Kecman V., ***Learning in Adaptive Backthrough Control Structure***, Proceedings of the IEEE Third International Conference on Algorithms and Architectures for Parallel Processing ICA^{3}PP -97), Melbourne, Australia pp. 611-624, World Scientific, Singapore, 1997

LEARNING IN ADAPTIVE BACKTHROUGH CONTROL STRUCTURE

VOJISLAV KECMAN
The University of Auckland, Department of Mechanical Engineering
Private Bag 92019, Auckland, New Zealand
E-mail: v.kecman@auckland.ac.nz

The paper presents new neural networks (NN) based *Adaptive Backthrough Control (ABC)* scheme for both linear and nonlinear dynamic plants. Unlike other *feedforward NN based control schemes* the ABC proposed here comprises of one neural network only which simultaneously acts as both - plant model (emulator) and the controller (inverse of the emulator). For linear plants, without noise, the resulting feedforward controller, providing that the order of the plant and plant model are equal, is a perfect adaptive poles-zeros canceller. In the case of nonlinear dynamic system, and for the monotonic nonlinearity, the proposed ABC control represents the nonlinear predictive controller. The ABC scheme is based on the discrete nonlinear (NARMAX) dynamic model. For such models and for monotonic nonlinearity, the calculation of the desired control signal is the result of the nonlinear optimization procedure with guaranteed convex search function and consequently with a unique solution.

1 Introduction and Basic Control Structures

This paper focuses on the NN and FLM based adaptive control[1]. In particular, after presenting the basic guiding ideas of NN based control approaches, we will introduce the *Adaptive Backthrough Control (ABC)* scheme as one of the most serious candidate for the future control of the large class of nonlinear, partially known and time-varying systems [Kecman, 1997; Kecman and Rommel, 1997; Rommel, 1997]. Recently, the area of NN control has been exhaustively investigated and there is a large number of different NN based control methods. (Rigorous comparisons show that the NN based controllers perform far better than the well established conventional options when the plant characteristics are poorly known [Bošković and Narendra, 1995]). A systematic classification of the very different NN control structures is a formidable task indeed [Agarwal, 1997]. Here, we will describe the approach based on feed-forward networks having 'static' neurons. Such networks are very closely related with the fuzzy models [Kecman and Pfeiffer, 1994].

So-called 'standard' or 'classic' NN based control is the one that uses two neural networks, Fig 1. This control structure comprises of NN_2 that represents the (approximate) model of the plant, and of NN_1 that functions as a controller. The latter one represents (approximate, again) inverse of NN_2. Note that NN_1 is an 'inverse' of the plant model and not of the plant.

[1] In what follows NN will stand for neural, fuzzy, neuro-fuzzy or fuzzy-neuro models.

The structure given in Fig 1 is almost standard one in the field of 'neurocontrol'. In this respect, here proposed *Adaptive Backthrough Control (ABC),* is in the line with the basic results and approaches from [Psaltis et al, 1988; Saerens and Soquet, 1991; Garcia and Morari, 1982; Jordan, 1993; Hunt and Sbarbaro, 1991; Narendra and Parthasarathy, 1991 and Widrow and Walach, 1996].

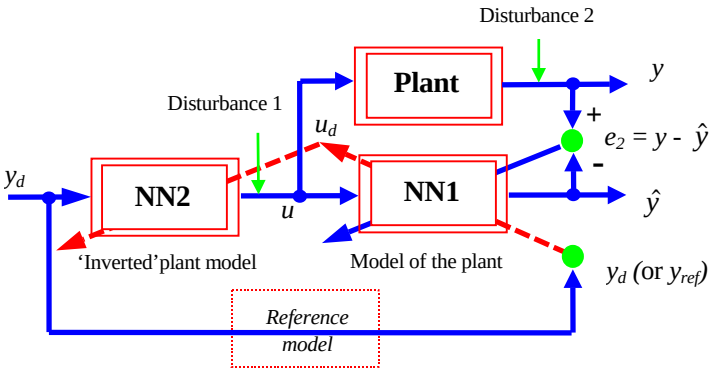

Figure 1 Neural Networks Based Adaptive Backthrough Control (ABC) Scheme

While being 'similar' in their graphical appearance, there are few important distinctive features that differ ABC approach from all other standard NN based control methods.

