts due to over/under compensations of the robot dynamics.

The Bayesian optimization is initialized with $n_{\text{in}} = 10$ initial experiments with randomly generated parameters $\mathbf{m}$, and terminated after additional 30 iterations. Tests are interrupted if the joint position error $\mathbf{e}_{q,i}(t)$ is larger than $\bar{e} = 5°$. The achieved results are summarized in Fig. 6, 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 11 times (out of 40) for the variable-frequency 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. 6, the experiments are stopped mainly in the first 10 initial iterations corresponding to randomly chosen parameters $\mathbf{m}$.

At the second stage, once the link-mass parameters are optimized, the PID controllers are designed. The Bayesian optimization is initialized with $n_{\text{in}} = 20$ initial experiments with randomly generated PID parameters, and terminated after additional 70 iterations. Tests are interrupted if the joint position error $\mathbf{e}_{q,i}(t)$ is larger than $\bar{e} = 1.5°$. The achieved results are summarized in Fig. 7. For the sake of visualization, only the PID gains for joint 1 are plotted, together with the performance cost J over the Bayesian optimization iterations. In this stage, the tests are stopped 61 times (out of 90) for the variable-frequency sinusoidal

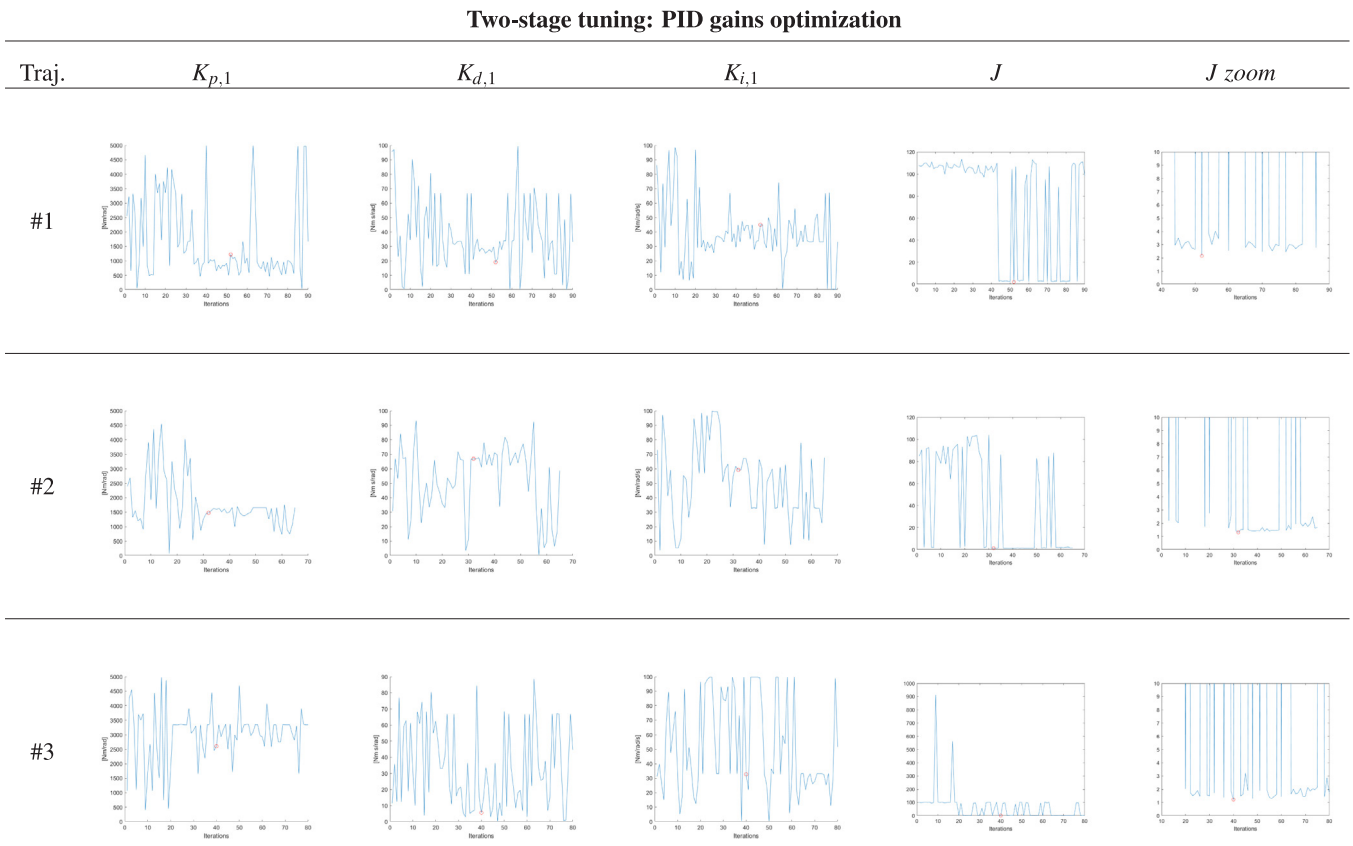


Fig. 7. Achieved results: number of the tested trajectory (first column); PID gains $K_{p,1}$, $K_{d,1}$ and $K_{i,1}$ (second, third and fourth column); and objective function J (fifth column) *vs* Bayesian optimization iterations. Zoomed evolution of the objective function J around its minimum (sixth column). Optimal iteration is highlighted by a red circle. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

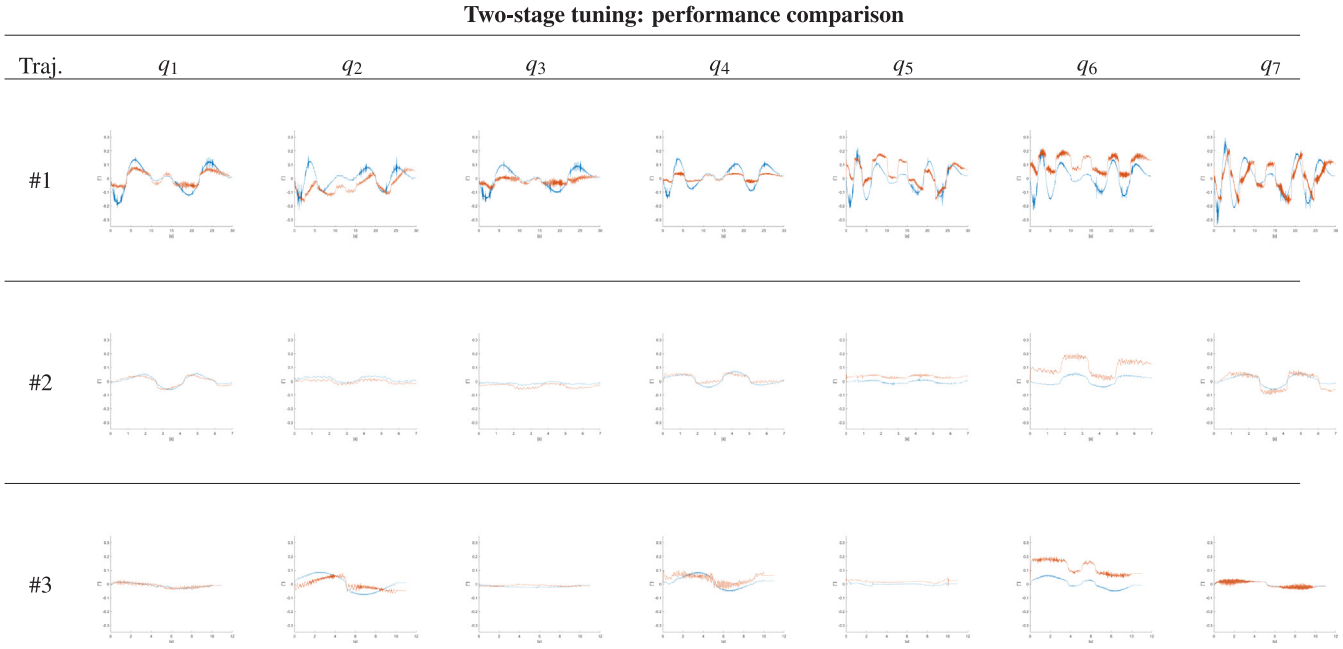


Fig. 8. Trajectory tracking errors for each robot joint achieved by the FRANKA Emika controller (blue line) and by the two-stage tuning procedure (red line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Link-mass parameters tuned with the proposed approaches *vs.* link-mass parameters identified in Gaz et al. (2019) and in gaz (2020).

Link-mass parameters [kg]							
	$m_{l,1}$	$m_{l,2}$	$m_{l,3}$	$m_{l,4}$	$m_{l,5}$	$m_{l,6}$	$m_{l,7}$
One-stage Traj. #1	0.66	2.41	0	0	6.80	0	1.91
One-stage Traj. #2	0.96	2.89	0	0	6.56	0	2.13
One-stage Traj. #3	0.83	3.70	0	0	7.40	0	1.75
Two-stage Traj. #1	1.22	5.03	0	0	8.32	0	1.47
Two-stage Traj. #2	1.08	2.73	0	0	8.1	0	1.46
Two-stage Traj. #3	1.05	3.88	0	0	5.55	0	2.70
(Gaz et al., 2019)	1.51	8.35	3.41	2.26	1.84	1.90	0.71
(gaz, 2020)	3.06	2.34	2.36	2.38	2.42	3.49	1.46

joint trajectory, 48 times for the circular Cartesian, and 26 times for the sinusoidal Cartesian trajectory.

5.3.3. Discussion

Link-mass parameters tuning: the tuned link-mass parameters are shown in Table 1 for both the one-stage and two-stage algorithms, for each performed trajectory in Section 5.2. The tuned link-mass parameters are compared with the identified parameters from Gaz, Cognetti, Oliva, Giordano, and De Luca (2019) and from gaz (2020). It can be seen that the tuned link-mass parameters are in the range of the values proposed in Gaz et al. (2019) and gaz (2020), having the link-mass parameter related to link 5 ($m_{l,5}$) including also the contributions from links 3, and 4 ($m_{l,3}$ and $m_{l,4}$), and the link-mass parameter related to link 7 ($m_{l,7}$) including also the contribution from link 6 ($m_{l,6}$). It has to be underlined that the link center of mass parameters are not optimized in the proposed methodology. Therefore, differences in the achieved link-mass parameters tuning w.r.t. (Gaz et al., 2019) and (gaz, 2020) are also related to such parameters. The tuned link-mass parameters are slightly different for each proposed testing trajectory due to the fact that each trajectory excites the robot dynamics differently. It has to be remarked that such link-mass parameters tuning is not an identification methodology; its main purpose is related to trajectory control performance. The tuning of the link-mass parameters can be improved to achieve more realistic values including all the link-mass values and their center of mass parameters into the optimization. Such increasing number of optimization variables increases, however, the complexity of the approach (*i.e.*, increasing the required optimization iterations and, therefore, the required experiments and tuning time) without guaranteeing a significant improvement of control performance;
BO algorithm performance: the BO approach allows to optimize and explore the behavior of a process in the same time (Pelikan et al., 1999). Therefore, the convergence of such a method is not shown by the evolution of the objective function over the iterations itself. In fact, since this method balances optimization and exploration, far-from-optimal values of the objective function are observed even at higher number of iterations. After termination of tuning algorithm, the parameters corresponding to the best value of the objective function are selected (red circles in Fig. 4, Fig. 6, Fig. 7).

