Enhancing the Episodic Natural Actor-Critic Algorithm by a Regularisation Term to Stabilize Learning of Control Structures

Andreas Witsch, Roland Reichle, Kurt Geihs
Distributed Systems Group
Universität Kassel, Germany
{witsch, reichle, geihs}@vs.uni-kassel.de

Sascha Lange, Martin Riedmiller
Machine Learning Lab
Albert-Ludwigs-Universität Freiburg, Germany
{slange, riedmiller}@informatik.uni-freiburg.de

***Abstract*—Incomplete or imprecise models of control systems make it difficult to find an appropriate structure and parameter set for a corresponding control policy. These problems are addressed by reinforcement learning algorithms like policy gradient methods. We describe how to stabilise the policy gradient descent by introducing a regularisation term to enhance the episodic natural actor-critic approach. This allows a more policy independent usage.**

We used the resulting algorithm to optimise a z-transformed rational function representing the control policy. This representation facilitates simultaneous optimisation of the control structure and its parameters in time space and can be analysed in terms of control theory to predict the control behaviour for arbitrary scenarios.

Furthermore we present a solution to the general problem of finding a initial parameter set with the help of a single demonstrated trajectory. The approach is evaluated on a cartpole simulation for demonstrating the expressiveness of the policy. Furthermore, a real soccer robot scenario demonstrates the ability of the proposed approach to deal with real world scenarios.

I. Introduction

Most existing approaches for learning controllers without process knowledge make use of neural networks, in order to approximate the transfer function. Hence, they have the drawback of being black boxes and the behaviour is unknown for new environments.

Classical control structures like a PID controller have the advantage that their behaviour is well known but a system model is needed in order to find optimal parameters. Especially higher control structures are difficult to set up for systems where only a rough approximation can be found. Thus, an exact model is needed for an optimal design. This leads to the problem of finding the optimal control structure and gathering optimal parameters for this structure.

This paper presents a new approach that overcomes these problems by learning the parameters for a z-transformed rational function, which is used as a control policy. This general representation allows learning the control structure and optimising its parameters at the same time. The change of parameters in z-space can influence the control structure as well as its parameters in the time space. Furthermore, z-transformed functions are very well known in control theory, which offers diverse and well-established methods for analysing.

The z-space transfer function is represented by time shifting states, weighted by its parameters, and leading to a state machine. In order to simplify the parameter determination, the initial computation is inspired by the training of echo state networks [1], which train their weights by *teacher forcing*: The past controller outputs can be seen as inputs for the time shifting states of the z-space policy, which leads to a system of equations, solvable by linear regression, and determined by a sample trajectory. We propose this new imitation approach to find initial parameters for a policy in z-space without having a model of the system.

This allows to apply policy gradient methods to improve the policy. In this context we show that the most valuable policy gradient algorithm – the episodic natural actor-critic – can benefit by a regularisation term. This term allows using the algorithm more independently of the target policy.

The approach is evaluated through a standard use case – a cartpole simulation – to show the expressiveness of this policies and a RoboCup soccer robot ball interception scenario to show that this approach is robust against noisy sensor data. The evaluation will show that this improvement step is able to find a local optimum for policy parameters with respect to a given reward function.

The remainder of this paper is organised as follows. In Section II we present the episodic natural actor-critic learning algorithm, Section III shows the z-policy representation and how to apply policy gradient algorithms. Section IV explains how we improved the actor-critic algorithm by a regularisation term, the results of the two experiments is presented in Section V followed by the final conclusions in Section VI.

II. Learning Algorithm

This section introduces the episodic natural actor-critic algorithm (eNAC), which is a policy gradient approach, and belongs to the class of reinforcement learning algorithms. We make use of this algorithm in order to optimise the parameters of a control policy. Before introducing policy gradient methods, we first define some general assumptions. Afterwards we

describe the basics of policy gradient algorithms leading to the eNAC algorithm. Finally the policy gradient for a normally distributed policy is determined.

A. Reinforcement Learning Framework

We assume the state probability distribution shown in Equation 1. It characterises the probability of reaching a new state under the assumption of a specific action a_k in a state x_k:

$$x_{k+1} \sim p(x_{k+1}|x_k, a_k) \tag{1}$$

For policy gradient methods all actions of the agent are generated by a probabilistic policy to facilitate exploration:

$$a_k \sim \pi_\theta(a_k, x_k) \tag{2}$$

The index θ denotes the parameters of the policy. Furthermore, the value function and state-action function are given by the Bellman equations [2]:

$$V^\pi(x) = r(x) + \alpha \sum_{x'} p(x'|x, \pi(x)) V^\pi(x') \tag{3}$$

$$Q^\pi(x, a) = \alpha \sum_{x'} p(x'|x, a) V(x') \tag{4}$$

where α denotes the discount factor for problems with unlimited horizon. The goal of policy gradient methods is to optimise the expectation of the cumulated reward function

$$J(\theta) = \frac{1}{\alpha_\Sigma} E\left\{\sum_{k=0}^{H} \alpha_k r_k\right\} \tag{5}$$

The term α_Σ denotes a normalisation factor to ensure that $\sum_{k=0}^{H} \frac{\alpha_k}{\alpha_\Sigma} = 1$. Note that we asume a value of $\alpha_k = 1$ for all further equations and experiments of this paper. An alternative representation of Equation 5 computes this expectation as the integral over the state distribution and the integral over all actions of the policy by

$$J(\theta) = \int_X d^\pi(x) \int_A \pi_\theta(a, x) r(a, x) dx da \tag{6}$$

The next sections use this equation to derive the policy gradient for the parameters θ.

