Direct MRAC with Dynamically Constructed Neural Controllers

Yakov Frayman[1] and Lipo Wang[2]
[1]Deakin University, School of Computing and Mathematics,
662 Blackburn Road, Clayton, Victoria 3168, Australia
[2]Nanyang Technological University, School of Electrical and Electronic Engineering,
Block S2 Nanyang Avenue, Singapore 639798
E-mail: yfraym@deakin.edu.au, elpwang@ntu.edu.sg

Abstract

Research in neural control mostly concentrates on indirect control schemes while insufficient attention has been paid to direct model reference adaptive control (MRAC) scheme. In addition, at present the emphasis of neural control is on parameter tuning instead of structural tuning, i.e., to find the minimal controller capable of achieving an optimal performance. The stability of the neural control schemes (i.e. the requirement of persistency of excitation and bounded learning rates) also requires more attention. Furthermore, localized architectures are needed in order to deal with the moving target problem (i.e. the difficulty for global neural networks to perform several separate computational tasks in closed-loop control). The purpose of the present paper is to show that direct MRAC using dynamically constructed neural controllers, such as the fuzzy neural and the cascade correlation, satisfy above requirements and offers a method for automatic discovery of an efficient controller.

1. Control Problem Statement

A general multi-input multi-output (MIMO) nonlinear dynamical process can be represented by the following state-space representation

$$\vec{x}(t+1) = \vec{f}[\vec{x}(t), \vec{u}(t), \vec{d}(t)] \ , \qquad (1)$$

$$\vec{y}(t) = \vec{g}[\vec{x}(t), n(t)] \ . \qquad (2)$$

Here $\vec{u} \equiv \{u_1, u_2, .., u_{mu}\}$, $\vec{y} \equiv \{y_1, y_2, .., y_{my}\}$ and $\vec{d} \equiv \{d_1, d_2, .., d_p\}$ are the control signal, process output and disturbance input vectors, respectively, $\vec{x} \equiv \{x_1, x_2, .., x_n\}$ are the process states, and $\vec{n} \equiv \{n_1, n_2, .., n_m\}$ is the measurement noise, and t is the time.

The vector maps $\vec{f}$ and $\vec{g}$ (i.e. the process model) are assumed to be unknown. The aim is to dynamically construct satisfactory controller using only the measurements of the process states and outputs, and the desired performance specification in form of a reference model. The desired output response $\vec{y}_d$ is obtained from the output of the reference model using the state variables $\vec{x}$ as inputs to this model. The objective here is to satisfy the persistency of excitation condition for the exponential stability of an adaptive algorithm. The excitation in such case is due to actual process signals (states) affected by the disturbance signals $\vec{d}$ and noise $\vec{n}$ (see also [14] for discussion on injection of persistently excited signals in closed-loop control). The objective function to minimize is the squared difference between the outputs of the process $\vec{y}$ and the reference model $\vec{y}_d$. On the other hand, the use of a squared difference between the desired output setpoint $\vec{y}^{sp}$ and the process output $\vec{y}$ as the objective function can result in bursting phenomena (parameters drift), since setpoints are not persistently exciting [1].

A reference model in the form of a filter with a desired transfer function of the process is used to ensure a variance in the desired dynamic characteristics of the control system. Such reference models can be used in order to achieve the required performance of the process by the change in the process transfer function (optimization-based design using Modulus Optimum method [8]). A linear stable reference model (Butterworth's filter [2]) is used. The coefficients of Butterworth's filter are selected to correspond to Modulus Optimum criteria for the desired performance in terms of standard control objectives such as overshoot, settling time, and steady-state error. The controller is designed to transfer the original process transfer function to a desired one.

A block diagram of the overall neural control system is presented in Fig. 1. To use neural networks for control, the effectiveness of the structural tuning as opposed to the parameter tuning needs to be discussed. The structural tuning of the fuzzy neural controller

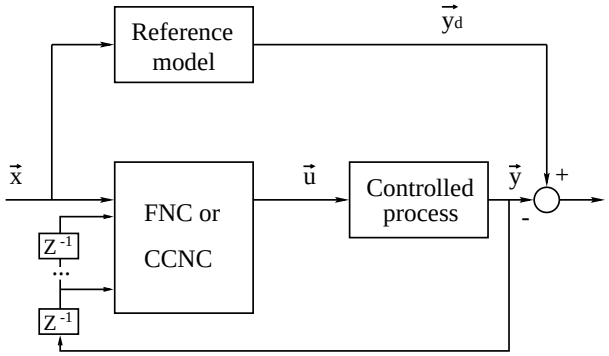

Fig. 1: Block diagram of the overall control system.