The principal feature of ABC is that, unlike the other approaches, it doesn't use standard training errors as the learning signals for adapting the controller (NN_1) weights. Rather, the true desired value y_d (signal that should be tracked, reference signal) is used for the training of NN_1. In this way, the desired but unknown control signal u_d, results from the *backward* transformation of the y_d *through* NN_2. The origin of the label for this approach as a *backthrough* method, lies in this backward step for the calculation of u_d. In this way the ABC basically represents a younger (and it seems more direct and powerful) relative of the 'distal teacher' idea from [Jordan, 1993] or of the [Saerens and Soquet, 1990], as well as of [Saerens, Renders, and Bersini, 1996] approach. These three algorithms use the error signal e_2 and its sign respectively. Besides, they use the steepest descent for the optimization. In ABC, as long as the control problem is linear in parameters (linear dependence of the cost function upon the NN weights) the recursive least squares (RLS) learning algorithm is strictly used. Note that in many cases for both NN and fuzzy logic model based controller this assumption about the linear in parameters model is a realistic and acceptable one. Note, however, that the proposed ABC algorithm doesn't strictly depend upon the use of RLS technique and that the standard gradient (EBP) or any other learning procedures can also be used.

Similarly to the adaptive inverse control (AIC) devised by Widrow, ABC control scheme proposed here is effective as long as the plant is a stable one. It solves the problems

of tracking and disturbance rejection for any stable plant. The same will be true in the case of unstable plants as long as the unstable plant is stabilized by some classic control method first. The control structure in Fig 1 has some of good characteristics of the idealized control system design with a positive internal feedback that doesn't require plant model NN_2 to be a perfect model of the plant [Tsypkin, 1972]. The later control scheme is structurally equal to the internal model control (IMC) approach. The reference block presented in Fig 1 is not required, unless some control of the control signal variable u is needed. All results below are obtained by using $G_{ref}(s) = 1$.

The structure (as well as a performance) of the NN based control system which comprises two NN is described in [Kecman, 1997; Kecman and Rommel, 1997; Rommel, 1997]. This structure is 'inherited' from the previously mentioned approaches and it is directly related to the classic EBP learning. The task of the network NN1 which acts as a controller is to learn the inverse dynamics of the controlled plant. Being properly trained and after receiving the desired plant output signal y_d, NN1 should be able to produce the best control signal u_d which would drive the plant to output this desired y_d. However, the ABC learning is different from the EBP algorithm. Note that in the ABC algorithm the best control signal u_d is calculated in each operating step and it is used for the adaptation of NN1's weights in order that this controller produces an output signal u, which should be equal or very close to the u_d. Thus, there is a great deal of a redundancy and it seems as though both the very structure of the whole control system and the learning can be halved. Having the signal u_d calculated, the controller network NN1 is not needed any longer. The ABC structure with only one NN which simultaneously acts as the plant model and as a controller (inverse plant model) is shown below in Fig 2.

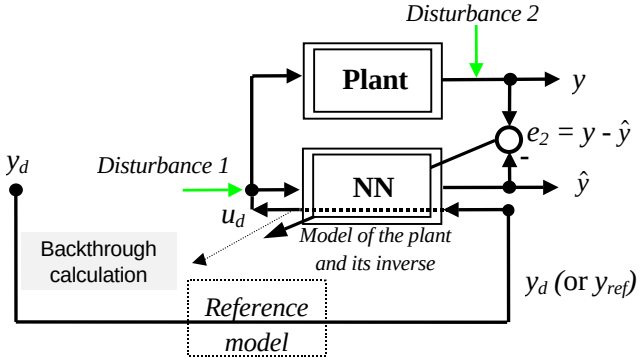

Figure 2 Neural or fuzzy network based Adaptive Backthrough Control (ABC) scheme with one network which simultaneously acts as the plant model and as a controller (inverse plant model).

The standard control task and the basic problem in controlling an unknown dynamic plant is to find the proper, or desired, control (actuation) value u_d as an input to the plant which should ensure that,

$$y(t) = y_d(t), \quad \forall t \tag{1}$$

where the subscript *d* stands for *desired.* The variables *y(t)* and $y_d(t)$ denote the actual plant output and desired (reference) plant output respectively. A controller that could produce this value u_d would be the best controller and the output of the plant would exactly follow the desired input y_d. In *linear* control, (51) will be ensured when,

$$G_{ci}(s) = G_p^{-1}(s). \tag{2}$$

Hence, the ideal controller transfer function $G_{ci}(s)$ should be the inverse of the plant transfer function $G_p(s)$. Because of many practical constraints, this is an idealized control structure [Kecman, 1988]. However, we can try to get as close as possible to this ideal controller solution ($G_{ci}(s)$). The ABC approach that is presented in this section, can achieve a great deal (sometimes even nearly all) of this ideal controller. The block diagram of the *ideal control* of any nonlinear system is given in Fig 3.

