Model-Plant Mismatch Compensation Using Reinforcement Learning

Ivan Koryakovskiy, Manuel Kudruss, Heike Vallery, Robert Babuška, and Wouter Caarls

Abstract—**Learning-based approaches are suitable for the control of systems with unknown dynamics. However, learning from scratch involves many trials with exploratory actions until a good control policy is discovered. Real robots usually cannot withstand the exploratory actions and suffer damage. This problem can be circumvented by combining learning with a model-based control. In this letter, we employ a nominal model-predictive controller that is impeded by the presence of an unknown model-plant mismatch. To compensate for the mismatch, we propose two approaches of combining reinforcement learning with the nominal controller. The first approach learns a compensatory control action that minimizes the same performance measure as is minimized by the nominal controller. The second approach learns a compensatory signal from a difference of a transition predicted by the internal model and an actual transition. We compare the approaches on a robot attached to the ground and performing a setpoint reaching task in simulations. We implement the better approach on the real robot and demonstrate successful learning results.**

Index Terms—**Learning and adaptive systems, humanoid robots.**

I. Introduction

MECHANICALLY and electronically, robotics have advanced to the point where cognitive abilities have become the main limiting factor. While robots can flawlessly execute a set of commands to achieve a task, these commands are mostly encoded or tuned by hand. Reinforcement learning (RL) allows to find an optimal sequence of commands without any prior assumption about the world. However, the application of pure learning to real systems is very limited due to intrinsically damaging exploratory policies. For example, when learning from scratch the robot Leo depicted in Fig. 1 can withstand only five minutes of operation due to large and rapidly changing motor torques and frequent falls [1].

Usually, the dynamics of physical systems are known, but various uncertainties do not allow achieving optimal performance with model-based control methods [2]. Whereas for the estimation of parametric uncertainties moving horizon estimation techniques [3] can often be employed, for structural uncertainties, such as backlash, Coulomb friction or wear and tear, this is not easily possible. Nevertheless, model-based methods can predict the evolution of the system for a short horizon, thus enabling the implementation of safety barriers to limit risky exploration in dangerous state space regions.

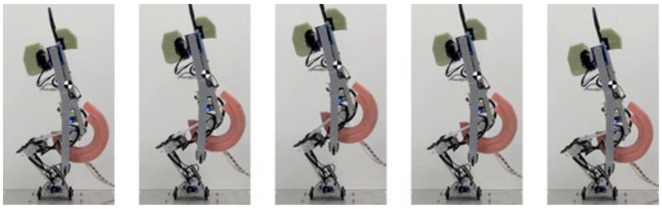

Fig. 1. Robot Leo performs up and down motions. Root point is shown by the circle with black and white sectors.

By safety, we mean the prevention of actions that cause damage to the system. For example, in a bipedal robot, safety is particularly related to the robot not falling. Falls result in impact forces applied to the limbs and gearboxes, and some robots cannot withstand even a single fall. In model-based control, it is natural to constrain the angles and velocities to remain within an admissible range, and to enforce static stability constraints such that the center of mass projection is some distance away from the support polygon border. These constraints help prevent falls, but a momentary violation of them does not necessarily result in the one. In RL, it is possible to consider angle and velocity constraints by means of negative rewards. However, to learn avoiding such constraints, they need to be violated multiple times in different robot configurations. Random exploration exacerbates the problem and can lead to a very large number of falls.

Therefore, we propose to combine RL and nonlinear model predictive control (NMPC) in one framework that allows RL to gather the required experience without damaging a many-degree-of-freedom system. The experience is used by RL to compensate the difference between the internal model of the system and the real one. Any model-based nominal controller is suitable, but the choice of NMPC is particularly motivated by the complexity of the robot.

We propose two different approaches shown in Fig. 2. Similarly to [4], the first approach learns a compensatory control action, but instead of a proportional-derivative (PD) controller, we use NMPC, which introduces an additional optimization problem. Since both NMPC and RL optimize similar

Manuscript received September 10, 2017; accepted January 12, 2018. Date of publication January 31, 2018; date of current version March 29, 2018. This letter was recommended for publication by Associate Editor B. Argall and Editor D. Lee upon evaluation of the reviewers' comments. This work was supported in part by the European Project (KoroiBot) FP7-ICT-2013-10/611909 and in part by ROBOTIS which provided motors for Leo. (*Corresponding author: Ivan Koryakovskiy.)*

I. Koryakovskiy, H. Vallery, and R. Babuška are with the Robotics Institute, TU Delft, Delft 2628 CD, The Netherlands (e-mail: i.koryakovskiy@tudelft.nl; h.vallery@tudelft.nl; r.babuska@tudelft.nl).

M. Kudruss is with the Interdisciplinary Center for Scientific Computing, Heidelberg University, Heidelberg 69117, Germany (e-mail: manuel.kudruss@iwr.uni-heidelberg.de).

W. Caarls is with the Department of Electrical Engineering, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro 22451-000, Brazil (e-mail: wouter@caarls.org).

