Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 9502–9507

Enforcing Constraints over Learned Policies via Nonlinear MPC: Application to the Pendubot

G. Turrisi, B. Barros Carlos, M. Cefalo, V. Modugno, L. Lanari, G. Oriolo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma, Italy
(e-mail: lastname@diag.uniroma1.it)

Abstract: In recent years Reinforcement Learning (RL) has achieved remarkable results. Nonetheless RL algorithms prove to be unsuccessful in robotics applications where constraints satisfaction is involved, e.g. for safety. In this work we propose a control algorithm that allows to enforce constraints over a learned control policy. Hence we combine Nonlinear Model Predictive Control (NMPC) with control-state trajectories generated from the learned policy at each time step. We prove the effectiveness of our method on the Pendubot, a challenging underactuated robot.

Keywords: Learning control, Optimal control, Robotics, Constraint satisfaction problem, Real time, Model based control

1. INTRODUCTION

In recent years Reinforcement Learning (RL) has proven to be particularly successful in extremely complex application scenarios. From beating professional gamers (Silver et al., 2016; Vinyals et al., 2019) to dexterous in-hand manipulation for solving a Rubik's cube (OpenAI et al., 2018), the RL methods are at the forefront of the artificial intelligence research. These results are possible because in RL the learning process generally does not require any analytical model of the controlled system. The final policy, which can be highly nonlinear, is the result of the optimization of the expected sum of the reward signals gathered by the agent while interacting with the environment, as shown by Sutton and Barto (1998).

Nonetheless, real robotics applications pose many challenges which greatly diverge from the classical RL assumptions (Kober et al., 2013): continuous control and state spaces, need to be data efficient if the learning procedure is performed directly on the robot, and the necessity of state and control constraints which can guarantee, for example, safety conditions.

In the last two decades many efforts have been devoted to improve the effectiveness of RL in robotics scenarios. Policy Search (PS) methods (Deisenroth et al., 2013) represent a first attempt to overcome the issue of directly estimating the optimal cost-to-go for each state, a problem that becomes rapidly intractable for robotics applications. PS approaches optimize directly the expected return of a parametrized policy. More recently, other solutions have been proposed to deal with continuous control-state space such as Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016). Moreover, in the literature it has been shown that PS approaches can effectively cope with the data efficiency issue if a model of the controlled system is learned in combination with a control policy (Deisenroth and Rasmussen, 2011). Chatzilygeroudis et al. (2017) present a state-of-the-art model-based PS method which combines a black-box optimization with a gaussian process surrogate model.

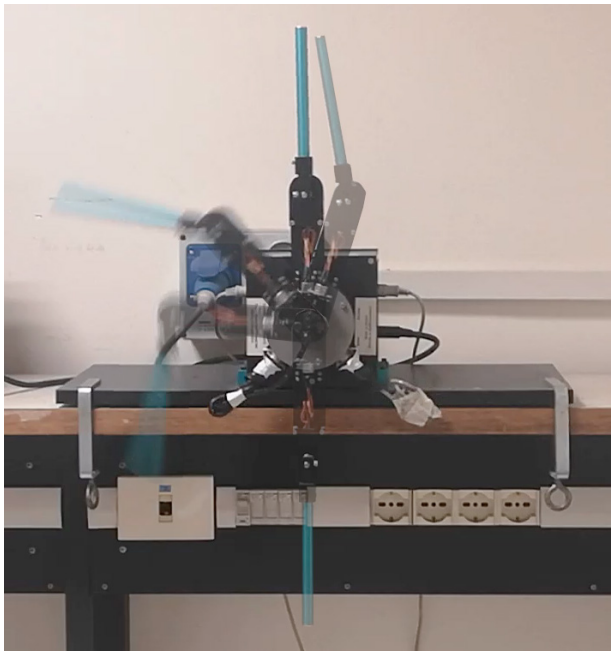

Fig. 1. The Pendubot performing a swing-up maneuver with the proposed approach.

Fewer works consider the problem of constraints satisfaction. In RL this issue is usually addressed by extending the reward with a penalty term. This approach has however several drawbacks and feasibility is not always assured. Achiam et al. (2017) propose an actor-critic approach based on a constrained Markov decision process formulation, which is solved by approximations. Similarly as in Cheng et al. (2019), a penalization term is added to limit constraint violations, thus increasing the safety

10.1016/j.ifacol.2020.12.2426

of the learned policy. Another recent work (Heim et al., 2019) introduces an algorithm to learn the viability kernel directly in the control-state space. Even though it provides an estimation of a safety measure while learning, this procedure is conservative by construction and might converge slowly in a large control-state space.