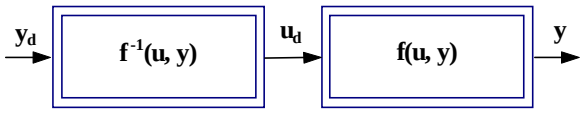

Figure 3 The ideal (feedforward) control structure for any plant.

f(u, y) represented in Fig 3 stands for any nonlinear mapping between an input **u(*t*)** and an output **y(*t*)**. In a general case of a dynamic system **f(u, y)** represents a system of nonlinear differential equations. Here we will primarily be concerned with discrete-time systems, and the model of the plant in the discrete-time domain will be in the form of nonlinear discrete equation **y(*k*+*1*) = f(u(*k*), y(*k*))**. Now, the basic problem is how to learn, or obtain, the inverse model of the unknown dynamic plant by using NN?

The wide application of NN in control is based on the universal approximation capacity of neural networks and fuzzy models. Thus a learning (identification, adaptation, training) of the plant and inverse plant dynamics represents both the basic mathematical tool and the basic problem to be solved. Therefore the analysis presented below assumes a complete controllability and observability of the plant.

So far as the representation of dynamic system is concerned, we use a so-called NARMAX model here. In the extensive literature on modeling dynamic plants, it was proved that under some mild assumptions any nonlinear, discrete and time invariant system can always be represented by the following NARMAX model,

$$y(k+1) = f\{y(k), \cdots, y(k-n); u(k), \cdots, u(k-m)\}, \tag{3}$$

where y_k and u_k are the input and output signals at instant *k,* and y_{k-i} and u_{k-j} (i = 1, ..., n and j = 1, ... ,m) represent the past values of these signals. Typically one can work with $n = m$. (3) is a

simplified deterministic version of the NARMAX model (there is no noise terms in it), and is valid for dynamic systems with K outputs and L inputs. For $K = L = 1$ we obtain the so-called SISO (single-input single-output) system which is studied here.

In reality, the nonlinear function f from (3) is very complex and generally unknown. The whole idea in the application of NN is to try to *approximate* f by using some known and simple functions which, in the case of the application of NN and FLM, are their activation and membership functions respectively.

This identification phase of the mathematical model (3) can be given a graphical representation (Fig 4). Note that two different identification schemes are presented in Fig 4 - *series-parallel and parallel.* (The names are due [Landau, 1979]). The identification can be done by using either of the two schemes,

$$y(k+1) = f\{y(k),\cdots,y(k-n);u(k),\cdots,u(k-n)\} \quad (4)$$

Series-Parallel

$$y(k+1) = \{\hat{y}(k),\cdots,\hat{y}(k-n);u(k),\cdots,u(k-n)\} \quad (5)$$

Parallel

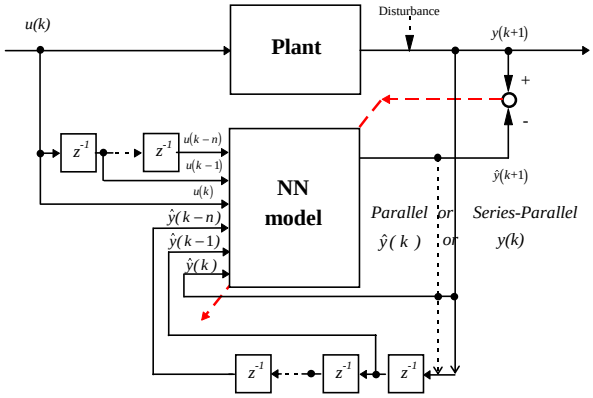

Figure 4 Identification scheme using NN.

It is hard to say which scheme is a better one. Narendra and Annaswamy (1989) showed (for linear systems) the series-parallel method to be globally stable but similar results are not available for the parallel model yet. The parallel method has the advantage of avoiding noises existing in real plant output signals. On the other hand series-parallel scheme uses actual (meaning correct) plant outputs and this generally enforces identification.

'Historically', seemingly the strongest stream of the NN based control strategies are the *feed-forward control* schemes. There were a few relatively independent and partly dissimilar basic directions in the search for a good control strategy. However, the leading idea was the same in these, otherwise different, control schemes. The final goal was always the determination of a

good inverse model of the plant dynamics $\mathbf{f}^{-1}\mathbf{(u, y)}$ as required in ideal feedforward control structure in Fig 3.

