Continuous Adaptive Critic Designs

Thomas Hanselmann
Dept. of Electrical and Electronic Eng.
The University of Melbourne
Parkville, VIC 3010, Australia
E-mail: t.hanselmann@ee.mu.oz.au

Lyle Noakes
School of Mathematics and Statistics
The University of Western Australia
Perth, WA 6009, Australia
E-mail: lyle@maths.uwa.edu.au

Anthony Zaknich
School of Engineering Science
Murdoch University
Perth, WA 6150, Australia
E-mail: tonko@ieee.org

Abstract— **A continuous 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. A second order actor adaptation, based on Newton's method, is established for fast actor convergence. Also a fast critic update for concurrent actor-critic training is outlined that keeps the Bellman optimality correct to first order approximation after actor changes.**

I. Introduction

There are many terminologies used, depending from which aspects the problem is viewed from, but basically adaptive critic designs (ACDs) are a framework to approximate dynamic programming and are used in decision making with the objective of a minimal long-term cost. ACDs approximate dynamic programming in the way that they parameterize the long-term cost, $J(\mathbf{x})$, or its derivative (λ-critic, dual heuristic programming (DHP)), or a combination thereof (global dual heuristic programming GDHP). Other terminologies are also used, especially reinforcement learning, which was inspired from a biological view-point. There are only a few publications dealing with continuous adaptive critics [1], [2], [3]. In this paper, the aim is to extend the discrete approach to continuous 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).

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

There is no space here to introduce BPTT and RTRL in depth, but their details can be found in [4], [5] and [6], [7]. BPTT calculates total derivatives of a quantity that is a function of previously evaluated functions with respect to some previous argument, as seen by (4) and (6). RTRL, calculates total derivatives $\frac{d\mathbf{x}^T(t)}{d\mathbf{w}}$ forward in time based on a transition matrix $\Phi(t)$. In the context of ACDs, function approximators are used like neural networks. Then, the state $\mathbf{x}(t)$ refers to all the nodes in a network and its dimensionality can be quite large.

II. Continuous Version of 'Ordered' Total Derivatives

A simple method for 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 formulae (5) or (6). This is basically the chain-rule and was first introduced by Werbos in the context of adapting parameters of a long-term cost-function [8]. The notation $\frac{\partial^+ z_n}{\partial z_k}$ means the total derivative

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

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

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

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 only of $\mathbf{x}$ and $\mathbf{w}$, so $\dot{\mathbf{x}} = \mathbf{h}(\mathbf{x}; \mathbf{w})$.

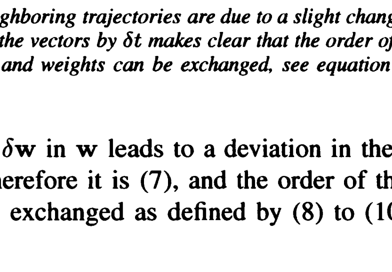

Fig. 1. *The neighboring trajectories are due to a slight change in the weights. Multiplying all the vectors by δt makes clear that the order of derivatives with respect to time and weights can be exchanged, see equation (7).*

A deviation $\delta\mathbf{w}$ in $\mathbf{w}$ leads to a deviation in the trajectory $\mathbf{x}$, say $\mathbf{x}_{\delta\mathbf{w}}$. Therefore it is (7), and the order of the differentiations can be exchanged as defined by (8) to (10). See figure 1.

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

$$\frac{d}{dt}\frac{d\mathbf{x}^T}{d\mathbf{w}} = \frac{d\mathbf{h}^T}{d\mathbf{w}} = \frac{d\mathbf{x}^T}{d\mathbf{w}}\frac{\partial \mathbf{h}^T}{\partial \mathbf{x}} + \frac{\partial \mathbf{h}^T}{\partial \mathbf{w}}, \quad (8)$$

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

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

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 $\frac{d\mathbf{x}^T}{d\mathbf{w}}$. Using a new variable $\mathbf{q}$, the differential equation can be rewritten as defined by (11) to (13), ready to be solved by a standard integration routine.

$$\mathbf{q} := \frac{d\mathbf{x}^T}{d\mathbf{w}} \quad (11)$$

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

with initial condition

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

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, only at infinitesimal time steps as defined by (14) to (17).