Digital Object Identifier 10.1109/LRA.2018.2800106

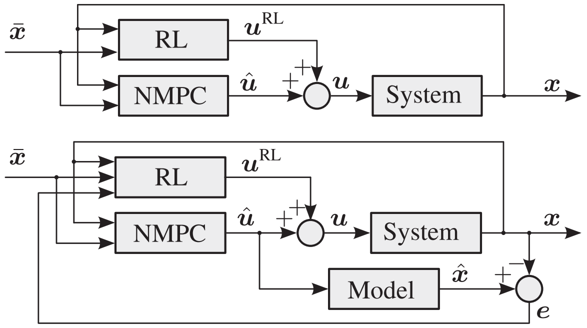

Fig. 2. Compensatory action learning (*top*). Model-plant mismatch learning (*bottom*).

performance measures, the obtained policy is optimal with respect to the real system. The second approach learns a compensatory signal from the difference of transitions predicted by the internal model and the actual transition. In this case, RL uses a different optimization goal, which does not divert NMPC from reaching its objective. As a result, the model-plant mismatch is eliminated by forcing the real system to behave as if it has no uncertainties.

We conduct simulated and real experiments with Leo and demonstrate the advantage of our proposal in the presence of temperature- and torque-dependent Coulomb friction.

II. Related Work

From a control theoretic viewpoint, our approaches can be compared to adaptive internal model control (IMC) [5]. The implementation requires an explicit model of the plant to be used as part of the controller. However, in adaptive IMC, the structure of the unknown system is determined offline, while its parameters can be inferred by online parameter estimation [3]. A particular shortcoming is that the structure needs to be identified precisely; otherwise a model-plant mismatch remains. The proposed approaches require neither precise identification of the structure nor of the parameters.

As a learning controller, we employ model-free on-policy deterministic policy gradient (DPG) [6]. In principle, any model-free RL algorithm such as [7]–[10] can be used. However, lack of safety measures and sample complexity of the algorithms limits their application to real systems.

Learning the forward model of the system demonstrates the lowest number of interactions with it [11], [12]. Learning the inverse model [13], [14] assumes that the model can connect successive states prescribed by the nominal controller.

When the approximate model of the system is available, it is possible to pre-train the initial policy, which can speed up learning. The two-step sequential approach is proposed in [15]. First, an iterative linear-quadratic-Gaussian algorithm is used to design an initial policy. Then, the policy is refined using the PI^2 algorithm on the real system. Another approach is to iteratively learn the difference model of the measured state and the state obtained on the approximate model and adopt this difference model for improving the policy [16], [17]. Finally, the authors in [18] use an ensemble of slightly perturbed model parameters to learn a robust policy.

Learning involving off-line planning or human-expert demonstrations [1], [19]–[22] constrains the problem space, thus reducing hardware damage. This option requires either a hand-coded suboptimal policy or a skilled human operator.

For a bipedal robot where any failure can be catastrophic, the above methods are not suited even given a good starting policy, because it is likely that RL will result in at least several failures during subsequent policy improvement episodes.

It is possible to guarantee safe learning when one can either predict repercussions of bad actions [23] or has a backup policy to lead the system back to safe states [24], [25]. In this article, we do not guarantee safe learning, though the proposed approaches in practice can be safe.

Our contribution is twofold. First, we propose approaches that can in principle compensate any type of uncertainty which preserves the Markov property, without a time-consuming structure identification process and expert-designed models of friction, backlash, etc. The approaches can be implemented on top of an existing model-based controller which makes it easier to integrate into the various fields of engineering, such as robotics, chemistry or computer science. Second, we demonstrate successful learning results on a real robot.

III. Background

A. Problem Statement

Consider the nonlinear time-invariant system in the form of

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\rho}) \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ the system state at time t, $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ is the control vector of joint motor voltages applied to the system at time t, and $\boldsymbol{\rho}$ is an unknown structural uncertainty. The presence of uncertainty causes the model-plant mismatch $\boldsymbol{e}$ which is formulated as the difference between the real system state $\boldsymbol{x}$ and the simulated state of the model $\hat{\boldsymbol{x}}$, see Fig. 2. We do not make any assumption on how the uncertainty enters the equations. Thus it represents the general concept of mismatch.

B. Nonlinear Model Predictive Control

The nominal feedback is provided by NMPC, a closed-loop control strategy in which the control action is computed from the current system state by solving an open-loop optimal control problem on a finite prediction horizon $[0, T]$ online,

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \quad \int_0^T L(\boldsymbol{x}(t), \boldsymbol{u}(t))\, \mathrm{d}t \tag{2a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), 0), \tag{2b}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0,$$

$$\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq 0. \tag{2c}$$

Here we strive to find a control trajectory $\boldsymbol{u}(t)$ such that an objective function composed of a Lagrange term $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is minimized. The state trajectory $\boldsymbol{x}(t)$ is characterized by the dynamic system (1). In (2b), we assume the idealized model, $\boldsymbol{\rho} = 0$, because nothing is known about the uncertainties in the real system. In addition, we impose mixed state-control path constraints $\boldsymbol{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ such as constraints on joint angles ensuring the static stability of the robot together with constraints formulating limits on the maximum motor input voltage.

In the simulated experiment, we use a nominal NMPC scheme that is based on direct multiple shooting [26]. Controls $\boldsymbol{u}$ are approximated as piecewise constant functions. The cost of some discretized trajectory $\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{x}_1, \boldsymbol{u}_1, \ldots$ obtained using policy $\boldsymbol{u}_k = \boldsymbol{\pi}(\boldsymbol{x}_k)$ is denoted as $\mathcal{L} = \sum_k L(\boldsymbol{x}_k, \boldsymbol{u}_k)$.

To achieve real-time control on the robot, we implement a parallelized version of the NMPC scheme [27], where one controller provides a fast feedback by efficiently reusing control problem linearizations of the last iteration, while the second controller prepares the next nonlinear step.