(FNC) concerns the partition of variables universe of discourse (the number of membership functions) and the number of rule nodes. The parameter tuning of the FNC concerns adjustments of membership functions (fuzzy weights) and the output weights. Similarly, for the cascade correlation neural controller (CCNC) the structural tuning concerns the size of the hidden layer, whereas the parameter tuning concerns the weight adjustments. Most of the existing efforts in neural control concentrate on parameter tuning selecting the structure on a trial-and-error basis. At the same time, insufficient efforts have been made concerning structural tuning, i.e., to find the minimal controller structure capable of achieving an optimal performance. However, there are no simple ways to determine in advance the minimal size of the partition of the universe of discourse or the minimal size of the hidden layer necessary to achieve the desired performance. This type of structure selection may require in-depth knowledge about the underlying nonlinear process that is rarely available.

Dynamic construction of the controller is also necessary in order to guard the network against too much flexibility and to reduce the risk of over-fitting. The large networks are able to generalize as well as the small ones for problems where the data is uniformly distributed. However, for the cases where the data samples are concentrated within a small area of the mapping space, large networks may fall to generalize at all, especially for continuous data [10]. In closed-loop control data samples are often concentrated in the small region of the domain for extended periods due to constrains imposed by process dynamics and the performance specification [7], consequently small networks are needed.

Such data fixation can also have negative effects in case of a global learning. If a parameter with a global effect on the input/output mapping is repeatedly adjusted to correct the mapping in one input region, this may result in deterioration of the mapping in other regions, and effectively erase the effects of previous learning [7]. A manifestation of this problem is the *moving target problem* [6], where the need for the global sigmoidal network to perform two (or more) separate computational tasks can result in unsatisfactory performance and long training time. Consequently, spatially localized architectures where the learning in one part of the domain has marginal effects on the knowledge in other parts are needed [7]. Several networks, including cerebellar model articulation controller (CMAC), radial basis function networks (RBF), and the FNC have such spatially localized characteristic. The FNC produces outputs for inputs that partially match rules created during initialization and training. This partial matching ensures that similar inputs produce similar outputs whereas dissimilar inputs produce independent outputs. Thus, the network generalizes locally [3].

2. Fuzzy Neural Controller

The fuzzy neural controller (FNC) has an input layer, a rule layer, and an output layer. Input nodes represent input variables consisting of the current network inputs (process states) and the previous outputs of the process. Such outer recurrent feedback loop provides the possibility to include temporal information [15]. Input nodes are connected to rule nodes for these inputs through membership functions (fuzzy weights). Piecewise-linear triangular membership functions that correspond to second-order B-splines [3] are used.

Rule node i represents fuzzy rule ($i = 1, 2, ..., r$):

$$\begin{aligned}
&\text{IF } x_1(t) \text{ is } A^i_{x_1} \text{ AND } ... \ x_n(t) \text{ is } A^i_{x_n} \\
&\text{AND } y_1(t-1) \text{ is } A^i_{y_{11}} \text{ AND } ... \ y_{m1}(t-1) \text{ is } A^i_{y_m} \\
&\quad ... \\
&\text{AND } y_1(t-z) \text{ is } A^i_{y_{1z}} \text{ AND } ... \ y_{mz}(t-z) \text{ is } A^i_{y_{mz}} \\
&\quad ... \\
&\text{AND } y_1(t-c) \text{ is } A^i_{y_{1c}} \text{ AND } ... \ y_{mc}(t-c) \text{ is } A^i_{y_{mc}} \\
&\text{THEN } u_1(t) = w^i_1 \ , \ ... \ , \ u_m(t) = w^i_m \ . \qquad (3)
\end{aligned}$$

Here w^i_l is the weight connecting rule node i and output node l. A^i_q ($q = x_1, .., x_n, y_{11}, .., y_{mc}$) is the membership function of the antecedent part of rule node i for input node q, and z ($z = 1, 2, ..., c$) is the time delay.

Rule nodes are connected to input and output nodes for these rules. The membership value μ_i of the premise of the ith rule that indicates the degree to which the compound antecedent of the rule is satisfied, is calculated as fuzzy AND using the product operator. The

use of the product operator makes fuzzy inference to be fully differentiable at any point [3]. In the output layer, nodes receive inputs from rule nodes connected to these output nodes. Output u_l of the FNC is obtained using the weighted average (or center of gravity defuzzification). The use of the weighted average allows us to avoid problems with over-aggressive control [13]. In addition, it produces a smoother output than the mean of maxima (MOM) defuzzification method and greatly reduces both the computational cost and the storage requirement of the algorithm [3].