This can be done by using many different and, more or less suitable, approaches. We will only mention the three most popular ones as presented in [Psaltis et al, 1988]. In their paper they introduced and discussed - a *general, indirect* and *specialized* learning architecture for the NN based control of the *stable nonlinear* plants. (Independently, the same approach as the general architecture was developed in [Jordan and Rumelhart, 1992] and it was named as a *direct inverse modeling.* This is basically an off-line procedure and for nonlinear plants it will usually precede the on-line phase. (If the plant is unstable a stabilization with a feedback loop is necessary. That can be done with any standard control algorithm).

Detailed description of these approaches can be found in [Kecman, 1997]. Here we will discuss only the *specialized learning architecture* as an introduction to the presentation of the ABC scheme.

1.1 Specialized learning architecture

The specialized learning architecture is given in Fig 5 below.

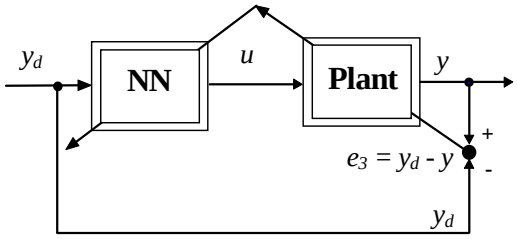

Figure 5 Specialized learning architecture.

This structure operates in an on-line mode and it trains a neural network to act as a controller in the region of interest (meaning that it is 'a goal directed'). In this way this scheme avoids some of drawbacks of the general and indirect structures. Here in the specialized learning architecture, the controller no longer learns from *its* input-output relation, but from a *direct* evaluation of the *system's* performance error $e_3 = y_d - y$. The network is trained in order to find the best control value u that drives the plant to an output $y = y_d$. This is accomplished by using a steepest decent (EBP) learning procedure. Despite the fact that the specialized architecture operates in an on-line mode, in the case of nonlinear plant, a pretraining or off-line phase, will usually be both very useful and highly recommended here.

A critical point of the specialized learning architecture is that the EBP learning algorithm requires knowledge of the Jacobian matrix of the plant. The emergence of the Jacobian

is obvious. The subjects of the learning are the weights of NN and, in order to correct the weights in the right directions, learning algorithm should have information of the error caused by wrong weights. But, there is no such direct information because the plant intervenes between the unknown NN outputs or control signals *u*, and the desired plant outputs *y*. The 'teacher' in the EBP algorithm is typically an error (here the performance error $e_3 = y_d - y$) and this teacher is a distal one now [Jordan and Rumelhart, 1992].

Let us show the EBP algorithm for the general 'distal teacher' learning situation. In order to apply the EBP algorithm, the NN and the plant are treated as a single neural network in which the plant represents a fixed (unmodifiable) output layer. In this way the real OL of NN becomes the hidden one. The whole EBP learning now concerns a calculation of proper *deltas*, or error signals δ, associated with each neuron. In order to find these signals, the delta signals δ_o for the true output layer (OL) neurons of NN should be determined first. For the sake of simplicity we will show how this can be done for the SISO plant avoiding in this way matrix notation. Having δ_o enables a relatively straightforward calculation of all other deltas and specific weights changes.

Assume that NN is a network operating in parallel mode by having 2*n* inputs (where *n* represents the model order), or that NN is given by model $u(k+1) = f\{y_d(k),\cdots,y_d(k-n); u(k),\cdots,u(k-n)\}$. There are 'enough' hidden layer (HL) neurons which can provide a good approximation, and there is one *linear* OL neuron having the output *u*. The plant is given as $y = f(u, y)$. The EBP algorithm for learning the NN weights is a steepest descent procedure, and the cost (error) function to be optimized is,

$$E = \frac{1}{2}e^2 = \frac{1}{2}(y_d - y)^2 = E(w_{ij}). \tag{6}$$

Note that $y = f(u, y)$ and $u=(f_n(u_n))$, so that $y = f[f_n(u_n), y]$ where f_n and u_n stand for the activation function of, and input signal to, the OL neuron respectively. (For a linear OL neuron, *f* represents identity, $u = u_n$). In order to calculate the OL neuron's error signal δ_o we apply chain rule in calculation of the cost function's gradient,

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial u_n}\frac{\partial u_n}{\partial w_i} =$$

$$\underbrace{(y_d - y)(-1)\frac{\partial f(u,y)}{\partial u} f_n'}_{-\delta_o}\frac{\partial u_n}{\partial w_i} = -\delta_o\frac{\partial u_n}{\partial w_i} \tag{7}$$