C. Reinforcement Learning

RL is a trial-and-error method which does not require an explicitly given model, and can naturally adapt to uncertainties in the real system [28]. RL assumes the system is stochastic, and thus it maximizes the expected discounted return

$$G_k^\gamma = \mathbb{E}\left\{\sum_{i=0}^{\infty} \gamma^i r(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}, \boldsymbol{x}_{k+i+1})\right\} \tag{3}$$

$$r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = \begin{cases} -L(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) & \text{if } \boldsymbol{g}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) \geq 0 \\ R_{\text{a}} & \text{otherwise.} \end{cases} \tag{4}$$

Here $r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1})$ is the scalar reward given for a transition from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$ caused by the control signal $\boldsymbol{u}_k = \boldsymbol{\pi}(\boldsymbol{x}_k) + \boldsymbol{n}, \boldsymbol{n} \sim \boldsymbol{\mathcal{N}}$ chosen from some policy $\boldsymbol{\pi}$. Discount rate $\gamma \in [0, 1)$ is required for integrability of the infinite sum. Its role is similar to NMPC horizon T. Constraints (2d) are established by means of the large negative reward R_{a}. If constraints are violated, the episode is terminated, and the system is restarted in state $\boldsymbol{x}_0$. Usually, RL requires at least several repetitions to estimate the return (3) correctly. These repetitions are obtained by adding exploration noise $\boldsymbol{\mathcal{N}}$ to control signals $\boldsymbol{u}_k$ at every time step. The outcome of repetitions is not known in advance and therefore may be damaging to the system.

An important aspect of learning is the preservation of the Markov property, which assumes that the next state $\boldsymbol{x}_{k+1}$ depends only on the current state $\boldsymbol{x}_k$ and action $\boldsymbol{u}_k$, but not on previous states or actions [28].

We solve problem (3) using DPG with linear function approximation and compatible features, chosen for its ability to optimize continuous control policies and fast convergence.

IV. Proposed Combination Approaches

A. Compensatory Action Learning (CAL)

In the proposed combination schemes shown in Fig. 2, we use $\hat{\boldsymbol{u}}$ notation for the output of the NMPC controller and $\boldsymbol{u}^{\text{RL}}$ notation for the output of the RL controller.

CAL approach learns a compensatory control action added to the control input computed by nominal NMPC. For learning, we use the NMPC-inspired reward (4), which establishes similar optimization goals for both controllers. Due to small differences in formulation and function approximations in RL, the obtained policy might be suboptimal compared to NMPC.

B. Model-Plant Mismatch Learning (MPML)

In the MPML approach, RL maximizes return (3) where the reward is given by

$$r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = -\|\boldsymbol{e}_{k+1}\|_2 = -\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1}\|_2. \tag{5}$$

Cumulative model-plant mismatch along some trajectory is calculated as the negative value of the undiscounted return $\mathcal{E} = \sum_k \|\boldsymbol{e}_k\|_2$.

In the following, we prove two theorems. The first one explains the behavior of the system when the model-plant mismatch is minimized by RL. The subsequent corollary considers the case when the cumulative return ($\gamma > 0$) is useful for discovering a better control policy. The second theorem specifies conditions under which the system retains the Markov property. If the property is preserved, then RL will not diverge, and the mismatch can be minimized. In this case, the MPML performance depends on the RL learning capability.

Theorem 1: The outcome of the control policy approaches the outcome of the optimal policy with respect to the idealized model *iff* the model-plant mismatch $\boldsymbol{e}_k \to 0$ when $k \to \infty$.

Proof: Writing the mismatch as $\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \to 0$ results in $\boldsymbol{x}_k \to \hat{\boldsymbol{x}}_k$. Assuming the nominal controller can reach the setpoint on the model, $\hat{\boldsymbol{x}}_k \to \bar{\boldsymbol{x}}_k$, implies that the system state will also approach the setpoint, $\boldsymbol{x}_k \to \bar{\boldsymbol{x}}_k$. Since the mismatch $\boldsymbol{e}_k$ is minimized in every point of the reference trajectory $\bar{\boldsymbol{x}}_k$, we arrive at the proof of the theorem. The same logic holds for the reverse. ■

Corollary 1: If there is a time step k such that $\boldsymbol{e}_k \neq 0\ \forall \mathbf{u}_k$, then $\min_{\boldsymbol{\pi}}|_{\gamma=0}\, \mathcal{E} \geq \min_{\boldsymbol{\pi}}|_{\gamma>0}\, \mathcal{E}$. When the mismatch is eliminated along the reference trajectory, strict equality holds.

The corollary is based on the RL result that larger γ improves the quality of the policy [29]. However, if there exists a control action which achieves zero mismatch in every state, then maximizing immediate rewards ($\gamma = 0$) is desirable because the problem becomes computationally easier.

In the following theorem, we assume that the system (1) can be discretized as $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho})$ and the setpoint $\bar{\boldsymbol{x}}$ can be included into the state $\boldsymbol{x}$ for simplicity.

Theorem 2: The system controlled by the nominal controller is Markov w.r.t. RL if (a) the system itself is Markov w.r.t. RL, $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho})$; (b) the internal model is Markov w.r.t. the nominal controller, $\hat{\boldsymbol{x}}_{k+1} = f(\boldsymbol{x}_k, \hat{\boldsymbol{u}}_k, 0)$; and (c) the nominal controller response $\hat{\boldsymbol{u}}_k \equiv \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \boldsymbol{m}, \boldsymbol{m} \sim \boldsymbol{\mathcal{M}}$ is stochastic with some stationary distribution $\boldsymbol{\mathcal{M}}$.

