Mach Learn (2011) 84:137–169
DOI 10.1007/s10994-011-5235-x

Reinforcement learning in feedback control

Challenges and benchmarks from technical process control

Roland Hafner · Martin Riedmiller

Received: 26 February 2010 / Revised: 3 January 2011 / Accepted: 8 January 2011 /
Published online: 27 February 2011

Abstract Technical process control is a highly interesting area of application serving a high practical impact. Since classical controller design is, in general, a demanding job, this area constitutes a highly attractive domain for the application of learning approaches—in particular, reinforcement learning (RL) methods. RL provides concepts for learning controllers that, by cleverly exploiting information from interactions with the process, can acquire high-quality control behaviour from scratch.

This article focuses on the presentation of four typical benchmark problems whilst highlighting important and challenging aspects of technical process control: nonlinear dynamics; varying set-points; long-term dynamic effects; influence of external variables; and the primacy of precision. We propose performance measures for controller quality that apply both to classical control design and learning controllers, measuring precision, speed, and stability of the controller. A second set of key-figures describes the performance from the perspective of a learning approach while providing information about the efficiency of the method with respect to the learning effort needed. For all four benchmark problems, extensive and detailed information is provided with which to carry out the evaluations outlined in this article.

A close evaluation of our own RL learning scheme, NFQCA (Neural Fitted Q Iteration with Continuous Actions), in acordance with the proposed scheme on all four benchmarks, thereby provides performance figures on both control quality and learning behavior.

Keywords Reinforcement learning · Feedback control · Benchmarks · Nonlinear control

1 Introduction

Reinforcement learning (RL) aims at learning control policies in situations where the available training information is basically provided in terms of judging success or failure of the

Editors: S. Whiteson and M. Littman.

R. Hafner (✉) · M. Riedmiller
Machine Learning Lab, Albert-Ludwigs University Freiburg, Freiburg im Breisgau, Germany
e-mail: rhafner@informatik.uni-freiburg.de

M. Riedmiller
e-mail: riedmill@informatik.uni-freiburg.de

observed behaviour (Sutton and Barto 1998). Because this is a very general scenario, a wide range of different application areas can be addressed. Successful applications are known from such different areas as game playing (Tesauro 1992), routing (Boyan and Littman 1994), dispatching (Crites and Barto 1996) and scheduling (Gabel and Riedmiller 2008), robot control (Peters and Schaal 2006; Riedmiller et al. 2009), and autonomic computing (Tesauro et al. 2004) (to name but a few).

In this article, we particularly focus on applications originating from the area of technical process control; the design of high quality controllers is an essential requirement for the operation of nearly every high-level technical system. Application examples can be found in complex technical systems such as aircraft, magnetic levitation trains, and chemical plants, but also in objects of daily life such as air conditioners and computer drives (Krishnakumar and Gundy-burlet 2001; Martinez et al. 2009; Kaloust et al. 1997). However, the design of good controllers is a tedious and demanding job; the classical controller design procedure incorporates careful analysis of the process dynamics, the building of an abstract mathematical model, and finally, the derivation of a control law that meets certain design criteria.

In contrast to the classical design process, reinforcement learning is geared towards learning appropriate closed-loop controllers by simply interacting with the process and incrementally improving control behaviour. The promise of such an approach is tempting: instead of being developed in a time consuming design process, the controller learns the whole of its behaviour by interaction—in the most extreme case completely from scratch. Moreover, the same underlying learning principle can be applied to a wide range of different process types: linear and nonlinear systems; deterministic and stochastic systems; single input/output and multi input/output systems.

Learning the complete control law from scratch is also the underlying scenario for the control tasks presented here—meant as a challenge for the reinforcement learning algorithms under examination. Of course, in a practical application, it is often advisable to use as much prior knowledge as available, and for example to start with the best controller that can be designed in a classical way and use the learning controller then to improve over the inadequacies of the designed controller.

With the advent of increasingly efficient RL methods, one also observes a growing number of successful control applications, e.g. helicopter control (Ng et al. 2004), car driving (Riedmiller et al. 2007a), learning of robot behaviours (Peters and Schaal 2006; El-Fakdi and Carreras 2008), control of soccer robots (Riedmiller et al. 2009), and engine control (Liu et al. 2008). The bibtex collection of Csaba Szepesvari provides a nice overview over successful applications (Szepesvari 2009).

This article complements this line of research on technical process control by providing four different and challenging benchmark systems that are easily accessible for comparison of different learning and non-learning control design approaches. The system equations of each benchmark system are completely specified in the appendix and can be easily implemented.

Benchmarking learning controllers has a long tradition (Anderson and Miller 1990) and has gained an increasing amount of interest in the last couple of years (Whiteson et al. 2010; Riedmiller et al. 2007b) (also see the website at www.rl-competition.org). In parallel, software environments such as RL-Bench (Tanner and White 2009) or CLSquare (Riedmiller et al. 2006) have been developed that support easy study and standardized benchmarking of tasks in many different domains. This article suggests both new benchmarks and new performance measures, which address the needs from the area of technical process control. Also, some domain-specific requirements are introduced (e.g. changing setpoints), that may be considered to play an important role in the further development of the benchmark environments.

Using performance measures that can be applied by both classical controller design methods as well as by learning methods, this article is also intended as a contribution with which to pave the way for comparisons between classical control design techniques and learning methods.

1.1 Contributions

This article presents four challenging tasks from the area of technical process control. The tasks were selected to particularly highlight five central aspects of technical process control: the need for nonlinear control laws; the ability to cope with long-range dynamic effects; the need for highly precise control laws; the ability to cope with the influence of external variables; and the ability to deal with changing set-points. All of the presented systems are particularly interesting from the viewpoint of classical controller design. We therefore hope to stimulate discussion between classical control engineers and machine learning researchers. Whenever available, we will present a classical control solution as a reference. To further encourage the link to classical controller design, we distinguish between two types of performance measures: (a) the performance of the controller is measured in terms of key figures considering precision, speed, and stability of the closed-loop system and (b) the performance of the learning algorithm, measured by key figures describing the efficiency of the learning system with respect to the amount of data needed. Classical controller designs can therefore be compared with learning approaches at least on the level of controller performance.

For the four benchmarks, we will report the results of our own reinforcement learning controller scheme called NFQCA (Neural Fitted Q-Iteration with Continuous Actions). Besides results obtained from classical control approaches, we will primarily give a report of the results obtained from our own method. We do not make comparison to other learning methods on purpose, since the explicit purpose of this article is to provide a solid base for benchmarking in technical process control, and not to show the superiority of our method over others. By concentrating on making performance figures clear and easy to compute, we hope to encourage other teams to apply their algorithms and to publish their results in a similar manner. We believe, that a procedure, where everybody is advocating his or her favourite algorithm and produce the according figures, will lead to the most fair comparison.

In particular, the article makes the following contributions:

- introduction of four benchmark tasks, highlighting various important aspects of typical technical control tasks
- tasks are easy to re-implement (all equations are published in the Appendix)
- suggestions for quantitative performance measures of controller quality, which can be easily determined, general enough for a wide range of control approaches
- suggestions for quantitative performance measures of learning behaviour, which can be easily determined general enough for a wide range of learning approaches
- description of our learning control scheme NFQCA
- systematic evaluation and reporting of NFQCA on all four tasks

As mentioned, it is not our intention to show the superiority of our learning control scheme NFQCA over other approaches. Therefore, no figures for other learning schemes are given. Furthermore, the classical controllers are among the best solutions that we found, but most likely, better solutions exist and hopefully will be advocated by classical control engineers. It is the intention of the article to serve as a starting point for benchmarking both learning and classical controller designs on challenging control tasks.

2 Learning feedback control for technical processes

The classical feedback-control loop describes the application-specific influence of a control device on a controlled process. Within this interaction loop the control device applies appropriate control actions, u, to bring the controlled process variables, y, in close proximity to external set-points or reference inputs, w. A deviation of the controlled process variables from the set-point can occur due to external disturbances on the process and/or to the external change of the set-point. The decision of the control device is based on information that is fed back from the process. In this article we refer to the feedback control schematic depicted in Fig. 1. The controller is reduced to a control law that can be computed on a digital device, e.g. a computer or microcontroller. All physical properties of the devices involved in the feedback control loop are integrated in a process that exhibits a certain composite behaviour. Furthermore, it is assumed that both the process and controller can communicate digital information in discrete time intervals. In each time interval the process communicates a process state, χ (a vector of measured process variables), to the controller that then answers with a vector of control actions. In this way, the resulting control problem can be treated as a state controller where the controlled process variables, y, are contained in the observed process state, χ.

In this article we focus explicitly on processes that can be described as time discrete dynamic systems of the form given in (1).

$$\chi_{t+1} = f(\chi_t, u_t, \sigma_t) \tag{1}$$

It is therefore possible that the dynamic of the process contains strong nonlinearities. This allows for the formulation of a broad range of challenging control applications. Furthermore, the system equation, f, can be subject to a reasonable bounded noise parameter, σ. Assuming this formulation we address one of the most general forms of time discrete, dynamic systems.

Using this formulation, we place some restrictions on the considered dynamic systems and the resulting control problems to set up the control benchmarks for reinforcement learning. As the system equation f does not change over time, we consider autonomous nonlinear time discrete systems. Furthermore, the process variables are assumed to be fully observable and can be observed without noticeable time delay. Dropping these restrictions results in interesting research topics that could also be addressed by reinforcement learning, but which are beyond the focus of the benchmarks presented in this article. However, the presented benchmarks can be extended easily for these settings making them even more challenging.

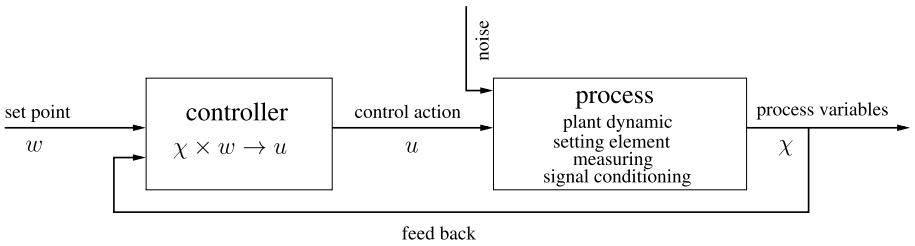

Fig. 1 The schematic of the closed loop control for learning state based feedback control. A process subsumes all physical characteristics of the controlled process, while the controller is an algorithm that is executed on a computer or microcontroller

To solve a control problem it is necessary to find an appropriate control law, $u_t = \pi(\chi_t, w_t)$, such that the closed dynamic system, $\chi_{t+1} = f(\chi_t, \pi(\chi_t, w_t), \sigma_t)$, exhibits certain properties. A broad range of standardized methods to design such control laws are provided by the classical control theory. However, for the design of the controller a high amount of a priori knowledge must be collected concerning the properties of the process and its dynamic. This process is known as "system identification" (Ljung 1999; Goodwin and Payne 1977; Nelles 2001), which is a separate research area with roots in many different fields of research, including statistics, statistical learning, physics, machine learning, and others. One possibility is to use parametric models that are based on the physical properties of the process (white-box models) and to fit the model parameters to data from the real process. Another possibility is to use a regression directly on measured data from the process with nonparametric models (black-box models), e.g. neural networks (Sjöberg et al. 1995). In-between these two approaches exists a wide range of grey-models, which in turn, combine the two approaches. Because the quality of the system identification influences the quality of the designed controller, this step must be performed with a degree of precision. For parametric models, a large degree of expert knowledge of the process is required to be successful. Using nonparametric models, the process of fitting the model is a complex task, which results depend mainly upon the experience with the chosen method and the experiment design in order to collect the appropriate data from the real process.

For linear dynamic systems there exists a broad range of standardized methods for the controller synthesis from an accurate model. For nonlinear systems the methods are less standardized. To design an appropriate control law for a nonlinear system, either the solution will ignore some properties of the process (by treating it as a linear system), or a very high amount of expert knowledge is required (Slotine and Li 1991).

To overcome the loss of controller quality by a poor system model identification, the research field of robust control (Dullerud 2000) exists explicitly to deal with uncertainty in the design. Robust control methods are designed to guarantee a robust solution so long as uncertain parameters or disturbances are within some specified range, however, they do not provide an optimal solution.

