IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS
Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

Self-Correcting Quadratic Programming-Based Robot Control

Farshad Khadivar[1†*], Konstantinos Chatzilygeroudis[2,3†], and Aude Billard[1]

***Abstract*—Quadratic Programming (QP)-based controllers allow many robotic systems, such as humanoids, to successfully undertake complex motions and interactions. However, these approaches rely heavily on adequately capturing the underlying model of the environment and the robot's dynamics. This assumption, nevertheless, is rarely satisfied, and we usually turn to well-tuned end-effector PD controllers to compensate for model mismatches. In this paper, we propose to augment traditional QP-based controllers with a learned residual inverse dynamics model and an adaptive control law that adjusts the QP online to account for model uncertainties and unforeseen disturbances. In particular, we propose (i) learning a residual inverse dynamics model using the Gaussian Process and linearizing it so that it can be incorporated inside the QP-control optimization procedure and (ii) a novel combination of adaptive control and QP-based methods to avoid the manual tuning of end-effector PID controllers and faster convergence in learning the residual dynamics model. In simulation, we extensively evaluate our method in several robotic scenarios ranging from a 7-DoFs manipulator tracking a trajectory to a humanoid robot performing a waving motion for which the model used by the controller and the one used in the simulated world do not match (unmodeled dynamics). Finally, we also validate our approach in physical robotic scenarios where a 7-DoFs robotic arm performs tasks where the model of the environment (mass, friction coefficients, etc.) is not fully known.**

***Index Terms*—Learning control systems, model reference adaptive control, quadratic programming, torque control, residual inverse dynamics.**

I. INTRODUCTION

In multi-body robotic systems, we need to carefully coordinate a large number of degrees of freedom (DoFs) and interaction forces. Such coordination becomes more challenging when operating in task-space, even for the simplest tasks, e.g., a bimanual robot interacting with an object [1]. Quadratic programming (QP)-based methods provide a principled control framework to tackle these challenges [2, 3, 4]. In QP, task requirements are formulated as an optimization process for minimizing an objective function together with satisfying constraints such as joint limits, system dynamics, and contact forces [5, 6]. Due to the smart formulation and modern computation, this optimization process can run up to 1 KHz, allowing for high-frequency control even in humanoids [2].

However, similar to most model-based controllers [7], the key assumption of any QP-based control is the precision of the model and whether it adequately captures the underlying robot/environment model. This is, of course, rarely the case since in many real-world scenarios, the model at hand and the real model do not match. Even when the system is coupled with accurate state estimators and high-gain PID feedback, real-world experiments are subject to frequent failures due to model imperfections, friction, actuator nonlinearities, etc. [8]. In practice, for these reasons, configuring a QP-based controller for a convoluted robot (e.g., humanoid) or complex interactions (e.g., handling objects) almost always involves a great deal of hand-tuning of the model parameters and the cost function for each specific instance of the task [9]. In this study, we aim to address these limitations of QP-based controllers. We focus on the problem of model imprecision and feedback inadequacy of the control law.

Humans and animals have a remarkable way of performing new tasks or adapting to unforeseen situations. They learn from their mistakes and, by trial-and-error, master new skills. We envision a similar robotic system that learns through trial-and-error: it tries to achieve the task with a QP-based controller, fails (e.g., the box slips from the hands of the robot), and tries again until it manages to realize the goal. This adaptation is functional only if it is fast; e.g., imagine having to wait 2 days for a robot to learn how to perform a search and rescue scenario task. Thus we would like the procedure to happen in as short an interaction time as possible[1] [10].

Therefore, the main question that arises here is: how can we improve a QP-based controller as per previous trials? Assuming an accurate/perfect QP solver, three central elements can be updated: (i) the *task specification* (i.e., end-effector desired accelerations), (ii) the *cost function* of the QP, and (iii) the *model* of the QP. The first would require an external oracle to give us feedback on whether or not we were performing the task well, and then one would have to find which is the best oracle to do so. We, therefore, assume that the task specifications are well-designed, and we do not update them. Altering the cost function freely is likewise dangerous. The QP solver might fail or produce weird motions, and this is why most related approaches model the cost function as a series of waypoints or attractors/repulsors [11, 12]. In contrast, improving the model of the QP with data gathered from the physical trials will make the task of the QP solver easier.

In this paper, we focus on QP-based inverse dynamics (ID) controllers and investigate means by which we can update the *cost function* to ensure the stability of the system and improve the precision of the *ID model* efficiently. More precisely, we formulate an ID model learning procedure to improve the model of the QP and show that it applies to a variety of robots with different dynamics. We then introduce a novel QP-based control scheme that is able to overcome model

*Corresponding author: farshad.khadivar@epfl.ch
† Equal Contribution
[1]*LASA, Swiss Federal School of Technology in Lausanne, Switzerland*
[2]*Computer Technology Institute & Press "Diophantus", Patras, Greece*
[3]*Computer Engineering and Informatics Department, University of Patras, Greece*

This work was supported by the European Research Council, Advanced Grant agreement No 741945, Skill Acquisition in Humans and Robots.

[1]Minimizing the interaction time with the system is equivalent to minimizing the number of trials in episodic settings.

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

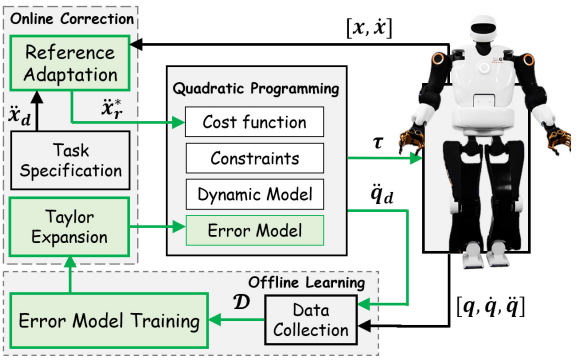

Fig. 1: Our proposed framework for robot torque control using a quadratic programming scheme. Green components, indicating our contributions, are developed in this study to compute the joint torques τ for realizing a desired end-effector acceleration $\ddot{x}_d$. Through online reference adaptation, given the feedback from robot end-effectors, and error model exploitation, using Taylor expansion, our approach enables the QP to correct itself such that the *expected* joint accelerations, $\ddot{q}_d$, converge to the *actual* values, $\ddot{q}$.

inaccuracies and large model mismatches by combining the slow ID model learning with a fast online adaptive control law in task-space to regulate the cost function of the QP (see Figure 1). Using adaptive control per se for robot control is not new, however, to the best of our knowledge, we have, for the first time, employed online adaptation of nonlinearities and model uncertainties to close the control loop for QP-based controllers.

In a nutshell, our approach builds upon existing ID model learning techniques to improve the QP dynamics model over time [8, 13, 14] and existing adaptive control methods [15] that can regulate the feedback term and change the cost function of the QP on-the-fly. This is different from previous methods of learning for QP-based controllers that mostly attempt to alter the QP cost function only (e.g., [11, 12]). Our main contributions are:

i. An episodic procedure of learning residual ID model for QP-based control where we use an expressive and differentiable Gaussian Process with Rigid Body Dynamic (RBD) model as prior. In our method, ID is differentiable with respect to the optimization variables; hence, it actively exploits the learned ID model within the optimization of the QP-based control.
ii. A novel scheme for continuous online reference adaptation in the cost function QP. To this end, we employ an adaptive controller that avoids manual tuning, addresses model uncertainties online, and results in a faster convergence for ID model learning.

We validate our approach extensively in simulations and also perform experiments in a physical robotic setup. Our results showcase that our method (i) is able to compensate for large model inaccuracies in little interaction time (i.e., in a few trials), and (ii) consistently outperforms the baselines.

II. RELATED WORK

Control structures based on optimization are used widely in general-purpose simulations and automatic motion synthesis. Among early studies of interest, Abe et al. [16] controlled the balancing of animated humans by integrating constraints, namely the frictional and non-planar contact model. Later, De Lasa et al. [17] and Coros et al. [18] extended the approach and incorporated physical features into prioritized levels of optimization for locomotion with periodic contact switching.

QP is an example of an optimization-based controller and has also been employed frequently in various applications in robotics such as trajectory planning/tracking [19], multi-body control [5], multi-task planning [20], and dynamic balancing [21]. For instance, regarding multi-body robots, Zhang et al. [5] proposed a torque optimization scheme for controlling robots with kinematic redundancy. QP also allows nesting multiple optimizations for task planning [22]. Similarly, for multi-tasking, several approaches have been proposed: the soft-hierarchy method [20], learning task priorities [23], and priority determining via stochastic optimization [9]. More recently, Marcucci et al. [24] defined a multi-parametric non-linear program to generate offline trajectories and adapt the task parameters online.

The simple pipeline for integrating constraints such as physical limits [3], task-space restrictions [25], and robot dynamics [26, 27] into the control scheme is the other advantage of QP-based methods. For instance, promisingly in [26], Herzog et al. proposed a momentum-based balance algorithm and implemented a task-space torque controller on a robot in real-time. Contacts and interactions are other critical constraints in the dynamics model. Bouyarmane et al. [27] investigated task-space force control and embedded contact and interaction constraints into multi-body QP control by utilizing sensory feedback for contact.

The performance of the optimization solver for QP-based controllers was the primary focus in successful studies like [28, 19] and also [2] where the control rate for multi-body systems with large DoFs was enhanced. However fast the solver, these controllers heavily rely on the presence of accurate models. Such a requirement highly limits QP's success in real experiments as small model inaccuracies can lead to catastrophic behavior in multi-body systems [29]. To avoid this, some authors have sought to find an analytical solution for a simplified version of ID [30], whereas others have tried to employ separate optimization schemes for ID and inverse kinematics (IK) [31]. Still, these approaches neither guarantee stability nor convergence to an optimal solution, especially in the presence of inconsistency between ID and IK solutions.

As an alternative method for model identification [32], some authors tried non-parametric model learning approaches [8, 33] for robot dynamics. Nonetheless, the accuracy of such models depends on the richness of the training data, which is arduous to capture in systems with larger DoFs. In most cases, these models are learned in an offline fashion: we excite the system in pre-defined trajectories and learn the models from the gathered data [34]. These settings assume that we already have controllers working under many different scenarios and that we know how to generate meaningful trajectories, which is a strong assumption to make for any system, i.e., it is much easier to do with a manipulator than with a humanoid [35, 8].

