Kecman V., Vlacic Lj., Salman R., ***Learning in and performance of the new neural network based adaptive backthrough control structure,*** Proceedings of the 14th IFAC Triennial World Congress, Beijing, PR China, Vol. K, pp. 133-140, Pergamon, 1999

LEARNING IN AND PERFORMANCE OF THE NEW NEURAL NETWORK BASED ADAPTIVE BACKTHROUGH CONTROL STRUCTURE

Vojislav Kecman[1], LjubišaVlačić[2], Raied Salman[1]

[1]*The University of Auckland, Department of Mechanical Engineering*
Private Bag 92019, Auckland, New Zealand, v.kecman@auckland.ac.nz
[2]*Griffith University, School of Microelectronic Engineering,*
Nathan Campus TB 3.10, Brisbane, Australia

Abstract: This paper presents a new neural network (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 is comprised of one neural network only that simultaneously acts as both - plant model and the controller (plant inverse). For linear noiseless plants, the resulting feedforward controller, for equal orders of the plant and plant model, is a perfect adaptive poles-zeros canceller. In the case of monotonic nonlinear dynamic system, the proposed ABC control represents the nonlinear predictive controller. The ABC scheme is based on the discrete nonlinear (NARMAX) dynamic model. For monotonic nonlinearities, the desired control signal results from the nonlinear optimization procedure with guaranteed convex search function and consequently with a unique solution. *Copyright 1999 IFAC*

Keywords: Neural Networks Based Control; Adaptive Backthrough Control (ABC),

1. INTRODUCTION AND BASIC CONTROL STRUCTURE

This paper focuses on the NN based adaptive control. Due to the 'equality' of NN and fuzzy logic (FL) models (see Kecman and Pfeiffer, 1994) the same approach can be applied to the adaptive FL based control schemes. In particular, after presenting the basic guiding ideas of the 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. See, [Kecman, 1997; Kecman and Rommel, 1997; Rommel, 1997, Salman and Kecman, 1998].

Recently, the area of NN control has been exhaustively investigated and there is a large number of different NN based control methods. A systematic classification of the very different NN control structures is a formidable task indeed [Agarwal, 1997].

So-called 'standard' or 'classic' NN based control uses two neural networks. This control structure comprises NN_2 which represents the (approximate) model of the plant, and 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.

Here proposed *Adaptive Backthrough Control (ABC)*, is in the spirit of 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, 1990 and Widrow and Walach, 1996]. While being 'similar' in spirit, there are few important distinctive features that differ the ABC approach from all the other standard NN based control methods.

First of all, the *ABC uses one neural network only* [Kecman, 1997, Salman and Kecman, 1998]. Second principal feature of the ABC is that, unlike the other approaches, it doesn't use standard training errors as the learning signals for modeling the plant and adapting the controller. (Note that at the ABC approach, the single NN is both the plant model and the controller). Rather, the true desired value y_d (signal that should be

tracked, reference signal) is used for the training of the NN. In this way, the *desired* but unknown control signal u_d, results from the *backward* transformation of the y_d *through* the NN. 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.

Similarly to the adaptive inverse control (AIC) devised by Widrow, ABC control scheme proposed here is effective as long as the plant is stable. 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 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 $G_{ref}(s) = 1$.

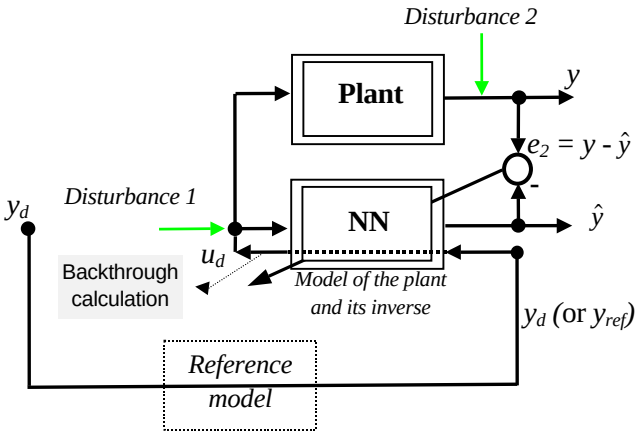


Figure 1 Neural (or fuzzy) network based Adaptive Backthrough Control (ABC) scheme with one network which simultaneously acts as the plant model and as a controller (inverse plant model).