The main idea of learning feedback control is to have an intelligent controller component that acts in the standard feedback control loop and learns to control the process by experience made by interaction. Using reinforcement learning methods, a controller can be learned with only a small amount of a priori knowledge of the process (see Sect. 3). Furthermore, the identification of the classical controller design is implicitly incorporated into the learning method. This has several additional advantages: (1) The controller learns with the real process behaviour and thus does not suffer from model inaccuracy or simplifications made in the design process. (2) In contrast to the nonparametric identification models, the problem of where to sample data is also incorporated within the learning process and can be concentrated on regions important to the control application. Also, in contrast to the robust control approach, the learning controller can concentrate on the stochastic behaviour of the process and does not suffer from uncertain parameter estimations of the model.

In our point of view, feedback control is perfectly suited for serving as a test-bed to compare the capability of learning controllers—both against classical controllers as well as against other learning methods. For this comparison three different aspects are important:

- the controller performance that can be achieved by a specific method
- the experience that is required to gain information of the process in form of interaction time with the process
- the amount of a priori knowledge that is required to apply the learning or the design method

A classical nonlinear controller design would require an extremely high amount of a priori knowledge in form of the system model and an expert knowledge of the synthesis process. While the amount of experience is very small—required only at identification—the controller performance will depend on the quality of the design, however, can be expected to be rather high. In contrast to this, a reinforcement learning approach can be expected to require much less a priori knowledge, but on the other hand, an increased number of interactions with the process. In the following section we will propose measures for the controller performance and the required amount of interactions to define a suitable test-bed for benchmarking. Not only is the focus placed on the benchmarking of RL setups against one another, but also against other existing setups.

As it is not possible to specify the required amount of a priori knowledge in any concrete measure, we do not try to identify this amount quantitatively. Instead, we encourage researchers of the different disciplines to accompany their publication of benchmark results with a qualitative description of the required a priori knowledge used.

2.1 Evaluation of controller performance

Classical control theory does not lack tools with which to analyze the control quality of linear controllers applied to linear processes. As we wish to address nonlinear process dynamics and nonlinear controllers, these tools can not be used to serve as an evaluation criterion. Therefore, an appropriate choice for an evaluation criterion of a controller at any given process must originate from the nonlinear dynamic system analysis.

For the analysis in this context, it is necessary to analyze the behaviour of a process dynamic, $\chi(t+1) = f(\chi(t), u(t))$, in combination with a certain control law, $u(t) = \pi(\chi(t), w)$, at a particular set-point, w.

Thus, we are required to analyze a closed dynamic system of the form $\chi(t+1) = h(\chi(t))$. For this system we are interested in the dynamics of the control deviation, $e(t) = w - y(t)$, over time where $y(t)$ comprise the subset of controlled process variables in $\chi(t)$. We assume that there is a solution for the given set-point and thus the dynamic system has an equilibrium point χ_e with the corresponding control deviation $e_e = 0$. For the analysis, we are interested in the stability properties of the equilibrium point e_e. This means we want to know what happens to a control deviation, $e(t)$, for $t > t_0$, corresponding to the initial condition $\chi(t_0)$ and $e(t_0)! = 0$.

Control theory knows different classes of stability (referred to as stability in the sense of Lyapunov), e.g. the equilibrium point $e_e(t)$ is called:

(uniformly) stable if for any $\epsilon > 0$, there exists $\delta(\epsilon) > 0$, such that $|e(t_0)| < \delta(\epsilon) \rightarrow |e(t)| < \epsilon$ for all $t > t_0$.

(uniformly) asymptotic stable if it is (uniformly) stable and there exists $\delta > 0$ independent of t, such that for all $\epsilon > 0$ there exists $N(\epsilon) > 0$, such that $|e(t_0)| < \delta \rightarrow e(t) < \epsilon$ for all $t > t_0 + N(\epsilon)$.

(uniformly) exponentially stable if for any $\epsilon > 0$ there exists $\delta(\epsilon) > 0$, such that $|e(t_0)| < \delta(\epsilon) \rightarrow e(t) < \epsilon e^{-a(t-t_0)}$ for all $t > t_0$ for some $a > 0$.

These definitions of stability include the intuitive idea that the control deviation is bounded, but can be made arbitrarily small through restriction of the initial condition. Asymptotic stability requires the system to converge against the equilibrium point whereas exponential stability requires at least an exponential rate of convergence. In practical applications with perturbations and noise, these concepts of stability are not practicable. Furthermore, when dealing with learning controllers, the control law, $u(t) = \pi(\chi(t), w)$, is often represented

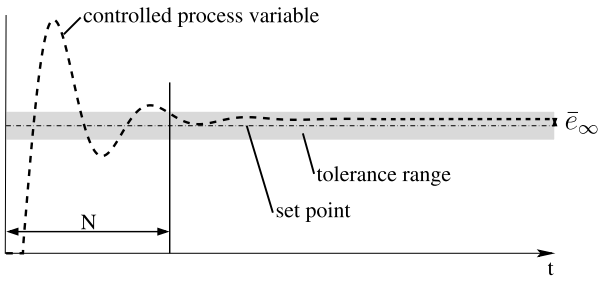

Fig. 2 The schematic diagram of a UUB stable controlled system, where the controlled variable is stable in the tolerance range of a certain set-point

by approximation schemes. Due to numerical issues and approximation errors we can not guarantee that $e(t)$ can be made arbitrarily small by starting close enough to the set-point.

What we would like to expect from a learning feedback controller is a concept originating from boundedness (Kwan et al. 1999; Farrel and Polycarpou 2006). We say the solution is uniformly ultimately bounded (UUB) if there exists a compact set $U \subset \mathbb{R}^n$, such that for all $e(t_0) \in U$ there exists an $\epsilon > 0$ and a number $N(\epsilon, e(t_0))$, such that $|e(t)| < \epsilon$ for all $t \geq t_0 + N(\epsilon, e(t_0))$.

In other words, if the process is started from the initial process state, χ_0, it is ensured that, after some maximum number of time-steps, the controlled variables reach a tolerance range around the set-point and do not leave it again (see Fig. 2). To deduce an evaluation criterion for the controller performance from this definition, we define a benchmark-specific tolerance range, μ. The evaluation of the controller can be achieved by interaction with the process on trajectories of length T. If N is measured for various initial process states and set-points, we obtain a measure for the time-optimality of the controller. Because this criterion evaluates only the time-optimality of the controller, a second criterion is required that provides information about the preciseness of the controller in the tolerance range.

We assume that time-optimality and controller precision are good general measures of control quality. For the evaluation from a fixed starting point, χ_0, with a fixed set-point, w, we define

- $N(\chi_0, w)$ as the number of time-steps after the controlled process variable enters the tolerance range around the set-point w (and does not leave it again). More formally, N is the smallest t for which $|y_{t+k} - w| < \mu$ for all $k \in 0, \ldots, T - t$ holds ($N = T$ if no such t exists).
- e_∞ as the mean absolute off-set the controlled process variable has from the set-point after a certain number of time-steps N_{max}

$$e_\infty(\chi_0, w) = \frac{1}{T - N_{max}} \sum_{t=N_{max}}^{T} |y_t - w| \tag{2}$$

To evaluate the controller performance over a broad range of working conditions, we take the mean values of these measurements over J runs, each with a predefined start point and set-point. In this way we are able to obtain a measure representing the average performance of the controller for steps in the set-point over the whole working range: $\bar{e}_\infty = \frac{1}{J} \sum_{j=1}^{J} e_\infty(\chi_0^j, w^j)$ and $\bar{N} = \frac{1}{J} \sum_{j=1}^{J} N(\chi_0^j, w^j)$.

These evaluation criterion are defined on single steps of the set-point. In many applications there are certain kinds of set-point trajectories of special importance, e.g. rapidly changing, or even continuously changing, set-points. To evaluate the overall performance of the controller we define an additional criterion:

– $e_T(\chi_0, w(t))$ as the mean absolute off-set that the controlled process variable has from the set-point on a predefined reference trajectory $w(t)$

$$e_T = \frac{1}{T_{traj}} \sum_{t=0}^{T_{traj}} |y_t - w(t)| \tag{3}$$

This criterion is defined on a separate trajectory to which the controller is applied. Typically, this trajectory has a much longer duration, T_{traj}, than the other measures as well as an application-specific form.

The parameters that describe the evaluation of the controller performance in a certain benchmark can be specified by the following: the trajectory length, T; the start of the evaluation period for determining the remaining control deviation, N_{max}; a set of J start point and set-point tuples $B = \{(\chi_0^j, w^j), j = 1, \ldots, J\}$ for evaluating the preciseness, $\bar{e}_\infty$, and the time-optimality, $\bar{N}$. In addition, the reference set-point trajectory length T_{traj}, the initial process state for the trajectory, χ_0, and a set-point trajectory, $w(t)$, define the evaluation of the overall performance of the controller e_T.

2.2 Evaluation of learning performance

Especially for learning controllers the pure evaluation of the controller performance that can be learned is not sufficient. An additional measure is required that describes quantitatively how much effort is required to learn the controller in the given task. Especially for learning feedback control the typical application that should come in reach for learning controllers are real devices or machines. For this application it is a central key feature how long the controller is occupied with learning, how much wear and tear the learning procedure causes and when the system can be productive again.

To characterize learning performance, we propose a strict separation of two phases, namely a learning phase and an evaluation phase. The evaluation phase is used to determine the performance of the controller for comparison. Therefore, in the evaluation phase, the conditions are exactly specified, e.g. the set of starting states, the length of the trajectories, changes in set-points, etc. For the learning phase, these settings might be chosen individually to fit the needs of the learning method at hand.

Learning performance then is reported in the number of interactions that are required during the learning phase to reach a certain performance level. The rationale behind this is that the lower the number of interactions required for learning, the easier it is to bring the controller into a corresponding real-world application.

Another interesting figure is the time, that a controller spends interacting with the process in a corresponding real world scenario. To identify this number, we simply have to multiply the number of interactions with the length of the control interval. This number, the actual interaction time with the process, is also provided in the following experiments.

3 Reinforcement learning controllers

In this chapter we briefly describe our approach in formulating the various requirements of technical process control within the framework of a reinforcement learning controller. Our approach is based on batch learning of a Q-value function based on experiences of state transitions. In particular, we discuss learning design issues such as the choice of the immediate cost function and the choice of the inputs with which to realize a set-point controller. Furthermore, we present our learning scheme for learning neural network based controllers with continuous action values (NFQCA).

3.1 An RL formulation for feedback control

Our approach is based on the formulation of the control problem as a Markov Decision Process (MDP) (Sutton and Barto 1998; Bellman 1957). In this setting a learning agent interacts with its environment in discrete time steps. In every time step, t, the agent observes a state $x_t \in X \subset \mathbb{R}^n$ from the environment and chooses an action $u_t \in U \subset \mathbb{R}^m$, based on its current policy $\pi(x_t) : X \to U$. In the subsequent time-step, $t+1$, the state of the environment is assumed to change according to a transition probability $P(x_{t+1}|x_t, u_t)$. The successor state can be observed by the agent, accompanied by an immediate cost signal $c(x_t, u_t)$. The task is to find an optimal control law or policy, π^*, that minimizes the expected accumulated cost, $\sum_{t=0}^{\infty} c(x_t, u_t)$, for every initial starting point x_0. For simplicity, we assume that the vector of process variables χ can be observed.

Dealing with multiple set-points An important characteristic of feedback controllers is that they have to cope with varying set-points for the target values of the process variables. Whereas the majority of typical RL tasks is characterized by a single goal state (or by a single goal region, respectively), here, the set-points that determine the goal states may vary continuously, and therefore an infinite number of goal states must be managed.

As the controller learns by interaction with the control process it has to generalize from samples of interaction—not only over the state and action space, but also over different set-points. By consequence, the information of the current set-point has to be integrated in the state of the MDP. In general, there are different possibilities to represent this information in the MDP state. In our setup the state of the MDP is given as $x_t = [\chi_t, e_t]$: a combination of the process variables, χ_t, and the recent control deviation, $e_t = w_t - y_t$.

