17th Mediterranean Conference on Control & Automation
Makedonia Palace, Thessaloniki, Greece
June 24 - 26, 2009

A Penalty Function Method for Exploratory Adaptive-Critic Neural Network Control

Gianluca Di Muro and Silvia Ferrari

Abstract— **A constrained penalty function method for exploratory adaptive-critic neural network (NN) control is presented. While constrained approximate dynamic programming has been effective to guarantee closed-loop system performance and stability objectives, in the presence of a change in the plant dynamics it may not have the necessary plasticity to explore and fully adapt to the new behaviors of the plant, if these violate the constraints. A generalized constrained approach is introduced to overcome these limitations. Through this methodology it is shown that NNs are not only capable to acquire new plasticity when necessary, but also can adjust their parametric structure reducing their hidden nodes and becoming more computationally efficient.**

Index Terms— **Approximate dynamic programming (ADP), constrained optimization, neural networks (NNs), forgetting, penalty function.**

I. Introduction

OPTIMAL control finds a plethora of applications in different areas, spanning from finance [1] to buffer management [2] and aerospace engineering [3]–[5]; yet, the known curse of dimensionality [6] often makes finding and implementing solutions impractical. Approximate Dynamic Programming (ADP) [7], has emerged as an efficient answer to this issue. ADP approximates the optimal solution through functional parametric structures, as Artificial Neural Networks (ANNs). The method has been proved to converge to the optimal solution with proper policy iteration (proof in [8]); besides it does not require *a priori* knowledge of the process dynamics, but the system is capable to learn the optimal policy (control) on-line. Albeit these appealing features, ADP NN controllers have not found to assure and maintain a high level of safety and performance at all times and they are often criticized for the lack of closed-loop stability and performance guarantees that characterize classical controllers.

A recent approach [9] avoids these limitations, adopting a constrained formulation, through which prior knowledge of linearized equations of motion is used to guarantee minimum performance requirements, crucial in real-time aerospace applications, among others. As shown in [9] ADP NN controllers do provide the same performances of classical controllers if the plant is in a linear parameter-varying (LPV) condition, still being capable to adapt online to new nonlinear dynamics and approximate the optimal solution over time, whereas classical controllers are suboptimal. Nonetheless, constrained optimization gives enough plasticity to learn new dynamics online, but does not allow violation of the constraints; if new dynamics contradict the linear-parameter varying region, constrained ADP is not capable to adapt and approximate the new optimal solution.

In this paper we introduce new plasticity to the network, giving further exploratory features to the parametric structure (i.e.: NNs), introducing a penalty function. Through this formulation, if new knowledge contradicts Long Term Memory (LTM) constraints, the importance of constraints is lessened over time; moreover if new information is capable to give better performance, the constraints are gradually discharged. Also, new memory is released and the network is capable to dismiss useless hidden nodes, reducing its adjustable parameters and its computational complexity.

This work was supported by National Science Foundation, under grants ECS CAREER 0448906, and ECS 0823945.

G. Di Muro is a graduate student of Mechanical Engineering at Duke University, Durham, NC 27707, USA gianluca.dimuro@duke.edu.

S. Ferrari is with Faculty of Mechanical Engineering at Duke University, Durham, NC 27707, USA sferrari@duke.edu.

II. Methodology

Given the infinite-horizon optimal control problem, the cost function to be minimized with respect to the control law takes the form,

$$J = \lim_{t_f \to \infty} \sum_{t_k=t_0}^{t_f-1} \mathcal{L}[\mathbf{x}(t_k), \mathbf{u}(t_k)] \tag{1}$$

according to [10], subject to the dynamic constraints imposed by the plant dynamics, which are not supposed to be known *a priori*. The lagrangian $\mathcal{L}$ is the cost associated with one time increment, $\mathbf{x}_k \in \mathbf{X} \subset \mathbb{R}^n$ is the state, $\mathbf{u}_k \in \mathbf{U} \subset \mathbf{R}^m$ is the control, whereas the plant dynamics are subjected by a difference equation of the form

