# Benchmarking model-free and model-based optimal control

Koryakovskiy, Ivan; Kudruss, Manuel; Babuska, Robert; Caarls, Wouter; Kirches, Christian; Mombaur, Katja; Schlöder, Johannes P.; Vallery, Heike

**DOI**
10.1016/j.robot.2017.02.006

**Publication date**
2017

**Document Version**
Accepted author manuscript

**Published in**
Robotics and Autonomous Systems

**Citation (APA)**
Koryakovskiy, I., Kudruss, M., Babuška, R., Caarls, W., Kirches, C., Mombaur, K., ... Vallery, H. (2017). Benchmarking model-free and model-based optimal control. Robotics and Autonomous Systems, 92, 81-90. https://doi.org/10.1016/j.robot.2017.02.006

Link to formal publication in Robotics and Autonomous Systems (Elsevier): http://dx.doi.org/10.1016/j.robot.2017.02.006

# Benchmarking model-free and model-based optimal control☆

Ivan Koryakovskiy[a], Manuel Kudruss[*,b], Robert Babuška[e], Wouter Caarls[c,d], Christian Kirches[b], Katja Mombaur[b], Johannes P. Schlöder[b], Heike Vallery[a]

[a]*Delft Biorobotics Lab, Faculty of Mechanical Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands*
[b]*Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany*
[c]*Department of Computer Science, Federal University of Rio de Janeiro, 22451-900 Rio de Janeiro, RJ Brazil*
[d]*Department of Electrical Engineering, Pontifical Catholic University of Rio de Janeiro, 21941-909 Rio de Janeiro, RJ Brazil*
[e]*Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands*

**Abstract**

Model-free reinforcement learning and nonlinear model predictive control are two different approaches for controlling a dynamic system in an optimal way according to a prescribed cost function. Reinforcement learning acquires a control policy through exploratory interaction with the system, while nonlinear model predictive control exploits an explicitly given mathematical model of the system. In this article, we provide a comprehensive comparison of the performance of reinforcement learning and nonlinear model predictive control for an ideal system as well as for a system with parametric and structural uncertainties. The comparison is based on two different criteria, namely the similarity of trajectories and the resulting rewards. The evaluation of both methods is performed on a standard benchmark problem: a cart-pendulum swing-up and balance task. We first find suitable mathematical formulations and discuss the effect of the differences in the problem formulations. Then, we investigate the robustness of reinforcement learning and nonlinear model predictive control against uncertainties. The results demonstrate that nonlinear model predictive control has advantages over reinforcement learning if uncertainties can be eliminated through identification of the system parameters. Otherwise, there exists a break-even point after which model-free reinforcement learning performs better than nonlinear model predictive control with an inaccurate model. These findings suggest that benefits can be obtained by combining these methods for real systems being subject to such uncertainties. In the future, we plan to develop a hybrid controller and evaluate its performance on a real seven-degree-of-freedom walking robot.

*Key words:* Reinforcement Learning, Optimal Control, Nonlinear Model Predictive Control, Parametric Uncertainties, Structural Uncertainties

## 1. Introduction

In robotics, one cannot expect to work with ideal models of the systems under control, or of their environments. Rather, we have to face unforeseen situations and unknown conditions, and aim for reactions that are feasible and, ideally, optimal with respect to given task performance criteria. A typical task is bipedal locomotion, where a robot needs to maintain stability and pace on an uneven floor with uncertain roughness and slope [1].

Two common approaches to control dynamic systems are Nonlinear Model Predictive Control (NMPC) and Reinforcement Learning (RL). Both approaches can cope with uncertainties in the form of model-plant mismatch. Reinforcement learning has been proven suitable as a real-time closed-loop control concept in robotics [2], and NMPC in industry [3]. However, the use of NMPC in robotic applications, especially humanoid robotics and bipedal walking, is still an open research field [4, 5, 6].

In this article, we use a swing-up and balancing problem for a *cart-pendulum* system [7, 8] to quantitatively assess both control approaches. Our choice of this benchmark problem is motivated by the fact that main features of passive dynamic walking can be modeled by an inverted pendulum [9]. The same equivalence holds for the upper body of a more detailed model of a bipedal walker. The study presented in this article highlights the differences in performance of NMPC and RL under structural and parametric uncertainties for this benchmark problem.

*Nonlinear model predictive control.* Nonlinear model predictive control is a closed-loop control strategy in which the control action at the current sampling instant is computed by solving an open-loop optimal control problem over a finite prediction horizon. NMPC, as a model-based optimal control method, relies on a given mathematical model of the real-world system to be controlled. In this context, advanced direct methods of optimal control, see the survey [10], are the methods of choice for computing NMPC feedback control actions in real-time.

For NMPC, full state and parameter information of the model is required to compute the control action. Whenever the full state is not measurable or model parameters are not exactly known, methods of on-line state and parameter estimation have

☆The first two authors contributed equally to this work.
*Corresponding author

to be applied. For this purpose, extended Kalman filters [11] or Moving Horizon Estimation (MHE) techniques [12, 13] have been successfully applied. In this article, MHE is used to estimate uncertain parameters in the model.

*Reinforcement learning.* Reinforcement learning is an active research area in the field of artificial intelligence and machine learning, with applications in control. The most important feature of RL is its ability to learn without prior knowledge about the system. The goal of the learning task is supplied externally in the form of a reward function. RL is a trial-and-error method, which generally takes many iterations before it finds an optimal solution. To reduce the number of interactions with the system, model-learning methods such as Dyna [14], learning from demonstration [15, 16], or optimized control policy parameterizations [17] can be applied. Because RL does not require an explicitly given model, it can naturally adapt to uncertainties of the real system. In this sense, RL can be viewed as a model-free adaptive optimal control approach [18].

*Related work of comparison and combination of RL and NMPC.* In this article, we provide a comprehensive comparison of the performance of RL and NMPC both for an ideal system as well as in the presence of parametric and structural uncertainties. To our knowledge, this is the first time this is done in a systematic and quantitative way for uncertain systems. Based on the presented comparison results, we identify the strong and weak points of both algorithms, which suggests a presence of mutual benefits for their combination.

A related comparative study for ideal systems can be found in [19]. The authors highlight similarities of NMPC and RL, including optimality of methods, truncation of a time horizon, and continuous vs. discrete actions. They show that, for an electrical power oscillations damping problem, NMPC slightly outperforms RL, yet both policies were essentially similar. Furthermore, the authors propose the idea of combining RL and NMPC. In an on-line (local) mode, NMPC could start optimization from the initial guess of the optimal trajectory precomputed by RL in an off-line (globally optimal) mode. Our conclusions go beyond the validation of similarity of solutions or computational benefits for ideal models. We provide numerical evidence that under uncertainties, situations may arise in which one or the other method can be favorable for performance.

A distinction of both methods was observed and successfully realized in a hybrid approach, a variant of the Guided Policy Search algorithm [20]. The approach was able to learn obstacle avoidance policies for a quadrotor [21]. It adopted a collection of MPC roll-outs obtained under full state observation and trained a deep control policy that required only the on-board sensors of the vehicle.

A study of the influence of the RL reward function on steady-state error was performed in [22]. It was shown that direct translation of a quadratic objective function used in standard linear quadratic regulators (LQR) resulted in a consistent, though not acceptable steady-state error. In contrast, using the absolute-value reward function yielded a response with negligible error.