Some approaches attempted to update QP-based controllers with trial-and-error learning techniques [10]. Spitz et al. [11]

adopted an episodic learning scheme and used the state-space trajectories of failed trials to introduce repulsors. They used these repulsors to push the QP controller away from the already visited states. Lober et al. [12] used a Bayesian Optimization procedure to select waypoints that the whole-body controller should follow when controlling a humanoid robot. Modugno et al. [36, 9] used an offline optimization procedure, in simulation, to learn the temporal profiles of the task weights [36]. Using reinforcement learning techniques in simulation, Ichnowski et al. [37] learn policies that produce different QP solver parameters at each time step resulting in faster and more reliable convergence of the QP optimization.

Although these results are successful and promising, the methods do not update the model of the environment and can produce harmful behaviors for the robot. Moreover, most methods (especially in the literature about combining learning and QP-based controllers) utilize position-controlled schemes. By contrast, we focus mainly on torque-controlled scenarios that allow for more compliant and finer control.

The computation of control efforts from QP is open-loop, and for closing the loop, PID controllers are mainly used. In this study, we use adaptive controllers [15], which are capable of compensating for model inaccuracies and unmodeled dynamics. Furthermore, adaptive controllers do not discard the valuable information from the previous trials. The adaptive nature of the controller mitigates the challenge of gain tuning and provides higher flexibility for various desired behaviors or tasks [15, 38]. For instance, in a collaborative scenario for wheeled robots, Wang et al. [39] proposed distributing the control among agents and controlling the leader robot by an adaptive control law. Within the adaptive class of controllers, model reference adaptive control (MRAC) methods have demonstrated substantial performance in addressing model uncertainties, thereby emerging in numerous studies [40, 41, 42]. For instance, Culbertson et al. [43] manipulate an object by using a decentralized MRAC for controlling a group of collaborative robots where the dynamic and geometrical properties of the object are unknown. Similarly, we employ MRAC for online reference adaptation in QP-based controllers to react online to nonlinearities and model uncertainties.

III. PRELIMINARIES

For many applications in robotics, it is more convenient and intuitive to design the desired behavior in the Cartesian coordinate system of the end-effector(s), also known as the task-space. In the ID formulation, we have as input desired end-effectors accelerations $\ddot{\boldsymbol{x}}_d \in \mathbb{R}^l$ (l is the dimension of the task-space), and we would like to find the joint-level torques needed to achieve these accelerations. The equations of motion and constraint equations for a robot can be described as [31]:

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{W} \\ \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} &= \ddot{\boldsymbol{x}}_r \end{aligned} \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^j$ is the full state of the system (with j being the number of DoFs of the robot, including the 6-DoFs of the floating base, if present)[2], $\boldsymbol{x} \in \mathbb{R}^l$ is the concatenation of the poses (containing position and orientation) in Cartesian space of all the contact points (if present), $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{j\times j}$ is the inertia matrix, $\boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^j$ is the sum of the gravitational, centrifugal and Coriolis forces, $\boldsymbol{S} \in \mathbb{R}^{j\times j}$ is a selection matrix where the first 6 rows are all zeros and the rest is the identity matrix, $\boldsymbol{W} \in \mathbb{R}^{6n_c\times j}$ is the concatenation of all n_c contact wrenches (in world frame), $\boldsymbol{J} \in \mathbb{R}^{6n_c\times j}$ is the concatenation of the Jacobians of all the contact points, and $\boldsymbol{\tau} \in \mathbb{R}^j$ is the vector containing the control torques of all DoFs of the system. We can re-write the equations of motion as [31]:

$$\begin{bmatrix} \boldsymbol{M}(\boldsymbol{q}) & -\boldsymbol{S} & -\boldsymbol{J}(\boldsymbol{q})^T \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\tau} \\ \boldsymbol{W} \end{bmatrix} + \boldsymbol{C_g}(\boldsymbol{q},\dot{\boldsymbol{q}}) = 0 \tag{2}$$

This formulation is interesting as given a state $(\boldsymbol{q},\dot{\boldsymbol{q}})$, the equations of motion are linear with respect to $\begin{bmatrix}\ddot{\boldsymbol{q}} & \boldsymbol{\tau} & \boldsymbol{W}\end{bmatrix}^T$. By defining $\mathcal{X} = \begin{bmatrix}\ddot{\boldsymbol{q}} & \boldsymbol{\tau} & \boldsymbol{W}\end{bmatrix}^T$ in this paper, we can now express the ID formulation as a QP-based whole-body control problem [31]:

$$\begin{aligned} \min_{\mathcal{X}} \; & -\frac{1}{2}\mathcal{X}^T\boldsymbol{G}\mathcal{X} + \boldsymbol{g}^T\mathcal{X} \\ \text{s.t. } & \boldsymbol{H}_E\mathcal{X} = \boldsymbol{b}_E \\ & \boldsymbol{H}_I\mathcal{X} \geq \boldsymbol{b}_I \end{aligned} \tag{3}$$

where we are optimizing tasks of the form $\frac{1}{2}\|\boldsymbol{H}\mathcal{X} - \boldsymbol{b}\|^2$, and where $\boldsymbol{G} = \boldsymbol{H}^T\boldsymbol{H}$ and $\boldsymbol{g} = -\boldsymbol{H}^T\boldsymbol{b}$. In particular, we turn the equations of motion into equality constraints ($\boldsymbol{H}_E$ and $\boldsymbol{b}_E$), and we turn joint limits and other constraints, such as friction cone or center of pressure constraints, into inequality constraints ($\boldsymbol{H}_I$ and $\boldsymbol{b}_I$). We then define desired accelerations of some end-effector by filling $\boldsymbol{G}$ and $\boldsymbol{g}$ appropriately[3] (the QP task objectives). To make things more concrete, imagine a manipulator which is rigidly attached to the world, and we treat the base of its gripper as the end-effector. In this case, $\boldsymbol{x}$ represents the position and orientation of the base of the gripper of the manipulator (see Section V-A1 for an example on how to fill $\boldsymbol{G}$ and $\boldsymbol{g}$). When we have multiple tasks with different weights w_i, we can decompose $\boldsymbol{H}$ and $\boldsymbol{b}$ as $\boldsymbol{H} = \begin{bmatrix}\mathrm{w}_1\boldsymbol{H}_1^T, \mathrm{w}_2\boldsymbol{H}_2^T, \ldots, \mathrm{w}_n\boldsymbol{H}_n^T\end{bmatrix}^T$, and $\boldsymbol{b} = \begin{bmatrix}\mathrm{w}_1\boldsymbol{b}_1, \mathrm{w}_2\boldsymbol{b}_2, \ldots, \mathrm{w}_n\boldsymbol{b}_n\end{bmatrix}^T$.

In this paper, we compute the reference end-effector accelerations $\ddot{\boldsymbol{x}}_r$ by:

$$\ddot{\boldsymbol{x}}_r = \ddot{\boldsymbol{x}}_f + \ddot{\boldsymbol{x}}_d \tag{4}$$

where $\ddot{\boldsymbol{x}}_d$, the end-effector desired acceleration, is the feed-forward control term specified by a higher-level controller and can change over time based on the defined current task. $\ddot{\boldsymbol{x}}_f$ is the feedback term which closes the control loop. Previous work has usually applied PID or PD control structure to close the loop: $\ddot{\boldsymbol{x}}_f = -k_p(\boldsymbol{x} - \boldsymbol{x}_d) - k_v(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_d)$. Finding proper gains ($k_p \in \mathbb{R}^+$ and $k_v \in \mathbb{R}^+$) for each task requires heavy gain tuning with no stability guarantee and no control over the transient error behavior. In this paper, we close the control loop via an MRAC control scheme. In other words, $\ddot{\boldsymbol{x}}_r$ is derived from a nonlinear adaptive controller presented in Section IV-A. By doing so, we modulate $\ddot{\boldsymbol{x}}_r$ in an online fashion to follow

[2]We denote time derivatives with an upper dot: e.g., $\dot{x}$.

[3]We use the `whc` library: https://github.com/costashatz/whc.

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

a desired dynamical system, thereby controlling the transient behavior of the error signal, and we guarantee the stability of the system in the feedback line with appropriate adaptive laws.

IV. SELF-CORRECTING QP-BASED CONTROL

We propose a novel QP-based control scheme, called the Self-Correcting QP-based Control framework (SCQP, see Figure 1), where a nonlinear adaptive controller computes the reference accelerations for the QP-based ID (Section IV-A), and a learning procedure improves the ID model of the QP (Section IV-B). Overall, in the SCQP, we adopt an episodic learning scheme and perform the following steps (see also Algorithm. 1):

1) Design the task specifications: $\boldsymbol{x}_d(t), \dot{\boldsymbol{x}}_d(t), \ddot{\boldsymbol{x}}_d(t)$.
2) Configure the adaptive controller and model learning procedure.
3) Perform an episode. For each time step:
   a) Compute the reference accelerations, $\ddot{\boldsymbol{x}}_r$ using our nonlinear adaptive controller (Section IV-A).
   b) Compute the cost function for the QP given $\ddot{\boldsymbol{x}}_r$;
   c) Get the torques $\boldsymbol{\tau}$ from the QP with the updated cost function, and the learned ID model, $\boldsymbol{h}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ via linearization.
   d) Apply the torques to the robot and collect data.
   e) Update the adaptive controller at each time step.
4) Learn the ID model with Gaussian Processes with all the collected data.
5) Go back to step 3 until convergence.

Throughout the paper, we use n, l, and $m \in \mathbb{N}$ to refer to the dimension of the state-space, task-space, and control input, respectively. For easier reference, in Table I, we list all the notations necessary to follow our approach and derivations; hereafter, refer to Table I for the variable-related dimensions. We use subscripts r and d for indexing *reference* and *desired* variables and utilize the following conventions throughout the article: typeface for scalars (e.g., a), lowercase bold font to represent vectors (e.g., $\boldsymbol{a}$), and uppercase bold font to refer to matrices (e.g., $\mathbf{A}$). For brevity, we drop the variable-related indexing of each variable in Table I.