$$\mathbf{x}(t_{k+1}) = \mathbf{f}[\mathbf{x}(t_k), \mathbf{u}(t_k)] \tag{2}$$

It should be stressed that in (2) the form of $\mathbf{f}$ may not be known *a priori* and may be learned online. ADP provides an approximate solution to the minimization of (1) subject to (2). Specifically, parametric structures (in our case feedforward NNs) are used to approximate the optimal control law and the relative cost function. In [11] several algorithms are presented to update these two functions; without loss of generality, we are going to focus to the Heuristic-Dynamic Programming (HDP) Algorithm. Given the recursive relation of dynamic programming, the value function may be minimized exclusively with respect to the present value of the control, $\mathbf{u}_k$, if the future cost of operation is optimal; ADP provides an estimate of the future cost-to-go

through a policy improvement routine. Therefore we have:

$$V^*(\mathbf{x}_k^*) = \min_{\mathbf{u}_k}\{\mathcal{L}(\mathbf{x}^*, \mathbf{u}_k) + V^*(\mathbf{x}_{k+1}^*\} \tag{3}$$

where in eq:recurrRel V^* and $\mathbf{x}^*$ refer to the optimal policy and the optimal state, respectively.

Through this formulation, the optimal control law is approximated by a NN, called the *actor*, and the NN weights are adjusted to approximate the optimal solution; similarly the value function (cost-to-go) is approximated by another NN, called the *critic*. As shown in [8], if the control law and the value function approximations, c_l and V_l at a generic iterative step l, are updated through a policy improvement routine, for which the control law is such to minimize the forecasted value function and then the value function determination is based on this improvement, they eventually converge to the optimal solution, provided that the plant dynamics do not suddenly change over time. Specifically the next iteration for the actor is given by:

$$\mathbf{c}_{l+1}[\mathbf{x}(t_K)] = \arg\min_{\mathbf{u}(t_K)}\{\mathcal{L}(\mathbf{x}(t_K), \mathbf{u}_k) + V_l[\mathbf{x}(t_K), \mathbf{c}_l]\} \tag{4}$$

where $\mathbf{c}_l$ is approximated by a NN and the minimization problem given by (4) has to be expressed in terms of the adjustable parameters of the NN; similarly the critic update is given by:

$$V_{l+1}[\mathbf{x}(t_K), \mathbf{c}_{l+1}] = \mathcal{L}[\mathbf{x}(t_K), \mathbf{c}_l] + V_l[\mathbf{x}(t_K), \mathbf{c}_{l+1}] \tag{5}$$

The ADP algorithms (4) and (5) are locally implemented using only one state sample at every iteration $k = l$. According to the methodology described in [12], the NN parameters are partitioned in Long Term Memory (LTM) and Short Term Memory (STM) connections, and former are expressed as a function of the latter in order to guarantee stability performance and enforcing the gain-scheduled controllers dynamics, when the plant is in the LPV regime. Therefore we can claim that:

$$\mathbf{w}_L = \mathbf{g}(\mathbf{w}_S) \tag{6}$$

where in (6) we have addressed with $\mathbf{w}_L$ and $\mathbf{w}_S$ the LTM and STM weights, respectively and $\mathbf{g}$ is the function which maps the ones into the others and assures the satisfaction of the constraints. Hence we can formulate the problem as a constrained optimization problem and at every iteration step we are aiming to apply equations (4) and (5), subject to (6).

There are two different well established methodologies to solve the constrained optimization problem. A first approach is to augment the function to be minimized by the implicit equality constraints, obtaining an augmented Lagrangian to be minimized with respect to all variables, which are given by the ANN's weights. Another approach is to ensure the relaxed constraints through direct elimination. In order to introduce new plasticity into the network and let the LTM to be gradually forgotten, a new type of agumented Lagrangian may be introduced, defined as:

$$J_l = e_l(\mathbf{w}_L, \mathbf{w}_s) + \boldsymbol{\lambda}_l^T \mathbf{F}_l \mathbf{g}(\mathbf{w}_L, \mathbf{w}_s) + \frac{1}{2}\, c_l \|\mathbf{g}(\mathbf{w}_L, \mathbf{w}_s)\| \tag{7}$$

where in (7) $\mathbf{F}_l$ models the ability of the NN to forget the LTM, expressed by the function $\mathbf{g}$; finally the index l refers to the l^{th} epoch in sequential training. In this paper we will adopt the approach of direct elimination and it will be described in detail in section II-A.

