






# On-Line Learning for Planning and Control of Underactuated Robots With Uncertain Dynamics

Giulio Turrisi , Marco Capotondi , Claudio Gaz, Valerio Modugno , Giuseppe Oriolo , *Fellow, IEEE*,  
and Alessandro De Luca , *Fellow, IEEE*

**Abstract**—We present an iterative approach for planning and controlling motions of underactuated robots with uncertain dynamics. At its core, there is a learning process which estimates the perturbations induced by the model uncertainty on the active and passive degrees of freedom. The generic iteration of the algorithm makes use of the learned data in both the planning phase, which is based on optimization, and the control phase, where partial feedback linearization of the active dofs is performed on the model updated on-line. The performance of the proposed approach is shown by comparative simulations and experiments on a Pendubot executing various types of swing-up maneuvers. Very few iterations are typically needed to generate dynamically feasible trajectories and the tracking control that guarantees their accurate execution, even in the presence of large model uncertainties.

**Index Terms**—Underactuated robot, model learning for control, optimization and optimal control.

## I. INTRODUCTION

UNDERACTUATION in mechanical systems occurs when there are less independent actuation inputs than generalized coordinates. This situation may be due to the nature of the mechanism, to its prevailing design, or it may be the result of an intentional choice aimed at reducing weight, cost or energy consumption. Many advanced robotic platforms are indeed underactuated, including manipulators with some passive joints, most underwater and aerial vehicles, legged robots, and prehensile manipulation systems.

An adverse effect of underactuation is that generic state space trajectories become unfeasible, since the dynamics of the passive degrees of freedom represents a set of second-order differential constraints that must be satisfied throughout any motion [1]; in practice, this limits the directions of instantaneous accelerations that can be commanded to the system. As a consequence, trajectory planning in the absence of obstacles, which is a relatively trivial issue for fully actuated robots, becomes a challenging problem in the presence of underactuation.

Manuscript received July 9, 2021; accepted October 23, 2021. Date of publication November 10, 2021; date of current version November 23, 2021. This letter was recommended for publication by Associate Editor D. Fontanelli and Editor L. Pallottino upon evaluation of the reviewers' comments (*Corresponding author: Marco Capotondi*).

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, 00185 Roma, Italy (e-mail: turrisi@diag.uniroma1.it; capotondi@diag.uniroma1.it; gaz@diag.uniroma1.it; modugno@diag.uniroma1.it; oriolo@diag.uniroma1.it; deluca@diag.uniroma1.it).

This letter has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2021.3126899>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3126899

Motion control is also made more difficult by underactuation. One way to appreciate this is to consider that the dynamic models of fully actuated robots can always be made exactly linear and decoupled by using static feedback linearization [2], provided that an accurate model of the robot dynamics is available. In underactuated robots this cannot be achieved, and one has to deal directly with — actually, make use of — nonlinear, coupled dynamic effects.

In the literature, several model-based techniques have been proposed for planning and stabilizing motions of specific underactuated robots, with notable emphasis on manipulators with passive joints [3]. In particular, the problem of state transfer between equilibria has been addressed mainly on two benchmark platforms, i.e., the Pendubot and the Acrobot; these are both 2R robots moving in the vertical plane with a single actuated joint (respectively, the first and the second). A classical approach is to use collocated or non-collocated Partial Feedback Linearization (PFL) in combination with energy-based controllers [4], [5]. Swing-up maneuvers of these robots have been achieved using passivity-based approaches [6], [7], orbit stabilization [8], impulse-momentum techniques [9], and sequential action control [10]. Typically, the maneuver includes a final balancing phase realized through a Linear Quadratic Regulator (LQR) designed around the target equilibrium.

Although effective, the above approaches have two main limitations from the viewpoint of this paper. First, the design techniques used for trajectory planning and control are invariably specific (or had to be specialized) for the considered robot, and sometimes also for the particular maneuver. Second, and even more important, all of them require an accurate knowledge of the robot dynamic model for successful performance. Exceptions are [11], [12], which propose robust control of the Pendubot via adaptive and fuzzy sliding modes respectively; however, these methods are tailored to the platform and do not tolerate large model uncertainties in practice.

To avoid the need for an accurate dynamic model, modern learning techniques have been applied for deriving feedforward and feedback control of robots [13]. In [14], a semi-parametric regression is used to reconstruct the inverse dynamics of a manipulator. In [15], [16], [17] and [18], Reinforcement Learning (RL) procedures are proposed to generate robot control policies in a data-efficient way. However, this class of algorithms is not able in general to ensure satisfaction of hard constraints in a specific robot task. Along the same lines, the authors in [19] propose a meta-learning approach for a domain adaptation problem; in a

new scenario, only the regressor weights are updated while the set of basis functions is kept the same, ultimately limiting the effectiveness of the model correction. An optimization-based iterative learning approach is used in [20], [21], where experience from previous robot trials is used to build incrementally the feedforward command needed to follow a desired output trajectory. In [22], we have presented a learning-based approach for trajectory tracking which relies on the existence of a nominal feedback linearizing control law for the robotic system. Similar techniques are proposed in [23], [24]. These works, however, assume full actuation capability or at least feedback linearizability via dynamic feedback [25].

Other works that are more closely related to our approach have been published recently. In [26], a method for the swing-up of an Acrobot has been proposed which avoids the need for a model by using deep RL, requiring however a huge number of experiments for training. A robust control scheme for trajectory tracking under repetitive disturbances has been presented in [27] for a 3R planar manipulator with two actuators and one passive joint. The control design is tailored to this specific system, and cannot be easily extended to generic underactuated robots. In [28], a learning scheme is proposed to realize trajectory tracking of underactuated balance robots (e.g., a Furuta pendulum); because of the simpler balancing task, the reference trajectory of the active joints is not replanned and stabilization in the large is never addressed.

In this paper, we build upon our learning method for fully actuated robots [22] to devise an iterative approach for planning and controlling transfers between (stable or unstable) equilibria of underactuated robots in the presence of large dynamic uncertainties. The basic idea is to alternate off-line optimization-based planning and on-line PFL control, using regression to learn model corrections for the active and passive degrees of freedom. As a result, dynamically feasible state reference trajectories are learned and convergence to zero trajectory tracking error is obtained over the iterations. The main benefits of the proposed approach are:

- it applies to any underactuated robot;
- it applies to any state transfer maneuver;
- convergence is reached even in the presence of large uncertainties on the robot dynamics, requiring very few iterations in the considered case studies;
- more accuracy in the nominal dynamic model leads to even faster convergence;
- additional constraints (on state, on input, obstacle avoidance, etc.) can be explicitly taken into account in the optimization problem of the planning phase.