B. REINFORCE Policy Gradient

In order to optimise the parameters of a policy, we have the goal to maximise the cumulated reward J. The reward expectation over all trajectories τ of Equation 6 depends on the parameters θ of the chosen policy and can be rewritten as

$$J(\theta) = \int_T p_\theta(\tau) r(\tau) d\tau \tag{7}$$

where $r(\tau)$ denotes the cumulated rewards of a trajectory and $p_\theta(\tau)$ the probability distribution of a specific trajectory. By deriving $J(\theta)$ with respect to the policy parameters, we get the policy gradient as shown in [3]:

$$\nabla_\theta J(\theta) = \int_T \nabla_\theta p_\theta(\tau) r(\tau) d\tau \tag{8}$$

$$= \int_T \frac{p_\theta(\tau)}{p_\theta(\tau)} \nabla_\theta p_\theta(\tau) r(\tau) d\tau \tag{9}$$

$$= \int_T p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau) d\tau \tag{10}$$

$$= E\{\nabla_\theta \log p_\theta(\tau) r(\tau)\} \tag{11}$$

While the reward function is not influenced by the policy parameters, it is sufficient to compute only the gradient of the trajectory distribution function

$$p_\theta(\tau) = p_\theta(x_0) \prod_{k=0}^{H} p(x_{k+1}|x_k, a_k) \pi_\theta(a_k, x_k) \tag{12}$$

where x_0 represents the fixed start state of all trajectories. Computing the logarithm of Equation 12 results in

$$\log p_\theta(\tau) = \log p_\theta(x_0) + \sum_{k=0}^{H} \log p(x_{k+1}|x_k, a_k) \pi_\theta(a_k, x_k) \tag{13}$$

$$\nabla_\theta \log p_\theta(\tau) = \sum_{k=0}^{H} \phi_k \tag{14}$$

where $\phi_k = \nabla_\theta \log \pi_\theta(x_k, a_k)$. Bringing everything together results in the gradient estimation g:

$$g = \nabla_\theta J(\theta) = E\{\nabla_\theta \log \pi_\theta(\tau) r(\tau)\} \tag{15}$$

$$\approx E\left\{\left(\sum_{k=0}^{H} \phi_k\right)\left(\sum_{l=0}^{H} r(a_l, x_l)\right)\right\} \tag{16}$$

The update rule for the parameter set is given by $\theta_{n+1} = \theta_n + \alpha g$, where α denotes the learning rate.

C. Variance Reduction by Introducing a Baseline

Usually the REINFORCE policy gradient has a high variance, which requires many trials in order to find a suitable estimation. This drawback can be attenuated by introducing a baseline, which does not bias the gradient estimation, because the integral over the differentiated trajectory distribution is always zero:

$$\int_T p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau = \int_T \nabla_\theta p_\theta(\tau) d\tau = \nabla_\theta 1 = 0 \tag{17}$$

$$\Rightarrow \int_T p_\theta(\tau) \left(\sum_{k=0}^{H} \phi_k\right) b d\tau = 0 \tag{18}$$

Thus, a baseline b can be used to minimise variance of the gradient estimation:

$$\nabla_\theta J(\theta) = E\{\sum_{k=0}^{H} \phi (r(a, x) - b)\} \tag{19}$$

$$\Rightarrow g = \int_T p_\theta(\tau) \left(\sum_{k=0}^{H} \phi_k\right)\left(\sum_{l=0}^{H} r_l - b\right) d\tau \tag{20}$$

Due to Equation 18, the baseline may be chosen arbitrary. With the goal is to minimise the variance of the gradient estimation, the optimal baseline for a gradient element i can be calculated by adopting the variance definition:

$$\sigma_i^2 = E\{g_i^2\} - \left(\nabla_{\theta_i} J\right)^2 \tag{21}$$

$$\Rightarrow \min_{b_i} \sigma_i^2 \geq E\{\min_{b_i} g_i^2\} - \left(\nabla_{\theta_i} J\right)^2 \tag{22}$$

with a minimised variance by solving g_i^2 for the minimum:

$$g_i^2 = \left(\int_T p_\theta(\tau) \sum_{k=0}^{H} \phi_k \left(\sum_{l=0}^{H} r_l - b_i \right) d\tau \right)^2 \tag{23}$$

$$\Rightarrow b_i = \frac{\int_T p_\theta(\tau) \left(\sum_{k=0}^{H} \phi_k\right)^2 \left(\sum_{l=0}^{H} r_l\right) d\tau}{\int_T p_\theta(\tau) \left(\sum_{k=0}^{H} \phi_k\right)^2 d\tau} \tag{24}$$

This estimation of the baseline may still be improved by making use of the knowledge of the performed exploration, which leads finaly to the eNAC algorithm.

D. Episodic Natural Actor-Critic

As shown by Sutton et al. [4], a compatible function approximation exists for $Q^\pi(x,a) - b^\pi(x)$:

$$Q^\pi(x,a) - b^\pi(x) \equiv f_w^\pi(x,a) = \phi^T w \tag{25}$$

This function approximation can be seen as a critic representing an advantage function of an action a in state x. Note that the compatible function approximation does not equal $Q^\pi(x,a) - b^\pi(x)$, but in terms of Equation 20 both expressions are compatible. In order to get an estimation, we insert the compatible function approximation of Equation 25 into the Bellman Equation. It estimates the parameters of the critic

$$Q^\pi(a,x) = \phi^T w + V^\pi(x) \tag{26}$$

$$= r(x,a) + \alpha \int_X p(x'|x,a) V^\pi(x') dx' \tag{27}$$

The class of actor-critic algorithms builds on this equation. Actor-critic algorithms compute the gradient with a function approximation for the value function $V^\pi(x)$ through weighted basis functions $V^\pi(x) = \phi(x)^T v$. A detailed description of this class of algorithms can be found in Peters et al. [5].