On the other hand, Model Predictive Control (MPC) is an optimization-based control technique applied to dynamical systems. MPC optimizes an objective function based on a predicted evolution of the system state (Rawlings and Mayne, 2009; Borrelli et al., 2017). The possibility of including input and state constraints in the optimization problem, together with its inherent robustness, make MPC one of the most successful strategies. Even though a large number of algorithms have been developed for efficiently computing a solution, in general, for nonlinear control problems, the use of MPC in real time is typically not straightforward. Usually a simplified prediction model is convenient and, when the final state is not reachable in a small time frame, the knowledge of a reference trajectory for the entire task is required.

Many works aim at achieving better performance through the combination of an MPC with data driven approaches. One of the first attempt to combine RL and MPC has been presented by Zhong et al. (2013), where an offline estimation of the value function is used as a final cost in an unconstrained iterative linear quadratic regulator. Tamar et al. (2017) propose iterative learning of a term in the objective function to incorporate long-term reasoning into the MPC. Other works have applied data driven techniques to learn a predictive model for MPC (Nagabandi et al., 2018; Kamthe and Deisenroth, 2018). More recent contributions towards the combination of MPC and RL are due to Mansard et al. (2018) and Farshidian et al. (2019).

Here we introduce an algorithm for combining MPC and RL that tries to overcome the aforementioned issues. We propose an optimal controller based on state-of-the-art Nonlinear MPC (NMPC) solvers that allows for constraints satisfaction and can be used in real-time on challenging robotic systems. We explore how the continuous interaction between the control policy and the MPC can take care of constraints satisfaction in a real application scenario. The performance of our method is tested both in simulation and experiments on a challenging underactuated robotic system, the Pendubot (see Fig. 1).

This paper is organized as follows. Sec. 2 describes the control architecture, the RL algorithm and the NMPC formulation. The Pendubot case study is developed in Sec. 3, which also presents the simulation and experimental results. Future developments are briefly discussed in Sec. 4.

2. THE CONTROL ALGORITHM

The proposed method consists of two different phases. In the first phase any RL approach can be used to learn an offline policy that solves the desired task. In our case we use Deep Deterministic Policy Gradient (DDPG) from Lillicrap et al. (2016). In the second phase, once a control policy is learned, we can employ this information online to provide a guidance for an MPC. Here we define an NMPC

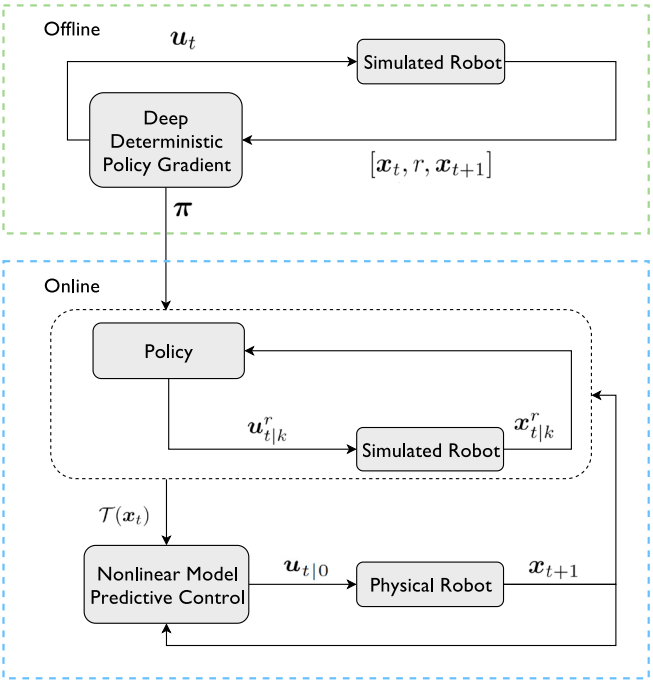

Fig. 2. Block scheme of the proposed approach.

problem and provide an online solution using the Real-Time Iteration scheme (RTI) (Diehl et al., 2005), which is a Sequential Quadratic Programming (SQP) variant. At each time step, starting from the current state, the learned policy is applied to a simulated robot model over a fixed prediction horizon. The resulting control-state trajectory is passed as a reference to the cost function of the NMPC, which allows to transfer the policy learned offline into an optimal control problem to enforce the constraints satisfaction. Afterwards, the resulting optimal control action is applied to the physical robot, which in turn reaches a new state where the procedure is then restarted. A block scheme of the proposed algorithm is shown in Fig. 2.

