A Neural Adaptive Feedback Linearization Control for CSTR, Using NARMA-L2 Model

FARZAD HOURFAR, KARIM SALAHSHOOR, ALIREZA PAIVAR
Automation & Instrumentation Department
Petroleum University of Technology
Tehran, Iran

Abstract: In this paper we present a method for adaptive feedback linearization control of CSTR. Because of the nature of CSTR, the parameters of the system, like feed flowrate, feed temperature, concentration of components in the feed and inlet temperature of coolant, may change with time. So for detecting these changes, online system identification is performed and the NARMA-L2 model of the plant is updated in each sampling time. Experimental results demonstrate quick learning, excellent closed-loop performance, and robustness of the adaptive neurocontroller.

Keywords: CSTR, Adaptive Control, Feedback Linearization, NARMA-L2

1 Introduction

As for all practical systems, the CSTR exhibit nonlinear behaviour. Hence the required system performance, when use is made of a standard PID controller with fixed gains, is reduced; especially when the controller is required to operate over a region about the point of tuning. One solution to alleviate this problem is to use a nonlinear controller, which has been developed recently to improve consistency in terms of performance over a range about the point of tuning.

The nonlinear controller offers improvement over a linear controller, further improvement in performance is sought and it was considered that neural network techniques may be good candidates for achieving this. The aim of this paper is to describe adaptive feedback linearization controller based on NARMA-L2.

2 Feedback Linearization

Feedback linearization is commonly discussed in continuous-time framework. The fundamental assumption made about the system is that the model of it be written in the canonical form

\[
\begin{bmatrix}
\dot{x}_1(t) \\
\dot{x}_2(t) \\
\vdots \\
\dot{x}_n(t)
\end{bmatrix} =
\begin{bmatrix}
x_2(t) \\
x_3(t) \\
\vdots \\
\tilde{f}[x(t)] + \tilde{g}[x(t)]u(t)
\end{bmatrix}
\] (1)

With \( \tilde{f} \)
Assuming the function $f$ & $g$ are known, introduction of the following control redefinition will linearize the system at the sampling instants:

$$u(k) = \frac{w(k) - f[y(k),...,y(k-n+1),u(k-1),...,u(k-m+1)]}{g[y(k),...,y(k-n+1),u(k-1),...,u(k-m+1)]}$$

(8)

Selecting the virtual control input, $w$, as the reference plus an appropriate linear combination of the past outputs again allows for an arbitrary assignment of the closed-loop poles. The control design can be regarded as nonlinear counterparts to pole placement with full zero cancellation.

### 3 Feedback Linearization Using Neural Network Model

In case the system is unknown, a model can be induced from data by letting two separate neural networks approximate the function $f$ and $g$:

$$\hat{y}(k|\theta) = \hat{f}[y(k-1),...,y(k-n),u(k-2),...,u(k-m)] + \hat{g}[y(k-1),...,y(k-n),u(k-2),...,u(k-m)]u(k-1)$$ or:

$$\hat{y}(k|\theta) = \hat{f}[\phi(k),\theta_f] + \hat{g}[\phi(k),\theta_g]u(k-1)$$

(10)

Where:

$$\phi(k) = [y(k-1),...,y(k-n_g),u(k-n_h),...,u(k+1-n_h-n_k)]^T$$

(11)

$\theta$ is a vector containing the weights (and biases), $\phi$ is the regression vector and $f$ and $g$ are nonlinear function used to predict the output, $n_g$ is number of past outputs used for determining the prediction, $n_h$ is number of past inputs used for determining prediction and $n_k$ is time delay (usually equal to 1).

The closed-loop system consisting of controller and system to be controlled is depicted in fig 1. Derivation of a training method for determination of the weights in the two networks used for approximation $f$ and $g$ is straightforward.
The prediction error approach requires knowledge of the derivative of the model output with respect to the weights. In order to calculate this derivative, the derivative of each network output with respect to the weights in respective network must be determined first.

\[ \psi_f(k, \theta_f) = \frac{\partial \hat{f}(k)}{\partial \theta_f} \]

\[ \psi_g(k, \theta_g) = \frac{\partial \hat{g}(k)}{\partial \theta_g} \]  

The derivative of the model output with respect to the weights is often composed of derivatives of each network in the following manner:

\[ \psi(k, \theta) = \psi_f(k, \theta_f) + \psi_g(k, \theta_g) u(k-1) \]  

With this derivative in hand, any of the training methods can be used without further modification.

