Radial Basis Functions Neural Network Based Self-Tuning Regulator

SYED SAAD AZHAR ALI , HUSSAIN AL-DUWAISH and MUHAMMAD MOINUDDIN
Electrical Engineering Department
King Fahd University of Petroleum & Minerals
Dhahran-31261, Saudi Arabia
saadali, hduwaish, moinudin@kfupm.edu.sa http://www.kfupm.edu.sa

Abstract: In this paper a new technique is proposed to design an online control algorithm using the Radial Basis Functions Neural Network (RBFNN). The controller is an RBFNN based direct self-tuning regulator (STR) that overcomes several shortcomings of the inverse control design using the neural networks. The control algorithm performs equally good to both minimum phase and non-minimum phase plants. The plant parameters are estimated online and are used to update the weights of the RBFNN. The weight update equations are derived based on the well known least mean squares principle. The RBFNN virtually models the inverse of the plant and thus the output tracks the reference trajectory. The proposed algorithm is successfully verified using simulations for both minimum and non-minimum phase plants.

Keywords: STR, RBFNN, Adaptive Control, LMS, ARMA

1 Introduction

Adaptive control schemes are used for the control of plants, where the parameters of the plant are not known exactly or slowly time varying [1], [2]. Adaptive control methods were developed as an attempt to overcome difficulties connected with the ignorance of system structure and critical parameter values as well as changing control regimes [1], [3]. Different approaches have been adopted to design the adaptive controllers.

The *self-tuning regulator* attempts to automate the tasks involved in the adaptive control scheme namely modeling, design of a control law, implementation, and validation. This is illustrated in Fig. 1.

The parameters of the process model are estimated online by a recursive estimator. The controller design is performed under the specified conditions along with selected process parameters needed depending on the controller type. The controller normally acts as the plant inverse resulting in producing the desired output. The alternate to this scheme could be the adaptive inverse control. In this scheme, the inverse of the plant is calculated or modelled separately and then a copy the inverse is cascaded with the actual plant. This is an indirect technique requiring additional computations and extra blocks in addition to the note that inverse of a non-minimum phase plant maybe difficult to handle.

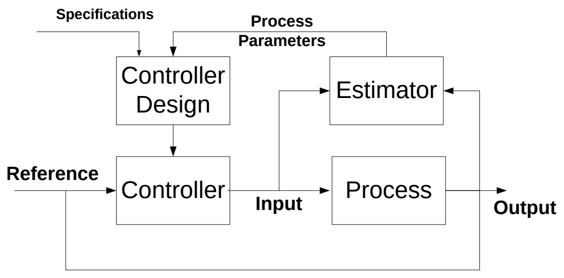

Figure 1: Block diagram of a self-tuning regulator

Neural Network (NN) has been used in the controller structures and it has been proven that these control methods show excellent performance even for nonlinear plants [4]. Various NN based controllers for several adaptive and non-adaptive schemes have been proposed [5, 3, 6, 7, 8, 9]. The methods developed for neuro-adaptive tracking, neuro-adaptive model reference control and neuro-self tuning regulators the plant is considered to be minimum phase which also is a standard assumption in the adaptive control schemes [10, 11]. NN based controllers have been proven more robust and fault tolerant compared to classical adaptive controllers on the cost of slow learning and more complex structures.

In this paper a new technique is proposed that overcomes several shortcomings of using the inverse control and NN. An online control algorithm is structured using the Radial Basis Functions Neural Network (RBFNN). The plant can be either minimum phase or non-minimum phase. The plant parameters are estimated online and are used to update the weights of the RBFNN. The weight update equations are derived based on the well known least mean squares principle. The RBFNN virtually models the inverse of the plant and thus the output tracks the reference trajectory. This scheme is a direct STR that estimates the plant parameters and simultaneously incorporates them in training the RBFNN, outperforming other indirect approaches. In addition RBFNN is known to be a better choice among the NNs compared to the multi-layered feedforward neural networks that are trained using the back-propagation algorithm. RBFNN bears the same universal approximation capabilities as of MFNN, despite the fact that it has only one layer, resulting in a reduced training time in contrast to the use of back-propagation for MFNN [10, 11, 12, 13].

2 Proposed Structure

The proposed structure shown in Fig 2 comprises of the estimator block that runs in parallel to the actual process as a black-box identifier. Auto-regressive moving average (ARMA) is used to model the plant. The estimates are updated online for any changes in real-time. These estimates are fed to the weight update block for the RBFNN. In effect only the zeroes of the process are needed by the weight update block to train the RBFNN. The RBFNN thus adapts itself to act as the inverse of the process.

2.1 Radial Basis Functions Neural Networks

A SISO RBFNN is shown in Fig. 3. It consists of an input node $r(t)$, a hidden layer with n_o neurons and an output node $x(t)$. Each of the input node is

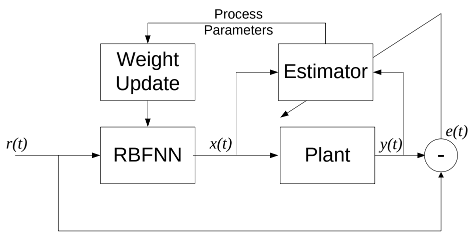

Figure 2: Proposed self-tuning regulator structure

connected to all the nodes in the hidden layer through unity weights (direct connection). While each of the hidden layer nodes is connected to the output node through some weights $w_1, \ldots, w_{n_o}$.

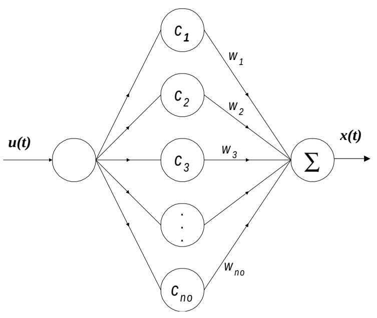

Figure 3: A general RBF network

Each neuron finds the distance, normally applying Euclidean norm, between the input and its centre and passes the resulting scalar through a nonlinearity. So the output of the hidden neuron is given by $\phi(\|r(t) - c_i\|)$, where n_o is the number of hidden layer nodes (neuron), $r(t)$ is the input, c_i is the centre of i^{th} hidden layer node where $i = 1, 2, \ldots, n_o$, and $\phi(\cdot)$ is the nonlinear basis function. Normally this function is taken as a Gaussian function of width β. The output $(x(t))$ is a weighted sum of the outputs of the hidden layer, given by

$$\begin{aligned} x(t) &= W\Phi(t), \\ x(t) &= \sum_{i=1}^{n_o} w_i \phi(\|r(t) - c_i\|), \end{aligned}$$

$$\begin{aligned} where \qquad W &= [w_1 \, w_2 \, \ldots \, w_{n_o}], \\ and \quad \Phi(t) &= [\|r(t) - c_1\| \; \ldots \; \|r(t) - c_{n_o}\|]^T. \end{aligned}$$

$x(t)$ is the output, w_i is the weight corresponding to the i^{th} hidden neuron.

2.2 ARMA Model

The plant is modeled by an ARMA model, whose output is given by

$$y(t) = \sum_{i=1}^{n} a_i y(t-i) + \sum_{j=0}^{m} b_j x(t-j), \qquad (1)$$

or in terms of q^{-1} operator

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})} q^{-d} x(t). \qquad (2)$$

3 Training Algorithm for the RBFNN based STR

Considering Fig 2, the objective is to develop a recursive algorithm by which the parameters of the ARMA model and the weights of the RBFNN can be adjusted, so that the RBFNN replicates the inverse of the plant. The training algorithm is developed based on LMS principle. The parameters (weights of RBFNN and the coefficients of ARMA) are updated by minimizing the performance index I given by,

$$\begin{aligned} I &= \frac{1}{2} e^2(t), \\ e(t) &= r(t) - y(t), \end{aligned} \qquad (3)$$

where $r(t)$ is the reference input signal and $y(t)$ is the actual output of the the plant. The coefficients of the ARMA model and the weights of the RBFNN are updated in the negative direction of the gradient as,

$$\theta(K+1) = \theta(K) - \alpha \frac{\partial I}{\partial \theta(K)}, \qquad (4)$$

and

$$W(K+1) = W(K) - \alpha \frac{\partial I}{\partial W(K)}, \qquad (5)$$

where $\theta = [a_1 \; \ldots \; a_n \; b_o \; \ldots \; b_m]$ is the parameter vector , $W = [w_1 \; w_2 \; \ldots \; w_{n_o}]$ is the weight vector for RBFNN and α is the learning parameter. The variable K is used to show the iteration number of training.

Keeping the regressions of the variables in the system in a regression vector ψ as $\psi(t) = [y(t-1) \; \ldots \; y(t-n) \; x(t-d) \; \ldots \; x(t-m-d)]$ and finding partial derivatives.

$$\begin{aligned}
\frac{\partial I}{\partial \theta} &= \frac{1}{2} \frac{\partial e^2(t)}{\partial \theta}, \\
&= e(t) \frac{\partial}{\partial \theta} \left(r(t) - y(t) \right), \\
&= e(t) \frac{\partial}{\partial \theta} \left(r(t) - \frac{B(q^{-1})}{A(q^{-1})} q^{-d} x(t) \right), \\
&= e(t) \frac{\partial}{\partial \theta} (r(t) - \left(a_1 q^{-1} + \ldots + a_n q^{-n} \right) y(t) \\
&\qquad - \left(b_o + \ldots + b_m q^{-m} \right) q^{-d} x(t)), \\
&= -e(t) \frac{\partial}{\partial \theta} (a_1 y(t-1) + \ldots + a_n y(t-n) + \\
&\qquad b_o x(t-d) \ldots + b_m x(t-m-d)), \\
&= -e(t) [y(t-1) \; y(t-2) \; \ldots \; y(t-n) \\
&\qquad x(t-d) \; \ldots \; x(t-m-d)], \\
\frac{\partial I}{\partial \theta} &= -e(t) \psi(t).
\end{aligned} \qquad (6)$$

Now, the gradient in Eq. 6 is used to find the updated parameters at $(K+1)^{th}$ instant along with Eq. 4. The final parameter update equation will be,

$$\theta(K+1) = \theta(K) + \; \alpha \, e(t) \psi(t). \qquad (7)$$

The partial derivatives for the weights are derived as follows,

$$\begin{aligned}
\frac{\partial I}{\partial W} &= \frac{1}{2} \frac{\partial e^2(t)}{\partial W}, \\
&= e(t) \frac{\partial}{\partial W} \left(r(t) - y(t) \right), \\
&= e(t) \frac{\partial}{\partial W} \left(r(t) - \frac{B(q^{-1})}{A(q^{-1})} q^{-d} x(t) \right), \\
&= e(t) \frac{\partial}{\partial W} (r(t) - \left(a_1 q^{-1} + \ldots + a_n q^{-n} \right) y(t) \\
&\qquad - \left(b_o + b_1 q^{-1} + \ldots + b_m q^{-m} \right) q^{-d} W \Phi(t)), \\
&= -e(t) \left(b_o + b_1 q^{-1} + \ldots + b_m q^{-m} \right) q^{-d} \Phi(t), \\
\frac{\partial I}{\partial W} &= -e(t) B(q^{-1}) q^{-d} \Phi(t).
\end{aligned}$$

This gradient is used to find the weight update equation, along with Eq. 5,

$$W(K+1) = W(K) + \; \alpha \, e(t) B(q^{-1}) q^{-d} \Phi(t). \qquad (8)$$

The classical control techniques require the cancellation of the plant's poles and zeros, with the reservation of minimum phase plants only, since cancelling an unstable zero of a non-minimum phase plant introduces unstability in the controller itself. In the proposed technique the inverse of the plant is not obtained by another filter that might be the unstable inverse of the non-minimum phase plant, but modelled by RBFNN, that only requires the estimate of the zeroes of the plant. This can also be seen in the weight update Eq. 8 for the RBFNN. This contributes towards the relaxation of the restriction of having minimum phase plants only.

4 Simulation Results

The proposed STR structure is tested for both minimum and non-minimum phase plants. The plant parameters are estimated online and are used to update the weights for the RBFNN at the same instant.

4.1 Minimum Phase Plant

The minimum phase plant considered is,

$$H(z) = \frac{z^{-1}(0.1065 + .0902z^{-1})}{(1 - z^{-1} + 0.25z^{-2})}. \qquad (9)$$

This plant has one zero at -0.8469, two poles both at 0.5 with a unit. A normally distributed additive noise with a variance of 0.01. A square wave is used as reference input. The RBFNN is composed of just 2 neurons in the input space with a width β of 0.6.

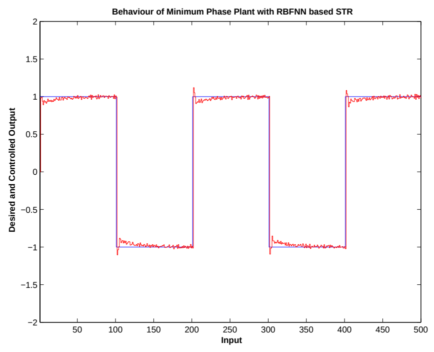

Figure 4: Tracking trajectory for the Minimum Phase system

Fig. 4 shows the efficient response of the RBFNN to the square input. Fig. 5 shows the error signal between the reference and the RBFNN output. The high spikes in this signal show the transition error at every changing edge, otherwise the error is very small.

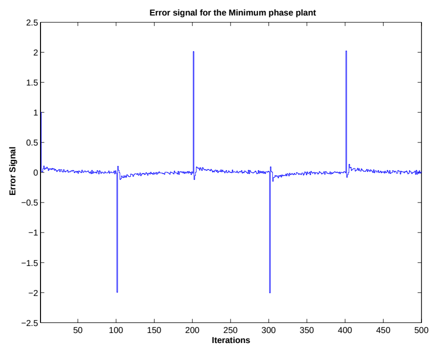

Figure 5: The error signal

4.2 Non-minimum Phase Plant

In this example a non-minimum phase plant is considered given by,

$$H(z) = \frac{z^{-1}(1 + 2z^{-1})}{(1 - z^{-1} + 0.25z^{-2})}. \qquad (10)$$

This plant has one zero at -2 (non-minimum phase), two poles both at 0.5 with a unit delay. The input and the noise characteristics are kept the same as in the example for minimum phase system. The same RBFNN is used to show the consistency for the minimum and non-minimum phase systems.

