Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 4920–4928

Relations between Model Predictive Control and Reinforcement Learning

Daniel Görges

Juniorprofessorship for Electromobility, University of Kaiserslautern, Erwin-Schrödinger-Straße 12, 67663 Kaiserslautern, Germany
E-mail: goerges@eit.uni-kl.de

Abstract: In this paper relations between model predictive control and reinforcement learning are studied for discrete-time linear time-invariant systems with state and input constraints and a quadratic value function. The principles of model predictive control and reinforcement learning are reviewed in a tutorial manner. From model predictive control theory it is inferred that the optimal value function is piecewise quadratic on polyhedra and that the optimal policy is piecewise affine on polyhedra. Various ideas for exploiting the knowledge on the structure and the properties of the optimal value function and the optimal policy in reinforcement learning theory and practice are presented. The ideas can be used for deriving stability and feasibility criteria and for accelerating the learning process which can facilitate reinforcement learning for systems with high order, fast dynamics, and strict safety requirements.

Keywords: Model predictive control, multi-parametric programming, reinforcement learning, approximate dynamic programming, actor-critic structure

1. INTRODUCTION

Model predictive control (MPC) (Mayne et al., 2000; Rawlings and Mayne, 2009; Borrelli et al., in press) and reinforcement learning (RL) (Lewis and Vrabie, 2009; Wang et al., 2009; Lewis et al., 2012; Buşoniu et al., 2010; Zhang et al., 2012) are major methods for optimal control of dynamic systems. Model predictive control has been developed in the control systems community while reinforcement learning has been promoted in the computational intelligence community. The methods have therefore evolved mostly independently (Lewis and Vrabie, 2009).

Model predictive control and reinforcement learning have, however, many similarities. The similarities have been revealed first for the linear quadratic regulator (LQR) (Bradtke, 1992; Bradtke et al., 1994; ten Hagen and Kröse, 1998; Landelius, 1997; Peters and Schaal, 2008; Al-Tamimi et al., 2008; Lewis and Vrabie, 2009; Lewis and Vamvoudakis, 2011; Lewis et al., 2012) and later for model predictive control (Bertsekas, 2005; Ernst et al., 2009). In (Bradtke, 1992; Bradtke et al., 1994) LQR based on Q-learning has been studied and convergence has been proved. In (ten Hagen and Kröse, 1998) a comparison between LQR based on Q-learning and LQR based on system identification has been given. In (Landelius, 1997) LQR based on different actor-critic structures has been investigated and convergence has been shown. In (Al-Tamimi et al., 2008; Lewis and Vrabie, 2009; Lewis and Vamvoudakis, 2011; Lewis et al., 2012) previous results have been further generalized. In (Bertsekas, 2005; Ernst et al., 2009) the relations between MPC and RL have been explored formally (Bertsekas, 2005) and by simulations (Ernst et al., 2009). All these works have made significant contributions. Several relations are, however, still undiscovered, particularly for systems with constraints.

Model predictive control and reinforcement learning have also differences. The properties of model predictive control and reinforcement learning are compared in Table 1.

Model predictive control is model-based, is not adaptive, and has a high online complexity, but also has a mature stability, feasibility and robustness theory as well as an inherent constraint handling. In recent years adaptive model predictive control has been studied for providing adaptivity (Fukushima et al., 2007; Adetola et al., 2009; Adetola and Guay, 2011; Aswani et al., 2013; Heirung et al., 2013; Rathouskỳ and Havlena, 2013; Chowdhary et al., 2013; Benosman et al., 2014; Marafioti et al., 2014; Weiss and Di Cairano, 2014; Di Cairano, 2015a,b; Hernandez and Trodden, 2015; Benosman et al., 2016; Benosman and Farahmand, 2016). The approaches are generally based on robustness and therefore tend to be conservative. Explicit model predictive control (Bemporad et al., 2002; Grieder et al., 2004; Johansen, 2004; Alessio and Bemporad, 2009; Borrelli et al., in press) and neural network-based model predictive control (Parisini and Zoppoli, 1995; Cavagnari et al., 1999; Åkesson and Toivonen, 2006) have been proposed for reducing the online complexity. These methods can usually only be used for low-order systems.

Reinforcement learning is model-free, is adaptive, and has a low online complexity, but also has an immature stability, feasibility and robustness theory as well as a difficult constraint handling. In recent years stability and robustness have been intensely studied. Important results are given in (Balakrishnan et al., 2008; Liu et al., 2012; Sokolov and Kozma, 2013; Sokolov et al., 2015) and (Al-Tamimi et al., 2007; Jiang and Jiang, 2014). Note, however, that the notion of stability is different in the control systems and computational intelligence community. The control systems community concentrates on stability of the closed-

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2017.08.747

Property	Model Predictive Control	Reinforcement Learning
Model	required ✗	not required ✓
Convexity	required (usually) ✗	not required ✓
Adaptivity	immature (usually based on robustness) ✗	mature (inherent) ✓
Online Complexity	high (except explicit and neural MPC) ✗	low ✓
Offline Complexity	low (except explicit and neural MPC) ✓	high ✗
Stability Theory	mature (e.g. based on terminal cost) ✓	immature ✗
Feasibility Theory	mature (e.g. based on terminal constraints) ✓	immature ✗
Robustness Theory	mature (e.g. based on tubes or ISS) ✓	immature ✗
Constraint Handling	mature (inherent) ✓	immature (except input constraints) ✗

Table 1. Properties of Model Predictive Control and Reinforcement Learning

loop system while the computational intelligence community focuses on stability, or more precisely convergence, of the learning algorithm. Stability of the closed-loop system is essentially addressed with certainty equivalence in the computational intelligence community. From this perspective a unified stability and robustness theory is not available yet. Constraint handling has been addressed indirectly by introducing penalties for constraint violations in the cost function (Ernst et al., 2009; Riedmiller, 2012) or directly (He and Jagannathan, 2007; Zhang et al., 2009). Direct constraint handling has, to the best of the author's knowledge, only been investigated for input constraints. Feasibility which has been intensely investigated for model predictive control has not been studied for reinforcement learning yet. Here a substantial research gap exists. The offline complexity has been tackled by exploiting knowledge on the policy, the value function or the dynamics (Buşoniu et al., 2010, Sections 3.7.3 and 5.4).

The properties of model predictive control and reinforcement learning are clearly complementary. Cross-fertilizing model predictive control and reinforcement learning seems therefore very promising, but has not been addressed in the literature yet, although relations have been recognized already in the early days of reinforcement learning (Sutton et al., 1992). In this paper ideas for activating synergies between model predictive control and reinforcement learning for discrete-time linear time-invariant systems with state and input constraints and a quadratic cost function exploiting knowledge about the structure of the policy and value function are presented. The paper particularly provides insights into the relations between model predictive control and reinforcement learning which can be a basis for further research. The paper should furthermore have a tutorial value for readers both from the control systems and computational intelligence community.

Reinforcement learning has traditionally been formulated for systems with discrete states and inputs (Sutton and Barto, 1998). Approximate dynamic programming (ADP) which can be considered as a variant of reinforcement learning has been devised for systems with continuous states and inputs (Werbos, 1990; Lewis and Vrabie, 2009; Wang et al., 2009; Lewis et al., 2012; Buşoniu et al., 2010; Zhang et al., 2012). The discussion in this paper follows the rationale of approximate dynamic programming along the lines of (Lewis and Vrabie, 2009; Si and Wang, 2001; Liu et al., 2012; Sokolov and Kozma, 2013; Sokolov et al., 2015).