The error signal of the OL neuron is denoted as δ_o f_n' stands for the derivative of the OL neuron activation function and here, for a linear neuron, $f_n' = 1$. For the multilayer perceptron network, where the input signal to the neuron is obtained as a scalar product $u_n = \mathbf{w}^T\mathbf{x}$, the

derivative $\partial u_n / \partial w_i = x_i$. (Note that in the RBF networks, this expression for the OL error signal δ_o will be identical. There will be a difference in the expressions for the learning of the HL neuron weights though).

It is important to realize that the derivative $\partial f(u, y) / \partial u$ represents the Jacobian of the plant. Here, for SISO plant this is a scalar, or more precisely (1, 1) vector. Generally, a plant dynamics and the Jacobian are unknown which is a serious shortcoming of this final result that is otherwise 'nice and handy'. There are two basic approaches to overcome this difficulty which will be commented below.

Let us first give some final comments concerning the weights adaptation in the specialized learning architecture with the following assumptions: the Jacobian is known, the OL neuron is linear and input to the neuron is calculated as a scalar product. With these assumptions we can directly use the standard EBP algorithm. Having the error signal δ_o we can easily calculate the HL deltas, as well as corresponding weight changes $\Delta w_i = \eta \delta x_i$. Hence, in this *backpropagation through the plant algorithm,* the determination of the networks' OL delta signal is the most important step. In order to do that the Jacobian of the plant must be known.

Generally this is not the case, and two alternative approaches on how to handle the ignorance of the plant Jacobian are: *the approximation of the plant Jacobian by its sign* and *the distal teacher approach.* The first approach is presented in [Saerens and Soquet, 1991)] and the second one in [Jordan and Rumelhart, 1992].[2] This second, or distal teacher approach, differs significantly in using *the Jacobian of the plant forward model* instead of the real plant's Jacobian, or instead of the signs of the Jacobian derivatives of the real plants. The whole feedforward control system now comprises the two neural networks. One is a model of the plant and the second NN, which will be trained with the help of the first one, acts as a controller. This structure is practically same as the one of the ABC as given in Fig 1. (The structures only are the same but the learning is different).

2 Learning in the ABC Structures Which Comprises One NN Only

Fig 2 shows the ABC scheme having one NN which acts as both the plant model (emulator) and the controller of the plant. It is closely related with Fig 5 with a difference that in the ABC structure the desired control signal u_d, and not the error signal δ is calculated. This calculation is done in an on-line mode. In the case that the plant is nonlinear the standard approach is that this on-line part is preceded by the off-line learning of the plant model. The advantage is that the off-line learning will produce a better set of initial weights for the on-line operation. In the

[2] Similar approaches and structures have been proposed and used in many papers from Widrow and his coworkers under the global name of the *adaptive inverse control.*

case of *nonlinear plants pretraining of NN is essential.* In the case of the linear plant this pretraining is not that important.

Sometimes it may be useful to introduce a reference model, too. This step is not crucial for the ABC approach but an important result could be that with a reference model a tuning of the control effort is possible. This might be necessary for many real existing systems because the actuators usually operate only within a specific range, and leaving this range is either not possible or can harm the system's performance.

The basic idea of the ABC is to design a plant emulator which simultaneously acts as the inverse of the plant (or, as an adaptive controller). In this way, the problem of finding the 'best' control signal u_d will be solved. In general, this value u_d is not available. By using the ABC approach we can find this desired control values u_d that will usually be very close to the ideal ones.

During the operation of the whole system (meaning, during the adaptation or learning of both the plant model and controller parameters) there are several error signals that may be used for the adjustment of these parameters. Similarly to [Jordan and Rumelhart, 1992] we define the following errors in Table 1. (If the reference model is used the value y_d should be replaced with the output value of the reference model y_{ref}).