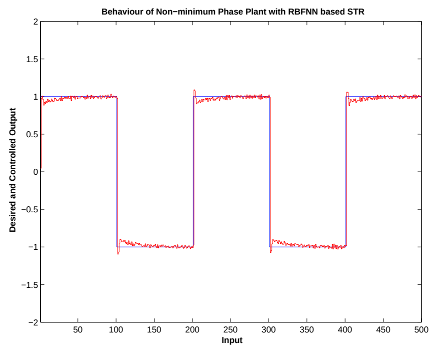

Figure 6: Tracking trajectory for the Non-minimum Phase system

Fig. 6 shows the efficient response of the RBFNN to the square input. Fig. 7 depicts the error signal in the reference and the RBFNN output.

The network size and training speed are also a salient feature for the proposed structure. There are only 2 neurons used for both the examples as compared to the adaptive inverse control using MFNN in [14] that uses the network twice having 4 neurons.

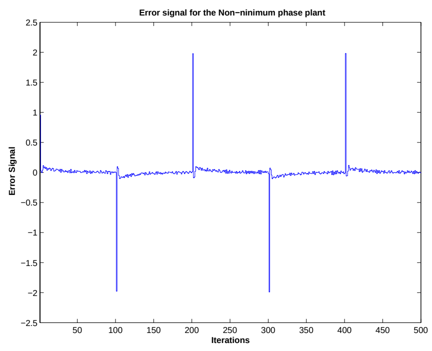

Figure 7: The error signal

5 Conclusions

A new method of adaptive control is introduced. The new method is direct STR type and makes use of the RBFNN. Training algorithm is developed for the new STR. The proposed method is a very simple structure that updates itself online. The exact model of the plant need not to be known and just the estimates are enough to drive the RBFNN as the process inverse. The restrictions for the non-minimum phase and unstable plant are relaxed, thus outperforming others of its kind. Simulation results depict satisfactory tracking behaviour for both minimum and non-minimum phase plants.

6 Acknowledgements

The authors would like to acknowledge the support of King Fahd University of Petroleum & minerals.

References

[1] K. J. Astrom and B. Wittenmark. *Adaptive Control*. Addison Wesley, 1995.

[2] K. J. Astrom and B. Wittenmark. Self-tuning controller based on pole-zero placement. *IEE Proc. D*, 120:120–130, 1980.

[3] T. Yahagi and J. Lu. On self-tuning control of non-minimum phase discrete time systems using approximate inverse systems. *Journal of Dynamic Systems, Measurement, and Control*, 115:12–18, 1993.

[4] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical system using neural networks. *IEEE Transactions on Neural Networks*, 1:4–27, 1990.

[5] S. Omatu, M. Khalid, and R. Yusof. Neuro-control and its application. *Springer-Verlag, London*, 1, 1996.

[6] M. Norgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen. Neural networks for modeling and control of dynamic systems. 2000.

[7] D. H. Nguyen and B. B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, 10:18–23, 1990.

[8] S. Akhyar and S. Omatsu. Self-tuning pid control by neural networks. *IJC on Neural Networks*, 1:2749–2752, 1993.

[9] R. Carelli and D. Camacho, E. F.and Patino. A neural network based feedforward adaptive controller for robots. *IEEE transactions on Systems, Man and Cybernetics*, 25:1281–1288, 1995.

[10] S. Haykin. *Neural Networks:A Comprehensive Foundation II*. Macmillan/IEEE Press, 1994, 1999.

[11] W. L. Jun-Dong, P. Jones. Comparison of neural network classifiers for nscat sea ice flag. *IEEE International Geoscience and remote sensing symposium proceedings.*, pages 2237–2239, 1998.

[12] L. Fortuna, G. Muscato, and M. G. Xibilia. A comparison between hmlp and hrbf for attitude control. *IEEE transactions on neural networks*, 122:318–328, March 2001.

[13] R. A. Finan, A. T. Sapeluk, and R. I. Damper. Comparison of multilayer and radial basis functions neural networks for the text dependent speaker recognition. *IEEE International conference on Neural Networks*, 4:1992–1997, 1996.

[14] Shafiq M. and Moinuddin M. Adaptive inverse control using multi layer perceptron neural network. *Modeling, Identification and Control.*, pages 595–599, 2003.