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

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

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

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

Again, this is the integral formulation of the differential equation (12) with initial condition (13) and $\frac{d\mathbf{x}^T}{d\mathbf{w}_a} =: \mathbf{q}$.
Therefore, the summation is exchanged by the integration and the partial derivative has to be included in 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 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 up into a part of total and partial derivatives, for indirect and direct influence on the target quantity, just as with the discrete case.
This trick of solving for a total derivative by integration is the key to continuous adaptive critics.

A. Continuous Adaptive Critics

For continuous adaptive critics the plant and the cost-density function are continuous and the one-step cost is an integral of a cost-density function over a sufficiently long time interval $[t_0, t_1]$. The short-term cost is given by (18).

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

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

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

As seen before, in adaptive critic designs an estimator is sought that is minimal 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.

$$\hat{J}(\mathbf{x}(t_0); \mathbf{w}_c) := \hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_0); \mathbf{w}_c) \quad (20)$$

$$= \int_{t_0}^{t_1} \phi(\mathbf{x}(t), \dot{\mathbf{x}}(t))dt + \int_{t_1}^{\infty} \phi(\mathbf{x}(t), \dot{\mathbf{x}}(t))dt \quad (21)$$

$$= U(t_0, t_1) + \hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1); \mathbf{w}_c) \quad (22)$$

Firstly, 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.

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

Applying an adaptation law to the critic parameters $\mathbf{w}_c$ to force the temporal error to zero, ensures optimality for the given policy $\hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a)$ with fixed parameters $\mathbf{w}_a$. For example, (24).

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

Secondly, the policy can be improved by forcing (25,27) to be zero.

$$\frac{d\left(U(t_0, t_1; \mathbf{w}_a) + \hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1); \mathbf{w}_c)\right)}{d\mathbf{w}_a} =$$

$$\frac{dU(t_0, t_1; \mathbf{w}_a)}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T(t_1)}{d\mathbf{w}_a}\frac{d\hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1); \mathbf{w}_c)}{d\mathbf{x}(t_1)} \quad (25)$$

$$\overset{!}{=} 0 \quad (26)$$

$$= \int_{t_0}^{t_1} \frac{d\mathbf{x}^T(t)}{d\mathbf{w}_a}\left[\frac{\partial \phi}{\partial \mathbf{x}} + \frac{\partial \hat{\mathbf{g}}^T}{\partial \mathbf{x}}\frac{\partial \mathbf{f}^T}{\partial \mathbf{u}}\frac{\partial \phi}{\partial \dot{\mathbf{x}}}\right]dt$$
$$+ \frac{d\mathbf{x}^T(t_1)}{d\mathbf{w}_a}\frac{d\hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_1); \mathbf{w}_c)}{d\mathbf{x}(t_1)} \quad (27)$$

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 $\frac{d\hat{J}^{\pi(\mathbf{w}_a)}(\mathbf{x}(t_0); \mathbf{w}_c)}{d\mathbf{w}_a}$, 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 contribution to the total derivative are always taken into account where as in the discrete case the total derivatives taken over one step only, misses out on the indirect contributions, as the following examples shows. Naturally, a multi-step discrete version of the temporal difference, starts approximating the continuous case and this disadvantage starts disappearing. An example is a 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 $\mathbf{w}$ is given by (28) which is equal to (29) because $\frac{d\mathbf{x}^T(t)}{d\mathbf{w}_a} = \mathbf{0}$.