Our method is capable to find a successful solution under the assumption that the control policy has learned a sequence of actions to accomplish the given task at least in a subset of the feasible region. In practice, an extensive training phase and a large exploration of the control-state space can ensure the validity of the aforementioned hypothesis.

2.1 Offline Policy Learning

Let us consider $X \subseteq \mathbb{R}^{n_x}$ and $U \subseteq \mathbb{R}^{n_u}$ the state and control space of our system. Moreover, we denote with $\boldsymbol{\pi}(\boldsymbol{x}) : X \to U$ the control policy, and with $\boldsymbol{u}_t \in U$, $\boldsymbol{x}_t \in X$ respectively the control action and the system state defined at time t.

DDPG is an actor-critic algorithm where the control policy $\boldsymbol{\pi}(\boldsymbol{x}|\boldsymbol{\theta}^{\boldsymbol{\pi}})$ and the Q-Value function $\boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}|\boldsymbol{\theta}^{\boldsymbol{Q}})) : X \times U \to U$ are both approximated with two neural networks, where $\boldsymbol{\theta}^{\boldsymbol{\pi}} \in \mathbb{R}^{n_{\boldsymbol{\pi}}}$ and $\boldsymbol{\theta}^{\boldsymbol{Q}} \in \mathbb{R}^{n_{\boldsymbol{Q}}}$ are their parameters. The actor $\boldsymbol{\pi}$, given the actual state $\boldsymbol{x}_t$, outputs a control action $\boldsymbol{u}_t$, while the critic $\boldsymbol{Q}$ provides an estimate of the Q-Value function for the pair $(\boldsymbol{u}_t, \boldsymbol{x}_t)$. During the offline training

phase, at each time step, the agent applies an action $\boldsymbol{u}_t = \boldsymbol{\pi}(\boldsymbol{x}_t)$ to the simulated robot and collects a reward r_t from the environment. The generated experience represented by $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1})$ is stacked inside the Replay Buffer (RB), an array that contains a fixed number of elements I. To stabilize the learning procedure, two additional copies of the actor, $\boldsymbol{\pi'}(\boldsymbol{x}|\boldsymbol{\theta}^{\pi'})$ and the critic $\boldsymbol{Q'}(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x})|\boldsymbol{\theta}^{Q'})$ are used in the algorithm.

The DDPG alternates between updating the Q-Value function and the control policy. The learning process is performed at a fixed frequency during the exploration phase of the agent. The update of the parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}^{Q}, \boldsymbol{\theta}^{\pi}, \boldsymbol{\theta}^{Q'}, \boldsymbol{\theta}^{\pi'})$ follows the subsequent steps: first, we randomly retrieve n samples from the RB, where n represents the chosen batch size. For each experience i only the new state $\boldsymbol{x}_{i+1}$ is passed to the actor copy network $\boldsymbol{\pi'}$ which returns the action $\boldsymbol{u}_{i+1}$. The pair $(\boldsymbol{u}_{i+1}, \boldsymbol{x}_{i+1})$ is used as input to $\boldsymbol{Q'}$ that estimates the associated Q-Value. The obtained result is then combined with the sampled experience reward r_i to compute the temporal difference, which is defined as

$$y_i = r_i + \gamma\, \boldsymbol{Q'}(\boldsymbol{x}_{i+1}, \boldsymbol{\pi'}(\boldsymbol{x}_{i+1}|\boldsymbol{\theta}^{\pi'})|\boldsymbol{\theta}^{Q'}),$$

where γ is the discount factor. The final loss L, calculated by subtracting to y_i the $\boldsymbol{Q}$ function computed over the sampled values $(\boldsymbol{u}_i, \boldsymbol{x}_i)$ as

$$L = \frac{1}{n}\sum_i (y_i - \boldsymbol{Q}(\boldsymbol{x}_i, \boldsymbol{u}_i|\boldsymbol{\theta}^{Q})^2),$$

is then used to retrieve the gradient for the critic $\boldsymbol{Q}$. Finally the actor gradient $\boldsymbol{\pi}$ can be approximated as

$$\nabla_{\boldsymbol{\theta}^{\pi}} \boldsymbol{\pi} \approx \frac{1}{n}\sum_i \nabla_{\boldsymbol{u}} \boldsymbol{Q}(\boldsymbol{x}, \boldsymbol{u}|\boldsymbol{\theta}^{Q})|_{\boldsymbol{x}_i, \boldsymbol{\pi}(\boldsymbol{x}_i)} \nabla_{\boldsymbol{\theta}^{\pi}} \boldsymbol{\pi}(\boldsymbol{x}|\boldsymbol{\theta}^{\pi})|_{\boldsymbol{x}_i}.$$

Each gradient is then back-propagated to update the values of the parameters $\boldsymbol{\theta}$.