TABLE I: Core Notations

Symbol	Dimension	Description
$\boldsymbol{x}$	$\mathbb{R}^l$	Robot end-effector Cartesian position
$\boldsymbol{\zeta}$	$\mathbb{R}^n$	Robot task-space states
$\boldsymbol{\nu}$	$\mathbb{R}^m$	Adaptive output control signal
$\boldsymbol{\Phi}(\cdot)$	$\mathbb{R}^p$	Vector of basis functions
$\mathbf{P}$	$\mathbb{R}^{n\times n}$	A symmetric positive definite matrix
$\mathbf{Q}$	$\mathbb{R}^{n\times n}$	A positive definite matrix
$\mathbf{A}$	$\mathbb{R}^{n\times n}$	State matrix of a dynamical system
$\mathbf{B}$	$\mathbb{R}^{n\times m}$	Input matrix of a dynamical system
$\boldsymbol{r}(t)$	$\mathbb{R}^n$	Input signal of the reference model
$\bar{\boldsymbol{\Psi}}_v$	$\mathbb{R}^{m\times n_v}$	Estimated control gain for $v = \boldsymbol{\zeta}, \boldsymbol{r}, \boldsymbol{\phi}$
Λ_v	$\mathbb{R}^{m\times n_v}$	Adaptation gain for $v = \boldsymbol{\zeta}, \boldsymbol{r}, \boldsymbol{\phi}$

A. Task-Space Adaptive Control

Let us define $\boldsymbol{\zeta} = [\boldsymbol{x}^T, \dot{\boldsymbol{x}}^T]^T$ to be the states of the end-effector in the task-space. Then, the objective of our adaptive controller is to ensure that the $\boldsymbol{\zeta}$ converge to the desired states $\boldsymbol{\zeta}_d = [\boldsymbol{x}_d^T, \dot{\boldsymbol{x}}_d^T]^T$.

1) Reference Model: The state selection depends on the task requirement; for instance, one might define $\boldsymbol{\zeta} = \dot{\boldsymbol{x}}$ where $\dot{\boldsymbol{\zeta}}_d$ can be directly given or derived from a stable dynamical system $\dot{\boldsymbol{\zeta}}_d = \boldsymbol{f}_d(\boldsymbol{\zeta})$. We consider a general case where the full states of the robot end-effector, $\boldsymbol{\zeta}$, need to follow a desired reference model given by:

$$\dot{\boldsymbol{\zeta}}_r = \mathbf{A}_r\boldsymbol{\zeta}_r + \mathbf{B}_r\boldsymbol{r}(t) \tag{5}$$

where $\boldsymbol{r}(t)$ is a bounded regulation signal. $\mathbf{A}_r$, $\mathbf{B}_r$, and $\boldsymbol{r}(t)$ are design parameters to shape the reference model and have to be such that the dynamic model (5) is stable, from a Lyapunov perspective, at $\boldsymbol{\zeta}_d$, meaning that $\|\boldsymbol{\zeta}_r - \boldsymbol{\zeta}_d\| \to 0$ as $t \to \infty$. In addition to stable control, this model allows modulating the *transient behavior* of the stable convergence. For instance, matrix $\mathbf{A}_r$ controls how fast $\boldsymbol{\zeta}_r$ converges to $\boldsymbol{\zeta}_d$ while $\mathbf{B}_r$ regulates $\boldsymbol{\zeta}_r$ to track $\boldsymbol{\zeta}_d$. The regulation signal, $\boldsymbol{r}(t)$, as the reference signal could be computed from $\boldsymbol{r}(t) = -\mathbf{B}_r^\dagger\mathbf{A}_r\boldsymbol{\zeta}_d$ with $\mathbf{B}_r^\dagger$ being the pseudo inverse of $\mathbf{B}_r$.

2) End-Effector Model: Given $\boldsymbol{\zeta}$ being the concatenation of end-effector position, $\boldsymbol{x}$, and velocity, $\dot{\boldsymbol{x}}$, we can formulate a dynamic model that governs the derivative of these states, $\dot{\boldsymbol{\zeta}}$, by having the acceleration, $\ddot{\boldsymbol{x}}_f$, from Eq.4, to be the input:

$$\dot{\boldsymbol{\zeta}} = \mathbf{A}\boldsymbol{\zeta} + \mathbf{B}\boldsymbol{\nu} + \mathrm{F}(\boldsymbol{\zeta}_d, \boldsymbol{\zeta}) \tag{6}$$

and $\boldsymbol{\nu} = \ddot{\boldsymbol{x}}_f$ is the control effort. Matrices $\mathbf{A}$, and $\mathbf{B}$ are unknown and to be determined by adaptation laws. $\mathrm{F}(\cdot) \in \mathbb{R}^n$, either fully or partially unknown, is a smooth function, and we approximate it by employing universal function approximators $\mathrm{F}(\cdot) = \boldsymbol{\Psi}_\phi^*\boldsymbol{\Phi}(\cdot)$ explained in Appendix IX. Such a model implies that to perfectly track the desired states $\boldsymbol{\zeta}_d$, the controller effort has to take into account unmodeled nonlinearities and adapt for unexpected uncertainties. In a nutshell, the dynamic model (5) and the defined $\boldsymbol{r}(t)$ build the reference model for MRAC that is utilized to find $\ddot{\boldsymbol{x}}_f$ for Eq. (4).

3) Control Rules & Adaptive Laws: To ensure that $\boldsymbol{\zeta}$ track the reference dynamics model (5) in the presence of nonlinearities and uncertainties, we propose the following control rule for the system (6):

$$\boldsymbol{\nu} = \boldsymbol{\Psi}_\zeta\boldsymbol{\zeta} + \boldsymbol{\Psi}_r\boldsymbol{r}(t) + \boldsymbol{\Psi}_\phi\boldsymbol{\Phi}(\boldsymbol{e}) \tag{7}$$

where $\boldsymbol{\Psi}_\zeta$, $\boldsymbol{\Psi}_r$, and $\boldsymbol{\Psi}_\phi$ are approximated online. Our task-space adaptive controller takes $\boldsymbol{\zeta}$ and $\boldsymbol{\zeta}_d$ as input, and outputs $\ddot{\boldsymbol{x}}_r$ to be fed to QP optimization:

$$\ddot{\boldsymbol{x}}_r = \ddot{\boldsymbol{x}}_d + \bar{\boldsymbol{\Psi}}_\zeta\boldsymbol{\zeta} + \bar{\boldsymbol{\Psi}}_r\boldsymbol{r}(t) + \bar{\boldsymbol{\Psi}}_\phi^T\boldsymbol{\Phi}(\boldsymbol{e}) \tag{8}$$

in which $\bar{\boldsymbol{\Psi}}_\zeta$, $\bar{\boldsymbol{\Psi}}_r$, and $\bar{\boldsymbol{\Psi}}_\phi$ are the approximated matrices in Eq.7, based on the following adaptation laws:

$$\begin{aligned}
\dot{\bar{\boldsymbol{\Psi}}}_{\boldsymbol{\zeta}} &= -\Lambda_\zeta\,\mathbf{B}_r^T\,\mathbf{P}\,\boldsymbol{e}\,\boldsymbol{\zeta}^T \\
\dot{\bar{\boldsymbol{\Psi}}}_r &= -\Lambda_r\,\mathbf{B}_r^T\,\mathbf{P}\,\boldsymbol{e}\,\boldsymbol{r}^T(t) \\
\dot{\bar{\boldsymbol{\Psi}}}_\phi &= -\Lambda_\phi\,\mathbf{B}_r^T\,\mathbf{P}\,\boldsymbol{e}\,\boldsymbol{\Phi}(\boldsymbol{e})^T
\end{aligned} \tag{9}$$

where Λ_ζ, Λ_r, and Λ_ϕ are positive definite matrices that tune the convergence rate of the adaptive gains. $\mathbf{P}$ with $\mathbf{A}_r$ satisfy $\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P} = -\mathbf{Q}$ as the necessary and sufficient

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

stability condition for tracking the reference model (5); see Appendix VIII for the stability proof. Note that, in the first trial, we need to initialize the adaptive gains $\bar{\boldsymbol{\Psi}}_\zeta$, $\bar{\boldsymbol{\Psi}}_r$, and $\bar{\boldsymbol{\Psi}}_\phi$ in Eq.9 with imposed upper and lower bounds. The adaptation rate has to be faster than the actual dynamics. Once a task is completed, we can store the trained adaptive gains and use them as the initial guess for new trials. Due to the adaptation laws (9), the proposed controller is now able to cope online with errors in the QP dynamic model and unforeseen uncertainties.

As for the whole system stability when using QP-based controllers, Bouyarmane et al. [4] show that under certain assumptions, QP is stable in terms of solution existence, uniqueness, robustness to perturbation, and continuity. Also, a QP-based controller can be seen as a one-step horizon model predictive controller (MPC). Regarding the stability of MPC, in [44], authors prove (i) the recursive feasibility by showing the existence of a feasible control sequence when starting from a feasible initial point and (ii) the stability by showing that the optimal cost function is a Lyapunov function.

B. Inverse Dynamics Learning Procedure

The task of ID is to provide a model $\boldsymbol{h}^*$ that gives us the torques needed to apply to the system to achieve some desired joint accelerations in a particular robot state:

$$\boldsymbol{\tau} = \boldsymbol{h}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d) \tag{10}$$

This is a classic reformulation of Eq. (1) in order to "see" the equation as a data-driven model/mapping to learn (see [8, 35] for a detailed overview of ID model learning). This is a supervised learning task, and we can employ any suitable learning algorithm. Learning ID model can give us accurate models that can operate inside a control loop and improve tracking performance [35]. To reduce the sample complexity, or in other words to reduce the number of samples needed for achieving good accuracy, we can learn the difference from an available analytic model $\bar{\boldsymbol{h}}$:

$$\boldsymbol{h}^*(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d) = \bar{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d) + \boldsymbol{e}_{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_d) \tag{11}$$

where $\bar{\boldsymbol{h}} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_d + \boldsymbol{C}_{\boldsymbol{g}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{W}$. In essence, we insert prior information coming from our inaccurate yet useful analytical RBD modeling.

In order to be able to exploit the ID model inside the optimization process, it needs to be linear with respect to the optimization variables $\mathcal{X} = \begin{bmatrix}\ddot{\boldsymbol{q}}_d & \boldsymbol{\tau} & \boldsymbol{W}\end{bmatrix}^T$ of the QP. This is required to take full advantage of the ID model that otherwise operates as a *static offset*. Thus, the error model $\boldsymbol{e}_{\boldsymbol{h}}(.) \in \mathbb{R}^j$ must not violate this linearity constraint. At the same time, training a linear model would deteriorate the performance and reduce the flexibility of both the model and the controller.