The first papers on the NN based ABC system had described the ABC structure comprised of the two NN [Kecman and Rommel, 1997] and [Rommel, 1997]. This structure was 'inherited' from the previously mentioned approaches and it is directly related to the classic EBP learning. The task of the first network NN_1 was to act as a controller i.e., to learn the inverse dynamics of the controlled plant. The second NN, NN_2 was used as the plant model (needed as the part of the EBP procedure for learning of the NN_1's weights). Being properly trained and after receiving the desired plant output signal y_d, NN_1 was able to produce the best control signal u_d which would drive the plant to output the desired y_d. However, the ABC learning is different from the EBP algorithm. Note that in the ABC algorithm by using the desired trajectory y_d, the best control signal u_d can be calculated directly by the backward step through the single NN. Thus, having the two NN in the control structure, 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 NN_1 is no longer needed. The ABC structure with only one NN, which simultaneously acts as the plant model, and as a controller (inverse plant model) is shown in Fig 1.

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, (1) 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 which 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 2.

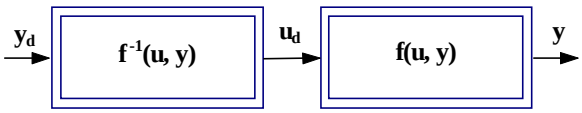


Figure 2 The ideal (feedforward) control structure for any plant.

f(u, y) represented in Fig 2 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.

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,

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

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. Both the identification part and the control part in NN can be given a graphical representation (Fig 3). Note that the two different identification schemes are presented in Fig 3 - *series-parallel* and *parallel.* (The names are due [Landau, 1979]). The identification can be performed by using either the

Series-Parallel scheme

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

or the Parallel one

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

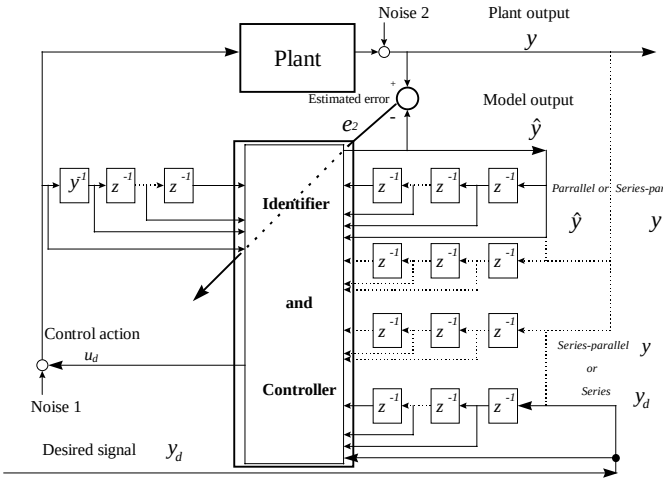


Figure 3 Identification and control scheme using NN.

It is hard to say which identification scheme is better. Narendra and Annaswamy (1989) showed (for linear systems) the series-parallel method to be globally stable. 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. There are also two controller schemes presented in Fig 3: a *series-parallel* and a *series* scheme. The detailed analysis of the performance of both schemes, in the case of the linear plant only, is given in [Salman and Kecman, 1998], where it is shown that under some mild assumptions on the identification part, the *series-parallel control* scheme is superior in the linear case (see the Appendix). The determination of a good inverse model 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 present only the basic ideas and the performance of the ABC scheme.

2. LEARNING IN THE ABC STRUCTURES WHICH COMPRISES ONE NN ONLY

Fig 1 shows the ABC scheme having one NN which acts as both the plant model (emulator) and the controller of the plant. The most important part of the learning is the calculation of the desired control signal u_d. 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 case of *nonlinear plants pretraining of NN is essential.* For the linear plant this pretraining is not 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.

