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

Loris Roveda[1], Marco Forgione[1] and Dario Piga[1]

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

I. Introduction

A. Context

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

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

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

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

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

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

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

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

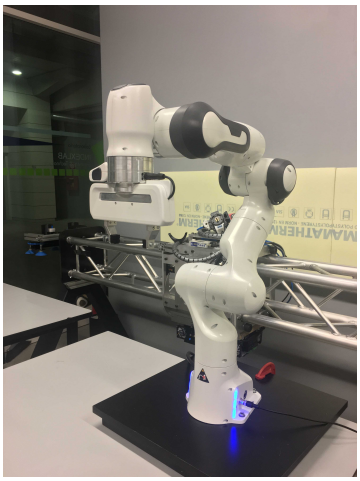


Fig. 1. The FRANKA Emika manipulator used as a test platform to evaluate the proposed procedures for control parameters auto-tuning.

compensation (*i.e.*, friction) and control parameters tuning for force-tracking applications; [20] proposes a sensor-based strategy to self-tune the impedance control for interaction tasks.

C. Paper contribution

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

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

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

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

II. Problem setting

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

A. Robot dynamics

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

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

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 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.

B. 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).

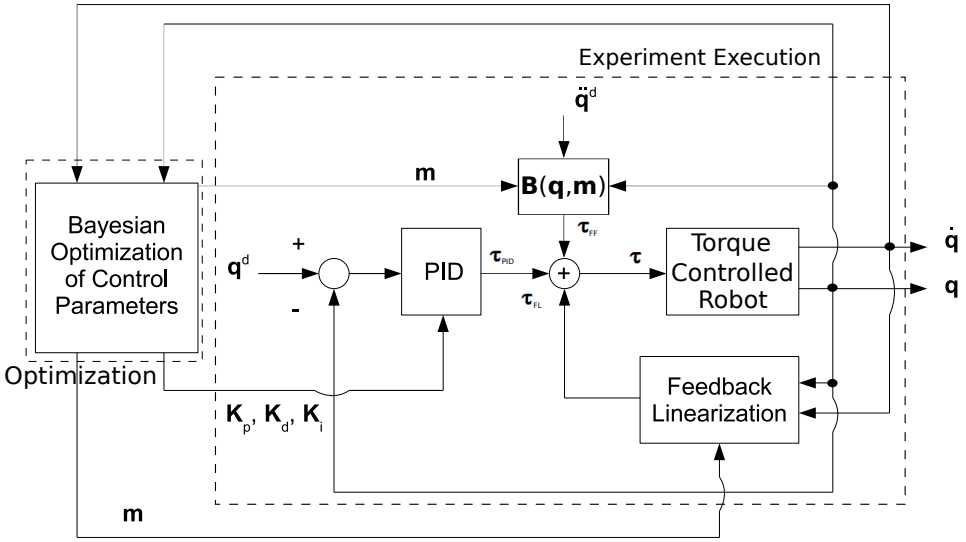


Fig. 2. Proposed architecture of control parameters self-tuning for the trajectory tracking application.

C. Objective function

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

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

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 i-th joint position error over the execution time T, *i.e.*,

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

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

with Δt the sampling time of the controller, and $\mathbf{e}_{q,i}$ denoting the i-th element of the vector $\mathbf{e}_q$, namely, the joint position error of the i-th joint. The terms $\dot{e}_i^{\max}$ and $\dot{e}_i^{\text{mean}}$ in (4) are defined similarly for the velocity error.

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

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

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

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

III. Bayesian optimization for parameters tuning

Using the controller structure described in Section II-B, the cost function J in (4) depends on tunable *design parameters*, namely, the PID gains $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_i$ and the link-mass parameters $\mathbf{m}$. 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 in a 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, *Bayesian optimization* (BO) is generally the most efficient in terms of number of function evaluations [29], [30] and it is thus the most promising approach to deal with (*iv*).

In BO, the cost J is simultaneously learnt and optimized by sequentially performing experiments on the robot. Specifically, at each iteration i of the algorithm, an experiment is

performed for a given controller parameter $\boldsymbol{\theta}_i$ 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.