To overcome this limitation, we propose using more expressive and differentiable models and linearizing them around the current state. In particular, if we assume that the system is in a state $(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t)$, we take the first two terms of the Taylor series expansion of $\boldsymbol{e}_{\boldsymbol{h}}(\cdot, \cdot, \cdot)$:

$$\begin{aligned}\boldsymbol{h}^*(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_{t+1}) = \bar{\boldsymbol{h}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t) + \boldsymbol{e}_{\boldsymbol{h}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t) \\ + (\ddot{\boldsymbol{q}}_{t+1} - \ddot{\boldsymbol{q}}_t)\frac{\partial \boldsymbol{e}_{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})}{\partial \ddot{\boldsymbol{q}}}\Big|_{\substack{\boldsymbol{q}=\boldsymbol{q}_t \\ \dot{\boldsymbol{q}}=\dot{\boldsymbol{q}}_t \\ \ddot{\boldsymbol{q}}=\ddot{\boldsymbol{q}}_t}}\end{aligned} \tag{12}$$

where $\ddot{\boldsymbol{q}}_{t+1}$ contains the desired joint accelerations and is one of the variables optimized by the QP-based controller (i.e., $\ddot{\boldsymbol{q}}_d \equiv \ddot{\boldsymbol{q}}_{t+1}$). Here, it is important to note that for QP optimization, the $\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t$ take fixed values that cannot change during an optimization at each time step.

This linearization yields a loss in expressivity but allows us to insert the models inside the QP-based controller. Nevertheless, because our QP-based controller runs at high frequency (usually $\geq$ 200Hz), the linearized version of the model captures quite well the behavior of the full model in the optimization range and, as we show in the experiments, does not affect the system performance.

1) Gaussian Processes for Inverse Dynamics Learning: We use Gaussian Process Regression (GP) [45] to learn the ID model. We choose GPs because they are accurate, generalize well, and, hence, are suitable for learning from few data points [14, 46]. Another key property of GPs, when combined with prior information, is that they are guaranteed to fall back, in regions far from the data, to the prior model. This property ensures that the QP optimization, which is sensitive to the model it accepts, never fails. In preliminary experiments with neural networks, we observed frequent failures in QP optimization; see also Section IV-B2.

As inputs, we use tuples made of the state vector $\tilde{\boldsymbol{q}} = (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t)$. As training targets, we use the difference between the prediction of the analytic model and the actual command sent: $\boldsymbol{e}_t = \bar{\boldsymbol{h}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_{t+1}) - \boldsymbol{\tau}_t$, where $\boldsymbol{\tau}_t$ is the torque command sent at time t. We use independent GPs to model each dimension of the difference vector $\boldsymbol{e}_t$. For each dimension d of $\boldsymbol{e}_t$, the GP is computed as ($k_{\boldsymbol{e}_d}$ is the kernel function):

$$\hat{\boldsymbol{e}}_d(\tilde{\boldsymbol{q}}) \sim \mathcal{GP}(\mu_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}, k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{q}}'))) \tag{13}$$

Assuming $D^d_{1:N} = \{\boldsymbol{e}_d(\tilde{\boldsymbol{q}}_1), ..., \boldsymbol{e}_d(\tilde{\boldsymbol{q}}_N)\}$ is a set of observations, we can query the GP at a new input point $\tilde{\boldsymbol{q}}_*$:

$$p(\hat{\boldsymbol{e}}_d(\tilde{\boldsymbol{q}}_*)|D^d_{1:N}, \tilde{\boldsymbol{q}}_*) = \mathcal{N}(\mu_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*), \sigma^2_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*)) \tag{14}$$

The mean and variance predictions of this GP are computed using a kernel vector $\boldsymbol{k}_{\hat{\boldsymbol{e}}_d} = k(D^d_{1:N}, \tilde{\boldsymbol{q}}_*)$, and a kernel matrix $K_{\hat{\boldsymbol{e}}_d}$ with entries $K^{ij}_{\hat{\boldsymbol{e}}_d} = k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{q}}_j)$:

$$\begin{aligned}\mu_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*) &= \boldsymbol{k}^T_{\hat{\boldsymbol{e}}_d} K^{-1}_{\hat{\boldsymbol{e}}_d} D^d_{1:N} \\ \sigma^2_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*) &= k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_*, \tilde{\boldsymbol{q}}_*) - \boldsymbol{k}^T_{\hat{\boldsymbol{e}}_d} K^{-1}_{\hat{\boldsymbol{e}}_d} \boldsymbol{k}_{\hat{\boldsymbol{e}}_d}\end{aligned} \tag{15}$$

We use the exponential kernel [45] in this study: $k_{\hat{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_p, \tilde{\boldsymbol{q}}_q) = \sigma^2_d \exp(-\frac{1}{2}(\tilde{\boldsymbol{q}}_p - \tilde{\boldsymbol{q}}_q)^T \boldsymbol{\Lambda}^{-1}_d (\tilde{\boldsymbol{q}}_p - \tilde{\boldsymbol{q}}_q)) + \delta_{pq}\sigma^2_{n_d}$, where δ_{pq} equals 1 when $p = q$ and 0 otherwise, and $[\boldsymbol{\Lambda}_d, \sigma^2_d, \sigma^2_{n_d}]$ is the vector of hyperparameters of the kernel (length scales for each dimension, signal variance, and noise).

To linearize the learned model around a query point, we need to compute the derivative of our GPs. The derivative is another GP, and its existence depends on the differentiability of its kernel function [45]. In our particular case, the squared exponential kernel that we use is infinitely differentiable, and the associated GP has infinitely many derivatives. In this paper, we do not use the predicted GP variance, and we only detail the derivatives of $\mu_{\hat{\boldsymbol{e}}_d}$ with respect to the input point $\tilde{\boldsymbol{q}}_*$. If we look at Eq.15 closely, only $\boldsymbol{k}_{\hat{\boldsymbol{e}}_d}$ depends on the input point $\tilde{\boldsymbol{q}}_*$, meaning that we just need to differentiate the kernel function.

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

Assuming that we have only one sample for training $\tilde{\boldsymbol{q}}_i$, we can compute the derivative of the kernel as follows:

$$\begin{aligned}\frac{\partial k_{\dot{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{q}}_*)}{\partial \tilde{\boldsymbol{q}}_*} &= \frac{\partial\Big(\sigma_d^2 \exp(-\frac{1}{2}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i)^T \boldsymbol{\Lambda}_d^{-1}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i))\Big)}{\partial \tilde{\boldsymbol{q}}_*} \\ &= \frac{\partial\Big(-\frac{1}{2}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i)^T \boldsymbol{\Lambda}_d^{-1}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i)\Big)}{\partial \tilde{\boldsymbol{q}}_*} k_{\dot{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{q}}_*) \\ &= -\boldsymbol{\Lambda}_d^{-1}(\tilde{\boldsymbol{q}}_* - \tilde{\boldsymbol{q}}_i) k_{\dot{\boldsymbol{e}}_d}(\tilde{\boldsymbol{q}}_i, \tilde{\boldsymbol{q}}_*) \end{aligned} \tag{16}$$

It is trivial to generalize/compute the gradient when considering a set of training points.

2) Practical Considerations: Gaussian Process Regression has a training time complexity of $O(n^3)$ and is thus impractical when having to deal with many samples. Since our controllers operate at high frequency (around 200 Hz in our experiments), we can easily gather big datasets. There are many approaches to approximately learning GPs that reduce the time complexity [47], but we chose to subsample the data in order to reduce the number of points. In practice, we keep 10% to 20% of the data. In order to avoid unbalanced subsampling and catastrophic forgetting, we keep all the data (ordered in time) and take one sample every 5 or 10 timesteps. We performed initial experiments in the simulated environments with and without subsampling, and we did not observe any significant deterioration of the performance of the SCQP algorithm while achieving real-time querying of the GPs.

Another important point that we considered is the careful mixing of the prior model and the error model learned by the GPs. The QP controllers are model sensitive and can easily fail if there are inconsistencies. For this reason, we did not optimize the hyperparameters of the GPs so we could consistently fall back to the prior model away from data points. Initial experiments with hyperparameter optimization frequently led to QP optimization failures. Conversely, the GPs without hyperparameter optimization rarely triggered QP failures. We use the limbo C++11 library for GP regression and the GP derivatives [48]. In all experiments (simulation and real-world), querying from GPs is in real-time, and the maximum processing time we have observed for updating GPs between episodes (line 12 of Algorithm 1) was less than 3s.

Algorithm 1 Self-Correcting QP-based Control

1: Design the task specifications: $\boldsymbol{x}_d(t)$
2: Configure the adaptive controller and the model learning procedure
3: **for** $n = 1 \rightarrow N_{\text{episodes}}$ **do** ▷ For each episode
4: **for** $t = 0 \rightarrow T$ **do** ▷ For each time step
5: Get $\ddot{\boldsymbol{x}}_r$ from Eq.8
6: Update the reference of cost function in QP given the $\ddot{\boldsymbol{x}}_r$
7: Get $\boldsymbol{\tau}_t$ from the QP using the updated cost function and the learned model $\boldsymbol{h}^*$ (from Eq.12)
8: Apply $\boldsymbol{\tau}_t$ to the robot
9: Collect data $\{\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \ddot{\boldsymbol{q}}_t, \boldsymbol{\tau}_t\}$
10: Update the adaptive controller with Eq.9
11: **end for**
12: ID model learning (Section IV-B)
13: **end for**

V. SIMULATED EXPERIMENTS

With our simulated experiments, we aim to answer the following questions:

- Can the SCQP method cope with big model mismatches?
- Can the SCQP method generalize to several different scenarios and robot setups?
- Can the SCQP method generalize to high-dimensional robots (e.g., humanoids) and contact-rich tasks?
- Is learning the ID alone or using task-space adaptive control alone enough?

To answer the questions, we devise the following scenarios:

i. A 7-DoFs KUKA LBR iiwa manipulator (14kg version) that tracks end-effector trajectories with an unknown mass attached to the end-effector.
ii. A 32-DoFs PAL Robotics Talos humanoid robot[4] that performs a waving task while having unknown masses attached to both of the hands and/or unknown friction coefficients.
iii. Two 7-DoFs KUKA LBR iiwa manipulators (14kg versions) that coordinate in order to manipulate a box with unknown mass; we provide preliminary results for this scenario.