If we consider a constant set-point, w, the transition probabilities, $P(x_{t+1}|x_t, u_t)$, for the MDP are entirely defined by the dynamic, $\chi_{t+1} = f(\chi_t, u_t, \sigma_t)$, of the process. A problem arises if we wish to allow changing setpoints w_t. The change of the set-point from w_t to w_{t+1} has a direct impact on the MDP state and, hence, on the observed transitions. If the set-point is not frequently changed, this impact could be modelled as additional noise that will not exhibit too large an effect on the learning agent. But if the set-point is changed frequently—or even continuously—this will violate the Markov Property (see Sutton and Barto 1998) of the modelled MDP. To prevent this we can add a kind of sample and hold element for the set-point within the interaction loop of the controller. Upon execution of the controller in every time-step, t, the control action, u_t, is computed using the recent set-point, w_t, as $u_t = \pi([\chi_t, w_t - y_t])$. In the subsequent time-step the controller observes the process state χ_{t+1} and builds a transition (x_t, u_t, x_{t+1}) for the MDP using set-point w_t as $([\chi_t, w_t - y_t], u_t, [\chi_{t+1}, w_t - y_{t+1}])$. In other words, we build a consistent transition for the MDP that does not include a change in set-point. Using these transitions we can train a controller that will compute the optimal control action for a process state and set-point, under the assumption that the set-point will not change in the subsequent time-step. This behaviour agrees completely with the functionality of feedback control wherein the controller reacts only to the control deviation of the recent time-step.

Specifying the immediate costs One of our most important design objectives is to incorporate as little prior knowledge into the specification for the learning controller as possible. In other words, we are always looking for generic settings that can be applied to a wide range of tasks without tuning. A very general choice for the immediate cost function is given by the following definition, that only considers the error between the desired and the actual

value to determine the immediate costs:

$$c(x,u) = c(e) = \begin{cases} 0 & |e| < \mu \\ \mathcal{C} & \text{else} \end{cases} \tag{4}$$

Here, $\mathcal{C}$ is a positive constant value while μ defines the tolerance of the target region and therefore determines the expected precision. This formulation of the cost signal is a good representation of the UUB stability claims. The learning agent will optimize its policy to reach the tolerance range in a minimum number of time-steps and strives for forever staying within. Because all states within the tolerance range are not punished, we can expect time-optimal and stable control, but not precise control. To achieve a more precise control law, the cost function can be refined while retaining the advantages of the above definition. In (5) a smooth and differentiable cost function is given and enables the agent to learn precise and time-optimal control (a similar cost function is used in Deisenroth et al. 2009; Hafner 2009).

$$\begin{aligned} c(x,u) &= c(e) \\ &= \tanh^2(|e| * w) * \mathcal{C} \\ w &= \tanh^{-1}\left(\frac{\sqrt{0.95}}{\mu}\right) \end{aligned} \tag{5}$$

In Fig. 3a the original cost function (dashed line) and the precise time-optimal cost function are plotted in a one-dimensional setting. A typical resulting value function for the two different cost functions is illustrated in Fig. 3b. In 3b-I a typical effect of the original formulation on the optimal value function is depicted. The accute steps of the direct cost function results in similarly accute steps in the optimal value function. Especially when using approximation schemes, these sharp steps in the value function are a problem and result in a high approximation error. In contrast to this, the smooth immediate cost function of (5) ehibits an additional advantage: the resulting optimal value function is smooth and can therefore be approximated much more accurately.

3.2 Neural reinforcement learning controllers

In the following section we provide the basic idea of a recently developed value function based RL algorithm—the Neural Fitted Q-Iteration for Continuous Actions (NFQCA)—

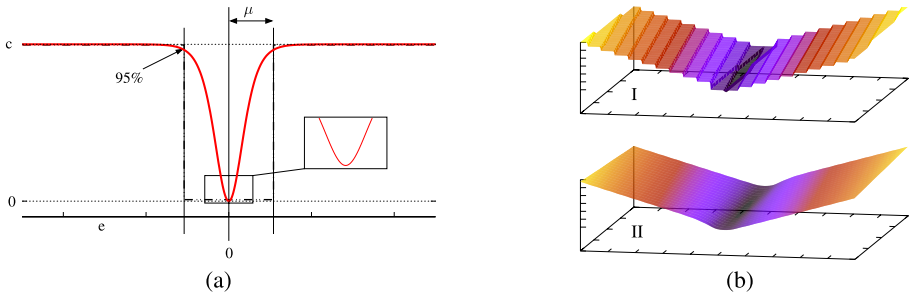

Fig. 3 (Color online) (**a**) A sketch of the definition of the direct cost signal over a one dimensional control deviation, e, with tolerance range, μ. Pure time-optimal definition: *dashed dotted* (*black*); precise time-optimal definition: *solid* (*red*). (**b**) An exemplary visualisation of typical forms of value functions that results from the definitions of the direct cost functions. *I*: the pure time-optimal definition; *II*: the precise time-optimal $\tanh^2$ definition

which is used to generate the evaluations for the benchmarks we introduce for learning feedback control. Within the scope of this article we provide a general description of the algorithm and its properties to accompany the benchmark results (more details on this can be found in Hafner (2009) and in a forthcoming paper specifically devoted to this algorithm). The NFQCA algorithm is a expansion of the Neural Fitted Q-Iteration (NFQ) algorithm (Riedmiller 2005; Hafner and Riedmiller 2007) that was developed especially within the context of reinforcement learning for feedback control applications. It was developed to overcome one of the main shortcomings of the NFQ algorithm, namely the restriction of NFQ to discrete actions. To overcome this problem NFQCA is designed as a fitted actor critic algorithm for continuous state and action spaces, based on the principles of NFQ.

NFQ utilizes a learning agent that interacts with its environment in discrete time-steps. In every time-step the agent observes the state, x, of the environment, chooses an action, u (based on its recent policy $\pi(x)$), and observes a successor state x' in the subsequent time-step. All of the experience is stored in form of the observed transitions in a dataset, $\mathfrak{D}$, with entries $d = (x, u, x')$. NFQ is an iterative algorithm that represents the Q-function (Watkins 1989; Watkins and Dayan 1992) in form of a neural network, $\mathcal{Q}(x, u, w_q)$, with weights w_q. In a certain iteration step, k, a new target value $\hat{Q}_{x,u} = c(x, u) + \min_b Q_k(x', b)$ is computed for each transition sample in the dataset, $\mathfrak{D}$, using the standard Q-update function. From this information we are able to build a training set, P, with entries $(p^{input}, p^{target}) = ((x, u), \hat{Q}_{x,u})$. Using the training set P we can apply an efficient epoch-based supervised learning method with which to adjust the weights of the neural Q-function for the iteration $k + 1$ of the algorithm. In our approach we use Resilient Propagation (RProp) (Riedmiller and Braun 1993) as an epoch-based learning method that proved to be very robust with respect to the choice of the learning parameters and the topology of the network.[1]

With discrete actions, the standard Q-learning rule can be applied directly. With continuous actions this simply is not possible. The main reason for this is that we can not directly find the action with the minimal Q-value for a given state in the neural Q-function. To overcome this problem, in NFQCA—in addition to the neural Q-function, that serves as the representation of the critic—the actor is explicitly represented by a neural policy function, $\pi(x, w_\pi)$, with weights, w_π. In iteration step k of the NFQCA algorithm we assume that the recent policy π_k represents the greedy evaluation of the Q-function: $\pi_k(x) \approx \text{argmin}_u(\mathcal{Q}_k(x, u))$. With this assumption we can formulate the Q-update without a minimization step over all actions as $\hat{Q}(x, u) = c(x, u) + \mathcal{Q}_k(x', \pi_k(x'))$. Analogous to NFQ, with this update rule a training set can be built to adjust the weights, w_q, for $\mathcal{Q}_k + 1$ with RProp. After updating the neural Q-function, the weights of the neural policy function w_π must be updated so that it represents a greedy evaluation of the updated Q-function, according to the base assumption of the critic update. In NFQCA again, a gradient descend algorithm is used to adjust the weights of the policy. For a set of states, each state, x, is propagated forward through the policy network. The same state x and the policy output $\pi(x, w_\pi)$ are then propagated forward through the neural Q-function. As a property of neural networks, the partial derivatives of the Q-function with respect to the action inputs of the network, $\frac{\partial \mathcal{Q}(x,\pi(x,w_\pi),w_q)}{\partial u}$, can afterwards be computed by backpropagation through the neural Q-function. These partial derivatives can be propagated backwards through the neural policy function to compute $\frac{\partial \mathcal{Q}(x,\pi(x,w_\pi),w_q)}{\partial w_\pi}$: the partial derivatives of the Q-function

[1] A comparison of different optimized Backpropagation algorithms for supervised training of neural networks can be found in literature, e.g. Schiffmann et al. (1993); in context of supervised learning in RL see Hafner (2009).

with respect to the weights of the policy net. With these gradients and a set of states x, e.g. the states stored in the dataset, $\mathfrak{D}$, RProp, as an epoch based gradient descend scheme, can be applied to the weights of the policy net.

This basic idea of propagating the gradient of a neural Q-function through the policy network can also be found in earlier approaches (Jordan and Jacobs 1990; Prokhorov and Wunsch 1997; Wang and Si 2001). However, with NFQCA we combine this idea with a model-free batch reinforcement learning approach based on conventional Q-learning. The resulting method is very efficient with respect to the number of required interactions, and is able to learn high quality continuous control laws. Also, the actor critic setting of NFQCA outperforms a pure gradient-based search of $\text{argmin}_u(\mathcal{Q}_k(x,u))$ using $\frac{\partial \mathcal{Q}(x,u)}{\partial u}$ in a single neural Q-function for general control applications (Hafner 2009). One of the advantages of NFQCA is that a gradient-based search in a neural Q-function requires an iterative procedure of propagating values forward and backward through the network for a sufficient number of search steps, starting from a number of different initial values to prevent local minimas. Hence even for a modest network and state-action size the computation of the policy requires several milliseconds, where we can compute the policy with NFQCA in just one propagation step. This allows high control frequencies (up to a few kilo-hertz) that are required for many real-time control applications.

Learning feedback control in this article uses an explorative learning process for both learning algorithms. The learning process has two phases which alternate. In the first phase, the process is controlled by the recent policy for a certain number of time-steps and the observed transitions are added to the dataset, $\mathfrak{D}$. In the second phase, an update iteration of the learning algorithm is applied using the recent experience contained in $\mathfrak{D}$. Using this procedure the learning process can be started with an empty dataset, $\mathfrak{D}$, and randomly initialized neural functions to learn a policy for the given task.

4 Benchmark environments for technical process control

A crucial point for the selection of the four benchmarks presented in the following, is that they are interesting and challenging both from the perspective of classical control theory and from the perspective of machine learning. In particular, for three of the four benchmarks, an analytically designed controller is known, which can be used as a reference for controller performance. Furthermore, the benchmarks are selected, since they explicitly shed light on one or more of the key properties, that are essential in the domain of technical process control. The key properties examined here are listed in the following:

1. The existence of an external setpoint, that can take arbitrary values, is a general requirement of applications in technical process control. Up to now, in the RL research community external setpoints only play a minor role (if at all).
2. The request of a highly accurate control behaviour, that reaches the given setpoint with high precision. For a learning controller, this is a considerable challenge, which might for example require the use of continuous actions.
3. The presence of nonlinear system behaviour, where the application of RL can have a considerable advantage, compared to classical controller design methods.
4. The presence of long-range dynamic effects, where the system has to be controlled over several hundred of time-steps to reach a target state. This is particularly challenging for RL controllers based on dynamic programming methods. In particular, value function based approaches as they are used here must be able to accurately estimate path costs over a long range of control steps.

Table 1 Proposed benchmark tasks and to what extent they shed light on the respective properties

Property	Underwater Vehicle	Pitch Control	Magnetic Levitation	Heating Coil
nonlinear dynamics	+++		+++	++
long-range dynamics		+++	+	+
precise control	++	++	+++	+
changing setpoints	+++	+++	+++	+++
external variables				+++

5. The presence of external system variables, that can not be influenced by the controller but represent noise in form of some external environmental changes the controller has to cope with.

Table 1 gives a quick overview over the tasks presented in the following and to which extent the tasks reflect the key properties. A more detailed description of the challenges is given in the following subsections.