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. For the ABC of **linear systems**, the calculation

of u_d is straightforward. The forward model (NN in Figs 1 and 3) is given as,

$$\hat{y}(k+1) = \sum_{i=1}^{N} w_{2,i} \cdot x_{2,i} = \mathbf{w}_2^T \cdot \mathbf{x}_2 \qquad (6)$$

where $N = 2n$, n is the order of the model and x_2 is an input vector to the NN 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,

$$\hat{u}_d(k) = \left(\frac{y_d(k+1) - w_{2,1} y_d(k) - \cdots - w_{2,n} y_d(k-n+1) - w_{2,n+2} \hat{u}(k-1) - \cdots - w_{2,2n} \hat{u}(k-n+1)}{w_{2,n+1}} \right) \qquad (7)$$

Therefore, when applied to the control of linear systems, the calculation of the control signal u_d by using (7) 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. This is so-called *series* model presented in Fig 3. 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 $y(k), \cdots, y(k-n)$. This is so-called *series-parallel* model presented in Fig 3 which is in the case of linear plant superior to the series control model.

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 inverse mapping of the plant outputs y into its inputs u) control signal u_d can be calculated by an *iterative algorithm* which guarantees finding of proper u_d for any desired y_d. *This is the crucial result in the proposed ABC algorithm.* All the details of the numerical part of the *backthrough* calculation of u_d can be found in [Kecman, 1997].

3. SIMULATIONAL RESULTS

Example 1: Nonlinear *monotonic* 1st order dynamic plant adapted from [Narendra and Parthasarathy, 1990] 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 which 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, 1990]. 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 4 (top) 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.

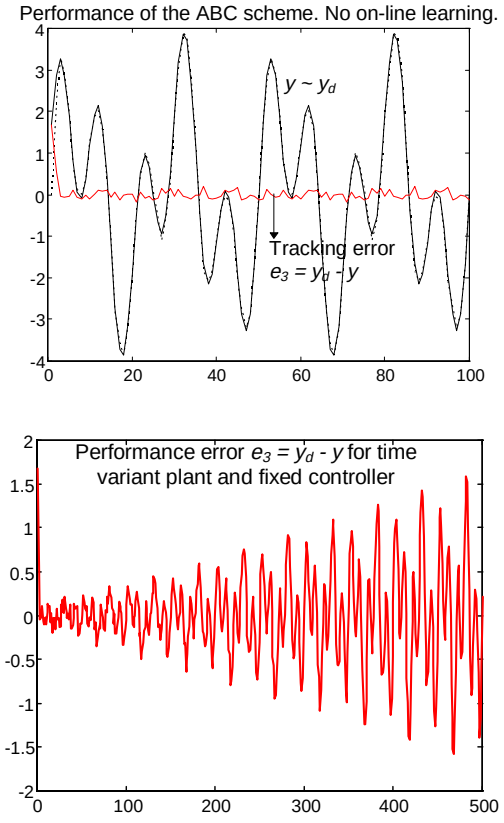


Figure 4 ABC: Perfect tracking in the case of nonlinear monotonic time invariant plant (top). Error for fixed pretrained NN controlling the time variant plant (bottom). The gain is halved in 500 steps.

Fig 4 (bottom) shows the error when the pretrained but fixed NN tried to control fast changing plant given below,

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

This is a model of the plant that halves the plant gain in 500 steps. Without an adaptation the performance error $e_3 = y_d - y$ increases rapidly (Fig 4, bottom). Fig 5 shows error 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 stable and, in comparison to the error in Fig 4, the final error in Fig 5 is three times smaller.

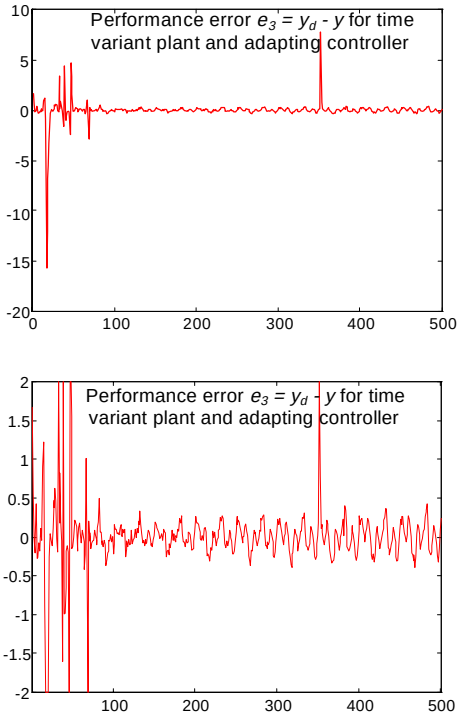


Figure 5 ABC: Performance error at controlling the time variant plant with an on-line adaptation of the NN OL weights. Forgetting factor λ=0.985. Bottom graph is in the same scale as Fig 6 bottom).