For the above scenarios (except the experimental bimanual task), we experiment with the following approaches:

i. Our SCQP approach (see Section IV-B).
ii. Learning the ID model (as in Section IV-B1) combined with a PID end-effector controller (the case where the adaptive controller is absent).
iii. Using the adaptive controller without any model learning.

If not stated otherwise, we control the robot(s) at 200 Hz, and each episode has a length of $10\,s$. We utilize the DART simulator [49] with the robot_dart wrapper[5]. Note that all robots used in this paper can be controlled directly in torque mode, i.e., we can send the computed joint torques as the direct command for the robot actuators.

A. KUKA LBR iiwa Trajectory Tracking

We begin the evaluation of our methods using a 7-DoFs KUKA iiwa manipulator that needs to track specified end-effector trajectories. We will perform two types of experiments: (a) a task that involves moving the end-effector only along one axis (z-axis/gravity direction) and (b) a task that involves moving the end-effector along two axes (the yz-plane). We perform these two tasks to extensively evaluate our method against baselines. In order to emulate real-world uncertainties, we consider the following model mismatches:

i. The QP controller assumes perfect actuators (no Coulomb friction or damping), whereas, in the real world, the actuators have both Coulomb friction and damping.
ii. A 1 kg mass attached to the end-effector of the actual KUKA that is not presented to the QP model.

The first mismatch represents the typical differences between the *ideal* and the *actual* model of the actuators. The second

[4]We disable the hands and use the 30-DoFs.

[5]https://github.com/resibots/robot_dart/

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

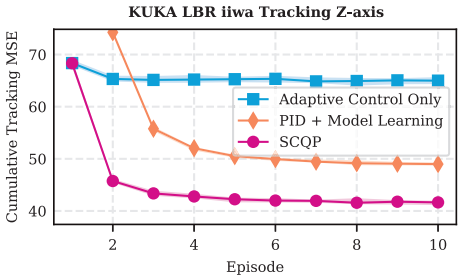

Fig. 2: Results of the simulated experiment with KUKA iiwa, z-axis tracking. The experiment is repeated 20 times, each consisting of 10 episodes in succession. The tracking error for an episode is the cumulative mean square tracking error over time steps. Solid lines are the median over 20 replicates, and the shaded regions are the regions between the 5th and 95th percentiles.

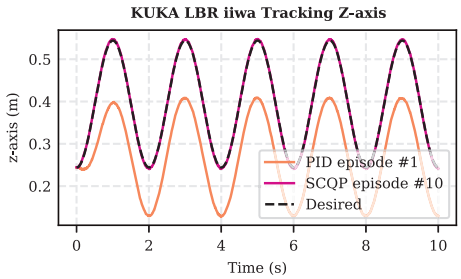

Fig. 3: KUKA iiwa z-axis tracking trajectories before and after learning: the first episode with PID and wrong QP model where the model mismatch affects the tracking, and the last episode with the SCQP approach which is able to track the desired trajectory.

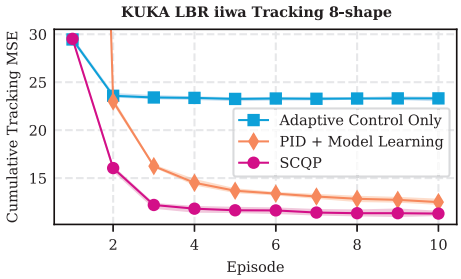

Fig. 4: Results of the 8-shape trajectory tracking experiment on KUKA iiwa in a simulated environment. The experiment is repeated 20 times, each consisting of 10 episodes in succession. The tracking error for an episode is the cumulative mean square tracking error over time steps. Solid lines are the median over 20 replicates, and the shaded regions are the regions between the 5th and 95th percentiles.

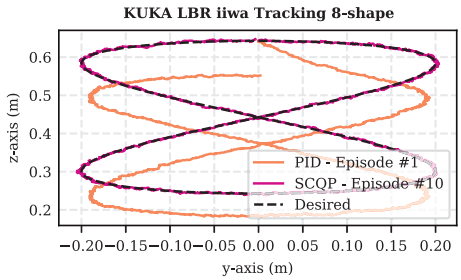

Fig. 5: KUKA iiwa yz-axis tracking before and after learning: typical trajectories across episodes with PID and the SCQP approach. SCQP enables accurate tracking of the desired trajectory.

mismatch has a significant effect on the dynamics of the robots and simulates an exaggerated case of an unexpected situation. Imagine the case, for instance, where the robot is lifting an object, and abruptly, the force-torque sensor fails and outputs zero wrenches at the end-effector. Lastly, we add noises in the joint position and velocity measurements, emulating noisy real-sensor feedback.

1) QP Configuration: Here, we configure the QP with the desired Cartesian acceleration of the end-effector given by:

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{17}$$

Thus, the QP formulation can be written as follows:

$$\begin{aligned} \boldsymbol{H}_{\text{acc}} &= [\boldsymbol{J}(\boldsymbol{q}) \quad \boldsymbol{0} \quad \boldsymbol{0}] \\ \boldsymbol{b}_{\text{acc}} &= \ddot{\boldsymbol{x}}_r - \dot{\boldsymbol{J}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \end{aligned} \tag{18}$$

Finally, we add regularization constraints such as joint position and velocity limits, preferred null joint configurations, actuator torque limits, etc. Hence the optimization variables do not violate the functional limits, and the robot remains stable.

2) Z-axis Tracking: The objective here is that the robot's end-effector has to follow a sinusoidal trajectory on the z-axis. The results showcase that when using the SCQP, the learning converges faster (i.e., in fewer trials/episodes) and to a more accurate model than when using the baselines; see Figure 2. The PID controller, despite being tuned with high gains, yields poor tracking with a high cost. This is because the model mismatch is, indeed, significant, and the PID controller cannot compensate for it without the aid of the learned ID model. The adaptive controller achieves a relatively better cost even in the initial trials, and the tracking performance, without the learned model, improves over the remaining trials. However, it is inadequate to compensate for all the unmodeled dynamics, for which it needs the learned ID model. In other words, we observe that the model flexibility issue in fast online learning with the adaptive controller is tackled by learning the residual dynamics in SCQP.

Qualitatively, the SCQP approach can quickly compensate for model mismatches and requires less than 5-6 episodes to achieve desirable trajectory tracking. Figure 3 showcases typical z-axis trajectories before and after learning.

3) YZ-plane Tracking: In this task, the robot's end-effector has to follow an 8-shaped trajectory in the yz-plane. The results closely follow the outcomes of the previous scenario (Figure 4). The SCQP approach converges in fewer episodes than both baselines. Model learning with a PID end-effector controller can achieve good results but requires more episodes to converge. The adaptive controller alone cannot sufficiently compensate for all the model mismatches.

Qualitatively, the SCQP approach can quickly compensate for model mismatches and requires less than 5-6 episodes to achieve desirable trajectory tracking. Figure 5 illustrates typical yz-plane trajectories of the SCQP and the PID+model learning approaches.

B. Talos Humanoid Task

Here, we control a 32-DoFs Talos humanoid robot. While maintaining balance, the robot has to follow a sinusoidal trajectory with the right arm and keep the left arm in place; see Figure 6, left. We devised this scenario to evaluate our approach in high-dimensional state/action spaces with a more

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

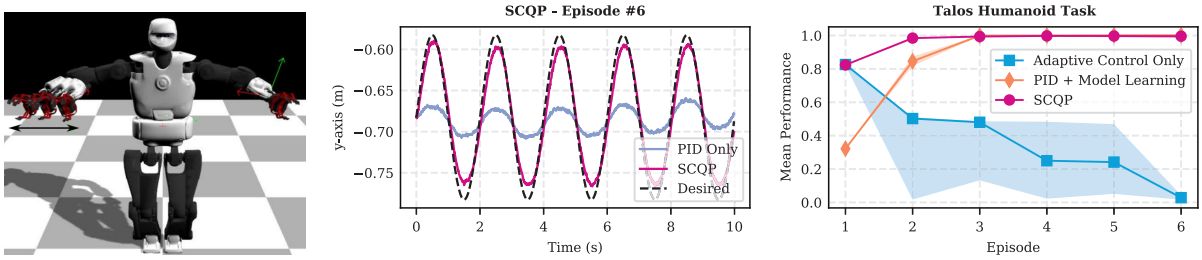

Fig. 6: Successful trial of the Talos task. Through ID learning and online adaptation, SCQP learns to perform the waving task (left) without falling despite relatively big mismatches in the mass of the arms and the friction coefficient. The phase tracking accuracy during the waving task (middle) indicates the effectiveness of model learning in SCQP after only a few trials compared to the other approaches (right). Solid lines are the median over 10 replicates, and the shaded regions are the areas between the 25th and 75th percentiles.

complex QP task. In addition to considering, for each arm, the same mismatch models as those in Section V-A, we assume that the friction coefficients of the contact surfaces are not known. The real world has a coefficient of friction set to 0.7, whereas the QP model takes a coefficient of 1, i.e., contact surfaces are more slippery than what the QP expects. This assumption attempts to create a model mismatch in a crucial part of the environment, strongly affecting the stabilization of the humanoid robot. We again add noise to the joint position and velocity measurements.

1) QP Configuration: We define six Cartesian acceleration tasks following Eq.18: (i) one tracking task per arm end-effector, (ii) two tracking tasks for the torso (COM position and upright preference), and (iii) one zero acceleration task (without feedback) for each foot. We also define one 6D contact constraint per foot to handle the balance of the humanoid (we assume the contact points are in the middle of the feet). More precisely, for each foot, we define a contact as inequality constraints of the following form:

$$\begin{aligned} \boldsymbol{H}_{I\text{contact}} &= [\mathbf{0} \quad \mathbf{0} \quad \boldsymbol{C}] \\ \boldsymbol{b}_{I\text{contact}} &= \begin{bmatrix} -\infty & -\infty & 0 & 0 & F_{min} \\ 0 & 0 & \infty & \infty & F_{max} \end{bmatrix} \end{aligned} \tag{19}$$

where $\boldsymbol{C}$ is defined as: $\begin{bmatrix} -\mu \mathbf{1}_2 \\ \mu \mathbf{1}_2 \\ 1 \end{bmatrix} \boldsymbol{n}^T + \begin{bmatrix} \boldsymbol{I}_2 \\ \boldsymbol{I}_2 \\ 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{t}_1 & \boldsymbol{t}_2 \end{bmatrix}^T$ and where $\mu \in \mathbb{R}^+$ is the coefficient of friction, $\boldsymbol{n} \in \mathbb{R}^3$ is the contact normal, and $\boldsymbol{t}_i \in \mathbb{R}^3$ are the tangential directions. We also add a few rows for constraining the center of pressure similar to [31]. Instead of using a center of pressure constraint, one can use four contact points in the corners of the foot.