We are aware, that many more interesting features for potential benchmark tasks can be identified, like high-dimensional state spaces, multi-dimensional actions, time-delay of sensor information, or partial observability of state variables. To keep a reasonable focus, we concentrated here on the five properties of the list above. However, it is straight-forward to extend the proposed benchmarks to introduce features like time-delay or partial observability. The performance figures presented in this article might then serve as a reference for the 'ideal' case.

4.1 Underwater vehicle

The first benchmark problem has only a loose connection to a real system, and is a kind of synthetic problem setup that is especially designed to show interesting properties for learning feedback control. As the process state has only one dimension, the structure of the benchmark setup is relatively simple. Nevertheless, the dynamic of the benchmark has highly nonlinear properties. An aspect of this worth noting is that, for the time-optimal and precise control of the problem at each state, an appropriate continuous control action must be chosen carefully.

We concentrate on the velocity control of a virtual, miniature underwater vehicle that is driven by a propeller. The only process variable is the velocity, v, of the vehicle submerged in water. The mass, m, and drag coefficient, c, of the vehicle are assumed not to be constant. Instead they are replaced by the equivalent mass function, $m(v)$, and equivalent drag coefficient, $c(v)$, that represent the complex dynamic motion effects of a vehicle in a fluid (for a similar system see Slotine and Li 1991).

Furthermore we assume the control action u to influence the thrust produced by the propeller e.g. thought of the velocity (or ideal thrust) of the propeller that influences the effective thrust. The effective thrust that acts on the vehicle is computed by a coefficient, $k(v, u)$, that represents the efficiency the propeller exhibits at certain vehicle speeds and

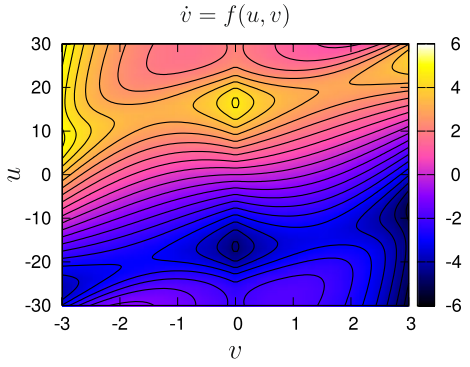

Fig. 4 A plot of the dynamic of the underwater vehicle. The resulting acceleration, $\dot{v} = f(v, u)$, is plotted over the velocity, v, and control action, u. As illustrated, the dynamic of the system is highly nonlinear and shows intriguing properties for control applications

Table 2 Characterization of the process for the velocity control of the underwater vehicle

χ	v	velocity of vehicle		[m/s]
y	v			[m/s]
w	v_d	desired velocity	$\in [-3, 3]$	[m/s]
u	u	ideal thrust	$\in [-30, 30]$	
Δt			$= 0.03$	[s]

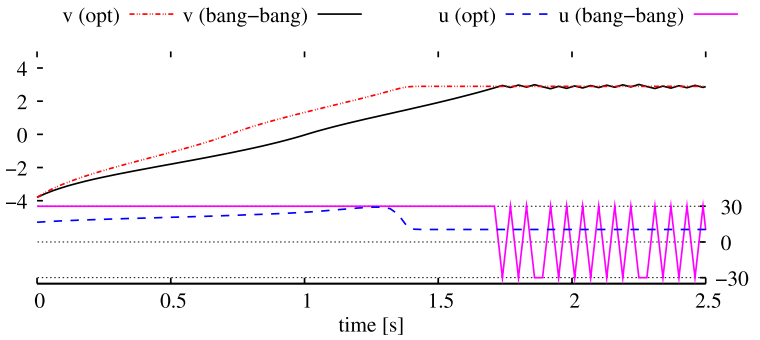

Fig. 5 (Color online) Comparison of a bang-bang controller and a nearly optimal controller. The velocity of the underwater vehicle is started from −4 m/s and should be controlled to 3 m/s. The *solid line* in the upper part of the plot (*black*) shows the velocity of the vehicle when controlled by the bang-bang controller with actions plotted as *solid line* in the lower part of the plot (*magenta*). The *dashed dotted line* (*red*) when controlled by the nearly optimal controller with actions plotted as a *dashed line* (*blue*). The bang-bag controller is slower in reaching the set-point and can not reach it precisely

control actions. In Fig. 4 the dynamic of the underwater vehicle is given as a plot of the resulting acceleration, $\dot{v} = f(v, u)$, when applying a certain control action, u, at certain vehicle velocities, v. For the benchmark we simulate the process dynamic (see A.1.1) over a time interval of 0.03 s.

4.1.1 Control challenge

For the underwater vehicle, the control task is to appropriately control its velocity (see Table 2). Based on the dynamics of the system in Fig. 4, it follows that a true time-optimal and precise control can not be achieved by a bang-bang controller with minimal and maximal control actions (see Fig. 5). If the controller is expected to generate the maximum possible

Table 3 Characterization of the benchmark parameters for evaluation

T	150 time-steps (4.5 seconds)
N_{max}	50 time-steps (after 1.5 seconds)
μ	0.3 rad
J	50 runs
χ_0^j	v uniformly distributed $\in [-5$ m/s, 5 m/s$]$
w^j	uniformly distributed $\in [-3$ m/s, 3 m/s$]$
T_{traj}	800 time-steps (24 seconds)
$w_{traj}(t)$	step and continuously changing

Table 4 Parameters for learning the velocity control of the underwater vehicle

x	2 dim.	v	velocity of vehicle
		$v_d - v$	control deviation
$c(x,u)$	$\tanh^2$	x^d	(–, 0)
		μ	(0, 0.3)
		$\mathcal{C}$	0.01
u	1 dim.	u	ideal thrust
	NFQCA	$\mathcal{U} =$	$[-30, 30]$
	NFQ	$U =$	$\{\pm 30, \pm 15, 0\}$
$Q(x,u)$	neural	topology	3-6-1 (600 epochs RProp)
$\pi(x)$	neural	topology	2-15-1 (400 epochs RProp)

acceleration, it must very carefully choose an appropriate control action at every velocity the vehicle is travelling. In this point of view the benchmark is an example for a control task that has a highly nonlinear dynamic behaviour and requires a continuous control law, if precise and time-optimal control are to be achieved. It is therefore an excellent challenge for learning approaches that are able to deal with continuous actions.

4.1.2 Benchmark environment

In order to evaluate $\bar{N}$ and $\bar{e}_\infty$, the controllers are tested on a set of 50 trajectories. Each trajectory has a length of 150 time-steps (4.5 seconds) and is started with a uniformly distributed initial velocity. At the beginning of every trajectory, a set-point between -3 m/s and 3 m/s is chosen and kept constant over the whole trajectory (see Table 3).

To evaluate e_T a reference trajectory is defined for the vehicle, starting with $v_0 = 0$ m/s. The reference trajectory has several parts, combining steps and continuously changing characteristics (see Fig. 7 and Table 18).

4.1.3 Benchmark results

Learning with NFQCA and NFQ (parameters in Table 4) is done using interaction trajectories with a length of 50 time-steps, where each trajectory is started from a uniformly distributed initial velocity, $v_0 \in [-5$ m/s, 5 m/s$]$, with a uniformly distributed set-point, $v_d \in [-3$ m/s, 3 m/s$]$.

The learning curve for NFQCA is depicted in Fig. 6. After only a few iterations (approx. 30 iterations), the controller is much better in the time criterion, $\bar{N}$, as a reference bang-bang controller. In the subsequent iterations the controller improves $\bar{N}$, but also $\bar{e}_\infty$ and e_T. We

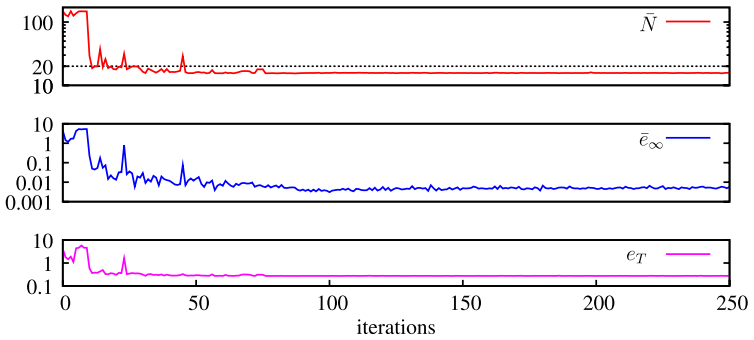

Fig. 6 Learning curves for NFQCA on the underwater vehicle benchmark task. In every iteration 50 interaction samples with the process are collected using the recently learned controller. Afterwards, an update of the controller is performed with all samples collected up to that point. The *black-dotted line* shows the value, $\bar{N} = 20.04$, of a bang-bang controller (with actions: -30 and 30)

Table 5 Benchmark evaluation for the underwater vehicle control challenge. Smaller values in the evaluation criteria mean better performance of the controller. For comparison, the results of a bang-bang controller with minimal and maximal actions are shown

Controller	$\bar{N}$	$\bar{e}_\infty$	e_T
bang-bang	20.04	0.131	0.65
NFQ	18.22	0.054	0.34
NFQCA	15.68	0.003	0.27

report the results of the controller with the lowest value for the time-criterion, $\bar{N}$, as the best controller. If multiple controllers have the same value in $\bar{N}$, the one with the lowest value $\bar{e}_\infty$ is reported. For NFQCA, the best controller was learned after 100 interaction trajectories. This corresponds to 5,000 interaction steps with the process (or 2.5 minutes of interaction with the corresponding real-time process). With NFQ and 5 discrete actions, however, we needed 140 iterations and interaction trajectories until the best controller was learned (7,000 interactions or 3.5 minutes of interaction with the real-time process). As the benchmark results in Table 5 show, by using continuous actions, the controller is better in the time criterion, $\bar{N}$, and also more precise ($\bar{e}_\infty$). Furthermore, the overall quality of the controller on the set-point trajectory clearly shows the benefit of continuous actions (e_T).

In Fig. 7 the learned NFQCA controller is shown on the reference trajectory. With NFQCA, a smooth control law can be learned that controls the velocity such that it follows the set-point very closely.

4.2 Pitch control

This control benchmark refers to an autopilot that controls the pitch of a Boeing airliner aircraft. It is taken from a collection of detailed control examples (CTM 1996) where several classical reference controllers are explained for educational purposes. Though the equations governing the motion of an aircraft are a very complicated set of six non-linear differential equations, under certain assumptions they can be decoupled and linearised. Here we will focus on the longitudinal and linearised problem of pitch control of the aicraft in a steady cruise at constant altitude and velocity. This implies that we can disregard the effects of

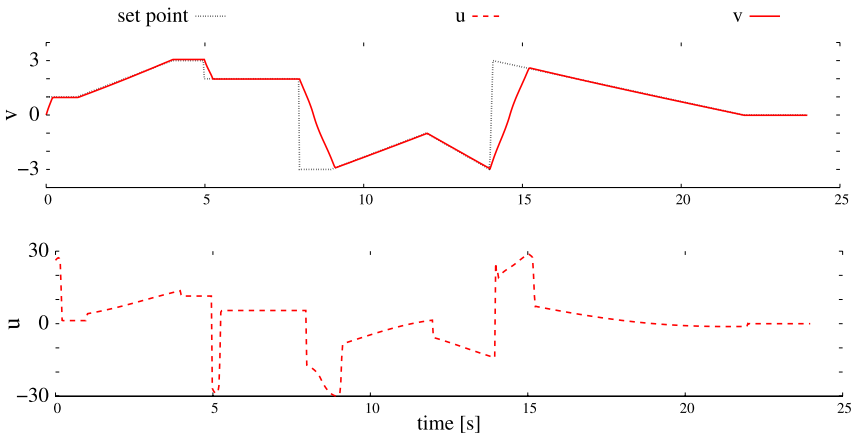

Fig. 7 NFQCA controller on a set-point trajectory of the underwater vehicle task

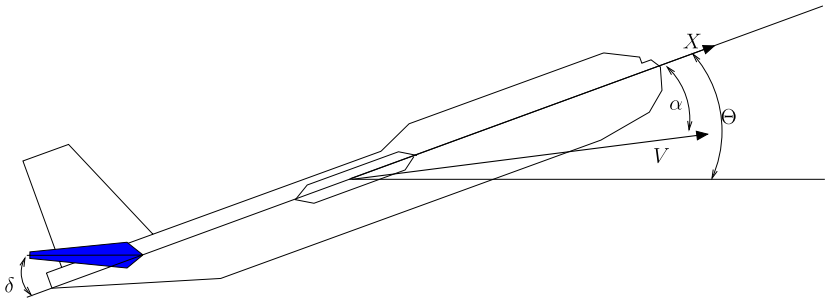

Fig. 8 Schematic of the pitch control process. X denotes the base coordinate axis along the main axis of the aircraft

thrust, drag, and lift in the dynamic of the aircraft. We also assume that any change in the pitch angle does not change the speed of the aircraft under any circumstances.

Figure 8 shows the schematic of the benchmark process. We assume the base coordinate system to run along the main axis of the aircraft (denoted by X in Fig. 8). Also the aircraft is assumed to move with constant velocity, V, under an varying angle of attack, α, with respect to the main axis. By setting the elevator deflection angle, δ, a controller can influence the angle of attack, the pitch rate, q, and the pitch angle, θ, of the aircraft.

4.2.1 Control challenge

The control task here is to provide appropriate elevator deflection angles, δ, to allow the pitch angle, θ, to come as close as possible to the current set-point, θ_d (see Table 6). The range of set-points, θ_d, is restricted between ± 0.5 rad, while the model dynamics provide a reasonable approximation of the real behaviour of the aircraft. For a learning state controller we have a controller state vector, x, that contains at least three of the following process variables: α, q and θ. As the process dynamic is linear and does not depend on the recent pitch angle, we can replace θ with $e = \theta_d - \theta$ in the controller state representation. By con-

Table 6 Characterization of the pitch control process

χ	α	angle of attack		[rad]
	q	pitch rate		[rad/s]
	θ	pitch angle		[rad]
y	θ			[rad]
w	θ_d	desired pitch angle	$\in [-0.5, 0.5]$	[rad]
u	δ	elevator deflection angle	$\in [-1.4, 1.4]$	[rad]
Δt			$= 0.05$	[s]

Table 7 Characterization of the benchmark parameters for evaluation

Parameter	
T	300 time-steps (15 seconds)
N_{max}	50 time-steps (after 2.5 seconds)
μ	0.06 rad
J	50 runs
χ_0^j	constant $= (0, 0, 0)$
w^j	uniformly distributed $\in [-0.5$ rad, 0.5 rad$]$
T_{traj}	1000 time-steps (50 seconds)
$w_{traj}(t)$	step and continuously changing (see A.2.2)

sequence we have a three-dimensional controller state, $x = (\alpha, q, e)$, and a one-dimensional controller action, $u = \delta$.

Though linear, the process dynamic exhibits an interesting property for learning controllers. By changing the elevator deflection angle, δ, the controller can rapidly change the pitch-angle of the aircraft. The angle of attack, α, however, has a very slow dynamic that requires several hundred time-steps to adapt to a control action. Consequently, a good controller only requires a maximum of 20 time-steps to bring the controlled process variable, θ, very close to an arbitrary set-point, θ_d, within the entire working range. However, after reaching the set-point, the controller has to actively keep the controlled process variable as close as possible to the set-point while the process variable, α, slowly changes over several hundred time-steps. Because the length of the trajectories to the desired and stable states are very long, this represents a particularly challenging environment for RL controllers.

4.2.2 Benchmark environment

In order to evaluate the precision and time-optimality of the controllers in terms of $\bar{N}$ and $\bar{e}_\infty$, they are tested on a set of 50 trajectories. Each trajectory has a length of 300 time-steps (15 seconds) and is started at $\chi_0 = (0, 0, 0)$. This corresponds to an aircraft in a steady state where the angle of attack, pitch rate, and pitch angle are all zero. Preceding every trajectory, a set-point between -0.5 rad and 0.5 rad is chosen and kept constant over the entire trajectory (see Table 7).

To evaluate the overall performance of the controller in a typical situation, the criterion e_T is determined on a predefined set-point trajectory. For this trajectory the process is again started in the state $\chi_0 = (0, 0, 0)$. The set-point trajectory is made up of four parts (see Fig. 11). In the first phase, the set-point is kept constant at a value of 0 for 1 second. A perfect controller will keep the starting state in this phase. Afterwards, the set-point is changed to a value of -0.2 for a duration of 6 seconds, in order to represent a typical step that is not

long enough to reach the steady state in the slow process variable. The next part is a linear change of the set-point from -0.2 to 0.2 over 13 seconds, followed by a constant value of 0.2 for the remainder of the trajectory.

4.2.3 Benchmark results

In CTM (1996) a LQR controller design for the pitch control challenge is represented that can serve as a reference controller for benchmarking the performance of learning controllers. For a controller state, $x = (\alpha, q, \theta - \theta_d)$, the control law, $\delta = -Kx$, and certain values for R and Q, the LQR controller design yields a gain vector $K = (-0.6435, 169.6950, 7.0711)$ (see CTM 1996 for details).

For learning a controller with NFQCA and NFQ (parameters in Table 8), interaction trajectories with a length of 300 time-steps (15 seconds) are made. Each interaction trajectory starts from $\chi_0 = (0, 0, 0)$ under a uniformly distributed set-point $\theta_d \in [-0.5 \text{ rad}, 0.5 \text{ rad}]$.

The learning curve for NFQCA is depicted in Fig. 9. After only 14 interaction trajectories and iterations of NFQCA, the resulting controller exhibits a comparable quality to the LQR controller in the criteria e_T on the reference trajectory. The best controller (as outlined in 4.1.3) is learned after 174 interaction trajectories (or iterations of NFQCA). This

Table 8 Parameters for learning the pitch control of the aircraft

x	3 dim.	α	angle of attack
		q	pitch rate
		$\theta_d - \theta$	control deviation
$c(x, u)$	$\tanh^2$	x^d	$(-, -, 0)$
		μ	$(0, 0, 0.06)$
		$\mathcal{C}$	0.01
u	1 dim.	δ	elevator deflection angle
	NFQCA	$\mathcal{U} =$	$[-1.4, 1.4]$
	NFQ	$U =$	$\{\pm 1.4, \pm 0.5, \pm 0.2, \pm 0.1, 0\}$
$Q(x, u)$	neural	topology	4-15-1 (600 epochs RProp)
$\pi(x)$	neural	topology	3-5-1 (400 epochs RProp)

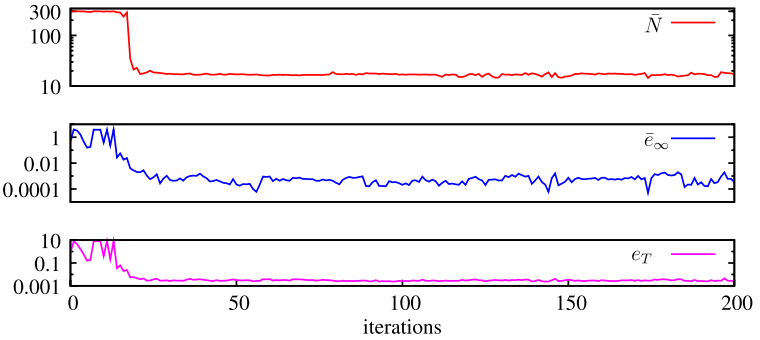

Fig. 9 Learning curves for NFQCA on the pitch control benchmark task. In every iteration 300 interaction samples with the process are collected using the recent learned controller. Afterwards an update of the controller is done with all samples collected so far

Table 9 Benchmark evaluation for the pitch control challenge

Controller	$\bar{N}$	$\bar{e}_\infty$	e_T
LQR	27.33	0.00033	0.00636
NFQ	14.82	0.00032	0.00110
NFQCA	14.51	0.00005	0.00027

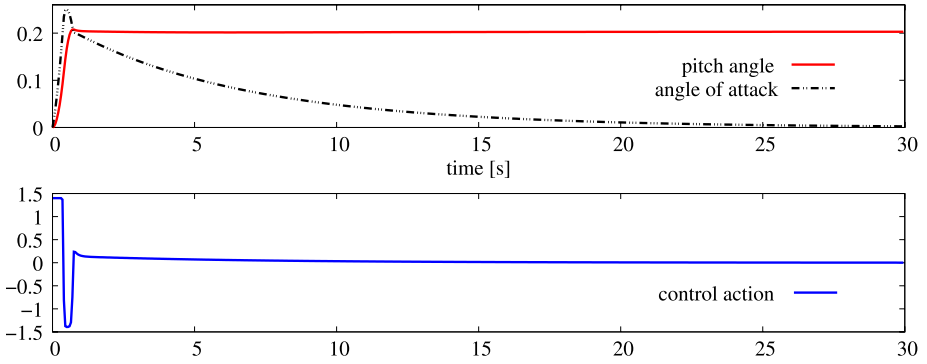

Fig. 10 (Color online) NFQCA result on a set-point step of 0.2. The controller learned with NFQCA on a single set-point step of 0.2 rad. The *solid* (*red*) *line* is the pitch angle of the aircraft. As *dashed dotted* (*black*) *line* the angle of attack is plotted that has a very slow dynamic and is stabilizing after more than 25 seconds

corresponds to 52,200 interaction steps (or 43.5 minutes of interaction with the real-time process). With NFQ we obtain comparable values for the best controller, that was learned after 215 iterations (64,500 interaction steps or 53.75 minutes of interaction with the real-time process). As listed in Table 9, the learning controllers perform better in the criteria $\bar{N}$ than the classically designed LQR controller, where the precision in $\bar{e}_\infty$ is comparable. This is primarily due to a low rise time and an overshoot of the classical LQR controller. In order to achieve a controller design with low rise time and less overshoot, one must search and discover other parameters with which to design a more efficient LQR controller for this control task. In contrast, with the learning controller we need not search for this parameters, as we specify what the controller should achieve and not how it should be achieved. In Fig. 10 a trajectory of the learned NFQCA controller is shown for the pitch control task under a single set-point change. The controlled variable reaches the set-point very quickly with nearly no overshoot. After the set-point is reached, the slow process variable (dashed dotted line) requires more than 25 seconds to stabilize. During this time, the controller has to adapt the control action within a small range such that the controlled process variable is stabilised in close proximity to the set-point. In Fig. 11 the learned controller is shown on the benchmark reference trajectory. As shown, the controller is able to follow the set-point steps as well as the continuous change of the set-points. The evaluation of single set-point changes on each trajectory evaluates the behaviour of the controller starting in a balanced system. In contrast to this, the evaluation of the reference trajectory shows the behaviour of the controller in a broader working range—a range wherein the controller receives a set-point change in unbalanced situations as well.

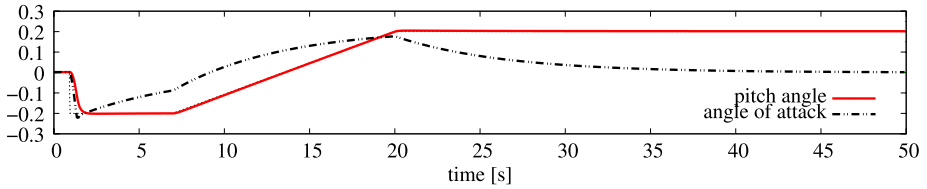

Fig. 11 (Color online) The controller learned with NFQCA on the benchmark reference trajectory. The *solid* (*red*) line is the pitch angle of the aircraft that follows the reference trajectory. As *dashed dotted* (*black*) *line* the angle of attack is plotted that has a very slow dynamic

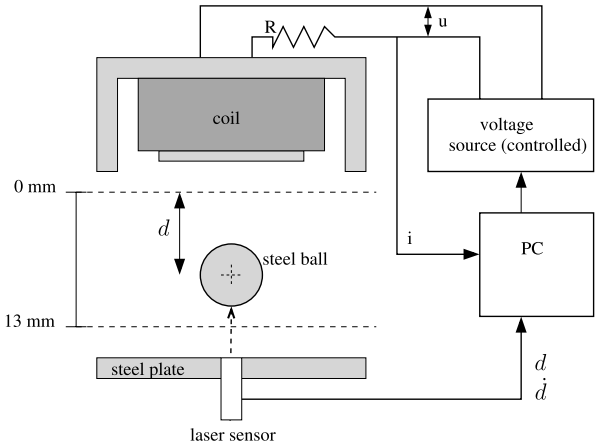

Fig. 12 The process setup for the electromagnetic levitation of a steel ball. Here a computer is depicted as commanding an adjustable voltage source with which to apply a voltage to the coil. The distance between the steel ball and the steel base plate is measured by a laser sensor

4.3 Magnetic levitation of a steel ball

The technique of contact-less positioning of an object with ferromagnetic properties in a controlled electromagnetic field has a wide range of technical applications. Examples for such applications can be found in contact-less bearings, magnetic levitation trains, or contact-less positioning for precise measuring. When it comes to these kinds of applications, the swift and precise positioning of the object represent both the highest priority for nonlinear control design and its most challenging task.

There are many different technologies and setups with which to realise magnetic levitation systems. The system we introduce here representing a benchmark for reinforcement learning feedback control, is a standardized one-dimensional levitation model used to develop nonlinear controllers (proposed in Yang and Minashima 2001). The schematic in Fig. 12 shows the setup of the process. A solenoid can apply forces to a steel ball with mass, M, that is positioned on a steel plate. The control action is the voltage, u, that is applied to the solenoid. The characteristic of the solenoid and the generated electromagnetic field is defined by the parameters R, x_∞, L_∞ and Ξ. For contact-less measuring of the position and velocity of the steel ball a laser sensor is placed under the steel plate. The process variables are given by the position, d, of the steel ball (as the length of the air-gap between the solenoid and the ball), the velocity of the ball, $\dot{d}$, and the current in the coil of the solenoid (see Table 10).

Table 10 The process variables, controlled process variables, set-point, and control action for the magnetic levitation control challenge

χ	d	position of steel ball	$\in [0.000, 0.013]$	[m]
	$\dot{d}$	velocity of steel ball		[m/s]
	I	current in coil		[A]
y	d			
w	d_d	desired position of steel ball	$\in [0.000, 0.013]$	[m]
u	u	applied voltage to coil	$\in [-60, 60]$	[V]
Δt			$= 0.004$	[s]

4.3.1 Control challenge

The quick and precise positioning of the steel ball by applying voltages to the solenoid poses a difficult nonlinear control problem. The open-loop dynamic behaviour is extremely unstable and has a very fast system dynamic. Also, due to the magnetic properties, the amount of nonlinear dynamics is extremely high. Linear controllers can only be developed for single working points and are only valid within a very small proximity to that point. Therefore, advanced nonlinear control design concepts must be carried out with a high amount of design effort and expert knowledge (Yang et al. 2007, 2008; Yang and Tateishi 2001).

For reinforcement learning controllers the amount of nonlinear behaviour in the system dynamics is challenging. Within the control law, and given the strong discontinuities required to lift the ball from the steel plate and to stop it in a time-optimal way at a desired position, these discontinuities are not easy to learn, as a sequence of a few, nearly optimal actions are required. As the open-loop system is extremely unstable, an already levitating ball will easily pass to the upper or lower position if the control law is not appropriate—also for some of the time-steps. For RL methods a good exploration scheme is required that does not lead to the degenerated stable points at the steel plate or solenoid.

4.3.2 Benchmark environment

For the benchmark setup (see Table 12), in every trajectory the process is started with the ball placed at rest on the steel plate ($d = 13$ mm, $\dot{d} = 0$) with no current in the coil. To follow the procedure in Yang and Minashima (2001), a constant voltage of 15 V is applied to the coil for the duration of 0.5 seconds. This voltage is applied in a pre-run phase before each trajectory starts. This initialization phase ensures that the current is in an admissible range and that magnetic saturation is reached. After the initialization phase the controller observes the recent state of the system (the initialisation phase only changes I, not position or velocity of the ball) and the control loop is started.

For the evaluation of $\bar{N}$ and $\bar{e}_\infty$ of the controllers, 50 trajectories are executed, each with a uniformly distributed set-point, $d_d \in [0.000\text{ m}, 0.013\text{ m}]$. Each of the trajectories is 50 time-steps (2 seconds) in duration. To evaluate the overall behaviour the controllers are tested on a reference trajectory with 4,000 time-steps (16 seconds). This reference trajectory starts with the ball resting at the base plate. Every 80 time-steps (0.32 seconds) the set-point is changed (chosen uniformly distributed $\in [0.000\text{ m}, 0.013\text{ m}]$) and stays constant for the subsequent 80 time-steps (see Fig. 15).

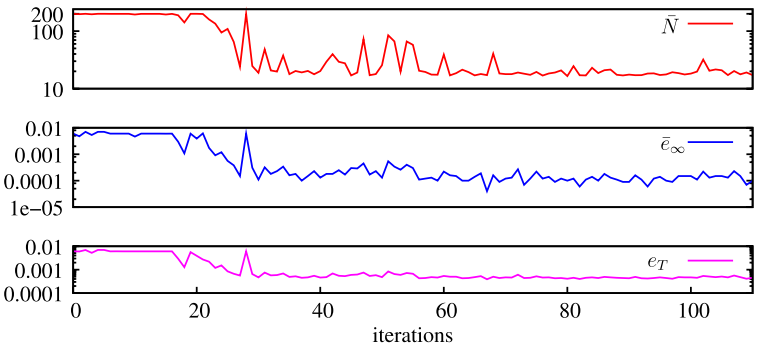

Fig. 13 Learning curves for NFQCA on the magnetic levitation benchmark task. In every iteration 160 interaction samples with the process are collected using the recent learned controller. Afterwards an update of the controller is done with all samples collected so far

Table 11 The learning parameters for the magnetic levitation challenge

x	4 dim.	d	position of ball
		$\dot{d}$	velocity of ball
		I	current in coil
		$d_d - d$	control deviation
$c(x,u)$	$\tanh^2$	x^d	(–, –, –, 0)
		μ	(0, 0, 0, 0.002)
		C	0.01
u	1 dim.	u	voltage applied to coil
$Q(x,u)$	neural	topology	5-15-20-1 (600 epochs RProp)
$\pi(x)$	neural	topology	4-15-1 (1000 epochs RProp)

Table 12 Characterization of the benchmark parameters for evaluation

Parameter	
T	500 time-steps (2 seconds)
N_{max}	50 time-steps (after 0.2 seconds)
μ	0.0005 m
J	50 runs
χ_0^j	constant = (0.013, 0, 0)
w^j	uniformly distributed $\in$ [0.000 m, 0.013 m]
T_{traj}	4000 time-steps (16 seconds)
$w_{traj}(t)$	change every 80 time-steps, uniformly distributed $\in$ [0.000 m, 0.013 m]

4.3.3 Benchmark results

For learning a controller with NFQCA (parameters in Table 11) interaction trajectories with a length of 160 time-steps (0.64 seconds) are made. Each interaction trajectory starts with the ball resting at the steel plate. The set-points for the trajectories are drawn uniformly $\in$ [0.000 m, 0.013 m]. Unlike the other interaction trajectories, the set-point is kept constant

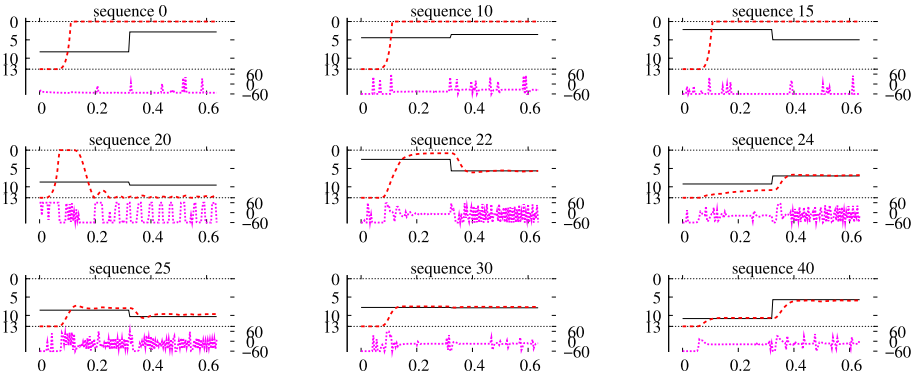

Fig. 14 Examples of interaction trajectories of the learning controller (NFQCA) at different stages (number of interaction sequences) of learning

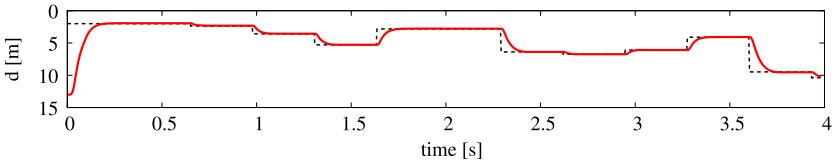

Fig. 15 The learned NFQCA controller applied on the first 4 seconds of the reference trajectory with random changing setpoints

Table 13 Benchmark evaluation of the magnetic levitation challenge

controller	$\bar{N}$	$\bar{e}_\infty$	e_T
NFQCA	17.63	0.0605 mm	0.4 mm

for 80 time-steps, after which it is changed again. This enables the controller to acquire more experience within the entire working range of the process. In Fig. 14, nine interaction trajectories during learning are shown after different numbers of iterations with NFQCA. In the first 20 interaction trajectories the agent has not yet learned to levitate the ball and thus bounces it to the limits of its range of motion. When the learning process continues and more experience is collected alongside additional iterations of NFQCA, the controller improves and learns the positioning of the ball at the desired positions. The learning curve for NFQCA is depicted in Fig. 13. After only 92 iterations and interaction trajectories, NFQCA generated the best (as outlined in 4.1.3) controller. This corresponds to 14,720 interaction steps (or 0.98 minutes interaction with the real-time process). As referenced in Table 13, the precision of the learned controller is very high. In Fig. 15 the first 4 seconds of the learned controller applied to the reference trajectory are shown.

4.4 Heating coil

The control challenge of the heating coil belongs to the set of "Heating, Ventilation, and Air Conditioning" (HVAC) problems. HVAC problems have received much attention in past and recent years (Anderson et al. 1997; Kretchmar 2000), as the existing methods for control

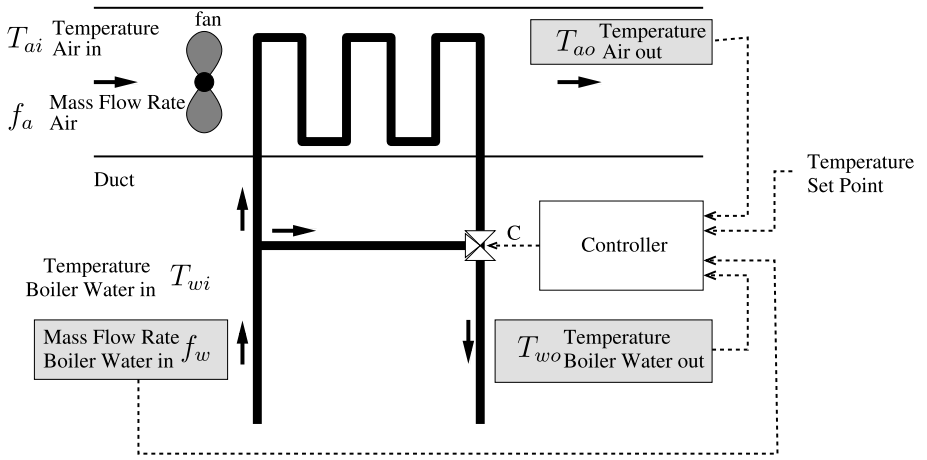

Fig. 16 A typical setup of a Heating Ventilation and Cooling task. The schematic references to the benchmark setup used in this article

leave significant room for improvement. The system dynamic of an HVAC problem typically shows highly nonlinear properties and varies widely at different operating points. In most cases linear model will fail here as a sufficient representation of the real plant dynamics. In addition, the different components of a typical HVAC system (heating coils, fans, valves, etc.) can not be modelled as separate systems as their dynamics are highly coupled. As a consequence, linear control laws are prone to having highly variable gains at different operation points that are difficult to determine with classical control design methods.

Besides the nonlinear dynamic properties of the HVAC process, a very interesting aspect of this is the influence of environment on the dynamic of the process. For example, the change of weather condition, the desired scheduling, or human behaviour can be measured, but hardly predicted. This makes it challenging to determine a control law even with advanced classical control methods. This is the case because these techniques make assumptions about the underlying dynamics and form of the system. The process used as control challenge here is based on a nonlinear model developed in Underwood and Crawford (1991) that was adapted to a real HVAC system by Andrew et al. in Anderson et al. (1997). A schematic diagram of the process is shown in Fig. 16. The dynamic of the process depends on three internal process variables: the flow rate of the incoming boiler water, f_w, the temperature of the incoming boiler water, T_{wo}, and the temperature of the out coming air. This internal process variables are directly influenced by the controller. In addition the process dynamics depends on three external process variables: the temperature of the input air, T_{ai}, the temperature of the water that goes back to the boiler, T_{wi}, and the flow rate of the incoming air, f_a. These external process variables cannot be influenced by the controller as they represent the influence of the environment on the process.

4.4.1 Control challenge

The control task is to set appropriate openings for the control valve, c, such as to have the output air temperature reach as closely as possible to the current set-point, T_d, under different and changing environmental conditions (see Table 14). The change of environmental conditions acts as noise on the system dynamic. As a speciality in this setup, the current

Table 14 The process variables, controlled process variables, set-point and control action for the heating coil control challenge

χ	f_w	flow rate boiler water		[kg/s]
	T_{wo}	temperature boiler water in		[C]
	T_{ao}	temperature air out		[C]
	T_{ai}	temperature air in		[C]
	T_{wi}	temperature boiler water out		[C]
	f_a	flow rate air in		
y	T_{ao}		$\in [40, 45]$	[C]
w	T_d	desired temperature	$\in [0.000, 0.013]$	
u	c	opening of valve	$\in [670, 1400]$	

Table 15 Characterization of the benchmark parameters for evaluation

Parameter	
T	200 time-steps
N_{max}	50 time-steps
μ	$1C$
J	50 runs
χ_0^j	$f_w = 0.128\ T_{wo} = 43.24\ T_{ao} = 40.1$
w^j	uniformly distributed $\in [40, 45]$
T_{traj}	500 time-steps (16 seconds)
$w_{traj}(t)$	three steps from $40C$ to $45C$

external process variables are not influenced by the controller, however, can be measured. This scenario represents a nonlinear process dynamic with a high amount of noise. From the perspective of both RL and classical control, it is a challenging task to produce time-optimal and robust controllers that work precisely within a wide range of operating points.

4.4.2 Benchmark environment

For the benchmark setting (see Table 15), the variables, T_{ai}, T_{wi}, and, f_a, are modified by random walk every 30 time-steps in order to model the disturbances and changing conditions that would occur in actual heating and air-conditioning systems. The admissible ranges for the random walk are $4 \leq T_{ai} \leq 12C$, $73 \leq T_{wi} \leq 81C$ and $0.7 \leq f_a \leq 0.9$ kg/s.

For the evaluation of $\bar{N}$ and $\bar{e}_\infty$ of the controllers, 50 trajectories are executed, each with a uniformly distributed set-point, $T_d \in [40C, 45C]$. Each of the trajectories is 200 time-steps long and starts with $f_w = 0.128$, $T_{wo} = 43.24$, and $T_{ao} = 40.1$. The external process variables are chosen randomly within the admissible range. To evaluate the overall behaviour, the controllers are tested on a reference trajectory with 500 time-steps. This reference trajectory starts with the same initial conditions as the other evaluation trajectories.

4.4.3 Benchmark results

The learning curve for NFQCA is depicted in Fig. 17. Learning a controller with NFQCA (parameters in Table 16) requires only 163 interaction trajectories and iteration of the algorithm. This corresponds to 32,600 interaction steps. In Table 17, the performance of the learned controller is compared to a reference PI controller developed in Kretchmar (2000). In Fig. 18 the reference PI controller is shown when executed on the reference trajectory. In

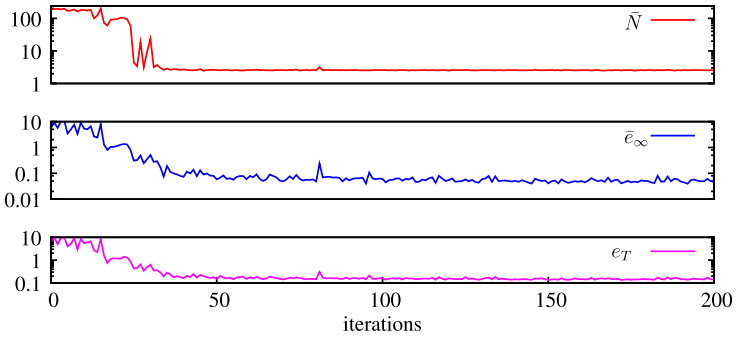

Fig. 17 Learning curves for NFQCA on the heating coil benchmark task. In every iteration 200 interaction samples with the process are collected using the recent learned controller. Afterwards an update of the controller is done with all samples collected so far

Table 16 The learning parameters for the heating coil control task

x	7 dim.	f_w	low rate boiler water
		T_{wo}	temperature boiler water in
		T_{ao}	temperature air out
		T_{ai}	temperature air in
		T_{wi}	temperature boiler water out
		f_a	flow rate air in
		$T_{ao}^d - T_{ao}$	control deviation
$c(x,u)$	$\tanh^2$		
		μ	1.0
		$\mathcal{C}$	0.01
u	1 dim.	u	valve state
$Q(x,u)$	neural	topology	8-10-10-1 (600 epochs RProp)
$\pi(x)$	neural	topology	4-15-1 (1000 epochs RProp)

Table 17 Benchmark evaluation of the heating coil control challenge

controller	$\bar{N}$	$\bar{e}_\infty$	e_T
PI	10.42	0.270	0.2057
NFQCA	2.49	0.044	0.1401

contrast to the learned controller in Fig. 19, the PI controller has a high overshoot and reacts on the external noise very slow. The controller learned with NFQCA exhibits a very good, time-optimal behaviour and can react on the external changes very quickly.

5 Discussion

The contribution of this article is the presentation of four benchmarking scenarios for reinforcement learning in the field of technical process control. From the perspective of both classical control and reinforcement learning, the presented benchmark settings exhibit intriguing and challenging attributes. For the benchmark, all of the information is provided

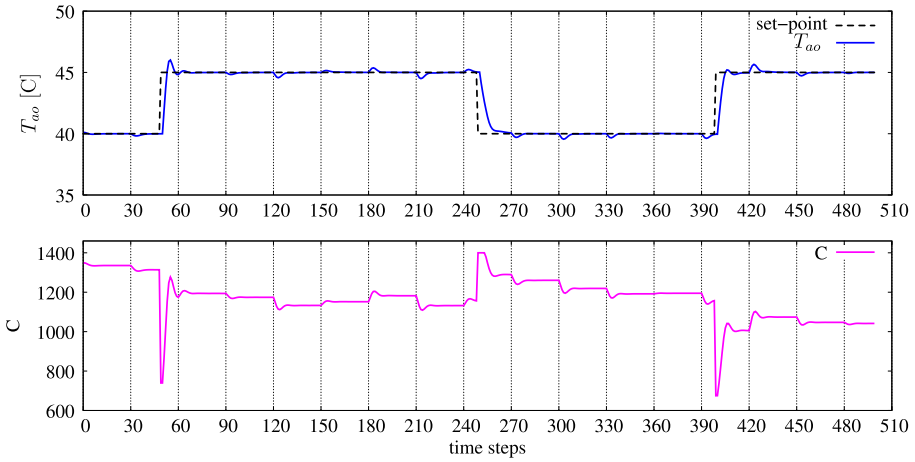

Fig. 18 PI controller result on the reference trajectory of the heating coil control challenge

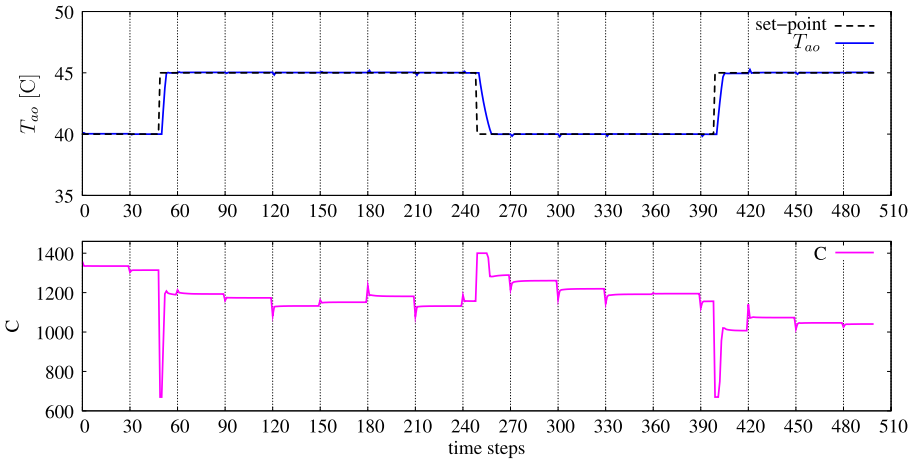

Fig. 19 NFQCA controller result on the reference trajectory of the heating coil control challenge

such as to allow and encourage the implementation of the benchmark settings and, therefore, the cataloguing of results using other methods. The presented benchmark environments and evaluation setups will be implemented in the next official release of our software package for benchmarking the "Closed Loop System Simulator (CLSquare)".[2]

The proposed quantitative performance measures for the quality of the learned controller and the learning performance allows for a comparison between different control methods. The comparison of application-specific, classical control methods against learning controllers, as well as the comparison of different learning methods, is possible.

[2]Available at www.ml.uni-freiburg.de.

By reporting the results of our own reinforcement learning algorithm—the Neural Fitted Q-Iteration with Continuous Actions (NFQCA)—on the four presented benchmark problems, a baseline for other benchmarking results is given. The evaluation of the NFQCA algorithm within the benchmarks showed a very efficient learning behaviour. With an amount of interaction that falls within the range of a few hundred (or even under one hundred) of interaction trajectories (corresponding to minutes of interaction on a real time process), the application to real-world systems and problems comes into view. The benchmark results show that we are able to learn high quality, continuous and nonlinear control laws with NFQCA. It is certainly worth noting that what the benchmarks showed, is that we can learn these high quality, continuous control laws with the same amount of interactions that are needed for the NFQ algorithm with discrete actions.

Appendix: Benchmark details

A.1 Underwater vehicle

A.1.1 System dynamics

The system dynamic of the underwatervehicle is given by the dynamic equation:

$$\begin{aligned} \dot{v} &= f(v,u) = \frac{u \cdot k(v,u) - c(v) \cdot v \cdot |v|}{m(v)} \\ c(v) &= 1.2 + 0.2 \cdot \sin(|v|) \\ m(v) &= 3.0 + 1.5 \cdot \sin(|v|) \\ k(v,u) &= -0.5 \cdot \tanh[(|(c \cdot v \cdot |v|) - u| - 30.0) \cdot 0.1] + 0.5 \end{aligned} \tag{6}$$

For simulating the magnetic levitation system we use the Runge Kutta 4 (RK4) numeric integration scheme with 2 intermediate steps.

A.1.2 Benchmark details

To evaluate a controller in the underwater vehicle benchmark the reference trajectory is depicted in Table 18.

A.2 Pitch control

A.2.1 System dynamics

Given the process state, $\chi = [\alpha, q, \theta]^T$, and control action, δ, the dynamic of the process is described by:

$$\begin{aligned} \dot{\alpha} &= -0.313\alpha + 56.7q + 0.232\delta \\ \dot{q} &= -0.0139\alpha - 0.426q + 0.0203\delta \\ \dot{\Theta} &= 56.7q \end{aligned} \tag{7}$$

For simulating the pitch control system we use the Runge Kutta 4 (RK4) numeric integration scheme with 10 intermediate steps.