Proof: First, by looking at the bottom diagram of Fig. 2 we write the condition (a) and show that the distribution of states $\boldsymbol{x}_{k+1}$ is defined by the current state $\boldsymbol{x}_k$

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f(\boldsymbol{x}_k, \hat{\boldsymbol{u}}_k + \boldsymbol{u}_k^{\text{RL}}, \boldsymbol{\rho}) \\ &= f(\boldsymbol{x}_k, \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \boldsymbol{\pi}(\boldsymbol{x}_k) + \boldsymbol{m} + \boldsymbol{n}, \boldsymbol{\rho}). \end{aligned}$$

Next, we show that the reward is also defined by $\boldsymbol{x}_k$.

$$\begin{aligned} r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) &= -\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1}\|_2 \\ &= -\|\boldsymbol{x}_{k+1} - f(\boldsymbol{x}_k, \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \boldsymbol{m}, 0)\|_2 \end{aligned}$$

The expected return averages the sum of discounted rewards over the distribution of states and controls. Since both the dynamics and the return are predictable from the current state

$\boldsymbol{x}_k$, we conclude that the system controlled by the nominal controller is Markov w.r.t. RL. ■

Note that the real system does not have to be Markov with respect to NMPC, which means that any uncertainty $\boldsymbol{\rho}$ can be compensated. We use this observation in the real example where $\boldsymbol{\rho}$ depends on motor temperature which does not enter the model, but is used in RL as an extra state variable.

In the special case of an affine system (1) w.r.t. controls, $\boldsymbol{x}_{k+1} = f^x(\boldsymbol{x}_k, \boldsymbol{\rho}) + f^u(\boldsymbol{x}_k, \boldsymbol{\rho})\boldsymbol{u}_k$, a perfectly learned RL control, i.e., $\boldsymbol{e} = 0$, is explicitly given by

$$\begin{aligned} \boldsymbol{u}_k^{\mathrm{RL}} = {} & \left(f^u(\boldsymbol{x}_k, \boldsymbol{\rho})^\top f^u(\boldsymbol{x}_k, \boldsymbol{\rho})\right)^{-1} f^u(\boldsymbol{x}_k, \boldsymbol{\rho})^\top [f^x(\boldsymbol{x}_k, 0) \\ & - f^x(\boldsymbol{x}_k, \boldsymbol{\rho}) + (f^u(\boldsymbol{x}_k, 0) - f^u(\boldsymbol{x}_k, \boldsymbol{\rho}))\,\hat{\boldsymbol{u}}_k], \end{aligned}$$

where $f^x(\boldsymbol{x}_k, \boldsymbol{\rho}) \in \mathbb{R}^{n_x \times 1}$ and $f^u(\boldsymbol{x}_k, \boldsymbol{\rho}) \in \mathbb{R}^{n_x \times n_u}$ are terms independent of $\boldsymbol{u}_k$. As expected, the control $\boldsymbol{u}^{\mathrm{RL}}$ captures the model-plant mismatch caused by $\boldsymbol{\rho}$.

V. Experiments

A. Robot Leo

Robot Leo is depicted in Fig. 1. We focus on the task of reaching upper and lower setpoints which together realize a squatting motion. For this purpose, the robot is fixed to the ground plate below its feet. Falling situation is recognized when absolute torso angle becomes larger than 57.3°. Leo has seven degrees of freedom driven by Dynamixel XM430 servo motors, three in each leg at ankle, knee and hip and one motor in the shoulder. Gearboxes are subject to Coulomb friction dependent on motor temperature and torque.

The robot state $\boldsymbol{x} = (\boldsymbol{\phi}, \dot{\boldsymbol{\phi}}, p, \tau_{\text{knee}})^\top$ is defined as a vector of all but shoulder joint angles $\boldsymbol{\phi}$, corresponding angular velocities $\dot{\boldsymbol{\phi}}$, setpoint height $p \in \{0.28\ \text{m}, 0.35\ \text{m}\}$ and mean temperature of knee motors τ_{knee}.

Exploiting the symmetry of Leo, we apply the same control voltages to both legs. The shoulder is actuated using a PD controller. The setpoints are switched over when the robot root point appears to be within ±0.01 m away from it. In the simulated experiment, we add shoulder angle and velocity to the state, and the shoulder voltage is also learned. The robot is initialized in a setpoint chosen randomly at the beginning of every episode. The idealized model is made such that no friction in joints is present. For the realistic model, we add Coulomb friction $\boldsymbol{u}_{\text{fr}} = -0.2\tanh(2000\dot{\boldsymbol{\phi}})$ in all joints.

The control delay of $13.0 \pm 1.7\,\text{ms}$ comprises measurement, computation and actuation delays. A sampling period of 33.3 ms is chosen to be larger than the control delay.

B. Objective Function and Constraints

The NMPC objective function is defined by (2a) with

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{u}) = {} & 0.05\,(h(\boldsymbol{\phi}) - p)^2 + 0.10\,(x_c(\boldsymbol{\phi}) - \bar{x}_c)^2 \\ & + 0.05\,(pose(\boldsymbol{\phi}) - 0.3)^2 + 0.003\,\dot{\boldsymbol{\phi}}^\top \dot{\boldsymbol{\phi}}. \end{aligned}$$

Here, the first term accounts for the vertical distance $h(\boldsymbol{\phi})$ to a setpoint p, the second term maintains the horizontal position of the center of mass $x_c(\boldsymbol{\phi})$ close to predefined value $\bar{x}_c$, the third term containing $pose(\boldsymbol{\phi}) = \phi_{\text{ankle}} + \phi_{\text{knee}} + \phi_{\text{hip}}$ is used as a regularization term improving the stability of the robot, and the last term favors small velocities.

We formulate static stability as a constraint $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = (x_t - x_c(\boldsymbol{\phi}), x_c(\boldsymbol{\phi}) - x_h)^\top$, where x_t, x_h denote the position of the tip and the heel of robot feet. Additionally, robot angles and controls are subject to constraints

$$\begin{bmatrix} -1.57 \\ -2.53 \\ -0.61 \end{bmatrix} \le \phi_i \le \begin{bmatrix} 1.45 \\ -0.02 \\ 2.53 \end{bmatrix} \qquad \begin{aligned} & \{|u_i|, |\hat{u}_i|, |u_i^{\mathrm{RL}}|\} \le 10.0\ \text{V} \\ & i \in \{\text{ankle}, \text{knee}, \text{hip}\}. \end{aligned}$$

For MPML approach the reward (5) is calculated based on the joint angles $r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = -\|\boldsymbol{\phi}_{k+1} - \hat{\boldsymbol{\phi}}_{k+1}\|_2$.

C. Parameters

The time horizon T for NMPC optimization is selected to be 1 s. One learning or testing episode lasts for 15 s. Advantage, critic and actor learning rates are chosen to be 0.01, 0.10 and 0.01, respectively. Additional parameters include discount rate $\gamma = 0.97$, and an eligibility trace decay rate of 0.65. We rely on NMPC to avoid falls of the robot, therefore negative reward R_{a} is not used. Exploration is achieved by Ornstein-Uhlenbeck noise $\Delta\boldsymbol{u}_{k+1} = 0.5\,\Delta\boldsymbol{u}_k + \mathcal{N}(0, \sigma)$ with $\sigma = 0.005$. For the real experiment, we select a higher advantage learning rate and increase exploration.

D. Evaluation

For quantitative assessment, we evaluate objective (2a) separately for reaching upper and lower setpoints $\mathcal{L}^{\{\mathrm{u,l}\}} = \sum L(\boldsymbol{x}, \boldsymbol{u})$. Second, we evaluate the minimization of model-plant mismatch $\mathcal{E}^{\mathrm{u,l}} = \sum \|\boldsymbol{e}\|_2$ for reaching both setpoints separately. Third, we calculate root mean squared error (RMSE) between transitions obtained by both approaches and NMPC executed on the idealized model. Finally, to experimentally demonstrate safety barriers imposed by NMPC, we calculate the cumulative number of falls and violation of NMPC constraints at multiple levels of exploration noise σ for two proposed approaches and DPG.

For qualitative assessment, we calculate the number of squats the robot performs during the testing episode. This measure should be accounted only as a learning progress indicator since it is not included in the optimization objective.

E. Simulation Results

In Fig. 3 and Table I, we demonstrate a significant difference in the performance of standalone NMPC on the idealized and realistic models. On the idealized model, NMPC realizes three squats. On the realistic model, NMPC can reach neither upper nor lower switching points, which results in the inability to squat and high costs $\mathcal{L}$. This result motivates the need for an adaptive component in the controller.

To compare the performance of the proposed approaches to the baseline performance of NMPC, we perform learning for 10^6 time steps. The time was enough for CAL and MPML to converge, while DPG required about hundred times more steps. Thus, its results were excluded from the comparison.

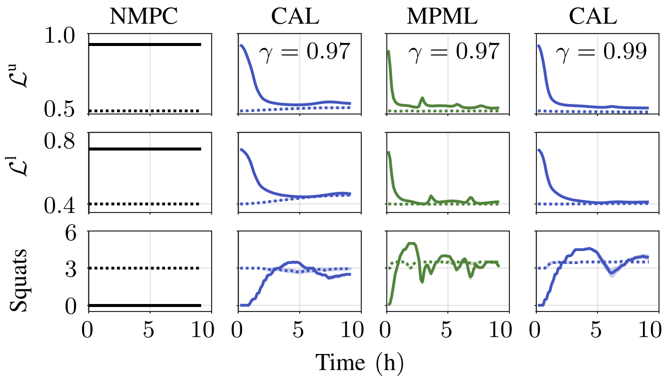

Fig. 3. Learning to reach upper (*top row*) and lower setpoints (*middle row*) in simulation. Number of squats is given in the *bottom row*. *Dotted* and *solid* lines show learning on the idealized and realistic models, respectively. Means with upper and lower 95% confidence limits are shown for 10 runs.

TABLE I
FINAL PERFORMANCE OF METHODS

Method	$\mathcal{L}^{\mathrm{u}} \times 10$	$\mathcal{L}^{\mathrm{l}} \times 10$	Number of squats	RMSE$\times 10^4$
Idealized model				
NMPC	5.2	**4.0**	**3.0**	**0.0**
CAL$^{\gamma = 0.97}$	5.4	4.4	3.0(0.1)	6.4(1.5)
MPML	5.2	**4.0**	3.5	1.7(0.1)
CAL$^{\gamma = 0.99}$	**5.1**	**4.0**	3.5	11.8(4.0)
Realistic model				
NMPC	9.3	7.1	0	20.1
CAL$^{\gamma = 0.97}$	5.6	4.6	2.5	10.4(0.5)
MPML	**5.4**	**4.1**	**3.2(0.1)**	**4.3(1.2)**
CAL$^{\gamma = 0.99}$	**5.4**	**4.1**	3.9(0.2)	13.2(1.2)
Real robot				
NMPC$^{38\,^\circ\mathrm{C}}$	22.0(2.1)	–	0	
MPML	**7.9(2.9)**	**4.4(1.8)**	**3.3(1.1)**	**94.9(26.5)**