Example 2: Consider the ABC of the following nonlinear *non-monotonic* dynamic plant,

$$y_{k+1} = \sin(y_k)*\sin(u_k) - u_k/\pi$$

This plant is an non-monotonic nonlinear function. In other words, there is one-to-many mapping of the u_k to the y_{k+1}. However, the function $y_{k+1} = f(u_k, y_k)$ represents an one-to-one mapping and the ABC can successfully model both the plant dynamics (mapping of u to y) and the plant inverse dynamics (mapping of y to u). The NN was optimized by using a feedforward orthogonal least square method. The basis functions in all neurons are the 2-dimensional Gaussians with the same covariance matrix $\Sigma = diag(0.0735, 0.1815)$. At the beginning of the RBF selection, there were 169 symmetrically placed neurons in hidden layer and at the and 47 centers were chosen. Such a network models the plant very well. (Note that this structure corresponds to the fuzzy logic model with a rule basis comprised of 47 rules).

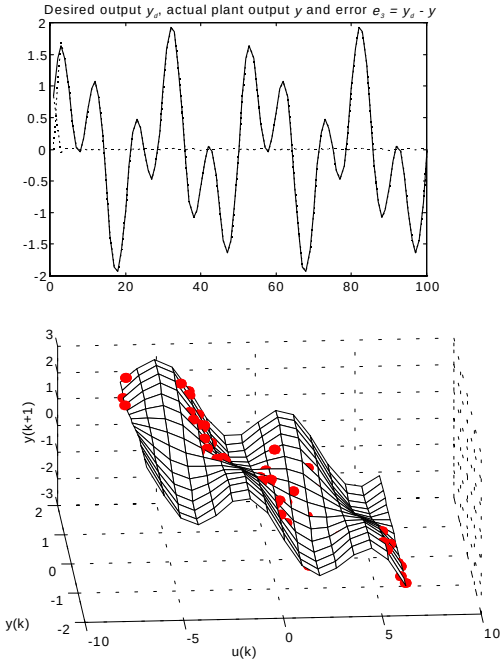
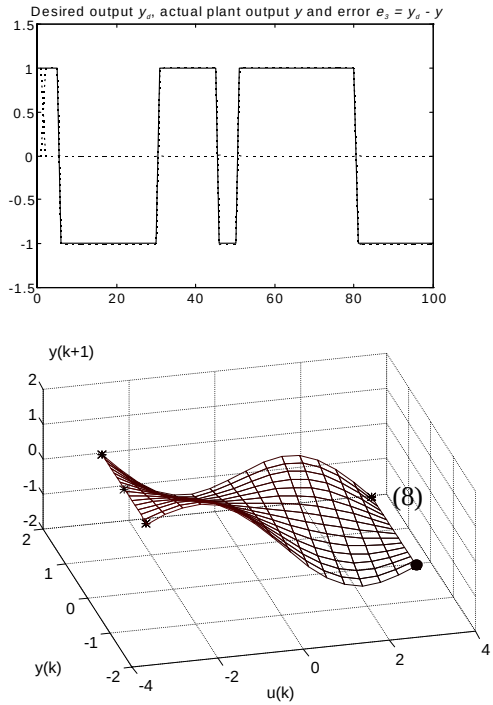


Figure 6 ABC *Top:* Perfect tracking of the desired signal $y_d = sin(2\pi k / 25) + sin(2\pi k / 10)$ for the time invariant plant (8). Pretrained NN weights are fixed. No adaptation. *Bottom:* Trajectory shown by dots lies on the surface described by (8).



(8)

Figure 7 ABC *Top:* Perfect tracking of the desired rectangular signal for the time invariant plant (8). Pretrained NN weights are fixed. No adaptation. *Bottom:* Trajectory shown by dots lies on the surface described by (8).