The paper is organized as follows. The optimal control problem is introduced in Section 2. Model predictive control and reinforcement learning for solving the optimal control problem are reviewed in Sections 3 and 4. The synergies between model predictive control and reinforcement learning are discussed in Section 5. Conclusions are finally given in Section 6.

Throughout the paper notation from the control systems community is used. Correspondence to the notation from the computational intelligence community is, however, generally provided. Furthermore the following notation is used: Vectors and matrices are indicated by bold symbols, scalars by non-bold symbols. $\boldsymbol{I}$ denotes the identity matrix, $\boldsymbol{0}$ the zero matrix.

2. OPTIMAL CONTROL PROBLEM

2.1 System Model and Cost Function

Consider the discrete-time linear time-invariant system

$$\boldsymbol{x}(t+1) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u} \in \mathbb{R}^m$ the input vector, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ the system matrix, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ the input matrix, $t \in \mathbb{N}_0$ the discrete time, and the pair $(\boldsymbol{A}, \boldsymbol{B})$ is assumed stabilizable.

Let $\boldsymbol{x}(t)$ be the state vector measured at discrete time t and $\boldsymbol{x}_{t+k}$ be the state vector predicted at discrete time $t+k$ using state equation (1) with initial condition $\boldsymbol{x}_t = \boldsymbol{x}(t)$.

Consider further the constraints

$$\boldsymbol{x}(t) \in \mathbb{X} \subseteq \mathbb{R}^n, \quad \boldsymbol{u}(t) \in \mathbb{U} \subseteq \mathbb{R}^m \tag{2}$$

where $\mathbb{X}$ and $\mathbb{U}$ are polyhedral sets containing the origin in their interiors.

Consider finally the cost function

$$V_N(\boldsymbol{x}(t)) = r_\mathrm{T}(\boldsymbol{x}_{t+N}, \boldsymbol{u}_{t+N}) + \sum_{k=0}^{N-1} \gamma^k r(\boldsymbol{x}_{t+k}, \boldsymbol{u}_{t+k}) \tag{3}$$

with the stage cost

$$r(\boldsymbol{x}_{t+k}, \boldsymbol{u}_{t+k}) = \boldsymbol{x}_{t+k}^T \boldsymbol{Q} \boldsymbol{x}_{t+k} + \boldsymbol{u}_{t+k}^T \boldsymbol{R} \boldsymbol{u}_{t+k} \tag{4}$$

and the terminal cost

$$r_\mathrm{T}(\boldsymbol{x}_{t+N}, \boldsymbol{u}_{t+N}) = \boldsymbol{x}_{t+N}^T \boldsymbol{P} \boldsymbol{x}_{t+N} \tag{5}$$

where N is the prediction horizon, $\gamma \in (0, 1]$ a discount factor, $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ the state weighting matrix, $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ the input weighting matrix, and $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ the terminal weighting matrix. The weighting matrices are assumed symmetric and positive definite and the pair $(\boldsymbol{Q}^{1/2}, \boldsymbol{A})$ is assumed detectable. The terminal weighting matrix $\boldsymbol{P}$ is chosen as the solution of the algebraic Riccati equation

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^T \boldsymbol{P} (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) - \boldsymbol{P} + \boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K} = \boldsymbol{0} \tag{6}$$

with

$$\boldsymbol{K} = -(\boldsymbol{R} + \boldsymbol{B}^T\boldsymbol{P}\boldsymbol{B})^{-1}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{A}. \tag{7}$$

Note that the cost function is often denoted as value function.

2.2 Finite-Horizon Optimal Control Problem (FHOCP)

Consider the finite-horizon optimal control problem

$$V_N^*(\boldsymbol{x}(t)) = \min_{\boldsymbol{U}(t)} r_\mathrm{T}(\boldsymbol{x}_{t+N}, \boldsymbol{u}_{t+N}) + \sum_{k=0}^{N-1} \gamma^k r(\boldsymbol{x}_{t+k}, \boldsymbol{u}_{t+k}) \tag{8a}$$

subject to

$$\boldsymbol{x}_{t+k+1} = \boldsymbol{A}\boldsymbol{x}_{t+k} + \boldsymbol{B}\boldsymbol{u}_{t+k}, \quad k = 0, \dots, N-1 \tag{8b}$$
$$\boldsymbol{x}_{t+k} \in \mathbb{X}, \quad k = 1, \dots, N \tag{8c}$$
$$\boldsymbol{u}_{t+k} \in \mathbb{U}, \quad k = 0, \dots, N-1 \tag{8d}$$
$$\boldsymbol{x}_t = \boldsymbol{x}(t) \tag{8e}$$

with input sequence $\boldsymbol{U}(t) = (\boldsymbol{u}_t^T \ \cdots \ \boldsymbol{u}_{t+N-1}^T)^T \in \mathbb{R}^{Nm}$.

2.3 Infinite-Horizon Optimal Control Problem (IHOCP)

Consider infinite-horizon optimal control problem

$$V_\infty^*(\boldsymbol{x}(t)) = \min_{\boldsymbol{U}(t)} \sum_{k=0}^{\infty} \gamma^k r(\boldsymbol{x}_{t+k}, \boldsymbol{u}_{t+k}) \tag{9a}$$

subject to

$$\boldsymbol{x}_{t+k+1} = \boldsymbol{A}\boldsymbol{x}_{t+k} + \boldsymbol{B}\boldsymbol{u}_{t+k}, \quad k = 0, 1, \dots \tag{9b}$$
$$\boldsymbol{x}_{t+k} \in \mathbb{X}, \quad k = 1, 2, \dots \tag{9c}$$
$$\boldsymbol{u}_{t+k} \in \mathbb{U}, \quad k = 0, 1, \dots \tag{9d}$$
$$\boldsymbol{x}_t = \boldsymbol{x}(t) \tag{9e}$$

with input sequence $\boldsymbol{U}(t) = (\boldsymbol{u}_t^T \ \boldsymbol{u}_{t+1}^T \ \cdots)^T$. Note that the terminal cost is not relevant in the IHOCP.

3. MODEL PREDICTIVE CONTROL

3.1 Receding Horizon Principle

Model predictive control relies on solving the FHOCP (8) or the IHOCP (9) for the measured state vector $\boldsymbol{x}(t)$ to obtain the optimal input sequence $\boldsymbol{U}^*(t)$ and applying the first element of the optimal input sequence $\boldsymbol{u}^*(t) = (\boldsymbol{I} \ \ \boldsymbol{0} \ \ \cdots \ \ \boldsymbol{0})\boldsymbol{U}^*(t) = \boldsymbol{u}_t^*$ to the system (1). This is repeated at each discrete time t with a receding prediction horizon. The concept is therefore denoted as the receding horizon principle. Model predictive control is consequently also denoted as receding horizon control (RHC).

In the following sections the solution of the FHOCP (8) and the IHOCP (9) is reviewed. Note that these results apply for the discount factor $\gamma = 1$.