Note: Significant Width of the confidence interval is shown in brackets

We notice that on the idealized model the performance of both approaches becomes slightly worse than the baseline performance of NMPC. For CAL$^{\gamma = 0.97}$, we observe deviation from the optimal policy which is seen in the increase of $\mathcal{L}$ cost. Yet, the approach is able to keep the number of squats close to the baseline value. For MPML, costs $\mathcal{L}$ do not change, but the number of squats increases by 0.5 which indicates that the approach reaches the upper setpoint right before the episode ends. Deviation of the learned trajectory from the idealized one is captured by RMSE which is nonzero for both approaches. For the CAL and MPML approaches the mean RMSE is 68.6% and 91.7% below the reference of 20.4 ± 0.2 which is RMSE of NMPC trajectory obtained on the realistic model.

Results of the realistic model experiment show that both approaches improve the performance of NMPC. The decrease of the $\mathcal{L}$ cost is at least 35.2% and 41.9% for CAL$^{\gamma = 0.97}$ and MPML approaches, respectively. In terms of squats, both learning approaches overshoot the NMPC baseline of 3 squats and then continuously reduce the number towards the baseline. RMSE increases comparing to the idealized model experiment, but still remains significantly below the reference value.

To find the reason of CAL$^{\gamma = 0.97}$ performance decrease on the idealized model, we test the approach with a discount rate of $\gamma = 0.99$. Increasing γ leads to a longer planning horizon which makes RL return (3) more similar to NMPC objective (2a). It turns out that CAL$^{\gamma = 0.99}$ obtains lower $\mathcal{L}$ costs not only comparing to CAL$^{\gamma = 0.97}$ but also comparing to the baseline NMPC. We believe the reason of this is due to the early switching of setpoints described above. While CAL$^{\gamma = 0.99}$ can learn this fact, NMPC is not aware of it.

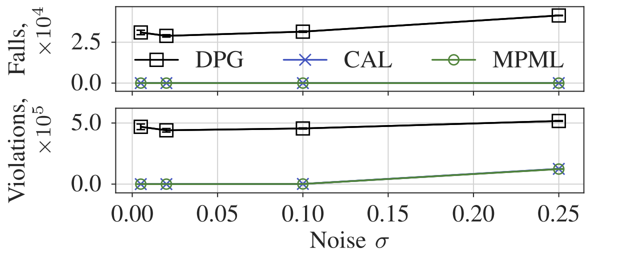

Fig. 4. The number of falls (*top*) and NMPC constraint violations (*bottom*) as functions of σ. Means with upper and lower 95% confidence limits are shown for 10 runs.

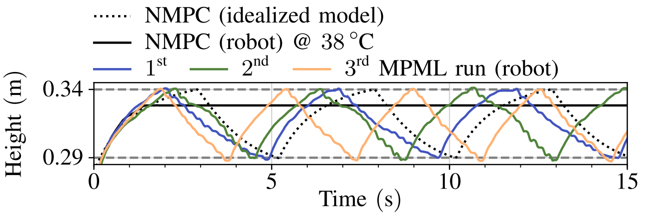

Fig. 5. Robot root point trajectories obtained after learning.

In Fig. 4, we plot the number of falls and NMPC constraint violations accumulated over 10^6 time steps. Here, we prematurely stopped DPG for the sake of results comparability. The proposed approaches are almost identical. Both approaches prevent the robot from falling, while constraints get violated at $\sigma > 0.1$. A different picture is seen in DPG results. The smallest number of falls and constraint violations is achieved for the value of $\sigma = 0.02$. Smaller σ reduces the learning pace, while larger values increase chances of fall.

MPML learns almost twice as fast as CAL and does not exhibit deviating behavior, which are the main reasons for testing the approach on the real robot.

F. Results on the Real Robot

Results of standalone NMPC on Leo are shown in Fig. 5. While on the idealized model NMPC successfully reaches switching points, on the real robot the controller is not able to do so. The reason is due to Coulomb friction in gearboxes, which depends on motor temperature and the applied torque. Modeling these effects is possible but requires a precise identification of the underlying physical processes.

To circumvent this problem, we apply the proposed MPML approach. In Fig. 5, results of three independent runs are shown. MPML successfully realizes squatting by learning the compensation signal. The variation of motor temperature leads to noticeable differences in the trajectories. The trajectory obtained in the 3rd run is less noisy and squatting is faster than the one achieved in 1st and 2nd runs. In particular, the gradient of the downwards motion in the 3rd run is very similar to the idealized