For a good estimation of $V^\pi(x)$, basis functions have to be known, where a suitable linear combination with v can be found. In most cases, we do not know, which basis functions qualify for a good approximation. In the episodic case, however, we can avoid this by summing up Equation 27 over the whole episode:

$$\sum_{k=0}^{H} \alpha_k f_w^\pi(x_k, a_k) = \alpha_{H+1} V^\pi(x_{H+1}) + \sum_{k=0}^{H} \alpha_k r(x_k, a_k) - V^\pi(x_0) \tag{28}$$

The term $a_{H+1} V^\pi(x_{H+1})$ disappears for discounted learning as well as for the episodic case, where $r(x_H, a_H)$ is the last reward. Thus, every roll-out results in a equation with the parameter vector w. This leads to a linear regression problem with the start state $J_0 = V^\pi(x_0)$:

$$\sum_{k=0}^{H} \alpha_k {\phi_k}^T w + J_0 = \sum_{k=0}^{H} \alpha_k r(x_k, a_k) \tag{29}$$

$$\Rightarrow \begin{bmatrix} w \\ J_0 \end{bmatrix} = (\Psi^T \Psi)^{-1} \Psi^T R \tag{30}$$

with

$$\Psi_i = \left[\textstyle\sum_{k=0}^{H} \alpha_k \phi_k^T, 1 \right] \tag{31}$$

$$R = \sum_{k=0}^{H} \alpha_k r(x_k, a_k) \tag{32}$$

Note that the matrix $(\Psi^T \Psi)$ is only invertible if least $\dim w + 1$ linearly independent vectors exist.

1) Relation between Compatible Function Approximation and Gradient: We derived an estimation of the parameter vector w for a compatible function approximation in Equation 30. To take advantage of this estimation, we apply it to the policy gradient:

$$\begin{aligned} \nabla_\theta J(\theta) &= w \int_X d^\pi(x) \int_A \nabla_\theta \pi_\theta(a,x) \phi^T da dx \\ &= w \int_X d^\pi(x) \int_A \pi_\theta(a,x) \phi \phi^T da dx \\ &= G_\theta w \end{aligned} \tag{33}$$

In this case, it is still difficult to estimate G_θ. One can show that $G_\theta = F$, where F is the *Fisher information matrix* [6]. It is defined as

$$F = E\left\{ \nabla_\theta \log \prod_{\ell=0}^{n} f_\theta(X_\ell) \nabla_\theta \log \prod_{\ell=0}^{n} f_\theta(X_\ell) \right\} \tag{34}$$

where f_θ is the probability density function for the parameter θ and X_ℓ is a random variable. Given policy gradients, there is only one random variable ($n = 1$):

$$F = E\left\{ \nabla_\theta \log \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) \right\} \tag{35}$$

$$= \int_X d^\pi(x) \int_A \pi_\theta(a,x) \phi \phi^T da dx \tag{36}$$

$$= G_\theta \tag{37}$$

Neural network training algorithms adopt the natural gradient as a metric:

$$g_{NG} = F^{-1} g_{\text{euclid}} \tag{38}$$

Inserting Equation 33 shows that the natural gradient corresponds to our estimation of the vector w of the compatible function approximation

$$g_{NG} = F^{-1} G_\theta w = w \tag{39}$$

This means that an estimation of w suffices and there is no need to estimate G_θ. Usual gradient descent methods use the assumption of an Euclidian metric in the parameter space without any a-priory reason. The Fisher information

Listing 1. episodic natural actor-critic with constant baseline

repeat

perform N trials, obtain: $\{x_{0:H}\ ,\ u_{0:H}\ ,\ r_{0:H}\}$

Policy derivatives $\phi_n = \nabla_\theta \log \pi(a_n, x_n)$

Fisher information $F = \sum_{n=0}^{N}((\sum_{j=0}^{H} \phi_j)(\sum_{i=0}^{H} \phi_i))$

Cumulated reward $r = \sum_{n=0}^{N}(\sum_{l=0}^{H} r_l)$

Eligibility $\Phi = \sum_{n=0}^{N}(\sum_{j=0}^{H} \phi_j)$

Vanilla gradient $g = \sum_{n=0}^{N}((\sum_{j=0}^{H} \phi_j)(\sum_{l=0}^{H} r_l))$

Baseline

$$b = m^{-1}(1 + \Phi(mF - \Phi\Phi^T)^{-1})(r - \Phi^T F^{-1} g)$$

Natural gradient $g_{\text{eNAC1}} = F^{-1}(g - \Phi b)$

until g_{eNAC1} converged

Listing 2. episodic natural actor-critic with time-variant baseline

repeat

perform N trials, obtain: $\{x_{0:H}\ ,\ u_{0:H}\ ,\ r_{0:H}\}$

Policy derivatives $\phi_n^m = \sum_{t=0}^{n} \nabla_\theta \log \pi(a_t^m, x_t^m)$

Fisher information $F = \sum_{j=0}^{m}\sum_{i=0}^{n}(\phi_i^j \phi_i^{j^T})$

Reward vector $\bar{r} = \sum_{j=0}^{m}\sum_{i=0}^{n} r_i^j e_i$

Eligibility $\Phi = \sum_{j=0}^{m}[\phi_0^j, \phi_1^j, ..., \phi_m^j]$

Vanilla gradient $g = \sum_{j=0}^{m}\sum_{i=0}^{n} \phi_i^j r_i^j$

Baseline

$$b = m^{-1}(I_n + \Phi^T(mF - \Phi\Phi^T)^{-1}\Phi)(\bar{r}^T - \Phi^T F^{-1} g)$$

Natural gradient $g_{\text{eNACn}} = F^{-1}(g - \Phi b)$

until g_{eNACn} converged

metric discloses information about the quality of the gradient estimation.