As an application, we provide an extensive evaluation of the performance of our approach on a Pendubot which must execute various swing-up maneuvers and state transfers between unstable equilibria (see Fig. 1).

The paper is organized as follows. Section II introduces the dynamic model of underactuated robots, highlighting how model uncertainties affect the active and passive subsystems. The proposed iterative approach is presented in Section III, discussing both the planning and the control phases and describing the data collection procedures and the regressors adopted for

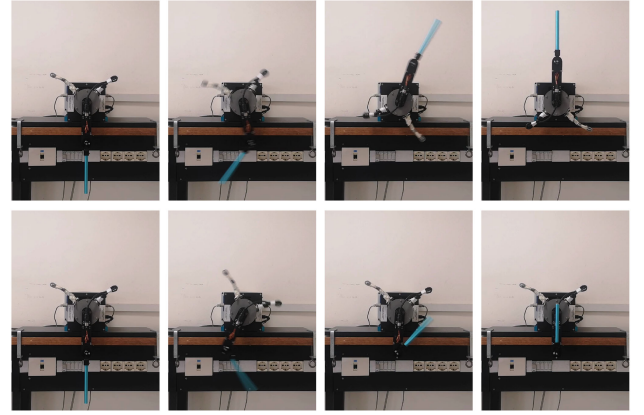


Fig. 1. The Pendubot performing two different swing-up maneuvers using the proposed method: the targets are the up-up equilibrium (first row) and the down-up equilibrium (second row). See the accompanying video.

learning. In Section IV, we report on the application to the Pendubot, showing comparative simulation and experimental results. Finally, some general conclusions about the approach are drawn in Section V.

## II. PROBLEM FORMULATION

For a robot with  $n$  degrees of freedom (dof) and  $m < n$  actuators, the dynamics can be expressed [3] as

$$M_{aa}(q)\ddot{q}_a + M_{ap}(q)\ddot{q}_p + n_a(q, \dot{q}) = \tau \quad (1)$$

$$M_{pa}(q)\ddot{q}_a + M_{pp}(q)\ddot{q}_p + n_p(q, \dot{q}) = 0, \quad (2)$$

where  $q = (q_a, q_p)$  is the  $n$ -dimensional configuration vector, with  $q_a, q_p$  representing respectively the  $m$  active and the  $n - m$  passive generalized coordinates. The inertia matrix  $M$  and the vector  $n$  of the remaining nonlinear terms are partitioned accordingly. The  $m$  generalized forces  $\tau$  only perform work on the  $q_a$  coordinates. We do not assume any structural control property (e.g., feedback linearizability or flatness) for system (1-2), nor any particular degree of underactuation.

In the presence of model perturbations (incorrect parameters and/or unmodeled dynamics), we can write

$$M = \hat{M} + \Delta M \quad n = \hat{n} + \Delta n. \quad (3)$$

Only the nominal terms  $\hat{M}$  and  $\hat{n}$  are known and available for control design.

Let  $x = (q, \dot{q})$  be the robot state. Given a start and a goal equilibrium points, respectively denoted by  $x_s = (q_s, 0)$  and  $x_g = (q_g, 0)$ , we want to plan and execute in a fixed time  $T$  a transfer motion from the start to the goal, while satisfying constraints on state and/or inputs, collectively expressed in the form  $h(q, \tau) \leq 0$ . This *transfer between equilibria* problem is particularly challenging for robots that are not fully actuated because not all trajectories between two equilibria are feasible.

For the following developments, it is convenient to perform a preliminary nonlinear feedback aimed at exactly linearizing the *nominal* active dynamics. This collocated PFL controller is

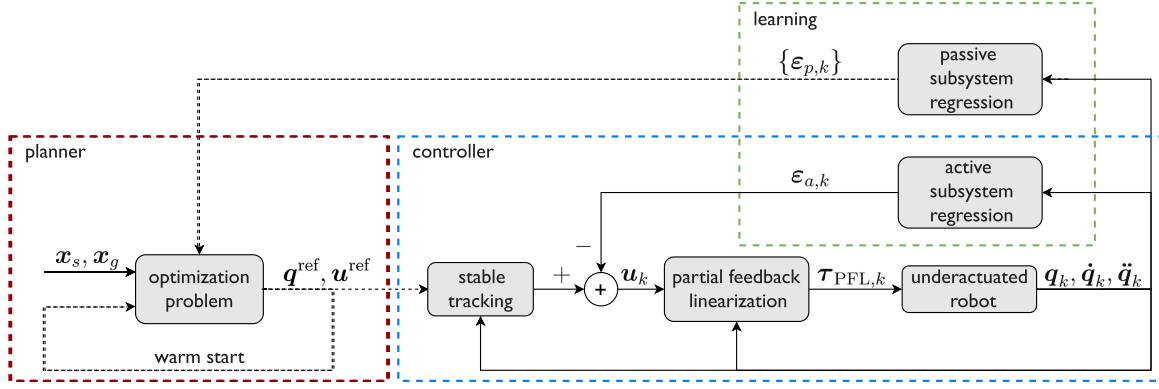


Fig. 2. Block diagram of the generic iteration of the proposed algorithm. Solid signal lines represent data that are used at each time step, whereas dashed lines are data transferred at the end of the iteration.

always well defined and takes the form

$$\tau_{\text{PFL}} = \left( \hat{M}_{aa} - \hat{M}_{ap} \hat{M}_{pp}^{-1} \hat{M}_{pa} \right) u + \hat{n}_a - \hat{M}_{ap} \hat{M}_{pp}^{-1} \hat{n}_p, \quad (4)$$

where  $u \in \mathbb{R}^m$  is the new input, i.e., the acceleration of the active dofs.