A. Forgetting: exploratory adaptive function approximation

Suppose that an Artificial Neural Network (ANN) is composed of Short Term Memory (STM) and Long Term Memory (LTM) connections and that it has been algebraically trained to satisfy the LTM training sets $\mathcal{T}_{\text{LTM}}^{1\ \text{out}} = \{\xi^j, \zeta^j\}_{j=1,\dots,r_{\text{LTM}}}$ and $\mathcal{T}_{\text{LTM}}^{1\ \text{der}} = \{\nu^l, \chi^l\}_{l=1,\dots,p_{\text{LTM}}}$ which contain input/output and derivative samples, respectively and that it has also been re-trained through a constrained backpropagation approach (CPROP) to suppress interference using an STM training set $\mathcal{T}_{\text{STM}}^1 = \{y^k, u^k\}_{k=1,\dots,r_{\text{STM}}}$, as proposed in [13]. Only to deal with simpler notation, we assume that $p_{\text{LTM}} = r_{\text{LTM}}$ and $\mathcal{I}_{\text{LTM}} = \{\xi^j\}_{j=1,\dots,r_{\text{LTM}}} = \{\nu^j\}_{j=1,\dots,r_{\text{LTM}}}$; i.e.: derivative information is available for the same set, for which the output information is provided. In the ADP context the aforementioned ANN might be the parametric structure to approximate the actor function and the constraints would be constituted by linearized dynamics, provided by a set of linear-time invariant vertex controllers, as proposed in [9].

Now let us suppose that there is a new training set $\mathcal{T}_{\text{STM}}^2$ accessible at a later time, which partially contradicts the LTM constraints and contains the previous STM training set; therefore we have that $\mathcal{T}_{\text{STM}}^2 = \mathcal{T}_{\text{STM}}^1 \bigcup \hat{\mathcal{T}}_{\text{STM}}^2$, where $\mathcal{T}_{\text{STM}}^1 = \{y^k, u^k\}_{k=1,\dots,r_{\text{STM}}^2}$ and $\hat{\mathcal{T}}_{\text{STM}}^2 = \{y^k, u^k\}_{k=1,\dots,r_{\hat{\text{STM}}}}$, with $\mathcal{I}_{\text{STM}}^2 \subset \mathcal{I}_{\text{LTM}}$ but different targets. From now on we will refer to the STM connections of the network using Latin letters, whereas Greek letters will address LTM connections. Also, inputs with a bar ($\bar{}$) will belong to the $\mathcal{T}_{\text{LTM}}$ training set, whereas inputs with a breve ($\breve{}$) will refer to the $\mathcal{T}_{\text{STM}}$ training set. For example $\bar{\mathbf{S}}^0$ refers to the STM transfer function matrix of zeroth order fed with the LTM inputs and similarly $\breve{\boldsymbol{\Sigma}}^1$ refers to the LTM transfer function matrix of first order fed with STM inputs; whereas the new targets, provided by the new training set $\hat{\mathcal{T}}_{\text{STM}}^2$, will be denoted by the hat ($\hat{}$).

Since CPROP analytically preserves the knowledge embedded into the LTM connections, it does not have -as it is- sufficient plasticity to contradict it; therefore in order to acquire sufficient plasticity we have to generalize CPROP, using appropriate weighting functions and a new inner product for the error, as we shall see. First let us partition the original LTM training input set $\mathcal{I}_{\text{LTM}}$ into the memory to be retained and the contradicted one to be gradually forgotten indicated as $\mathcal{I}_{\text{LTM}}^{\text{R}}$ and $\mathcal{I}_{\text{LTM}}^{\text{F}}$, respectively; thus we have that $\mathcal{I}_{\text{LTM}} = \mathcal{I}_{\text{LTM}}^{\text{R}} \bigcup \mathcal{I}_{\text{LTM}}^{\text{F}}$.

B. The Forgetting Algorithm

Taking inspiration from what has been found in biological neural networks [14]–[16], we claim that the association of new targets referred to a subset of the old LTM training set is induced by proactive interference. Hence, at this purpose, we introduce the persisting memory coefficient (PMC) $\eta \in (0,1)$: it may be thought as the probability that a certain neuron preserves its past LTM memory. If new samples, contradicting previous knowledge, are presented several times to the NN, the PMC is progressively reduced and approaches zero, which is equivalent to total forgetting. Also, we can exclude the extreme values (0 and 1): since they represent a reduced NN and the original NN, respectively.