3.2 Solution of the FHOCP

The FHOCP (8) can be rewritten as

$$V_N^*(\boldsymbol{x}(t)) = \frac{1}{2}\boldsymbol{x}^T(t)\boldsymbol{Y}\boldsymbol{x}(t) + \min_{\boldsymbol{U}} \frac{1}{2}\boldsymbol{U}^T\boldsymbol{H}\boldsymbol{U} + \boldsymbol{x}^T(t)\boldsymbol{F}\boldsymbol{U} \tag{10a}$$

subject to

$$\boldsymbol{G}\boldsymbol{U} \leq \boldsymbol{W} + \boldsymbol{E}\boldsymbol{x}(t) \tag{10b}$$

where $\boldsymbol{Y}$, $\boldsymbol{H}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{W}$, and $\boldsymbol{E}$ can be determined by substituting the solution of the state equation (1) given by

$$\boldsymbol{x}_{t+k} = \boldsymbol{A}^k\boldsymbol{x}(t) + \sum_{\kappa=0}^{k-1} \boldsymbol{A}^\kappa \boldsymbol{B}\boldsymbol{u}_{t+k-1-\kappa} \tag{11}$$

into the FHOCP (8) (Bemporad et al., 2002). Problem (10) is a quadratic programming problem which obviously depends on the state vector $\boldsymbol{x}(t)$ measured at the discrete time t and must thus be solved at each discrete time t. This can be done online (implicitly) using standard quadratic programming (QP) methods or offline (explicitly) using multi-parametric quadratic programming (mp-QP) methods. The explicit solution resulting from multi-parametric quadratic programming is very insightful and thus studied in the following.

The concept of multi-parametric quadratic programming consists in considering the state vector $\boldsymbol{x}(t)$ as a parameter and determining the optimal input vector $\boldsymbol{u}^*(t)$ as an explicit function of the state vector $\boldsymbol{x}(t)$.

Theorem 1. Consider the mp-QP (10). The set of state vectors $\boldsymbol{x}(t)$ for which the mp-QP (10) is feasible $\mathbb{X}_\mathrm{f}$ is convex, i.e.

$$\mathbb{X}_\mathrm{f} = \{\boldsymbol{x}(t) | \exists \boldsymbol{U} : \boldsymbol{G}\boldsymbol{U} \leq \boldsymbol{W} + \boldsymbol{E}\boldsymbol{x}(t)\} \subseteq \mathbb{X} \text{ convex}, \tag{12}$$

the optimal input sequence $\boldsymbol{U}^* : \mathbb{X}_\mathrm{f} \to \mathbb{R}^{Nm}$ is continuous and piecewise affine (PWA) on polyhedra $\mathbb{P}_r$, i.e.

$$\boldsymbol{U}^*(\boldsymbol{x}(t)) = \boldsymbol{F}_r^U\boldsymbol{x}(t) + \boldsymbol{G}_r^U \text{ if } \boldsymbol{x}(t) \in \mathbb{P}_r, \tag{13}$$

and the optimal value function $\boldsymbol{V}^* : \mathbb{X}_\mathrm{f} \to \mathbb{R}$ is continuous, convex, and piecewise quadratic (PWQ) on polyhedra $\mathbb{P}_r$, i.e.

$$V_N^*(\boldsymbol{x}(t)) = \boldsymbol{x}^T(t)\boldsymbol{H}_r^V\boldsymbol{x}(t) + \boldsymbol{h}_r^{V^T}\boldsymbol{x}(t) + \boldsymbol{l}_r \text{ if } \boldsymbol{x}(t) \in \mathbb{P}_r. \tag{14}$$

The polyhedra $\mathbb{P}_r$ are described by

$$\mathbb{P}_r = \left\{\boldsymbol{x} | \boldsymbol{H}_r^\mathbb{P}\boldsymbol{x} \leq \boldsymbol{h}_r^\mathbb{P}\right\}, \quad r = 1, \dots, R \tag{15}$$

and form a partition of $\mathbb{X}_\mathrm{f}$, i.e. $\bigcup_{r=1}^R \mathbb{P}_r = \mathbb{X}_\mathrm{f}$.

Proof. The proof is given in (Bemporad et al., 2002; Grieder et al., 2004).

Corollary 2. The optimal input vector $\boldsymbol{u}^*(t)$ results from the PWA state feedback control law

$$\boldsymbol{u}^*(t) = \boldsymbol{F}_r\boldsymbol{x}(t) + \boldsymbol{g}_r \text{ if } \boldsymbol{x}(t) \in \mathbb{P}_r. \tag{16}$$

Proof. The proof follows from (13) considering that the optimal input vector $\boldsymbol{u}^*(t)$ can be extracted from the optimal input sequence $\boldsymbol{U}^*(t)$ using $\boldsymbol{u}^*(t) = (\boldsymbol{I} \ \boldsymbol{0} \ \cdots \ \boldsymbol{0})\boldsymbol{U}^*(t)$.

Remark 3. The polyhedra $\mathbb{P}_r$ are also denoted as regions.

Remark 4. The polyhedra $\mathbb{P}_r$ can be computed offline and stored. Efficient mp-QP algorithms are available for this computation (Alessio and Bemporad, 2009, Section 2.2). The PWA state feedback control law (16) must be evaluated online with a region membership test. Efficient search algorithms are available for this evaluation (Alessio and Bemporad, 2009, Section 4).

Remark 5. The number of regions R is not known until the mp-QP (10) has been solved. Only an upper bound

$$R \leq \sum_{i=0}^{2^q-1} i! q^i \tag{17}$$

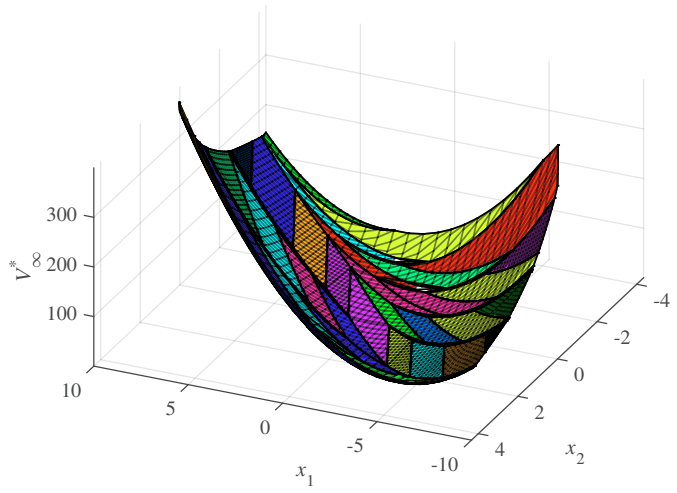

Fig. 1. PWQ Optimal Value Function (14)

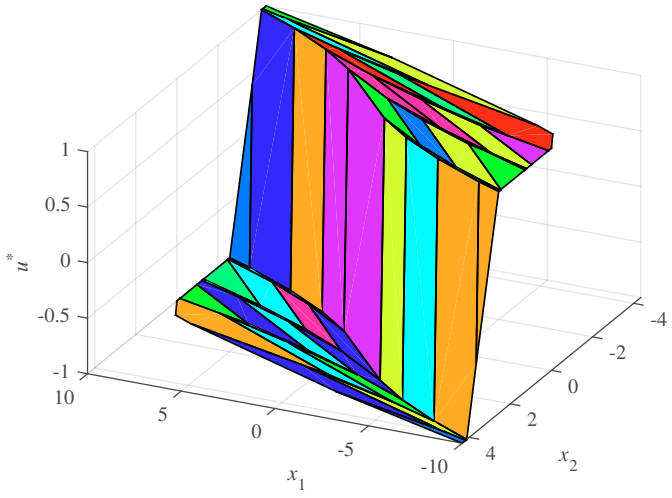

Fig. 2. PWA State Feedback Control Law (16)

where q is the number of constraints (10b) can be computed (Bemporad et al., 2002, Section 4.4), which is, however, very conservative.