Using (4) and (3) in (1–2), we obtain the perturbed closed-loop dynamics

$$\ddot{q}_a = u + \delta_a(q, \dot{q}, u) \quad (5)$$

$$\ddot{q}_p = -\hat{M}_{pp}^{-1} \left( \hat{n}_p + \hat{M}_{pa} \ddot{q}_a \right) + \delta_p(q, \dot{q}, \ddot{q}_a), \quad (6)$$

where  $\delta_a$  and  $\delta_p$  represent the cumulative effect of perturbations on the active and passive subsystems, respectively.

### III. THE PROPOSED ITERATIVE APPROACH

The presence of model perturbations affects the considered planning and control problem at two levels. First, planning based on the nominal model would produce trajectories that may not be feasible, and in any case do not land at the goal equilibrium. Second, even when the reference trajectory is feasible, effective tracking is not achieved if the controller is designed on the nominal model.

In this section, we describe an iterative scheme for concurrent planning and control. At its core there is a *learning process* (Sects. III-C–III-E) which continuously updates two regressors  $\varepsilon_a$  and  $\varepsilon_p$ , respectively estimates of the perturbations  $\delta_a$  and  $\delta_p$  in (5–6). Both regressors are reconstructed from position measurements during robot motion.

Each iteration consists of an off-line *planning* phase and an on-line *control* phase. In the planning phase (Sect. III-A), the nominal model is corrected by taking into account  $\varepsilon_p$ ; an optimization problem is then solved to compute a reference trajectory  $q^{\text{ref}}(t)$  leading this model to  $x_g$  at time  $T$ . In the control phase (Sect. III-B), the robot tracks  $q^{\text{ref}}(t)$  under the action of a PFL control law given by (4), in which the corrective term  $\varepsilon_a$  is added to the commanded acceleration  $u$ . During the motion, new data points are collected and used in the learning process.

A block diagram of the generic iteration of the proposed approach is shown in Fig. 2.

#### A. Planning

In the planning phase, a reference trajectory is computed by solving a numerical optimal control problem for the underactuated robot. In particular, a prediction model is obtained by setting  $\delta_a = 0$  and  $\delta_p = \varepsilon_p$  in eqs. (5–6):

$$\ddot{q}_a = u \quad (7)$$

$$\ddot{q}_p = -\hat{M}_{pp}^{-1} \left( \hat{n}_p + \hat{M}_{pa} u \right) + \varepsilon_p(q, \dot{q}, u). \quad (8)$$

In other words, we are assuming in (7) that partial feedback linearization has been achieved in spite of model perturbations. The rationale is that the control law will try to cancel  $\delta_a$  as much as possible using a correction term equal to its current estimate  $\varepsilon_a$  (see Sect. III-B). Moreover, the available estimate  $\varepsilon_p$  of the perturbation on the passive subsystem has been used in (8). Upon convergence of the overall scheme, eq. (7) will become exact, and  $\varepsilon_p$  in (8) will eventually be equal to  $\delta_p$ .

In principle, we could have also included  $\varepsilon_a$  in the right-hand side of (7). The learning transient would be similar and, upon convergence, the obtained system behavior would be the same. However, the separate use of one regressor ( $\varepsilon_p$ ) in the planning phase and of the other ( $\varepsilon_a$ ) in the control phase proves to be computationally more efficient.

We consider a discrete-time setting in which the input  $u$  is piecewise-constant over  $N$  sampling intervals of duration  $T_s = T/N$ . Denoting by  $f(\cdot)$  a discretization of the state-space representation corresponding to (7–8), with the robot state  $x_i = x(t_i)$  and the starting and goal equilibrium points  $x_s$  and  $x_g$  defined in Section II, the optimization problem (OP) is written as

$$\min_{u_0, \dots, u_{N-1}} \sum_{i=0}^{N-1} J(x_i, u_i) + J_N(x_N)$$

subject to

$$x_{i+1} - f(x_i, u_i) = 0, \quad i = 0 \dots, N-1,$$

$$g(x_i) \leq 0, \quad i = 1, \dots, N,$$

$$h(u_i) \leq 0, \quad i = 1, \dots, N-1,$$

with  $x_0 = x_s$ . The objective function is the sum of a stage cost  $J$  and a terminal cost  $J_N$ , both penalizing the state error



with respect to the goal  $\mathbf{x}_g$  and the control effort, while  $\mathbf{g}$  and  $\mathbf{h}$  represent state and input constraints, respectively. The cost terms take the form

$$J(\mathbf{x}_i, \mathbf{u}_i) = \|\mathbf{x}_g - \mathbf{x}_i\|_{\mathbf{Q}}^2 + \|\mathbf{u}_i\|_{\mathbf{R}}^2,$$

$$J_N(\mathbf{x}_N) = \|\mathbf{x}_g - \mathbf{x}_N\|_{\mathbf{Q}_N}^2.$$

where  $\mathbf{Q}$ ,  $\mathbf{Q}_N$  and  $\mathbf{R}$  are positive-definite, symmetric matrices of weights. The solution of OP is a reference trajectory with the associated nominal input, represented by discrete sequences  $\mathbf{q}^{\text{ref}} = \{\mathbf{q}_1^{\text{ref}}, \dots, \mathbf{q}_N^{\text{ref}}\}$  and  $\mathbf{u}^{\text{ref}} = \{\mathbf{u}_0^{\text{ref}}, \dots, \mathbf{u}_{N-1}^{\text{ref}}\}$  respectively. The reference velocities  $\dot{\mathbf{q}}^{\text{ref}} = \{\dot{\mathbf{q}}_1^{\text{ref}}, \dots, \dot{\mathbf{q}}_N^{\text{ref}}\}$  are also available.

To speed up convergence to a solution, one typically uses the reference trajectory of the previous iteration as a warm start when solving the current OP.