For the regression vector given by equation (11), \( n_b \) previous values of plant input and \( n_a \) previous values of plant output are used based on the knowledge of the dynamic system. The functions \( f(\varphi(k), \theta) \) and \( g(\varphi(k), \theta) \) can be chosen as any smooth functions, which we select to be feedforward three-layer (input layer-hidden layer & output layer) neural networks with nonlinear activation function for the hidden layer. The resulting plant model is known as nonlinear autoregressive moving average model (NARMA-L2). The NARMA-L2 model can be expressed by:

\[ \hat{y}(k) = f^2(a^2) + g^2(a^2)u(k-1) = f^2(W_f^2 a^2) + g^2(W_g^2 a^1)u(k-1) = f^2(W_f^2 f^1(W_f^1 a^0)) + g^2(W_g^2 g^1(W_g^1 a^0))u(k-1) \]  

where \( f^1(x) = g^1(x) = tanh(x) \) and \( f^2(x) = g^2(x) = x \) are activation functions of the hidden layer and the output layer, respectively. The quantities \( a^0, a^1, a^2 \) are the outputs of input, hidden and output neurons, respectively. The weights of hidden and output layers are given by \( W_f, W_g \) and \( W_f, W_g \), respectively.

It's highly recommended to remove the mean and scale all signals to the same variance (usually zero mean and variance 1). The signals are likely to be measured in different physical units and without scaling there is a tendency that the signal of largest magnitude will be too dominating. Moreover, scaling makes the training algorithm numerically robust and leads to a faster convergence. Finally, experience has shown that it simply tends to give better models.

If the network model is a three-layer neural network (Input layer- Hidden layer- Output layer), with linear output units, it's straightforward to rescale the weights after the training session completed. In this way the final network model can work on unscaled data.

Neural network learning speed is very important for real time system identification. To realize fast learning, a recursive Levenberg–Marquardt minimization method [12] is used. It is an intermediate method between the steepest descent and Gauss–Newton, and it has good convergence properties.

The online training algorithm is derived to minimize the following criterion:

\[ F(\theta_k) = \hat{e}^2(k) = (\hat{y}(k) - y_r(k))^2 \]  

In equation (15), \( \hat{e}(k) \) is the prediction error, \( y_r(k) \) is the actual plant output, and \( \theta_k \) consists of all weights of the neural network. The weights of the network are adjusted according to [12]:

\[ \theta_k = \theta_k - \Delta \theta_k = \theta_k - [G(\theta_k)G^T(\theta_k) + \lambda_k I]^{-1} G(\theta_k) e(\theta_k) \]  

where \( G \) is the Jacobian matrix \( (\partial F(\theta_k)/\partial \theta_k) \) and \( I \) is the identity matrix. This algorithm reduces to the steepest descent method with small learning rate as \( \lambda_k \)
is increased. If $\lambda_k$ is decreased to zero the algorithm approaches the Gauss–Newton method. Thus, the algorithm provides a nice compromise between the speed of Newton’s method and the guaranteed convergence of steepest descent. Since the system identification is performed online in the presence of disturbances acting on the system, an estimated disturbance model is reflected in the identified model and there is no need to model the disturbances separately.

4 Control of CSTR with Online Feedback linearization

Figure (2) shows the schematic of cooled exothermic CSTR. The reaction is first order in reactant $A$ and has a heat of reaction which is based on reactant $A$. A well-mixed cooling jacket surrounds the reactor to remove the heat of reaction. Cooling water is added to the jacket at a rate of $F_j$ and at an inlet temperature of $T_{j0}$. The volume $V$ of the contents of the reactor and the volume $V_j$ of water in the jacket are both constant. So in this case we have:

$$u = F_j = F_{j0}$$

$$F_0 = F$$

(17)