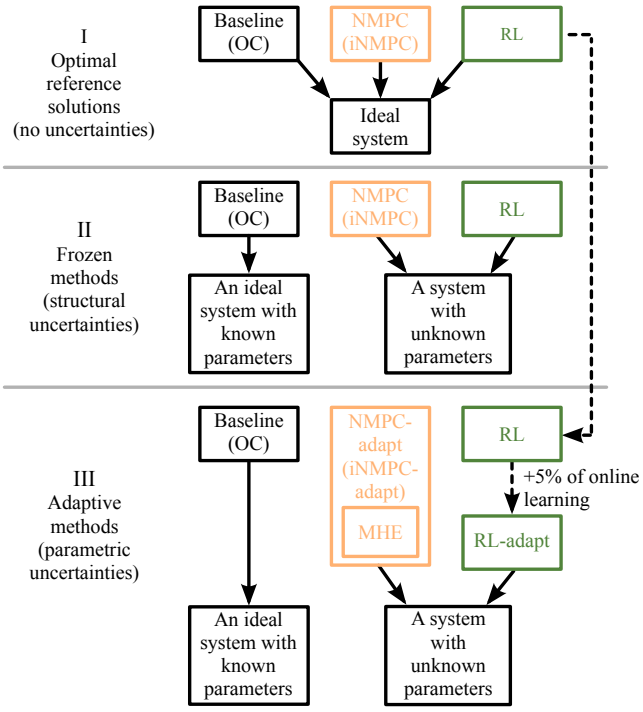


Figure 1: Overview of the computational study. Step I corresponds to a verification of state and control trajectories when problem formulations are equivalent. In steps II and III uncertainties of a varied magnitude are introduced. In the former case "NMPC", "iNMPC" and "RL" are not equipped with an adaptation mechanism while in the latter case they are. In "NMPC-adapt" and "iNMPC-adapt" adaptation is accomplished by means of MHE, and for "RL-adapt" we allow 5 % of additional interaction with the real system.

*Computational study.* The study conducted in this article is set up as follows, see Figure 1. In the first step (I), we establish optimal control ("OC") solutions for the ideal benchmark problem. Then we consider the NMPC formulation and derive the corresponding RL formulation from it. We highlight the changes introduced in both formulations and discuss their effects.

Subsequently, we address the strengths and weaknesses of NMPC and RL in terms of their ability to adapt to *structural* and *parametric* uncertainties. In the second step (II), we investigate NMPC and RL methods that are explicitly unable to adapt to uncertainties. We introduce the term *frozen* to refer to this inability. In the third step (III), the effect of uncertainties and the ability to adapt to them is analyzed for NMPC methods that have explicitly been equipped with the knowledge about the uncertainties and for RL that is allowed to interact with the real system for an additional 5 % of the learning time. We introduce the term *adaptive* to distinguish these from the *frozen* methods.

In the remainder of the article, we use a single RL method denoted as "RL", and two NMPC versions denoted as "iNMPC" and "NMPC". "iNMPC" is an ideal NMPC controller that neglects computational time and returns an optimal control signal immediately. In turn, "NMPC" represents an actual NMPC implementation tuned for real-time feasible control.

## 2. Model-based and model-free optimal control methods

### 2.1. Optimal control

The optimal solutions used as baseline for the comparison are the solutions of the open-loop optimal control problem given by

$$\min_{x(\cdot),u(\cdot)} \quad \int_0^T L(x(t),u(t))\,\mathrm{d}t + M(x(T)) \tag{1a}$$
$$\text{s.t.} \quad \dot{x}(t) = f(x(t),u(t),p), \quad t \in \mathcal{T}, \tag{1b}$$
$$x(0) = x_0, \tag{1c}$$
$$0 \le g(x(t),u(t)), \quad t \in \mathcal{T}, \tag{1d}$$

where we strive to find a control trajectory $u : [0,T] \to \mathbb{R}^{n_u}$ such that an objective function composed of a Lagrange term $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and a Mayer term $M : \mathbb{R}^{n_x} \to \mathbb{R}$ is minimized on a finite time interval $\mathcal{T} = [0,T] \subset \mathbb{R}$. The state trajectory $x : \mathcal{T} \to \mathbb{R}^{n_x}$ is characterized by the dynamic system defined by a set of ordinary differential equations with right hand side $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ , which depends in particular on the model parameters $p \in \mathbb{R}^{n_p}$ of the system. In addition, mixed state-control path constraints $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ are imposed on the system.

We follow a direct and all-at-once approach to optimal control and discretize the control trajectory $u(\cdot)$ on a time grid $0 = t_0 < t_1 < \cdots < t_K < t_{K+1} = T$ by means of basis functions parametrized by piecewise constant control parameters. A direct multiple shooting approach to optimal control [23] then further parametrizes the state trajectory $x(\cdot)$ by introducing $K+1$ variables $\chi_i \in \mathbb{R}^{n_x}$ and by solving initial value problems separately on the time grid. From this discretization and parametrization, a large but structured nonlinear programming problem is obtained that can be solved efficiently with tailored structure-exploiting sequential quadratic programming methods. The evaluation of sensitivities requires $L, M, f, g, h$ in the OC problem (1) to be at least twice continuously differentiable with respect to the optimization variables $x$ and $u$.

### 2.2. Nonlinear model predictive control

NMPC is 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 $\mathcal{T}_{NMPC} \subseteq \mathcal{T}$ on-line, therefore NMPC is also denoted as receding horizon control. In contrast to objective function (1a) of the OC problem, the tracking NMPC minimizes a nonlinear least-squares function composed of

$$L(x(t),u(t)) = \|x(t) - \bar{x}(t)\|_W^2 + \|u(t)\|_V^2, \tag{2}$$

which minimizes the distance to given reference trajectories $\bar{x} : \mathcal{T}_{NMPC} \to \mathbb{R}^{n_x}$ by means of a weighted $L^2$-norm $\|x\|_W = \sqrt{x^T W x}$ with a positive definite weighting matrix $W$.

At the current time instant $t = 0$, the full state $\hat{x}_0 \in \mathbb{R}^{n_x}$ and parameter vector $\hat{p} \in \mathbb{R}^{n_p}$ of the system are embedded into the open-loop optimal control problem by additional constraints replacing (1c) of the OC problem

$$0 = \hat{x}_0 - x(0), \tag{3a}$$
$$0 = \hat{p} \ - p. \tag{3b}$$

In contrast to the OC problem, the parameters $p \in \mathbb{R}^{n_p}$ are part of the optimization variables for the NMPC problem and we denote an estimate of the respective quantity by $\hat{\cdot}$.

State-of-the-art NMPC methods based on nonlinear programming rely on the real-time iteration scheme due to [24, 25] to compute feedback in real-time. This is achieved by careful initialization of the sequential quadratic programming method by separating each iteration into three phases, i.e. 1) preparation (setup of quadratic program), 2) feedback (solution of quadratic program) as soon as the current system information $\hat{x}_0$, $\hat{p}$ is estimated and 3) transition (perform step and shifting). In this way, computationally expensive parts can be separated from time-critical ones and the computational delay of the feedback is reduced to the time required to solve a single quadratic program. Advanced methods further these ideas by dividing the real-time iteration into sub steps that can provide feedback even faster by evaluating only parts of the required Jacobian information, c.f. [26, 27], or applying specialized iterative linear algebra, as in [28].

*Moving horizon state and parameter estimation.* For the state and parameter estimates, i.e. $\hat{x}_0$, $\hat{p}$ in (3), we apply moving horizon estimation (MHE), c.f. [13]. After the estimates for $\hat{x}_0$ and $\hat{p}$ have been obtained, they are embedded in the NMPC problem via the constraints (3) and are considered in the computation of the next feedback control action. In this way, model-plant mismatch and uncertainties can be tackled by including parameters in the NMPC formulation and computing an estimate in every pass of the control loop. Approaches similar to those used to achieve real-time feedback control for NMPC can be used to solve the MHE problem on-line.

In contrast to NMPC, the MHE horizon $\mathcal{T}_{MHE}$ refers to the passed time and the objective function minimizes the squared error between the model response $h$ and measurements $\eta = h(x(p^*),p^*) + \epsilon$ from the real system defined by the true but unknown parameters $p^*$, subject to an additive measurement error $\epsilon \sim \mathcal{N}(0,\Xi)$ with zero-mean and covariance matrix $\Xi = \mathrm{diag}(\xi_0,\ldots,\xi_{n_h})$ given by

$$\sum_{k=-K}^{0} \|(h_k(x(t_k),p) - \eta_k)\|^2_{\Xi_k^{-1}}.$$

Here, the parameters $p$ are part of the optimization variables and the control actions $u$ are fixed to the ones applied to the system in the past.

### 2.3. Reinforcement learning

A common approach in RL is to model the task as a Markov decision process. The process is defined as a quadruple $\langle X, U, \mathcal{P}, \mathcal{R}\rangle$, where $X \subset \mathbb{R}^{n_x}$ is a set of possible states, $U \subset \mathbb{R}^{n_u}$ is a set of possible control actions, $\mathcal{P} : X \times U \times X \to [0,1]$ is a transition function that defines the probability of ending up in state $x_{k+1} \in X$ after executing action $u_k \in U$ in state $x_k \in X$. The reward function $\mathcal{R} : X \times U \times X \to \mathbb{R}$ gives a real-valued reward $r_{k+1} = \mathcal{R}(x_k, u_k, x_{k+1})$ for the particular transition between states. A Markov decision process satisfies the Markov property, which assumes that the next state $x_{k+1}$ depends only on

the current state $x_k$ and action $u_k$, but not on previous states or actions.

A deterministic control policy $\pi : X \rightarrow U$ defines an action $u_k$ taken in a state $x_k$. The goal of the learning process is to find an optimal control policy $\pi^*$ that maximizes the discounted return

$$\Re_k = \mathbb{E}\left\{\sum_{i=0}^{K} \gamma^i r_{k+i+1}\right\},$$

where immediate rewards $r$ are exponentially decayed by the discount rate $\gamma \in [0, 1]$ the further they lie in the future. The task that we consider in this article is a *continuing* task, for which the final time step is infinite, $K \rightarrow \infty$. This requires the use of $\gamma < 1$ to avoid infinite returns.

The value function $V^\pi(x)$ denotes the expected return assuming that the system starts in the state $x$ and then follows a prescribed control policy $\pi$. The optimal control policy $\pi^*$ maximizes the value for each state. Therefore, an optimization of the control policy is tightly coupled with the maximization of the value function in RL.

For real-world systems, continuous control is preferred. This requires a parametrization of the policy $\pi(x)$, e.g. using a set of basis functions and their associated weights. The weights are usually optimized by gradient-descent methods [29, 30], or by global gradient-free methods [31, 32]. In this article we use a standard gradient-descent method because the latter methods require a substantial number of interactions with the robot which can be especially damaging in case of walking tasks. Since the estimation of policy gradients often results in a high variance, the policy update is often coupled with an explicit estimation of a parametrized value function $V^\pi(x)$. This combination is known as the actor-critic method, where the policy is referred to as the actor, and the value function is referred to as the critic.

The method we use is the standard model-free temporal-difference-based method described in [29]. Since RL is a trial-and-error learning method, the quality of the policy as well as the learning speed depend on the exploration method. Exploration is commonly achieved either by perturbation of the so far optimal action, or by optimistic initialization, or by both. Optimistic initialization is a method of initializing the value function with a value equal to or greater than the maximum possible value of a state. This causes the visited states to become less attractive than the states that have not been visited yet [33]. Optimistic initialization can speed up the learning in the absence of negative rewards. For the parametrization of the policy and value-function we use a linear in parameters tile coding approximator [34].

## 3. Benchmark system

The two-dimensional benchmark example studied in this article is a pendulum attached to a cart [7, 8], which is shown in Figure 2. The system consists of a cart with mass $m_M$ and a pendulum that is attached to the cart's center of mass $C_M$.

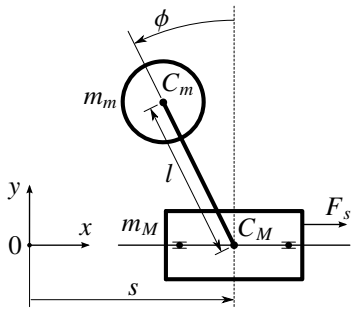


Figure 2: The inverted pendulum on a movable cart.

The pendulum is a point mass $m_m$ attached at the end of a massless rod of length $l$. The system has two degrees of freedom, namely the linear motion of the cart along the $x$-axis, described here by the coordinate $s \in \mathbb{R}$, and the rotary motion of the pendulum with respect to the cart, described by the angle $\phi \in \mathbb{R}$. The only actuation is realized by a horizontal force $F_s \in \mathbb{R}$ acting on the cart body.

The system's state is given as $x = \left[s, \phi, \dot{s}, \dot{\phi}, \right]^T \in \mathbb{R}^4$. Here, $s, \dot{s}$ denote the cart position and velocity, and $\phi, \dot{\phi}$ denote the pendulum's angle and angular velocity, respectively. The control $u = F_s$ is the force acting on the cart body.

In an ideal scenario, both the cart and the pendulum can move without friction along their respective degrees of freedom. For our second and third experiment, we employ uncertainty in the form of viscous friction at the rotary joint, i.e. in the pendulum joint bearing. This produces an internal torque $\tau_\phi = -\kappa\mu\dot{\phi}$ applied to the pendulum, where $\kappa$ is a coefficient that depends on the configuration of the rotary joint, and $\mu$ is a viscous friction coefficient. Depending on whether or not this friction is included in the model, uncertainty in friction can be considered as a parametric or as a structural uncertainty.

More details of the benchmark implementation are given in Appendix A.

## 4. Problem formulation

In this section, we provide formulations of the objective function used in the OC, NMPC and RL problems.

We investigate control scenarios for swing-up motions of the cart-pendulum system from the given initial state $x(0) = [s(0), \phi(0), \dot{s}(0), \dot{\phi}(0)]^T = [0, \pi, 0, 0]^T$, which implies the system starts from rest, with the cart in the origin of the coordinate system and the pendulum pointing downwards. The goal of the task is to swing the pendulum up and to drive the cart back to the origin, i.e. to reach the final state $x(T) = [s(T), \phi(T), \dot{s}(T), \dot{\phi}(T)]^T = [0, 0, 0, 0]^T$. This is realized for both the OC and the NMPC problem by the Lagrange term in the objective function

$$L(x(t), u(t)) = \|x(t) - \bar{x}(t)\|_W^2 + \|u(t)\|_V^2 \tag{4}$$

as defined in (2), where the weights $W = \mathrm{diag}(1, 0.5, 2, 0.2)$ and $V = \mathrm{diag}(0.0005)$ were chosen to scale the state elements to approximately the same range. Set-point $\bar{x} \equiv [0, 0, 0, 0]^T$

is set according to the definition of the task. The benchmark constraints defined in Appendix A can be directly formulated as path constraints (1d) on both states and controls, while the prediction horizon of the NMPC controller is a subinterval of the problem horizon.

The discount rate $\gamma$, which is inevitable for solving a continuing task in RL, affects the obtained RL solution. Therefore, to make the NMPC and RL results comparable, we include the same discount rate value into the objective function of OC and NMPC. The effect of the discount on the NMPC formulation is that it increases the focus on the beginning of the horizon by providing a weighting over time, i.e. the further the event lies in the future of the horizon, the less it will be considered for the computation of the optimal behavior.

To allow solving the control problem in real-time using NMPC alone and together with MHE, the problem formulation has to be adapted for the current algorithmic setup in the optimal control software package MUSCOD-II [35, 36]. Due to the nonlinear behavior of the cart-pendulum system, at least a control rate of 40Hz has to be chosen to generate sufficient contraction in the real-time iteration scheme and to enable the standard structure exploitation for the sequential quadratic programming method. However, this increases the number of shooting nodes and the computational time. Therefore, the horizon was set to 3 s.

In RL, we construct the reward from the same Lagrange term (4), but we additionally add a negative reward and a shaping function $S(x_k, x_{k+1})$:

$$r(x_k, x_{k+1}) = \begin{cases} -1000 & \text{if } x_{k+1} \in X_\mathrm{a}, \\ -L(x_{k+1}, u_k) + S(x_k, x_{k+1}) & \text{otherwise,} \end{cases} \quad (5)$$

where $X_\mathrm{a}$ is a set of absorbing states that lie outside of the cart's position constraints defined in Appendix A.

The shaping function denoted by $S(x_k, x_{k+1})$ leaves the target objective unchanged but allows to reduce steady-state error. Due to the quadratic terms in the definition of $L(x_{k+1}, u_k)$, rewards become small for balancing states where all elements of the state are close to zero, except possibly the cart position $s_k$. This effect has previously been described in [22], where authors noticed that the quadratic reward, the $L^2$-norm, penalized large velocities much more than small steady-state errors. They solved the issue by showing that the absolute value reward, the $L^1$-norm, yielded a response with negligible errors. This solution is not directly applicable here, because our aim is to obtain results as similar to OC as possible, which uses a quadratic cost function. Instead, inspired by [22], we introduce a potential-based shaping function [37] encoded as $S(x_k, x_{k+1}) = \gamma\Psi(x_{k+1}) - \Psi(x_k)$, where $\Psi(x_k) = \psi\|Wx_k\|_1$ is the sum of absolute values of weighted state elements multiplied by a shaping weight $\psi$. The purpose of this shaping function is to provide a stronger guidance towards $x = [0, 0, 0, 0]^T$ in the region of the state space where the quadratic reward function fails to do so. Influence of the shaping function is analyzed in Appendix B.

It is possible to include hard constraints directly into the OC formulation by (1d). However, in the RL formulation, they have to be reformulated as soft constraints, which is done by including them directly in the reward function in the form of a negative reward as in (5). This essentially changes the original optimization problem by introducing a trade-off between receiving positive rewards and avoiding negative ones. For example, once a very large negative reward is received, it will force the system to never violate this constraint again, even at a price of obtaining lower positive rewards. On the contrary, a small negative reward will allow infrequent violation of the constraint, which will slow down learning and may even damage a real-world system. In this benchmark example, the trade-off has a mild effect, because the cart position constraints are rather loose. While constraints are violated a few times in the process of learning, the final result is free of constraint violations.

For the cart-pendulum benchmark, the optimal combination of parameters can be found in Table 1. For NMPC, the tolerances were chosen according to best practices, the horizon length as well as the sampling period were chosen such that "iNMPC" uses the same formulation as "OC" and that "NMPC" computes feedback in less than 5 ms. The discount rate $\gamma$ was chosen according to the RL formulation. For RL, we found the parameters using the exhaustive grid search. It generated tuples of candidate parameters by selecting them from a set of predefined parameter values commonly used in the actor-critic literature, c.f. [29].

For the RL policy and value function approximation, we used tile coding with 16 tilings, each of size $[2.5, 0.1\pi, 2.5, 0.5\pi]^T$. However, pendulum states close to the state $x = [0, 0, 0, 0]^T \in \mathbb{R}^{n_x}$ require a finer resolution of the function approximator. Therefore, before projecting a state on the tiles, we rescale each state element to the interval $[-1, 1]$, and then apply a squashing function with the parameter $\rho = 5$:

$$\Omega(x^j, \rho) = \frac{(1+\rho)x^j}{1+\rho|x^j|}, \quad \forall j \in \{1, \ \ldots, \ n_x\},$$

where $x^j$ denotes a scaled element. This effectively controls resolution by a multiplier that varies continuously, from $(1 + \rho)^{-1}$ in the downward position of the pendulum, to $1 + \rho$ in the balancing state $x = [0, 0, 0, 0]^T$.

## 5. Evaluation protocol

### 5.1. Notations and methodology

As summarized in Figure 1, we use the "OC" notation to denote the optimal solution obtained by off-line optimal control. The cost of this solution serves as a baseline for all subsequent methods. As structural uncertainties, we consider uncertainties that originate from the lack of knowledge about the true physics of the underlying dynamic system. Examples in walking robots might include model-plant mismatch due to uneven floor, friction in joints, softness of the ground, etc. Being unaware of possible uncertainties in a system, the following three *frozen* methods are not explicitly equipped with an ability to adapt to the system:

Table 1: Parameters of OC, NMPC and RL for the cart-pendulum problem. The first group of parameters is relevant to OC and NMPC, where the value in brackets is given for real-time NMPC. The second group of parameters is relevant only to RL.

| Parameter | | Value |
|---|---|---|
| OC/NMPC: | | |
| Horizon length | $T$ | 5 s (3 s) |
| Discount rate | $\gamma$ | 0.99 |
| Sampling period | $T_s^{\mathrm{OC/NMPC}}$ | 0.05 s |
| KKT-Tolerance | | $10^{-7}$ |
| Integration accuracy | | $10^{-6}$ |
| RL: | | |
| Episode length | $T$ | 5 s |
| Discount rate | $\gamma$ | 0.99 |
| Sampling period | $T_s^{\mathrm{RL}}$ | 0.05 s |
| Number of learning episodes | | $2.0 \cdot 10^5$ |
| Additional learning episodes | | 5% |
| Eligibility discount rate | | 0.65 |
| Exploration variance | $\Sigma_u$ | $0.20^2\, u_{\max}^2$ |
| Critic learning rate | $\alpha_{\mathrm{c}}$ | 0.10 |
| Actor learning rate | $\alpha_{\mathrm{a}}$ | 0.01 |

- "iNMPC" denotes an ideal NMPC controller that neglects computational time constraints and returns an optimal control signal immediately.
- "NMPC" denotes an NMPC controller tuned to real-time performance for the specific task.
- "RL" denotes an RL controller that plays the optimal policy $\pi^*$ after having learned on an ideal system.

Neither "iNMPC" nor "NMPC" apply MHE for state and parameter estimation.

As parametric uncertainties, we consider parameters which are included in the dynamic model of the system and whose values are not known *a priori*, but can be inferred from interactions with the real system, i.e. the parameters are observable. Accordingly, by the term *adaptive* we denote methods that have (in the case of NMPC) been explicitly equipped with knowledge about the uncertainties and an algorithm to adapt to them, or that are (in the case of RL) given additional time to interact with the system:

- "iNMPC-adapt" denotes a controller of a combination of both MHE and NMPC. The controller is able to estimate a specified unknown parameter of a model and adjust its control signal accordingly.
- "NMPC-adapt" denotes a combination of both MHE and NMPC tuned to real-time performance for the specific task.
- "RL-adapt" denotes the RL controller that is initialized using the optimal policy $\pi^*$ learned by "RL" on the ideal system. To cope with uncertainties in the system, we allow "RL-adapt" to learn for an additional small number of episodes, c.f. Table 1.

Note that the NMPC approach requires explicit specification of the parameters to be estimated in the model, while RL can cope with them without explicit consideration.

### *5.2. Description of experiment and measures*

For the benchmark problem, we provide a comprehensive comparison of the described methods for an ideal system, and for a system with structural and parametric uncertainties.