$$\begin{aligned} \frac{d\mathbf{x}^T(t+1)}{d\mathbf{w}_a} &= \frac{d\mathbf{x}^T(t)}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T(t)}{d\mathbf{w}_a}\frac{\partial \mathbf{f}^T(\mathbf{x}(t), \mathbf{u}(t))}{\partial \mathbf{x}(t)} \\ &+ \frac{d\hat{\mathbf{g}}^T(\mathbf{x}(t); \mathbf{w}_a)}{d\mathbf{w}_a}\frac{\partial \mathbf{f}^T(\mathbf{x}(t), \mathbf{u}(t))}{\partial \mathbf{u}(t)} \quad (28) \\ &= \frac{\partial\hat{\mathbf{g}}^T(\mathbf{x}(t); \mathbf{w}_a)}{\partial\mathbf{w}_a}\frac{\partial \mathbf{f}^T(\mathbf{x}(t), \mathbf{u}(t))}{\partial \mathbf{u}(t)} \quad (29) \end{aligned}$$

If this procedure is now repeated at every time step and the starting time t_0 is always reset to the current time t, indirect influence through $\mathbf{x}(t)$ and all its later dependencies, like $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$, are always going to be missed. This can amount to a serious problem as substantial parts, like $\frac{\partial \mathbf{f}^T(\mathbf{x}(t),\mathbf{u}(t))}{\partial \mathbf{x}(t)}$ or $\frac{\partial \hat{\mathbf{g}}^T(\mathbf{x}(t),\mathbf{u}(t))}{\partial \mathbf{x}(t)}$, are ignored as well. That is the reason why $BPTT(h > 0)$ is so much more powerful than just having the instantaneous gradient as in $BPTT(0)$. The same applies to the continuous formulation adopted here with the additional benefit of having variable step size control from the integration routine. One final remark to $RTRL$ and $BPTT$. The $BPTT$ algorithm is considered more efficiently because in its recursive formulation, gradients are calculated with respect to a scalar target, while in $RTRL$ the quantity $\frac{\partial \mathbf{x}^T(t)}{\partial \mathbf{w}}$ 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. However, as $\mathbf{x}$ is the state vector of the system and not the vector of all nodes as in a simultaneous recurrent network (SRN) using the $RTRL$ algorithm, $\mathbf{x}$ is most likely to be of much smaller dimensionality. Therefore, having $\mathbf{q}$ as a matrix might not be a too severe drawback.

B. Second Order Adaptation for Actor Training

As seen before, the short-term cost from time t_0 to t_1, starting in state $\mathbf{x}(t_0)$ is given by (32).

$$\begin{aligned} U(\mathbf{x}(t_0), t_0, t_1; \mathbf{w}_a) &= \int_{t_0}^{t_1} \phi(\mathbf{x}, \dot{\mathbf{x}})dt \quad (30) \\ &= \int_{t_0}^{t_1} \phi(\mathbf{x}, \mathbf{f}(\mathbf{x}, \hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a)))dt \quad (31) \\ &= \int_{t_0}^{t_1} \tilde{\phi}(\mathbf{x}, \mathbf{w}_a)dt \quad (32) \end{aligned}$$

Assuming a stationary environment, the long-term cost in state $\mathbf{x}(t_0)$ and following the policy given by $\hat{\mathbf{g}}(\mathbf{x}; \mathbf{w}_a)$ is also known as Bellman's optimality condition as in (33) and (34).

$$\begin{aligned} J(\mathbf{x}(t_0); \mathbf{w}_a) &= U(\mathbf{x}(t_0), t_0, t_1; \mathbf{w}_a) + J(\mathbf{x}(t_1); \mathbf{w}_a) \quad (33) \\ J_0 &= U + J_1, \quad \text{for short.} \quad (34) \end{aligned}$$

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 adaptive critic designs 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 (35,36,37).