At the end of the training phase, the control policy $\boldsymbol{\pi}$ is obtained and will be quickly used to generate the reference trajectory for the NMPC. During the offline policy learning phase constraints cannot be explicitly enforced, but their violation can only be penalized trough the reward function.

2.2 NMPC for Online Constraints Satisfaction

NMPC is an advanced control method that uses a nonlinear dynamic model of a system to define a finite-time constrained Optimal Control Problem (OCP) that is solved numerically in an iterative fashion. We define with N the prediction horizon length. Let

$$\mathcal{T}(\boldsymbol{x}_t) = \left\{(\boldsymbol{u}^r_{t|k}, \boldsymbol{x}^r_{t|k}), k = 0, \dots, N\right\}$$

represents the control-state reference trajectory where $\boldsymbol{x}^r_{t|k}$ and $\boldsymbol{u}^r_{t|k} = \boldsymbol{\pi}(\boldsymbol{x}^r_{t|k})$ are the state and control input at $t + k$ generated by forward integrating the simulation robot model with $\boldsymbol{\pi}(\cdot)$ from the initial condition $\boldsymbol{x}_t \in X$. Given $\boldsymbol{x}_{t|k} \in X$, $\boldsymbol{u}_{t|k} \in U$ and $\boldsymbol{s}_{t|k} \in \mathbb{R}^{n_c}$ which denote, respectively, states, controls and slack variables for hard constraints relaxation, the resulting NMPC problem is

$$\begin{aligned}
\min_{\substack{\boldsymbol{u}_{t|0},\dots,\boldsymbol{u}_{t|N-1}\\ \boldsymbol{s}_{t|0},\dots,\boldsymbol{s}_{t|N}}} \quad & \frac{1}{2}\sum_{k=0}^{N-1} ||(\boldsymbol{x}_{t|k} - \boldsymbol{x}^r_{t|k},\ \boldsymbol{u}_{t|k} - \boldsymbol{u}^r_{t|k})||^2_{\boldsymbol{W}} + \\
& \frac{1}{2}||\boldsymbol{x}_{t|N} - \boldsymbol{x}^r_{t|N}||^2_{\boldsymbol{W}_N} + \frac{\rho}{2}\sum_{k=0}^{N} ||\boldsymbol{s}_{t|k}||^2_2 \\
\text{s.t.} \quad & \\
& \boldsymbol{x}_{t|0} - \boldsymbol{x}_t = 0, \\
& \boldsymbol{x}_{t|k+1} - \boldsymbol{f}(\boldsymbol{x}_{t|k}, \boldsymbol{u}_{t|k}) = 0, \qquad k = 0, \dots, N-1, \\
& \boldsymbol{h}(\boldsymbol{x}_{t|k}, \boldsymbol{u}_{t|k}) + \boldsymbol{s}_{t|k} \le 0, \qquad k = 0, \dots, N-1, \\
& \boldsymbol{h}_N(\boldsymbol{x}_{t|k+N}) + \boldsymbol{s}_{t|k+N} \le 0,
\end{aligned}$$

where the nonlinear function $\boldsymbol{f}(\cdot)$ represents the discrete model dynamics, and $\boldsymbol{h}(\cdot), \boldsymbol{h}_N(\cdot) \in \mathbb{R}^{n_c}$ the vector of path and terminal constraints. Finally, the positive-definite matrices $\boldsymbol{W}, \boldsymbol{W}_N > 0$ and the scalar $\rho > 0$ represent the weights terms.

It is important to notice that each time the system reaches a new state $\boldsymbol{x}_{t+1}$, before solving the NMPC problem, an update of the reference control-state trajectory is required. Hence, to generate the new trajectory, the policy $\boldsymbol{\pi}$ learned offline is applied to the simulated robot. The resulting control-state trajectory $\mathcal{T}(\boldsymbol{x}_{t+1})$ is used as a reference in the cost function of the NMPC problem. Updating the reference trajectory plays a central role in the proposed algorithm. Thus, in the event that the current reference trajectory violates the constraints, the NMPC will drive the system away from the current reference trajectory, which will become obsolete.