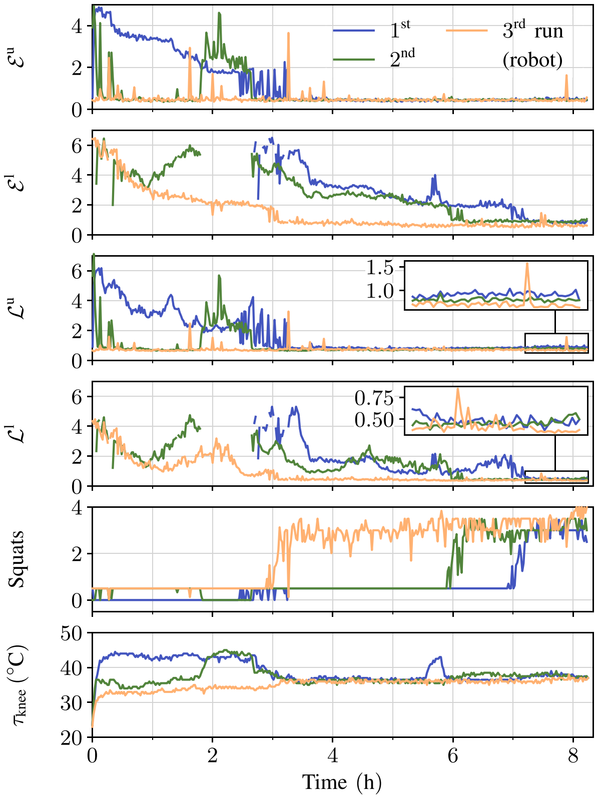

Fig. 6. Model-plant mismatch (*upper two*), nominal controller objective (*middle two*), number of squats and the mean temperature of knee motors obtained during three real learning experiments. Measurements of $\mathcal{E}^{\mathrm{l}}$ and $\mathcal{L}^{\mathrm{l}}$ are missing when upper switching point is not reached.

NMPC run, except for the later part where the slow approach towards the setpoint diminishes.

Variation of motor temperature also leads to differences in the learning progress, see Fig. 6. 1st and 2st runs require substantially longer time before the squatting cycle is observed. This is due to the increase of motor temperature above $40.0\,^{\circ}\mathrm{C}$ which requires additional exploration of the state space. Nevertheless, all runs successfully attain a stable squatting cycle after 7.25 h.

Model-plant mismatch $\mathcal{E}^{\mathrm{u}}$ and $\mathcal{E}^{\mathrm{l}}$ is minimized to about 0.5 and 0.8 for reaching upper and lower setpoints, respectively. As it was expected, minimization of model-plant mismatch leads to minimization of the nominal controller objective (2a) shown by plots $\mathcal{L}^{\mathrm{u}}$ and $\mathcal{L}^{\mathrm{l}}$. The smallest final costs are incurred by the 3rd run because it was stuck the least due to temperature fluctuations, while the largest costs are incurred by the 1st run which was stuck the most.

RMSE after learning is calculated in Table I. RMSE of MPML trajectories is much higher than for the realistic model, and it also exhibits more variability. Unfortunately, it is not possible to obtain the reference RMSE value of the real robot.

Fig. 7 shows the MPML knee control signal and the RL compensation component of it. For reference, NMPC control on the idealized model is also shown. MPML controls are very oscillatory comparing to NMPC. Nonetheless, the robot neither fell down, nor were its motors damaged, which is a significant result, cf. [1]. As is expected with Coulomb friction compensation, RL learned to apply positive and negative controls for the upward and downward motions, respectively.

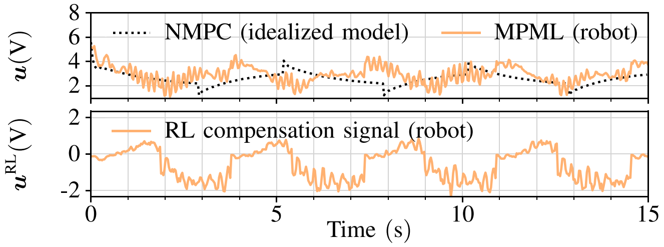

Fig. 7. Knee control signal of NMPC applied to the idealized model and of MPML applied to the real robot after learning (*top*). Compensation signal learned by MPML (*bottom*).

VI. Discussion

Even though the final CAL policy is optimal w.r.t. the real system, the policy is suboptimal w.r.t. the objective of the nominal controller (2a). This result can be explained by the fact that RL and NMPC objectives are not exactly same. While NMPC optimizes the undiscounted cost up to horizon T, RL optimizes γ-discounted reward on the infinite horizon. All in all, RL views the system and the nominal controller as a hybrid entity and the obtained policy becomes optimal w.r.t. the RL objective. This observation also explains the inability of CAL$^{\gamma=0.97}$ to reach the optimal performance on the realistic model, even though it can significantly improve the performance of the nominal controller. The longer prediction horizon used by CAL$^{\gamma=0.99}$ attains a better performance.

Another problem of CAL is the slow convergence which is probably caused by the fact that the reward constructed from the quadratic objective function of the nominal controller results in small gradients [2]. This hypothesis is supported by the fact that DPG with a quadratic cost function learns the task extremely slowly. To mitigate this, the RL cost function can be modified. The downside of it can be the difficulty of predicting the outcome of such modification, e.g., robot velocity may change drastically.

The MPML approach is free from these complications. However, it should be emphasized that MPML optimizes policy w.r.t. the internal model, that is RL forces the system to behave like the idealized model. In principle, a large mismatch may pose a problem because the obtained policy will be less optimal w.r.t. the real system and control constraints may prevent the necessary compensation to be applied. However, in our experiments, this is not a problem. Minimization of the model-plant mismatch $\mathcal{E}$ closely follows minimization of the nominal controller cost $\mathcal{L}$. MPML successfully learns to compensate the unknown Coulomb friction as well as its dependency on motor temperature and torque.

The MPML approach obtains the lowest RMSE. This does not come as a surprise, as MPML directly minimizes the mismatch by the specifically constructed reward function. CAL also minimizes RMSE, even though its primary goal is not defined in terms of such minimization. Arguably, in order for NMPC to successfully complete the task, the realistic model should resemble the idealized one which is achieved by learning with RL in both approaches.