Remark 6. An explicit solution of the FHOCP (8) can be determined based on multi-parametric programming also for discrete-time nonlinear time-invariant systems (Johansen, 2004). The ideas presented in Section 5 may therefore be extendable to these systems.

Example 7. Consider a discrete-time linear time-invariant system (1) with

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

representing a double integrator. Consider furthermore the constraints

$$|x_1| \leq 10, \quad |x_2| \leq 10, \quad |u| \leq 1.$$

Consider finally the cost function (3) with

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 0.01 \end{pmatrix}, \boldsymbol{R} = 0.01, \boldsymbol{P} = \begin{pmatrix} 2.03 & 1.04 \\ 1.04 & 1.07 \end{pmatrix}, N = 5$$

where $\boldsymbol{P}$ is the solution of the algebraic Riccati equation (6). The resulting FHOCP (8) can be solved by multi-parametric quadratic programming as discussed in Section 3.2, e.g. under MATLAB with the Multi-Parametric Toolbox (MPT) (Herceg et al., 2013). The resulting PWQ optimal value function (14) and PWA state feedback control law (16) are shown in Figs. 1 and 2. The number of regions is $R = 59$. The set $\mathbb{X}_\mathrm{f}$ is significantly smaller than the set $\mathbb{X}$ as can be observed from Fig. 2. The example shows that even for systems as simple as the double integrator the PWQ optimal value function (14) and the PWA state feedback control law (16) can be quite complex.

3.3 Solution of the IHOCP

Theorem 8. There is a finite prediction horizon $\overline{N}(\boldsymbol{x}(t))$ such that for all prediction horizons $N \geq \overline{N}(\boldsymbol{x}(t))$ the FHOCP (8) and the IHOCP (9) are equivalent.

Proof. The proof is given in (Sznaier and Damborg, 1987; Chmielewski and Manousiouthakis, 1996; Scokaert and Rawlings, 1998).

Remark 9. Theorem 8 implies that the IHOCP (9) can be solved by solving the FHOCP (8) for the prediction horizon $\overline{N}(\boldsymbol{x}(t))$ and that Theorem 1 and Corollary 2 apply analogously for the IHOCP (9).

Remark 10. Various methods for the computation of the prediction horizon $\overline{N}(\boldsymbol{x}(t))$ are given in (Chmielewski and Manousiouthakis, 1996; Scokaert and Rawlings, 1998; Bemporad et al., 2002; Grieder et al., 2004).

Remark 11. The PWA state feedback control law (16) resulting for an infinite prediction horizon $N = \infty$ is denoted as constrained linear quadratic regulator (CLQR) (Bemporad et al., 2002; Grieder et al., 2004).

3.4 Stability and Feasibility

Stability and feasibility are guaranteed inherently for model predictive control based on the IHOCP (9) and can be easily guaranteed for model predictive control based on the FHOCP (8) by imposing a terminal constraint $\boldsymbol{x}_{t+N} \in \mathbb{X}_N$ (Mayne et al., 2000) or utilizing a sufficiently large prediction horizon $N > N_\mathrm{stab}$ (Grüne and Pannek, 2011, Chapter 6).

4. REINFORCEMENT LEARNING

4.1 Reinforcement Learning Principle

Reinforcement learning relies on solving the IHOCP (9) by evaluating a state feedback control law $\boldsymbol{u}(t) = \boldsymbol{h}(\boldsymbol{x}(t))$ for the measured state vector $\boldsymbol{x}(t)$, applying the resulting input vector $\boldsymbol{u}(t)$ to the system (1), measuring the resulting state vector $\boldsymbol{x}(t+1)$, evaluating the cost $r(\boldsymbol{x}(t), \boldsymbol{u}(t))$, and improving the controller $\boldsymbol{h}$ based on the evaluation. This is repeated at each discrete time t. Note that a model of the system (1) is not required for reinforcement learning. This is a major difference to model predictive control.

In the following section dynamic programming as a cornerstone of reinforcement learning will be introduced.

4.2 Dynamic Programming

The value function (3) can be rewritten as

$$V_\infty(\boldsymbol{x}(t)) = r(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \gamma \sum_{k=0}^{\infty} \gamma^k r(\boldsymbol{x}_{t+k+1}, \boldsymbol{u}_{t+k+1}) \tag{18}$$

by separating the first sum term. Realizing that the remaining sum corresponds to the shifted value function and substituting the control law $\boldsymbol{u}(t) = \boldsymbol{h}(\boldsymbol{x}(t))$ into (18) yields

$$V_\infty(\boldsymbol{x}(t)) = r(\boldsymbol{x}(t), \boldsymbol{h}(\boldsymbol{x}(t))) + \gamma V_\infty(\boldsymbol{x}(t+1)) \tag{19}$$

This difference equation is denoted as Bellman equation. The solution of the Bellman equation yields the value $V_\infty(\boldsymbol{x}(t))$ of the policy $\boldsymbol{h}$ for the state vector $\boldsymbol{x}(t)$.

The optimal value function results from the Bellman equation (19) as

$$V_\infty^*(\boldsymbol{x}(t)) = \min_{\boldsymbol{h}} \left(r(\boldsymbol{x}(t), \boldsymbol{h}(\boldsymbol{x}(t))) + \gamma V_\infty(\boldsymbol{x}(t+1)) \right) \tag{20}$$

Bellman's optimality principle says that "An optimal policy has the property that no matter what the previous decisions (i.e. controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions" (Bellman, 1957). With Bellman's optimality principle (20) can be rewritten as

$$V_\infty^*(\boldsymbol{x}(t)) = \min_{\boldsymbol{h}} \left(r(\boldsymbol{x}(t), \boldsymbol{h}(\boldsymbol{x}(t))) \right) + \gamma V_\infty^*(\boldsymbol{x}(t+1)). \tag{21}$$

This difference equation is denoted as Bellman optimality equation or Hamilton-Jacobi-Bellman (HJB) equation.

The optimal policy results from the Bellman optimality equation (21) as

$$\boldsymbol{h}^*(\boldsymbol{x}(t)) = \arg\min_{\boldsymbol{h}} \left(r(\boldsymbol{x}(t), \boldsymbol{h}(\boldsymbol{x}(t))) \right) + \gamma V_\infty(\boldsymbol{x}(t+1)). \tag{22}$$

The Bellman optimality equation can be solved by (i) value iteration, (ii) policy iteration, and (iii) policy search as outlined e.g. in (Buşoniu et al., 2010, Chapters 2 and 3). In the following section policy search using an actor-critic structure (Lewis and Vrabie, 2009; Si and Wang, 2001; Liu et al., 2012; Sokolov and Kozma, 2013; Sokolov et al., 2015) for solving the Bellman optimality equation will be discussed. The actor-critic structure is selected since it is (i) very suitable for combination with model predictive control, (ii) very intuitive, and (iii) very well illustrating the concepts of reinforcement learning. Value iteration, policy iteration, and policy search based on policy gradient methods can be considered for combination with model predictive control as well, but are not discussed for conciseness. Note that adopting structures tailored to the LQR like (Bradtke, 1992; Bradtke et al., 1994; ten Hagen and Kröse, 1998; Landelius, 1997; Al-Tamimi et al., 2008; Lewis and Vrabie, 2009; Lewis and Vamvoudakis, 2011; Lewis et al., 2012) is difficult for systems with state and input constraints.