First, we investigate whether the three frozen methods produce similar trajectories on our benchmark system. For that we employ the coefficient of determination, $R^2$, as a similarity measure of trajectories. The measure quantifies the deviation of the trajectories obtained by "RL", "iNMPC", and "NMPC" from an optimal trajectory. Denoting a trajectory $\zeta$ as a sequence of states and controls, $\zeta = \{\zeta_k\}$, $0 \le k \le K$ and $\zeta_k = (x_k^T, u_k^T)^T$, we measure similarity between corresponding components by means of $R^2$. Formally, the $R^2$ measure is defined as

$$R^2 = 1 - \frac{\sum_i^K (\zeta_i^\ddagger - \zeta_i^{\mathrm{OC}})^2}{\sum_i^K (\zeta_i^\ddagger - \bar{\zeta})^2}, \quad \bar{\zeta} = \frac{1}{K}\sum_i^K \zeta_i^\ddagger,$$

where ‡ is a wildcard for one of {"RL", "RL-adapt", "iNMPC", "iNMPC-adapt", "NMPC", "NMPC-adapt"}. For the "RL" method, which exhibits variability in trajectories, we compute both the $R^2$ measure of the mean trajectory, and the mean of $R^2$ values obtained across individual trajectories.

Second, after the similarity of the methods is verified, we employ regret as a measure to evaluate the performance of the methods against uncertainty $\Delta\Theta$. This measure is commonly used in evaluation of online machine learning and optimization methods [38, 39]. Regret quantifies the amount of additional cost which is incurred due to suboptimal actions taken by a controller with respect to the optimal control actions. Lower values of regret indicate a controller, whose behavior is closer to the optimal one. Since the optimal controller has zero regret, it becomes convenient to plot regrets of the methods instead of the direct costs.

We compute the regret $R^\ddagger(\zeta; \Delta\Theta)$, defined as the difference between $C^\ddagger(\zeta)$, the total cost of the method denoted by the wildcard ‡, and the baseline $C^{\mathrm{OC}}(\zeta; \Delta\Theta)$, i.e.

$$R^\ddagger(\zeta; \Delta\Theta) = C^\ddagger(\zeta) - C^{\mathrm{OC}}(\zeta; \Delta\Theta),$$

where we use the NMPC cost (2) for all methods directly,

$$C(\zeta) = \sum_{i=0}^{K} \gamma^i L(x_i, u_i) \Delta t_i. \tag{6}$$

By means of $\Delta t_i$, we take into account the different sampling periods of the methods. Note that all of the tested methods are unaware of the true extent of the uncertainty introduced.

In this article, we pursue a generic comparison across methods and benchmarks. Therefore, we do not include specific stability measures, such as Lyapunov stability, orbital stability, or gait sensitivity norm, but rather stick to a general notion of the cost of trajectories.

Due to the stochastic nature of RL, we plot a mean value of the regret averaged over 50 runs. If a run for a given uncertainty $\Delta\Theta$ is prematurely terminated due to the violation of constraints, then the corresponding value is not drawn.

## 6. Results on the cart-pendulum

In order to assess the similarity of the frozen methods, we analyze their performance for the cart-pendulum system without uncertainties. The resulting trajectories and the $R^2$ measure results are shown in Figure 3 and Table 2, respectively. All three methods show a qualitatively similar behavior and are successful in swinging the pendulum up and balancing it there. An overall similarity between "RL" and "iNMPC" of more than 90.3 % was achieved in terms of the $R^2$-measure. Comparing the $R^2$ values of a mean trajectory with the mean of $R^2$ values of the individual trajectories of "RL", we observe that the mean trajectory is closer to "iNMPC", while the individual trajectories exhibit some variability around their mean. The control trajectories $F_s$ of "iNMPC" and "RL" differ in a small time delay and are otherwise comparable. However, after approximately 1.0 s, we find that "RL" demonstrates variability between control trajectories compared to "iNMPC". Due to differences in control actions, the state trajectories start to differ slightly after approximately 0.5 s and recover from that after approximately 2.0 s; only the "RL" cart position $s$ remains to show small steady-state errors. "NMPC" results deviate more from "iNMPC", and an overall similarity of approximately 48.2 % was achieved in terms of the $R^2$-measure.

Next, we show the behavior of both frozen and adaptive methods under the effect of uncertainties. Simulation results against variations of the friction parameter $\mu$ are shown in Figure 4. To indicate the scale of the plot, we employ three filled markers at $\mu = 0\,\mathrm{N\,s\,m^{-2}}$ that correspond to the trajectories in Figure 3. A further reference for interpretation of the scale: If the cart stood still and the pendulum hung down, then the regret of the solution would be equal to 11.73. In the absence of friction, "iNMPC" does not reach zero regret due to numerical approximations. All of "RL", "iNMPC" and "NMPC" show a similar asymptotic behavior in reaction to the variation of the friction coefficient. "iNMPC" shows the lowest regret for low values of viscosity and "NMPC" regret is the largest. Larger values increase the regret, and for the value of $\mu = 0.09\,\mathrm{N\,s\,m^{-2}}$, "RL" yields a lower regret than "iNMPC". All three frozen methods violate the position constraints of the cart. For "RL", this happens at $\mu = 0.12\,\mathrm{N\,s\,m^{-2}}$ and for "iNMPC" and "NMPC" at $\mu = 0.18\,\mathrm{N\,s\,m^{-2}}$ and $\mu = 0.19\,\mathrm{N\,s\,m^{-2}}$, respectively.

The adaptive methods, "RL-adapt", "iNMPC-adapt", and "NMPC-adapt", show a different reaction to the variation of the friction coefficient. Both "iNMPC-adapt" and "NMPC-adapt" show a constant performance under the effect of the variation

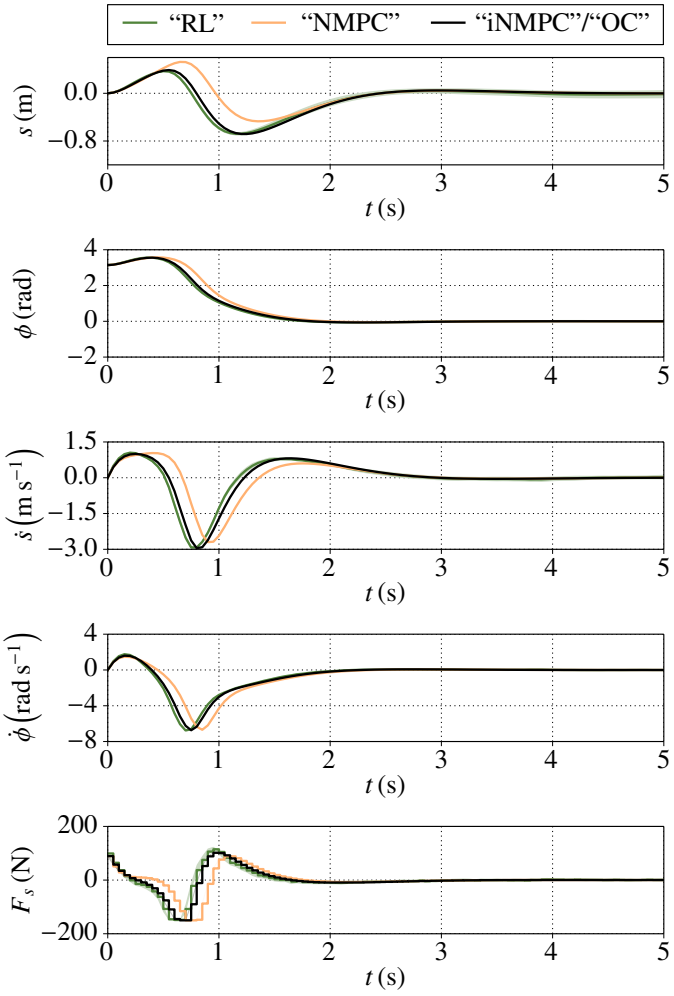


Figure 3: State and control trajectories obtained by the frozen methods for the cart-pendulum system without uncertainties. For "RL" the mean and standard deviation of 50 trajectories is shown. $Y$-axes variables $s, \dot{s}$ denote the cart position and velocity and $\phi, \dot{\phi}$ are the respective quantities of the pendulum. $F_s$ is the force acting on the cart body.

of friction. Note the logarithmic scale; the variances in performance of "iNMPC-adapt" and "NMPC-adapt" are similar, but appear differently due to the logarithmic scale. "RL-adapt" performs better than "NMPC-adapt" for smaller uncertainties of up to $0.07\,\mathrm{N\,s\,m^{-2}}$ and is then outperformed by NMPC. "RL-adapt" regret is an order of magnitude higher than the regret of "iNMPC-adapt", and the RL performance deteriorates with higher friction. For friction coefficients larger than $0.09\,\mathrm{N\,s\,m^{-2}}$, "RL-adapt" shows a much higher variance in regret than for lower friction.