For both proposed approaches, a little deviation caused by RL exploration leads to an immediate setback reaction from NMPC. Our simulated experiment reveals a wide range of admissible exploration noise σ for which the number of NMPC constraint violations and robot falls is zero. This result demonstrates the role of NMPC which provides safety barriers to constrain RL exploratory actions near dangerous state space regions. However, there are disadvantages. First, the formulated task demands deliberate control learning which is difficult in the presence of Coulomb friction. If the robot starts moving after a slight overshoot caused by RL exploration, this immediately causes the decrease of friction (Stribeck effect), and at the next sampling moment, the system displacement appears to be too large. NMPC counteracts, so that resulting trajectories appear to be oscillatory. The other reason of oscillations is due to the large control delay. Given the results, it is hard to assess the role of NMPC counter-reaction in the oscillatory trajectories, but we expect that reduction of sampling time and control delay will reduce oscillations. Second, NMPC can drive the system very close to constraint boundaries. For some systems violation of constraints can be very critical, however, this is not true in our case.

We note that the success of model-plant mismatch compensation depends on the learning capabilities of RL on the hybrid system mentioned above. Whether there is a decrease or increase of computational complexity of that system against the original system remains an open problem.

It is common to compensate for a steady-state error in a task completion by adding an integral term to the objective that is tuned by experimental data. Learning the actual model-plant mismatch with MPML goes far beyond cost tuning. It allows to predict the outcome of executed actions since the learned trajectory is expected to be optimal w.r.t. the idealized model.

VII. Conclusion

We proposed two learning approaches to compensate model-plant mismatch. Our simulation results demonstrated the feasibility of both approaches. We implemented the better one on a real robot affected by torque and temperature dependent friction and autonomously learned a squatting task. Trying to achieve a similar performance with the standalone nominal controller would require tedious identification of the law of such a dependency. During learning, the robot did not fall.

Several avenues can be explored in future. First, one may reduce learning time by using different function approximators or RL algorithms. A wide range of recent model-free RL can be utilized in a straightforward way. Second, it is important to validate MPML on more challenging tasks. It is also a relatively straightforward implementation if the nominal controller is already set up, and RL actions are bounded.

Acknowledgment

The authors would like to thank A.V. Proskurnikov for his insightful comments.

References

[1] E. Schuitema, "Reinforcement learning on autonomous humanoid robots," Ph.D. dissertation, TU Delft, Delft, The Netherlands, 2012.

[2] I. Koryakovskiy *et al.*, "Benchmarking model-free and model-based optimal control," *Robot. Auton. Syst.*, vol. 92, pp. 81–90, 2017.

[3] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, and H. G. Bock, "A real-time algorithm for moving horizon state and parameter estimation," *Comput. Chem. Eng.*, vol. 35, no. 1, pp. 71–83, 2011.

[4] Y. E. Bayiz and R. Babuska, "Nonlinear disturbance compensation and reference tracking via reinforcement learning with fuzzy approximators," *IFAC Proc. Vol.*, vol. 47, no. 3, pp. 5393–5398, 2014.

[5] A. Datta and L. Xing, "The theory and design of adaptive internal model control schemes," in *Proc. Amer. Control Conf.*, 1998, vol. 6, pp. 3677–3684.

[6] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

[7] S. M. Kakade, "A natural policy gradient," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 1531–1538.

[8] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[9] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Represent.*, 2016.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

[11] S. Kamthe and M. P. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," arXiv:1706.06491, 2017.

[12] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2829–2838.

[13] M. Kawato, "Feedback-error-learning neural network for supervised motor learning," *Adv. Neural Comput.*, vol. 6, no. 3, pp. 365–372, 1990.

[14] P. Christiano *et al.*, "Transfer from simulation to real world through learning deep inverse dynamics model," arXiv:1610.03518, 2016.

[15] F. Farshidian, M. Neunert, and J. Buchli, "Learning of closed-loop motion control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 1441–1446.

[16] S. Ha and K. Yamane, "Reducing hardware experiments for model learning and policy optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2620–2626.

[17] 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.

[18] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "EPOpt: Learning robust neural network policies using model ensembles," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[19] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *Int. J. Robot. Res.*, vol. 29, no. 13, pp. 1608–1639, 2010.

[20] J. Peters, K. Mlling, and Y. Altun, "Relative entropy policy search," in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 1607–1612.

[21] S. Levine and V. Koltun, "Guided policy search," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1–9.

[22] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 528–535.

[23] T. M. Moldovan and P. Abbeel, "Safe exploration in Markov decision processes," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 1711–1718.

[24] A. Hans, D. Schneegass, A. M. Schafer, and S. Udluft, "Safe exploration for reinforcement learning," in *Proc. Eur. Symp. Artif. Neural Netw.*, 2008, pp. 143–148.

[25] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. H. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," arXiv:1705.01292, 2017.

[26] H. G. Bock, M. Diehl, E. A. Kostina, and J. P. Schlöder, "Constrained optimal feedback control of systems governed by large differential algebraic equations," in *Proc. Real-Time PDE-Constrained Opt.*, L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, Eds. Philadelphia, PA, USA: SIAM, 2007, ch. 1, pp. 3–24.

[27] M. Kudruss, I. Koryakovskiy, H. Vallery, K. Mombaur, and C. Kirches, "Combining multi-level real-time iterations of nonlinear model predictive control to realize squatting motions on Leo," Optimization Online, Jan. 2018. [Online]. Available: http://www.optimization-online.org/DB_HTML/2018/01/6425.html

[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[29] N. Jiang, A. Kulesza, S. Singh, and R. Lewis, "The dependence of effective planning horizon on model accuracy," in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2015, pp. 1181–1189.