Table 1 Definition of the errors

$e_1 = \hat{u}_d - \hat{u}$	controller error
$e_2 = y - \hat{y}$	prediction error
$e_3 = y_d - y$	performance error
$e_4 = y_d - \hat{y}$	predicted performance error

Other methods [Psaltis et al, 1988; Widrow and Walach, 1996; Widrow and Plett, 1996; Saerens and Soquet, 1989 and Jordan and Rumelhart, 1992] use different approaches in order to find the error signal term that can be used to train the controller (see [Kecman, 1997]). The ABC structure originated from these previous approaches with a few basic and important differences. First, the estimate of the desired control signal u_d can be calculated, and an error (delta) signal as in 'distal teacher' approach is not needed. For the ABC of **linear systems**, the calculation of u_d is straightforward. The forward model (NN in Fig 2) is given as,

$$\hat{y}(k+1) = \sum_{i=1}^{N} w_{2,i} \cdot x_{2,i} = \mathbf{w}_2^T \cdot \mathbf{x}_2 \qquad (8)$$

where $N = 2n$, n is the order of the model and x_2 is an input vector to NN_2 comprised of present and previous values of u and y. For the calculation of the desired value $\hat{u}_d$ this equation should and can be rearranged in respect to the input of the neural network NN_2,

$$\hat{u}_d(k) = \left(\frac{y_d(k+1) - w_{2,1} y_d(k) - \cdots - w_{2,n} y_d(k-n+1) - w_{2,n+2} \hat{u}(k-1) - \cdots - w_{2,2n} \hat{u}(k-n+1)}{w_{2,n+1}} \right) \quad (9)$$

Therefore, when applied to the control of linear systems, the calculation of the control signal u_d by using (9) is similar to the predictive (deadbeat) controller approach. Note that in the calculation of the best estimates of desired control signal $\hat{u}_d(k)$ to the plant and to the NN, the desired output values of the system $y_d(k+1), y_d(k), \cdots, y_d(k-n)$ are used. It is interesting to note that instead of using the present and previous desired values, one can use the present and previous actual plant outputs $y(k), \cdots, y(k-n)$. It seems as though this second choice of the variables is a better one.

In the case of the **nonlinear system** control, the calculation of the desired control signal u_d which corresponds to the desired output from the plant y_d, is much more involved task. For the *monotonic* nonlinearities (i.e., for the one-to-one-mapping of the plant inputs u into its outputs y) control signal u_d can be calculated by an *iterative algorithm* that guarantees finding of proper u_d for any desired y_d. Two other alternative approaches to the calculation of the desired u_d which deserve more investigations are given in [Rommel, 1997]. *This is the crucial result in the proposed ABC algorithm.*

2.1 Iterative calculation of u_d with steepest descent (gradient method) for nonlinear plants

Given y_d, the desired value u_d can be calculated by using a standard (iterative) steepest descent method, which basically represents a gradient search algorithm. Note that the NN in Fig 2 is a NARMAX model as given in (4) or (5). We rewrite the series-parallel model equation below,

$$y(k+1) = f\{y(k), \cdots, y(k-n); u(k), \cdots, u(k-n)\}. \quad (10)$$

If the function f of the identified plant model is a monotone increasing or decreasing one, than this NARMAX model represents an one-to-one mapping of the desired control signal u_d (and corresponding previous values of u and y) into the desired y_d.

Now, the basic idea of the adaptive backthrough calculation of u_d for any given y_d is the same as in linear case. But unlike in linear case above, where the solution is given by (9), in the case of *general nonlinear model* which is represented by NN in Fig 2, it is not possible to express u_d explicitly any longer. Therefore, the solution should be obtained by some numerical iterative procedure. Here, we propose the use of the standard gradient algorithm.

Proposition: In case of *monotonic nonlinearity*, it is always possible to find the desired control signal u_d to any desired degree of accuracy, by using *sufficiently small* optimization step of the gradient optimization method.

Proof: A proof follows from the standard properties of the gradient optimization algorithms. Having a NN as a NARMAX model (10), we define the function,

$$e(k) = y(k+1) - f = 0, \tag{11}$$

and problem to solve is to find $u_d(k)$ for known $y_d(k+1)$. Note that all past values of y and u which appear in f are known, and the problem to solve is to find the root $u_d(k)$ of (11). This *one-dimensional* search problem will be solved by finding the minimum of the function,

$$E = e(k)^2. \tag{12}$$

Thus, we transformed the problem of finding the root of the nonlinear equation (11) into the minimization problem of the equation (12). In this specific case of a monotonic mapping f, the 'hypersurface' E is a convex function having a known minimum $E(u_d) = 0$. For a given $y_d(k+1)$ and known past values of y and u, the root u_d represents the mapping f^{-1} of the known point from a $2n$-dimensional space into one-dimensional space ($\Re^{2n} \rightarrow \Re$). For a monotonic nonlinear mapping $f\{y(k),\cdots,y(k-n);u(k),\cdots,u(k-n)\}$, the solution u_d is unique and it can be obtained by any one-dimensional search technique. (The solution in the case of a non-monotonic functions is slightly more involved, it requires additional conditions (constraints) and it is the subject of current research).