First of all let us assume, without the loss of generality, that the first $\bar{P}$ LTM samples are not reproposed to the network and that the remaining $\bar{H} = r_{\text{S}\hat{\text{T}}\text{M}} - \bar{P}$ are affected by proactive interference. We can partition the LTM synapses accordingly so that $\boldsymbol{\omega}^T = [\boldsymbol{\omega}_1 \ \boldsymbol{\omega}_2]^T$, with $\boldsymbol{\omega}_1 \in \mathbb{R}^{1\times 2\bar{P}}$ and $\boldsymbol{\omega}_2 \in \mathbb{R}^{1\times 2\bar{H}}$, if output and derivative information is provided for every LTM sample. We assume that the connections deputed to the part of the memory which has to be updated are partially active, with a percentage, expressed by the PMC, which may also be interpreted as fraction of the memory connections still active and decays as long as new STM samples contradicts the old LTM constraints. According to this hypothesis the actual NN output, computed on $\mathcal{T}^2_{\text{STM}}$, is given by:

$$\mathbf{u} = \check{\mathbf{S}}^0 \ \mathbf{v}^T + \check{\boldsymbol{\Sigma}}^0_1 \ \boldsymbol{\omega}_1^T + \check{\boldsymbol{\Sigma}}^0_2 \ \eta \ \boldsymbol{\omega}_2^T \tag{8}$$

and the LTM constraints may be written as:

$$\bar{\boldsymbol{\Phi}}_{11} \ \boldsymbol{\omega}_1^T + \ \bar{\boldsymbol{\Phi}}_{12} \ \eta \ \boldsymbol{\omega}_2^T \ = \ \bar{\mathbf{t}}_1 - \bar{\mathbf{F}}_1 \ \mathbf{v}^T \tag{9}$$

$$\bar{\boldsymbol{\Phi}}_{21} \ \boldsymbol{\omega}_1^T + \ \bar{\boldsymbol{\Phi}}_{22} \ \boldsymbol{\omega}_2^T \ = \ \bar{\mathbf{t}}_2 - \bar{\mathbf{F}}_2 \ \mathbf{v}^T \tag{10}$$

where we have partition all the operators accordingly. The first $2\bar{P}$ equations (9) are relative to the LTM knowledge to be retained and so take into account the induced interference to preserve the previous memory, whereas the remaining $2\bar{H}$ constraints (10) are affected by interference and can not take into account the reduction of the efficacy of the LTM connections $\boldsymbol{\omega}_2$, induced by the PMC. Moreover, we have to modify the cost function in order to differently weight the STM samples which contradict part of the LTM ones. In order to achieve that we introduce a new cost function, which takes the form:

$$\tilde{J} = \check{\mathbf{e}}^T \mathbf{G} \check{\mathbf{e}} \tag{11}$$

where $\check{\mathbf{e}}$ is the error vector of the whole NN on the STM training set and $\mathbf{G} \in \mathbb{R}^{r^2_{\text{STM}} \times r^2_{\text{STM}}}$, with $\mathbf{G} > 0$ to introduce a consistent metric. Therefore it exists the operator $\sqrt{\mathbf{G}}$ and we can define the *weighted error* vector as:

$$\tilde{\mathbf{e}} = \sqrt{\mathbf{G}} \ \check{\mathbf{e}} \tag{12}$$

thus we can express the weighted cost function (11) through (12) simply as:

$$\tilde{J} = \tilde{\mathbf{e}}^T \ \tilde{\mathbf{e}}$$

therefore we can use, again, CPROP to train the NN where the error vector is defined using (12). In order to have a minor weight of the STM samples, which contradict the LTM ones, we have simply chosen $\mathbf{G}$ to be:

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

where $\mathbf{I}$ is the identity matrix of r^2_{STM} order and $\mathbf{B} \in \mathbb{R}^{r_{\text{S}\hat{\text{T}}\text{M}} \times r_{\text{S}\hat{\text{T}}\text{M}}}$ and $\mathbf{B} > 0$ and consistently with the definition of the CPM has been chosen to be: $\mathbf{B} = (1-\eta)\ \mathbf{I}$.