A. 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.*,

$$p(J(\boldsymbol{\theta})|\mathcal{D}_i) = \frac{p(\mathcal{D}_i|J(\boldsymbol{\theta}))p(J(\boldsymbol{\theta}))}{p(\mathcal{D}_i)}. \tag{8}$$

Such a probabilistic representation describes the experimenter's uncertainty for configurations that have not yet been tested. This is the key feature distinguishing BO from classic *response surface* algorithms, that fit a deterministic surrogate of the cost function to the observations [30]. Intuitively, the probabilistic model $p(J(\boldsymbol{\theta})|\mathcal{D}_i)$ 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) [31], which is a flexible and non-parametric model allowing one to describe arbitrarily complex functions. In a GP modelling framework, the prior model for cost $J(\cdot)$ is a Gaussian Process with mean $\mu(\cdot) : \boldsymbol{\Theta} \to \mathbb{R}$ 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. $\boldsymbol{\theta}$.

B. 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) [29].

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

$$\boldsymbol{\theta}_{i+1} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{A}(\boldsymbol{\theta}). \tag{9}$$

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

$$\mathcal{A}(\boldsymbol{\theta}) = \mathbf{EI}(\boldsymbol{\theta}) = \mathbb{E}\left[\max\{0, J_i^- - J(\boldsymbol{\theta})\}\right], \tag{10}$$

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

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

and the expectation in (10) is taken over the posterior distribution $p(J(\boldsymbol{\theta})|\mathcal{D}_i)$.

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 [29]. Furthermore, it has an analytic expression in terms of the posterior mean and variance of the GP process [30] and can thus be optimized using standard gradient-based techniques.

C. Algorithm outline

The overall Bayesian optimization procedure for autotuning of the controller parameters $\boldsymbol{\theta}$ 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 $j = 1, \ldots, i$. 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.

IV. Experimental results

A. Setup description

The FRANKA Emika manipulator has been used as test platform, implementing a 1-kHz torque control. The number of link-mass parameters $\mathbf{m}$ 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. Robot dynamics has been computed using the C++ *KDL* library [32]. Bayesian optimization has been performed exploiting the C++ *limbo* [33] and the *NLopt* [34] libraries.

B. Use case trajectories

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

- joint variable-frequency sinusoidal motion (Traj. #1)

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

where $i = 1,\ldots,7$ 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, $\mathbf{A} = [40 \;\; 20 \;\; 30 \;\; 20 \;\; 20 \;\; 20 \;\; 30]^\circ$, and $\boldsymbol{\phi}$ 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 1. The chosen trajectory is commonly used for robot dynamics identification [19]. 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).

- circular Cartesian trajectory (Traj. #2) in the $y-z$ plane

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

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. 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.

C. Performance evaluation

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

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

- link-mass parameters $\mathbf{m}$ 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 one-shot procedure described in Section III.