$$\begin{aligned} \frac{dJ(\mathbf{x}(t_0))}{d\mathbf{w}_a} &= \frac{dU(\mathbf{x}(t_0), t_0, t_1; \mathbf{w}_a)}{d\mathbf{w}_a} + \frac{dJ(\mathbf{x}(t_1))}{d\mathbf{w}_a} \overset{!}{=} \mathbf{0} \quad (35) \\ \frac{dJ_0}{d\mathbf{w}_a} &= \frac{dU}{d\mathbf{w}_a} + \frac{dJ_1}{d\mathbf{w}_a} \overset{!}{=} \mathbf{0}, \quad \text{for short} \quad (36) \\ &= \frac{dU}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{dJ_1}{d\mathbf{x}} \overset{!}{=} \mathbf{0} \quad (37) \end{aligned}$$

1) Newton's method: In traditional adaptive critic designs (35) is used to train the actor parameters via a simple gradient descent method. To speed up the traditional approach, Newton's method could be used, though with the additional cost of computing the Jacobian of the function $\frac{dJ_0}{d\mathbf{w}_a}$ with respect to $\mathbf{w}_a$. In the context here, Newton's method for zero search is given by (38) to (43).

$$F(X) = 0, \quad (38)$$

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

$$X^{k+1} := X^k - \left[\frac{\partial F}{\partial X}\right]^{-1} F(X^k), \quad (39)$$

identifying

$$\begin{aligned} F &:= \frac{dJ_0}{d\mathbf{w}_a} \quad (40) \\ X &:= \mathbf{w}_a \quad (41) \\ \frac{\partial F}{\partial X} &:= \frac{d^2 J_0}{d\mathbf{w}_a^2}, \quad (42) \end{aligned}$$

yields

$$\mathbf{w}_a^{k+1} := \mathbf{w}_a^k - \left[\frac{d^2 J_0}{d\mathbf{w}_a^2}\right]^{-1} \frac{dJ_0}{d\mathbf{w}_a} \quad (43)$$

To calculate the Jacobian, equation (37) is differentiated again with respect to $\mathbf{w}_a$, yielding (45) to (47), where $\frac{dJ_1}{d\mathbf{x}}$ and $\frac{d^2 J_1}{d\mathbf{x}^2}$ might be approximated by a backpropagated J-approximator or a λ-critic and by a backpropagated λ-critic, respectively.

$$\frac{d^2 J_0}{d\mathbf{w}_a^2} = \frac{d}{d\mathbf{w}_a}\left(\frac{dU}{d\mathbf{w}_a} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{dJ_1}{d\mathbf{x}}\right) \quad (44)$$

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

$$= \frac{d^2U}{d\mathbf{w}_a^2} + \frac{d^2\mathbf{x}^T}{d\mathbf{w}_a^2}\frac{dJ_1}{d\mathbf{x}} + \frac{d}{d\mathbf{w}_a}\left(\frac{dJ_1}{d\mathbf{x}}\right)\frac{d\mathbf{x}}{d\mathbf{w}_a} \quad (46)$$

$$= \frac{d^2U}{d\mathbf{w}_a^2} + \frac{d^2\mathbf{x}^T}{d\mathbf{w}_a^2}\frac{dJ_1}{d\mathbf{x}} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{d^2J_1}{d\mathbf{x}^2}\frac{d\mathbf{x}}{d\mathbf{w}_a} \quad (47)$$

It has to be mentioned that $\frac{d^2\mathbf{x}^T}{d\mathbf{w}_a^2}$ is a third order tensor, but with "the inner-product multiplication over the components of $\mathbf{x}$", the term $\frac{d^2\mathbf{x}^T}{d\mathbf{w}_a^2}\frac{dJ_1}{d\mathbf{x}}$ 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 later 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 (47) is used, given by (48), where the expectation is taken over a set of sampled start states $\mathbf{x}(t_0)$ according to their probability distribution from the domain of interest.