Similarly to be able to use a standard CPROP approach and compute the NN output as usual, we introduce the weighted LTM output weights, defined as: $\tilde{\boldsymbol{\omega}}^T = [\boldsymbol{\omega}_1 \ \tilde{\boldsymbol{\omega}}_2]^T$ and $\tilde{\boldsymbol{\omega}}_2 = \eta \boldsymbol{\omega}_2$; with this position the NN output (8) may be easily written as:

$$\mathbf{u} = \check{\mathbf{S}}^0 \ \mathbf{v}^T + \check{\boldsymbol{\Sigma}}^0 \ \tilde{\boldsymbol{\omega}}^T \tag{13}$$

and the LTM constraints in the new variables take the usual form:

$$\tilde{\boldsymbol{\Phi}} \tilde{\boldsymbol{\omega}}^T = \bar{\mathbf{t}} - \bar{\mathbf{F}} \mathbf{v}^T \tag{14}$$

where $\bar{\mathbf{t}} = [\bar{\mathbf{t}}_1 \ \bar{\mathbf{t}}_2]$, $\bar{\mathbf{F}} = \left[\bar{\mathbf{F}}_1 \ \bar{\mathbf{F}}_2\right]$ and $\tilde{\boldsymbol{\Phi}}$ is defined as follows:

$$\tilde{\boldsymbol{\Phi}} = \begin{bmatrix} \bar{\boldsymbol{\Phi}}_{11} & \bar{\boldsymbol{\Phi}}_{12} \\ \bar{\boldsymbol{\Phi}}_{21} & \tilde{\boldsymbol{\Phi}}_{22} \end{bmatrix}$$

with $\tilde{\boldsymbol{\Phi}}_{22} = \bar{\boldsymbol{\Phi}}_{22} \ \boldsymbol{\Delta}^{-1}$, and $\Delta = \eta \ \mathbf{I}$, in this case, where $\mathbf{I}$ is the identity matrix of $2\bar{H}$ order.

C. Another interpretation of the PMC

It is possible to show that the augmented network, given by the system of equations (9) provides an output which is a linear combination between the STM and the LTM targets, provided that the constraints are constituted only by output information. We will consider two extreme cases. First suppose the NN is trained to approximate the STM targets and the constraints which are in disagreement with the new STM training set are completely released: therefore the NN output, when fed with the new STM samples -i.e.: the ones contradicting a subset of the old LTM- is given by:

$$\bar{\boldsymbol{\Sigma}}_{21} \boldsymbol{\omega}_1^T + \check{\mathbf{S}}_2 \mathbf{v}^T = \check{\mathbf{t}} \tag{15}$$

where we have used the bar on $\boldsymbol{\Sigma}_{21}$ since the LTM inputs coincide with the STM inputs for the given set. Now consider the same NN augmented with $\boldsymbol{\omega}_2^T$ weights to enforce the satisfaction of the old output; thus the following equation holds:

$$\bar{\boldsymbol{\Sigma}}_{21} \boldsymbol{\omega}_1^T + \bar{\boldsymbol{\Sigma}}_{22} \boldsymbol{\omega}_2^T + \check{\mathbf{S}}_2 \mathbf{v}^T = \bar{\mathbf{t}} \tag{16}$$

According to the methodology we can use (16) to find an explicit expression for $\boldsymbol{\omega}_2^T$, since $\bar{\boldsymbol{\Sigma}}_{22}$ is a known square matrix and predesigned to be invertible; if we substitute from (15) into (16) to evaluate the expression of $\check{\mathbf{S}}_2 \mathbf{v}^T$, we have that :

$$\boldsymbol{\omega}_2^T = [\bar{\boldsymbol{\Sigma}}_{22}]^{-1} \ (\bar{\mathbf{t}} - \check{\mathbf{t}}) \tag{17}$$

Finally, taking this estimate of $\boldsymbol{\omega}_2^T$, and using it in the actual output of the network for the forgetting algorithm (10), under

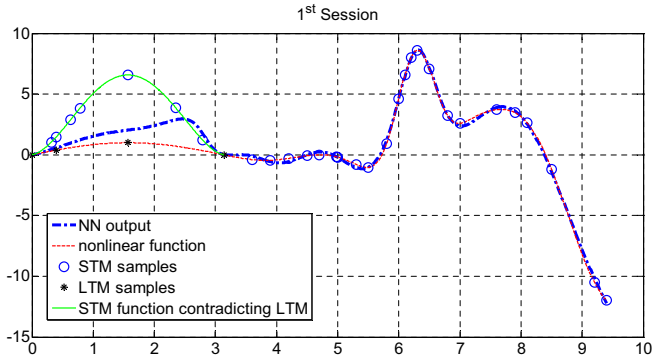

Fig. 1. Neural Network output, after the 1^{st} training session, composed of 300 epochs; PMC $\eta = .9988$

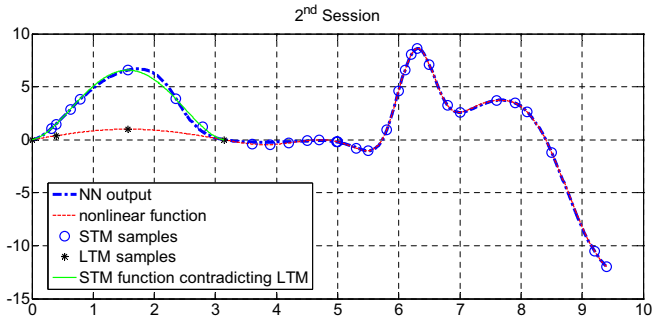

Fig. 2. Neural Network output, after the 2^{nd} training session, composed of 300 epochs; PMC $\eta = .33$

the assumption that (15) holds, we get that the NN output, $\tilde{\mathbf{u}}$, is given by:

$$\tilde{\mathbf{u}} = \breve{\mathbf{t}} + \eta \ (\bar{\mathbf{t}} - \breve{\mathbf{t}}) \tag{18}$$

which gives the parametric description of the segment individuated by $\breve{\mathbf{t}}$ and $\bar{\mathbf{t}}$.

III. Applications and Results

As an example, consider the nonlinear function plotted by a dashed line in Fig. 1 over a domain $\mathcal{D} = [0, 3\pi] \subset \mathbb{R}$. Suppose the shape of the function over a bounded subset $\mathcal{S} = [0, \pi] \subset \mathcal{D}$ is known *a priori* to be a sine function and $\mathcal{T}_{\text{LTM}}$

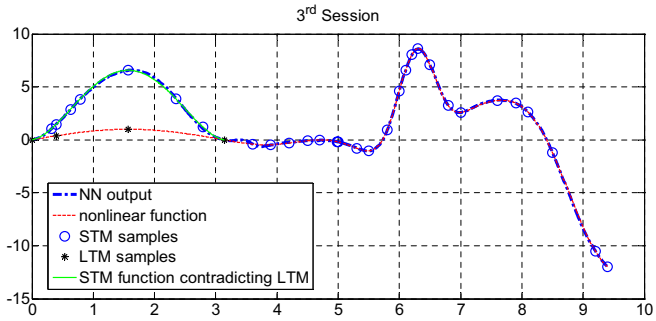

Fig. 3. Neural Network output, after the 3^{rd} training session, composed of 900 epochs; PMC $\eta = 1.\ 10^{-5}$

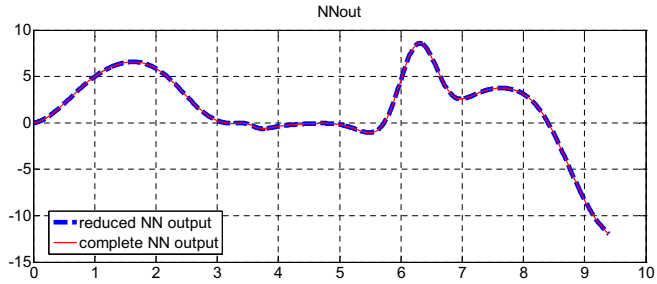

Fig. 4. Comparison between complete NN and reduced NN (without LTM weights) : solid and dashed line, respectively