4.3 Actor-Critic Structure

Consider the actor-critic structure shown in Fig. 3 where the notation $V_\infty(t) = V_\infty(\boldsymbol{x}(t))$ is used for conciseness. The actor applies an action $\boldsymbol{u}$ obtained from the policy $\boldsymbol{h}$ to the system. The critic assesses the value V_∞ of the action. Based on the assessment the policy is improved. These steps are iterated until the value function V_∞ and policy $\boldsymbol{h}$ have converged. The actor and critic essentially

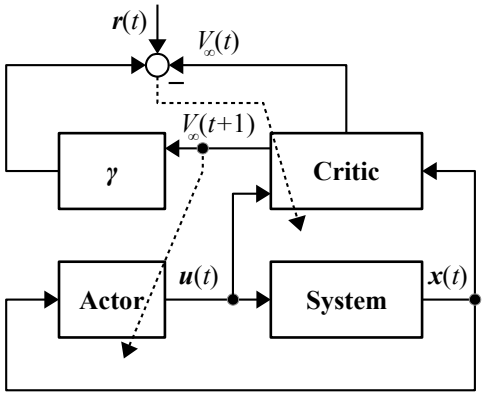

Fig. 3. Actor-Critic Structure

approximate the optimal policy $\boldsymbol{h}^*$ and optimal value function V_∞^*. Various methods can be used for the approximation, specifically parametric methods like neural networks and nonparametric methods like support vector machines (Buşoniu et al., 2010, Section 3.3). In the following sections the approximation using neural networks will be discussed.

4.4 Critic Neural Network

The critic neural network (CNN) approximates the value function $V_\infty(\boldsymbol{x}(t))$. The approximation error can after rearranging the Bellman equation (19) be defined as

$$e_\mathrm{c}(t) = r(\boldsymbol{x}(t), \boldsymbol{h}(\boldsymbol{x}(t))) + \gamma V_\infty(\boldsymbol{x}(t+1)) - V_\infty(\boldsymbol{x}(t)). \tag{23}$$

The approximation error can be considered as a prediction error between the value predicted by the neural network and the value observed from the system. The approximation error is defined based on the difference of the values at discrete time $t+1$ and t and thus also denoted as temporal difference (TD) error. Note that the definition of the TD error according to (23) follows the definition in (Lewis and Vrabie, 2009, Equation (38)) while the definition in (Si and Wang, 2001, Equation (3)) is slightly different.

The approximation error must be minimized. For this purpose the cost function

$$E_\mathrm{c}(t) = \frac{1}{2} e_\mathrm{c}^2(t) \tag{24}$$

is introduced. The weight vector $\boldsymbol{w}_\mathrm{c}$ of the neural network detailed in Section 4.6 can then be updated using the gradient method, yielding

$$\boldsymbol{w}_\mathrm{c}(t+1) = \boldsymbol{w}_\mathrm{c}(t) + \Delta \boldsymbol{w}_\mathrm{c}(t) \tag{25}$$

with

$$\begin{aligned} \Delta \boldsymbol{w}_\mathrm{c}(t) &= -l_\mathrm{c} \frac{\partial E_\mathrm{c}(t)}{\partial \boldsymbol{w}_\mathrm{c}(t)} \\ \frac{\partial E_\mathrm{c}(t)}{\partial \boldsymbol{w}_\mathrm{c}(t)} &= \frac{\partial E_\mathrm{c}(t)}{\partial V_\infty(\boldsymbol{x}(t))} \frac{\partial V_\infty(\boldsymbol{x}(t))}{\partial \boldsymbol{w}_\mathrm{c}(t)} \end{aligned} \tag{26}$$

where $l_\mathrm{c} > 0$ is the step size of the gradient method.

4.5 Actor Neural Network

The actor neural network (ANN) approximates the policy $\boldsymbol{h}$. Essentially the optimal policy minimizing the value function must be determined. The approximation error can therefore be defined as

$$e_\mathrm{a}(t) = V_\infty(\boldsymbol{x}(t)). \tag{27}$$

The approximation error must be minimized. For this purpose the cost function

$$E_{\mathrm{a}}(t) = \frac{1}{2} e_{\mathrm{a}}^2(t) \tag{28}$$

is introduced. The weight vector $\boldsymbol{w}_{\mathrm{a}}$ of the neural network detailed in Section 4.6 can then be updated using the gradient method, yielding

$$\boldsymbol{w}_{\mathrm{a}}(t+1) = \boldsymbol{w}_{\mathrm{a}}(t) + \Delta \boldsymbol{w}_{\mathrm{a}}(t) \tag{29}$$

with

$$\begin{aligned} \Delta \boldsymbol{w}_{\mathrm{a}}(t) &= -l_{\mathrm{a}}(t) \frac{\partial E_{\mathrm{a}}(t)}{\partial w_{\mathrm{a}}(t)} \\ \frac{\partial E_{\mathrm{a}}(t)}{\partial \boldsymbol{w}_{\mathrm{a}}(t)} &= \frac{\partial E_{\mathrm{a}}(t)}{\partial V(\boldsymbol{x}(t))} \frac{\partial V(\boldsymbol{x}(t))}{\partial \boldsymbol{w}_{\mathrm{a}}(t)} \end{aligned} \tag{30}$$

where $l_{\mathrm{a}} > 0$ is again the step size of the gradient method.

4.6 Learning Algorithm

The critic neural network and the actor neural network can be represented by feedforward neural networks with one hidden layer as shown in Fig. 4.

For the critic neural network the output $V_\infty(t)$ is given by

$$V_\infty(t) = \sum_{i=1}^{N_{\mathrm{hc}}} w_{\mathrm{c}i}^{(2)}(t) p_{\mathrm{c}i}(t) \tag{31}$$

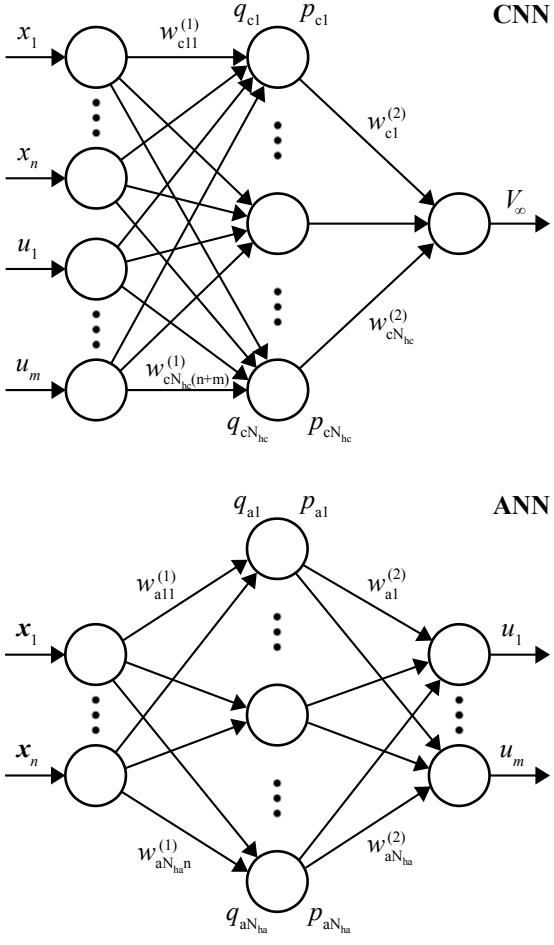

Fig. 4. Critic and actor neural network (CNN and ANN)