2) Results: The results show that our SCQP method can scale to high-dimensional systems and handle passive contacts; see Figure 6. The SCQP converges faster (i.e., in fewer trials/episodes) to high-performance controllers than the baselines. In this scenario, the adaptive controller alone might diverge if not properly tuned (due to the smaller value of the coefficient of friction). Thus, to showcase this drift issue and also the importance of the learned ID model, we chose a parameter configuration in which the adaptive controller alone diverges. Then, we used the same parameters for SCQP where model learning is present. The gains of the PID controller, however, are tuned to get its best performance for maintaining the robot's stability. In Figure 6 (right), we observe that the learned model "stabilizes" the adaptive controller that would diverge if left on its own.

Qualitatively, the SCQP approach is able to quickly compensate for model mismatches and requires less than 3-4 episodes to achieve desirable trajectory tracking. Figure 6 (middle) showcases a typical y-axis trajectory. The supplementary video shows an example of the learning procedure[6].

C. Preliminary Experiments on Bimanual Manipulation

In this scenario, we perform a bimanual manipulation task in which two KUKA LBR iiwa arms have to lift a box of unknown mass; see Figure 7. This experiment is used to simulate contact-rich tasks and tasks where objects need to be manipulated. We consider the following model mismatches that act in combination (similar to the previous):

i. The QP controller assumes perfect actuators (that is, no Coulomb friction or damping), whereas in the real world, the actuators have both Coulomb friction and damping.
ii. The mass of the box is not well-calibrated, and there is a mismatch of 0.5 kg (the real box has a mass of 1.5 kg, whereas the QP model assumes 1 kg).

1) QP Configuration: To control the robots while performing the bimanual task, we extend the decision variables to contain the acceleration and torques of all entities, i.e., the two robots and the box. We add one constraint for the dynamics of each entity; see Eq.2. To "connect" the two arms with the box, we have two sets of contact forces for the contact between each arm and the box. We then couple the dynamics of the individual entities by inserting the forces at the correct places, i.e., $\boldsymbol{J}_{arm}^T \boldsymbol{W}$ for the arms and $-\boldsymbol{J}_{box}^T \boldsymbol{W}$ for the box, since the forces acting on the box are identical in magnitude and in the opposite direction to the actions of the ones on the arms.

2) Results: We provide preliminary results of this contact-rich task that involves active contacts for manipulation. The results illustrate that our SCQP approach can be used to learn this type of task. The most challenging part is learning the ID model. We were able to consistently learn a good model within three episodes, but after the 3rd episode, our model learning pipeline did not work consistently: we obtained significant mean square errors in the training set, meaning that something did not go well with the model learning. Nevertheless, three episodes were enough for the SCQP approach to improve the

[6]The video is also available at https://youtu.be/cA-_SKoO_9c.

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

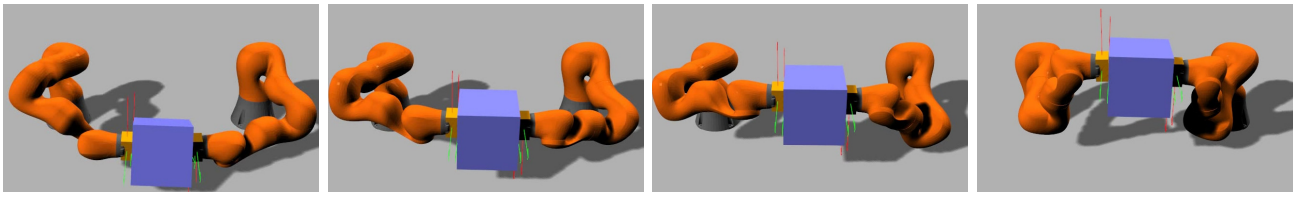

Fig. 7: From left to right, screenshots of a successful trial of a bimanual manipulation task: grasping and manipulating a box with unknown mass. Through ID learning and online adaptation, SCQP learns to insert higher contact force to avoid slippage. Also, thanks to the torque control scheme, the robot configuration varies during task execution to maintain compliant joint behavior.

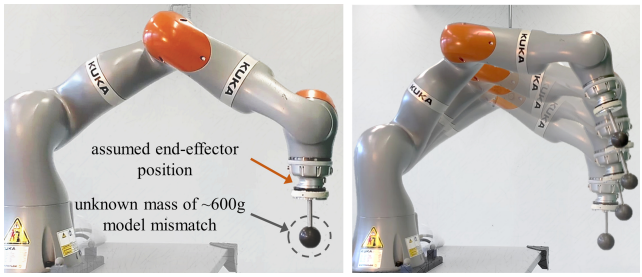

Fig. 8: Physical robot setup with a mass mismatch at the end-effector. The KUKA manipulator needs to track a desired trajectory. This task is similar to Section V-A but is here applied to a real robot where overcoming joints' friction and nonlinearities in real-time control add to the control problem.

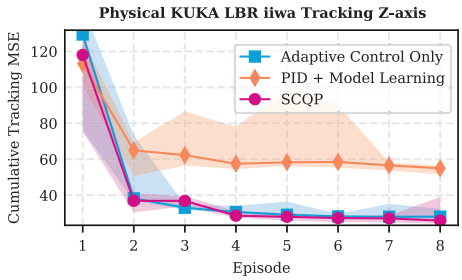

Fig. 9: Results of real KUKA iiwa experiments with end-effector mass mismatch in the z-axis tracking. The experiment is repeated 10 times, each consisting of 10 episodes in succession. The tracking error for an episode is the cumulative mean square tracking error over time steps. Solid lines are the median over 10 replicates, and the shaded regions depict the region of the 5th to 95th percentiles.

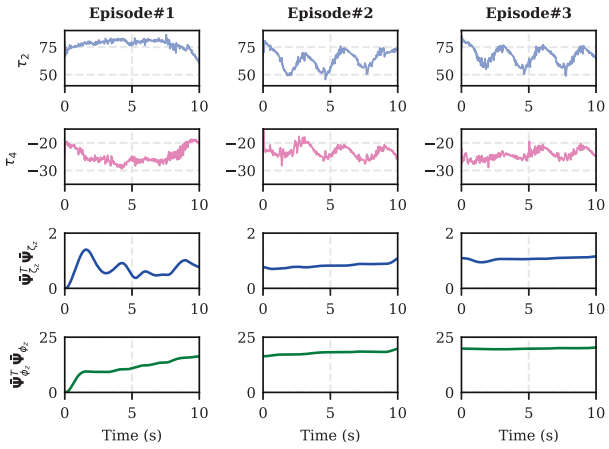

Fig. 10: Examples of control responses and adaptive gains during the first three episodes of the periodic trajectory tracking task (see Figure 9). The first two rows are the second and the fourth joint torques (the most load-bearing joints) in $N.m.s^{-1}$, respectively. The third and the fourth rows are the norms of linear, $\bar{\mathbf{\Psi}}_{\zeta_z}^T \bar{\mathbf{\Psi}}_{\zeta_z}$, and nonlinear, $\bar{\mathbf{\Psi}}_{\phi_z}^T \bar{\mathbf{\Psi}}_{\phi_z}$, adaptation gains active on the z-axis, respectively. After the first episode, the learned residual dynamic model results in modifying the torque commands to achieve a higher tracking accuracy consistently; see Figure 9. Also, adaptive gains converge to constant values, and since the residual model learning handles big model mismatches, the online adaptation remains reactive to uncertainties faced online.

VI. PHYSICAL ROBOT EXPERIMENTS

To validate our SCQP approach in the physical world, we devise two setups. The first is similar to Section V-A, where a KUKA manipulator needs to track a desired trajectory. The second setup demonstrates the SCQP application in a pick-and-place task using a robotic hand while both the hand's and objects' dynamics are unknown. Apart from the imposed mass mismatch in both setups, there is also the "reality gap" in this scenario since many small yet important details are not modeled in our original QP model (e.g., idealistic actuators). The robot is controlled at 200 Hz, and each episode lasts $10\,s$.

A. Tracking Periodic Trajectory on Z-axis

As in the simulated experiment in Section V-A, the robot has to follow a periodic trajectory on the z-axis with a mass mismatch at the end-effector (around 0.6 kg mismatch, see Figure 8). The results show that SCQP works in a physical system and provides similar performance to the simulated variant; see Figure 9. In particular, the SCQP converges to low-error trajectory tracking in less than $30 - 40\,s$ of interaction time (3-4 episodes); see Figures 9 and 10. Additionally, it performs better than the PID+model learning baseline and performs comparably to the adaptive control alone. Given the lower variance in tracking error (over 10 replicates), the SCQP shows higher consistency and robustness in tracking compared to the baselines. Qualitatively, the SCQP approach can quickly compensate for model mismatches and requires less than 5-6 episodes to achieve desirable trajectory tracking robustly. The supplementary video shows an example of the learning and control procedure.

B. Pick-and-Place with a Robotic Hand

In the experiment, we perform a pick-and-place scenario with a robotic hand. The task is to grasp different objects from a certain place and drop them into a bucket fixed in another position; see Figure 11. In this experiment, the dynamics and physical properties (e.g., mass and inertia) of the robotic hand and the objects are unknown. We use objects of different weights for the task to showcase that SCQP can account for

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

online uncertainties in addition to residual model learning. The experiment is an extreme use case of SCQP where it knows nothing about the hand and the objects to be grasped (a rather unrealistic assumption since we usually have some idea about the properties of the hand and the objects). Figure 11 shows that SCQP can execute the task successfully despite significant unmodeled dynamics at the end-effector and model changes from one object to the other. The supplementary video shows examples of task execution for this experiment.

VII. CONCLUSION

In this work, we proposed a novel combination of an MRAC scheme with an ID model augmented QP-controller. Our main intuition was to merge a fast online adaptive control law in task-space to regulate the cost function of the QP with a slower ID model learning procedure inserted inside the QP model. This pipeline, called SCQP, was effective, and we could successfully apply it to many different scenarios and robots.