Comparing "RL-adapt" with the frozen method "iNMPC", the graphs show that "iNMPC" cannot compete with RL after the break-even point at $0.04\,\mathrm{N\,s\,m^{-2}}$. This viscous friction coefficient value corresponds to 6.1 % of the difference in energy consumed by the ideal ($0.00\,\mathrm{N\,s\,m^{-2}}$) and disturbed ($0.04\,\mathrm{N\,s\,m^{-2}}$) system. Compared to all three frozen methods, "RL-adapt" performs much better after the break-even point, and the gap grows with larger uncertainties.

A summary of the main results of comparison is presented in

Table 2: Similarity of the cart-pendulum trajectories in terms of the coefficient of determination ($R^2$). For RL we first report similarity of the mean trajectory, and second we report the mean of $R^2$ values.

| Methods | | $s$ | $\phi$ | $\dot{s}$ | $\dot{\phi}$ | $F_s$ |
|---|---|---|---|---|---|---|
| "RL"- "OC", $R^2$ mean trajectory | (%) | 98.1 | 99.8 | 96.4 | 98.4 | 92.9 |
| "RL"- "OC", mean $R^2$ | (%) | 93.9 | 99.8 | 95.3 | 98.2 | 90.3 |
| "NMPC"- "OC" | (%) | 56.6 | 98.6 | 66.7 | 86.9 | 48.2 |
| "iNMPC"- "OC" | (%) | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 |

Table 3.

## 7. Discussion

Our results demonstrate that, with a proper formulation of the optimal control task, it is possible to obtain similar results for the three frozen methods on an ideal system. For the cart-pendulum benchmark example, a good similarity between "RL", "OC" and "iNMPC" was achieved. However, for "NMPC", the expected deviation from the optimal solution due to the tuning towards a real-time feasible controller is observed in Figure 4. We have to stress that this is a problem of the implementation and not of the approach. A speed-up of the implementation by using a multi-level real-time iteration scheme [26, 40], by using a state-of-the-art sequential quadratic programming method tailored for multiple-shooting [41] and replacing the quadratic program solver [42] could address the current time limitations. With a speed-up of the computations, a theoretical coverage of the area between the curves of "iNMPC" and "NMPC" is therefore possible. This will however not influence the already found break-even points, because the asymptotic behavior of the frozen NMPC methods is determined through "iNMPC" as the best possible outcome. Considering this, only an improvement for smaller variations of the friction coefficient is to be expected.

The adaptive methods successfully avoid constraint violations and substantially reduce regret compared to their frozen counterparts over the whole range of viscous friction coefficients. Interestingly, when the coefficient is not present in the system or has low values, "iNMPC-adapt" results in a higher regret than "iNMPC", and the same holds for the fast versions of NMPC. The reason for this is that the combination of NMPC and MHE in the form of "iNMPC-adapt" and "NMPC-adapt" starts with an initial guess of the parameter that is adapted during the actual run of the system. Any mismatch between measurements and values predicted by the model will lead to adaption of the parameters in MHE. This is a crucial difference to RL, which adapts to the uncertainty through multiple trials prior to an assessment run, while NMPC is unaware of the mismatch at first. The cart-pendulum task is very sensitive to changes during movement initiation. Therefore, a wrongly identified parameter in the beginning can already cause substantial differences in terms of regret, which is seen in Figure 4.

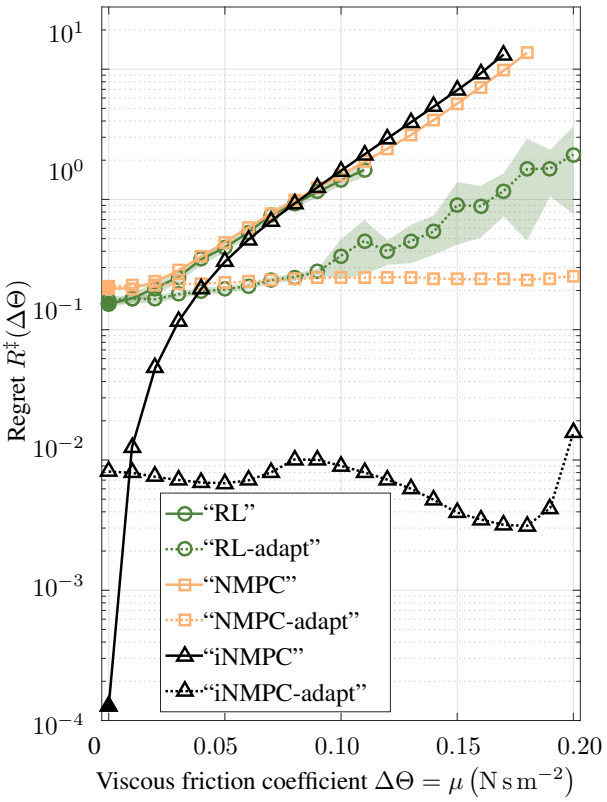


Figure 4: The graph of regrets for the cart-pendulum system that shows the optimal performance of the described methods. The means with the upper and lower 95% confidence limits are shown for controllers with a stochastic component (50 samples per viscous friction coefficient were used).

This effect is amplified for "NMPC-adapt" through the mentioned performance loss due to the real-time feasibility tuning. A speed-up of the computations would lead to a theoretical coverage of the area between the curves of "iNMPC-adapt" and "NMPC-adapt", boosting performance.

In the absence of uncertainties, "iNMPC-adapt" performs superior to both "RL" and "RL-adapt". This does not come as a surprise, as NMPC methods were running off-line, they were using the model of the correct system and, moreover, the uncertain parameter was defined explicitly. However, "iNMPC" outperforms RL methods only for small values of uncertainties. In case of medium and large uncertainties, there exist break-even points after which "RL" and "RL-adapt" obtain lower regret. We remark that the estimation of the difference between ideal and uncertain systems in terms of energy is ad hoc, and more generic measures for model uncertainties should be used in the future.

In the non-ideal setting, performance of NMPC becomes comparable to RL. Nonetheless, one cannot directly report similarity of "NMPC-adapt" and "RL-adapt"; while regret of "NMPC-adapt" is almost constant for the whole range of uncertainties, the regret of "RL-adapt" significantly increases. The

explanation for this effect is twofold. First, for any value of uncertainty, "RL-adapt" was learning for a fixed additional 5 % of time. The larger the value of uncertainty, the more time "RL-adapt" requires to adapt to a new parameter value. This can be supported by the fact of an increasing variance of RL regret, which indicates that the actor-critic algorithm simply did not have enough time to converge in areas of high uncertainties. Second, for large uncertainties, it might be necessary to significantly change the control strategy, i.e. to learn a new policy rather than adapt an ideal one. This will probably require more learning efforts, to first unlearn the initial policy, and then to learn a realistic one.

Several issues were encountered while formulating the benchmark problem with the aim of obtaining identical results. These issues are known to OC and RL communities, but, to the best of our knowledge, they were never explained in the same context before.

OC-related issues:

1. In contrast to RL, the derivative-based methods of optimization used to solve the discretized OC problem require a continuously differentiable formulation of the problem.
2. The performance of the ideal NMPC-MHE combination ("iNMPC-adapt"), for which computational time was neglected, is the order of magnitude better in terms of regret than the corresponding real-time version, mainly caused by a shortened prediction horizon used in the latter.