Fig 6 demonstrates the geometry of the procedure for the simplest case of NN representing a ($\Re \rightarrow \Re$) mapping and having two neurons with Gaussian AF only.

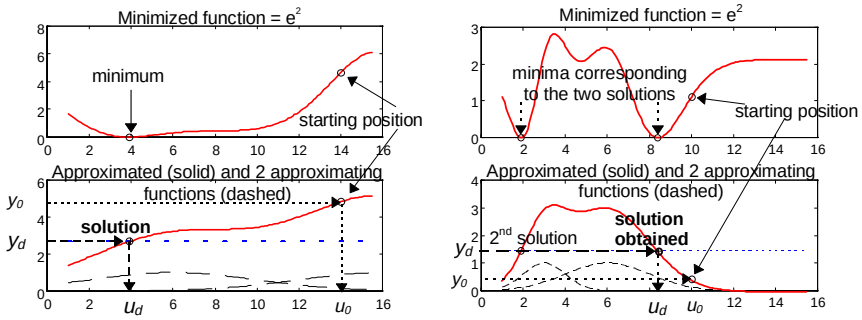

Figure 6 Iterative calculation of u_d with gradient method. Monotonic NL function (left) and a non-monotonic NL function (right)

Left graphs show a convex function $E = e^2$ (above) for a monotonic nonlinear function f (below), while the right graphs show the solutions for the non-monotonic nonlinear mapping f. The mathematics in the case of nonlinear dynamic systems is much more involved and without a great hope for graphical presentation of the corresponding optimization procedure. In the case of the lowest (or first) order dynamic system the graphical representation is possible but

hard to follow. The details of the numerical part of the *backthrough* calculation of u_d can be found in [Kecman, 1997].

3 Simulational results

Nonlinear 1st order dynamic plant adapted from [Narendra and Parthasarathy, 1991] should be controlled by the ABC scheme comprised of the one network only. The plant equation is given below,

$$y(k) = \frac{y(k-1)}{1+y^2(k-1)} + u^3(k-1).$$

The neural network that simultaneously acts as a plant model and as its controller is comprised of 39 neurons in hidden layer. Basis functions in all HL neurons are the two-dimensional Gaussians with the same covariance matrix $\Sigma = diag(0.2750, 0.0833)$, and with positions determined by an orthogonal least squares selection procedure (Orr, 1996). NN was pretrained by using 1000 data pairs. Training input signal was a uniformly distributed random signal. (Note that the ABC control structure is much simpler than the one in (Narendra and Parthasarathy, 1991). They used two NN for the identification and one as a controller. Each network had 200 neurons. Besides, in the off-line training phase they used 25 000 training pairs).
After the training was done, a number of simulation runs had proved very good performance of the ABC scheme while controlling *time invariant nonlinear* system. Fig 7 (left) shows the plant response while tracking input $y_d = sin(2\pi k / 25) + sin(2\pi k / 10)$. The plant response is indistinguishable from the desired trajectory. The tracking is perfect.
Much more complex task is to control the *time variant nonlinear* plant. There is no *general* theory, approach or method in *adaptive control of nonlinear time variant* plants. These are the toughest control problems anyway. Here, we only present initial results on how the ABC scheme cope with such plants. We do not pretend to answer any open question in this field, but rather we try to put a little light on its performance. Fig 7 (right) shows the error when the pretrained but fixed NN tried to control fast changing plant given below,

$$y(k) = \frac{y(k-1)}{1+y^2(k-1)} + (1 - 0.001k) * u^3(k-1).$$

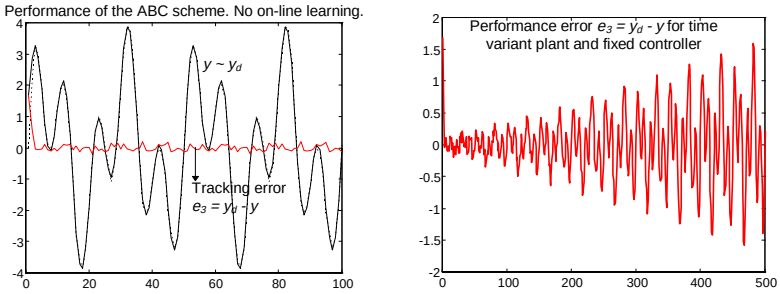

Figure 7 ABC: Perfect tracking in the case of nonlinear monotonic time invariant plant (left). Performance error for fixed pretrained NN controlling the time variant plant (right). (The plant gain is halved in 500 steps).