In particular, the SCQP was able to compensate, in a handful of trials, for large model inaccuracies in tasks ranging from simple end-effector tracking to humanoid balancing and contact-rich tasks. Using SCQP, one can avoid tedious PID controller tuning while capturing significant unmodeled dynamics with the learned ID model.

Despite the successful application of SCQP, there remain several limitations that we would like to address in the future: (a) SCQP requires either a different model for each contact configuration of the system or a learning model that can generalize to different contact configurations, and (b) learning the ID model is itself a difficult task (as we observed in Section V-C). Although there exist methods for learning ID models with contacts [50], learning effectively ID models from unstructured data[7] remains an open problem that deserves further investigation.

Finally, in this work, we assumed that the high-level tasks remained fixed throughout the process. It is straightforward to combine SCQP with methods that update online the high-level planning part. Task modulating can be crucial if the original specifications are not achievable; for example, the left arm is damaged and the task has to be performed with the right arm. Moreover, our approach can be combined with reinforcement learning algorithms. One idea is to enable exploration around the commands the QP outputs while ensuring safety and not allowing deviations that could potentially harm the robot.

REFERENCES

[1] C. Chen, Z. Liu, Y. Zhang, and S. Xie, "Coordinated motion/force control of multiarm robot with unknown sensor nonlinearity and manipulated object's uncertainty," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 7, pp. 1123–1134, 2017.

[2] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[3] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[4] K. Bouyarmane and A. Kheddar, "On weight-prioritized multitask control of humanoid robots," *IEEE Trans. on Automatic Control*, vol. 63, no. 6, pp. 1632–1647, 2017.

[5] Y. Zhang, S. S. Ge, and T. H. Lee, "A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators," *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 5, pp. 2126–2132, 2004.

[6] C. Collette *et al.*, "Dynamic balance control of humanoids for multiple grasps and non coplanar frictional contacts," in *International Conference on Humanoid Robots*, 2007.

[7] J. Nakanishi *et al.*, "Operational space control: A theoretical and empirical comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.

[8] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive processing*, vol. 12, no. 4, pp. 319–340, 2011.

[9] V. Modugno *et al.*, "Learning soft task priorities for control of redundant robots," in *International Conference on Robotics and Automation*, 2016.

[10] K. Chatzilygeroudis *et al.*, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *IEEE Trans. on Robotics*, 2019.

[11] J. Spitz *et al.*, "Trial-and-error learning of repulsors for humanoid qp-based whole-body control," in *International Conference on Humanoid Robots*, 2017.

[12] R. Lober, V. Padois, and O. Sigaud, "Efficient reinforcement learning for humanoid whole-body control," in *International Conference on Humanoid Robots*, 2016.

[13] K. Chatzilygeroudis and J.-B. Mouret, "Using parameterized black-box priors to scale up model-based policy search for robotics," in *International Conference on Robotics and Automation*, 2018.

[14] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

[15] P. A. Ioannou and J. Sun, *Robust adaptive control*. Courier Corporation, 2012.

[16] Y. Abe, M. Da Silva, and J. Popović, "Multiobjective control with frictional contacts," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 249–258, 2007.

[17] M. De Lasa, I. Mordatch, and A. Hertzmann, "Feature-based locomotion controllers," *ACM Trans. on Graphics (TOG)*, vol. 29, no. 4, pp. 1–10, 2010.

[18] S. Coros, P. Beaudoin, and M. Van de Panne, "Generalized biped walking control," *ACM Trans. on Graphics (TOG)*, vol. 29, no. 4, pp. 1–9, 2010.

[19] Z. Li *et al.*, "Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 6, pp. 740–749, 2015.

[20] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex

[7] We do not have access to a full static dataset, but rather collect data as we apply the controller on the system.

Digital Object Identifier 10.1109/TSMC.2023.3262954
Link to the paper on IEEE Xplore: https://ieeexplore.ieee.org/document/10102575

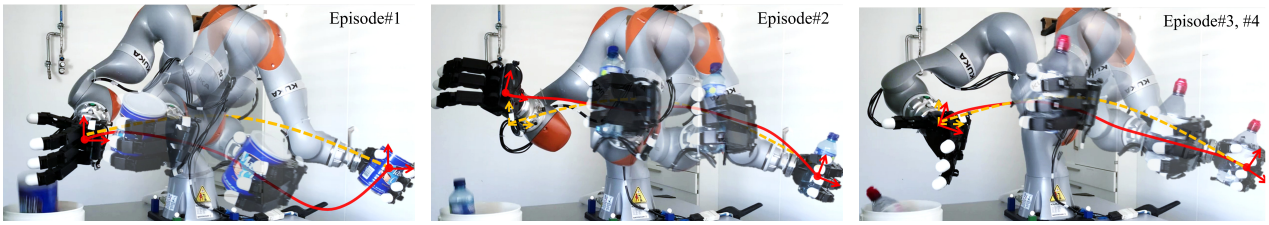

Fig. 11: Examples of pick-and-place task with a robotic hand. The robot needs to grasp different objects, follow a specific trajectory (dashed yellow line) while keeping the objects vertical, and place them into a fixed bucket. The dynamics and physical properties of the robotic hand and the objects are unknown (the robotic hand is around 1.5kg, and the objects are 250 ± 40g), and different objects are used for each episode. In the first episode, the robot has a significant deviation (solid red line) from the desired trajectory; however, after residual model learning, the tracking error reduces progressively in the other episodes. In the third episode, the robot fails to drop the object in the target place; nevertheless, the task is successfully completed in the fourth episode with the same object.

humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *International Conference on Robotics and Automation*, 2011.

[21] Z. Li *et al.*, "Dynamic balance optimization and control of quadruped robot systems with flexible joints," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 10, pp. 1338–1351, 2016.

[22] S. Kuindersma *et al.*, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, no. 3, pp. 429–455, 2016.

[23] R. Lober, V. Padois, and O. Sigaud, "Multiple task optimization using dynamical movement primitives for whole-body reactive control," in *International Conference on Humanoid Robots*, 2014.

[24] T. Marcucci *et al.*, "Parametric trajectory libraries for online motion planning with application to soft robots," in *Robotics Research*, pp. 1001–1017, Springer, 2020.

[25] F. Burget, A. Hornung, and M. Bennewitz, "Whole-body motion planning for manipulation of articulated objects," in *International Conference on Robotics and Automation*, 2013.

[26] A. Herzog *et al.*, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[27] K. Bouyarmane *et al.*, "Quadratic programming for multirobot and task-space force control," *IEEE Trans. on Robotics*, vol. 35, no. 1, pp. 64–77, 2018.

[28] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *International Conference on Robotics and Automation*, 2014.

[29] J. Vaillant, K. Bouyarmane, and A. Kheddar, "Multi-character physical and behavioral interactions controller," *IEEE Trans. on visualization and computer graphics*, vol. 23, no. 6, pp. 1650–1662, 2016.

[30] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition," in *International Conference on Robotics and Automation*, 2010.

[31] S. Feng *et al.*, "Optimization based full body control for the atlas robot," in *International Conference on Humanoid Robots*, 2014.

[32] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[33] W. He *et al.*, "Model identification and control design for a humanoid robot," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 45–57, 2017.

[34] F. Khadivar *et al.*, "Efficient configuration exploration in inverse dynamics acquisition of robotic manipulators," in *International Conference on Robotics and Automation*, 2021.

[35] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *International Conference on Robotics and Automation*, 2010.

[36] V. Modugno *et al.*, "Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization," in *International Conference on Humanoid Robots*, 2016.

[37] J. Ichnowski *et al.*, "Accelerating quadratic optimization with reinforcement learning," *arXiv preprint arXiv:2107.10847*, 2021.

[38] C. Sun *et al.*, "Adaptive neural network control of biped robots," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 2, pp. 315–326, 2017.

[39] Z. Wang *et al.*, "Distributed formation control of nonholonomic wheeled mobile robots subject to longitudinal slippage constraints," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 5, pp. 2992–3003, 2019.

[40] G. Tao, *Adaptive control design and analysis*, vol. 37. John Wiley & Sons, 2003.

[41] V. Azimi *et al.*, "Model-based adaptive control of transfemoral prostheses: Theory, simulation, and experiments," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 2, pp. 1174–1191, 2019.

[42] M. Sharifi *et al.*, "Adaptive cpg-based gait planning with learning-based torque estimation and control for exoskeletons," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8261–8268, 2021.

[43] P. Culbertson, J.-J. Slotine, and M. Schwager, "Decentralized adaptive control for collaborative manipulation of rigid bodies," *IEEE Trans. on Robotics*, vol. 37, no. 6, pp. 1906–1920, 2021.

[44] D. Limón *et al.*, "On the stability of constrained mpc without terminal constraint," *IEEE Trans. on automatic control*, vol. 51, no. 5, pp. 832–836, 2006.

[45] C. E. Rasmussen and C. K. Williams, *Gaussian processes*

for machine learning, vol. 1. MIT press Cambridge, 2006.

[46] K. Chatzilygeroudis *et al.*, "Black-box data-efficient policy search for robotics," in *International Conference on Intelligent Robots and Systems*, 2017.

[47] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[48] A. Cully *et al.*, "Limbo: A flexible high-performance library for gaussian processes modeling and data-efficient optimization," *Journal of Open Source Software*, vol. 3, no. 26, 2018.

[49] J. Lee *et al.*, "DART: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, p. 500, Feb 2018.

[50] R. Calandra *et al.*, "Learning inverse dynamics models with contacts," in *International Conference on Robotics and Automation*, 2015.

[51] P. Culbertson and M. Schwager, "Decentralized adaptive control for collaborative manipulation," in *International Conference on Robotics and Automation*, 2018.

[52] H. K. Khalil and J. W. Grizzle, *Nonlinear systems*, vol. 3. Prentice hall Upper Saddle River, NJ, 2002.

[53] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela, "Bayesian nonparametric adaptive control using gaussian processes," *IEEE Trans. on neural networks and learning systems*, vol. 26, no. 3, pp. 537–550, 2014.