### B. Control

In the control phase, the robot moves under the action of a digital<sup>1</sup> control law aimed at driving  $\mathbf{q}$  along the current reference trajectory  $\mathbf{q}^{\text{ref}}$ . To achieve stable tracking of  $\mathbf{q}_a^{\text{ref}}$ , the commanded acceleration  $\mathbf{u}_k$  in  $[t_k, t_{k+1})$  is chosen as

$$\mathbf{u}_k = \mathbf{u}_k^{\text{ref}} + \mathbf{K}_P(\mathbf{q}_{a,k}^{\text{ref}} - \mathbf{q}_{a,k}) + \mathbf{K}_D(\dot{\mathbf{q}}_{a,k}^{\text{ref}} - \dot{\mathbf{q}}_{a,k}) - \varepsilon_{a,k}, \quad (9)$$

with  $\mathbf{K}_P, \mathbf{K}_D > 0$ . Here, the nominal input produced by the planner is used as feedforward term, and the current regressor  $\varepsilon_{a,k}$  has been added to cancel at best the perturbation  $\delta_a$  affecting the active subsystem (5). Note that as soon as  $\mathbf{q}_a$  will be able to follow exactly  $\mathbf{q}_a^{\text{ref}}$ , the passive variables  $\mathbf{q}_p$  will evolve as planned in the previous phase.

Next, we use (4) to compute the generalized force as

$$\boldsymbol{\tau}_{\text{PFL},k} = \hat{\mathbf{B}}_k \mathbf{u}_k + \hat{\boldsymbol{\eta}}_k, \quad (10)$$

where

$$\hat{\mathbf{B}}_k = \hat{\mathbf{M}}_{aa}(\mathbf{q}_k) - \hat{\mathbf{M}}_{ap}(\mathbf{q}_k) \hat{\mathbf{M}}_{pp}^{-1}(\mathbf{q}_k) \hat{\mathbf{M}}_{pa}(\mathbf{q}_k)$$

and

$$\hat{\boldsymbol{\eta}}_k = \hat{\mathbf{n}}_a(\mathbf{q}_k, \dot{\mathbf{q}}_k) - \hat{\mathbf{M}}_{ap}(\mathbf{q}_k) \hat{\mathbf{M}}_{pp}^{-1}(\mathbf{q}_k) \hat{\mathbf{n}}_p(\mathbf{q}_k, \dot{\mathbf{q}}_k).$$

### C. Regressors for Estimating Perturbations

In this work, we employ Gaussian Processes (GP) regressors [29] for reconstructing  $\varepsilon_a$  and  $\varepsilon_p$ , given the good performance that this technique displays in the on-line learning context. However, it is important to notice that other techniques such as Neural Networks, Generalized Linear Regression or Support Vector Machine could have been adopted without any modifications on the structure of the framework.

Considering  $\mathcal{D} = \{(\mathbf{X}_i, Y_i = \phi(\mathbf{X}_i) + \omega_i) | 1 \leq i \leq n_d\}$  representing a set of input-output noisy observations where  $\omega \sim \mathcal{N}(\mathbf{0}, \Sigma_\omega)$  and  $\phi$  indicates a unidimensional function to reconstruct. With  $\mathbf{K}(\mathbf{X}, \mathbf{X})$  we define the covariance matrix whose elements are computed through the inner product

<sup>1</sup>For simplicity, it is assumed that the control sampling interval is the same  $T_s$  used for planning.

operation  $k(\cdot, \cdot)$  and  $\mathbf{k}^T(\cdot)$  is a row vector obtained by computing  $k$  between a new point and every element of  $\mathbf{X}$ . Let us denote the value of the function to reconstruct at an arbitrary point as  $Y_{n_d+1} = \phi(\mathbf{X}_{n_d+1})$ . Exploiting the properties of Gaussian processes, the ensemble of observation in the dataset defined as  $\mathbf{Y}_{1:n_d}$  and  $Y_{n_d+1}$  are jointly Gaussian:

$$\begin{bmatrix} \mathbf{Y}_{1:n_d} \\ Y_{n_d+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{X}_{n_d+1}, \mathbf{X}_{n_d+1}) \end{bmatrix}\right)$$

It is possible to define the predictive distribution that approximates the perturbation  $\delta(\hat{\mathbf{X}})$  for a generic query point  $\hat{\mathbf{X}}$  as

$$\varepsilon(\hat{\mathbf{X}}|\mathcal{D}) \sim \mathcal{N}(\mu(\hat{\mathbf{X}}), \sigma^2(\hat{\mathbf{X}}))$$

where

$$\mu(\hat{\mathbf{X}}) = \mathbf{k}^T(\hat{\mathbf{X}})(\mathbf{K} + \Sigma_\omega)^{-1} \mathbf{Y}$$

$$\sigma^2(\hat{\mathbf{X}}) = k(\hat{\mathbf{X}}, \hat{\mathbf{X}}) - \mathbf{k}^T(\hat{\mathbf{X}})(\mathbf{K} + \Sigma_\omega)^{-1} \mathbf{k}(\hat{\mathbf{X}}).$$

In this context,  $\mu$  represents the regressor prediction while  $\sigma^2(\cdot)$  describes the epistemic error associated to  $\mu(\cdot)$ . In this work, for reconstructing a multidimensional observation, we stack a set of unidimensional GPs.

Since no assumption is made on the structure of the unmodeled dynamics, we employ as  $k(\cdot, \cdot)$  a squared exponential kernel defined as

$$k(\mathbf{X}_i, \mathbf{X}_j) = a^2 \exp\left(-\frac{\|\mathbf{X}_i - \mathbf{X}_j\|^2}{2l^2}\right),$$

where the length-scale  $l$  and the amplitude  $a$  represent the hyperparameters of the GP regressor.

### D. Dataset Collection Procedure for the Active Dofs

We now show how to collect the data points that are used to learn the estimate  $\varepsilon_{a,k}$  to be used at each instant in the commanded acceleration (9).

The idea is to perform a regression based on the difference between the commanded acceleration and the actual acceleration for the actuated dofs. In fact, from eq. (5) we may write

$$\delta_{a,k} = \ddot{\mathbf{q}}_{a,k} - \mathbf{u}_k. \quad (11)$$

In view of eq. (11), a new data point is generated at the  $k$ -th control step as

$$\mathbf{X}_{a,k} = (\mathbf{q}_k, \dot{\mathbf{q}}_k, \mathbf{u}_k) \quad \mathbf{Y}_{a,k} = \ddot{\mathbf{q}}_{a,k} - \mathbf{u}_k.$$

with the acceleration  $\ddot{\mathbf{q}}_{a,k}$  to be reconstructed numerically. We note that the actual acceleration is functionally dependent through (5) on the robot state  $(\mathbf{q}_k, \dot{\mathbf{q}}_k)$  and on the commanded acceleration  $\mathbf{u}_k$ , i.e., on the input  $\mathbf{X}_{a,k}$  of the regression scheme.