RL-related issues:

1. In this paper, we use a model-free RL method, which means that transition model of a system is unknown *a* priori.
2. Learning a solution with a quality comparable to OC takes many episodes.
3. Constraints in the original OC problem are included into the RL formulation by means of negative rewards received for violating the constraints.
4. For the benchmark example, the OC objective function has been modified. Formulating a reward function by simply negating the OC objective results in a) a very slow learning in cases when no negative reward is used, b) an inability to learn or even a divergence of the value function if $\gamma = 1$.
5. For symmetrical problems, RL can use state space reduction techniques. For example, for the cart-pendulum example it is possible to wrap the pendulum angle to the $[-\pi, \pi)$ interval, which results in two equally possible optimal trajectories under our objective function. OC generally does not allow implementation of such techniques if they violate the smoothness assumptions.
6. When learning with a quadratic objective function, which is often used in OC, it is useful to implement learning techniques that are able to reduce steady-state error while leaving the objective function unchanged, for example reward shaping.

The presented quantitative comparison is particularly important for our future plans of combining RL and NMPC to control a more complex system with a high number of degrees of freedom. One possible combination could be that RL learns a real model for NMPC, while NMPC provides a backup of an RL exploratory policy. Another scheme could be that RL receives a control signal from NMPC as a suggestion. Initially RL passes this suggestion to the actuators, but at a later stage it takes over in state space areas where it is confident. Independently of the chosen combination strategy, for value function-based RL it is important to retain the Markov property, which may impose restrictions on the NMPC controller as well. For example, such RL methods usually avoid time as a state, hence, the trajectory-tracking NMPC should not be used in the suggestion-based scheme.

## 8. Conclusion

In this article, we provided an extensive comparison of model-free RL and model-based NMPC methods. We began with finding a proper formulation of NMPC and RL problems tackling the same task of a swing-up and balancing motion of a cart-pendulum system. The benchmark is standard and well-known in literature. To facilitate follow-up research, we provide the freely available source code of the benchmark online [43].

We showed that both methods were capable of solving the benchmark problem and that the resulting trajectories for states and controls were similar in terms of the coefficient of determination and regret.

In our experiments considering uncertainties, we showed that ideal NMPC with MHE is superior to RL for the whole range of uncertainties, but the realistic NMPC with MHE is comparable to RL. The major achievement is a quantification of a break-even point after which learning in a model-free setting becomes more beneficial than nonlinear model predictive control with an inaccurate model.

We expect that a proper combination of these methods will open the door to novel control strategies. In particular, we plan to develop a hybrid NMPC-RL controller and test it on a real seven-degree-of-freedom walking robot, specifically designed for the purpose of learning with RL.

## Acknowledgment

I. Koryakovskiy, H. Vallery, R. Babuška, M. Kudruss, C. Kirches, J. P. Schlöder and K. Mombaur were supported by the European project KOROIBOT FP7-ICT-2013-10/611909. W. Caarls was funded from CAPES/BRASIL under project number 88881.030341/2013-01. C. Kirches and M. Kudruss were supported by DFG Graduate School 220 (Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences) funded by the German Excellence Initiative. C. Kirches and J. P. Schlöder were supported by the German Federal Ministry of Education and Research program "Mathematics for Innovations in Industry and Service 2013–2016", grant nº 05M13VHA-GOSSIP.

Table 3: Summary of results.

| Category | Findings |
|---|---|
| Achieved similarity of "iNMPC" and "RL" methods on the ideal system | more than 90.3 % |
| Break-even point: the difference in energy consumed by ideal and noisy systems after which "RL-adapt" performance becomes better then "iNMPC" | 6.1 % |
| Best performing algorithm under parametric uncertainties | "iNMPC-adapt" |
| Best performing algorithm under structural uncertainties | "iNMPC" before the break-even point and "RL-adapt" after the break-even point |

## A. Cart-pendulum benchmark details

By summarizing the positions, velocities and accelerations in $q = [s, \phi]^T$, $\dot{q} = [\dot{s}, \dot{\phi}]^T$ and $\ddot{q} = [\ddot{s}, \ddot{\phi}]^T$, the forward dynamics are given by $\ddot{q} = (H(q))^{-1}(F - C(q, \dot{q}))$, where $H \in \mathbb{R}^{2\times2}$ is the system's mass matrix and $C$ contain Coriolis, centrifugal, and gravitational terms and $F = \left[F_s, \tau_\phi\right]^T \in \mathbb{R}^2$ denotes the actual actuation consisting of the one for the cart and for the pendulum.

We use the *Rigid Body Dynamics Library* [44, 45] for the efficient evaluation of the system's forward dynamics using the Articulated Body Algorithm from [46]. The respective dynamic system in the form of an ordinary differential equation (1b) and initial values (3a) is then retrieved from the forward dynamics.

For simulations, we use the following parameters:

$$m_M = 10.0\,\text{kg}; \quad m_m = 1\,\text{kg}; \quad l = 0.5\,\text{m}; \quad 0\,\text{s} \le t \le 5\,\text{s}.$$

The cart and the pendulum are subject to simple constraints that enforce limits on the cart position and applicable force

$$-2.4\,\text{m} \le s(t) \le 2.4\,\text{m}, \quad -150\,\text{N} \le F_s(t) \le 150\,\text{N}.$$

In the experiment with unknown viscous friction coefficient, the internal torque in the rotary joint of the pendulum is $\tau_\phi = -\kappa\mu\dot{\phi}$. We choose $\kappa = 1\,\text{m}^3$, and vary $\mu$ in the range

$$0.0\,\text{N s m}^{-2} \le \mu \le 0.2\,\text{N s m}^{-2}.$$

## B. Effect of shaping in RL

The results of the effect of the shaping weight $\psi$ on the cost are presented in Figure 5. According to the graph, for a shaping weight of up to 20, the total cost reduces, and then starts increasing again. Note that the total cost is calculated using (6), which does not include the shaping weight $\psi$. The error in position is more volatile, but one may notice that it gets smaller for higher shaping weights. We selected $\psi = 20$, because our primary goal was to reduce the total cost and accept a moderate steady-state error.

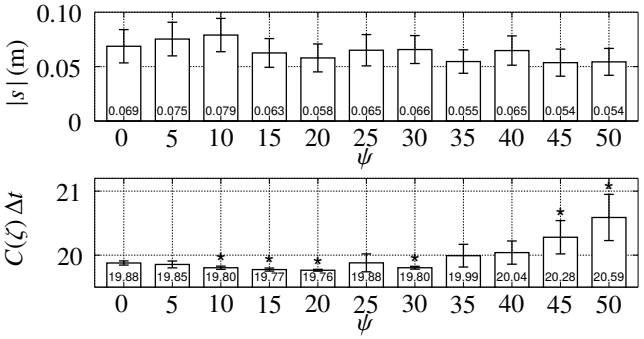


Figure 5: Influence of the shaping function on the results of the "RL" method. Top: absolute error in the cart position at the end of an episode depending on the weight $\psi$ of the shaping function. Bottom: total cost of the trajectory. Shaping is not used for $\psi = 0$. Numbers inside of the bars show the mean value of the error averaged over 50 independent runs, while the error bars show the upper and lower 95% confidence limits. Statistically significant result, for which the p-value is less then 0.05, is marked with $*$ above the bars.

## References


[1] E. Schuitema, Reinforcement Learning on autonomous humanoid robots, Ph.D. thesis, Delft University of Technology, Netherlands (2012).

[2] J. Kober, J. A. D. Bagnell, J. Peters, Reinforcement Learning in Robotics: A Survey, International Journal of Robotics Research.

[3] S. J. Qin, T. A. Badgwell, A survey of industrial model predictive control technology, Control Engineering Practice 11 (7) (2003) 733–764.

[4] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, M. Diehl, Online Walking Motion Generation with Automatic Foot Step Placement, Special Issue: Section Focused on Cutting Edge of Robotics in Japan 24 (5-6) (2010) 719–737. doi:10.1163/016918610X493552.

[5] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, E. Todorov, An integrated system for real-time Model Predictive Control of humanoid robots, in: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids, 2013, pp. 292–299. doi:10.1109/HUMANOIDS.2013.7029990.

[6] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, R. Tedrake, Optimization-based Locomotion Planning, Estimation, and Control Design for the Atlas Humanoid Robot, Autonomous Robots (2015) 1–27.

[7] A. G. Barto, R. S. Sutton, C. W. Anderson, Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems, IEEE Transactions on Systems, Man, and Cybernetics SMC-13 (5) (1983) 834–846. doi:10.1109/TSMC.1983.6313077.

[8] H. Kimura, S. Kobayashi, Stochastic real-valued reinforcement learning to solve a nonlinear control problem, in: IEEE International Conference on Systems, Man, and Cybernetics, Vol. 5, 1999, pp. 510–515.

[9] M. Wisse, Essentials of dynamic walking: Analysis and design of two-legged robots, Ph.D. thesis, Delft University of Technology, Netherlands (2004).

[10] L. T. Biegler, A Survey on Sensitivity-based Nonlinear Model Predictive Control, in: 10th IFAC International Symposium on Dynamics and Control of Process Systems, 2013, pp. 499–510.

[11] A. H. Jazwinski, Stochastic Processes and Filtering Theory, Dover Books on Electrical Engineering Series, Dover Publications, 2007.

[12] K. R. Muske, J. B. Rawlings, J. H. Lee, Receding horizon recursive state estimation, American Control Conference 30 (1993) 900 – 904. doi:10.1109/ACC.1993.4175983.
[13] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, H. G. Bock, A real-time algorithm for moving horizon state and parameter estimation, Computers & Chemical Engineering 35 (1) (2011) 71–83. doi:10.1016/j.compchemeng.2010.07.012.
[14] W. Caarls, E. Schuitema, Parallel Online Temporal Difference Learning for Motor Control, IEEE Transactions on Neural Networks and Learning Systems 27 (7) (2016) 1457–1468. doi:10.1109/TNNLS.2015.2442233.
[15] P. Abbeel, A. Coates, A. Y. Ng, Autonomous Helicopter Aerobatics Through Apprenticeship Learning, International Journal of Robotics Research 29 (13) (2010) 1608–1639.
[16] W. D. Smart, L. P. Kaelbling, Practical Reinforcement Learning in Continuous Spaces, in: Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 903–910.
[17] J. Kober, J. Peters, Policy search for motor primitives in robotics, Machine Learning 84 (1-2) (2011) 171–203.
[18] R. Sutton, A. Barto, R. J. Williams, Reinforcement learning is direct adaptive optimal control, Control Systems, IEEE 12 (2) (1992) 19–22.
[19] D. Ernst, M. Glavic, F. Capitanescu, L. Wehenkel, Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39 (2) (2009) 517–529.
[20] S. Levine, V. Koltun, Guided Policy Search, in: ICML '13: Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 1–9.
[21] T. Zhang, G. Kahn, S. Levine, P. Abbeel, Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search, in: IEEE International Conference on Robotics and Automation (ICRA, 2016, pp. 528–535.
[22] J.-M. Engel, R. Babuska, On-line Reinforcement Learning for Nonlinear Motion Control: Quadratic and Non-Quadratic Reward Functions, in: Proceedings of the 19th IFAC World Congress, Vol. 19, Cape Town, South Africa, 2014, pp. 7043–7048.
[23] H. G. Bock, K. J. Plitt, A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems, in: Proceedings of the 9th IFAC World Congress, Pergamon Press, Budapest, 1984, pp. 242–247.
[24] M. Diehl, Real-Time Optimization for Large Scale Nonlinear Processes, Ph.D. thesis, Heidelberg University, http://www.ub.uni-heidelberg.de/archiv/1659/ (2001).
[25] M. Diehl, H. G. Bock, J. P. Schlöder, A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control, SIAM Journal on Control and Optimization 43 (5) (2005) 1714–1736. doi:10.1137/S0363012902400713.
[26] H. G. Bock, M. Diehl, E. A. Kostina, J. P. Schlöder, Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations, in: L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, B. van Bloemen Waanders (Eds.), Real-Time PDE-Constrained Optimization, SIAM, 2007, Ch. 1, pp. 3–24.
[27] J. V. Frasch, L. Wirsching, S. Sager, H. G. Bock, Mixed–Level Iteration Schemes for Nonlinear Model Predictive Control, IFAC Proceedings Volumes 45 (17) (2012) 138 – 144. doi:http://dx.doi.org/10.3182/20120823-5-NL-3013.00085.
[28] T. Johnson, C. Kirches, A. Wächter, An Active-Set Quadratic Programming Method Based On Sequential Hot-Starts, SIAM Journal on Optimization 25 (2) (2015) 967–994.
[29] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, E. Schuitema, Efficient Model Learning Methods for Actor-Critic Control, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42 (3) (2012) 591–602. doi:10.1109/TSMCB.2011.2170565.
[30] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, M. Lee, Natural Actor-Critic Algorithms, TR09-10, University of Alberta, Canada (Jun. 2009).
[31] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.
[32] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, P. LEcuyer, et al., The Cross-Entropy Method for Optimization, Vol. 31, Elsevier Science, 2013.
[33] L. Matignon, G. Laurent, N. Le Fort-Piat, Reward Function and Initial Values: Better Choices for Accelerated Goal-Directed Reinforcement Learning, in: S. D. Kollias, A. Stafylopatis, W. Duch, E. Oja (Eds.), Artificial Neural Networks ICANN 2006, Vol. 4131 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 840–849.
[34] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.
[35] D. B. Leineweber, I. Bauer, H. G. Bock, J. P. Schlöder, An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: theoretical aspects, Computers & Chemical Engineering 27 (2) (2003) 157–166.
[36] D. B. Leineweber, A. Schäfer, H. G. Bock, J. P. Schlöder, An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization: Part II: Software aspects and applications, Computers & chemical engineering 27 (2) (2003) 167–174.
[37] A. Y. Ng, D. Harada, S. J. Russell, Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping, in: Proceedings of the Sixteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, pp. 278–287.
[38] S. Shalev-Shwartz, Online Learning and Online Convex Optimization, Foundations and Trends in Machine Learning 4 (2) (2012) 107–194. doi:10.1561/2200000018.
[39] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research 4 (1) (1996) 237–285.
[40] C. Kirches, L. Wirsching, H. G. Bock, J. P. Schlöder, Efficient Direct Multiple Shooting for Nonlinear Model Predictive Control on Long Horizons, Journal of Process Control 22 (3) (2012) 540–550.
[41] D. Janka, C. Kirches, S. Sager, A. Wächter, An SR1/BFGS SQP algorithm for nonconvex nonlinear programs with block-diagonal hessian matrix, Mathematical Programming Computationdoi:10.1007/s12532-016-0101-2.
[42] L. Schork, A parametric active set method for general quadratic programming, Master thesis, Heidelberg University (2015).
[43] W. Caarls, Generic Reinforcement Learning Library (2015-2017). URL {https://github.com/wcaarls/grl}
[44] M. L. Felis, RBDL: an Efficient Rigid-Body Dynamics Library using Recursive Algorithms, Autonomous Robots 41 (2) (2017) 495–511. doi:10.1007/s10514-016-9574-0.
[45] M. Felis, Rigid Body Dynamics Library (RBDL (2012-2017). URL {https://bitbucket.org/MartinFelis/rbdl}
[46] R. Featherstone, Rigid Body Dynamics Algorithms, Kluwer international series in engineering and computer science: Robotics, Springer, 2008.