with

$$p_{\mathrm{c}i}(t) = \frac{1 - \mathrm{e}^{-q_{\mathrm{c}i}(t)}}{1 + \mathrm{e}^{-q_{\mathrm{c}i}(t)}}, \quad i = 1, \ldots, N_{\mathrm{hc}} \tag{32}$$

$$q_{\mathrm{c}i}(t) = \sum_{j=1}^{n+m} w_{\mathrm{c}ij}^{(1)}(t) z_j(t), \quad i = 1, \ldots, N_{\mathrm{hc}} \tag{33}$$

where N_{hc} is the number of hidden nodes, $q_{\mathrm{c}i}(t)$ are the inputs of the hidden nodes, $p_{\mathrm{c}i}(t)$ are the outputs of the hidden nodes, $\boldsymbol{z}(t) = (x_1(t) \ \cdots \ x_n(t) \ u_1(t) \ \cdots \ u_m(t))^T$ contains the state and input variables, and the notation $V_\infty(t) = V_\infty(\boldsymbol{x}(t))$ is again used for conciseness. Note that a hyperbolic tangent function is used as activation function in (32). With the chain rule the update rule results as

$$\Delta w_{\mathrm{c}i}^{(2)} = -l_{\mathrm{c}} \frac{\partial E_{\mathrm{c}}(t)}{\partial w_{\mathrm{c}i}^{(2)}(t)} \tag{34}$$

$$\frac{\partial E_{\mathrm{c}}(t)}{\partial w_{\mathrm{c}i}^{(2)}(t)} = \frac{\partial E_{\mathrm{c}}(t)}{\partial V_\infty(t)} \frac{\partial V_\infty(t)}{\partial w_{\mathrm{c}i}^{(2)}(t)} = \gamma e_{\mathrm{c}}(t) p_{\mathrm{c}i}(t)$$

$$\Delta w_{\mathrm{c}ij}^{(1)} = -l_{\mathrm{c}} \frac{\partial E_{\mathrm{c}}(t)}{\partial w_{\mathrm{c}ij}^{(1)}(t)} \tag{35}$$

$$\begin{aligned} \frac{\partial E_{\mathrm{c}}(t)}{\partial w_{\mathrm{c}ij}^{(1)}(t)} &= \frac{\partial E_{\mathrm{c}}(t)}{\partial V_\infty(t)} \frac{\partial V_\infty(t)}{\partial p_i(t)} \frac{\partial p_{\mathrm{c}i}(t)}{\partial q_{\mathrm{c}i}(t)} \frac{\partial q_{\mathrm{c}i}(t)}{\partial w_{\mathrm{c}ij}^{(1)}(t)} \\ &= \gamma e_{\mathrm{c}}(t) w_{\mathrm{c}i}^{(2)}(t) \left[\frac{1}{2} \left(1 - p_{\mathrm{c}i}^2(t) \right) \right] z_j(t). \end{aligned}$$

For the actor neural network the output $\boldsymbol{u}(t)$ is given by

$$u_l(t) = \sum_{i=1}^{N_{\mathrm{ha}}} w_{\mathrm{a}li}^{(2)}(t) p_{\mathrm{a}i}(t) \tag{36}$$

with

$$p_{\mathrm{a}i}(t) = \frac{1 - \mathrm{e}^{-q_{\mathrm{a}i}(t)}}{1 + \mathrm{e}^{-q_{\mathrm{a}i}(t)}}, \quad i = 1, \ldots, N_{\mathrm{ha}} \tag{37}$$

$$q_{\mathrm{a}i}(t) = \sum_{j=1}^{n} w_{\mathrm{a}ij}^{(1)}(t) x_j(t), \quad i = 1, \ldots, N_{\mathrm{ha}} \tag{38}$$

where N_{ha} is the number of hidden nodes, $q_{\mathrm{a}i}(t)$ are the inputs of the hidden nodes, and $p_{\mathrm{a}i}(t)$ are the outputs of the hidden nodes. Note that again a hyperbolic tangent function is used as activation function in (37). With the chain rule the update rule results as

$$\Delta w_{\mathrm{a}li}^{(2)} = -l_{\mathrm{a}} \frac{\partial E_{\mathrm{a}}(t)}{\partial w_{\mathrm{a}li}^{(2)}(t)} \tag{39}$$

$$\begin{aligned} \frac{\partial E_{\mathrm{a}}(t)}{\partial w_{\mathrm{a}li}^{(2)}(t)} &= \frac{\partial E_{\mathrm{a}}(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u_l(t)} \frac{\partial u_l(t)}{\partial w_{\mathrm{a}li}^{(2)}(t)} \\ &= e_{\mathrm{a}}(t) \sum_{r=1}^{N_{\mathrm{hc}}} \left[w_{\mathrm{c}r}^{(2)}(t) \frac{1}{2} \left(1 - p_{\mathrm{c}r}^2(t) \right) w_{\mathrm{c}r(n+l)}^{(1)}(t) \right] p_{\mathrm{a}i}(t) \end{aligned}$$

$$\Delta w_{\mathrm{a}ij}^{(1)} = -l_{\mathrm{a}} \frac{\partial E_{\mathrm{a}}(t)}{\partial w_{\mathrm{a}ij}^{(1)}(t)} \tag{40}$$

$$\frac{\partial E_{\mathrm{a}}(t)}{\partial w_{\mathrm{a}ij}^{(1)}(t)} = \frac{\partial E_{\mathrm{a}}(t)}{\partial V_\infty(t)} \left(\frac{\partial V_\infty(t)}{\partial \boldsymbol{u}(t)} \right)^T \frac{\partial \boldsymbol{u}(t)}{\partial p_{\mathrm{a}i}(t)} \frac{\partial p_{\mathrm{a}i}(t)}{\partial q_{\mathrm{a}i}(t)} \frac{\partial q_{\mathrm{a}i}(t)}{\partial w_{\mathrm{a}ij}^{(1)}(t)}$$

$$= \frac{\partial E_{\mathrm{a}}(t)}{\partial V_\infty(t)} \sum_{l=1}^{m} \frac{\partial V_\infty(t)}{\partial u_l(t)} \frac{\partial u_l(t)}{\partial p_{\mathrm{a}i}(t)} \frac{\partial p_{\mathrm{a}i}(t)}{\partial q_{\mathrm{a}i}(t)} \frac{\partial q_{\mathrm{a}i}(t)}{\partial w^{(1)}_{\mathrm{a}ij}(t)}$$

$$= e_{\mathrm{a}}(t) \sum_{l=1}^{m} \sum_{r=1}^{N_{\mathrm{hc}}} \left[w^{(2)}_{cr}(t) \frac{1}{2} \left(1 - p^2_{cr}(t)\right) w^{(1)}_{\mathrm{cr}(n+l)}(t) \right] \cdot w^{(2)}_{\mathrm{a}li}(t) \frac{1}{2} \left(1 - p^2_{\mathrm{a}i}(t)\right) x_j(t).$$

4.7 Stability and Feasibility

Stability, precisely ultimate uniform boundedness (UUB), of the learning algorithm described in Section 4.6 has been addressed in (Liu et al., 2012; Sokolov and Kozma, 2013; Sokolov et al., 2015). The convergence criterion presented in (Sokolov and Kozma, 2013) can be summarized as

Theorem 12. Assume that the optimal weight vectors $\boldsymbol{w}^*_{\mathrm{c}}$ and $\boldsymbol{w}^*_{\mathrm{a}}$ are bounded, i.e.

$$\|\boldsymbol{w}^*_{\mathrm{c}}\| \leq \overline{w}_{\mathrm{c}}, \quad \|\boldsymbol{w}^*_{\mathrm{a}}\| \leq \overline{w}_{\mathrm{a}} \tag{41}$$

and that the reinforcement signal $r(\boldsymbol{x}(t), \boldsymbol{x}(t))$ is a bounded positive semidefinite function. Let the estimated weight vectors $\boldsymbol{w}_{\mathrm{c}}(t)$ and $\boldsymbol{w}_{\mathrm{a}}(t)$ be updated with the learning algorithm described in Section 4.6. Then the errors between the optimal weight vectors $\boldsymbol{w}^*_{\mathrm{c}}$ and $\boldsymbol{w}^*_{\mathrm{a}}$ and the estimated weight vectors $\boldsymbol{w}_{\mathrm{c}}(t)$ and $\boldsymbol{w}_{\mathrm{a}}(t)$ are uniformly ultimately bounded, i.e. for each $\varepsilon > 0$ there exists a $\delta(\epsilon, t_0) > 0$ and a $T(\varepsilon, \delta)$ such that

$$\|\boldsymbol{w}(t_0)\| \leq \delta \Rightarrow \|\boldsymbol{w}(t)\| \leq \varepsilon \quad \forall \geq t_0 + T \tag{42}$$

if the following conditions are fulfilled:

$$\frac{1}{\sqrt{2}} < \gamma, \quad l_{\mathrm{c}} < \frac{1}{\gamma^2 N_{\mathrm{hc}}} - \frac{1}{2\gamma N_{\mathrm{ha}}}, \quad l_{\mathrm{a}} < \frac{2}{3 N_{\mathrm{ha}}}. \tag{43}$$

Constraints on the state and input vector are not considered in the learning algorithm described in Section 4.6. The feasibility has therefore not been studied yet.

5. SYNERGIES BETWEEN MPC AND RL

The discussion in Sections 3 and 4 underlines that model predictive control and reinforcement learning have strong relations. In the context of the optimal control problem introduced in Section 2 the discussion particularly reveals that

- the critic neural network must approximate a value function $V_\infty(\boldsymbol{x}(t))$ which is convex, continuous, and piecewise quadratic over polyhedra, cf. (14)
- the actor neural network must approximate a policy $\boldsymbol{h}$ which is continuous and piecewise affine on polyhedra, cf. (16).

Neural networks as universal approximators (Hornik et al., 1989) can represent such functions with an arbitrary accuracy. The learning may, however, be slow. Exploiting the knowledge on the structure and the properties of the value function and policy may substantially facilitate the learning as shown e.g. in (Buşoniu et al., 2010, Sections 3.7.3 and 5.4). This can enable reinforcement learning for discrete-time linear time-invariant systems with state and input constraints and high order as well as fast dynamics.

Various approaches can be envisioned for exploiting the knowledge on the value function and policy:

- The value function (14) and the policy (16) can be used structurally as parametric approximators. The estimation of the parameters $\boldsymbol{H}^V_r$, $\boldsymbol{h}^V_r$, $\boldsymbol{l}_r$, $\boldsymbol{F}_r$, $\boldsymbol{g}_r$, $\boldsymbol{H}^{\mathbb{P}}_r$, $\boldsymbol{h}^{\mathbb{P}}_r$ is, however, not straightforward due to the region membership test. Furthermore the number of regions R is not known. Only the very conservative upper bound (17) is available. Note that methods for reducing the number of regions are given in the literature (Geyer et al., 2008), basically by merging regions. These methods may be included in the learning algorithm. A moderately conservative upper bound may then be manageable. Research in this direction may be fruitful.
- Various parametric and nonparametric approximators utilizing the knowledge on the value function and policy may be constructed. Candidates can be taken e.g. from (Buşoniu et al., 2010, Section 3.3). Particularly the convexity of the value function and the continuity of the value function and policy may be utilized. Moreover methods for finding approximators automatically (Buşoniu et al., 2010, Section 3.3) may substantially benefit from the knowledge on the value function and policy.
- Manifold parametric approximators for the PWQ value function (14) and the PWA policy (16) are reported in the MPC literature (Alessio and Bemporad, 2009, Section 3) for reducing the offline and online complexity. E.g., approximators with regions defined over simplices or hypercubes (Johansen, 2004) instead of polyhedra have been proposed. Such approximators may also be useful for reinforcement learning.

Investigating these approaches further seems promising. The knowledge on the value function and policy may also be employed for the construction of explicit and non-conservative convergence criteria for the learning algorithm and further for the assessment of stability and feasibility. Note that policy gradient methods which can be considered as actor-only structures (Buşoniu et al., 2010, Section 3.7.1) and which have received considerable attention recently (Deisenroth et al., 2013; Lillicrap et al., 2015) can benefit from these approaches as well.

6. CONCLUSIONS

In this paper relations between model predictive control and reinforcement learning for discrete-time-linear time invariant systems with state and input constraints and a quadratic cost function have been investigated. The structure and properties of the optimal value function and the optimal policy have been revealed and ideas for exploiting the knowledge on the structure and properties in reinforcement learning have been discussed. The findings may promote the convergence of model predictive control and reinforcement learning and stimulate further research both in the control systems and computational intelligence community which is an important purpose of the paper. Overall reinforcement learning enriched by knowledge on the value function and policy can be a basis for adaptive model predictive control with guaranteed stability and feasibility on the one hand and low conservatism and online complexity on the other hand.

REFERENCES

Adetola, V., DeHaan, D., and Guay, M. (2009). Adaptive model predictive control for constrained nonlinear systems. *Systems & Control Letters*, 58(5), 320–326.

Adetola, V. and Guay, M. (2011). Robust adaptive MPC for constrained uncertain nonlinear systems. *International Journal of Adaptive Control and Signal Processing*, 25(2), 155–167.

Al-Tamimi, A., Lewis, F.L., and Abu-Khalaf, M. (2008). Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics)*, 38(4), 943–949.

Al-Tamimi, A., Lewis, F.L., and Abu-Khalaf, M. (2007). Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica*, 43(3), 473–481.

Alessio, A. and Bemporad, A. (2009). A survey on explicit model predictive control. In L. Magni, D.M. Raimondo, and F. Allgöwer (eds.), *Nonlinear Model Predictive Control*, 345–369. Springer, Berlin.

Aswani, A., Gonzalez, H., Sastry, S.S., and Tomlin, C. (2013). Provably safe and robust learning-based model predictive control. *Automatica*, 49(5), 1216–1226.

Balakrishnan, S.N., Ding, J., and Lewis, F.L. (2008). Issues on stability of ADP feedback controllers for dynamical systems. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics)*, 38(4), 913–917.

Bellman, R.E. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3–20.

Benosman, M., Di Cairano, S., and Weiss, A. (2014). Extremum seeking-based iterative learning linear MPC. In *Proceedings of the 2014 IEEE Conference on Control Applications*, 1849–1854.