Every time a new data point is available, it is immediately used to update the regressor  $\varepsilon_a$ . However, the hyperparameters of the kernel function are only updated at the end of each iteration.

A potential issue of GP regression is that the computational complexity of the prediction is  $\mathcal{O}(n_d^3)$ , with  $n_d$  the size of the dataset. To keep the computation of  $\varepsilon_a$  fast enough for real-time control, an approximate regression is performed using a reduced set of only  $d$  datapoints, chosen on the basis of

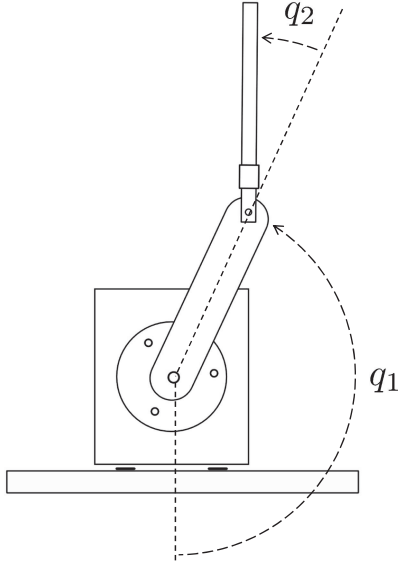


Fig. 3. The Pendubot and its generalized coordinates.

the information gain criterion [30]. This leads to a reduced complexity  $\mathcal{O}(d^2 \cdot n_d)$ .

#### E. Dataset Collection Procedure for the Passive Dofs

To learn an estimate  $\varepsilon_p$  of the model perturbation  $\delta_p$ , we compare the commanded and the actual acceleration for the passive dofs.

In fact, from eq. (6) we have

$$\delta_{p,k} = \ddot{\mathbf{q}}_{p,k} + \hat{\mathbf{M}}_{pp,k}^{-1}(\hat{\mathbf{n}}_{p,k} + \hat{\mathbf{M}}_{pa,k} \ddot{\mathbf{q}}_{a,k}). \quad (12)$$

Given numerical approximations of the actual accelerations  $\ddot{\mathbf{q}}_{a,k}$  and  $\ddot{\mathbf{q}}_{p,k}$ , a new data point is generated at the  $k$ -th step as

$$\begin{aligned} \mathbf{X}_{p,k} &= (\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_{a,k}) \\ \mathbf{Y}_{p,k} &= \ddot{\mathbf{q}}_{p,k} + \hat{\mathbf{M}}_{pp,k}^{-1}(\hat{\mathbf{n}}_{p,k} + \hat{\mathbf{M}}_{pa,k} \ddot{\mathbf{q}}_{a,k}). \end{aligned}$$

Differently from  $\varepsilon_a$ , all the data points computed during the iteration are used to update the passive subsystem regressor  $\varepsilon_p$  at the end of each trial; in fact, since the planning phase is performed off-line, the complexity associated to exact regression does not represent a problem here. As before, the hyperparameters of the kernel function are also updated at the end of the iteration.

### IV. APPLICATION TO THE PENDUBOT

The proposed approach has been validated through simulations and experiments on the Pendubot, a two-link arm moving in the vertical plane with an active joint at the shoulder and a passive joint at the elbow ( $\mathbf{q}_a = \mathbf{q}_1$  and  $\mathbf{q}_p = \mathbf{q}_2$ ). See Fig. 3 for the definition of the generalized coordinates and [31] for the dynamic model of the Pendubot, complete with nominal parameter values for our prototype.

In the following, we will address the problem of executing various transfer motions between equilibria in the presence of

severe uncertainty on the dynamic model. The proposed iterative method is used to steer the Pendubot to the basin of attraction of an LQR balancing controller designed around the goal state. The latter is obviously needed to stabilize the robot after the planning horizon  $T$ .

The discretized state-space model used in the planning phase has been obtained by Euler method. The sampling interval is set to  $T_s = 10$  ms. The cost function  $J$  in OP includes two quadratic terms that penalize the state error with respect to the goal  $\mathbf{x}_g = (q_{1,g}, q_{2,g}, 0, 0)$  as well as the control effort. Optimization is performed in MATLAB using the `fmincon` function, which implements a Sequential Quadratic Programming method. The joint velocities are bounded as  $|\dot{q}_1| \leq 8$  rad/s and  $|\dot{q}_2| \leq 15$  rad/s. Finally, terminal constraints are included to guarantee convergence at time  $T$  to the following basin of attraction of the balancing controller

$$|q_{j,N} - q_{j,g}| \leq 0.2, \quad |\dot{q}_{j,N}| \leq 0.5, \quad j = 1, 2,$$

which was found to be adequate for all goal states.

In the control phase, the PD gains in (9) are chosen as  $K_P = 50$  and  $K_D = 20$ , while the sampling interval is again 10 ms.

While all data points (with  $n_d$  equal to  $N$  times the number of iterations so far) are considered for updating  $\varepsilon_p$ , the maximum number of data points used for computing  $\varepsilon_a$  in real time is  $d = 180$ .

Refer to the accompanying video for clips from all simulations and experiments shown in the following.

#### A. Simulation Results

Two scenarios of transfer between equilibrium states will be presented. To show that the proposed method can achieve robust performance in the presence of severe model perturbations, we perturbed for control design the nominal values in [31], increasing by 30% the link masses  $m_1$  and  $m_2$  and reducing by the same percentage the distances  $a_1$  and  $a_2$  of the centers of mass of the two links from their respective joints. The link barycentric inertias  $I_1$  and  $I_2$  were changed accordingly.

In the first scenario, the start configuration is  $\mathbf{q}_s = (0, 0)$  while the goal is the *up-up* configuration  $\mathbf{q}_g = \mathbf{q}^{u-u} = (\pi, 0)$ , corresponding to a transfer from a stable to an unstable equilibrium (*swing-up*). The planning horizon is chosen as  $T = 1.6$  s ( $N = 160$ ).