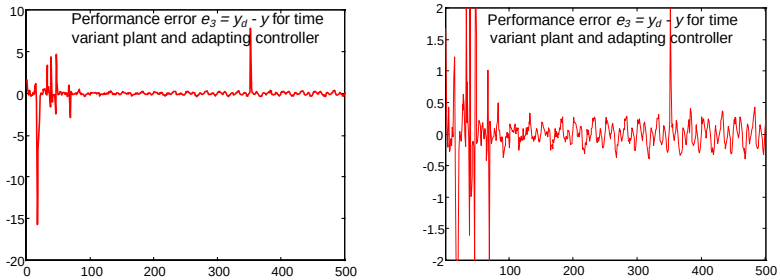

Figure 8 ABC: Performance error at controlling the time variant plant with an on-line adaptation of the NN OL weights. Forgetting factor λ=0.985. (Right graph is drawn in the same scale as Fig 7 right).

This is a model of the plant which halves the plant gain in 500 steps. Without an adaptation the performance error $e_3 = y_d - y$ increases rapidly (Fig 7, right). Fig 8 shows e_3 in the case of the on-line adaptation of neural network. Results are obtained by using a forgetting factor $\lambda = 0.985$. The adaptation and control process is a stable one and, in comparison to the error in Fig 7, the final error in Fig 8 is three times smaller.

4 Conclusions

This paper presented the neural network and fuzzy logic model based control of nonlinear dynamic plants. In particular it was shown that the neural network based adaptive back-through control (ABC) scheme can be successfully applied for control of both linear and non-linear dynamic plants. The principal idea of the ABC, the use of *controller error signal* e_1 instead of an error back propagated delta signal for the learning of the controller, is introduced. Due to this fact the ABC performance seems to be superior to the other NN based adaptive control approaches. For linear plants, the resulting feedforward controller, providing that the order of the plant and plant model are equal, is a perfect adaptive poles-zeros cancel-

ler. Thus, the ABC has a character of a predictive controller. Faced with nonlinear plants the ABC performs as a kind of nonlinear deadbeat controller. We introduced the idea of using only one NN, which should simultaneously act as both the plant model and the model of the plant inverse. In this way we avoided a great deal of a redundancy while training two networks.

References

Garcia, C. E. and M. Morari, 1982. *IEC Proc. Des. Dev.* **21**, 308

Hunt, K.J. and D. Sbarbaro, 1991. *IEE Proc.-D, Vol.* **138**, No. 5, 431-438

Jordan M.I. and Rumelhart D. E., 1992. *Journal of Cognitive Science* **16**, 307-354

Jordan M.I., 1993. *Course 9.641, MIT*, Cambridge, MA

Kecman, V., 1988. *Foundations of Automatic Control, (In Serbocroat),* Skolska knjiga, Zagreb

Kecman, V. and B.-M., Pfeiffer, 1994. *EUFIT '94, Proc., Vol.* **1**, pp. 58-66, Aachen

Kecman, V., 1997. *Report 575, The University of Auckland, NZ*

Kecman, V. and T. Rommel, 1997 *EUFIT '97, Proc.,* Aachen

Narendra, K.S., and K. Parthasarathy, 1991. *IEEE Transactions on Neural Networks* **1**, 4-27.

Psaltis, D., A. Sideris. and A. A. Yamamura, 1988. *IEEE Control System Mag.*, **8**, pp. 17-21

Rommel, T., 1997. *Report No. 97-30 The University of Auckland, NZ*

Saerens, M and A. Soquet, 1991. *IEE Proc.-F,* **138** (1), pp. 55-62

Saerens, M., Renders, J.-M., H. Bersini, 1996. *Chap. 7, In IEEE Press Book on Intelligent Control* Systems, Gupta M. and N. Sinha (eds.), IEEE Computer Society Press

Tsypkin, Ja. Z, 1972. *Fundamentals of Automatic Control Theory, (In Russian), Nauka, Moskva*

Widrow, B. and E., Walach, 1996. *Adaptive Inverse Control, Upper Saddle River, Prentice Hall*

Widrow B. and G.L. Plett, 1996. *Proc. of NICROSP '96, Intl. Workshop on NN for Identif., Control, Robot., and Sign./Image Process.*, Venice, Italy, IEEE 1996 0-8186-7456-3/9