In real-time applications, the NMPC needs to be solved at every sampling instant under tight runtime requirements. Here we employed the real-time iteration (RTI) scheme. The main idea consists in warm-starting the problem with the previous solution shifted by one sample and performing only one SQP iteration with Gauss-Newton Hessian approximation per each time step. A complete overview of the algorithm can be found in (Diehl et al., 2005).

3. CASE STUDY

We tested the proposed controller on the Pendubot (Fig. 1), an underactuated, planar, robotic system composed by two links and two rotary joints used in education and in research as a benchmark for nonlinear control schemes. The Pendubot moves in a plane. The base link is directly connected (no reduction gears) to a motor while the second link can only be passively driven by the existing dynamic coupling with the first link.

We consider the problem of swinging up and balancing the robot around the up-up equilibrium starting from the stable down-down equilibrium, while satisfying the imposed velocity constraints. We define a coordinate system where the desired final state is $\boldsymbol{x}_g = (q_1\ q_2\ \dot{q}_1\ \dot{q}_2)^T = (0\ 0\ 0\ 0)^T$, with q_i and $\dot{q}_i$ respectively the position and the velocity of the i-th joint (Fig. 3). The joint position q_1 is measured with respect to the upwards vertical axis, while the angle q_2 is defined relatively to the first joint.

In the following we will describe the mathematical model of the system, and the setup of the learning and control

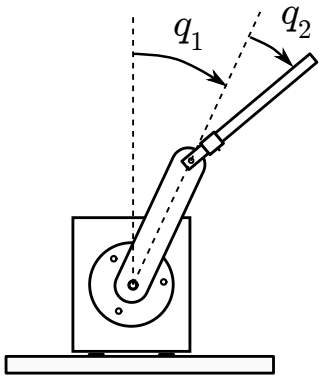

Fig. 3. Schematic of the Pendubot coordinate system.

phases. Moreover, we will show the behaviour of the Pendubot under two different constraint settings, either in simulation and on the real platform. Finally, we will show that the continuous update of the reference trajectory is critical to successfully achieve the desired task.

3.1 The Simulated Robot Model

Given the dynamics formulation of the system

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{u},$$

where $\boldsymbol{q} = (q_1\ q_2)^T$ and $\dot{\boldsymbol{q}} = (\dot{q}_1\ \dot{q}_2)^T$, the Pendubot model can be written as

$$\boldsymbol{B}(\boldsymbol{q}) = \begin{pmatrix} A_1 + 2A_3\cos q_2 & A_2 + A_3\cos q_2 \\ \text{symmetric} & A_2 \end{pmatrix},$$

$$\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{pmatrix} -A_3\sin q_2\dot{q}_2\dot{q}_1 - A_3\sin q_2(\dot{q}_1 + \dot{q}_2)\dot{q}_2 \\ A_3\sin q_2 \dot{q_1}^2 \end{pmatrix},$$

$$\boldsymbol{g}(\boldsymbol{q}) = \begin{pmatrix} A_4 g\sin q_1 + A_5 g\sin(q_1 + q_2) \\ A_5 g\sin(q_1 + q_2) \end{pmatrix}, \quad \boldsymbol{u} = \begin{pmatrix} \tau \\ 0 \end{pmatrix},$$

where g represents the gravity and τ the torque input. The A_i coefficients can be written as

$$A_1 = I_1 + I_2 + m_1 a_1^2 + m_2(l_1^2 + a_2^2), \quad A_2 = I_2 + m_2 a_2^2,$$
$$A_3 = m_2 l_1 a_2, \quad A_4 = m_1 a_1 + m_2 l_1, \quad A_5 = m_2 a_2,$$

where I_i and m_i are, respectively, the moment of inertia and the mass of the i-th link, l_i represents the length of i-th link and a_i is the distance of the center of mass of the i-th link from the center of the i-th joint. Moreover, the physical parameter values for the Pendubot are listed in Table 1.