To highlight the necessity of learning in both the planning and control phases, we have preliminarily considered two complementary situations where learning is not used. Figure 4, left, refers to the first situation, in which we use the nominal model for planning and the true model for control. The result shows that planning the motion of an underactuated robot based on an inaccurate model produces dynamically unfeasible trajectories, that cannot be tracked in spite of the ideality of the controller. Vice versa, in Fig. 4, right, the true model is used for planning and the nominal for control. As expected, the inaccuracy of the controller prevents the completion of the swing-up maneuver.

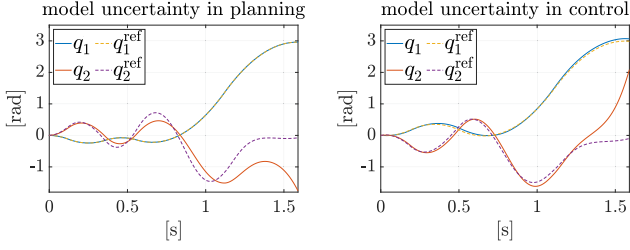


Fig. 4. Simulation scenario 1 (swing-up to  $q^{u-u}$ ): results without learning. Left: Using the nominal model for planning and the true model for control. Right: Vice versa.

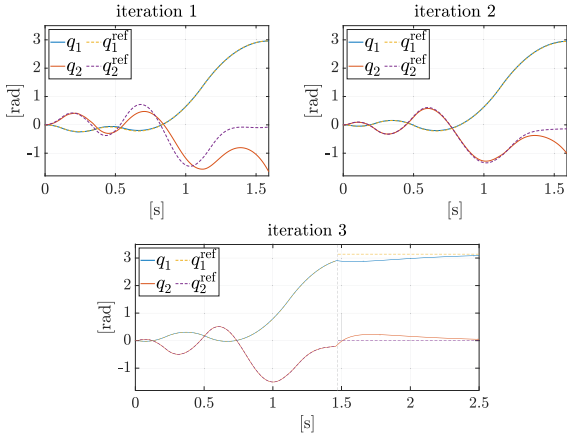


Fig. 5. Simulation scenario 1 (swing-up to  $q^{u-u}$ ): results with the proposed approach. Just before the end of the third iteration, the state has converged to the basin of attraction of the balancing controller, which is then activated (as indicated by the vertical dashed line).

Next, we tested the proposed approach on the same scenario, obtaining the results in Fig. 5. After three iterations, the Pendubot is able to track with sufficient accuracy the planned trajectory, ultimately entering the basin of attraction of the balancing controller to complete the swing-up maneuver. This shows that, in spite of the very large model uncertainty, the learning component of our method is able to reconstruct the correct model in few iterations. Further iterations of the planning-control sequence do not change significantly the resulting motion.

To put our result in perspective, we have applied to this scenario also the passivity-based swing-up method proposed in [7], using the same balancing controller in the final phase. As shown in Fig. 6, the method works perfectly if the robot model is exactly known, but is unable to complete the maneuver in the presence of the considered model uncertainty. In particular, while the first joint still converges to its target, the passive joint drifts away very quickly.

In the second scenario, the start is  $q_s = (\pi/4, 3\pi/4)$  while the goal is  $q_g = (5\pi/4, -\pi/4)$ ; this amounts to a transfer between two unstable equilibria. The planning horizon is chosen as  $T = 0.7$  s ( $N = 70$ ). The results are shown in Fig. 7. Two iterations are now sufficient to reach the basin of attraction of the balancing controller, thus completing the maneuver correctly. Indeed, a closer look at the joint motion (see also the accompanying video) reveals that in both iterations the transfer is performed with the

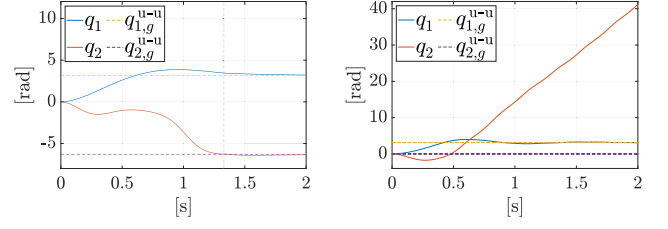


Fig. 6. Simulation scenario 1 (swing-up to  $q^{u-u}$ ): results with the method in [7]. Left: assuming exact model knowledge the state enters the basin of attraction of the LQR controller at  $t = 1.3$  s circa. Right: with the same model uncertainty of Fig. 5, convergence is not achieved.

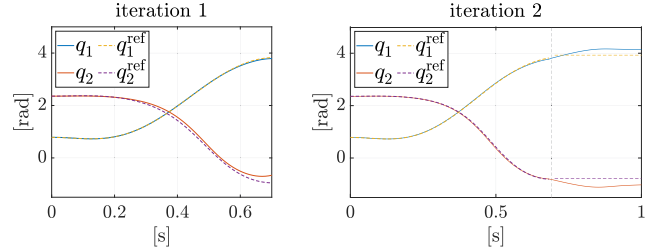


Fig. 7. Simulation scenario 2 (transfer between unstable equilibria): results with the proposed approach. Two iterations are needed to achieve convergence.

second link approximately vertical, a situation which inherently reduces the effect of the uncertain dynamic parameters, leading to a faster convergence.

We have performed further simulations on a 3R Pendubot with *two* passive joints that must execute a swing-up maneuver to  $q^{u-u}$  under perturbed conditions similar to scenario 1. Once again, convergence was achieved in 3 iterations, a result suggesting that our method performs effectively also for higher degrees of underactuation. See the accompanying video for an animated clip.

## B. Experimental Results

The proposed method has also been tested experimentally on our Pendubot prototype, using again the nominal model in [31] for planning and control design. Joint velocities and accelerations are obtained in real time via filtered numerical differentiation of encoder measurements. To further remove the noise affecting the learning dataset for the passive dofs, we used a non-causal Savitzky-Golay filter to compute  $\ddot{q}_2$ .

The first experiment replicates the swing-up scenario to  $q^{u-u}$  of Section IV-A, using the same planning horizon of  $T = 1.6$  s ( $N = 160$ ). The results are shown in Fig. 8. Only two iterations are required for our method to enter the basin of attraction of the LQR controller.