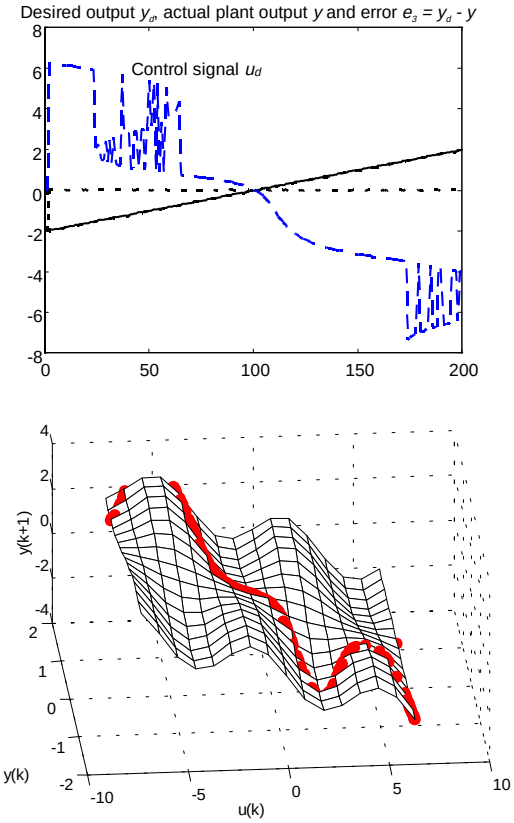


Figure 8 ABC *Top:* Perfect tracking of the desired ramp signal [-2, 2] for the time invariant plant (8) Pretrained NN weights are fixed. No adaptation. *Bottom:* Trajectory shown by dots lies on or 'sneaks through' the surface described by (8).

4. CONCLUSIONS

This paper presents the neural network (or fuzzy logic model) based control of nonlinear dynamic plants. In particular it was shown that the neural network based adaptive backthrough control (ABC) scheme can be successfully applied for control of both linear and nonlinear dynamic plants. We introduced the novel 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. The new learning algorithm based on the NARMAX type of discrete dynamic model is proposed. The ABC performance seems to be superior to the other NN based adaptive control approaches. For the linear plants, the resulting feedforward controller, providing that the order of the plant and of the plant model is equal, is a perfect adaptive poles-zeros canceller. Thus, the ABC has a character of a predictive controller. Faced with nonlinear plants the ABC performs as an nonlinear deadbeat controller. We presented the performance of the ABC algorithm faced with a much more complex task - with a control of the *time variant nonlinear* plant. First simulational results seem to be very encouraging.

5. REFERENCES

Agarwal, M., 1997. A Systematic Classification of Neural-Network-Based Control, IEEE Control Systems, **Vol. 17**, No 2, p.p. 75-93

Garcia, C. E. and M. Morari, 1982. Internal Model Control, 1. A Unifying Review and Some New Results, Ind. Eng. Chem. Process Des. Dev. , **21**, pp. 308-323

Hunt, K. J. and D. Sbarbaro, 1991. Neural networks for nonlinear internal model control, IEE Proc.-D, **Vol. 138**, No. 5, pp. 431-438

Jordan, M. I., 1993. Connectionist Models of Cognitive Processes, Course 9.641, MIT, Cambridge, MA, USA

Jordan, M. I. and Rumelhart, D. E., 1992. Forward models: Supervised Learning with a Distal Teacher, Journal of Cognitive Science **16**, 307-354

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

Kecman, V. and B.-M., Pfeiffer, 1994. Exploiting the structural equivalence of learning fuzzy systems and radial basis function neural networks, EUFIT '94, Proc., **Vol. 1**, pp. 58-66, Aachen, Germany

Kecman, V., 1997. Neural Networks and Fuzzy Logic Based Control, Report 575, The University of Auckland, Auckland, NZ

Kecman, V. and T. Rommel, 1997. Performance of Neural Networks Based Adaptive Backthrough Control, Proc. of ICONIPANZIIS/ANNES, 1997, Dunedin, NZ

Landau, I., 1979. Adaptive Control System: The model reference approach, Marcel Dekker, New York, USA

Narendra, K.S., and K. Parthasarathy, 1990. Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks, **1**, 4-27