The FNC structure generation and parameter tuning algorithms are as follows. We need to specify the allowable error threshold and/or the maximum number of rules (rule nodes) for learning to stop. Initially two equally spaced input membership functions are added along the operating range of each input variable. In such a way these membership functions satisfy ϵ-completeness. The initial rule layer is created using Eq. (3). The network is trained using the following learning rules:

$$w_l^i(t+1) = w_l^i(t) - \eta \frac{\partial \varepsilon_l}{\partial w_l^i} \ . \tag{4}$$

for adaptation of the weights between the rule layer and the output layer, and

$$A_q^i(t+1) = A_q^i(t) - \eta \frac{\partial \varepsilon_l}{\partial A_q^i} \ . \tag{5}$$

for adaptation of membership functions (fuzzy weights) between the input layer and the rule layer. Here η is a learning rate.

A variable learning rate is used that starts with a relatively large learning rate to enhance the learning speed. Whenever the error starts increasing, the learning rate is reduced. The gradually decreasing learning rate is aimed on preventing instability problem [15] resulting from over-aggressive control. In addition, for incremental learning the learning rate is bounded by stability requirements of the stochastic approximation theory. Accordingly, the learning rate must be small and diminishing in order to maintain stability of the learning algorithm [9].

If the degree of overlapping of membership functions is greater than a pre-specified threshold, we combine these membership functions using the *fuzzy similarity measure* [5]. We can thus reduce the size of the rule base that is necessary in order to protect the network from the 'curse of dimensionality'. Combining of the membership functions is also done to eliminate the underperforming membership functions, and to replace them with the new ones that are likely to perform better. If the performance of the controller is below the requirements, an additional membership function is added for all inputs at the point of the maximum error of the process output. By firstly eliminating the errors whose deviation from the target values is the greatest, the output errors can be reduced more efficiently and the convergence of the network can be improved. Next, the rule base is updated and the process is repeated until either we obtained a satisfactory performance, or the maximum pre-specified size of the network is exceeded.

3. Cascade Correlation Neural Controller

The Cascade Correlation learning algorithm [6] (or CCNC in our case) combines the cascade architecture, where hidden nodes are added to the network one at a time and do not change afterwards, and the structure learning algorithm that creates and installs new hidden nodes. The algorithm starts with a minimal network consisting only of an input layer and an output layer. All connections to the output layer are trained until the overall error of the network no longer decreases. If the network performance satisfies a prescribed accuracy target, the algorithm stops. In such case, as there is no hidden layer, the problem at hand is linear.

Remark. CCNC can thus be used to test if the problem at hand is *really* a nonlinear one. As pointed out by Mars *et al.* [11], there is no benefit in applying neural network to a linear or linearizable process, as this will result in degradation of network performance: the solution should not be more complex that the problem at hand. Same argument was raised earlier by Saridis [12] for control with self-organized and learning systems.

If the network performance is not satisfactory (and therefore the problem is really a nonlinear one), generate candidate nodes. Candidate nodes receive trainable connections from input nodes and from existing hidden nodes. The correlation between the activation of candidate nodes and the residual error of the network is maximized by training links to candidate nodes. The training stops when the correlation no longer improves. The candidate node with the maximum correlation is selected and changed into a hidden node by connecting it to output nodes. The algorithm is repeated until the overall error of the network falls below a pre-specified threshold.

4. Simulation Results

The performance of the dynamically constructed FNC and CCNC was evaluated by numerical simulations on

the polymerization process used to produce polymers from monomers in a continuous stirred tank polymerization reactor (CSTR) (Fig. 2). A free-radical poly-

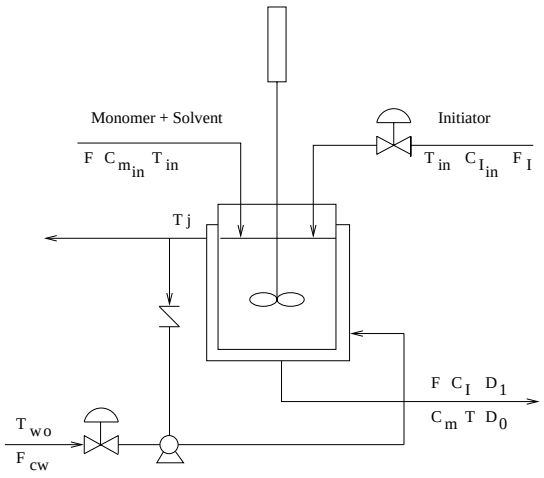

Fig. 2: Continuous stirred tank polymerization reactor. Here F, F_I, and F_{cw} are the volumetric flow rates of inlet and outlet streams, the inlet initiator stream, and the cooling water, respectively. $C_{m_{in}}$, C_m, $C_{I_{in}}$, and C_I, are the molar concentrations of monomer and initiator in inlet and outlet streams, respectively. T_{in}, T_{w_O}, T, and T_j are temperatures of the inlet streams, inlet coolant stream, the reactor, and the cooling jacket, respectively. D_1 and D_O are mass and molar concentrations of dead polymer chains, respectively.