5.4. Comparison with embedded position controller

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

Figs. 5 and 8 show the joint trajectory errors achieved by the controllers designed through the one-stage and the two-stage approaches, respectively. The maximum tracking error is less then 0.25° for both approaches on all joints. The same figures also show the error achieved with the FRANKA Emika embedded joint position controller. The obtained results show comparable performance among the different controllers. Compared to the embedded position controller, the main advantage of the proposed approach is in its limited number of experiments required to obtain high-performance trajectory tracking, without the need to model complex nonlinear dynamics (*e.g.*, friction). Moreover, considering the higher-complexity variable-frequency sinusoidal joint trajectory, achieved trajectory tracking results are even better then the one obtained with the embedded position controller.

5.5. Comparison with surrogate-based optimization

The proposed BO methodology is compared with a tuning of controller parameters based on optimization of a surrogate functions. The following two-stage tuning approach is considered:

- random sampling of the mass parameters $\mathbf{m}$ (within the same range used in BO) and experimental measurement of the corresponding performance index $J_{\mathrm{FL}}(\mathbf{m})$ in (21). 40 samples are drawn as for the two-stage methodology in Section 4;
- fit of a surrogate model to the measured indexes $J_{\mathrm{FL}}(\mathbf{m})$ and offline optimization of this surrogate model w.r.t. $\mathbf{m}$. For a fair comparison, the mean of a GP with the same structure as the one used by BO is taken as a surrogate function;
- random sampling of the PID gains $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i$ (within the same range used in BO) and experimental measurement of the corresponding closed-loop performance index J in (4). The values of the mass parameters are fixed to the ones optimized at the previous stage. 1000 samples of the PID gains are drawn[1];
- fit of a GP surrogate model to the measured indexes $J(\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i)$ and offline optimization of this surrogate model w.r.t. $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i$.

Trajectory #1 is taken as reference trajectory for comparison since such trajectory is able to better excite the robot dynamics.

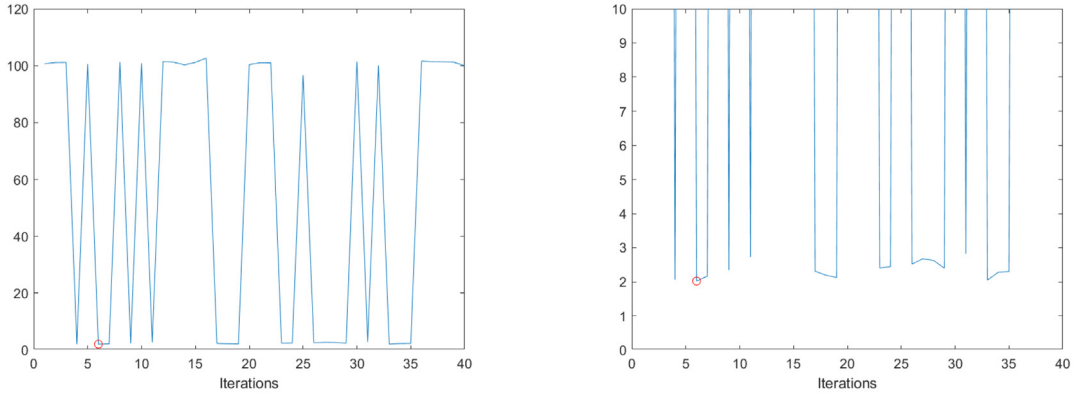
Fig. 9 shows the objective function J_{FL} obtained via random sampling of the mass parameters $\mathbf{m}$. It is worth remarking that the test has been interrupted 22 times (out of 40), while in the BO procedure the test has been stopped 11 times.

Fig. 10 shows the objective function J obtained via random sampling of the PID parameters $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i$. While the tests are stopped 61 times (out of 90, with a percentage of successful experiments of 32.2%) for two-stage BO algorithm, the tests are stopped 908 times (out of 1000, with a percentage of successful experiments of 9.2%) for the random sampling procedure. The final performance cost J achieved by the surrogate-based tuning is $J = 2.34$, slightly higher than $J = 2.15$ achieved by the BO approach in a lower number of experiments.