is formed using the LTM samples shown in Fig. 1 by the asterisks. A sigmoidal NN with 16 hidden nodes is trained to approximate $\mathcal{T}_{\text{LTM}}$ using the Levenber-Marquardt algorithm [17], [18] . Then a new STM training set is available and new knowledge contradicts LTM constraints. The unknown function is supposed to be $h(y) = \alpha y^4 + \beta y^3 + \gamma y^2 + \delta y$, with $\alpha = .9503, \beta = -5.9708, \gamma = 9.0606, \delta = 1$ and is depicted with a solid line. Only some of the data contradicts old LTM constraints. The PMC parameter is decreased at every training session and more plasticity is induced into the LTM connections, as shown in Figs. 1, 2, 3. Finally the old LTM connections are removed and the hidden nodes of the network are reduced to 12, without deteriorating the performance of the network, as shown in Fig. 4, since the output weights associated to the part of the memory to be forgotten are completely damped by the PMC, which assumes extremely low values in the last training

IV. Conclusions

A generalized constrained ADP with penalty function is introduced. If new dynamics occur or LPV strategy is not locally optimal anymore, the novel approach is capable to assure enough plasticity and gradually lessen previous sub-optimal constraints. The problem has been reformulated to previous constraint ADP research, through an appropriate redefinition of the operators involved. Also, the structure and the dimension of the NN is flexible and hidden nodes associated to small output weights may be automatically completely released, reducing the dimension of the problem. Further research might update the PMC depending on the ratio of the standard deviation of the STM samples over the global standard deviation of the process (if noise is included) to be able to discriminate between noise and new dynamics to be incorporated.

References

[1] P. Chen and S. M. N. Islam, *Optimal Control Models in Finance*, P. M. Pardalos and D. W. Hearn, Eds. Springer US, 2005, vol. 95.

[2] W. Zhang and J. Hu, "Dynamic buffer management using optimal control of hybrid systems," *Automatica*, vol. 44, no. 7, pp. 1831 – 1840, 2008.

[3] A. E. Bryson, *Applied Linear Optimal Control: Examples and Algorithms*. Cambridge University Press, 2002.

[4] A. E. Bryson, Jr., "Applications of optimal control theory in aerospace engineering." *Journal of Spacecraft and Rockets*, vol. 4, pp. 545–553, May 1967.

[5] R. H. Chen and J. L. Speyer, "Improved endurance of optimal periodic flight," *Journal of guidance, control, and dynamics*, vol. 30, no. 4, pp. 1123–1133, 2007.
[6] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.
[7] W. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2007.
[8] R. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press, 1960.
[9] S. Ferrari, J. Steck, and R. Chandramohan, "Adaptive feedback control by constrained approximate dynamic programming," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 38, no. 4, pp. 982–987, 2008.
[10] R. F. Stengel, *Optimal Control and Estimation*. Dover Publications, 1994.
[11] S. Ferrari and R. F. Stengel, *Model-based Adaptive Critic Designs*, J. Si, A. Barto, W. Powell, and D. Wunsch, Eds. IEEE Press and John Wiley & Sons, 2004.
[12] S. Ferrari and M. Jensenius, "A constrained optimization approach to preserving prior knowledge during incremental training," *IEEE Transactions On Neural Networks*, vol. 19, no. 6, pp. 996–1009, 2008.
[13] G. Di Muro and S. Ferrari, "A constrained-optimization approach to training neural networks for smooth function approximation and system identification," *Proc. International Joint Conference on Neural Networks, Hong Kong*, pp. 2354–2360, 2008.
[14] G. Bush, L. M. Shin, and S. L. Rauch, "The counting stroop: a cognitive interference task," *Nature Protocols*, vol. 1, pp. 230–233, June 2006.
[15] J. T. Wixted, "The psychology and neuroscience of forgetting," *Annual Review of Psychology*, pp. 235–269, 2004.
[16] S. Fusi, W. F. Asaad, E. K. Miller, and X. Wang, "A neural circuit model of flexible sensorimotor mapping: Learning and forgetting on multiple timescales," *Neuron*, no. 54, pp. 319–333, April 2007.
[17] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.
[18] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Journal of Applied Mathmatics*, vol. II, no. 2, pp. 164–168, 1944.