Fig 2. Schematic of CSTR

The first-principles model of the system is:

$$\frac{dC_A}{dt} = \frac{F_0}{V} (C_{A0} - C_A) - k_0 \exp\left(-\frac{E_a}{RT}\right)C_A$$

$$\frac{dT}{dt} = \frac{F_0}{V} (T_0 - T) - \frac{H_r}{\rho c_p V} k_0 \exp\left(-\frac{E_a}{RT}\right)C_A$$

$$+ \frac{UA}{\rho c_p V} (T_j - T)$$

(18)

$$\frac{dT_j}{dt} = \frac{UA}{\rho c_p V_j} (T - T_j) + \frac{u}{V_j} (T_{j0} - T_j)$$

Table 1. Nominal values of the model parameters

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
<th>Value and unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0$</td>
<td>Feed flowrate</td>
<td>0.2m³/min</td>
</tr>
<tr>
<td>$V$</td>
<td>Reactor volume</td>
<td>2m³</td>
</tr>
<tr>
<td>$K_0$</td>
<td>Reaction rate coefficient</td>
<td>$3.5 \times 10^{6}$/min</td>
</tr>
<tr>
<td>$E_a$</td>
<td>Activation energy</td>
<td>49.884kJ/mol</td>
</tr>
<tr>
<td>$R$</td>
<td>Ideal gas constant</td>
<td>$8.313 \times 10^{-3}$kJ/mol</td>
</tr>
<tr>
<td>$H_r$</td>
<td>Heat of reaction</td>
<td>500kJ/mol</td>
</tr>
<tr>
<td>$C_{A0}$</td>
<td>Concentration of A in feed</td>
<td>1000mol/m³</td>
</tr>
<tr>
<td>$T_0$</td>
<td>Feed temperature</td>
<td>30°C</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Density of solution</td>
<td>1000kg/m³</td>
</tr>
<tr>
<td>$c_p$</td>
<td>Heat capacity of solution</td>
<td>4.2kJ/kg°C</td>
</tr>
<tr>
<td>$UA$</td>
<td>Heat transfer coefficient (surface)</td>
<td>252kJ/min°C</td>
</tr>
<tr>
<td>$V_j$</td>
<td>Jacket volume</td>
<td>0.4m³</td>
</tr>
<tr>
<td>$T_{j0}$</td>
<td>Inlet temperature of coolant</td>
<td>10°C</td>
</tr>
</tbody>
</table>

$C_A(mol/m³)$ is the concentration of the $A$ component in the reactor, $T_0(°C)$ is the temperature of the reactor and $T_{j0}(°C)$ is temperature of the jacket of the reactor, while the input of the process is the flowrate of the cooling material $u(m³/min)$. The controlled output of the process is the reactor temperature. The parameters and their nominal values of the model are given in Table 1.

Because of the nature of CSTR, the adaptation is completely necessary for system identification. Because it’s very common that one of the parameters of the system like feed flowrate, feed temperature, concentration of components in feed and inlet temperature of coolant change with time. But with online identification of the system, the controller can compensate these changes.

In the first stage, we should generate input-output data for obtaining the NARMA-L2 model of the process (Figure 3). The inputs of the NARMA-L2 were selected $y(k-1), y(k-2), u(k-1) & u(k-2)$.

As it was mentioned, for making the training algorithm numerically robust and having a faster convergence, we scaled the input-output data to zero mean and variance 1 (Figure 4).

Figure 5 shows the model’s output and the real output and figure 6 shows the error between the model’s output and real output. We can see that the validation of the NARMA-L2 is acceptable.
In this case the desired polynomial has been selected to \( A_p(z) = z^2 - 1.4z + 0.49 \) corresponding to two poles in \( z = 0.7 \).

An important point is: although most CSTRs are designed for working in regulatory control problem, it can be seen that the controller is also good enough for servo control problems. In figure 7 we can see set-point, plant output and desired polynomial output.
5 Conclusions

This paper presented the development and evaluation of an NN based adaptive controller. The controller was designed to control the temperature of a CSTR. The controller development followed the feedback linearization control methodology with the plant represented by two multilayer feedforward NNs (for estimating functions \( f \) and \( g \)). The nonlinear plant identification was done on-line using the Levenberg–Marquardt algorithm for quick learning. With this method any changes in the parameters of the system like feed flowrate, feed temperature, concentration of components in feed and inlet temperature of coolant could be detected and remedial functions can be done. Experimental results demonstrate quick learning, excellent closed-loop performance, and robustness of the adaptive neurocontroller.

References: