Continuous-Time Adaptive Critics

Thomas Hanselmann, *Member, IEEE*, Lyle Noakes, and Anthony Zaknich, *Senior Member, IEEE*

***Abstract*—A continuous-time formulation of an adaptive critic design (ACD) is investigated. Connections to the discrete case are made, where backpropagation through time (BPTT) and real-time recurrent learning (RTRL) are prevalent. Practical benefits are that this framework fits in well with plant descriptions given by differential equations and that any standard integration routine with adaptive step-size does an adaptive sampling for free. A second-order actor adaptation using Newton's method is established for fast actor convergence for a general plant and critic. Also, a fast critic update for concurrent actor–critic training is introduced to immediately apply necessary adjustments of critic parameters induced by actor updates to keep the Bellman optimality correct to first-order approximation after actor changes. Thus, critic and actor updates may be performed at the same time until some substantial error build up in the Bellman optimality or temporal difference equation, when a traditional critic training needs to be performed and then another interval of concurrent actor–critic training may resume.**

***Index Terms*—Actor–critic adaptation, adaptive critic design (ACD), approximate dynamic programming, backpropagation through time (BPTT), continuous adaptive critic designs, real-time recurrent learning (RTRL), reinforcement learning, second-order actor adaptation.**

I. Introduction

THERE are many terminologies used for adaptive critic designs (ACDs) depending on how the problem is viewed, but basically ACDs represent a framework for dynamic programming approximation and they are used in decision making with the objective of minimal long-term cost. ACDs approximate dynamic programming by parameterizing the long-term cost, $J(\mathbf{x})$ [heuristic dynamic programming (HDP)], or its derivative [λ-critic, dual heuristic programming (DHP)], or a combination thereof [global dual heuristic programming (GDHP)]. Other versions are also used, especially reinforcement learning, which was inspired from a biological view point. There are only a few publications dealing with continuous-time adaptive critics [1]–[5]. This paper is an expansion of [5] and contains all the necessary equations to implement the proposed method, which is an extension of the discrete ACD approach to continuous-time systems of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}) \quad \text{system equations} \tag{1}$$

$$\mathbf{u} = \hat{\mathbf{g}}(\mathbf{x};\mathbf{w}_a) \quad \text{control equations} \tag{2}$$

with the objective of a minimal long-term cost function, given by (3) and to find a suitable controller (2). $\mathbf{x}$ denotes the system's state vector

$$\min_{\mathbf{u}\in\mathcal{U}} J(\mathbf{x}) = \min_{\mathbf{u}\in\mathcal{U}} \int_{t_0}^{\infty} \phi(\mathbf{x},\dot{\mathbf{x}})dt = \min_{\mathbf{u}\in\mathcal{U}} \int_{t_0}^{\infty} \tilde{\phi}(\mathbf{x},\mathbf{u})dt. \tag{3}$$

There is no space here to introduce backpropagation through time (BPTT) and real-time recurrent learning (RTRL) in depth; their details can be found in [6]–[8] and [9]. BPTT calculates total derivatives of a quantity that is a function of previously evaluated functions with respect to some previous argument, as seen in (4)–(6). RTRL calculates total derivatives $d\mathbf{x}_n^T(t)/d\mathbf{w}$ forward in time based on a transition matrix $\Phi(t)$. In the context of ACDs, function approximators are used like neural networks for plant identification, actor and critic modules, or they are all part of a large network. Then, the state $\mathbf{x}_n(t)$ refers to all the nodes in a network and its dimensionality can be quite large.

Parameters are denoted generically as $\mathbf{w}$ and to distinguish between actor and critic parameters, subscripts a and c are used. The actor, or controller, is given by (2), whereas the critic tries to estimate the quantity (3) by $\hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x};\mathbf{w}_c)$, where the superscript $\pi(\mathbf{w}_a)$ indicates the policy given controller (2) and determines the set $\mathcal{U}$ of possible controls.

A. ACD Review

The broad range of nomenclature, approximate dynamic programming, reinforcement learning, temporal difference learning, adaptive critics designs, and a group of names by Werbos, HDP, DHP, GDHP, and its action depend (AD) forms share many commonalities and differences are often a matter of taste. Werbos distinguishes designs based on properties of the critic: scalar valued long-term cost falls into HDP, whereas DHP uses the derivative of the long-term cost with respect to the state which is often more powerful because as a vector quantity it tells how the long-term cost will change depending on a state change. However, as not every vector-field is integrable, DHP may yield a solution that is not consistent with HDP. The proper combination yields GDHP which is a second-order method because its derivative estimate is integrable and consistent with the long-term cost estimator. More on these designs including convergence analysis is given in [3]. The best overall reference on the topic is currently [10] which is an exhaustive resource, particularly [10, Ch. 3 and 15] relate to this paper (see, also, [11]). Ferrari [10, Ch. 3] also gives a convergence proof for a standard ACD based on [12], which may be seen as first ACD. This proof is based on exact functional representation of the long-term cost and only recently have people found convergence proofs for ACDs with parameterized estimators. For nice mathematical convergence proofs of ACDs in the context of approximate policy iteration, see [13] where it is shown that the approximate cost-to-go function will be obtained within certain error tolerances from

Manuscript received July 18, 2005; revised April 14, 2006; accepted September 28, 2006. This work was supported in part by the Australian Research Council (ARC).

T. Hanselmann is with the Department of Electrical and Electronic Engineering, the University of Melbourne, Parkville, Vic. 3010, Australia (e-mail: t.hanselmann@ee.unimelb.edu.au).

L. Noakes is with the School of Mathematics and Statistics, the University of Western Australia, Crawley, W.A. 6009, Australia (e-mail: lyle@maths.uwa.edu.au).

A. Zaknich is with the School of Engineering Science, Murdoch University, Perth, W.A. 6150, Australia (e-mail: tonko@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2006.889499

the optimal one. Most notably is [14] where convergence for linear cost functions has been proven when sampled according to the steady-state probability distributions. Recently, this result was extended by Xu *et al.* [15] to nonlinear cost functions by using a nonlinear kernel-mapping from an input space to a high-dimensional but linear space, as done with support vector machines. There is a vast literature on temporal difference methods in conjunction with Markov chains where $TD(\lambda)$ [16], [17] and state-action-reward-state-action *(SARSA)* algorithm, the sequence undergone in the evaluation process, are some basic and powerful tools. For a good overview, see [18].

II. Continuous Version of "Ordered" Total Derivatives

A simple method for the calculation of total derivatives for ordered systems, defined by (4), was achieved by discretizing the continuous plant and utility or short-term cost and treating them as ordered systems, where total derivatives can be easily calculated by the formulas (5) or (6). This implements the chain rule and was first introduced by Werbos in the context of adapting parameters of a long-term cost function [19]. The notation $\partial^+ z_n/\partial z_k$ means the total derivative

$$z_i = z_i(z_1, z_2, \ldots, z_{i-1}) \qquad \forall z_i, 1 \le i \le n \tag{4}$$

$$\frac{\partial^+ z_n^T}{\partial z_k} = \frac{\partial z_n^T}{\partial z_k} + \sum_{j=k+1}^{n-1} \frac{\partial z_j^T}{\partial z_k}\frac{\partial^+ z_n^T}{\partial z_j} \tag{5}$$

$$= \frac{\partial z_n^T}{\partial z_k} + \sum_{j=k+1}^{n-1} \frac{\partial^+ z_j^T}{\partial z_k}\frac{\partial z_n^T}{\partial z_j}. \tag{6}$$

The chain rule can be applied analogously for continuous systems where $\mathbf{x}(t)$ represents the state of the system and is under the influence of infinitesimal changes during the infinitesimal time step dt. Given the setup of an adaptive critic design where $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}; \mathbf{w}))$, the goal is to adapt the weights $\mathbf{w}$ such that $\mathbf{x}$ is an optimal trajectory, in the sense that it has a minimal long-term cost. Clearly, $\dot{\mathbf{x}}$ can be seen as a function of only $\mathbf{x}$ and $\mathbf{w}$, so $\dot{\mathbf{x}} = \mathbf{h}(\mathbf{x}; \mathbf{w})$. A deviation $\delta\mathbf{w}$ in $\mathbf{w}$ leads to a deviation in the trajectory $\mathbf{x}$, say $\mathbf{x}_{\delta\mathbf{w}}$. Therefore, (7) holds, $\dot{\mathbf{x}} = \mathbf{h}(\mathbf{x}; \mathbf{w}) = \mathbf{f}(\mathbf{x}, \mathbf{u})$, and the order of the differentiations can be exchanged (see Fig. 1) as defined by

$$\frac{d}{dt}\left(\delta\mathbf{w}^T \frac{d\mathbf{x}^T}{d\mathbf{w}}\right) = \mathbf{h}^T(\mathbf{x}_{\delta\mathbf{w}}; \mathbf{w} + \delta\mathbf{w}) - \mathbf{h}^T(\mathbf{x}; \mathbf{w}) \tag{7}$$

$$\frac{d}{dt}\frac{d\mathbf{x}^T}{d\mathbf{w}} = \frac{d\mathbf{h}^T}{d\mathbf{w}} = \frac{d\mathbf{x}^T}{d\mathbf{w}}\frac{\partial \mathbf{h}^T}{\partial \mathbf{x}} + \frac{\partial \mathbf{h}^T}{\partial \mathbf{w}} \tag{8}$$

$$= \frac{d\mathbf{x}^T}{d\mathbf{w}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{x}} + \frac{d\mathbf{g}^T}{d\mathbf{w}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}} \tag{9}$$

$$= \frac{d\mathbf{x}^T}{d\mathbf{w}}\left[\frac{\partial \mathbf{f}^T}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}^T}{\partial \mathbf{x}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}}\right] + \frac{\partial \mathbf{g}^T}{\partial \mathbf{w}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}}. \tag{10}$$

This relation proves to be very useful as it is just a differential equation, which can easily be integrated for the otherwise hard to calculate total derivative $d\mathbf{x}^T/d\mathbf{w}$. Using a new variable $\mathbf{q}$, the differential equation can be rewritten as defined by (11)–(13), ready to be solved by a standard integration routine

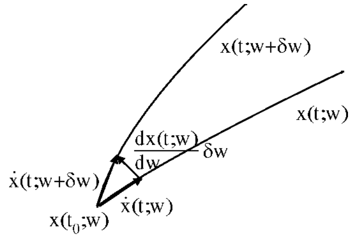

Fig. 1. Connection between neighboring trajectories due to a slight change in the weights. Multiplying all the vectors by δt makes it clear that the order of derivatives with respect to time and weights can be exchanged [see (7)].

$$\mathbf{q} := \frac{d\mathbf{x}^T}{d\mathbf{w}} \tag{11}$$

$$\dot{\mathbf{q}} = \mathbf{q}\left[\frac{\partial \mathbf{f}^T}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}^T}{\partial \mathbf{x}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}}\right] + \frac{\partial \mathbf{g}^T}{\partial \mathbf{w}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}} \tag{12}$$

with initial condition

$$\mathbf{q}(t_0) = \mathbf{0}. \tag{13}$$

If this is expressed in an integral form, the similarity with the discrete ordered system is easily seen. In the discrete system a summation is performed over the later dependencies of a quantity whose target sensitivity is calculated, whereas here an integration has to be performed, where the same total and partial derivatives appear, but only at infinitesimal time steps as defined by

$$\mathbf{x}(t_1) = \mathbf{x}(t_0) + \int_{t_0}^{t_1} \mathbf{f}(\mathbf{x}(t), \hat{\mathbf{g}}(\mathbf{x}(t); \mathbf{w}_a))dt \tag{14}$$

$$\frac{d\mathbf{x}^T(t_1)}{d\mathbf{w}_a} = \int_{t_0}^{t_1} \frac{d\mathbf{f}^T(\mathbf{x}(t), \hat{\mathbf{g}}(\mathbf{x}(t); \mathbf{w}_a))}{d\mathbf{w}_a}dt \tag{15}$$

$$= \int_{t_0}^{t_1} \left[\frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{\partial \mathbf{f}^T}{\partial \mathbf{x}} + \frac{d\hat{\mathbf{g}}^T}{d\mathbf{w}_a}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}}\right] dt \tag{16}$$

$$= \int_{t_0}^{t_1} \left[\frac{d\mathbf{x}^T}{d\mathbf{w}_a}\left(\frac{\partial \mathbf{f}^T}{\partial \mathbf{x}} + \frac{\partial \hat{\mathbf{g}}^T}{\partial \mathbf{x}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}}\right) + \frac{\partial \hat{\mathbf{g}}^T}{\partial \mathbf{w}_a}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}}\right] dt. \tag{17}$$

Again, this is the integral formulation of the differential (12) with initial condition (13) and $d\mathbf{x}^T/d\mathbf{w}_a =: \mathbf{q}$.

Therefore, the summation in (6) is exchanged by integration and the partial derivative has to be included into the integral. This is not surprising, because in the discrete case total derivatives of intermediate quantities are calculated recursively by the same formula (6). Instead of being a discrete ordered system, it is a distributed (over time) and ordered (structural dependencies) system in the continuous case, where infinitesimal changes are expressed in terms of total time derivatives of the target quantity $\dot{\mathbf{x}} = \mathbf{f}$ and split into total and partial derivatives, for indirect and direct influence on the target quantity, just as in the discrete case.

This trick of solving for a total derivative ($d\mathbf{x}(t)/d\mathbf{w}_a$) by integration is the key to continuous-time adaptive critics.

A. *Continuous-Time Adaptive Critics*

For continuous-time adaptive critics the plant and the cost-density function are continuous and the one-step (or short-term) cost is an integral of a cost-density function over a time interval $[t_0,\ t_1]$, given by

$$U(t_0,t_1) = \int_{t_0}^{t_1} \phi(\mathbf{x},\dot{\mathbf{x}})dt. \tag{18}$$

Given a long-term cost estimator (19), called a critic, with some parameters $\mathbf{w}_c$ which depend on the policy $\pi(\mathbf{w}_a) : \mathbf{x}(t) \mapsto \hat{\mathbf{g}}(\mathbf{x}(t);\mathbf{w}_a)$

$$\hat{J}(\mathbf{x}(t_1);\mathbf{w}_c) := \hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1);\mathbf{w}_c) \approx \int_{t_1}^{\infty} \phi(\mathbf{x}(t),\dot{\mathbf{x}}(t))dt. \tag{19}$$

As seen before, in adaptive critic designs an estimator is sought that is optimal with respect to its control output $\mathbf{u}$, and, respectively, to its parameters $\mathbf{w}_a$. Using Bellman's principle of optimality (21), (22) must hold and two objectives can be achieved simultaneously [12]

$$\begin{aligned}\hat{J}(\mathbf{x}(t_0);\mathbf{w}_c) &:= \hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_0);\mathbf{w}_c) && (20)\\ &\approx \int_{t_0}^{t_1} \phi(\mathbf{x}(t),\dot{\mathbf{x}}(t))dt + \int_{t_1}^{\infty} \phi(\mathbf{x}(t),\dot{\mathbf{x}}(t))dt && (21)\\ &= U(t_0,t_1) + \hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1);\mathbf{w}_c). && (22)\end{aligned}$$

First, the critic weights $\mathbf{w}_c$ can be adapted using the traditional adaptive critic updates, using an error (23) measuring the temporal difference of the critic estimates[1]

$$\begin{aligned}E(t=t_0) &:= E(\mathbf{x}(t_0),t_0,t_1;\mathbf{w}_a,\mathbf{w}_c)\\ &:= \int_{t_0}^{t_1} \phi(\mathbf{x}(t),\dot{\mathbf{x}}(t))dt + \hat{J}(\mathbf{x}(t_1);\mathbf{w}_c) - \hat{J}(\mathbf{x}(t_0);\mathbf{w}_c).\end{aligned} \tag{23}$$

Applying an adaptation law to the critic parameters $\mathbf{w}_c$ to force the temporal error towards zero ensures optimality for the given policy $\hat{\mathbf{g}}(\mathbf{x};\mathbf{w}_a)$ with fixed parameters $\mathbf{w}_a$. For example

$$\delta\mathbf{w}_c := -\eta\frac{\partial E(t)}{\partial \mathbf{w}_c}E(t). \tag{24}$$

Second, the policy can be improved by forcing (25) and (26) to be zero. Note: $a \overset{!}{=} b$ reads as a and has to be b

$$\begin{aligned}&\frac{d\left(U(t_0,t_1;\mathbf{w}_a) + \hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1);\mathbf{w}_c)\right)}{d\mathbf{w}_a}\\ &= \frac{dU(t_0,t_1;\mathbf{w}_a)}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T(t_1)}{d\mathbf{w}_a}\frac{d\hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1);\mathbf{w}_c)}{d\mathbf{x}(t_1)}\\ &\overset{!}{=} \mathbf{0}\end{aligned} \tag{25}$$

$$\begin{aligned}&= \int_{t_0}^{t_1} \frac{d\mathbf{x}^T(t)}{d\mathbf{w}_a}\left[\frac{\partial\phi}{\partial\mathbf{x}} + \frac{\partial\hat{\mathbf{g}}^T}{\partial\mathbf{x}}\frac{\partial\mathbf{f}^T}{\partial\mathbf{u}}\frac{\partial\phi}{\partial\dot{\mathbf{x}}}\right]dt\\ &\quad + \frac{d\mathbf{x}^T(t_1)}{d\mathbf{w}_a}\frac{d\hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1);\mathbf{w}_c)}{d\mathbf{x}(t_1)}.\end{aligned} \tag{26}$$

The superscript $\pi(\mathbf{w}_a)$ indicates that this equation is only valid for converged critics, given the current policy. Solving (15) with initial condition (13) yields the result for the total derivative $dJ^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_0);\mathbf{w}_c)/d\mathbf{w}_a$, which can be used to update the actor weights $\mathbf{w}_a$ in the usual steepest gradient manner. This is the continuous counterpart of the traditional adaptive critic designs. A comparison with a discrete one-step critic shows that in the continuous case indirect contributions to the total derivative are always taken into account whereas in the discrete case the total derivatives taken over one step only, miss out on the indirect contributions, as the following example shows. Naturally, a multistep discrete version of the temporal difference starts approximating the continuous case and this disadvantage starts disappearing.

An example is a discrete one-step development of the state $\mathbf{x}[t+1] = \mathbf{x}[t] + \mathbf{f}(\mathbf{x}[t],\mathbf{u}[t])$ with some control $\mathbf{u}[t] = \hat{\mathbf{g}}(\mathbf{x}[t];\mathbf{w}_a)$, such that the total derivative of $\mathbf{x}[t+1]$ with respect to the weights $\mathbf{w}_a$ is given by (27) which is equal to (28) because $d\mathbf{x}^T[t]/d\mathbf{w}_a = \mathbf{0}$

$$\begin{aligned}\frac{d\mathbf{x}^T[t+1]}{d\mathbf{w}_a} &= \frac{d\mathbf{x}^T[t]}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T[t]}{d\mathbf{w}_a}\frac{\partial\mathbf{f}^T(\mathbf{x}[t],\mathbf{u}[t])}{\partial\mathbf{x}[t]}\\ &\quad + \frac{d\hat{\mathbf{g}}^T(\mathbf{x}[t];\mathbf{w}_a)}{d\mathbf{w}_a}\frac{\partial\mathbf{f}^T(\mathbf{x}[t],\mathbf{u}[t])}{\partial\mathbf{u}[t]} && (27)\\ &= \frac{\partial\hat{\mathbf{g}}^T(\mathbf{x}[t];\mathbf{w}_a)}{\partial\mathbf{w}_a}\frac{\partial\mathbf{f}^T(\mathbf{x}[t],\mathbf{u}[t])}{\partial\mathbf{u}[t]}. && (28)\end{aligned}$$

If this procedure of calculating the total derivative is used repetitively at every time step t to update weights proportional to the gradient, the indirect influence through $\mathbf{x}[t]$ and all its later dependencies such as $\mathbf{f}(\mathbf{x}[t],\mathbf{u}[t])$ are always going to be missed. This can amount to a serious problem as substantial parts such as $\partial\mathbf{f}^T(\mathbf{x}[t],\mathbf{u}[t])/\partial\mathbf{x}[t]$ or $\partial\hat{\mathbf{g}}^T(\mathbf{x}[t],\mathbf{u}[t])/\partial\mathbf{x}[t]$, which are in general nonzero, are ignored as well. This procedure of adapting is basically BPTT($h = 0$), where h indicates the look-ahead horizon. The influence of the indirect path through $d\mathbf{x}^T[t+h]/d\mathbf{w}_a$ is lost for $h = 0$ and the gradient in this case is the instantaneous gradient used in BPTT($h = 0$). This is the reason why BPTT($h > 0$) is so much more powerful because with increased look-ahead h the gradient becomes more of a true total gradient.

The same applies to the continuous formulation adopted here which does not loose the indirect influence as infinitesimal influences are considered explicitly in the differential or integral formulation, given by (11)–(17), with the additional benefit of having variable step-size control from the integration routine. Adaptive step-size control is nothing else but an adaptive sampling scheme adjusting the step-size to different signal frequencies in the state–space variable $\mathbf{x}(t)$, so that basically the Nyquist criterion is always satisfied at any point in time. Automatic step-size adaptation has also an analogous second effect, as it implies a suitable number of look-ahead steps h (not necessarily equidistant sampled) such that the true gradient is adequately calculated (for a stiff system, h will be much larger and the step-size must be sufficiently small for the discrete counterpart BPTT(h) to work).

[1] As with traditional discrete ACDs, a discount factor γ may be introduced to discount future contributions. For continuous-time systems, an appropriate discounting term would be $\gamma^{t_1-t_0/T}$ with T being a unit time interval in which p percent interest is given on future costs, where $\gamma = 1/1+p$. For simplicity, we omit the inclusion here, or later just use γ for short.

The BPTT algorithm is considered to be more efficient because, in its recursive formulation, gradients are calculated with respect to a scalar target, while, in RTRL, the quantity $\partial \mathbf{x}^T[t]/\partial \mathbf{w}$ is a gradient of a vector, resulting in a matrix quantity. The same applies for the continuous calculation as well, where the matrix quantity $\dot{\mathbf{q}}$ has to be integrated. This is a drawback compared to BPTT(.) where only a gradient vector needs to be kept. The continuous formulation thus has more resemblance with RTRL. For the continuous-time formulation here this drawback may not be as severe if the state dimension is not too large and is certainly smaller than that of a simultaneous recurrent neural (SRN) trained with the RTRL algorithm to model the adaptive critic design, because the SRN would use a much larger state vector $\mathbf{x}_n$, where the subscript n symbolizes the network state, rather than the system state $\mathbf{x}$. Part of the motivation to use the continuous formulation is the module-like framework of plant, critic, and actor of an ACD framework, where the plant is conveniently given by the system differential (1).

B. Second-Order Adaptation for Actor Training

In this section, a second-order adaptation method for the actor parameters is developed. As seen before, the short-term cost from time t_0 to t_1, starting in state $\mathbf{x}(t_0)$ is given by (31)

$$U(\mathbf{x}(t_0), t_0, t_1; \mathbf{w}_a) = \int_{t_0}^{t_1} \phi(\mathbf{x}, \dot{\mathbf{x}}) dt \tag{29}$$

$$= \int_{t_0}^{t_1} \phi(\mathbf{x}, \mathbf{f}(\mathbf{x}, \hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a))) dt \tag{30}$$

$$= \int_{t_0}^{t_1} \tilde{\phi}(\mathbf{x}, \mathbf{w}_a) dt. \tag{31}$$

Assuming a stationary environment, the long-term cost in state $\mathbf{x}(t_0)$ and following the policy given by $\hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a)$ satisfies Bellman's optimality condition

$$J(\mathbf{x}(t_0); \mathbf{w}_a) = U(\mathbf{x}(t_0), t_0, t_1; \mathbf{w}_a) + J(\mathbf{x}(t_1); \mathbf{w}_a) \tag{32}$$

$$J_0 = U + J_1, \qquad \text{for short} \tag{33}$$

where $J(\mathbf{x}(t_0); \mathbf{w}_a)$ is the minimal cost in state $\mathbf{x}(t_0)$ following the policy $\pi : \hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a)$. Thus, a better notation would be $J^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_0))$ to indicate that J is actually a pure function of the state for a given policy. However, to simplify the notation, neither the superscript $\pi(\mathbf{w}_a)$ nor the argument $\mathbf{w}_a$ are used if not necessary. In ACDs, the long-term cost function $J(\mathbf{x}; \mathbf{w}_a)$ is approximated by $\hat{J}(\mathbf{x}; \mathbf{w}_c)$. This means that if for a certain policy $\hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a)$ Bellman's principle of optimality is satisfied, $\mathbf{w}_c$ is determined by the cost density ϕ and the policy parameters $\mathbf{w}_a$. An optimal policy is a policy that minimizes $J(\mathbf{x}; \mathbf{w}_a)$ and, therefore, a necessary condition is

$$\frac{dJ(\mathbf{x}(t_0))}{d\mathbf{w}_a} = \frac{dU(\mathbf{x}(t_0), t_0, t_1; \mathbf{w}_a)}{d\mathbf{w}_a} + \frac{dJ(\mathbf{x}(t_1))}{d\mathbf{w}_a} \overset{!}{=} \mathbf{0} \tag{34}$$

$$\frac{dJ_0}{d\mathbf{w}_a} = \frac{dU}{d\mathbf{w}_a} + \frac{dJ_1}{d\mathbf{w}_a} \overset{!}{=} \mathbf{0}, \qquad \text{for short} \tag{35}$$

$$= \frac{dU}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{dJ_1}{d\mathbf{x}} \overset{!}{=} \mathbf{0}. \tag{36}$$

1) Newton's Method: In traditional ACDs, (34) is used to train the actor parameters via a simple gradient-descent method. Newton's method could be used to speed up the traditional approach, although with the additional cost of computing the Jacobian of the function $dJ_0/d\mathbf{w}_a$ with respect to $\mathbf{w}_a$. In the context here, Newton's method for zero search is given by (37)–(42)

$$F(X) = 0 \tag{37}$$

find X by iterating $X^k \to X^{k+1}$ according to

$$X^{k+1} := X^k - \left[\frac{\partial F}{\partial X}\right]^{-1} F(X^k) \tag{38}$$

identifying

$$F := \frac{dJ_0}{d\mathbf{w}_a} \tag{39}$$

$$X := \mathbf{w}_a \tag{40}$$

$$\frac{\partial F}{\partial X} := \frac{d^2 J_0}{d\mathbf{w}_a^2} \tag{41}$$

yields

$$\mathbf{w}_a^{k+1} := \mathbf{w}_a^k - \left[\frac{d^2 J_0}{d\mathbf{w}_a^2}\right]^{-1} \frac{dJ_0}{d\mathbf{w}_a}. \tag{42}$$

To calculate the Jacobian, (36) is differentiated again with respect to $\mathbf{w_a}$, yielding (44)–(46), where $dJ_1/d\mathbf{x}$ and $d^2J_1/d\mathbf{x}^2$ might be approximated by a backpropagated J-approximator or a λ-critic and by a backpropagated λ-critic, respectively

$$\frac{d^2 J_0}{d\mathbf{w}_a^2} = \frac{d}{d\mathbf{w}_a}\left(\frac{dU}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{dJ_1}{d\mathbf{x}}\right) \tag{43}$$

$$= \frac{d^2 U}{d\mathbf{w}_a^2} + \frac{d}{d\mathbf{w}_a}\left(\frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{dJ_1}{d\mathbf{x}}\right) \tag{44}$$

$$= \frac{d^2 U}{d\mathbf{w}_a^2} + \frac{d^2\mathbf{x}^T}{d\mathbf{w}_a^2}\frac{dJ_1}{d\mathbf{x}} + \frac{d}{d\mathbf{w}_a}\left(\frac{dJ_1}{d\mathbf{x}}\right)\frac{d\mathbf{x}}{d\mathbf{w}_a} \tag{45}$$

$$= \frac{d^2 U}{d\mathbf{w}_a^2} + \frac{d^2\mathbf{x}^T}{d\mathbf{w}_a^2}\frac{dJ_1}{d\mathbf{x}} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{d^2 J_1}{d\mathbf{x}^2}\frac{d\mathbf{x}}{d\mathbf{w}_a}. \tag{46}$$

It has to be mentioned that $d^2\mathbf{x}^T/d\mathbf{w}_a^2$ is a third-order tensor, but with "the inner-product multiplication over the components of $\mathbf{x}$" the term $(d^2\mathbf{x}^T/d\mathbf{w}_a^2)(dJ_1/d\mathbf{x})$ gets the correct dimensions. Matrix notation starts to fail here and one is better advised to resort to tensor notation with upper and lower indices, which is done in the Appendix for more complicated expressions.

An important note has to be made about derivatives of critics and derivative ($\lambda-$) critics. They represent not instantaneous derivatives but rather averaged derivatives. Therefore, an averaged version of (46) is used as given by (47), where the $E\{.\}$ denotes the expectation operator, calculated as the expectation over a set of sampled start states $\mathbf{x}(t_0)$ according to their probability distribution from the domain of interest

$$E\left\{\frac{d^2 J_0}{d\mathbf{w}_a^2}\right\} = E\left\{\frac{d^2 U}{d\mathbf{w}_a^2} + \sum_{k=1}^{\dim(\mathbf{x})}\frac{d^2 x_k}{d\mathbf{w}_a^2}\frac{dJ_1}{dx_k} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{d^2 J_1}{d\mathbf{x}^2}\frac{d\mathbf{x}}{d\mathbf{w}_a}\right\}. \tag{47}$$

All the necessary terms in (47) are fully expanded in (92)–(134) in the Appendix.

Remark: The Hessian in (41) and (47) is assumed to be of full rank and therefore invertible. If it were not the case and $r := \operatorname{rank}(E\left\{d^2J_0/d\mathbf{w}_a^2\right\}) < \dim(\mathbf{w}_a)$, a submatrix of $E\left\{d^2J_0/d\mathbf{w}_a^2\right\}$ of rank r could be achieved by a singular value decomposition or Cholesky decomposition and then only r parameter components of $\mathbf{w}_a$ updated and the remaining components be left unchanged. This case would correspond to an overparametrization of the controller with respect to the system and long-term cost. The degenerate case of the Hessian being $\mathbf{0}$ is assumed not to occur, as in this case higher order terms would be necessary to determine optimality conditions. For nonlinear controller, system and long-term cost estimator $r < \dim(\mathbf{w}_a)$ will hardly be the case in practice and should it occur that some columns of the Hessian are linearly dependent, a few more initial points $\mathbf{x}(t_0)$ for the expectation calculation may be sufficient to remedy the problem as well.

Of course the Hessian will not be calculated in practice and the updates (38) and (42) will be solved more efficiently by Gaussian elimination, solving the affine system (48) for X^{k+1}

$$\left[\frac{\partial F}{\partial X}\right] X^{k+1} = \left[\frac{\partial F}{\partial X}\right] X^k - F(X^k) =: c. \tag{48}$$

To achieve an initial stabilizing control, the first actor training may be done based on minimizing costs accumulated during a midterm interval $[t_0\ t_1]$ and with an initial critic output of zero, e.g., $\mathbf{w}_c \equiv \mathbf{0}$ in a traditional neural network like an MLP with linear or affine outputs where weights and bias are encoded by $\mathbf{w}_c$ or a quadratic critic as in the example shown later. In the next cycle, the actor weights $\mathbf{w}_a$ are fixed and the critic weights $\mathbf{w}_c$ are adapted, by forming the standard Bellman error E_c according to (49) and (50)

$$\begin{aligned} E_c :=& U(\mathbf{x}(t_0), t_0, t_1; \mathbf{w}_a) + J^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1); \mathbf{w}_c) \\ &- J^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_0); \mathbf{w}_c) \end{aligned} \tag{49}$$

$$\delta\mathbf{w}_c := -\eta_c \frac{\partial E_c}{\partial \mathbf{w}_c} E_c. \tag{50}$$

After the convergence of the critic has been achieved, the error E_c is close to zero, and the critic $J^{\pi(\mathbf{w}_a)}(.; \mathbf{w}_c)$ is consistent with the policy $\pi(\mathbf{w}_a)$. A fast training method for the controller has been achieved with Newton's method. However, after one actor training cycle, the actor parameters $\mathbf{w}_a$ change to $\mathbf{w}_a + \delta\mathbf{w}_a$. To keep Bellman's optimality condition consistent, the critic weights $\mathbf{w}_c$ have to be adapted as well. Therefore, for converged critics $\mathbf{w}_c + \delta\mathbf{w}_c$ and $\mathbf{w}_c$ according to certain policies with parameters $\mathbf{w}_a + \delta\mathbf{w}_a$ and $\mathbf{w}_a$, respectively, the following conditions must hold:

$$\begin{aligned} &U(\mathbf{x}_0; \mathbf{w}_a + \delta\mathbf{w}_a) + J^{\pi(\mathbf{w}_a+\delta\mathbf{w}_a)}(\mathbf{x}_{\boldsymbol{\delta}\mathbf{w}_a}; \mathbf{w}_c + \delta\mathbf{w}_c) \\ &- J^{\pi(\mathbf{w}_a+\delta\mathbf{w}_a)}(\mathbf{x}_0; \mathbf{w}_c + \delta\mathbf{w}_c) \overset{!}{\doteq} 0 \end{aligned} \tag{51}$$

$$\begin{aligned} &U(\mathbf{x}_0, t_0, t_1; \mathbf{w}_a) + J^{\pi(\mathbf{w}_a)}(\mathbf{x};\mathbf{w}_c) \\ &- J^{\pi(\mathbf{w}_a)}(\mathbf{x}_0; \mathbf{w}_c) \overset{!}{\doteq} 0 \end{aligned} \tag{52}$$

where $\mathbf{x}_{\delta\mathbf{w}_a}$ means $\mathbf{x}(t)$ following the policy given by $\hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a + \delta\mathbf{w}_a)$, starting in state $\mathbf{x}_0 = \mathbf{x}(t_0)$. This is used in the following section to find the critic update $\delta\mathbf{w}_c$ due to an actor update $\delta\mathbf{w}_a$. Define a consistent actor–critic pair as a pair of a converged critic $J^{\pi(\mathbf{w}_a)}(\mathbf{x}; \mathbf{w}_c)$ and an actor $\hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a)$, such that Bellman's optimality principle (32), (51), and (52) holds for all $\mathbf{x}$ given the fixed actor representing policy $\pi : \mathbf{x} \mapsto \hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a)$, i.e., Bellman's optimality equation is satisfied with no error. Note that this does not imply that the policy is optimal and another actor with parameters $\tilde{\mathbf{w}}_a$ may yield another lower (or higher) cost function $J^{\pi(\tilde{\mathbf{w}}_a)}(.)$. The fact that these are different cost functions is expressed with the notation of a superscripted policy.

III. Almost Concurrent Actor and Critic Adaptation

Given a consistent actor–critic pair, actor training would induce an "error," or better, a change due to the new policy. This change $\Delta E_{\mathbf{w}_a}$ is given by (53) or its second-order approximation (55). Similarly, starting from a consistent actor–critic pair, with a fixed actor and changing critic weights would introduce an "error" $\Delta E_{\mathbf{w}_c}$, defined by (57) with a first-order approximation (59)

$$\begin{aligned} \Delta E_{\mathbf{w}_a} :=& U(\mathbf{w}_a + \delta\mathbf{w}_a) - U(\mathbf{w}_a) + J(\mathbf{x}_{\delta\mathbf{w}_a}; \mathbf{w}_c) \\ &- J(\mathbf{x}; \mathbf{w}_c) \end{aligned} \tag{53}$$

$$\begin{aligned} \doteq& \delta\mathbf{w}_a^T \left[\frac{dU}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{\partial J(\mathbf{x}; \mathbf{w}_c)}{\partial \mathbf{x}}\right] \\ &+ \frac{1}{2}\delta\mathbf{w}_a^T \left[\frac{d^2U}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{\partial^2 J(\mathbf{x}; \mathbf{w}_c)}{\partial^2 \mathbf{x}}\frac{d\mathbf{x}}{d\mathbf{w}_a}\right]\delta\mathbf{w}_a \end{aligned} \tag{54}$$

$$= \delta\mathbf{w}_a^T \frac{dJ(\mathbf{x}_0)}{d\mathbf{w}_a} + \frac{1}{2}\delta\mathbf{w}_a^T \frac{d^2J(\mathbf{x}_0)}{d\mathbf{w}_a^2}\delta\mathbf{w}_a \tag{55}$$

$$\approx \frac{1}{2}\delta\mathbf{w}_a^T \frac{d^2J(\mathbf{x}_0)}{d\mathbf{w}_a^2}\delta\mathbf{w}_a \tag{56}$$

$$\begin{aligned} \Delta E_{\mathbf{w}_c} :=& J(\mathbf{x}; \mathbf{w}_c + \delta\mathbf{w}_c) - J(\mathbf{x}_0; \mathbf{w}_c + \delta\mathbf{w}_c) \\ &- [J(\mathbf{x}; \mathbf{w}_c) - J(\mathbf{x}_0; \mathbf{w}_c)] \end{aligned} \tag{57}$$

$$=: \Delta J(\mathbf{x}_0; \mathbf{w}_c + \delta\mathbf{w}_c) - \Delta J(\mathbf{x}_0; \mathbf{w}_c) \tag{58}$$

$$\begin{aligned} &\doteq \delta\mathbf{w}_c^T \left[\frac{\partial J(\mathbf{x}; \mathbf{w}_c)}{\partial \mathbf{w}_c} - \frac{\partial J(\mathbf{x}_0; \mathbf{w}_c)}{\partial \mathbf{w}_c}\right] \\ &=: \delta\mathbf{w}_c^T \frac{\partial \Delta J(\mathbf{x}_0; \mathbf{w}_c)}{\partial \mathbf{w}_c}. \end{aligned} \tag{59}$$

To achieve consistency again after a training cycle involving actor and critic training, the change due to the actor $\Delta E_{\mathbf{w}_a}$ has to be matched by an appropriate critic change $\Delta E_{\mathbf{w}_c}$, i.e., $\Delta E_{\mathbf{w}_a} \overset{!}{\doteq} \Delta E_{\mathbf{w}_c}$. This has to hold for any starting point $\mathbf{x}_0^i$ and thus as before the expectation operator $E\{.\}$, approximated by (61),[2] has to be used. For a given actor change $\delta\mathbf{w}_a$ and an approximated expectation operator $E\{.\}$ over a set of a sufficiently

[2] Note that $f(.)$ is a generic function of any dimension (here vector and matrix for the "difference Jacobian" and the Hessian, respectively) and not the right-hand side of the system differential (1). The points $\mathbf{x}_0^i$ are sampled according to their *a priori* distribution $p_X(.)$.

large number $n_a \geq \dim(\mathbf{w}_a)$ of starting points $\mathbf{x}_0^i$, given by (61), it follows that (60) has to hold

$$\delta\mathbf{w}_a^T E_{n_a}\left\{\frac{dJ_0}{d\mathbf{w}_a}\right\} + \frac{1}{2}\delta\mathbf{w}_a^T E_{n_a}\left\{\frac{d^2J_0}{d\mathbf{w}_a^2}\right\}\delta\mathbf{w}_a \overset{!}{=} \delta\mathbf{w}_c^T E_{n_a}\left\{\frac{\partial\Delta J(.;\mathbf{w}_c)}{\partial\mathbf{w}_c}\right\} \tag{60}$$

$$E_{n_a}\{f(.)\} := \frac{1}{n_a}\sum_{i=1}^{n_a} f(\mathbf{x}_0^i). \tag{61}$$

To solve for $\delta\mathbf{w}_c$, there are two possibilities at first sight but only the second approach is working. Nevertheless, both approaches are discussed as it is not obvious why the first approach will not work. First, one might gather more points to build up a matrix A given by (64) and then calculate the pseudoinverse. However, due to correlation of the columns in A, the matrix A^TA is ill-conditioned and close to singular.

Remark: It can be seen that the columns are correlated. Because the long-term cost $J(\mathbf{x}_0^i;\mathbf{w}_c)$ is dependent on the actor parameters $\mathbf{w}_a$, where $\mathbf{w}_a$ are trained by an averaging process over many states $\mathbf{x}_0^i$, its derivatives along a single trajectory $\partial J(\mathbf{x}_0^i;\mathbf{w}_c)/\partial\mathbf{w}_c$ are dependent. This is because the trajectory is completely determined by the policy defined by $\mathbf{w}_a$. Thus, differences in the derivatives on one trajectory starting at $\mathbf{x}_0^i$ are very similar to differences in the derivatives on another trajectory starting at another point $\mathbf{x}_0^j$ because they follow the same controller law, given by $\hat{\mathbf{g}}(\mathbf{x};\mathbf{w}_a)$. Therefore, the subtraction of derivatives along a trajectory makes the columns more independent from individual starting points $\mathbf{x}_0^i$ (and, thus, counteracts the idea of using many different, randomly selected points $\mathbf{x}_0^i$ to achieve independence) and, therefore, correlates the columns of A. Also, the subtraction leads to cancellation and close to zero values for short-term evaluation $\Delta t = t_1 - t_0$ (remember: $\mathbf{x}^i = \mathbf{x}^i(t_1), \mathbf{x}_0^i = \mathbf{x}^i(t_0)$). The approach is written down for the sake of completeness but in practice the second approach, discussed later, is much more promising.

For the first approach, at least as many starting points as parameters $\mathbf{w}_c$ are needed: $n_c \geq \dim(\mathbf{w}_c)$. Furthermore, $\delta\mathbf{w}_a$ might be computed with a safeguarded Newton algorithm, where the safeguard could be a simple backstepping, taking only a fraction $\eta_a <= 1$ (η_a is estimated by the algorithm) of the original computed Newton update to ensure a decrease in the objective function $E\{dJ_0/d\mathbf{w}_a\}$ of Newton's method. Together, this yields the following training cycle:

$$\delta\mathbf{w}_a := -\eta_a\left[E_{n_a}\left\{\frac{d^2J_0}{d\mathbf{w}_a^2}\right\}\right]^{-1} E_{n_a}\left\{\frac{dJ_0}{d\mathbf{w}_a}\right\} \tag{62}$$

$$\delta\mathbf{w}_c := \left(A^TA\right)^{-1}A^T s\mathbb{I}_{\dim(\mathbf{w}_c)\times 1}, \quad \text{with} \tag{63}$$

$$A := \left[\frac{\partial\Delta J(\mathbf{x}_0^1;\mathbf{w}_c)}{\partial\mathbf{w}_c}\ldots\frac{\partial\Delta J(\mathbf{x}_0^{n_c};\mathbf{w}_c)}{\partial\mathbf{w}_c}\right]^T \tag{64}$$

$$\frac{\partial\Delta J(\mathbf{x}_0^i;\mathbf{w}_c)}{\partial\mathbf{w}_c} = \frac{\partial J(\mathbf{x}_1^i;\mathbf{w}_c)}{\partial\mathbf{w}_c} - \frac{\partial J(\mathbf{x}_0^i;\mathbf{w}_c)}{\partial\mathbf{w}_c} \tag{65}$$

$$s := \delta\mathbf{w}_a^T E_{n_a}\left\{\frac{dJ_0}{d\mathbf{w}_a}\right\} + \frac{1}{2}\delta\mathbf{w}_a^T E_{n_a}\left\{\frac{d^2J_0}{d\mathbf{w}_a^2}\right\}\delta\mathbf{w}_a \tag{66}$$

$$\mathbb{I}_{\dim(\mathbf{w}_c)\times 1} := [1\ldots1\ldots1]^T \tag{67}$$

where $E_{n_a}\{d^2J_0/d\mathbf{w}_a^2\}$ and $E_{n_a}\{dJ_0/d\mathbf{w}_a\}$ are given by (47) and (36), respectively.

The second approach expresses the difference $\Delta J(\mathbf{x}_0;\mathbf{w}_c + \delta\mathbf{w}_c)$ in long-term cost $J(\mathbf{x};\mathbf{w}_c)$ into a first-order Taylor series and selects $\delta\mathbf{w}_c := \eta(\partial\Delta J(\mathbf{x}_0;\mathbf{w}_c)/\partial\mathbf{w}_c)$, as follows:

$$\Delta J(\mathbf{x}_0;\mathbf{w}_c) = J(\mathbf{x};\mathbf{w}_c) - J(\mathbf{x}_0;\mathbf{w}_c) \tag{68}$$

$$\Delta J(\mathbf{x}_0;\mathbf{w}_c + \delta\mathbf{w}_c) \doteq \Delta J(\mathbf{x}_0;\mathbf{w}_c) + \delta\mathbf{w}_c^T\frac{\partial\Delta J(\mathbf{x}_0;\mathbf{w}_c)}{\partial\mathbf{w}_c} \tag{69}$$

$$= \Delta J(\mathbf{x}_0;\mathbf{w}_c) + \eta\left\|\frac{\partial\Delta J(\mathbf{x}_0;\mathbf{w}_c)}{\partial\mathbf{w}_c}\right\|^2. \tag{70}$$

Using $J(\mathbf{x};\mathbf{w}_c + \delta\mathbf{w}_c)$ as the new cost-to-go for the new policy $\pi(\mathbf{w}_a + \delta\mathbf{w}_a)$ leads to a critic update according to

$$\eta = \frac{\Delta E_{\mathbf{w}_a}}{\left\|\frac{\partial\Delta J(\mathbf{x};\mathbf{w}_c)}{\partial\mathbf{w}_c}\right\|^2} \tag{71}$$

$$\delta\mathbf{w}_c := \eta\frac{\partial\Delta J(\mathbf{x};\mathbf{w}_c)}{\partial\mathbf{w}_c} = \eta\left[\frac{\partial J(\mathbf{x};\mathbf{w}_c)}{\partial\mathbf{w}_c} - \frac{\partial J(\mathbf{x}_0;\mathbf{w}_c)}{\partial\mathbf{w}_c}\right] \tag{72}$$

where $\Delta E_{\mathbf{w}_a}$ is given by (54). With this choice, $\Delta E_{\mathbf{w}_a}$ is equal to $\Delta E_{\mathbf{w}_c} = \delta\mathbf{w}_c(\partial\Delta J(\mathbf{x}_0;\mathbf{w}_c)/\partial\mathbf{w}_c)$ given by (59), as demanded by a first-order approximation. To account for higher accuracy in Bellman's optimality condition, standard HDP training could improve consistency between actor and critic to an arbitrary degree. However, the first-order approximation introduced here might be sufficient to safely improve the current policy again, at least for a few actor–critic training cycles. Standard HDP critic training is marginally sped up ($\approx 4\%$) in the linear quadratic regulator (LQR) experiment in Section IV-B when (72) was applied before the standard critic update. A further speed up may result when HDP critic training is reduced by invoking it only every nth actor–critic training cycle. Especially in more difficult situations with neural network controllers and critics this may be helpful, because then parameter changes may need to be more gradually done due to local minima. Then, it is expected that many more actor–critic cycles are needed.

A. Some Remarks and Discussion

In the simple LQR example, this method of skipping standard critic training does not improve convergence times because critic parameters converge very fast to the neighborhood of the exact values and minor critic changes around the exact values cause substantial changes in actor parameters.

In a general setup with nonlinear critic, plant and controller, critic changes based on the proposed first-order concurrent actor–critic scheme may be enough to drive the actor to a new local minima and then cause the critic again to change enough to move the actor parameters further. It has been noted [3],

[20] that in ordinary stochastic HDP critic training the total gradient

$$\frac{d\langle E^2\rangle}{d\mathbf{w}_c} = 2\left\langle E\frac{dE}{d\mathbf{w}_c}\right\rangle = 2\left\langle E\left(\gamma\frac{\partial J_1}{\partial \mathbf{w}_c} - \frac{\partial J_0}{\partial \mathbf{w}_c}\right)\right\rangle$$

of the averaged squared Bellman error ($\langle E^2\rangle = \langle (U + \gamma J_1 - J_0)^2\rangle$) may in fact converge to the wrong parameters, whereas the partial gradient $\partial\langle E^2\rangle/\partial\mathbf{w}_c = -2\langle E(\partial J_0/\partial\mathbf{w}_c)\rangle$ achieves the correct values.[3] The $\langle . \rangle$ means the expectation over the noise as well as the expectation $E\{.\}$ with regard to initial state distribution.

This convergence to the wrong parameters is because the total gradient $dE/d\mathbf{w}_c$ goes the steepest decent towards the local minima, whereas the partial gradient allows for more exploration of the state–space. Adapting the actor by a greedy adaptation based on the two kinds (partial or total gradient) of critic adaptation thus leads to different controllers during training as well. However, this is an issue beyond this paper; here only a second-order actor adaptation is developed and some form of critic training is assumed, whether it be by partial gradients, total gradients, or some other form. An alternative for the total gradient to yield correct critic parameters would be to use the two-sample rule [21], [13], [3], [20] to avoid correlation of the error E with the gradient $dE/d\mathbf{w}_c$ two trajectories based on the *noisy* system equations ($\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\nu}$) with different noise realizations $\boldsymbol{\nu}$, such that $\langle E(dE/d\mathbf{w}_c)\rangle = 0$. Then, adaptation of the critic parameters would be proportional to $E_1(dE_2/d\mathbf{w}_c)$, with $E_i := U(\mathbf{x}(t_0), t_0, t_1) + \gamma\hat{J}(\mathbf{x}^i(t_1); \mathbf{w}_c) - \hat{J}(\mathbf{x}(t_0); \mathbf{w}_c)$, for i =1, 2 and $\mathbf{x}^i(t_1)$ is the state following the noisy dynamics of two simulation runs with two different noise realizations. Apart from doubling the simulation efforts, the additional noise realization of the two-sample rule may also slow down convergence.

These ideas have only been used in the context of discrete ACDs but may be used in continuous-time systems as outlined here as well. Because of the subtraction of the two partial gradients, Baird has suggested to further discount the term $\partial J_1/\partial\mathbf{w}_c$, moving adaptation towards the one-gradient rule [21], [20].

For continuous-time ACDs introduced here, it was focused on actor training rather than on critic training. Nevertheless, the same total gradient calculations can be used for critic training as well: (24) and (50) would have to be used with the two-sample rule in case of a nondeterministic system. In the continuous-time case, a nicer approach than the additional discounting with total gradients would be to modify the short-term horizon t_1 until the difference is substantial enough. As a matter of fact, the two-gradient rule was actually used here in (72) for the concurrent actor–critic training. It was observed that concurrent actor–critic adaptation based only on the one-gradient rule, i.e., setting the first term in (72) to zero did not work as well in the LQR example.

The selection of an appropriate short-term to midterm time t_1 may be seen as a form of shaping. Selection might even depend on the state–space and thus allow to concentrate training on more important areas. As mentioned before, to get an initial stabilizing policy, a midterm optimization might be performed at first with a zero long-term cost. This kind of shaping is certainly easier than having different look-ahead intervals h as in the discrete case with BPTT(h) as the necessary look-ahead h is a function of the time difference $t_1 - t_0$ and the underlying dynamics, whereas in the continuous-time case adaptive step-size control takes care of the latter.

In the sense of more exploration over exploitation, the proposed concurrent actor–critic training may even be of an advantage because the approximate consistency of actor and critic means that the Bellman equation only holds approximately and thus may help to explore different solutions or even escape local minima in more complex situations, similar to the use of the partial gradient but with the more robust convergence properties of the total gradient. However, this would have to be properly investigated.

Prokhorov [20] emphasizes that "Strictly speaking, critics lose validity as the weights of the action network are changed. A more rigorous approach would be to resume the critic training as soon as one action weight update is made." The procedure here with concurrent actor–critic training, suggests precisely this. However, as the error is only a linear approximation in terms of weight changes, after a while, some error might build up and a standard critic update can be performed to achieve an "error-free" Bellman equation. Nevertheless, this is probably as close as possible to having a concurrent actor and critic training without using higher order terms to model the influence of $\delta\mathbf{w}_a$ on necessary critic changes $\delta\mathbf{w}_c$.

B. Some Convergence Results

A few notes on the convergence of adaptive critics via actor and critic cycles for continuous domains should be made. In the offline mode, where multiple starting points $\mathbf{x}_0^i$ and an averaging takes place, proofs are available, see, e.g., [20] and [22], which rely on stochastic iterative algorithms in general [23] and in the context of neurodynamic programming proofs [13, Ch. 6], state that for given error tolerances ϵ and δ in the policy evaluation and (greedy) policy improvement steps, respectively, the resulting cost-to-go function will be close to the optimal one, within a zone $O(\delta + 2\gamma\epsilon)/(1-\gamma)^2$, where $\gamma \equiv \tilde{\gamma}^{(t_1 - t_0)/T}$ is the discount rate of future long-term costs $\hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1); \mathbf{w}_c)$ with proper scaling of an equivalent $\tilde{\gamma}$ and unit time T as per Footnote 1.

In this offline case, the algorithm presented here will converge to an optimal policy as well, assuming critic and actor approximations are within the same tolerances ϵ and δ. The argumentation is simply that the algorithm presented here is a version of an approximate policy iteration algorithm, provided an appropriate critic training in the nondeterministic case as outlined previously.

The conflicting objectives of exploration versus exploitation may be solved by augmenting the training process such that an auxiliary process controlling critic and actor training by switching between exploratory updates and greedy policy updates such that all states and actions are executed but that the fraction between exploratory and greedy updates gradually declines during training. Cybenko has proven convergence of

[3] In [20], the methods of using partial and total gradient are called one-gradient and two-gradient rules, respectively, because when using the total gradient, the term $\gamma(\partial J_1/\partial\mathbf{w}_c)$ is added.

this procedure in the context of approximate Q-learning and Markov processes [24].

For the online case, when only one start point $\mathbf{x}_0$ is taken to update critic and actor, proofs are much harder because of the problem of incremental optimization of nonconvex functions, where greedy approaches may fail. Additionally, local minima in the cost function over the actor's parameter space may be even more of a problem as in the offline mode, where different initializations, backtracking, or stochastic optimization methods like simulated annealing might be used. Nevertheless, Tsitisklis and Van Roy [14] have proven convergence for linear cost functions when appropriate sampling according to the steady-state probability distributions was used. Recently, this result was extended by Xu *et al.* [15] to nonlinear cost functions by using a nonlinear kernel mapping from an input space to a high-dimensional but linear space.

C. Some General Remarks

A final remark on "continuous backpropagation" and its discrete counterpart BPTT(h) is that normally all the algorithms will be implemented on a discrete clocked computer, which seems to be a plus for the discrete BPTT(h). However, any integration routine does basically a discretization but with variable time steps. This is an advantage over fixed step-size discrete ACDs because no time is wasted by cycling through areas with small steps when nothing happens. The truncation depth h in BPTT(h) is the same as the time t_1 used to indicate the short-term to midterm costs. Another difference is that calculating total derivatives in BPTT(h) is performed backwards, whereas with the "continuous backpropagation" a forward integration is performed.

In the previous section, a complete formulation to train the controller or actor based on second-order derivatives in conjunction with Newton's method has been introduced. In the present section, almost concurrent adaptation for critic weights based on actor changes is shown. However, to use this approach it is necessary to have second-order derivatives for the controller network $\hat{\mathbf{g}}(\mathbf{x};\mathbf{w}_a)$ as well as for the critic network $J(\mathbf{x};\mathbf{w}_c)$ [see, for example, (46)]. For the critic approximation network $\hat{J}(\mathbf{x};\mathbf{w}_c)$ this means $\partial^2\hat{J}(\mathbf{x};\mathbf{w}_c)/\partial\mathbf{x}^2$ has to be calculated,[4] and this is exactly what has to be done in GDHP, which is the most advanced adaptive critic design. The simplest way to calculate the second-order derivatives was suggested by Werbos [25] and implemented by Prokhorov for a one-layered multilayer perceptron [20]. Basically, for the given network $\hat{J}(\mathbf{x};\mathbf{w}_c)$, a dual network $\hat{\lambda}(\mathbf{x};\mathbf{w}_c)$ is constructed by applying the backpropagation algorithm on the network $\hat{J}(\mathbf{x};\mathbf{w}_c)$. Together, with the original $\hat{J}(\mathbf{x};\mathbf{w}_c)$ network this can be seen as a combined "forward" network which still has the same parameters $\mathbf{w}_c$ as the original network. Applying backpropagation on this combined network outputs $\hat{\lambda}(\mathbf{x};\mathbf{w}_c) = d\hat{J}(\mathbf{x};\mathbf{w}_c)/d\mathbf{x}$ and calculates precisely the desired second-order derivatives $\partial^2\hat{J}(\mathbf{x};\mathbf{w}_c)/\partial\mathbf{x}^2$. This is perhaps the most efficient implementation for calculating second-order derivatives; at least the authors of this paper are not aware of any better solution.

Another implementation of a second-order training method, called the extended Kalman filter (EKF), has been made and successfully experimented with (see, for example, [20] and [26]). The advantage of the EKF algorithm over Newton's method is that it is based on pattern-by-pattern updates, unlike the Newton method presented here. However, the method here, particularly (47) and (42), where the expectation operator and update equation are to be approximated by a batch update (this is not necessarily the case for further updates) as a running average, could be used, and, therefore, a more of pattern-by-pattern update version could easily be achieved. Also, the inverse Hessian could be updated using the matrix inversion lemma (Woodbury's equation) and so achieving a pattern-by-pattern update (see, for example, [27]). It is even better to avoid the inversion altogether and use a linear equation solver, as only one point X^{k+1} has to be determined, as suggested previously by (48).

Another more disturbing part for both EKF and Newton's method is that both algorithms are of complexity $O(N^2)$, where N is the number of parameters, whereas Werbos' method for GDHP outlined previously is only of $O(N)$.

In this paper, the discount factor γ has been left out but it is straight forward to introduce it in the equations corresponding to Bellman's optimality equation. This is done by modifying terms involving the cost-to-go function $J(\mathbf{x}(t_1);\mathbf{w}_c) =: J_1$ with a multiplicative factor γ. Under some benign assumptions the cost integral for the LQR system is finite, therefore no $\gamma^{(t_1-t_0)/T}$-factor has to be introduced anyway, or simply would only have to be set to 1.

While all the formulas have been developed for a generic nonlinear system (1) and controller (2), they are tested only on an LQR system because for this case the optimal parameters can be achieved by solving the corresponding algebraic Riccati equation. In general, nonaffine, nonlinear systems may only be controllable under certain conditions. An interesting approach has been done by Ge *et al.* [28], [29] where, with suitable design parameters, semiglobal uniform ultimate boundedness of all signals was achieved with an adaptive neural network controller for a general class of nonlinear single-input–single-output (SISO) systems.

IV. Experiment: Linear System With Quadratic Cost-to-Go Function (LQR)

The LQR system equations and cost density are defined by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad \mathbf{A} = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.5 & -1 \\ 0.5 & 2 \end{bmatrix} \tag{73}$$

$$\phi(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{x}^T\mathbf{Q}\;;\mathbf{x} + \dot{\mathbf{x}}^T\mathbf{R}\,\dot{\mathbf{x}} \quad \mathbf{Q} = \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{74}$$

The control $\mathbf{u} = \hat{\mathbf{g}}(\mathbf{x},\mathbf{w})$ should be of a state-feedback form with some parameters $\mathbf{w}$ and the cost-to-go function or performance index is given by

$$J(\mathbf{x},\dot{\mathbf{x}}) = \int_{t_0}^{\infty} \phi(\mathbf{x},\dot{\mathbf{x}})dt. \tag{75}$$

[4] For fixed parameters $\mathbf{w}_c$, partial and total derivatives are the same: $\partial^2\hat{J}(\mathbf{x};\mathbf{w}_c)/\partial\mathbf{x}^2 \equiv d^2\hat{J}(\mathbf{x};\mathbf{w}_c)/d\mathbf{x}^2$ and used interchangeably here.

A. *Optimal LQR-Control*

To solve the previous system with minimal performance index, an algebraic Riccati equation (ARE) has to be solved. Details can be found in [30, Ch. 14]. However, for numerical purposes, Matlab's *lqr*-function can be used to calculate the optimal feedback gain. To make use of Matlab's *lqr*-function the performance index has to be changed to (76), where a simple comparison with the original performance index yields $\tilde{\mathbf{Q}} = \mathbf{Q} + \mathbf{A}^T\mathbf{R}\mathbf{A}$, $\tilde{\mathbf{R}} = \mathbf{B}^T\mathbf{R}\mathbf{B}$, $\tilde{\mathbf{N}} = \mathbf{A}^T\mathbf{R}\mathbf{B}$, and $\mathbf{R}^T = \mathbf{R}$. Additional requirements are that the pair $(\mathbf{A}, \mathbf{B})$ be stabilizable, $\tilde{\mathbf{R}} > \mathbf{0}$, and $\tilde{\mathbf{Q}} - \tilde{\mathbf{N}}\tilde{\mathbf{R}}^{-1}\tilde{\mathbf{N}}^T \geq \mathbf{0}$ and that neither $\tilde{\mathbf{Q}} - \tilde{\mathbf{N}}\tilde{\mathbf{R}}^{-1}\tilde{\mathbf{N}}^T \geq \mathbf{0}$ nor $\mathbf{A} - \mathbf{B}\tilde{\mathbf{R}}^{-1}\tilde{\mathbf{N}}^T$ has an unobservable mode on the imaginary axis

$$\begin{aligned}\tilde{J}(\mathbf{x}, \mathbf{u}) &= \int_{t_0}^{\infty} \tilde{\phi}(\mathbf{x}, \mathbf{u}) dt \\ &= \int_{t_0}^{\infty} (\mathbf{x}^T\tilde{\mathbf{Q}}\,\mathbf{x} + \mathbf{u}^T\tilde{\mathbf{R}}\,\mathbf{u} + 2\mathbf{x}^T\tilde{\mathbf{N}}\mathbf{u}) dt. \end{aligned} \tag{76}$$

The optimal control law has the form $\mathbf{u}(\mathbf{x}) = \hat{\mathbf{g}}(\mathbf{x}; \mathbf{K}) = -\mathbf{K}\mathbf{x}$ with feedback matrix $\mathbf{K}$ which can be expressed as

$$\mathbf{K} = \tilde{\mathbf{R}}^{-1}\left(\mathbf{B}^T\mathbf{S} + \tilde{\mathbf{N}}^T\right) \tag{77}$$

where $\mathbf{S}$ is the solution to the ARE

$$\mathbf{0} = \mathbf{A}^T\mathbf{S} + \mathbf{S}\mathbf{A} - (\mathbf{S}\mathbf{B}^T + \tilde{\mathbf{N}})\tilde{\mathbf{R}}^{-1}(\mathbf{B}^T\mathbf{S} + \tilde{\mathbf{N}}^T) + \tilde{\mathbf{Q}}. \tag{78}$$

B. *Numerical Example*

1) $\mathrm{rank}(\mathbf{K}) = \dim(\mathbf{x})$: Using the following system values (79)–(81), the optimal feedback is given by (83)

$$A = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix}, \qquad \mathrm{eig}(A) = \{-2, -3\} \tag{79}$$

$$B = \begin{bmatrix} 0.5 & -1 \\ 0.5 & 2 \end{bmatrix} \tag{80}$$

$$Q = R = C = D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{81}$$

$$S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \text{solution to ARE (78)} \tag{82}$$

$$K^* = \begin{bmatrix} 0.6667 & -4.6667 \\ 0.3333 & -0.3333 \end{bmatrix}, \qquad \text{optimal feedback by (77)} \tag{83}$$

$$A^{cl*} = A - BK^* = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathrm{eig}(A^{cl}_{+}) = \{-1, -1\}. \tag{84}$$

In [4], there are also other feedback methods for LQR systems investigated for comparison with the adaptive critic methods to determine long-term costs and controls. One of them is derived from the calculus of variations (CoV), which is theoretically equivalent to dynamic programming in the sense that it minimizes the same cost function to find an optimal controller.[5] If the matrix B is full rank, all the (stable) methods investigated achieve the same optimal result for the feedback matrix K^*.

2) $\mathrm{rank}(\mathbf{K}) < \dim(\mathbf{x})$: Lowering the dimension of the control $\mathbf{u}$, and therefore the rank of the control matrix B and the feedback matrix K to impose constraints on the possible mappings $\hat{\mathbf{g}} : \mathbf{x} \overset{K}{\mapsto} \mathbf{u}$ fails all adaptive methods investigated in [4] except the adaptive critic design and of course the solution calculated via (77) and (78). The adaptation based on the calculus of variations violates the independence conditions of the fundamental lemma of CoV. In the case of a reduced rank feedback matrix, an adaptation law based on CoV with an augmented cost functional and the introduction of Lagrange multipliers would have to be developed. This seems far more complicated than the approach via ACDs. The optimal reduced rank feedback is given by (89), based on the system matrices (85)–(87)

$$A = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix}, \qquad \mathrm{eig}(A) = \{-2, -3\} \tag{85}$$

$$B = \begin{bmatrix} 1 \\ -3 \end{bmatrix} \tag{86}$$

$$Q = R = C = D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{87}$$

$$S = \begin{bmatrix} 1.0207 & -0.1865 \\ -0.1865 & 2.67881 \end{bmatrix}, \qquad \text{solution to ARE (78)} \tag{88}$$

$$K^* = [\,-0.2420 \quad 0.1777\,], \qquad \text{optimal feedback by (77)} \tag{89}$$

$$\begin{aligned} A^{cl*} = A - BK^* &= \begin{bmatrix} -0.7580 & -2.1777 \\ 0.2741 & -3.4669 \end{bmatrix}, \\ \mathrm{eig}(A^{cl}_{+}) &= \{-1.000, -3.225\}. \end{aligned} \tag{90}$$

Using traditional adaptive critics will find the correct optimal values for K in both cases independent of the rank of K. However, when used with the training methods introduced in Section II, only a few actor–critic training cycles are needed and speed up the traditional adaptive critic training considerably by at least 10%–30% (see Section IV-D for comparison with first-order update for the controller). To achieve higher accuracies the Bellman "error" was fixed at 10^{-5} and critic training could take a substantial amount of time. An adaptive error threshold would certainly improve overall training time, as well as adaptive learning rates would, like stochastic meta descent [31]. This method could easily improve critic training by another 15%, but was not used in the plots for better comparisons.

C. *Results for Continuous ACDs With Newton's Method*

In this section, the Newton training is tested on the same LQR example as has been used previously.

1) $\mathrm{rank}(\mathbf{K}) = \dim(\mathbf{x})$: Fig. 2 shows the actor or feedback parameters $\mathbf{K}$. The solid lines represent only periods of actor training with input and output values to the Newton routine. To improve stability, Newton's method was extended to only allow changes $d\mathbf{K}$ which satisfy $\|d\mathbf{K}\|_\infty \leq 10\|\mathbf{K}\|_\infty$; otherwise, $d\mathbf{K} := (\|\mathbf{K}\|_\infty / \|d\mathbf{K}\|_\infty) d\mathbf{K}$. It may occasionally happen

[5] Dynamic programming may also have other advantages, for example when having uncertain or disturbed states, or has a simpler formulation of the method [see the comments in the case of $\mathrm{rank}(\mathbf{K}) < \dim(\mathbf{x})$].

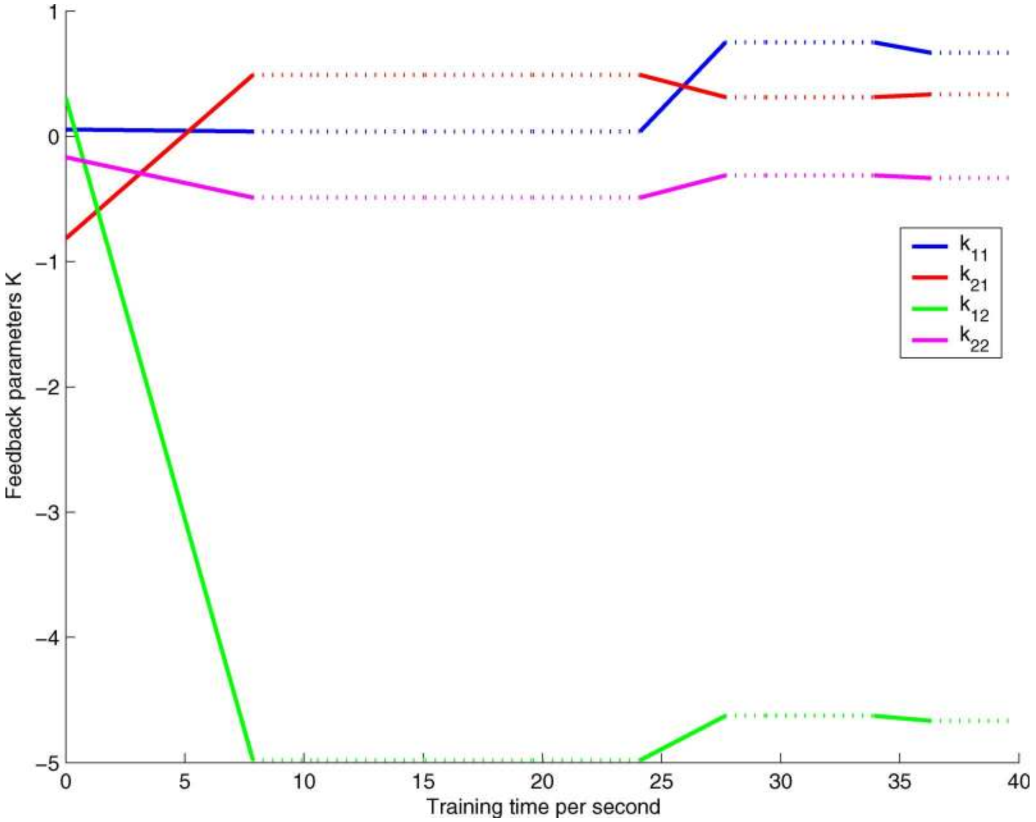

Fig. 2. Trajectory of the actor parameters $\mathbf{K}$ for the system given in Section IV-B1. The solid lines represent the time actor training via Newton's method. During the time indicated by the dashed lines, actor parameters are frozen and critic weights are adapted. After four actor–critic cycles the parameters are learned within an error better than 10^{-5}.

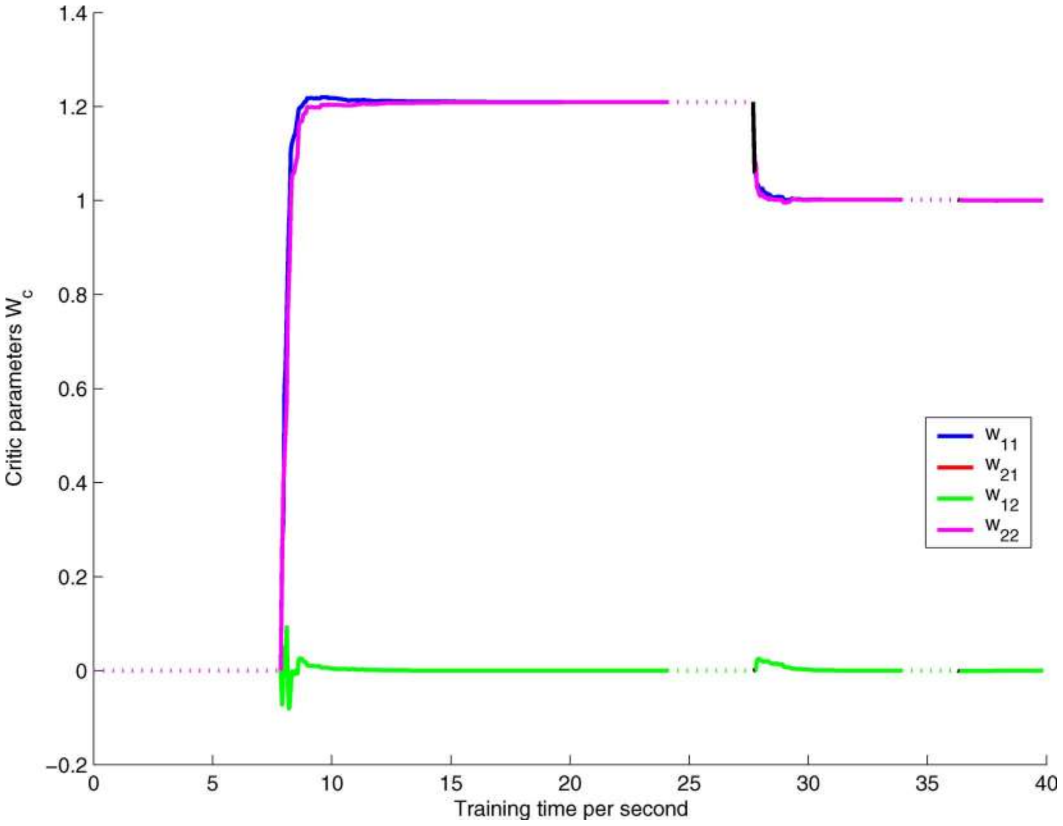

Fig. 3. Trajectory of critic parameters $\mathbf{W}_c$. The solid lines represent the time critic training is performed. After the first actor–critic cycle the actor–critic consistency is achieved and the proposed linear critic updates due to actor changes can be applied. This is shown by the black lines which represent a jump towards the optimal values, especially for the nonzero w_{11}, w_{22} at the second actor–critic cycle.

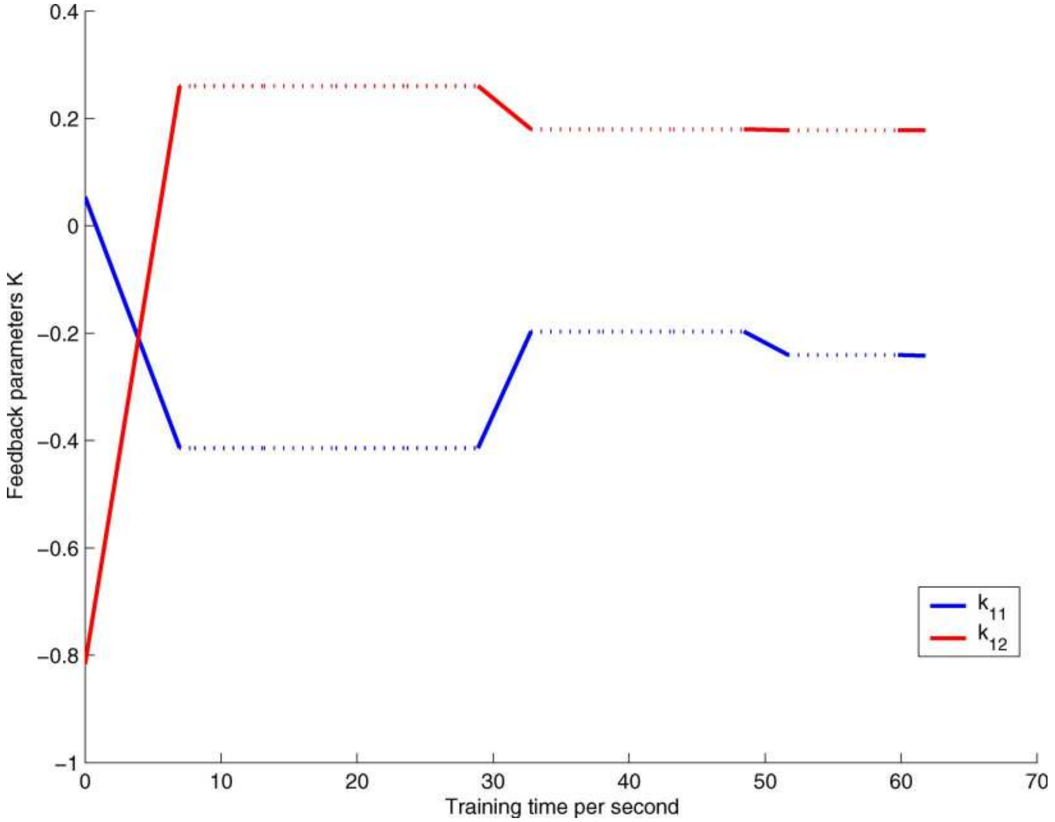

Fig. 4. Trajectory of the actor parameters $\mathbf{K}$ for the system given in Section IV-B2. The solid lines represent the time actor training via Newton's method. During the time indicated by the dashed lines, actor parameters are frozen and critic weights are adapted. After four actor–critic cycles the parameters are learned within an error better than 10^{-5}.

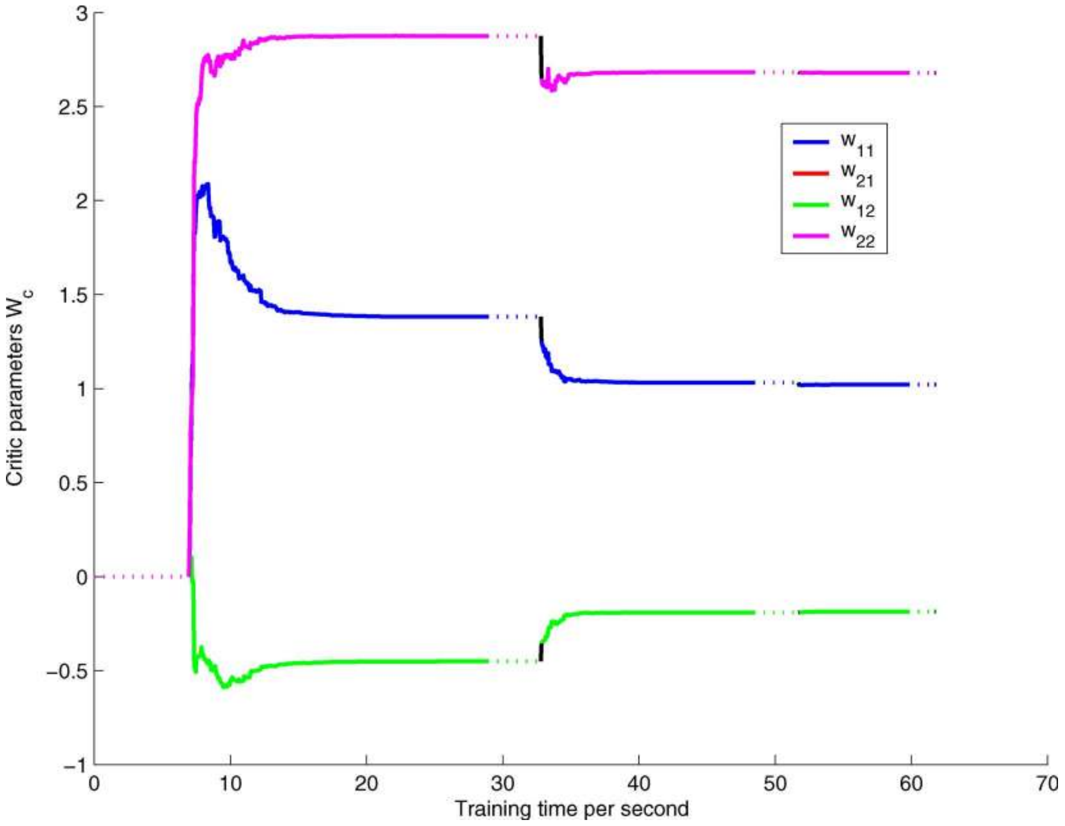

Fig. 5. Trajectory of critic parameters $\mathbf{W}_c$ (note: $w_{12} = w_{21}$). The solid lines represent the time critic training is performed. After the first actor–critic cycle, the actor–critic consistency is achieved and the proposed linear critic updates due to actor changes can be applied. This is shown by the black lines which represent a jump towards the optimal values, given by (72), especially for the nonzero w_{11} and w_{22} at the second actor–critic cycle.

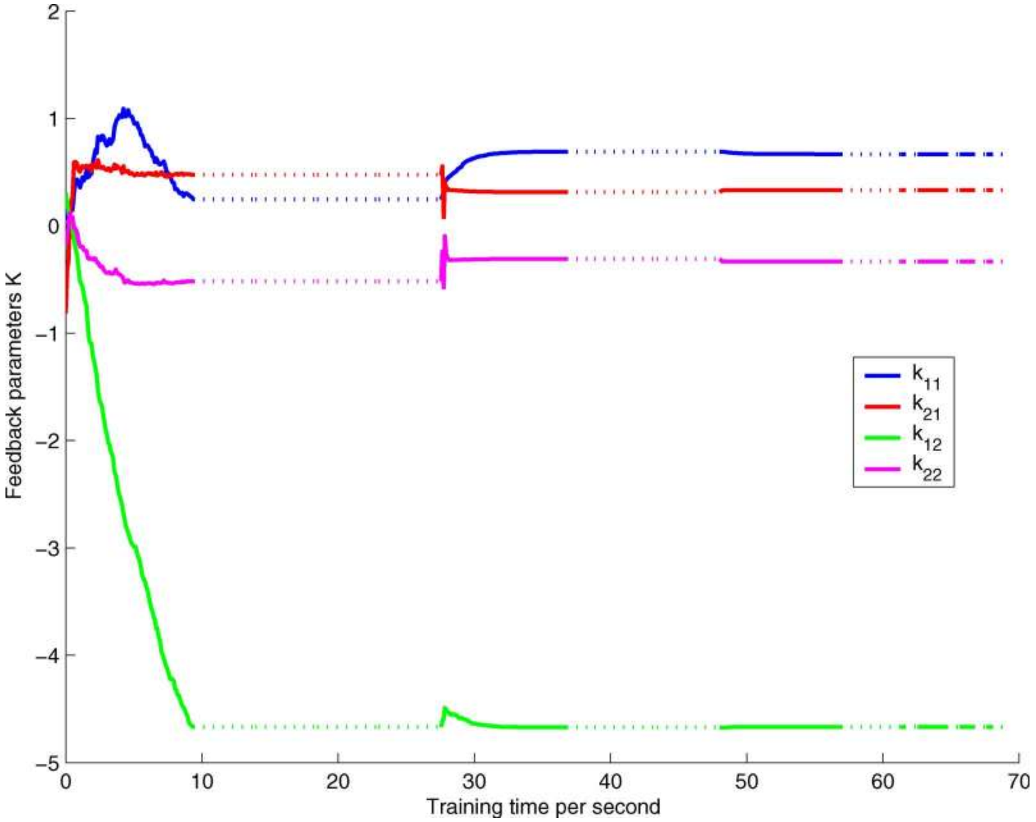

Fig. 6. Trajectory of the parameters $\mathbf{K}$ for traditional actor update. Parameter accuracy is better than 10^{-5}.

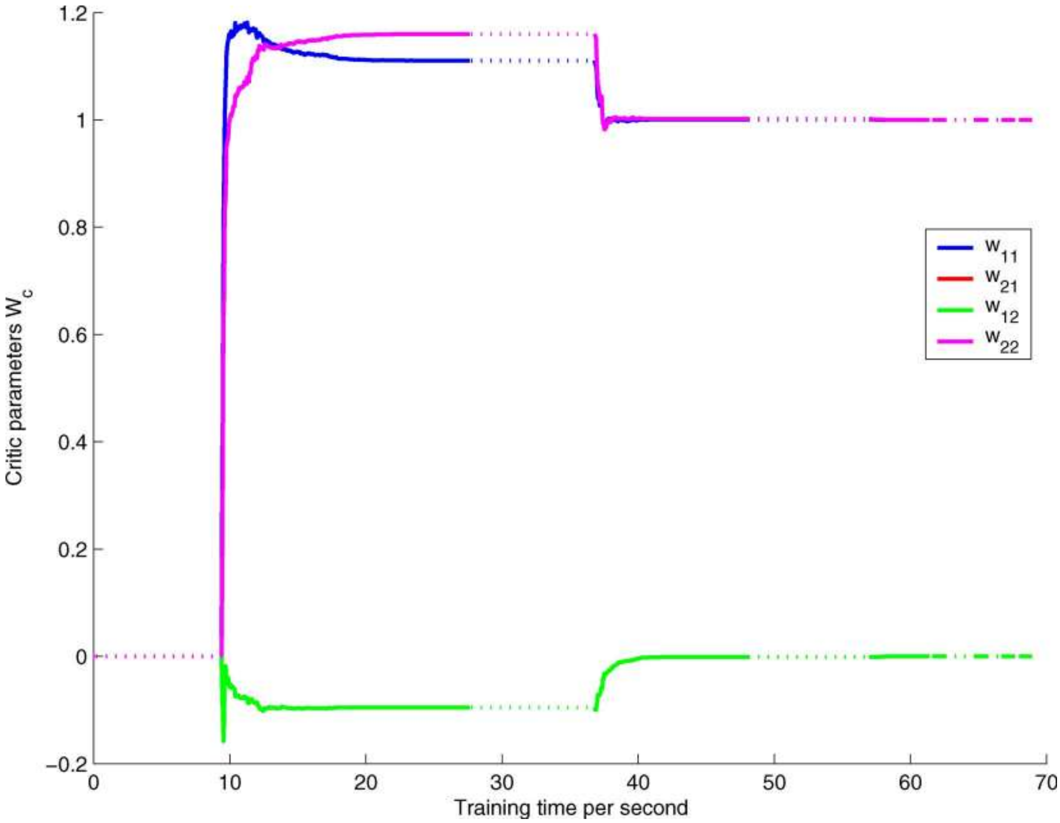

Fig. 7. Trajectory of critic parameters $\mathbf{W}_c$ for traditional actor update.

that Newton's method diverges from a random set of parameters, e.g., if their values are too large and with that feedback matrix even the short-term integral gets very large values and numerical problems occur, or, the proposed clipping might cause oscillations. In these relatively rare cases, the fastest way to solve the problem is through another initialization. Fig. 3 shows

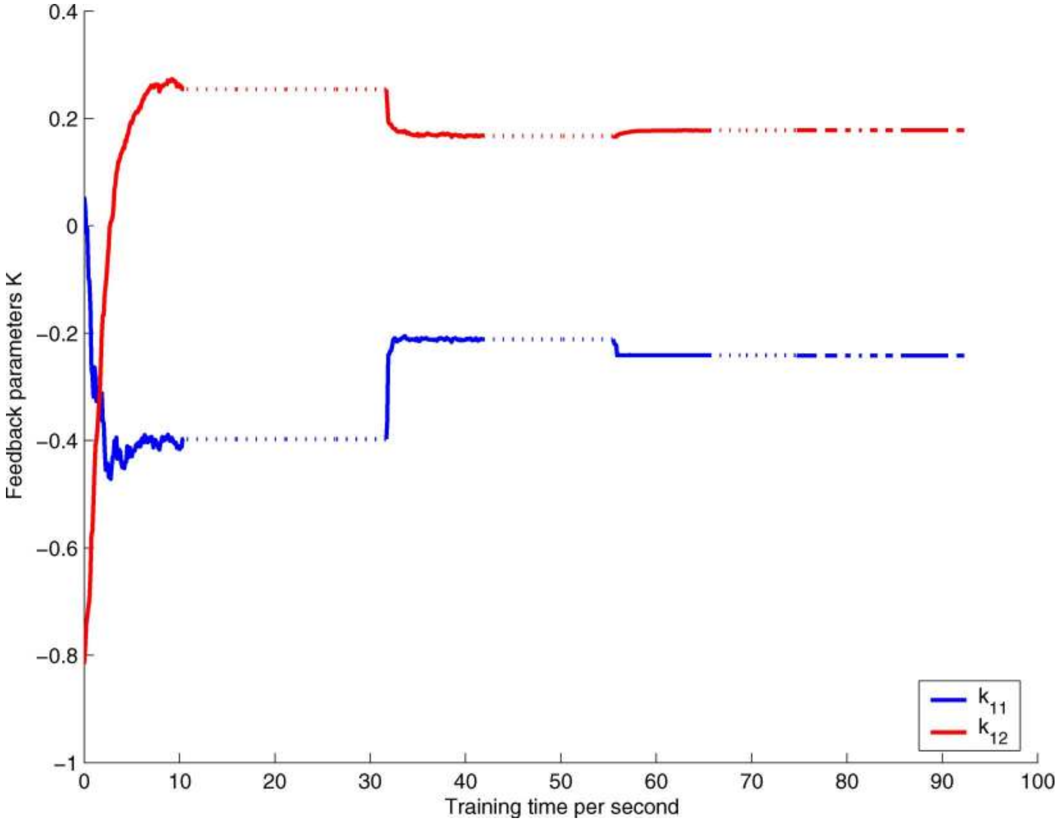

Fig. 8. Trajectory of the parameters $\mathbf{K}$ for traditional actor update. Parameter accuracy is less than 10^{-5} before stopping.

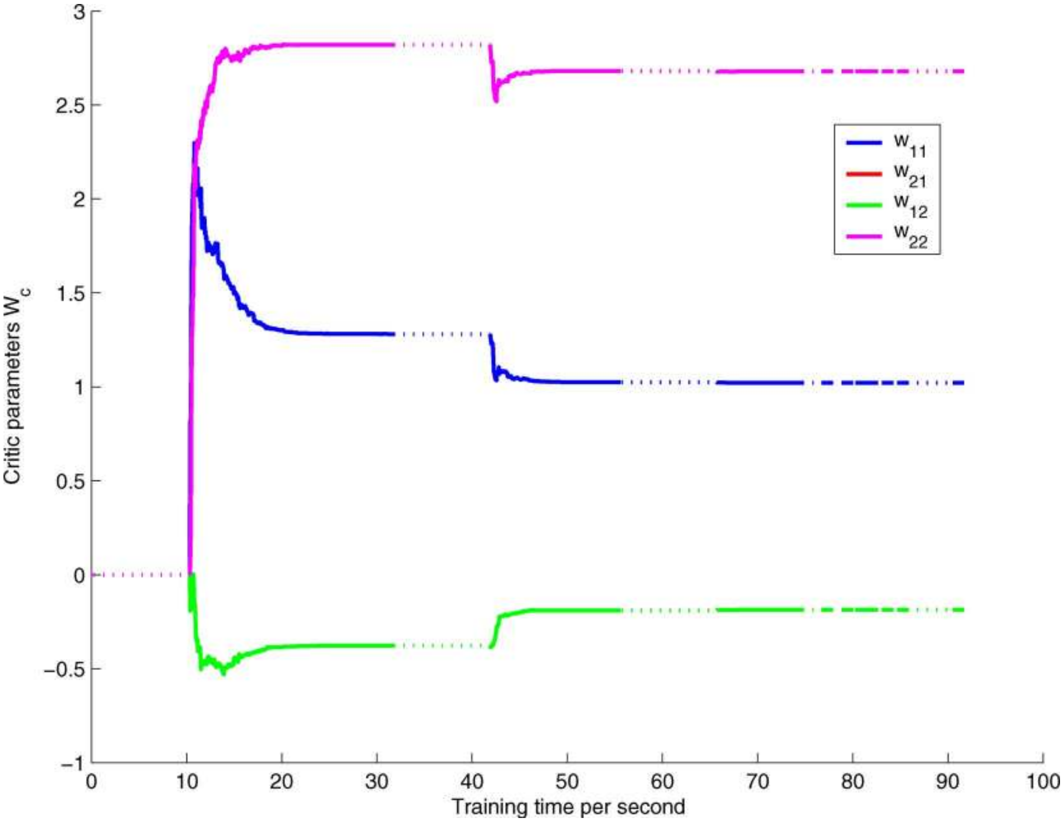

Fig. 9. Trajectory of critic parameters $\mathbf{W}_c$ for traditional actor update.

adaptation for the critic parameters $\mathbf{w}_c$. Remarkably, the linearized critic update proposed in Section III works well, especially when actor changes are of significant magnitudes and a 4% improvement could be achieved. However, it has to be mentioned that a gradient descent update will also converge fast in the beginning because errors are large. In this simple LQR

$$\frac{d^2U}{dw_i dw_j} = \int_{t_0}^{t_1} \left[\sum_{k=1}^{\dim(\mathbf{x})} \left(\frac{d^2 x_k}{dw_i dw_j} \frac{\partial \tilde{\phi}}{\partial x_k} + \frac{d}{dw_i}\left(\frac{\partial \tilde{\phi}}{\partial x_k}\right) \frac{dx_k}{dw_i} \right) + \frac{d}{dw_i}\left(\frac{\partial \tilde{\phi}}{\partial w_j}\right) \right] dt \tag{94}$$

$$\frac{d}{dw_i}\left(\frac{\partial \tilde{\phi}}{\partial x_k}\right) = \sum_{l=1}^{\dim(\mathbf{x})} \frac{dx_l}{dw_i} \frac{\partial^2 \tilde{\phi}}{\partial x_l \partial x_k} + \frac{\partial^2 \tilde{\phi}}{\partial w_i \partial x_k} = \sum_{l=1}^{\dim(\mathbf{x})} q_i^l \frac{\partial^2 \tilde{\phi}}{\partial x_l \partial x_k} + \frac{\partial^2 \tilde{\phi}}{\partial w_i \partial x_k} \tag{95}$$

$$\frac{d}{dw_i}\left(\frac{\partial \tilde{\phi}}{\partial w_j}\right) = \sum_{l=1}^{\dim(\mathbf{x})} \frac{dx_l}{dw_i} \frac{\partial^2 \tilde{\phi}}{\partial x_l \partial w_j} + \frac{\partial^2 \tilde{\phi}}{\partial w_i \partial w_j} = \sum_{l=1}^{\dim(\mathbf{x})} q_i^l \frac{\partial^2 \tilde{\phi}}{\partial x_l \partial w_j} + \frac{\partial^2 \tilde{\phi}}{\partial w_i \partial w_j} \tag{96}$$

$$\frac{d^2U}{dw_i dw_j} = \int_{t_0}^{t_1} \left[\sum_{k=1}^{\dim(\mathbf{x})} \left(r_{ij}^k \frac{\partial \tilde{\phi}}{\partial x_k} + \left(\sum_{l=1}^{\dim(\mathbf{x})} q_i^l \frac{\partial^2 \tilde{\phi}}{\partial x_l \partial x_k} + \frac{\partial^2 \tilde{\phi}}{\partial w_i \partial x_k} \right) q_j^k \right) + \sum_{l=1}^{\dim(\mathbf{x})} q_i^l \frac{\partial^2 \tilde{\phi}}{\partial x_l \partial w_j} + \frac{\partial^2 \tilde{\phi}}{\partial w_i \partial w_j} \right] dt \tag{97}$$

model, it is better to have the critic converging at every step, rather then swapping prematurely to update the controller because overall convergence is achieved in only a few iterations and highly accurate controller parameter demands very low errors in the Bellman "error" equation.

2) $\operatorname{rank}(\mathbf{K}) < \dim(\mathbf{x})$*:* Similar observations are made as in the case with a fully ranked feedback matrix and after only four actor–critic cycles the optimal values are achieved within an error of 10^{-5}. Figs. 4 and 5 show actor and critic weight adaptation, respectively.

D. Results for Continuous ACDs With Traditional Controller Update

Traditional actor update is achieved by forcing (25) and (26) to zero and updating actor weights according to

$$\mathbf{w}_a^{k+1} = \mathbf{w}_a^k - \eta_a \frac{dJ}{d\mathbf{w}_a} \tag{91}$$

with some step-size $\eta_a > 0$ and k indicating an iteration index. It is evident that compared with the second-order method it is more difficult to obtain accurate solutions when the gradient has to be zero, whereas the proposed Newton method can still do updates from the second order terms and obtain accurate controller weights. Critic training is the same as with Newton's method, apart from the concurrent actor–critic correction in Newton's method.

1) $\operatorname{rank}(\mathbf{K}) = \dim(\mathbf{x})$*:* The performance depends very much on the desired controller accuracy (see Figs. 6 and 7). For controller parameter accuracy beyond 10^{-5}, parameters need more attention. $\eta_a = 0.5$ and a maximum of 100 actor updates are allowed before switching to the critic training.

2) $\operatorname{rank}(\mathbf{K}) < \dim(\mathbf{x})$*:* In this more difficult case, controller parameter accuracy to 10^{-4} was achieved with reasonable training time (see Figs. 8 and 9). Step-size $\eta_a = 0.1$ as well as a maximum of 100 actor updates are allowed before switching.

V. Conclusion

A second-order adaptation for controller parameters for a continuous-time ACD has been developed in this paper. There are some valid reasons for using continuous-time ACDs. Plants are often described by differential equations such as (1) and those equations can be used directly, without discretization. This is done implicitly by the integration routine, which offers a second advantage, because an automatic step-size integrator does an adaptive sampling for free. An encapsulated approach of plant, controller and critic has been used which is a modular approach in contrast to a heterogenous approach sometimes used with a recurrent network trained by the RTRL algorithm, which often results in a much bigger network state $\mathbf{x}_n$ than the actual system state $\mathbf{x}$ of the proposed method, which otherwise may be seen as a somewhat continuous-time version thereof. This is important from a practical point of view as the computational complexity of the RTRL algorithm in space and time raises with $O(N^3)$ and $O(N^4)$, respectively, where N is the dimension of the state [6].

Another advantage of continuous-time ACD is the introduction of a correction algorithm for concurrent actor–critic training. This allows critic correction just after an actor training cycle to keep the Bellman optimality condition correct to first-order approximation of the introduced policy change by the actor update. This works well, at least for low-dimensional systems as in the demonstrated LQR example. For more complicated systems, it might be of advantage to approximate global cost functions by local quadratic ones as was done very successfully by Ferrari in [32]. However, the equations provided in theAppendix allow for the development of any systems and any cost functions, not only quadratic ones, so long as they are sufficiently differentiable.

Appendix

$$\frac{dU}{d\mathbf{w}_a} = \int_{t_0}^{t_1} \left(\frac{d\mathbf{x}^T}{d\mathbf{w}_a} \frac{\partial \tilde{\phi}}{\partial \mathbf{x}} + \frac{\partial \tilde{\phi}}{\partial \mathbf{w}_a} \right) dt \tag{92}$$

$$\frac{d^2U}{d\mathbf{w}_a^2} = \int_{t_0}^{t_1} \frac{d}{d\mathbf{w}_a} \left(\sum_{k=1}^{\dim(\mathbf{x})} \frac{dx_k}{d\mathbf{w}_a} \frac{\partial \tilde{\phi}}{\partial x_k} + \frac{\partial \tilde{\phi}}{\partial \mathbf{w}_a} \right) dt \tag{93}$$

or for one element see equations (94)–(97), shown at the top of the page, with the abbreviations

$$\mathbf{w}_a := \left[w_1, \ldots, w_i, \ldots, w_{\dim(\mathbf{w}_a)} \right]^T \tag{98}$$

$$q := \frac{d\mathbf{x}^T}{d\mathbf{w}_a} \tag{99}$$

$$q_j^k := \frac{dx_k}{dw_j} \tag{100}$$

$$r := \frac{d^2\mathbf{x}^T}{d\mathbf{w}_a^2} = \frac{d}{d\mathbf{w}_a}\left(\frac{d\mathbf{x}^T}{d\mathbf{w}_a}\right) = \frac{d}{d\mathbf{w}_a}(q) \tag{101}$$

$$r_{ij}^k := \frac{d}{dw_i}\left(q_j^k\right) = \frac{d}{dw_i}\left(\frac{dx_k}{dw_j}\right) = \frac{d^2 x_k}{dw_i dw_j} \tag{102}$$

$$\frac{\partial^2 h_k}{\partial x_n \partial x_m} = \frac{\partial}{\partial x_n}\left(\frac{\partial f_k}{\partial x_m} + \sum_{p=1}^{\dim(\mathbf{u})} \frac{\partial g_p}{\partial x_m}\frac{\partial f_k}{\partial u_p}\right) \tag{115}$$

$$= \frac{\partial^2 f_k}{\partial x_n \partial x_m} + \sum_{p=1}^{\dim(\mathbf{u})} \frac{\partial g_p}{\partial x_n}\frac{\partial^2 f_k}{\partial u_p \partial x_m} + \sum_{p=1}^{\dim(\mathbf{u})}\left(\frac{\partial^2 g_p}{\partial x_n \partial x_m}\frac{\partial f_k}{\partial u_p} + \left(\frac{\partial^2 f_k}{\partial x_n \partial u_p} + \sum_{r=1}^{\dim(\mathbf{u})}\frac{\partial g_r}{\partial x_n}\frac{\partial^2 f_k}{\partial u_r \partial u_p}\right)\frac{\partial g_p}{\partial x_m}\right) \tag{116}$$

$$\frac{\partial^2 h_k}{\partial w_i \partial x_m} = \frac{\partial}{\partial w_i}\left(\frac{\partial f_k}{\partial x_m} + \sum_{p=1}^{\dim(\mathbf{u})} \frac{\partial g_p}{\partial x_m}\frac{\partial f_k}{\partial u_p}\right) \tag{117}$$

$$= \sum_{p=1}^{\dim(\mathbf{u})}\frac{\partial g_p}{\partial w_i}\frac{\partial^2 f_k}{\partial u_p \partial x_m} + \sum_{p=1}^{\dim(\mathbf{u})}\left(\frac{\partial^2 g_p}{\partial w_i \partial x_m}\frac{\partial f_k}{\partial u_p} + \left(\sum_{r=1}^{\dim(\mathbf{u})}\frac{\partial g_r}{\partial w_i}\frac{\partial^2 f_k}{\partial u_r \partial u_p}\right)\frac{\partial g_p}{\partial x_m}\right) \tag{118}$$

$$= \sum_{p=1}^{\dim(\mathbf{u})}\left(\frac{\partial g_p}{\partial w_i}\frac{\partial^2 f_k}{\partial u_p \partial x_m} + \frac{\partial^2 g_p}{\partial w_i \partial x_m}\frac{\partial f_k}{\partial u_p} + \left(\sum_{r=1}^{\dim(\mathbf{u})}\frac{\partial g_r}{\partial w_i}\frac{\partial^2 f_k}{\partial u_r \partial u_p}\right)\frac{\partial g_p}{\partial x_m}\right) \tag{119}$$

$$\frac{\partial^2 h_k}{\partial x_n \partial w_j} = \sum_{p=1}^{\dim(\mathbf{u})}\left(\frac{\partial^2 g_p}{\partial x_n \partial w_j \partial u_p}\frac{\partial f_k}{} + \left(\frac{\partial^2 f_k}{\partial x_n \partial u_p} + \sum_{r=1}^{\dim(\mathbf{u})}\frac{\partial g_r}{\partial x_n}\frac{\partial^2 f_k}{\partial u_r \partial u_p}\right)\frac{\partial g_p}{\partial w_j}\right) \tag{120}$$

$$= \sum_{p=1}^{\dim(\mathbf{u})}\left(\frac{\partial^2 g_p}{\partial x_n \partial w_j}\frac{\partial f_k}{\partial u_p} + \frac{\partial^2 f_k}{\partial x_n \partial u_p}\frac{\partial g_p}{\partial w_j} + \left(\sum_{r=1}^{\dim(\mathbf{u})}\frac{\partial g_r}{\partial x_n}\frac{\partial^2 f_k}{\partial u_r \partial u_p}\right)\frac{\partial g_p}{\partial w_j}\right) \tag{121}$$

$$\frac{\partial^2 h_k}{\partial w_i \partial w_j} = \sum_{p=1}^{\dim(\mathbf{u})}\left(\frac{\partial^2 g_p}{\partial w_i \partial w_j}\frac{\partial f_k}{\partial u_p} + \left(\sum_{r=1}^{\dim(\mathbf{u})}\frac{\partial g_r}{\partial w_i}\frac{\partial^2 f_k}{\partial u_r \partial u_p}\right)\frac{\partial g_p}{\partial w_j}\right). \tag{122}$$

and the relations for their total time derivatives

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}; \mathbf{w}_a)) = \mathbf{h}(\mathbf{x}; \mathbf{w}_a) \tag{103}$$

$$\dot{q} = \frac{d\dot{\mathbf{x}}^T}{d\mathbf{w}_a} = \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{\partial \mathbf{h}^T}{\partial \mathbf{x}} + \frac{\partial \mathbf{h}^T}{\partial \mathbf{w}_a} \tag{104}$$

$$\dot{q}_j^k = \sum_{m=1}^{\dim(\mathbf{x})} q_j^m \frac{\partial h_k}{\partial x_m} + \frac{\partial h_k}{\partial w_j} \tag{105}$$

$$\dot{r}_{ij}^k = \frac{d}{dw_i}\left(\frac{d\dot{x}_k}{dw_j}\right)$$

$$= \frac{d}{dw_i}\left(\dot{q}_j^k\right) = \frac{d}{dw_i}\left(\sum_{m=1}^{\dim(\mathbf{x})} q_j^m \frac{\partial h_k}{\partial x_m} + \frac{\partial h_k}{\partial w_j}\right) \tag{106}$$

$$= \sum_{m=1}^{\dim(\mathbf{x})}\left(\frac{d}{dw_i}\left(q_j^m\right)\frac{\partial h_k}{\partial x_m} + \frac{d}{dw_i}\left(\frac{\partial h_k}{\partial x_m}\right)q_j^m\right) + \frac{d}{dw_i}\left(\frac{\partial h_k}{\partial w_j}\right) \tag{107}$$

with

$$\frac{d}{dw_i}\left(\frac{\partial h_k}{\partial x_m}\right) = \sum_{n=1}^{\dim(\mathbf{x})}\frac{dx_n}{dw_i}\frac{\partial^2 h_k}{\partial x_n \partial x_m} + \frac{\partial^2 h_k}{\partial w_i \partial x_m}$$

$$= \sum_{n=1}^{\dim(\mathbf{x})} q_i^n \frac{\partial^2 h_k}{\partial x_n \partial x_m} + \frac{\partial^2 h_k}{\partial w_i \partial x_m} \tag{108}$$

$$\frac{d}{dw_i}\left(\frac{\partial h_k}{\partial w_j}\right) = \sum_{n=1}^{\dim(\mathbf{x})}\frac{dx_n}{dw_i}\frac{\partial^2 h_k}{\partial x_n \partial w_j} + \frac{\partial^2 h_k}{\partial w_i \partial w_j}$$

$$= \sum_{n=1}^{\dim(\mathbf{x})} q_i^n \frac{\partial^2 h_k}{\partial x_n \partial w_j} + \frac{\partial^2 h_k}{\partial w_i \partial w_i} \tag{109}$$

it is

$$\dot{r}_{ij}^k r = \sum_{m=1}^{\dim(\mathbf{x})}\left(r_{ij}^m \frac{\partial h_k}{\partial x_m} + \sum_{n=1}^{\dim(\mathbf{x})} q_i^n \frac{\partial^2 h_k}{\partial x_n \partial x_m} q_j^m + \frac{\partial^2 h_k}{\partial w_i \partial x_m} q_j^m\right) + \sum_{n=1}^{\dim(\mathbf{x})} q_i^n \frac{\partial^2 h_k}{\partial x_n \partial w_j} + \frac{\partial^2 h_k}{\partial w_i \partial w_j}. \tag{110}$$

All these differential equations can be solved easily in principle, knowing that the initial condition is always zero. The tricky part is the complexity of the formulas achieved by using the product and chain rules and expressing derivatives of h in terms of derivatives of f and g. Depending on the complexity of the system at hand, it might be simpler to find $\mathbf{h}(\mathbf{x}, \mathbf{w}) = \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}; \mathbf{w}_a))$ and use discrete differences to approximate partial derivatives.

The first-order partial derivatives of $\mathbf{h}$ with respect to the states or weights are

$$\frac{\partial \mathbf{h}^T}{\partial \mathbf{x}} = \frac{\partial \mathbf{f}^T}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}^T}{\partial \mathbf{x}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}} \tag{111}$$

$$\frac{\partial h_k}{\partial x_m} = \frac{\partial f_k}{\partial x_m} + \sum_{p=1}^{\dim(\mathbf{u})}\frac{\partial g_p}{\partial x_m}\frac{\partial f_k}{\partial u_p} \qquad \forall k = 1, \ldots, \dim(\mathbf{x}) \qquad \forall m = 1, \ldots, \dim(\mathbf{x}) \tag{112}$$

$$\frac{\partial \mathbf{h}^T}{\partial \mathbf{w}_a} = \frac{\partial \mathbf{g}^T}{\partial \mathbf{w}_a}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}} \tag{113}$$

$$\frac{\partial h_k}{\partial w_j} = \sum_{p=1}^{\dim(\mathbf{u})}\frac{\partial g_p}{\partial w_j}\frac{\partial f_k}{\partial u_p} \tag{114}$$

and the second-order partial derivatives of $\mathbf{h}$ with respect to the states and/or weights are shown in (115)–(122) at the top of the page. The only remaining quantities to calculate are the partial

$$\frac{\partial^2 \tilde{\phi}}{\partial x_l \partial x_k} = \frac{\partial}{\partial x_l}\left(\frac{\partial \phi}{\partial x_k} + \sum_{p=1}^{\dim(\mathbf{x})} \frac{\partial h_p}{\partial x_k}\frac{\partial \phi}{\partial \dot{x}_p}\right) \tag{127}$$

$$= \frac{\partial^2 \phi}{\partial x_l \partial x_k} + \sum_{p=1}^{\dim(\mathbf{x})} \frac{\partial h_p}{\partial x_l}\frac{\partial^2 \phi}{\partial \dot{x}_p \partial x_k} + \sum_{p=1}^{\dim(\mathbf{x})}\left(\frac{\partial^2 h_p}{\partial x_l \partial x_k}\frac{\partial \phi}{\partial \dot{x}_p} + \left(\frac{\partial^2 \phi}{\partial x_l \partial \dot{x}_p} + \sum_{q=1}^{\dim(\mathbf{x})} \frac{\partial h_q}{\partial x_l}\frac{\partial^2 \phi}{\partial \dot{x}_q \partial \dot{x}_p}\right)\frac{\partial h_p}{\partial x_k}\right) \tag{128}$$

$$\frac{\partial^2 \tilde{\phi}}{\partial w_i \partial x_k} = \frac{\partial}{\partial w_i}\left(\frac{\partial \phi}{\partial x_k} + \sum_{p=1}^{\dim(\mathbf{x})} \frac{\partial h_p}{\partial x_k}\frac{\partial \phi}{\partial \dot{x}_p}\right) \tag{129}$$

$$= \sum_{p=1}^{\dim(\mathbf{x})} \frac{\partial h_p}{\partial w_i}\frac{\partial^2 \phi}{\partial \dot{x}_p \partial x_k} + \sum_{p=1}^{\dim(\mathbf{x})}\left(\frac{\partial^2 h_p}{\partial w_i \partial x_k}\frac{\partial \phi}{\partial \dot{x}_p} + \left(\sum_{q=1}^{\dim(\mathbf{x})} \frac{\partial h_q}{\partial w_i}\frac{\partial^2 \phi}{\partial \dot{x}_q \partial \dot{x}_p}\right)\frac{\partial h_p}{\partial x_k}\right) \tag{130}$$

$$\frac{\partial^2 \tilde{\phi}}{\partial x_k \partial w_j} = \frac{\partial}{\partial x_k}\left(\sum_{p=1}^{\dim(\mathbf{x})} \frac{\partial h_p}{\partial w_j}\frac{\partial \phi}{\partial \dot{x}_p}\right) \tag{131}$$

$$= \sum_{p=1}^{\dim(\mathbf{x})}\left(\frac{\partial^2 h_p}{\partial x_k \partial w_j}\frac{\partial \phi}{\partial \dot{x}_p} + \left(\frac{\partial^2 \phi}{\partial x_k \partial \dot{x}_p} + \sum_{q=1}^{\dim(\mathbf{x})} \frac{\partial h_q}{\partial x_k}\frac{\partial^2 \phi}{\partial \dot{x}_q \dot{x}_p}\right)\frac{\partial h_p}{\partial w_j}\right) \tag{132}$$

$$\frac{\partial^2 \tilde{\phi}}{\partial w_i \partial w_j} = \frac{\partial}{\partial w_i}\left(\sum_{l=1}^{\dim(\mathbf{x})} \frac{\partial h_l}{\partial w_j}\frac{\partial \phi}{\partial \dot{x}_l}\right) \tag{133}$$

$$= \sum_{l=1}^{\dim(\mathbf{x})}\left(\frac{\partial^2 h_l}{\partial w_i \partial w_j}\frac{\partial \phi}{\partial \dot{x}_l} + \left(\sum_{p=1}^{\dim(\mathbf{x})} \frac{\partial h_p}{\partial w_i}\frac{\partial^2 \phi}{\partial \dot{x}_p \dot{x}_l}\right)\frac{\partial h_l}{\partial w_j}\right). \tag{134}$$

derivatives of $\tilde{\phi}(\mathbf{x}, \mathbf{w}_a)$ with respect to the weights or states. First-order partials are

$$\frac{\partial \tilde{\phi}}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}}\left(\phi(\mathbf{x}, \mathbf{h}(\mathbf{x}, \mathbf{w}_a))\right) = \frac{\partial \phi}{\partial \mathbf{x}} + \frac{\partial \mathbf{h}^T}{\partial \mathbf{x}}\frac{\partial \phi}{\partial \dot{\mathbf{x}}} \tag{123}$$

$$\frac{\partial \tilde{\phi}}{\partial x_k} = \frac{\partial \phi}{\partial x_k} + \sum_{l=1}^{\dim(\mathbf{x})} \frac{\partial h_l}{\partial x_k}\frac{\partial \phi}{\partial \dot{x}_l} \tag{124}$$

$$\frac{\partial \tilde{\phi}}{\partial \mathbf{w}_a} = \frac{\partial \mathbf{h}^T}{\partial \mathbf{w}_a}\frac{\partial \phi}{\partial \dot{\mathbf{x}}} \tag{125}$$

$$\frac{\partial \tilde{\phi}}{\partial w_j} = \sum_{l=1}^{\dim(\mathbf{x})} \frac{\partial h_l}{\partial w_j}\frac{\partial \phi}{\partial \dot{x}_l} \tag{126}$$

and second-order partials are shown in (127)–(134) at the top of the page.

ACKNOWLEDGMENT

T. Hanselmann would like to thank D. V. Prokhorov and P. J. Werbos for pointing to some work regarding the EKF algorithm, initiated at Ford and their valuable feedback as the Ph.D. reviewers and useful suggestions made therein which had an influence on this paper. The authors would also like to thank the three anonymous reviewers for their valuable feedback that have improved the current paper significantly.

REFERENCES

[1] J. S. Dalton, "A critic based system for neural guidance and control," Ph.D. dissertation, Elect. Eng. Dept., Univ. Missouri—Rolla, Rolla, MO, 1994.

[2] J. S. Dalton and S. N. Balakrishnan, "A neighboring optimal adaptive critic for missile guidance," *Math. Comput. Model.*, vol. 23, no. 1-2, pp. 175–188, 1996.

[3] P. Werbos, "Stable adaptive control using new critic designs" 1998 [Online]. Available: http://xxx.lanl.gov/abs/adap-org/9810001

[4] T. Hanselmann, "Approximate dynamic programming with adaptive critics and the algebraic perceptron as a fast neural network related to support vector machines," Ph.D. dissertation, Univ. Western Australia, Perth, W. A., Australia, 2003.

[5] T. Hanselmann, L. Noakes, and A. Zaknich, "Continuous adaptive critic designs," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, Montreal, QC, Canada, Jul.-Aug. 31-4, 2005, pp. 3001–3006.

[6] S. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks and their computational complexity," in *Backpropagation: Theory, Architectures and Applications*, Chauvin and Rumelhart, Eds.: LEA, 1995, pp. 433–486.

[7] S. Haykin, *Neural Networks a Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1998, ch. 15.

[8] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, pp. 270–280, 1989.

[9] T. Hanselmann, A. Zaknich, and Y. Attikiouzel, "Connection between BPTT and RTRL," in *Computational Intelligence and Applications*, N. E. Mastorakis, Ed. Athens, Greece: World Scientific and Engineering Press, 1999, pp. 97–102.

[10] J. Si, A. G. Barto, W. B. Powell, and D. I. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*, J. Si, A. G. Barto, W. B. Powell, and D. I. Wunsch, Eds. New York: IEEE Press/Wiley, 2004.

[11] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic design," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.

[12] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.

[13] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming" Athena Scientific, Belmont, MA, 1996 [Online]. Available: http://world.std.com/~athenasc

[14] J. Tsitsiklis and B. V. Roy, "An analysis of temporal difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997.

[15] X. Xu, D. Hu, and X. Lu, "Kernel-based least-squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw.*, 2006, submitted for publication.

[16] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.

[17] P. Dayan, "The convergence of $\mathrm{td}(\lambda)$ for general λ," *Mach. Learn.*, vol. 8, pp. 341–362, 1992.

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

[19] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA, 1974.

[20] D. V. Prokhorov, "Adaptive critic designs and their applications," Ph.D. dissertation, Texas Tech Univ., Lubbock, TX, 1997.

[21] L. C. Baird, A. Prieditis and e. S. Russell, Eds., "Residual algorithms: Reinforcement learning with function approximation," in *Proc. 12th Int. Conf. Mach. Learn.*, San Francisco, CA, Jul. 9–12, 1995, pp. 30–37.

[22] D. Prokhorov and D. C. Wunsch, "Convergence of critic-based training," in *IEEE Int. Conf. Syst., Man, Cybern.*, Orlando, FL, Oct. 12–15, 1997, vol. 4, pp. 3057–3060.

[23] H. J. Kushner and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Berlin, Germany: Springer-Verlag, 1978.

[24] G. Cybenko, R. Gray, and M. Katsuhiro, "Q-learning: A tutorial and extensions," in *Mathematics of Artificial Neural Networks*. London, U.K.: Oxford Univ. Press, 1995.

[25] P. Werbos, "How to use the chain rule for ordered derivatives," in *Handbook of Intelligent Control*, D. A. White and D. A. Sofge, Eds. New York: Van Nostrand Reinhold, 1992, ch. 10.6.

[26] P. Eaton, D. Prohorov, and D. Wunsch, "Neurocontroller alternatives for fuzzy ball-and-beam systems with nonlinear, nonuniform friction," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 423–435, Mar. 2000.

[27] S. Haykin, *Neural Networks a Comprehensive Foundation*, Second ed. Upper Saddle River, NJ: Prentice-Hall, 1998.

[28] S. S. Ge, C. C. Hang, and T. Zhang, "Adaptive neural network control of nonlinear systems by state and output feedback," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 29, no. 6, pp. 818–828, Dec. 1999.

[29] S. S. Ge and C. Wang, "Adaptive NN control of uncertain nonlinear pure-feedback systems," *Automatica*, vol. 38, no. 4, pp. 671–682, Dec. 2002.

[30] W. L. Brogan, *Modern Control Theory*, 3rd ed. Upper Saddle River, New Jersey: Prentice-Hall, 1991.

[31] M. Bray, E. Koller-Meier, N. Schraudolph, and L. Van Gool, "Stochastic meta-descent for tracking articulated structures," in *IEEE Workshop Comput. Vis. Pattern Recognit.*, Jun. 2004, p. 7.

[32] S. Ferrari and R. Stengel, "On-line adaptive critic flight control," *J. Guid., Control Dyn.*, vol. 27, no. 5, pp. 777–786, Sep.-Oct. 2004.

Thomas Hanselmann (M'95) was born in Switzerland. He received the degree in electrical engineering from the Swiss Federal Institute of Technology (ETHZ), Zurich, Switzerland, in 1993 and the Ph.D. degree from the School of Electrical, Electronic and Computer Engineering and the School of Mathematics and Statistics, University of Western Australia (UWA), Crawley, W.A., Australia, in 2004.

He was a Software Engineer in Zurich for several years. From 2003 to 2004, he was an Associate Lecturer in the School of Electrical, Electronic and Computer Engineering, UWA. Currently, he is a Research Fellow at the University of Melbourne, Parkville, Vic., Australia. His research interests are intelligent signal processing for decision making and control, particularly approximative dynamic programming and statistical learning theory for fast adaptive algorithms.

Lyle Noakes was born in New Zealand. He received the B.Sc. degree in mathematics from the University of Otago, Dunedin, New Zealand, in 1969 and the M.Sc. and Ph.D. degrees in mathematics from Oxford University, Oxford, U.K., in 1970 and 1974, respectively.

He is a Professor in Mathematics at the University of Western Australia, Head of the Pure Mathematics and Chair of UWA's Mathematics and Information Science Discipline Group. He is an author or coauthor of 79 refereed research papers in mathematics and applications, especially applications of differential geometry.

Dr. Noakes received the top Oxford Senior Mathematics Prize for the Ph.D. in algebraic topology in 1974.

Anthony Zaknich (M'87–SM'00) was born in Vela Luka, Croatia, on March 27, 1952 and immigrated to Australia in 1957. He received the B.E. degree in electronics and the M.E.Sc. degree in electronics from the University of Western Australia (UWA), Crawley, W.A., Australia, in 1974 and 1986, respectively, the B.A. and B.Sc. degrees in psychology from Ambassador University, Pasadena, CA, both in 1978, and the Ph.D. degree from UWA in 1996.

He is currently an Associate Professor in the School of Engineering Science, Murdoch University, Perth, W.A., Australia, and an Adjunct Associate Professor at UWA, Centre for Intelligent Information Processing Systems (CIIPS). From 1990 to 1999, he was a Technical Manager for Industry Projects working as a Research Fellow and Lecturer at CIIPS in the Electrical and Electronics Engineering Department, UWA. His main work at CIIPS was involved with supervision, teaching, research and development related to signal processing and artificial neural networks at the undergraduate, postgraduate, and professional-development levels. Previously, he was involved in the research and development of underwater control and acoustic signaling systems in private enterprise, and also in the establishment of a public company, Nautronix Ltd., producing and marketing products in these areas for the international market. He has supervised numerous honors and ten postgraduate research projects, including three Ph.D. dissertations. He has also authored/coauthored more than 62 refereed papers in technical journals and conference proceedings, has contributed five research book chapters, and authored two books in his areas of interest since 1988. His research interest is in philosophy, theory and applications of intelligent signal processing—statistical learning theory, support vector machines, pattern recognition, artificial neural networks, adaptive systems, signal and image processing, system and signal modeling and identification, time-frequency filtering, time delay spectrometry, audio, Hi-Fi, and underwater acoustic communications systems.

Dr. Zaknich is a Member of the Audio Engineering Society (AES). He served on the IEEE Western Australian Regional Interest Group Committee on Neural Networks at various times since 1993.