Orr, M. J. L., 1996. Regularization in the selection of radial basis function centers, Report, Center of Cognitive Science, University of Edinburgh, UK

Psaltis, D., A. Sideris and A. A. Yamamura, 1988. A multilayered Neural Network Controller, IEEE Control System Magazine, **8**, pp. 17-21

Rommel, T., 1997. Neural networks based adaptive control, Report No. 97-30, The University of Auckland, Auckland, NZ

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

Saerens, M. and A. Soquet, 1991. Neural controllers based on backpropagation algorithm, IEE Proc.-F, **138(1)**, pp. 55-62

Salman, R. and V. Kecman, 1998. Feedforward Action Based on Adaptive Backthrough Control, Proc. of IPENZ '98, Feb 98, Auckland, NZ

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

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

APPENDIX

Controller Design and Performance without Noise

Results presented here are from [Salman and Kecman, 1998]. Note that in the overall ABC scheme we can combine two mentioned identification approaches with two different controller designs. Thus, there are four different design algorithms. Below, and due to the restricted space, we show the performance of the ABC structure for the linear systems without any noise, *assuming perfect* (i.e., bias-free) *identification*.

Series Connection in Controller Design

Here, in the calculation of the control signal *u*, we use the desired output signal y_d and its previous values as shown in Fig 3,

$$u(k)=(y_d(k+1)+\sum_{i=1}^{n}\hat{a}_i y_d(k-i+1)-\sum_{i=2}^{n}\hat{b}_i u(k-i+1))/\hat{b}_1 \quad \text{(A1)}$$

Plant model for the next sample is,

$$y(k+1)+\sum_{i=1}^{n}a_i y(k-i+1)=\sum_{i=1}^{n}b_i u(k-i+1)$$

Substitution of the control action (A1) in the above plant equation gives,

$$y(k+1)+a_1y(k)+...+a_ny(k-n+1)=b_1(y_d(k+1)+\hat{a}_1y_d(k)+...+\hat{a}_ny_d(k-n+1) -\hat{b}_2u(k-1)-...-\hat{b}_nu(k-n+1))/\hat{b}_1+b_2u(k-1)+...+b_nu(k-n+1) \quad \text{(A2)}$$

When the model order of the identifier equals the plant order then as sampling time *k* increases; ($k\to\infty$), the identifier is a bias-free one and $\hat{a}$ converges to *a* ($\hat{a}\to a$ **),** as well as $\hat{b}\to b$. Defining $e(k+1)=y_d(k+1)-y(k+1)$ (A2) becomes,

$$e(k+1)+a_1e(k)+a_2e(k-1)+...+a_ne(k-n+1)=0 \quad \text{(A3)}$$

Since the plant is a stable system i.e., all the poles of the plant will be inside the unit circle, (A3) always converges to zero. The rate of convergence depends on the poles of the system. The more close to the origin they are, the faster convergence occurs.

Series-Parallel Connection in Controller Design

Now, unlike the design procedure above, we use the desired signal y_d and the previous plant outputs signals (Fig 3). In this case (A1) can be written as follows,

$$u(k)=(y_d(k+1)+\sum_{i=1}^{n}\hat{a}_i y(k-i+1)-\sum_{i=2}^{n}\hat{b}_i u(k-i+1))/\hat{b}_1 \quad \text{(A4)}$$

The control signal of (A4) will drive the plant as follows,

$$y(k+1)+a_1y(k)+...+a_ny(k-n+1)=b_1(y_d(k+1)+\hat{a}_1y(k)+...+\hat{a}_ny(k-n+1)-\hat{b}_2u(k-1)-...-\hat{b}_nu(k-n+1))/\hat{b}_1+b_2u(k-1)+...+b_nu(k-n+1) \quad \text{(A5)}$$

Now, in accordance with our assumption about the perfect identification ($\hat{a}=a$, $\hat{b}=b$) (A5) becomes,

$$e(k+1)=0, \quad \text{(A6)}$$

which means that the error at any sampling instant equals zero. Unlike the series connection when the error needs some time to converge to zero, here the error equals zero as soon as the estimated parameters converge to the real one. Series-parallel scheme is superior in the case with noise too, [Salman and Kecman, 1998].

Controller Design and Performance with Noise

Suppose that there exist an additive white noise disturbance having variance σ^2 at the output of the plant due to disturbances of measurements or interfaces, (Noise 2 in Fig 3). Noise 1 in Fig 3 affecting the control signal only is less severe one, since the plant acts as a filter to this type of noise. This is why we will analyze the effect of Noise 2, in both the series and series parallel connections of the controller, below.

Series Connection in Controller Design

For simplicity of derivation let *n*=1, then the plant can be written as,

$$y(k+1)+a_1y(k)=b_1u(k)+\in(k+1) \quad \text{(A7)}$$

where $\in(k+1)$ is the disturbance signal at instant $k+1$. The control action (A1) is,

$$u(k)=(y_d(k+1)+\hat{a}_1y_d(k))/\hat{b}_1. \quad \text{(A8)}$$

The control signal *u(k)* of (A8) will act on the plant model (A7),

$$y(k+1)+a_1y(k)=b_1(y_d(k+1)+\hat{a}_1y_d(k))/\hat{b}_1+\in(k+1) \quad \text{(A9)}$$

Again assuming that the estimated parameters $\hat{a}_1$ and $\hat{b}_1$ converge to the true ones, i.e. $\hat{a}_1 = a_1$ and $\hat{b}_1 = b_1$ we obtain,

$$y(k+1) + a_1 y(k) = y_d(k+1) + \hat{a}_1 y_d(k) + \in(k+1)$$

Let $e(k+1) = y_d(k+1) - y(k+1)$ then,

$$e(k+1) + a_1 e(k) = -\in(k+1). \qquad \text{(A10)}$$

For $k = 0$, $k = 1$ and $k = 2$ it follows,

$$e(1) = -\in(1) - a_1 e(0),$$
$$e(2) = -\in(2) + a_1 \in(1) + a_1^2 e(0),$$
$$e(3) = -\in(3) + a_1 \in(2) - a_1^2 \in(1) - a_1^3 e(0).$$

The coefficient a_1 plays an important role in this algorithm. For $|a_1| \geq 1$ the error goes to the infinity and system shows unstable behavior. If $|a_1| < 1$, then the error at step k = 2 is affected mainly by the noise at the same step, because the noise signals at step $k = 1$ and $k = 0$ converge to zero. This is also true for higher order systems when there are many coefficients $a_1, a_2, \ldots.$, supposing that $|a_1| < 1$, where $i = 1,2,\ldots,n$ and n is the order of the system. For the first three steps, the above derivation may be arranged in the following matrix form;

$$\begin{bmatrix} e(1) \\ e(2) \\ e(3) \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ a_1 & -1 & 0 \\ -a_1^2 & a_1 & -1 \end{bmatrix} \begin{bmatrix} \in(1) \\ \in(2) \\ \in(3) \end{bmatrix} + e(0) \begin{bmatrix} -a_1 \\ a_1^2 \\ -a_1^3 \end{bmatrix} \qquad \text{(A11)}$$

The above conclusion is similar as the one in the study of the effect of the noise in the Internal Model Control (IMC) structure of [Garcia et al, 1982].

Series-Parallel Connection in Controller Design

For the first order system the plant output will be the same as in (A7), but the control action, (A8), can be written as,

$$u(k) = (y_d(k+1) + \hat{a}_1 y(k)) / \hat{b}_1 \qquad \text{(A12)}$$

Substitute (A12) in (A7) gives,

$$y(k+1) + a_1 y(k) = b_1 (y_d(k+1) + \hat{a}_1 y(k)) / \hat{b}_1 + \in(k+1)$$

if $\hat{a}_1 = a_1$ and $\hat{b}_0 = b_0$ then,

$$(y_d(k+1) - y(k+1)) = -\in(k+1) \qquad \text{(A13)}$$

i.e.,

$$e(k+1) = -\in(k+1) \qquad \text{(A14)}$$

At any sample, the output error is affected only by the noise at this sample irrespective of any previous noise. Equation (A14) applies for any order, assuming that the orders of the plant and the identifier are the same. The above structure looks better than the IMC structure [Garcia et al 1982] in dealing with noise because just the noise at step *k+1* is affecting the output.