merization of methyl methacrylate (MMA) (monomer) with azo-bis-isobutyronitrile (AIBN) as initiator and toluene as solvent is used. The dynamic behavior of the process is described by the mass and energy balances as a set of six ordinary differential equations. Consult [4] for the ordinary differential equations involved, kinetic data, physical parameters, and steady-state values used, and the assumptions made. For the CSTR, the dimensionless state variables are defined as follows: $x_1 = C_m, x_2 = C_I, x_3 = T, x_4 = D_0, x_5 = D_I, x_6 = T_j$ (Fig. 2). The control of a CSTR requires the number-average molecular weight (NAMW) $y_1 = D_I/D_0$ and the reactor temperature $y_2 = T$ to be regulated. This is achieved by manipulating the volumetric flow rate of initiator in the inlet stream $u_1 = F_I$ and the volumetric flow rate of the cooling water $u_2 = F_{cw}$. The process disturbances are the molar concentration of monomer in the inlet stream $d_1 = C_{m_{in}}$ and the temperature of the inlet stream $d_2 = T_{in}$.

As a reference model Butterworth's characteristic equation [2] for the 5th order system was used:

$$s^5 + 3.24\omega_n s^4 + 5.24\omega_n^2 s^3 + 5.24\omega_n^3 s^2 + 3.24\omega_n^4 s + \omega_n^5 \quad . \tag{6}$$

Here ω_n is a natural frequency of the system. This form of characteristic equation gives us a damping ratio $\xi = 0.71$, and the settling time can be determined through approximate relationship $t_s = 4/\xi\omega_n$. The coefficients of the Eq. 6 correspond to optimization-based design using Modulus Optimum (Section 1).

The input-output data generated with the above reference model was used to train both the FNC and the CCNC. The 1000 input-output data tuples were generated by numerical integration of the set of six ordinary differential equations describing the process using the fifth order Runge-Kutta integrator. First 500 samples are used for training, and the whole data for the final testing of the developed controllers. A square wave disturbance was applied to the molar concentration of inlet monomer and to the inlet temperature. A zero-mean white noise was applied to output measurements. The disturbances and noise were applied concurrently to represent a real situation where the disturbances and noise in the CSTR are present at the same time. For comparison, two PI controllers were used to control the same process (one for the reactor temperature control, another for the number average molecular weights control).

Simulation results are presented in Fig. 3-6. Fig. 3 shows the time responses of uncontrolled CSTR. The PI controllers, while rejecting the process disturbances and noise to some extend, show rather poor performance (Fig. 3). In contrast, the performances of both the CCNC (Fig. 4) and the FNC (Fig. 5) are significantly better.

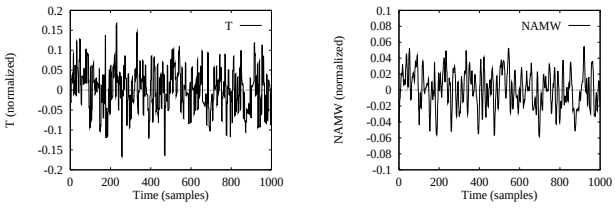

Fig. 3: The uncontrolled reactor.

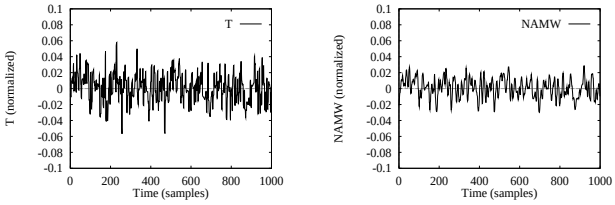

Fig. 4: The performance of the PI controller.

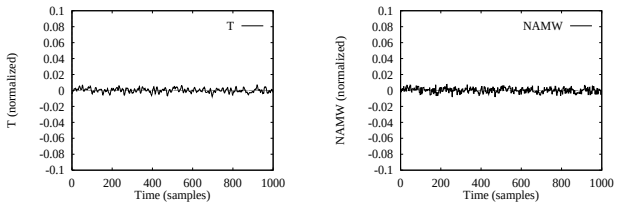

Fig. 5: The performance of the CCNC controller.