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

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

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

D. Comparison with embedded position controller

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

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

V. Conclusions

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

One-stage tuning

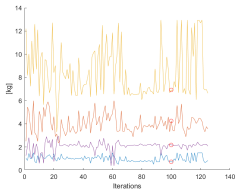
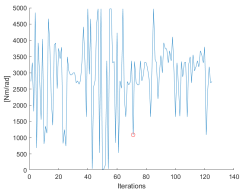
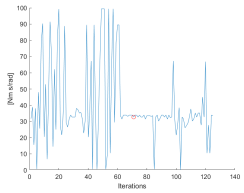
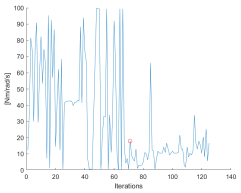
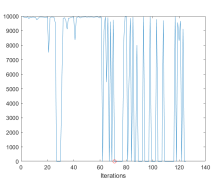
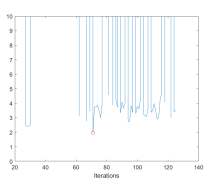
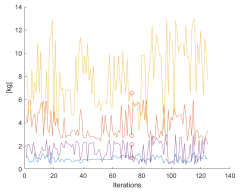
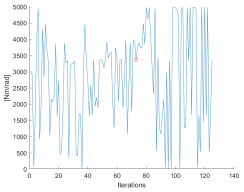
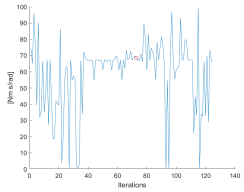
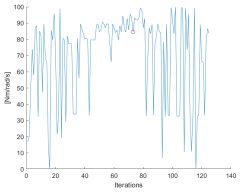
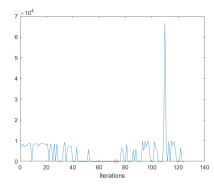
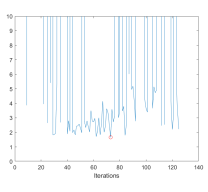
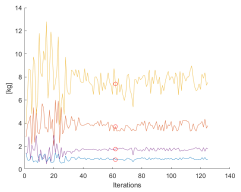
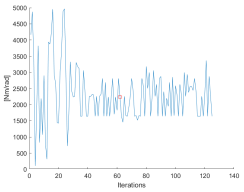
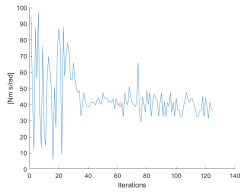
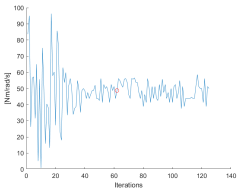
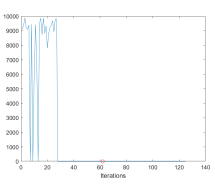
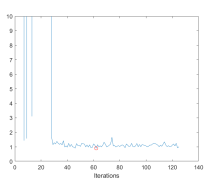
Traj.	Masses	$K_{p,1}$	$K_{d,1}$	$K_{i,1}$	J	J *zoom*
#1						
#2						
#3						

Fig. 3. Achieved results: number of the tested trajectory (first column); link-mass parameters **m** (second column, m_1 blue line, m_2 red line, m_3 yellow line, m_4 purple line); PID gains $K_{p,1}$, $K_{d,1}$ and $K_{i,1}$ (third, forth and fifth column); and objective function J (sixth column) *vs* Bayesian optimization iterations. Zoomed evolution of the objective function J around its minimum (seventh column). Optimal iteration is highlighted by a red circle.

One-stage tuning: performance comparison

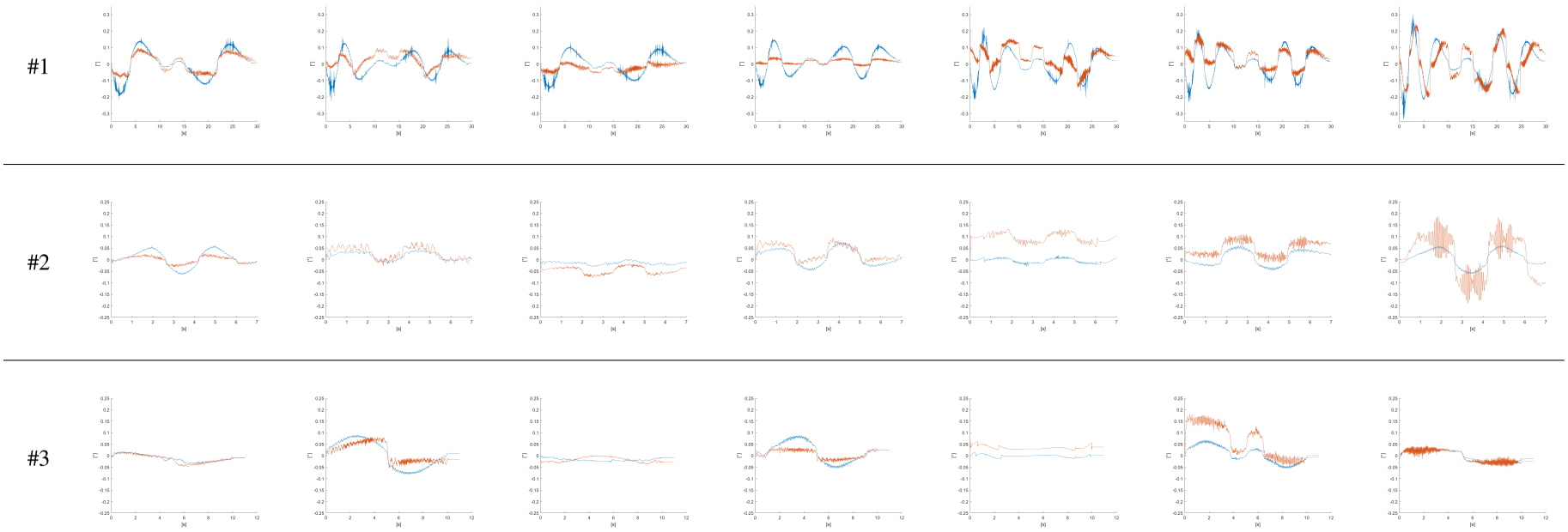


Fig. 4. Trajectory tracking errors for each robot joint achieved by the FRANKA Emika controller (blue line) and by the one-stage tuning procedure (red line).