VIII. Adaptive Control Stability Proof

Let $\mathbf{e} = \boldsymbol{\zeta} - \boldsymbol{\zeta}_r$ be the tracking error. Using Eq. 5, 6 and 7, the error's dynamics (time derivative) is given by:

$$\dot{\boldsymbol{e}} = (\mathbf{A} + \mathbf{B}\boldsymbol{\Psi}_\zeta)\boldsymbol{\zeta} - \mathbf{A}_r\boldsymbol{\zeta}_r + (\mathbf{B}\boldsymbol{\Psi}_r - \mathbf{B}_r)\boldsymbol{r}(t) + \mathbf{B}\boldsymbol{\Psi}_\phi\boldsymbol{\Phi}(\boldsymbol{e}) + \mathrm{F}(\boldsymbol{\zeta}_d, \boldsymbol{\zeta}). \quad \text{(A1)}$$

For the error dynamic (A1) to follow the reference dynamic (5), we set that the desired gain matrices $\boldsymbol{\Psi}^*_\zeta$, $\boldsymbol{\Psi}^*_r$, and $\boldsymbol{\Psi}^*_\phi$ satisfy:

$$\begin{aligned} \mathbf{A} + \mathbf{B}\boldsymbol{\Psi}^*_\zeta &= \mathbf{A}_r \\ \mathbf{B}\boldsymbol{\Psi}^*_r &= \mathbf{B}_r \\ \mathbf{B}\boldsymbol{\Psi}^*_\phi\boldsymbol{\Phi}(\boldsymbol{e}) &= -\mathrm{F}(\boldsymbol{\zeta}_d, \boldsymbol{\zeta}). \end{aligned} \quad \text{(A2)}$$

Given that system (5) is by construction stable (Matrices $\mathbf{P}$, $\mathbf{A}_r$ satisfy $\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P} = -\mathbf{Q}$; see Section IV-A), then $\boldsymbol{e} \to 0$. We define the gain prediction errors as $\widetilde{\boldsymbol{\Psi}}_\zeta = \boldsymbol{\Psi}_\zeta - \boldsymbol{\Psi}^*_\zeta$, $\widetilde{\boldsymbol{\Psi}}_r = \boldsymbol{\Psi}_r - \boldsymbol{\Psi}^*_r$, and $\widetilde{\boldsymbol{\Psi}}_\phi = \boldsymbol{\Psi}_\phi - \boldsymbol{\Psi}^*_\phi$. Replacing (A2) in (A1), we obtain:

$$\dot{\boldsymbol{e}} = \mathbf{A}_r\boldsymbol{e} + \mathbf{B}\widetilde{\boldsymbol{\Psi}}_\zeta\boldsymbol{\zeta} + \mathbf{B}\widetilde{\boldsymbol{\Psi}}_r\boldsymbol{r}(t) + \mathbf{B}\widetilde{\boldsymbol{\Psi}}_\phi\boldsymbol{\Phi}(\boldsymbol{e}) \quad \text{(A3)}$$

Theorem 1. *The error dynamics (A3) is Lyapunov stable, and the tracking error, $\boldsymbol{e}$, vanishes asymptotically under the proposed control law (7) and adaptation law (9).*

Proof. Consider the following Lyapunov function:

$$\mathbf{V}(\boldsymbol{e}, \widetilde{\boldsymbol{\Psi}}_\zeta, \widetilde{\boldsymbol{\Psi}}_r, \widetilde{\boldsymbol{\Psi}}_\phi) = \frac{1}{2}\boldsymbol{e}^T\mathbf{P}\boldsymbol{e} + \frac{1}{2}\mathbf{tr}\big(\widetilde{\boldsymbol{\Psi}}_\zeta^T\boldsymbol{\Theta}_\zeta\widetilde{\boldsymbol{\Psi}}_\zeta + \widetilde{\boldsymbol{\Psi}}_r^T\boldsymbol{\Theta}_r\widetilde{\boldsymbol{\Psi}}_r + \widetilde{\boldsymbol{\Psi}}_\phi^T\boldsymbol{\Theta}_\phi\widetilde{\boldsymbol{\Psi}}_\phi\big) \quad \text{(A4)}$$

where $P, \boldsymbol{\Theta}_\zeta$, $\boldsymbol{\Theta}_r$, and $\boldsymbol{\Theta}_\phi \succ 0$. V is positive. It is zero when both the tracking error vanishes and the gains no longer vary. Taking the time derivative of Eq.A4 yields:

$$\begin{aligned} \dot{\mathbf{V}} = &\frac{1}{2}\boldsymbol{e}^T(\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T\mathbf{P})\boldsymbol{e} \\ &+ \boldsymbol{e}^T\mathbf{P}\mathbf{B}\big(\widetilde{\boldsymbol{\Psi}}_\zeta\boldsymbol{\zeta} + \widetilde{\boldsymbol{\Psi}}_r\boldsymbol{r}(t) + \widetilde{\boldsymbol{\Psi}}_\phi\boldsymbol{\Phi}(\boldsymbol{e})\big) \\ &+ \mathbf{tr}\big(\widetilde{\boldsymbol{\Psi}}_\zeta^T\boldsymbol{\Theta}_\zeta\dot{\widetilde{\boldsymbol{\Psi}}}_\zeta + \widetilde{\boldsymbol{\Psi}}_r^T\boldsymbol{\Theta}_r\dot{\widetilde{\boldsymbol{\Psi}}}_r + \widetilde{\boldsymbol{\Psi}}_\phi^T\boldsymbol{\Theta}_\phi\dot{\widetilde{\boldsymbol{\Psi}}}_\phi\big). \end{aligned} \quad \text{(A5)}$$

From $\mathbf{B}\boldsymbol{\Psi}^*_r = \mathbf{B}_r$, we replace $\mathbf{B} = \mathbf{B}_r\boldsymbol{\Psi}^{*-1}_r$ (if $\mathbf{B}_r \succ 0$, then $\boldsymbol{\Psi}_r \succ 0$ and inevitable by construction [51]). Setting $\boldsymbol{\Theta}_\zeta = \boldsymbol{\Psi}^{*-T}_r\Lambda^{-1}_\zeta$, $\boldsymbol{\Theta}_r = \boldsymbol{\Psi}^{*-T}_r\Lambda^{-1}_r$, and $\boldsymbol{\Theta}_\phi = \boldsymbol{\Psi}^{*-T}_r\Lambda^{-1}_\phi$, we can rewrite Eq.A5 as:

$$\begin{aligned} \dot{\mathbf{V}} = &-\frac{1}{2}\boldsymbol{e}^T\mathbf{Q}\boldsymbol{e} + \boldsymbol{e}^T\mathbf{P}\mathbf{B}_r\boldsymbol{\Psi}^{*-1}_r\big(\widetilde{\boldsymbol{\Psi}}_\zeta\boldsymbol{\zeta} + \widetilde{\boldsymbol{\Psi}}_r\boldsymbol{r}(t) + \widetilde{\boldsymbol{\Psi}}_\phi\boldsymbol{\Phi}(\boldsymbol{e})\big) \\ &+ \mathbf{tr}\big(\widetilde{\boldsymbol{\Psi}}_\zeta^T\boldsymbol{\Psi}^{*-T}_r\Lambda^{-1}_\zeta\dot{\widetilde{\boldsymbol{\Psi}}}_\zeta + \widetilde{\boldsymbol{\Psi}}_r^T\boldsymbol{\Psi}^{*-T}_r\Lambda^{-1}_r\dot{\widetilde{\boldsymbol{\Psi}}}_r \\ &+ \widetilde{\boldsymbol{\Psi}}_\phi^T\boldsymbol{\Psi}^{*-T}_r\Lambda^{-1}_\phi\dot{\widetilde{\boldsymbol{\Psi}}}_\phi\big) \end{aligned} \quad \text{(A6)}$$

where matrices Λ_ζ, Λ_r, and Λ_ϕ are positive definite and tune the convergence rate of the adaptive gains. Replacing the adaptation law Eq. (9) in Eq. (A6), and taking advantage of the trace property for square matrices $\mathbf{tr}(\mathbf{AB}) = \mathbf{tr}(\mathbf{BA})$, all the right-hand side terms of Eq.A6, except the first one, vanish and the Lyapunov time derivative simplifies into $\dot{\mathbf{V}} = -\frac{1}{2}\boldsymbol{e}^T\mathbf{Q}\boldsymbol{e}$. Since $\mathbf{V} > 0$ and $\dot{\mathbf{V}} \leq 0$ the system in Eq. (A3) is Lyapunov stable, thus $\boldsymbol{e}$, $\widetilde{\boldsymbol{\Psi}}_\zeta$, $\widetilde{\boldsymbol{\Psi}}_r$, and $\widetilde{\boldsymbol{\Psi}}_\phi$ are bounded. As $\boldsymbol{r}(t)$ is bounded (see Section IV-A), the system given by Eq. (A3) and all variables in closed-loop remain bounded. By extension, $\ddot{\mathbf{V}} = -\boldsymbol{e}^T\mathbf{Q}\dot{\boldsymbol{e}}$ remain bounded at all time. Thus, from Barbalat's lemma [52] $\dot{\mathbf{V}} \to 0$ with $t \to \infty$ and $\boldsymbol{e} \to 0$.

IX. Function Approximation for Adaptive Control

In this paper, the adaptive control algorithm assumes that the system's non-linearities are encapsulated in a nonlinear function $\mathrm{F}(\cdot)$, which needs to be approximated online. For this, universal function approximators mostly in the form of RBF neural networks, are widely adopted [53]. $\mathrm{F}(\cdot)$ can be approximated by $\mathrm{F}(\cdot) = \boldsymbol{\Psi}^*_\phi\boldsymbol{\Phi}(\cdot) + \epsilon^*$ where ϵ^* denotes the network reconstruction error. With $\psi_i \in \mathbb{R}^n$ being the NN weight for the ith node, we can construct the weight matrix $\boldsymbol{\Psi}^*_\phi = [\psi_1 \quad \psi_2 \quad ... \quad \psi_p]$. Also $\boldsymbol{\Phi}(\cdot) = [\phi_1(\cdot) \quad \phi_2(\cdot) \quad ... \quad \phi_p(\cdot)]^T$ is a vector of radial basis functions in which $\phi_i(\cdot)$ the ith node function is given by $\phi_i(\mathrm{x}) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{\|\mathrm{x}-\boldsymbol{\xi}_i\|^2}{2\sigma_i^2}\right)$, with $\boldsymbol{\xi}_i \in \mathbb{R}^n$ the center, and σ_i the corresponding kernel width for RBF kernel of the ith node. It is noteworthy that basis function $\phi_i(\cdot)$ is not restricted to RBF kernels, and one could select other types of kernels, such as Cosine or Polynomial kernels. We adopt this methodology to estimate unmodeled non-linearities in multi-body systems, described in Section IV-A.