Table 18 A characterization of the reference trajectory for the underwater vehicle. The reference trajectory is a linear interpolation between these points—constant when no point behind is given (compare Fig. 7)

t [s]	t [steps]	$w_{traj}(t)$
0	0	1
1	33	1
4	133	3
5	166	3
5	167	2
8	266	2
8	267	−3
9	300	−3
12	400	−1
14	466	−3
14.1	470	3
22	733	0

Table 19 A characterization of the reference trajectory for the pitch control benchmark. The reference trajectory is a linear interpolation between these points—constant when no point behind is given (compare Fig. 11)

t [s]	t [steps]	$w_{traj}(t)$
0	0	0
1	20	0
1	20	−0.2
7	140	−0.2
20	400	0.2
50	1000	0.2

A.2.2 Benchmark details

The reference trajectory is defined by linear interpolations with intermediate points, given in Table 19.

A.3 Magnetic levitation of a steel ball

A.3.1 System dynamics

Given the process state, $\chi = [\chi_1, \chi_2, \chi_3]^T = [d, \dot{d}, I]^T$, and control action, u, the dynamic of the process is described by:

$$\begin{pmatrix} \dot{\chi}_1 \\ \dot{\chi}_2 \\ \dot{\chi}_3 \end{pmatrix} = \begin{pmatrix} \chi_2 \\ \alpha(\chi) \\ \beta(\chi) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \gamma(\chi) \end{pmatrix} u \tag{8}$$

$$\alpha(\chi) = g - \frac{\xi \chi_3^2}{2M(x_\infty + \chi_1)^2} \tag{9}$$

$$\beta(\chi) = \frac{\chi_3(\xi \chi_2 - R(x_\infty + \chi_1)^2)}{\xi(x_\infty + \chi_1) + L_\infty(x_\infty + \chi_1)^2} \tag{10}$$

$$\gamma(\chi) = \frac{x_\infty + \chi_1}{\xi + L_\infty(x_\infty + \chi_1)} \tag{11}$$

Table 20 The simulation parameters for the magnetic levitation system

mass of steel ball	$M = 0.8$	[kg]
electrical resistance	$R = 11.68$	[Ω]
coil parameter	$x_\infty = 0.007$	[m]
coil parameter	$L_\infty = 0.8052$	[H]
coil parameter	$\xi = 0.001599$	[Hm]
process action	$u \in [-60, 60]$	[V]
control interval	$\Delta_t = 0.004$	[s]

For simulating the magnetic levitation system we use the Runge Kutta 4 (RK4) numeric integration scheme with 2 intermediate steps and the system parameters given in Table 20.

A.4 Heating coil

A.4.1 System dynamics

Given the process state, $\chi = [\chi_1, \chi_2, \chi_3, \chi_4, \chi_5, \chi_6]^T = [f_w, T_{wo}, T_{ao}, T_{ai}, T_{wi}, f_a]^T$, and control action, u, the time discrete dynamic of the internal process variables is described by:

$$\chi_1(t+1) = 6.7210^{-10}u(t)^3 - 2.3010^{-6}u(t)^2 + 2.1810^{-3}u(t) - 0.2823 \tag{12}$$

$$\begin{aligned}\chi_2(t+1) = {} & \chi_2(t) + 0.649\chi_1(t)\chi_5(t) - 0.649\chi_1(t)\chi_2(t) \\ & - 0.012\chi_5(t+1) - 0.012\chi_2(t) + 0.023\chi_4(t+1) + 0.104\chi_1(t)\chi_4(t+1) \\ & - 0.052\chi_1(t)\chi_5(t+1) - 0.052\chi_1(t)\chi_2(t) + 0.028\chi_6(t+1)\chi_4(t+1) \\ & - 0.014\chi_6(t+1)\chi_5(t+1) - 0.014\chi_6(t+1)\chi_2(t)\end{aligned} \tag{13}$$

$$\begin{aligned}\chi_3(t+1) = {} & \chi_3(t) + 0.197 * \chi_6(t+1) * \chi_4(t+1) - 0.197 * \chi_6(t+1) * \chi_3(t) \\ & + 0.016 * \chi_5(t+1) + 0.016 * \chi_2(t) - 0.032 * \chi_4(t+1) \\ & + 0.077 * \chi_1(t) * \chi_5(t+1) + 0.077 * \chi_1(t) * \chi_2(t) \\ & - 0.015 * \chi_1(t) * \chi_4(t+1) + 0.022 * \chi_6(t+1) * \chi_5(t+1) \\ & + 0.022 * \chi_6(t+1) * \chi_2(t) - 0.045 * \chi_6(t+1) * \chi_4(t+1) \\ & + 0.206 * \chi_4(t) - 0.206 * \chi_4(t+1)\end{aligned} \tag{14}$$

The dynamic of the external process variables are functions over time, $\chi_4(t)$, $\chi_5(t)$, $\chi_6(t)$, and are dependent on the benchmark setting.

References

Anderson, C., & Miller, W. (1990). Challenging control problems. In *Neural networks for control* (pp. 475–410).

Anderson, C. W., Hittle, D., Katz, A., & Kretchmar, R. M. (1997). Synthesis of reinforcement learning, neural networks, and pi control applied to a simulated heating coil. *Journal of Artificial Intelligence in Engineering*, *11*(4), 423–431.

Bellman, R. (1957). *Dynamic programming*. Princeton: Princeton Univ Press.

Boyan, J., & Littman, M. (1994). Packet routing in dynamically changing networks—a reinforcement learning approach. In J. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems 6*.

Crites, R. H., & Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In: *Andvances in neural information processing systems 8*.

CTM (1996). Digital Control Tutorial. University of Michigan, www.engin.umich.edu/group/ctm (online).

Deisenroth, M., Rasmussen, C., & Peters, J. (2009). Gaussian process dynamic programming. *Neurocomputing*, *72*(7–9), 1508–1524.

Dullerud, G. P. F. (2000). *A course in robust control theory: A convex approach*. New York: Springer.

El-Fakdi, A., & Carreras, M. (2008). Policy gradient based reinforcement learning for real autonomous underwater cable tracking. In *International conference on intelligent robots and systems, 2008. IROS 2008. IEEE/RSJ* (pp. 3635–3640).

Farrel, J. A., & Polycarpou, M. M. (2006). *Adaptive approximation based control*. New York: Wiley Interscience.

Gabel, T., & Riedmiller, M. (2008). Adaptive reactive job-shop scheduling with reinforcement learning agents. *International Journal of Information Technology and Intellifent Computing*, *24*(4).

Goodwin, G. C., & Payne, R. L. (1977). *Dynamic system identification: experiment design and data analysis*. New York: Academic Press.

Hafner, R. (2009). *Dateneffiziente selbstlernende neuronale Regler*. PhD thesis, University of Osnabrueck.

Hafner, R., & Riedmiller, M. (2007). Neural reinforcement learning controllers for a real robot application. In *Proceedings of the IEEE international conference on robotics and automation (ICRA 07)*, Rome, Italy.

Jordan, M. I., & Jacobs, R. A. (1990). Learning to control an unstable system with forward modeling. In D. Touretzky (Ed.), *Advances in neural information processing systems (NIPS) 2* (pp. 324–331). San Mateo: Morgan Kaufmann.

Kaloust, J., Ham, C., & Qu, Z. (1997). Nonlinear autopilot control design for a 2-dof helicopter model. *IEE Proceedings. Control Theory and Applications*, *144*(6), 612–616.

Kretchmar, R. M. (2000). *A synthesis of reinforcement learning and robust control theory*. PhD thesis, Colorado State University, Fort Collins, CO.

Krishnakumar, K., & Gundy-burlet, K. (2001). *Intelligent control approaches for aircraft applications* (Technical report). National Aeronautics and Space Administration, Ames Research.

Kwan, C., Lewis, F., & Kim, Y. (1999). Robust neural network control of rigid link flexible-joint robots. *Asian Journal of Control*, *1*(3), 188–197.

Liu, D., Javaherian, H., Kovalenko, O., & Huang, T. (2008). Adaptive critic learning techniques for engine torque and air-fuel ratio control. *IEEE Transactions on Systems, Man and Cybernetics. Part B. Cybernetics*, *38*(4), 988–993.

Ljung, L. (1999). *System identification theory for the user* (2nd ed.). Upper Saddle River: PTR Prentice Hall.

Martinez, J. J., Sename, O., & Voda, A. (2009). Modeling and robust control of blu-ray disc servomechanisms. *Mechatronics*, *19*(5), 715–725.

Nelles, O. (2001). *Nonlinear system identification*. Berlin: Springer.

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., & Liang, E. (2004). Inverted autonomous helicopter flight via reinforcement learning. In *International symposium on experimental robotics*.

Peters, J., & Schaal, S. (2006). Policy gradient methods for robotics. In *Proceedings of the IEEE international conference on intelligent robotics systems (Iros 2006)*.

Prokhorov, D., & Wunsch, D. (1997). Adaptive critic designs. *IEEE Transactions on Neural Networks*, *8*, 997–1007.

Riedmiller, M. (2005). Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method. In *Proc. of the European conference on machine learning, ECML 2005*, Porto, Portugal.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In H. Ruspini (Ed.), *Proceedings of the IEEE international conference on neural networks (ICNN)*, San Francisco (pp. 586–591).

Riedmiller, M., Hafner, R., Lange, S., & Timmer, S. (2006). Clsquare—software framework for closed loop control. Available at http://ml.informatik.uni-freiburg.de/research/clsquare.

Riedmiller, M., Montemerlo, M., & Dahlkamp, H. (2007a). Learning to drive in 20 minutes. In *Proceedings of the FBIT 2007 conference*, Jeju, Korea. Berlin: Springer. Best Paper Award.

Riedmiller, M., Peters, J., & Schaal, S. (2007b). Evaluation of policy gradient methods and variants on the cart-pole benchmark. In *Proceedings of the IEEE international symposium on approximate dynamic programming and reinforcement learning (ADPRL 07)*, Honolulu, USA.

Riedmiller, M., Gabel, T., Hafner, R., & Lange, S. (2009). Reinforcement learning for robot soccer. *Autonomous Robots*, *27*(1), 55–74.

Schiffmann, W., Joost, M., & Werner, R. (1993). Comparison of optimized backpropagation algorithms. In *Proc. of ESANN'93*, Brussels (pp. 97–104).

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Deylon, B., Glorennec, Y. P., Hjalmarsson, H., & Juditsky, A. (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, *31*, 1691–1724.

Slotine, J. E., & Li, W. (1991). *Applied nonlinear control*. New York: Prentice Hall.

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

Szepesvari, C. (2009). Successful application of rl. Available at http://www.ualberta.ca/szepesva/RESEARCH/RLApplications.html.

Tanner, B., & White, A. (2009). RL-Glue: language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research*, *10*, 2133–2136.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, *8*, 257–277.

Tesauro, G., Chess, D. M., Walsh, W. E., Das, R., Segal, A., Whalley, I., Kephart, J. O., & White, S. R. (2004). A multi-agent systems approach to autonomic computing. In *AAMAS '04: Proceedings of the third international joint conference on autonomous agents and multiagent systems* (pp. 464–471). Washington: IEEE Computer Society.

Underwood, D. M., & Crawford, R. R. (1991). Dynamic nonlinear modeling of a hot-water-to-air heat exchanger for control applications. *ASHRAE Transactions*, *97*(1), 149–155.

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

Watkins, C. J. (1989). *Learning from delayed rewards*. PhD thesis, Cambridge University.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292.

Whiteson, S., Tanner, B., & White, A. (2010). The reinforcement learning competitions. *The AI Magazine*, *31*(2), 81–94.

Yang, Z.-J., & Minashima, M. (2001). Robust nonlinear control of a feedback linearizable voltage-controlled magnetic levitation system. *Transactions of the Institute of Electrical Engeneers of Japan*, 1203–1211.

Yang, Z.-J., & Tateishi, M. (2001). Adaptive robust nonlinear control of a magnetic levitation system. *Automatica*, *37*(7), 1125–1131.

Yang, Z.-J., Tsubakihara, H., Kanae, S., & Wada, K. (2007). Robust nonlinear control of a voltage-controlled magnetic levitation system using disturbance observer. *Transactions of IEE of Japan*, *127-C*(12), 2118–2125.

Yang, Z.-J., Kunitoshi, K., Kanae, S., & Wada, K. (2008). Adaptive robust output feedback control of a magnetic levitation system by k-filter approach. *IEEE Transactions on Industrial Electronics*, *55*(1), 390–399.