is widely used on industrial manipulators, therefore, easily implementable. Both the robot dynamic parameters and the PID gains are tuned in order to optimize the trajectory tracking performance. Safety constraints and maximum joint position errors are also taken into account.

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

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

REFERENCES

[1] S. Makridakis, "The forthcoming artificial intelligence (ai) revolution: Its impact on society and firms," *Futures*, vol. 90, pp. 46–60, 2017.

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

[3] J. Swevers, W. Verdonck, and J. De Schutter, "Dynamic model identification for industrial robots," *IEEE control systems magazine*, vol. 27, no. 5, pp. 58–71, 2007.

[4] J. Jin and N. Gans, "Parameter identification for industrial robots with a fast and robust trajectory design approach," *Robotics and Computer-Integrated Manufacturing*, vol. 31, pp. 21–29, 2015.

[5] S. Formentin, D. Piga, R. Tóth, and S. M. Saveresi, "Direct learning of LPV controllers from data," *Automatica*, vol. 65, pp. 98–110, 2016.

[6] D. Piga, S. Formentin, and A. Bemporad, "Direct data-driven control of constrained systems," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 331–351, 2018.

[7] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[8] P. J. Antsaklis and A. Rahnama, "Control and machine intelligence for system autonomy," *Journal of Intelligent & Robotic Systems*, vol. 91, no. 1, pp. 23–34, 2018.

[9] L. D. Rozo, S. Calinon, D. Caldwell, P. Jiménez, and C. Torras, "Learning collaborative impedance-based robot behaviors," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

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

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

[12] A. Venkatraman, R. Capobianco, L. Pinto, M. Hebert, D. Nardi, and J. A. Bagnell, "Improved learning of dynamics models for control," in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 703–713.

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

[14] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577–582, 2019.

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

[16] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in neural information processing systems*, 2016, pp. 64–72.

[17] H. Modares, I. Ranatunga, F. L. Lewis, and D. O. Popa, "Optimized assistive human–robot interaction using reinforcement learning," *IEEE transactions on cybernetics*, vol. 46, no. 3, pp. 655–667, 2015.

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

[19] L. Roveda, G. Pallucca, N. Pedrocchi, F. Braghin, and L. M. Tosatti, "Iterative learning procedure with reinforcement for high-accuracy force tracking in robotized tasks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1753–1763, 2018.

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

[21] R. Hernández-Alvarado, L. García-Valdovinos, T. Salgado-Jiménez, A. Gómez-Espinosa, and F. Fonseca-Navarro, "Neural network-based self-tuning pid control for underwater vehicles," *Sensors*, vol. 16, no. 9, p. 1429, 2016.

[22] J. Zhao, L. Han, L. Wang, and Z. Yu, "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)*. IEEE, 2016, pp. 556–559.

[23] M. Galicki, "Finite-time trajectory tracking control in a task space of robotic manipulators," *Automatica*, vol. 67, pp. 165–170, 2016.

[24] T. Hsiao and P.-H. Huang, "Iterative learning control for trajectory tracking of robot manipulators," *International Journal of Automation and Smart Technology*, vol. 7, no. 3, pp. 133–139, 2017.

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

[26] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, p. 503, 2015.

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

[28] B. Siciliano and L. Villani, *Robot Force Control*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

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

[30] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[31] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006, vol. 2, no. 3.

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

[33] A. Cully, K. Chatzilygeroudis, F. Allocati, and J.-B. Mouret, "Limbo: A fast and flexible library for bayesian optimization," *arXiv preprint arXiv:1611.07343*, 2016.

[34] S. G. Johnson, "The nlopt nonlinear-optimization package," http://github.com/stevengj/nlopt.