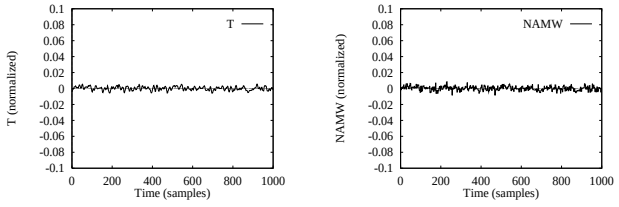

Fig. 6: The performance of the FNC controller.

5. Discussion and Conclusions

The results of the FNC may look to be just slightly better than that of the CCNC. However, the control of a CSTR is typical of many complex processes where a small improvement in performance can result in substantial economic benefits. Although the performance of the CCNC controller is generally satisfactory, the FNC is a preferable choice, especially if we take into account other advantages of the FNC over the CCNC. This includes the ability to incorporate existing knowledge and the ability to express control decisions in the form of IF-THEN rules that are easy to understand from the human (operators) point of view.

The FNC and the CCNC also differs in the ways they deal with the *moving target problem* (MTP) [6] (Section 1). In case of a CSTR, two separate system variables, i.e., the number-average molecular weight (NAMW) and the reactor temperature, need to be controlled. To handle the MTP, each hidden node in the CCNC is frozen after installation into the controller. This feature prevents the CCNC from unlearning (forgetting) the old information in order to follow well several different targets, as in our case of CSTR control. However, this is not always beneficial in real-world situations, especially with time-varying processes, since it limits the controller ability to adapt to changes. Freezing hidden nodes in the CCNC can be viewed as equivalent to a FNC with a fixed grid partition, in contrast to a variable grid partition employed by the FNC.

Both the FNC and the CCNC produce significantly better control compared to PI controllers, which indicates a distinct advantage of nonlinear control over linear control for *nonlinear* industrial processes.

References

[1] B. D. O. Anderson, "Adaptive systems, lack of persistency of excitation and bursting phenomena," *Automatica*, vol. 21, no. 3, pp. 247-258, 1985.

[2] K. Åström and B. Wittenmark, *Computer controlled systems, theory and design*, 2nd ed., Prentice-Hall, 1990.

[3] M. Brown and C. Harris, *Neurofuzzy adaptive modelling and control*, Prentice-Hall, 1994.

[4] P. Daoutidis, M. Soroush and C. Kravaris, "Feedback / feedforward control of multivariable nonlinear processes," *AIChE Journal*, vol. 36, no. 10, pp. 1471-1484, 1990.

[5] D. Dubois and H. Prade, "A unifying view of comparison indices in a fuzzy set theoretic framework," In R. R. Yager, (Ed.), *Fuzzy sets and possibility theory: recent developments*, Pergamon, NY, 1982.

[6] S. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*, San Mateo, CA: Morgan Kaufmann, pp. 524-532, 1990.

[7] J. A. Farrell and W. Baker (1993), "Learning control systems," in P. Antsaklis and K. Passino (Eds.), *An introduction to intelligent and autonomous control*, Kluwer Academic, pp. 237-262.

[8] T. Hägglund and K. Åström, "Automatic tuning of PID controllers," In W. S. Levine (Ed.), *The Control Handbook*, CRC Press, pp. 817-826, 1996.

[9] T. Hrycej, *Neurocontrol: towards an industrial control methodology*, John Wiley & Sons Inc., 1997.

[10] C. Ji and D. Psaltis, "Network synthesis through data-driven growth and decay," *Neural Networks*, vol. 10, no. 6, pp. 1133-1141, 1997.

[11] P. Mars, J. R. Chen, and R. Nambiar, *Learning algorithms: theory and applications in signal processing, control and communications*, CRC Press, Boca Raton, FL, 1996.

[12] G. N. Saridis, *Self-organizing control of stochastic systems*, Marcel Dekker Inc., 1977.

[13] J. J. Saade, "A unifying approach to defuzzification and comparison of the outputs of fuzzy controllers", *IEEE Trans. Fuzzy Systems*, vol. 4, no. 3, pp. 227-237, 1996.

[14] K. S. Tsakalis, "Performance limitations of adaptive parameter estimation and system identification algorithms in the absence of excitation," *Automatica*, vol. 32, no. 4, pp. 549-560, 1996.

[15] T. Yabuta and T. Yamada, "Neural network controller characteristics with regard to adaptive control," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 22, no. 1, pp. 171-177, 1992.