To overcome matrix inversion problems during the function approximation, we take advantage of the *matrix inversion lemma* [7], which gives rise to the algorithm shown in Listing 1. For a detailed derivation see Peters [8].

E. Episodic Natural Actor-Critic with Time Variant Baseline

The gradient estimation can be significantly improved in some cases by computing the gradient estimation and the corresponding baselines time variant. The time variant baselines can be interpreted as additional basis functions. Hence, the regression matrix of these functions has the following form:

$$X^T = \begin{bmatrix} \phi_0^0, \phi_1^0, ..., \phi_n^0, \phi_0^1, ..., \phi_n^m \\ e_0, e_1, ..., e_n, e_0, ..., e_n \end{bmatrix}, \tag{40}$$

where $\phi_n^m = \sum_{k=0}^{n} \nabla_\theta \log \pi(a_k^m, x_k^m)$ for a trial m and e_i represents the i-th unit vector of length n. The target matrix has the form:

$$Y^T = \left[r_0^0, r_1^0, ..., r_n^0, r_0^1, ..., r_n^m\right], \tag{41}$$

where r_k^m denotes the reward for trial m in time step k. The regression problem results in the form:

$$\begin{bmatrix} w \\ b \end{bmatrix} = (X^T X)^{-1} X^T Y \tag{42}$$

$$\Rightarrow \begin{bmatrix} g_{\text{eNACn}} \\ \bar{r} \end{bmatrix} = \begin{bmatrix} F & \Phi \\ \Phi^T & mI_H \end{bmatrix}^{-1} \begin{bmatrix} g \\ \bar{r} \end{bmatrix} \tag{43}$$

Applying the inversion lemma and leads to the algorithm shown in Listing 2.

F. Solution for the Policy Gradient

We now derive a general formula for calculating the policy gradient. It assumes a normally distributed exploration. A normally distributed policy in the n-dimensional space is defined with a covariance matrix of

$$\pi_\theta(\vec{a}, \vec{x}) = c \exp\left(-\frac{1}{2}(\vec{a} - \mu_\theta(\vec{x}))^\top \Sigma^{-1}(\vec{a} - \mu_\theta(\vec{x}))\right) \tag{44}$$

where $c = ((2\pi)^{n/2}\left|\Sigma\right|^{1/2})^{-1}$ is the normalisation term and μ is a transfer function. The action vector $\vec{a}$ is defined as $\vec{a} = \mu_\theta(\vec{x}) + \vec{\epsilon}$. To compute the policy gradient it is necessary to compute the logarithm of π:

$$\log \pi_\theta(\vec{a}, \vec{x}) = \log c - \frac{1}{2}(\vec{a} - \mu_\theta(\vec{x}))^\top \Sigma^{-1}(\vec{a} - \mu_\theta(\vec{x})) \tag{45}$$

In most cases, Σ is a diagonal matrix with the diagonal vector $(\sigma_1^2, \ldots, \sigma_n^2)$. The policy gradient is defined by

$$\nabla \log \pi_\theta(\vec{a}, \vec{x}) = -\nabla \frac{1}{2}(\vec{a} - \mu_\theta(\vec{x}))^\top \Sigma^{-1}(\vec{a} - \mu_\theta(\vec{x})) \tag{46}$$

$$= \Sigma^{-1}(\vec{a} - \mu_\theta(\vec{x}))\nabla\mu_\theta(\vec{x}) \tag{47}$$

By substituting $(\vec{a} - \mu_\theta(\vec{x}))$ with ϵ we get

$$\nabla \log \pi_\theta(\vec{a}, \vec{x}) = (\Sigma^{-1}\epsilon)\nabla\mu_\theta(\vec{x}) \tag{48}$$

This general formula calculates the policy gradient of a given policy under for a normally distributed exploration. It shows that the policy gradient is computed of derivated transfer function normalised by variance and the exploration.

III. Uniform Representation for Linear Controllers

Related to policy gradient methods we are interested in policies that achieve four main characteristics: First, the policy should be highly expressive so that it is able to approximate the function in an optimal way. Second, a policy should have only few parameters as the number of necessary learning trials grows with the number of parameters. Third, the error landscape should be suited well for gradient descent methods. We finally need a policy that is differentiable – with respect to its parameters – in order to calculate the policy gradient.

To overcome all these problems, we propose to model the transfer function in the z-transformed space as a rational function. The transfer function is represented as a state machine. For determining the initial policy parameters required for policy gradient methods, we introduce an approach, which is able to derive an initial parameter set by imitating a single sample trajectory.

A. Z-Transformed Controller

The z-transformation is defined as

$$X(z) = Z\{x(t)\} = \sum_{t=-\infty}^{\infty} x(t) z^{-t}, \tag{49}$$

where x is a transfer function, t a discrete time step, and z a general complex number. It is possible to represent, for example, differentiations, integrations, or oscillations. Note that the controller abilities depend on the sampling time of the controller. We define linear discrete control policies in z-space with n parameters using the following form:

$$\mu(z) = \frac{\theta_1 z^{1-m} + \theta_2 z^{2-m} + \cdots + \theta_m}{-\theta_{m+1} z^{1-m} - \theta_{m+2} z^{2-m} - \cdots + 1} \tag{50}$$

$$= \frac{\sum_{i=1}^{m} \theta_i z^{i-m}}{1 + \sum_{i=1}^{m-1} -\theta_{m+i} z^{i-m}} \tag{51}$$

This can be considered as IIR (**I**nfinite **I**mpulse **R**esponse) filter, which can be represented as a state machine. We refere to it as z-policy. The variable m is said to be the *degree* or *order* of the z-policy. It indicates how many past steps of the policy are involved in the computation of the current output value and it corresponds to the degree of the nominator polynom. A corresponding canonical structure – the "Transpose-Form IIR Filter Structure" [9] to the rational function of Equation 51 – is shown in Figure 1. The z^{-1} elements delay the input signal by one timestep and form "memory".

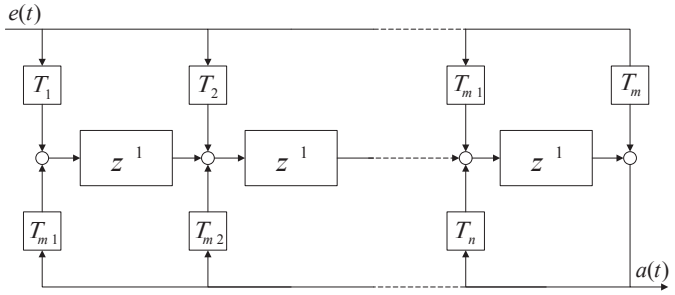

Fig. 1. Canonical direct structure of a rational z-space function

In order to apply policy gradient methods we need the derivative of the controller with respect to its parameters $\vec{\theta}$, which can be directly computed in its z-transformed form.

Proof by definition:

$$\sum_{t=-\infty}^{\infty} \frac{\delta x(t)}{\delta \theta_i} z^{-t} = \frac{\delta \sum_{t=-\infty}^{\infty} x(t) z^{-t}}{\delta \theta_i} = \frac{\delta X(z)}{\delta \theta_i} \tag{52}$$

■

Hence, we are able to introduce a state machine for every derivative of θ_j, given a rational function.

B. Initial Parameters by Imitation Learning

In order to complete the policy, we need values for $\vec{\theta}$, which can be optimised by policy gradient algorithms. These can get stuck in local minima. For this reason, the initial parameter values are important for finding a good final policy. But analytically initial parameters can only be found with a system model, or for simple policies like proportional controllers.

Because of the complex error landscape in parameter space, which is especially caused of imaginary poles it is necessary to find a parameter initialisation near to the desired trajectory. This problem can be solved by initialising the parameters through demonstrating a trajectory. The generated Input/output pairs are used for computing the parameters through linear regression. The idea is close to the training of Echo State Networks with teacher forcing [1]:

$$a(z) = \frac{\theta_1 z^{1-m} + \theta_2 z^{2-m} + \cdots + \theta_m}{-\theta_{m+1} z^{1-m} - \theta_{m+2} z^{2-m} - \cdots + 1} e(z) \tag{53}$$

$$= e(z)\theta_1 z^{1-m} + e(z)\theta_2 z^{2-m} + \cdots + e(z)\theta_m + \tag{54}$$

$$a(z)\theta_{m+1} z^{1-m} + a(z)\theta_{m+2} z^{2-m} + \cdots + a(z)\theta_n z^{-1}$$

Transformed in time space this leads to

$$\begin{aligned} a(t) = & e(t+1-m)\theta_1 + e(t+2-m)\theta_2 + \\ & \cdots + e(t)\theta_m + \\ & a(t+1-m)\theta_{m+1} + a(t+2-m)\theta_{m+2} + \\ & \cdots + a(t-1)\theta_n \end{aligned} \tag{55}$$

So each time step t of the demonstrated trajectory leads to an Equation 55. By applying the linear regression we get:

$$\vec{\theta} = (X^T X)^{-1} X^T Y \tag{56}$$

with

$$X_i = \big[e(i+1-m),\ \ldots,\ e(i),\ a(i+1-m),\ \ldots, a(i-1)\big] \tag{57}$$

for $0 \leq i \leq H$ and

$$Y = \big[a(0),\ a(1),\ \ldots,\ a(H)\big] \tag{58}$$

where H is the number of input/output pairs of the demonstrated trajectory.

IV. Regularisation Term

All natural policy gradient algorithms introduced so far share the same drawback: a differentiable policy cannot be chosen arbitrarily. To ensure that the Fisher matrix is invertible it is not allowed to have more than one parameter with the same derivative. Otherwise, this results in linearly dependent vectors in the matrix, which is a direct result of Equation 34. If derivatives differ only slightly, the inversion will lead to numerical problems. Note, that this is the case if two parameters have the same or similar influence to the resulting action. The following example illustrates, why it is hard to avoid a parameter dependency.

A. Necessity of a Regularisation Term

Given a third-order z-policy

$$\mu(z) = \frac{\theta_1 z^{-2} + \theta_2 z^{-1} + \theta_3}{\theta_4 z^{-2} + \theta_5 z^{-1} + 1} \tag{59}$$

with an initial parameter vector $\vec{\theta} = [0, 0, 1, 0, 0]$, which is actually the representation of an proportional controller with a gain of 1. The derivatives for θ_4 and θ_5 are

$$\nabla_{\theta_4} \mu(z) = \frac{(\theta_1 z^{-2} + \theta_2 z^{-1} + \theta_3) z^{-2}}{(\theta_4 z^{-2} + \theta_5 z^{-1} + 1)^2} \tag{60}$$

and

$$\nabla_{\theta_5}\mu(z) = \frac{(\theta_1 z^{-2} + \theta_2 z^{-1} + \theta_3) z^{-1}}{(\theta_4 z^{-2} + \theta_5 z^{-1} + 1)^2} \tag{61}$$

Rewriting this policy using a Gaussian exploration and taking the parameter vector into account, we get

$$\nabla_{\theta_4} \log \pi(z) = \frac{z^{-2}}{1} \frac{\epsilon}{\sigma} \tag{62}$$

and

$$\nabla_{\theta_5} \log \pi(z) = \frac{z^{-1}}{1} \frac{\epsilon}{\sigma} \tag{63}$$

This means that for the policy derivatives of the full trajectory $\phi = \sum_{t=1}^{H} \nabla_\theta \log \pi(a_t, x_t)$ for the eNAC, the only difference between $\nabla_{\theta_4} \log \pi(z)$ and $\nabla_{\theta_5} \log \pi(z)$ is the last value of $\nabla_{\theta_5} \log \pi(z)$. This is a result of the exponent of the variable z, which leads to a time delay between both derivatives. For a more complex parameter vector $\vec{\theta}$ this relation might not be present, but as shown in this example linear dependencies are not always obvious. The resulting numerical inversion problems of the Fisher matrix can lead to a high variance in the gradient estimation or a random walk.

Despite this example the problem is not only limited to z-policies. Especially in complex policies it is hard to predict if two parameters have the same influence to the resulting action or not. In order to simplify the task of choosing a policy, an algorithmic solution is desired.

B. Solution

In this paper we introduce a term λI for the update rule to avoid these problems:

$$g_{NG} = (F + \lambda I)^{-1} (g - \Phi b) \tag{64}$$

The Fisher information matrix is turning the gradient with respect to the performed exploration. This means if λ is huge (relative to the matrix F) the matrix only has small influence on direction changes of the gradient. Thus, we scaled λ by the determinant of the matrix:

$$\lambda = \frac{\alpha}{\det(F) + 1} \tag{65}$$

The constant value α is set to 0.01 in all experiments; we determined the value empirically. It facilitates the inversion of F but still leads to a gradient close to the natural gradient.

A further interpretation of this term is well known from the least squares regression [10]. This term is the result of solving

$$E(g_{NG}) = \frac{1}{2} \sum_{k=1}^{H} (r_k - g_{NG}\phi_k)^2 + \frac{\lambda}{2} g_{NG}^T g_{NG} \tag{66}$$

for the minimum. Minimising of $\lambda g_{NG}^T g_{NG}$ favours solutions with a small gradient, while λ is a weighting factor compared to least squares. With this concept we provide a simple algorithmic enhancement for computing the natural actor-critic policy gradient for arbitrary policies.

V. Results

A. Cartpole Experiment with a z-Policy

In this experiment we focus on the question whether the eNAC algorithm is able to compute parameter updates for strongly coupled parameters and whether the z-policy is able to control a standard cartpole simulation [11] only with the position information of the cart and the pole. The task of the controller is to balance a pole on a cart, which is able to move in one dimension. Note, that an internal represention of the velocities it is necessary for solving the balancing problem from arbitrary initial states. A further challenge is to find a linearisation for the operating point, which actually is a pole angle of $0°$.

1) Imitation: The chosen structure of the controller is the sum of two z-policy controllers R_1 and R_2 with a degree of 10. This gives enough freedom to find a good approximation. Thus, we have a single input multiple output system as shown in Figure 2. The reference value vector $\vec{w} = \vec{0}$ and the system output values y_1 and y_2 are the cart position and pole angle.

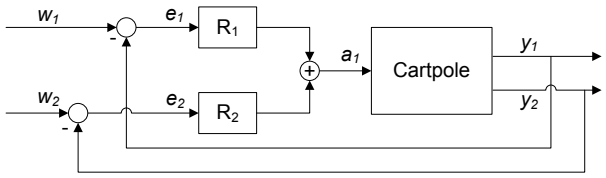

Fig. 2. Control structure of the cartpole system

In order to find initial parameters for the controller, we make use of the new *learning by demonstration* technique proposed in this paper. The goal is to find a controller that controls the pole angle, and a second controller for the cart position. The imitation step is extended by a decomposition step, that devides a sample trajectory in two new specific trajectories. The example trajectory is generated by a sample controller that can solve the problem in an imperfect way. See [12] for a detailed derivation of this part of the experiment.

2) Applying the Episodic Natural Actor-Critic: Both controllers have $n = 19$ parameters – 10 in the nominator polynomial and 9 in the denominator polynomial. In order to reduce the variance of the gradient estimation, a good coverage of the parameter space should be achieved. Therefore a parameter update is performed after $\frac{3}{2}n$ trials by the episodic natural actor-critic with constant baseline.

Furthermore, a reward function is needed. The goal of the controller is to reach a target cart position of $[-0.1\,\text{m}, 0.1\,\text{m}]$, while balancing the pole within an angle of $[-0.1\,\text{rad}, 0.1\,\text{rad}]$. Therefore, the reward function is specified as

$$r(\vec{x}, a) = \begin{cases} 0, \text{if the pole has target angle and position} \\ -1, \text{otherwise} \end{cases} \tag{67}$$

In order to allow the controller to reach these positions an ϵ-environment is specified around the target points. This reward function will lead to a controller that tries to achieve the goal as fast as possible.

The learning progress for this experiment is slow. Riedmiller et. al. [13] could show, that a similar problem can be learned with 5050 trials by determining a fixed policy structure in time space. As shown in Figure 3, the eNAC1 needs at least 200 parameter updates(5800 trails) to find the next local optimum. The figure averages the reward history of 20 learning experiments. Obviously, only a small learning rate can be used during a parameter update because the system is close to instability. This can be shown by manually changing parameters: Even very small parameter changes of about 10^{-3} can lead to an instable system. Parameter updates in eNAC1 lead to changes of up to 10^{-1}.

The error bars in Figure 3 represent standard deviation. The raising variance for a higher number of parameter updates indicates the existence of multiple local optima. Furthermore for one unique experiment a reward of -62 could be achieved. We were not able to reproduce these results in the 20 trials. Thus it is not considered in the statistics underlying this figure. Furthermore, the figure shows an example of the improvement without a sufficient regularisation term. One update step lead to a parameter set wich achieved a reward of -370 caused by a random walk. This shows the necessity of this term with a sufficient value for λ.

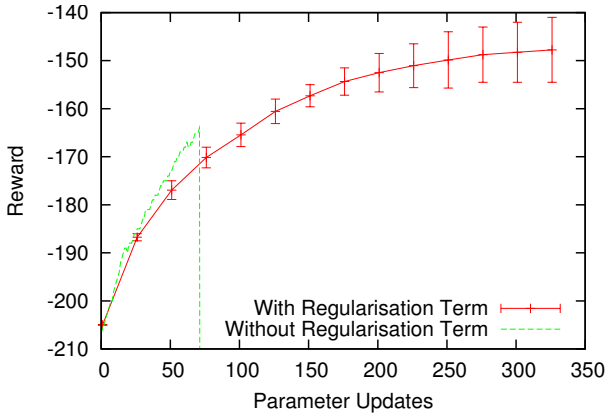

Fig. 3. Reward improvement with and without Regularisation Term

After the successful training it is still unknown whether the controller is able to generalise. In order to demonstrate this, the resulting controller was applied to 66 different initial states. We tested starting angles of $-0.1 \leq \alpha \leq 0.1$ rad together with starting positions between 1.0 and -1.0. The controller was able to solve the balancing problem for all initial states.

B. Learning and Optimising Ball Interception for a RoboCup Robot

We now propose a scenario that assigns the training result of z-policies to a real world system. The approach has been applied on middle size league RoboCup robot. The robot is driven by wheels arranged for omnidirectional movement [14] and has a height of 80 cm. The task of ball interception consists basicaly on the choice of the robot velocity and rotation. The main goal of this experiment is to demonstrate how the z-policy approach handles noisy ball and robot positions.

1) Imitation: The robot shall learn the translation allowing the interception of an unmoved ball. In order to solve this problem, the policy has to find a transfer function that maps a desired robot velocity to a specific ball distance. For the initial policy, the robot is moved by a human towards the ball from a distance of 2 m. The collected demonstration data are used to compute the initial parameters for a z-policy of degree 4.

Figure 4 depicts the demonstrated translation trajectory as well as the trajectory generated by the robot with respect to the ball distance. The discontinuous points in the plot indicate the noise of the estimation of the ball distance. It shows that a polynomial of degree of four is not sufficient for imitating the trajectory accurately. Furthermore, the error signal of the imitation differs from the training signal. However, the imitated trajectory still exhibits the same characteristics: Moving towards the ball with a high velocity and slowing down when getting close to it. It must be emphasised that the demonstrated trajectory is the result of only a *single trial*.

2) Optimisation: In the second part of the experiment, the eNAC1 algorithm is applied to the policy in order to optimise the robot's behaviour towards a local optimum. According to the degree of four of the z-policy, seven parameters are present. For this reason and in order to reduce the gradient variance, a parameter update is performed after 14 trials. Seven trials per parameter update are usually sufficient in this experiment but more trials stabilise the gradient estimation by reducing the variance. The reward function for rating the actions of a robot is given by

$$r(x,a) = \begin{cases} 0, \text{if robot has the ball} \\ -1, \text{otherwise} \end{cases} \tag{68}$$

In order to ensure that the robot is not too fast and thereby pushes the ball away, we compute the difference between their velocities as a final reward: $r_{\text{final}} = |v_{\text{robot}} - v_{\text{ball}}|$.

As shown in Figure 5, the reward could be increased from -198 to -135 within only five parameter updates. This results in a behaviour, which is able to intercept the ball about 2 seconds faster than the initial policy. The learning rate to achieve this goal is determined through computing three gradients to roughly estimate the magnitude and the effect of a parameter update.

Figure 4 shows the demonstrated trajectory (solid), the initial imitation of the z-policy (dashed) and the optimised behaviour (dotted). While improving the policy, the velocity is maximised while keeping the ball in the dribbling device.

The final robot policy for intercepting a ball performs better than previous manually implemented behaviours and was able to compete very well in a soccer match with the RoboCup world champion of 2009 in a soccer match during the RoboCup German Open in 2010.