5.6. Generalization capabilities

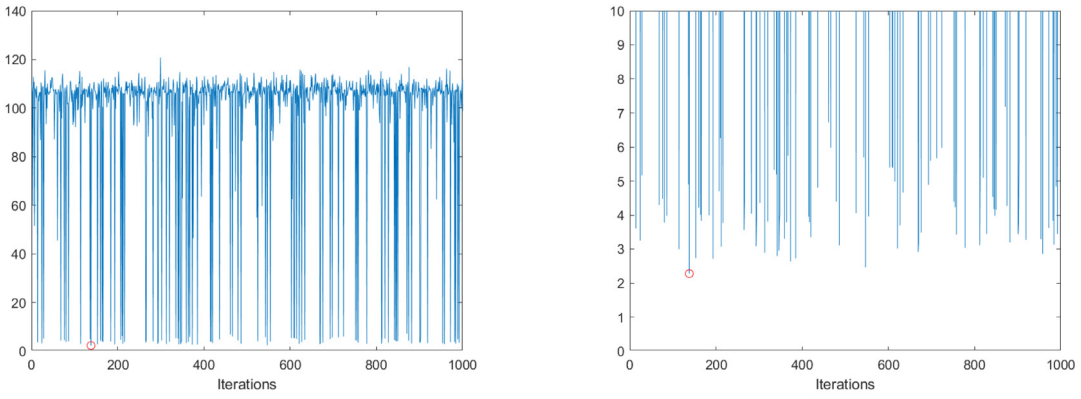
Since the equivalent link-mass parameters and the PID controller gains are tuned in order to track a specific trajectory, they take different values when optimized for different trajectories. In order to assess the generalization capabilities of the designed controllers, the controller optimized for the *variable-frequency sinusoidal joint trajectory* through the one-shot approach is used to track the other two trajectories. A maximum tracking error lower than 0.2° is obtained, as shown in the plots in Fig. 11, highlighting trajectory tracking performance that are comparable with the ones achieved by tuning the control parameters on the specific trajectory. This shows the generalization properties of the approach, when a sufficiently exciting trajectory is used for control tuning.

[1] The considered number of samples is much higher then the number of iterations executed in the BO procedure. In fact, in most of the experiments the robot is unstable, thus requiring to increase the number of samples in order to obtain an informative dataset.



J_{FL} J_{FL} zoomed

Fig. 9. Objective function J_{FL} for randomly sampled mass parameters $\mathbf{m}$ *vs* iterations (left panel); zoomed evolution of J_{FL} around its minimum (right panel).



J J zoomed

Fig. 10. Objective function J for randomly sampled PID gains $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i$ *vs* iterations (left panel); zoomed evolution of J around its minimum (right panel).

Controller optimized over trajectory #1 applied on trajectories #2 and #3.

Trajectory #2 Trajectory #3

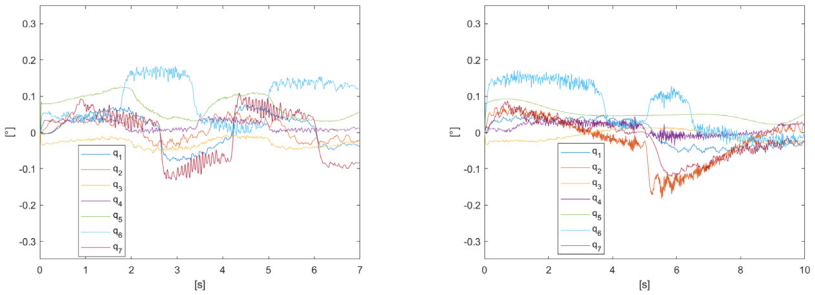


Fig. 11. Joint positions errors on trajectory #2 (left panel) and #3 (right panel). Controller parameters optimized on trajectory #1 using the one-shot approach.

6. Conclusions

A Bayesian-optimization based approach for auto-tuning of low-level trajectory tracking controllers for robot manipulators with unknown dynamics have been presented. The employed control architecture, consisting in a feedback linearizator, a feedforward action, and PID controllers at the joint level, is widely used on industrial manipulators, therefore, easily implementable.