The combination of on-line learning of the active joint dynamics together with the off-line re-planning of both joint trajectories, driven by the regressor built for the passive joint dynamics, allows a successful execution of the swing-up maneuver. When comparing the tracking errors between the single run without learning (Fig. 8, first column) and the first iteration of the method (Fig. 8, second column), no major changes are observed for

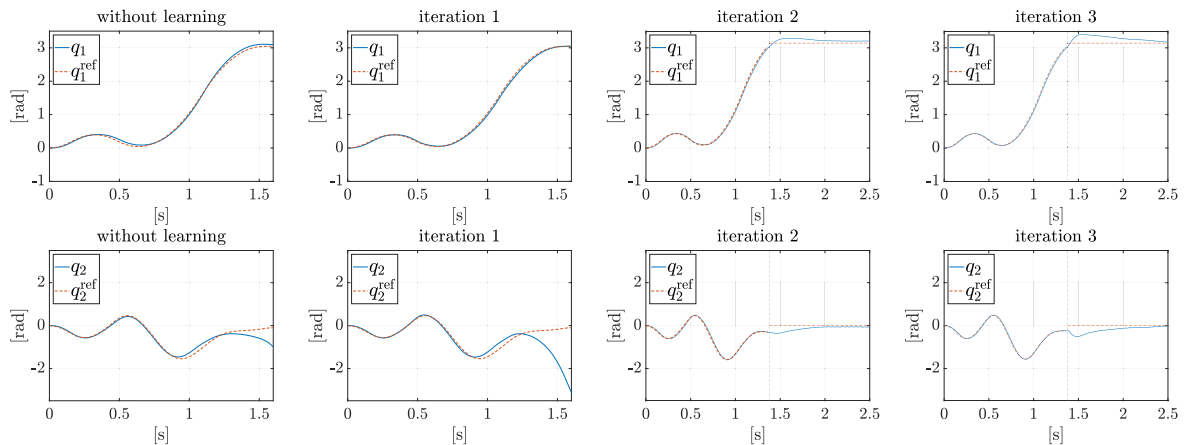


Fig. 8. Experimental scenario 1 (swing-up to  $\mathbf{q}^{u-u}$ ): results with the proposed approach. For comparison, the first column shows the results without learning, i.e., when partial feedback linearization and stable tracking for the first joint are computed on the nominal model.

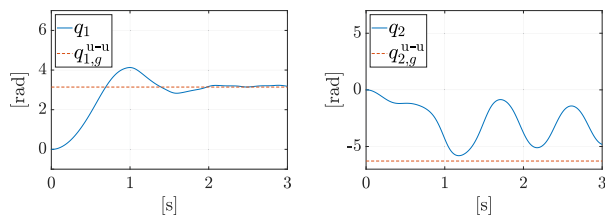


Fig. 9. Experimental scenario 1 (swing-up to  $\mathbf{q}^{u-u}$ ): results with the method in [7].

TABLE I  
TRACKING RMSE [RAD] IN THE EXPERIMENTS

	scenario 1		scenario 2	
	$q_1$	$q_2$	$q_1$	$q_2$
without learning	0.045	0.191	0.056	0.320
iteration 1	0.035	0.623	0.022	0.470
iteration 2	0.037	0.038	0.023	0.034
iteration 3	0.036	0.036	—	—

the active joint  $q_1$ , whereas the passive joint  $q_2$  behaves quite differently toward the end of the motion. In both cases, the error diverges (the second link falls in two opposite directions) because the currently planned trajectory is still dynamically unfeasible for the Pendubot. Already at the second iteration (third column), the robot is correctly driven to the basin of attraction of the desired equilibrium (the LQR stabilizer is triggered at about  $t = 1.4$  s). Performing a third planning-control iteration shows no significant variation of the obtained reference trajectory and control accuracy (fourth column).

Table I offers further insight on the performance in both scenarios. In particular, it shows that the tracking accuracy for  $q_1$  does not change significantly over the iterations, while the evolution of  $q_2$  gets increasingly closer to the planned trajectory, as the latter approaches feasibility thanks to the model learning procedure.

For comparison, Fig. 9 shows the experimental results obtained in this scenario with the method of [7] under the

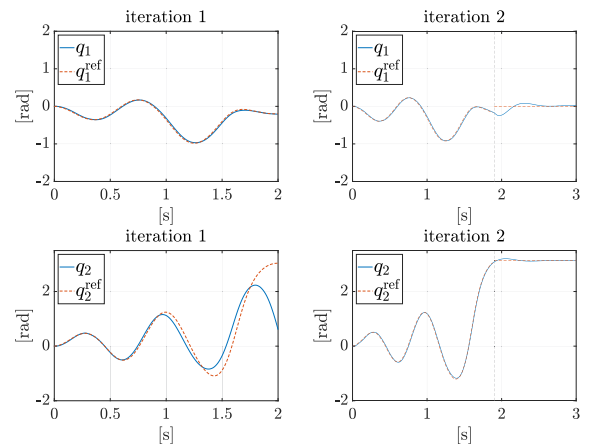


Fig. 10. Experimental scenario 2 (swing-up to  $\mathbf{q}^{d-u}$ ): results with the proposed approach. Just before the end of the second iteration the state converges to a region where the LQR controller can be successfully activated.

same nominal information on the robot dynamic model. While the active joint converges to its desired goal, the second joint oscillates (with a light damping due to friction) without ever entering the basin of attraction of the stabilizing controller. Therefore, we can claim that the learning procedure makes the proposed method able to withstand a level of model uncertainty which is not tolerated by purely model-based controllers.

The second experiment is again a swing-up scenario, but the goal is now the *down-up* configuration  $\mathbf{q}^{d-u} = (0, \pi)$ . The planning horizon has been set to  $T = 2$  s ( $N = 200$ ). As shown in Fig. 10, also in this case the learning procedure allows to complete the maneuver successfully after two iterations (see also the second column in Table I).

## V. CONCLUSION

We have proposed an iterative method for planning and controlling motions of underactuated robots in the presence of model uncertainty. The method hinges upon a learning process which estimates the induced perturbations on the dynamics of



the active and passive dofs. Each iteration includes an off-line planning phase and an on-line planning phase, which take advantage of the learned data to improve the feasibility of the planned trajectory and the accuracy of its tracking.

The proposed approach was validated by application to the Pendubot, a well-known underactuated platform consisting of a 2R planar robot with a passive elbow joint. In particular, numerical simulations of our iterative method starting with considerable errors in the nominal dynamic parameters ( $\pm 30\%$  of the true values) have shown that swing-up maneuvers and transfers between unstable equilibria can be executed successfully after very few iterations. This remarkable performance was confirmed in experimental tests on a real Pendubot.