$$E\left\{\frac{d^2J_0}{d\mathbf{w}_a^2}\right\} = E\left\{\frac{d^2U}{d\mathbf{w}_a^2} + \sum_{k=1}^{dim(x)}\frac{d^2x_k}{d\mathbf{w}_a^2}\frac{dJ_1}{dx_k} + \frac{d\mathbf{x}^T}{d\mathbf{w}_a}\frac{d^2J_1}{d\mathbf{x}^2}\frac{d\mathbf{x}}{d\mathbf{w}_a}\right\} \quad (48)$$

All the necessary terms in (48) can be fully expanded but due to space limitations here will be publish later in an extended version of this paper.
For the first actor training a mid-term interval $[t_0, t_1]$ could be chosen with a critic output of zero, e.g. $\mathbf{w}_c \equiv \mathbf{0}$. 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).

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

$$\delta\mathbf{w}_c := -\eta_c\frac{\partial E_c}{\partial \mathbf{w}_c}E_c \quad (50)$$

After 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 (51) and (52) must hold.

$$U(\mathbf{x}_0; \mathbf{w}_a+\delta\mathbf{w}_a) + J^{\pi(\mathbf{w}_a+\delta\mathbf{w}_a)}(\mathbf{x}_{\delta\mathbf{w}_a}; \mathbf{w}_c+\delta\mathbf{w}_c) - J^{\pi(\mathbf{w}_a+\delta\mathbf{w}_a)}(\mathbf{x}_0; \mathbf{w}_c+\delta\mathbf{w}_c) \overset{!}{=} 0 \quad (51)$$

$$U(\mathbf{x}_0, t_0, t_1; \mathbf{w}_a) + J^{\pi(\mathbf{w}_a)}(\mathbf{x}; \mathbf{w}_c) - J^{\pi(\mathbf{w}_a)}(\mathbf{x}_0; \mathbf{w}_c) \overset{!}{=} 0 \quad (52)$$

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 simulation but due to space limitations it is impossible to describe the full algorithm. The full algorithm will be published in an extended paper later.

III. Experiment: Linear system with quadratic cost-to-go function (LQR)

The LQR system equations and cost-density are defined by (53) to (54).

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \mathbf{A} = \begin{bmatrix}-1 & -2\\ 1 & -4\end{bmatrix}, \mathbf{B} = \begin{bmatrix}0.5 & -1\\ 0.5 & 2\end{bmatrix} \quad (53)$$

$$\phi(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{x}^T\mathbf{Q}\,\mathbf{x} + \dot{\mathbf{x}}^T\mathbf{R}\,\dot{\mathbf{x}}, \quad \mathbf{Q} = \mathbf{R} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \quad (54)$$

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 (55).

$$J(\mathbf{x}, \dot{\mathbf{x}}) = \int_{t_0}^{\infty}\phi(\mathbf{x}, \dot{\mathbf{x}})dt \quad (55)$$

A. Optimal LQR-control

To solve the system above with minimal performance index, an Algebraic Riccati Equation (ARE) has to be solved. Details can be found in an advanced book on modern control theory, e.g. the excellent book by Brogan [9], chapter 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 (56), 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, $\tilde{\mathbf{R}} > \mathbf{0}$, $\tilde{\mathbf{Q}} - \tilde{\mathbf{N}}\tilde{\mathbf{R}}^{-1}\tilde{\mathbf{N}}^T \geq \mathbf{0}$ 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.

$$\tilde{J}(\mathbf{x}, \mathbf{u}) = \int_{t_0}^{\infty}\tilde{\phi}(\mathbf{x}, \mathbf{u})dt = \int_{t_0}^{\infty}(\mathbf{x}^T\tilde{\mathbf{Q}}\,\mathbf{x} + \mathbf{u}^T\tilde{\mathbf{R}}\,\mathbf{u} + 2\mathbf{x}^T\tilde{\mathbf{N}}\mathbf{u})dt \quad (56)$$

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 (57) and (58).

$$\mathbf{K} = \tilde{\mathbf{R}}^{-1}\left(\mathbf{B}^T\mathbf{S} + \tilde{\mathbf{N}}^T\right), \quad (57)$$

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

$$\mathbf{0} = \mathbf{A}^T\mathbf{S} + \mathbf{S}\mathbf{A} - (\mathbf{S}\mathbf{B}^T + \tilde{\mathbf{N}})\tilde{\mathbf{R}}^{-1}(\mathbf{B}^T\mathbf{S} + \tilde{\mathbf{N}}^T) + \tilde{\mathbf{Q}} \quad (58)$$

B. Numerical Example

1) rank$(\mathbf{K}) = \dim(\mathbf{x})$: Using the following system values (59) to (61), the optimal feedback is given by (63).

$$A = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix}, \quad \mathrm{eig}(A) = \{-2, -3\} \tag{59}$$

$$B = \begin{bmatrix} 0.5 & -1 \\ 0.5 & 2 \end{bmatrix} \tag{60}$$

$$Q = R = C = D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{61}$$

$$S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{solution to ARE (58).} \tag{62}$$

$$K^* = \begin{bmatrix} 0.6667 & -4.6667 \\ 0.3333 & -0.3333 \end{bmatrix}, \text{optimal feedback by (57).} \tag{63}$$

In [3] there are also other feedback methods for LQR systems investigated for comparison to determine long-term costs and controls. One of them is derived from the Calculus of Variations (CoV), which is a theoretical equivalent method to dynamic programming. If the matrix B is full rank, all the (stable) methods investigated, achieve the same optimal result for the feedback matrix K^*.

2) 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 [3], except the adaptive critic design and of course the solution calculated via (57) and (58). The adaptation based on CoV violates the independence conditions of the fundamental lemma of the calculus of variations. 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 then the approach via ACDs. The optimal reduced rank feedback is given by (68), based on the system matrices (64) to (66).

$$A = \begin{bmatrix} -1 & -2 \\ 1 & -4 \end{bmatrix}, \mathrm{eig}(A) = \{-2, -3\} \tag{64}$$

$$B = \begin{bmatrix} 1 \\ -3 \end{bmatrix} \tag{65}$$

$$Q = R = C = D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{66}$$

$$S = \begin{bmatrix} 1.0207 & -0.1865 \\ -0.1865 & 2.67881 \end{bmatrix}, \text{solution to ARE (58)} \tag{67}$$

$$K^* = \begin{bmatrix} -0.2420 & 0.1777 \end{bmatrix}, \text{optimal feedback by (57).} \tag{68}$$

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 thus speed up the traditional adaptive critic training quite considerably.

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) rank$(\mathbf{K}) = \dim(\mathbf{x})$: Figure 2 shows the actor or feedback parameters 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 $d\mathbf{K} := \frac{||\mathbf{K}||_\infty}{||d\mathbf{K}||_\infty} d\mathbf{K}$. It may occasionally happen 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 seldom cases, another initialization is the fastest way to solve the problem. Figure 3 shows adaptation for the critic parameters $\mathbf{w}_c$. Remarkably, the linearized critic update, outlined, works very well and speeds up convergence, especially when actor changes are of significant magnitudes.

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

IV. Conclusion

There are some valid points in using continuous ACDs. Plants are often described by differential equations like (1) and those equations can be used directly, without discretization. This is done implicit by the integration routine which is a second advantage, as an automatic step-size integrator does an adaptive sampling for free. In contrast to RTRL a "direct state vector" $\mathbf{x}(t)$ can be used and no additional states, e.g. from a neural network have to be introduced which increases the computational load extensively due to the $O(N^3)$ and $O(N^4)$ computational complexity in space and time, respectively, of the general RTRL algorithm [4], where $N = \dim(\mathbf{x})$ is the dimensionality of the state vector.

Another advantage is the improvement of a correction algorithm for concurrent actor-critic training, which was only outlined here and will be published in an extended version later together with all the necessary adaptation equations for the implementation of arbitrary system and critic approximators. 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 it was done very successfully by Ferrari in [10]. However, the equations used allow the development for any system and any cost-function, not only quadratic ones, as long as they are sufficiently differentiable.

Acknowledgment

The first author would like to thank Danil V. Prokhorov and Paul J. Werbos for their valuable feedback as Ph.D. thesis reviewers and acknowledges the useful suggestions made therein which also had an influence on this paper, as

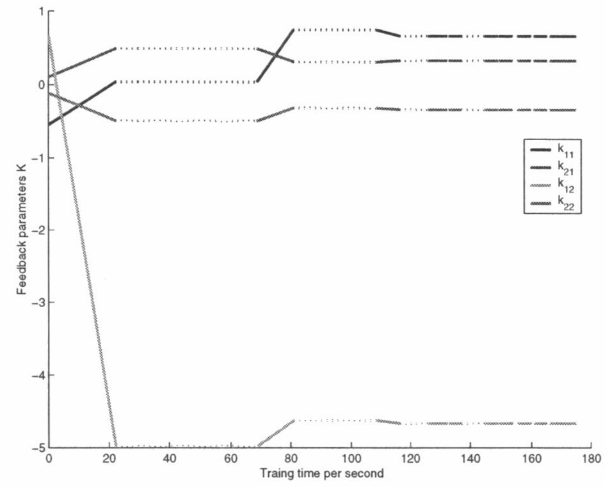

Fig. 2. *Trajectory of the parameters* $\mathbf{K}$ *for the system given in section III-B.1. The solid lines represent the time actor training via Newton's method. During the time indicated by the dashed lines, actor parameters are frozen and critic weights are adapted. After four actor-critic cycles the parameters are learned within an error better than* 10^{-5}.

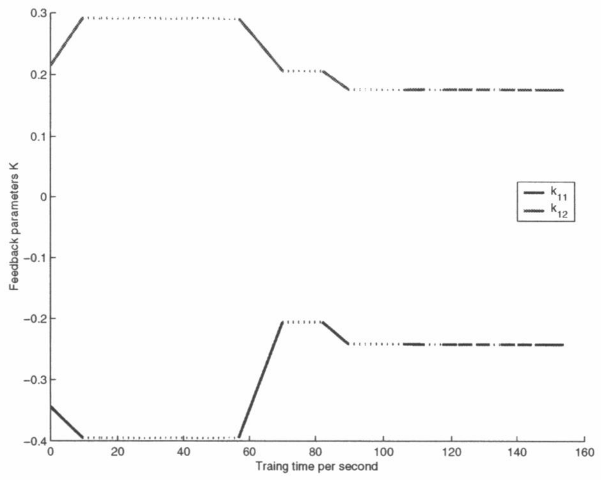

Fig. 4. *Trajectory of the parameters* $\mathbf{K}$ *for the system given in section III-B.2. The solid lines represent the time actor training via Newton's method. During the time indicated by the dashed lines, actor parameters are frozen and critic weights are adapted. After four actor-critic cycles the parameters are learned within an error better than* 10^{-5}.

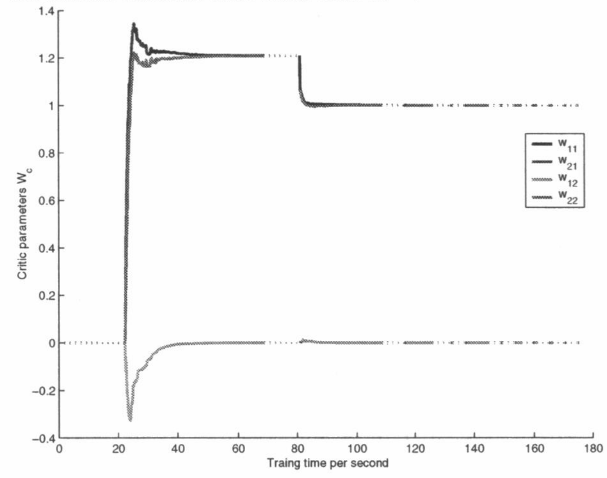

Fig. 3. *Trajectory of critic parameters* $\mathbf{W}_c$. *The solid lines represent the time critic training is performed. After the first actor-critic cycle the actor-critic consistency is achieved and the proposed linear critic updates due to actor changes can be applied. This is shown by the black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

