15

Adaptive PID Control of a Nonlinear Servomechanism Using Recurrent Neural Networks

Reza Jafari[1] and Rached Dhaouadi[2]
[1]*Oklahoma State University, Stillwater, OK 74078,*
[2]*American University of Sharjah, Sharjah,*
[1]*USA*
[2]*UAE*

1. Introduction

Recent progress in the theory of neural networks played a major role in the development of new tools and techniques for modelling, identification and control of complex nonlinear dynamic systems. Intelligent control, with a special focus on neuro-control has been used successfully to solve difficult real control problems which are nonlinear, noisy and relatively complex. This is due to the fact that neural networks have an inherent ability to learn from input-output data and approximate an arbitrarily nonlinear function well. The inclusion of semi-linear sigmoid activation functions offers nonlinear mapping ability for solving highly nonlinear control problems (Omatu et al., 1995).

A large number of identification and control structures have been proposed on the basis of neural networks in recent years (Jain & Medsker, 1999). Most of the developed neural networks use a feed-forward structure along with the back-propagation training algorithm. Recently, more research interest is given to recurrent networks with special application to dynamic systems. A Recurrent Neural Network (RNN) exhibits internal memory due to its feedback structure, which gives the network the possibility of retaining information to be used later. By their inherent characteristic of memorizing past information, for long or short-term periods, RNNs are good candidates for nonlinear system identification and control (Narendra & Pathasarathy, 1990).

Although control theory has made great advances in the last few decades, which has led to many sophisticated control schemes, PID control remains the most popular type of control being used in industry today. This popularity is partly due to the fact that PID controllers have simple structures that are easily implemented. On-line self-tuning PID controller offer an advantage for plants that have uncertain dynamics, time varying parameters, and nonlinearities. Recently a lot of attentions have been focused on neural based PID controller, and many efforts have been done to investigate different aspects of deploying neural networks in the area of adaptive PID control (Puskorius & Feldkamp, 1993), (Saikalis, 2001)

The concept of adaptive PID control was introduced to compensate the drawbacks of the fixed-gains PID controller. For example, if the operating point of a process is changed due to disturbances, there is a need to adjust the controller parameters manually in order to keep

the optimal settings. Usually this procedure known as tuning is difficult and time consuming for systems with interacting loops. In addition to these difficulties, the conventional PID tuning methods have major drawbacks. For example, the Ziegler-Nichols (Astrom & Wittenmark, 1989) tuning method is sensitive to disturbances because of its reliance on open loop experiments. The tuning method proposed by Nishikawa (Nishikawa et al. 1989) requires man-machine interaction in which the operator needs to generate input signals every time the parameters have to be modified in order to adapt to changes in the process dynamics. Adaptive controller with the ability of self-tuning is therefore the ideal solution to all these difficulties.

Substantial research effort is also ongoing in the general area of adaptive control with neural networks, both in designing structures and learning algorithms (Chang et al. 2003), (Kuc & Gie, 2000), (Ranger & Desbiens, 2003), (Liu, 2001), (Mandic & Chamers, 2001). The design and implementation of adaptive control for nonlinear dynamic systems is challenging and extremely difficult. In most cases, developing adaptive control strategies depend on the particular information of the nonlinear structure of the plant that needs to be controlled. Neural networks with the ability to deal with nonlinearities can be used to develop an adaptive controller for unknown systems. If the relationship between the input and the output of an unknown nonlinear plant is modeled by an appropriate neural network, the model obtained can be used to construct a proper controller. The whole procedure of training and construction of a neural based controller can be implemented on-line. The neural network model is updated by measured plant input and output data and then the controller parameters are directly tuned using the updated model.