In addition to applicability to general underactuated systems and independence from the specific maneuver, a further aspect of our method that deserves to be emphasized is that no torque measurement is required. In fact, only positions, velocities and accelerations must be available, so that implementation is possible using only encoders. Another interesting feature is the possibility to incorporate constraints on the robot states and/or inputs in the planning phase, as well as to handle (without any modification) also the presence of repetitive external disturbances.

Future work will consider the problem of guaranteeing hard constraints during the entire learning transient, by explicitly taking into account the covariance of the uncertainty during the planning phase. Moreover, we plan to test the method on different underactuated robots, namely quadrotor UAVs and humanoids.

## REFERENCES

- [1] G. Oriolo and Y. Nakamura, "Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators," in *Proc. 30th IEEE Conf. Decis. Control*, 1991, pp. 2398–2403.
- [2] A. Isidori, *Nonlinear Control Systems*, 3rd ed. New York: Springer, 1995.
- [3] A. De Luca, S. Iannitti, R. Mattone, and G. Oriolo, "Underactuated manipulators: Control properties and techniques," *Mach. Intell. Robotic Control*, vol. 4, no. 3, pp. 113–125, 2002.
- [4] M. Spong, "Partial feedback linearization of underactuated mechanical systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1994, pp. 314–321.
- [5] M. Spong, "Underactuated mechanical systems," in *Proc. Control Problems Robot. Automat.*, B. Siciliano and K. Valavanis, Eds., 1998, pp. 135–150.
- [6] I. Fantoni and R. Lozano, "The pendubot system," in *Proc. Non-Linear Control Underactuated Mech. Syst.*, 2002, pp. 53–72.
- [7] O. Kolesnichenko and A. Shiriaev, "Partial stabilization of underactuated euler-lagrange systems via a class of feedback transformations," *Syst. Control Lett.*, vol. 45, no. 2, pp. 121–132, 2002.
- [8] Y. Orlov, L. T. Aguilar, L. Aho, and A. Ortiz, "Swing up and balancing control of pendubot via model orbit stabilization: Algorithm synthesis and experimental verification," in *Proc. 45th IEEE Conf. Decis. Control*, 2006, pp. 6138–6143.
- [9] T. Albahkali, R. Mukherjee, and T. Das, "Swing-up control of the pendubot: An impulse-momentum approach," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 975–982, Aug. 2009.
- [10] A. R. Ansari and T. D. Murphey, "Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1196–1214, Oct. 2016.
- [11] D. Qian, J. Yi, and D. Zhao, "Hierarchical sliding mode control to swing up a pendubot," in *Proc. Amer. Control Conf.*, 2007, pp. 5254–5259.
- [12] S. Moussaoui, A. Boulkroune, and S. Vaidyanathan, "Fuzzy adaptive sliding-mode control scheme for uncertain underactuated systems," in *Proc. Adv. Appl. Nonlinear Control Syst.*, S. Vaidyanathan and C. Volos, Eds., 2016, pp. 351–367.
- [13] J. Peters, D. D. Lee, J. Kober, D. Nguyen-Tuong, J. A. Bagnell, and S. Schaal, "Robot learning," in *Proc. Springer Handbook Robot.*, 2nd ed, B. Siciliano and O. Khatib, Eds., 2016, pp. 357–394.
- [14] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 2677–2682.
- [15] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2016.
- [16] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 465–472.
- [17] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, "Black-box data-efficient policy search for robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 51–58.
- [18] M. Saveriano, Y. Yin, P. Falco, and D. Lee, "Data-efficient control policy search using residual dynamics learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 4709–4715.
- [19] E. Arcari, A. Carron, and M. N. Zeilinger, "Meta learning mpc using finite-dimensional gaussian process approximations," 2020. [Online]. Available: <https://arxiv.org/pdf/2008.05984v1.pdf>
- [20] A. Schoellig and R. D'Andrea, "Optimization-based iterative learning control for trajectory tracking," in *Proc. 10th Eur. Control Conf.*, 2009, pp. 1505–1510.
- [21] F. L. Mueller, A. Schoellig, and R. D'Andrea, "Iterative learning of feed-forward corrections for high-performance tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 3276–3281.
- [22] M. Capotondi, G. Turrisi, C. Gaz, V. Modugno, G. Oriolo, and A. De Luca, "An online learning procedure for feedback linearization control without torque measurements," in *Proc. Mach. Learn. Res. (3rd Conf. Robot Learning)*, 2020, vol. 100, pp. 1359–1368.
- [23] T. Westenbroek *et al.*, "Feedback linearization for uncertain systems via reinforcement learning," in *Proc. Int. Conf. Robot. Automat.*, 2020, pp. 1364–1371.
- [24] M. Greeff and A. Schoellig, "Exploiting differential flatness for robust learning-based tracking control using gaussian processes," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1121–1126, Oct. 2021.
- [25] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," in *Proc. ASME Int. Mech. Eng. Congr. Expo.*, 1995.
- [26] S. Gillen, M. Molnar, and K. Byl, "Combining deep reinforcement learning and local control for the acrobot swing-up and balance task," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 4129–4134.
- [27] P. Zhang, X. Lai, Y. Wang, and M. Wu, "Motion planning and adaptive neural sliding mode tracking control for positioning of uncertain planar underactuated manipulator," *Neurocomputing*, vol. 334, pp. 197–205, 2019.
- [28] K. Chen, J. Yi, and D. Song, "Gaussian processes model-based control of underactuated balance robots," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 4458–4464.
- [29] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
- [30] M. Seeger, C. Williams, and N. Lawrence, "Fast forward selection to speed up sparse gaussian process regression," in *Proc. 9th Int. Work. Artif. Intell. Statist.*, 2003, pp. 254–261.
- [31] G. Turrisi, B. Barros Carlos, M. Cefalo, V. Modugno, L. Lanari, and G. Oriolo, "Enforcing constraints over learned policies via nonlinear MPC: Application to the pendubot," *IFAC PapersOnLine*, vol. 53, no. 2, pp. 9502–9507, 2020.