Table 1. Pendubot physical parameters.

	l_i (m)	m_i (kg)	I_i (kg·m^2)	a_i (m)
joint 1	0.1492	0.193	0.0004	0.1032
joint 2	0.1905	0.073	0.0002	0.1065

3.2 Offline Learning Phase

The learning of the control policy $\boldsymbol{\pi}$ is performed offline using the simulated robot model described in Sec. 3.1. In order to train $\boldsymbol{\pi}$, a reward function must be defined. The function used here penalizes the distance between the current state and the desired one $\boldsymbol{x}_g = (0\ 0\ 0\ 0)^T$. Thus, we define the reward $r(\boldsymbol{x})$ as

$$r(\boldsymbol{x}) = -r_1(\boldsymbol{q}) - \alpha_v r_2(\dot{\boldsymbol{q}}) - \alpha_\tau|\tau|, \tag{2}$$

where α_v, α_τ are weighting parameters, and $r_1(\boldsymbol{q}) = |q_1| + |q_2|$. Moreover, as it is desired to have an upper limit on the joint velocities, both in experiments and in simulations, the quantity $r_2(\dot{\boldsymbol{q}}) = |\dot{q}_1| + |\dot{q}_2|$ is introduced in eq. (2). Finally, the term $-\alpha_\tau|\tau|$ is used to minimize the control effort.

3.3 Online Control Phase

We show that our approach can enforce constraints over an offline learned control policy while still successfully performing the task. The Pendubot system is characterized by an highly nonlinear and fast dynamics. Therefore an high frequency controller is necessary to reach the desired goal. The proposed framework runs in real-time with a control frequency of 500 Hz. To solve the NMPC for both the simulations and the experiments we used the `ACADO Toolkit` (Houska et al., 2011), with a fixed prediction horizon of $N = 10$. The simulated robot model defined in Sec. 3.1 is used both within the NMPC optimization problem and for generating the desired reference trajectory with the policy $\boldsymbol{\pi}$ at each time step.

The swing-up of the Pendubot can be performed either rapidly or with an energy pumping maneuver, reaching the desired state $\boldsymbol{x}_g$ through an oscillatory motion. In both cases, when the state is close to $\boldsymbol{x}_g$, a local controller, usually based on a linearization of the system around the final equilibrium point, can be used for the balancing phase. To this end we implemented a Linear Quadratic Regulator (LQR) for the experiment on the real platform.

Since the robot is only equipped with encoders, angular velocities are numerically derived from position measures.

3.4 Simulation and Experimental Results

In this section we show the results obtained in simulation and on the real robot, under two different constraint settings. The control policy, learned as explained in Sect. 2.1, is able to solve the swing-up including the balancing phase. The maximum velocities reached in simulation during the motion are 9.2 rad/sec and 9.8 rad/sec for the first and second joint, respectively. For clarity, in Fig. 4, Fig. 5 and Fig. 6, we represent with blue lines the solution obtained with the NMPC, and with the red dashed lines the state trajectory $\boldsymbol{q}_i^{\boldsymbol{\pi}}$ obtained using only the control policy $\boldsymbol{\pi}$.

In the first setting, we impose a symmetric velocity constraint $\dot{q}_{1,\mathrm{b}} = 7.2$ rad/sec on the first joint and a maximum admissible torque $\tau_{1,\mathrm{b}} = 0.4$ N · m. We compare our approach with an NMPC where the reference trajectory obtained by the control policy is never updated. In the first row of Fig. 4 we show the results obtained with our approach. Due to the imposed constraints, the proposed controller steers the system to new feasible states, forcing the policy to recalculate a different trajectory, while successfully reaching the up-up configuration. On the other hand, if an NMPC with a fixed reference trajectory is used, the robot does not reach the desired final configuration as shown in Fig. 4 (bottom). It is therefore clear from these results, that recomputing the control-state trajectory at each time step is crucial to successfully complete the desired task.

In the next simulation, we define a second constraints setting, with $\dot{q}_{1,\mathrm{b}} = 7.8$ rad/sec, $\tau_{1,\mathrm{b}} = 0.4$ N · m and