VI. Conclusions

In this paper we address the two major challenges of *finding an optimal control structure* and *optimal parameters for the specific structure*. While other learning approaches use black box function approximations, the presented approach is able

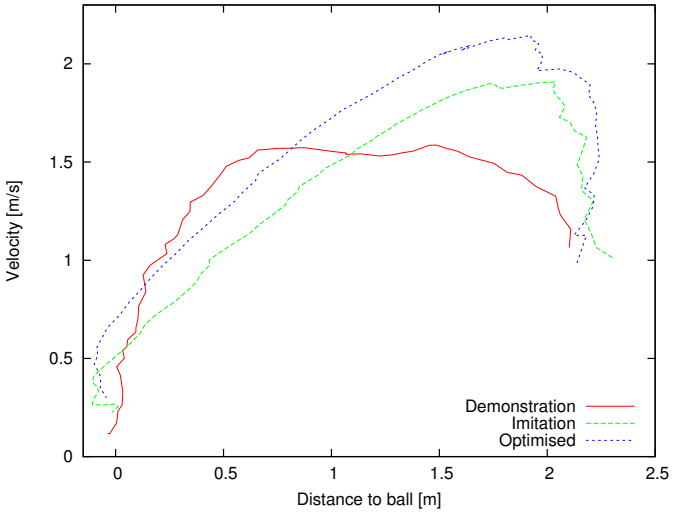

Fig. 4. Results of learning a translation trajectory for a ball interception: Trajectory demonstrated by a human (red), initial imitation of a z-policy with degree of 4 (green), eNAC1 optimised trajectory (blue).

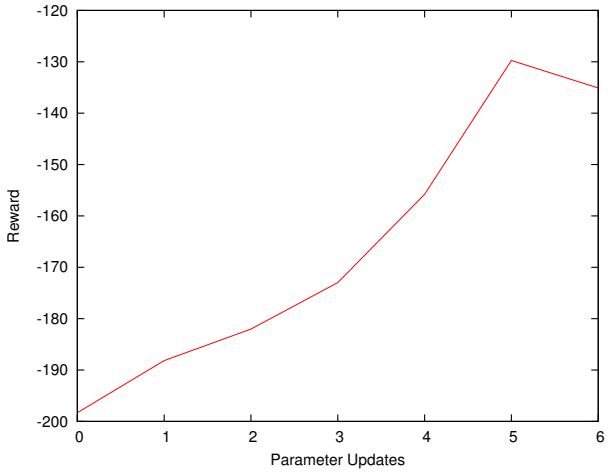

Fig. 5. Reward history for improving the demonstrated trajectory

to find a white box policy with only little a-priori knowledge about the target system.

This result is achieved by learning the parameters of a rational function in z-space with policy gradient methods. Classical control theory offers techniques for the analysis of z-policies.

The z-policy representation allows a implementation as state machine and the usage of the proposed imitation approach for control structures of linearisable control problems as shown in the cartpole experiment. Furthermore, we showed how to compute the policy gradient for z-policies of arbitrary degree. This facilitates the usage of the episodic natural actor-critic algorithm to optimise the policy parameters to a local minimum as shown in both case studies.

A regularisation term is introduced in order to stabilise the computation of the natural actor-critic independently of the policy structure. Policy parameters that have the same influence on the resulting action cannot be optimised without such a regularisation term. Not using this term would otherwise lead to numerical inversion problems of the Fisher matrix, which are avoided this way.

In order to show the feasibility of the proposed approaches – i.e. policy gradient methods – also in the presence of sensor noise, we performed an evaluation in a real RoboCup scenario.

In contrast to neural network-based approaches, the solution of the presented approach can be checked analytically. Finally, the z-policy is more expressive than PID controllers and easier to apply because of the imitation approach.

For futur research activities we propose to extend the policy structure for multiple inputs instead of making use of the decomposition step. It might be useful to involve some nonlinear basis functions, in order to increase the expressivity of the policy. Finally the way how the non episodic natural actor-critic can benefit from regularisation term has to be evaluated.

REFERENCES

[1] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," Tech. Rep., 2001.

[2] R. S. Sutton and A. G. Barto, "Reinforcement learning: Introduction," 1998.

[3] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Machine Learning*, 1992.

[4] R. S. Sutton and D. Mcallester et al., "Policy gradient methods for reinforcement learning with function approximation," 1999.

[5] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomput.*, vol. 71, no. 7-9, 2008.

[6] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, 2008.

[7] D. A. Harville, *Matrix Algebra From a Statistician's Perspective*. Springer, 2008, (last accessed 2010-06-09).

[8] J. R. Peters, "Machine learning of motor skills for robotics," Ph.D. dissertation, Los Angeles, CA, 2007.

[9] P. Regalia, Ed., *Adaptive IIR Filtering in Signal Processing and Control (Electrical and Computer Engineering)*. CRC Press, 1994.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, October 2007.

[11] M. Riedmiller, S. Lange, S. Timmer, and R. Hafner, "CLS2: Closed Loop Simulation System," 2004.

[12] A. Witsch, "Applying policy gradient reinforcement learning to optimise robot behaviours," Master's thesis, University of Kassel, Germany, 2010.

[13] M. Riedmiller, J. Peters, and S. Schaal, "Evaluation of policy gradient methods and variants on the cart-pole benchmark," in *ADPRL 2007*.

[14] P. Baer, R. Reichle, and H. Skubch et al., "Carpe Noctem 2009," in *RoboCup 2009 International Symposium*. TU Graz, 2009.