Benosman, M. and Farahmand, A.M. (2016). Bayesian optimization-based modular indirect adaptive control for a class of nonlinear systems. In *Proceedings of the 12th IFAC Workshop on Adaptation and Learning in Control and Signal Processing*, 253–258.

Benosman, M., Farahmand, A.m., and Xia, M. (2016). Learning-based modular indirect adaptive control for a class of nonlinear systems. In *Proceedings of the 2016 American Control Conference*, 733–738.

Bertsekas, D. (2005). Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5).

Borrelli, F., Bemporad, A., and Morari, M. (in press). *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press.

Bradtke, S.J., Ydstie, B.E., and Barto, A.G. (1994). Adaptive linear quadratic control using policy iteration. In *Proceedings of the 1994 American Control Conference*, 3475–3479.

Bradtke, S.J. (1992). Reinforcement learning applied to linear quadratic regulation. In *Proceedings of the Conference on Advances in Neural Information Processing Systems 5 (NIPS)*, 295–302.

Buşoniu, L., Babuška, R., De Schutter, B., and Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*, volume 39. CRC press, Boca Raton, FL.

Cavagnari, L., Magni, L., and Scattolini, R. (1999). Neural network implementation of nonlinear receding-horizon control. *Neural Computing & Applications*, 8(1), 86–92.

Chmielewski, D. and Manousiouthakis, V. (1996). On constrained infinite-time linear quadratic optimal control. *Systems & Control Letters*, 29(3), 121–129.

Chowdhary, G., Mühlegg, M., How, J.P., and Holzapfel, F. (2013). Concurrent learning adaptive model predictive control. In Q. Chu, B. Mulder, D. Choukroun, E.J. van Kampen, C. de Visser, and G. Looye (eds.), *Advances in Aerospace Guidance, Navigation and Control*, 29–47. Springer, Berlin.

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

Di Cairano, S. (2015a). Model adjustable predictive control with stability guarantees. In *Proceedings of the 2015 American Control Conference*, 226–231.

Di Cairano, S. (2015b). Indirect-adaptive model predictive control for linear systems with polytopic uncertainty. *arXiv preprint arXiv:1509.07170*.

Ernst, D., Glavic, M., Capitanescu, F., and Wehenkel, L. (2009). 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), 517–529.

Fukushima, H., Kim, T.H., and Sugie, T. (2007). Adaptive model predictive control for a class of constrained linear systems based on the comparison model. *Automatica*, 43(2), 301–308.

Geyer, T., Torrisi, F.D., and Morari, M. (2008). Optimal complexity reduction of polyhedral piecewise affine systems. *Automatica*, 44(7), 1728–1740.

Grieder, P., Borrelli, F., Torrisi, F., and Morari, M. (2004). Computation of the constrained infinite time linear quadratic regulator. *Automatica*, 40(4), 701–708.

Grüne, L. and Pannek, J. (2011). *Nonlinear Model Predictive Control*. Springer, London.

He, P. and Jagannathan, S. (2007). Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics)*, 37(2), 425–436.

Heirung, T.A.N., Ydstie, B.E., and Foss, B. (2013). An adaptive model predictive dual controller. In *Proceedings of the 11th IFAC Workshop on Adaptation and Learning in Control and Signal Processing*, 62–67.

Herceg, M., Kvasnica, M., Jones, C., and Morari, M. (2013). Multi-Parametric Toolbox 3.0. In *Proceedings of the 2013 European Control Conference*, 502–510.

Hernandez, B. and Trodden, P. (2015). Persistently exciting tube MPC. *arXiv preprint arXiv:1505.05772*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5), 359–366.

Jiang, Y. and Jiang, Z.P. (2014). Robust adaptive dynamic programming and feedback stabilization of nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5), 882–893.

Johansen, T.A. (2004). Approximate explicit receding horizon control of constrained nonlinear systems. *Au-*

tomatica, 40(2), 293–300.

Landelius, T. (1997). *Reinforcement Learning and Distributed Local Model Synthesis*. Ph.D. thesis, Department of Electrical Engineering, Linköping University, Linköping, Sweden.

Lewis, F.L. and Vamvoudakis, K.G. (2011). Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics)*, 41(1), 14–25.

Lewis, F.L. and Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3), 32–50.

Lewis, F.L., Vrabie, D., and Vamvoudakis, K.G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, 32(6), 76–105.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, F., Sun, J., Si, J., Guo, W., and Mei, S. (2012). A boundedness result for the direct heuristic dynamic programming. *Neural Networks*, 32, 229–235.

Marafioti, G., Bitmead, R.R., and Hovd, M. (2014). Persistently exciting model predictive control. *International Journal of Adaptive Control and Signal Processing*, 28(6), 536–552.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814.

Parisini, T. and Zoppoli, R. (1995). A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10), 1443–1451.

Peters, J. and Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, 71(7), 1180–1190.

Åkesson, B.M. and Toivonen, H.T. (2006). A neural network model predictive controller. *Journal of Process Control*, 16(9), 937–946.

Rathouskỳ, J. and Havlena, V. (2013). MPC-based approximate dual controller by information matrix maximization. *International Journal of Adaptive Control and Signal Processing*, 27(11), 974–999.

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

Riedmiller, M. (2012). 10 steps and some tricks to set up neural reinforcement controllers. In G. Montavon, G.B. Orr, and K.R. Müller (eds.), *Neural Networks: Tricks of the Trade*, 735–757. Springer, Heidelberg, 2nd edition.

Scokaert, P.O.M. and Rawlings, J.B. (1998). Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8), 1163–1169.

Si, J. and Wang, Y.T. (2001). Online learning control by association and reinforcement. *IEEE Transactions on Neural Networks*, 12(2), 264–276.

Sokolov, Y. and Kozma, R. (2013). Improved stability criteria of ADP control for efficient context-aware decision support systems. In *Proceedings of the 2013 International Joint Conference on Awareness Science and Technology and Ubi-Media Computing*, 41–47.

Sokolov, Y., Kozma, R., Werbos, L.D., and Werbos, P.J. (2015). Complete stability analysis of a heuristic approximate dynamic programming control design. *Automatica*, 59, 9–18.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA.

Sutton, R.S., Barto, A.G., and Williams, R.J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 12(2), 19–22.

Sznaier, M. and Damborg, M.J. (1987). Suboptimal control of linear systems with state and control inequality constraints. In *Proceedings of the 26th IEEE Conference on Decision and Control*, 761–762.

ten Hagen, S. and Kröse, B. (1998). Linear quadratic regulation using reinforcement learning. In *Proceedings of the 8th Belgian-Dutch Conference on Machine Learning*, 39–46.

Wang, F.Y., Zhang, H., and Liu, D. (2009). Adaptive dynamic programming: An introduction. *IEEE Computational Intelligence Magazine*, 4(2), 39–47.

Weiss, A. and Di Cairano, S. (2014). Robust dual control MPC with guaranteed constraint satisfaction. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, 6713–6718.

Werbos, P.J. (1990). A menu of designs for reinforcement learning over time. In W.T. Miller III, R.S. Sutton, and P.J. Werbos (eds.), *Neural Networks for Control*, 67–95. MIT Press, Cambridge, MA.

Zhang, H., Luo, Y., and Liu, D. (2009). Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints. *IEEE Transactions on Neural Networks*, 20(9), 1490–1503.

Zhang, H., Liu, D., Luo, Y., and Wang, D. (2012). *Adaptive Dynamic Programming for Control: Algorithms and Stability*. Springer, London.