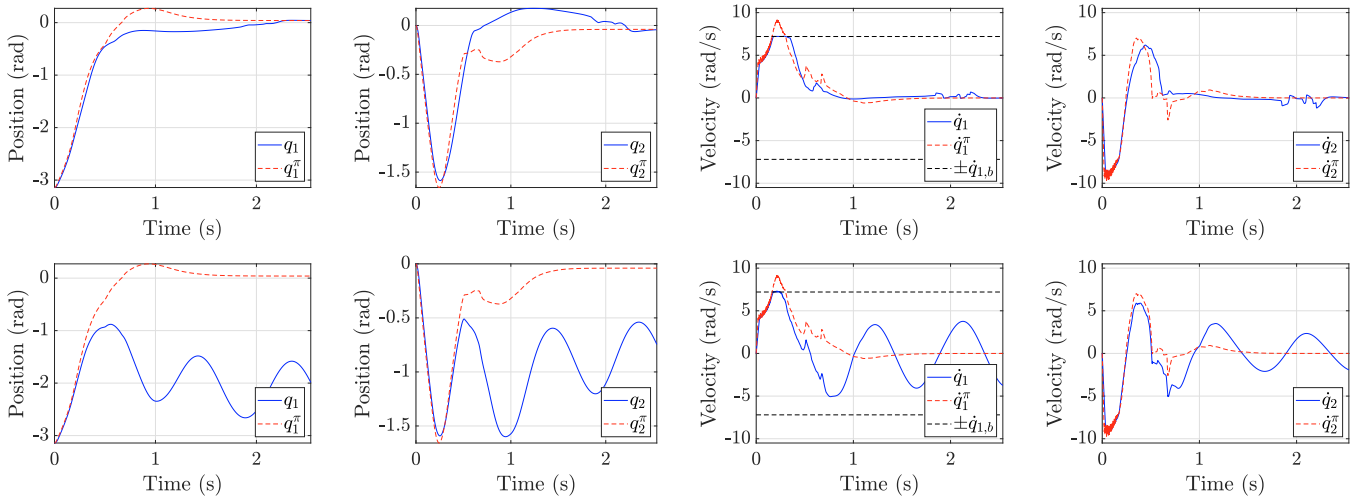

Fig. 4. Simulation results obtained with the proposed approach (top) and with an NMPC that uses a fixed reference trajectory (bottom) by imposing the constraint $\dot{q}_{1,\mathrm{b}} = 7.2$ rad/sec.

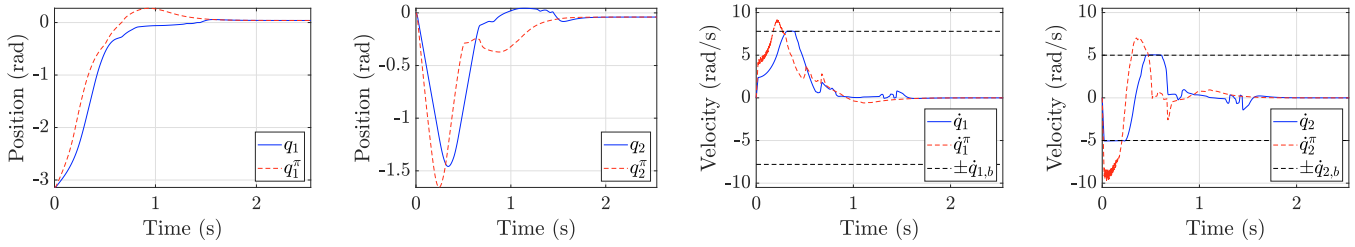

Fig. 5. Simulation result obtained with the constraints $\dot{q}_{1,\mathrm{b}} = 7.8$ rad/sec, $\dot{q}_{2,\mathrm{b}} = 5$ rad/sec.

$\dot{q}_{2,\mathrm{b}} = 5$ rad/sec for the angular velocity of the second joint. As shown in Fig. 5, our controller is still able to perform the desired task. The maximum admissible velocity for the second joint is quickly reached, and the final motion of the system has a larger change with respect to the solution obtained with the first constraints setting.

Finally, we validated our approach on the real Pendubot system using the second constraints setting. Controlling the real robot involves additional challenges due to model uncertainties. Fig. 6 shows that our method is able to successfully perform the swing-up maneuver despite the velocity constraints. In this experiment a final LQR is used to stabilize the system around the equilibrium point, and the use of soft constraints in the NMPC formulation becomes necessary to avoid infeasibility due to model inaccuracies. Indeed, the introduction of slack variables determines small constraint violations of 0.1 rad/s and 1 rad/s, respectively on the first and second joint velocities (see Fig. 6). Thus, the oscillatory behaviour, that is visible on the velocity of the second joint (Fig. 6), depends on the effect of the cost minimization associated to the constraints violation. Model and parameters uncertainties, due to mechanical wear and time varying dynamics (like the dynamic friction induced by the sliding contacts for the encoder of link 2), explain the difference between the results obtained in simulation and on the real robot.

A video with the experimental results is available at `https://youtu.be/5KATfbDwKlI`.

4. CONCLUSIONS AND FUTURE WORK

In this work we present an approach for imposing constraints to a learned control policy. We test the proposed controller both in simulation and on a real Pendubot robot, showing that the continuous interplay between the NMPC and the learned policy is at the base of the constraints enforcement.