Both the robot dynamic parameters (namely, the equivalent link masses parametrizing the feedback linearizator and the feedforward action) and the PID gains are tuned in order to optimize the performance in tracking a target trajectory. 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 as a test platform. 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.

A two-stage algorithm is also proposed, in order to split the overall design problem into two subproblems of smaller complexity. At the first stage, the feedback linearizator and the feedforward controller are designed independently of the design of the PID controllers. Specifically, according to a standard control design approach, the equivalent link-mass parameters are tuned in order to cancel the Coriolis and the gravitational torque. PID gains are optimized at a second stage. In the considered application, the one-stage and the two-stage approach show a comparable behavior, both in terms of tracking performance and computational complexity (measured in terms of number of iterations needed for parameter tuning). This is mainly due to the fact that only 4 out of 25 parameters characterize the feedback linearizator and the feedforward controller. Nevertheless, the two-stage approach is promising in the case of increasing robot dynamic model parameters (*e.g.*, including joint friction effects and joint elasticity).

Both procedures allow us to optimize the control parameters in a limited number of iterations (*i.e.*, 130 iterations), resulting in an efficient and effective auto-tuning procedure that can be easily implemented in real applications.

The generalization capabilities of the proposed approach have been shown. By properly selecting the reference trajectory (*i.e.*, a trajectory capable to properly excite the robot dynamics) for the tuning procedure, the proposed approach is capable to achieve comparable tracking errors on different trajectories with respect to *ad hoc* control parameters tuning on such specific trajectories.

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. In particular, friction compensation is under investigation. Due to the fact that friction compensation is affected by both the considered friction model and by its parameters tuning, the optimization of the friction compensator is challenging. Moreover, in order to further increase the generalization capabilities of the auto-tuned controllers, PID tuning procedures such as *loop shaping* will be considered to drive the tuning of the control gains.

CRediT authorship contribution statement

Loris Roveda: Fundig acquisition, Methodology, Sofware, Validation, Writing - original draft, Writing - review & editing, Data curation. **Marco Forgione:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Dario Piga:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The work has been developed within the project ASSASSINN, funded from H2020 CleanSky 2 under grant agreement n. 886977.

References

Franka link mass parameters. (2020). https://erdalpekel.de/?p=55, (Accessed 20 April 2020).

Ang, K. H., Chong, G., & Li, Y. (2005). PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, *13*(4), 559–576.

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

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & 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., & Ajoudani, A. (2018). A self-tuning impedance controller for autonomous robotic manipulation. In *2018 IEEE/RSJ International conference on intelligent robots and systems* (pp. 5885–5891). IEEE.

Bansal, S., Calandra, R., Xiao, T., Levine, S., & Tomiin, C. J. (2017). Goal-driven dynamics learning via Bayesian optimization. In *2017 IEEE 56th annual conference on decision and control* (pp. 5168–5173). IEEE.

Brochu, E., Cora, V. M., & 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 arXiv:1012.2599.

Bruzzone, A. G., Massei, M., Di Matteo, R., & 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* (pp. 433–444). Springer.

Calandra, R., Ivaldi, S., Deisenroth, M. P., Rueckert, E., & Peters, J. (2015). Learning inverse dynamics models with contacts. In *2015 IEEE international conference on robotics and automation* (pp. 3186–3191). IEEE.

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

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

Drieß, D., Englert, P., & Toussaint, M. (2017). Constrained bayesian optimization of combined interaction force/task space controllers for manipulations. In *2017 IEEE International conference on robotics and automation* (pp. 902–907). IEEE.

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

Formentin, S., Piga, D., Tóth, R., & Savaresi, 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.

Gaz, C., Cognetti, M., Oliva, A., Giordano, P. R., & De Luca, A. (2019). Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization. *IEEE Robotics and Automation Letters*, *4*(4), 4147–4154.

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

Hjalmarsson, H., Gevers, M., Gunnarsson, S., & Lequin, O. (1998). Iterative feedback tuning: Theory and applications. *IEEE Control Systems Magazine*, *18*(4), 26–41.

Hsiao, T., & 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., & Limnararat, S. (2016). Self-tuning control with neural network for robot manipulator. In *2016 16th international conference on control, automation and systems* (pp. 1073–1076). IEEE.

Jin, J., & 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. The NLopt nonlinear-optimization package. http://github.com/stevengj/nlopt.

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

Lecchini, A., Campi, M., & Savaresi, S. (2002). Virtual reference feedback tuning for two degree of freedom controllers. *International Journal of Adaptive Control and Signal Processing*, *16*(5), 355–371.

Letham, B., Karrer, B., Ottoni, G., Bakshy, E., et al. (2019). Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis*, *14*(2), 495–519.

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., & 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., & Lopes, M. (2015). Robot programming from demonstration, feedback and transfer. In *2015 IEEE/RSJ International conference on intelligent robots and systems* (pp. 1825–1831). IEEE.

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

Novara, C., Formentin, S., Savaresi, S. M., & Milanese, M. (2016). Data-driven design of two degree-of-freedom nonlinear controllers: The D2-IBC approach. *Automatica*, *72*, 19–27.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In *Proceedings of the 1st annual conference on genetic and evolutionary computation, Vol. 1* (pp. 525–532). Morgan Kaufmann Publishers Inc.

Pichler, A., Akkaladevi, S. C., Ikeda, M., Hofmann, M., Plasch, M., Wögerer, C., et al. (2017). Towards shared autonomy for robotic tasks in manufacturing. *Procedia Manufacturing*, *11*, 72–82.

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

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

Pinto, L., & Gupta, A. (2016). Supersizing self-supervision: Learning to grasp from 50 k tries and 700 robot hours. In *2016 IEEE International Conference on Robotics and Automation* (pp. 3406–3413). IEEE.

Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian processes for machine learning, Vol. 2, No. 3*. MA: MIT Press Cambridge.

Roveda, L., Pallucca, G., Pedrocchi, N., Braghin, F., & 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., & Torras, C. (2013). Learning collaborative impedance-based robot behaviors. In *Twenty-seventh AAAI conference on artificial intelligence*.

Selvi, D., Piga, D., & Bemporad, A. (2018). Towards direct data-driven model-free design of optimal controllers. In *2018 European control conference* (pp. 2836–2841). IEEE.

Siciliano, B., & Villani, L. (2000). *Robot force control* (3rd ed.). Norwell, MA, USA: Kluwer Academic Publishers.

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

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

Starke, G., Hahn, D., Pedroza Yanez, D., & Ugalde Leal, L. (2016). Self-organization and self-coordination in welding automation with collaborating teams of industrial robots. *Machines*, *4*(4), 23.

Swevers, J., Verdonck, W., & De Schutter, J. (2007). Dynamic model identification for industrial robots. *IEEE Control Systems Magazine*, *27*(5), 58–71.

Thoben, K.-D., Wiesner, S., & Wuest, T. (2017). "Industrie 4.0" and smart manufacturing-a review of research issues and application examples. *International Journal of Automation Technology*, *11*(1), 4–16.

Van Cuong, P., & Nan, W. Y. (2016). Adaptive trajectory tracking neural network control with robust compensator for robot manipulators. *Neural Computing and Applications*, *27*(2), 525–536.

Venkatraman, A., Capobianco, R., Pinto, L., Hebert, M., Nardi, D., & Bagnell, J. A. (2016). Improved learning of dynamics models for control. In *International symposium on experimental robotics* (pp. 703–713). Springer.

Wong, C., Yang, E., Yan, X., & Gu, D. (2017). Robots in industry: A shift towards autonomous and intelligent systems in the digital age. In *Industrial systems in the digital age conference 2017 (p. 1)*.

Yuan, K., Chatzinikolaidis, I., & 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., & 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* (pp. 556–559). IEEE.