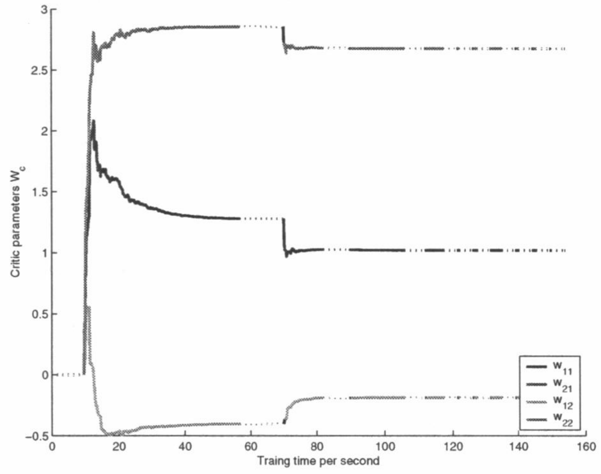

Fig. 5. *Trajectory of critic parameters* $\mathbf{W}_c$ *(note:* $w_{12} = w_{21}$. *The solid lines represent the time critic training is performed. After the first actor-critic cycle the actor-critic consistency is achieved and the proposed linear critic updates due to actor changes can be applied. This is shown by the black lines which represent a jump towards the optimal values, given by (67), especially for the non-zero* w_{11}, w_{22} *at the second actor-critic cycle.*

it is basically an excerpt from the first author's thesis [3]. This research was partly supported by the Australian Research Council (ARC).

REFERENCES

[1] J. S. Dalton. *A Critic based system for neural guidance and control.* Ph.d. dissertation in ee, University of Missoury Rolla, 1994.

[2] P. Werbos. Stable adaptive control using new critic designs. *http://xxx.lanl.gov/abs/adap-org/9810001*, March 1998.

[3] Thomas Hanselmann. *Approximate Dynamic Programming with Adaptive Critics and The Algebraic Perceptron as a Fast Neural Network related to Support Vector Machines.* Ph.d. dissertation, The Unviversity of Western Australia, Perth, Australia, 2003. Available by request from the author (thomash@ee.uwa.edu.au).

[4] S. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Chauvin and Rumelhart, editors, *Backpropagation: Theory, Architectures and Applications*, pages 433–486. LEA, 1995.

[5] S. Haykin. *Neural Networks a Comprehensive Foundation*, chapter 15. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1998.

[6] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1989.

[7] T. Hanselmann, A. Zaknich, and Y. Attikiouzel. Connection between *bptt* and *rtrl*. *3rd IMACS/IEEE International Multiconference on Circuits, Systems, 4-8 July 1999 Athens*, July 1999. Reprinted in Computational Intelligence and Applications, Ed. Mastorakis, Nikos, E., World Scientific, ISBN 960-8052-05-X,1999.

[8] P. Werbos. *Beyond regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph.d. dissertation, Harvard Univ., Cambridge, MA, 1974. Reprinted in The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting.

[9] William L. Brogan. *Modern Control Theory.* Prentice Hall, Upper Saddle River, New Jersey 07458, 3rd edition, 1991.

[10] S. Ferrari and R. Stengel. On-line adaptive critic flight control. *Journal of Guidance, Control and Dynamics*, 27(5):777–786, Sept.-Oct. 2004.