In the future, we plan to extensively study the NMPC steering capabilities. Not every constraint can be arbitrarily imposed in the optimization problem, since no assumption is made over the policy capability $\boldsymbol{\pi}$ to generate a satisfactory trajectory in the feasible region. A possible solution will be to automatically retrieve an index that measures the ability of the policy to accomplish the given task from any state, relaxing the constraints when it is needed using the slack variables. A second extension will be the improvement of the robustness over model uncertainties. Multiple policies can be learned in simulation with different dynamical parameters, and the choice of the appropriate $\boldsymbol{\pi}$ can be recomputed online within the NMPC formulation.

REFERENCES

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In *34th International Conference on Machine Learning*, volume 70, 22–31.

Borrelli, F., Bemporad, A., and Morari, M. (2017). *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 1st edition.

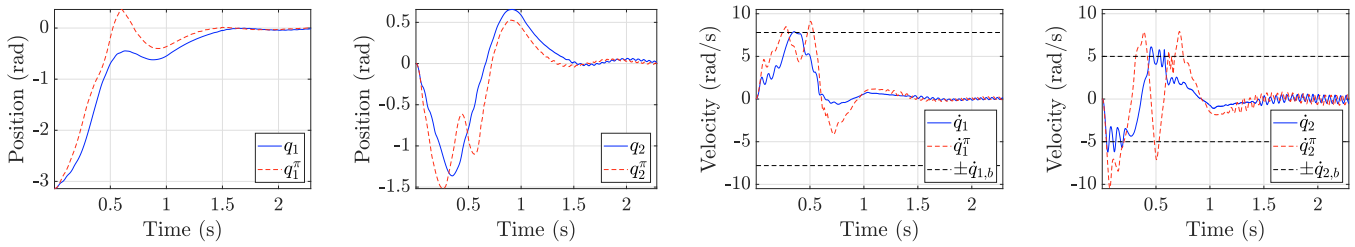

Fig. 6. Experimental result obtained with the constraints $\dot{q}_{1,\mathrm{b}} = 7.8$ rad/sec, $\dot{q}_{2,\mathrm{b}} = 5$ rad/sec.

Chatzilygeroudis, K.I., Rama, R., Kaushik, R., Goepp, D., Vassiliades, V., and Mouret, J. (2017). Black-box data-efficient policy search for robotics. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 51–58.

Cheng, R., Orosz, G., Murray, R.M., and Burdick, J.W. (2019). End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *AAAI-19 Conference on Artificial Intelligence*, volume 33, 3387–3395.

Deisenroth, M. and Rasmussen, C.E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *28th International Conference on Machine Learning*, 465–472.

Deisenroth, M.P., Neumann, G., and Peters, J. (2013). A survey on policy search for robotics. *Foundation and Trends in Robotics*, 2, 1–142.

Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5), 1714–1736.

Farshidian, F., Hoeller, D., and Hutter, M. (2019). Deep value model predictive control. In *3rd Annual Conference on Robot Learning*.

Heim, S., Rohr, A., Trimpe, S., and Badri-Spröwitz, A. (2019). A learnable safety measure. In *3rd Annual Conference on Robot Learning*.

Houska, B., Ferreau, H.J., and Diehl, M. (2011). ACADO toolkit - an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3), 298–312.

Kamthe, S. and Deisenroth, M.P. (2018). Data-efficient reinforcement learning with probabilistic model predictive control. In *21st International Conference on Artificial Intelligence and Statistics*, 1701–1710.

Kober, J., Bagnell, J.A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11), 1238–1274.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations*.

Mansard, N., Del Prete, A., Geisert, M., Tonneau, S., and Stasse, O. (2018). Using a memory of motion to efficiently warm-start a nonlinear predictive controller. In *2018 IEEE International Conference on Robotics and Automation*, 2986–2993.

Nagabandi, A., Kahn, G., Fearing, R.S., and Levine, S. (2018). Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation*, 7559–7566.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2018). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39, 3–20.

Rawlings, J.B. and Mayne, D.Q. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Pub.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529, 484–489.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Tamar, A., Thomas, G., Zhang, T., Levine, S., and Abbeel, P. (2017). Learning from the hindsight plan - episodic MPC improvement. In *2017 IEEE International Conference on Robotics and Automation*, 336–343.

Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., and Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575, 350–354.

Zhong, M., Johnson, M., Tassa, Y., Erez, T., and Todorov, E. (2013). Value function approximation and model predictive control. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 100–107.