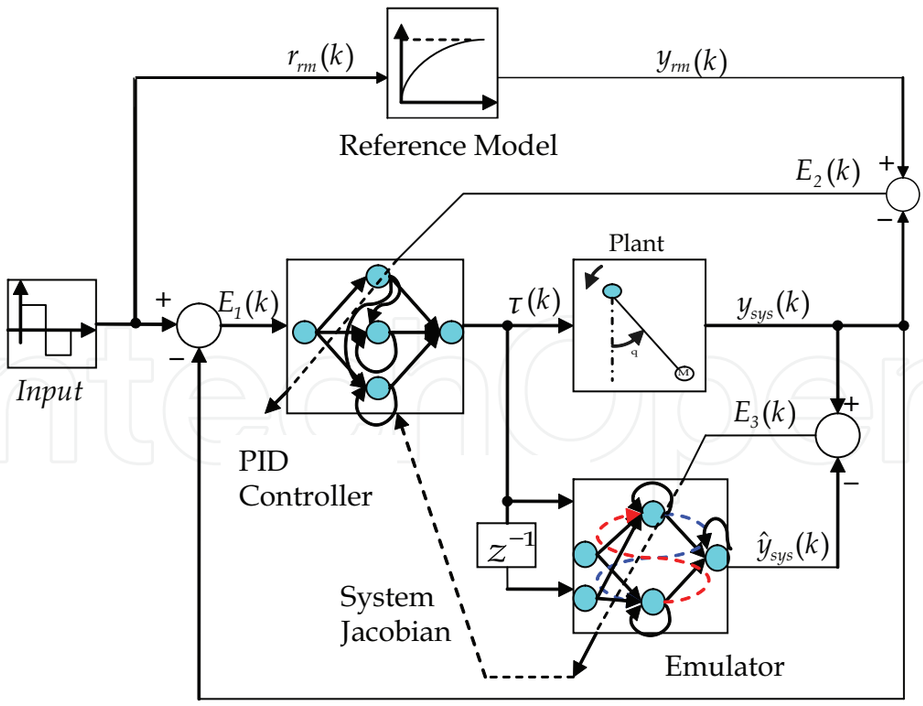


Fig. 1.1 Adaptive PID with RNN based emulator

In this chapter RNNs are used in system modeling and in the design of an adaptive PID controller for nonlinear electromechanical systems such as servo drives in robot

manipulators. The design of a servo drive system represents a difficult problem in most cases because of troublesome characteristics such as severe friction nonlinearities, variable parameters, time–varying process dynamics and unobservable system states and disturbances.

The proposed adaptive control scheme studied in this chapter is based on a RNN-PID controller combined with a RNN system emulator as shown in Fig. 1.1 (Dhaouadi & Jafari, 2007). The RNN-PID controller is designed based on the discrete equations of the PID controller transfer function. The parameters K_p, K_i and K_d of the partially connected RNN are regarded as the gains of the PID controller. These gains are not fixed, but can be adjusted on-line based on an adaptation law so as to achieve the desired control objectives. The plant direct model is constructed with the RNN emulator as shown in Fig. 1. The network is tuned online to provide the Jacobian coefficients of the system which are needed to adapt the PID gains. The self-tuning RNN will be trained using the gradient decent RTRL algorithm, (Kuc & Gie, 2000). The training of the RNN emulator will be first performed off-line so as to learn the dynamics of the plant and will be next optimized on-line.

Such control approach can be classified as indirect adaptive control, as the parameters of the plant model are adapted and control is computed based on the current model, rather than directly adapting the controller parameters.

Global asymptotic stability of the closed loop system is a challenging problem. Absolute stability analysis of RNN in general is investigated via Linear Matrix Inequality (LMI) (Barbanov & Prokhorov 2002). Barabanov and Prokhorov derived a sufficient condition for the network parameters which guarantees the absolute stability of RNN in a general form. The method is based on the sector condition. Barabanov and Prokhorov introduced later a new algorithm for global asymptotic stability of nonlinear discrete-time systems (Barbanov & Prokhorov, 2003). The new method, for reduction of a dissipativity domain of a discrete-time system, approximates level surface of Lyapunov function. In this paper, we develop a criterion to prove the stability of the RNN-PID controller in the sense of Lyapunov.

In summary, the analysis presented in this chapter shows that appropriately structured recurrent neural networks can provide conveniently parameterized dynamic models of nonlinear systems for use in adaptive PID control. The main features of the proposed new adaptive PID controller are

- Compensation of different process and unmodelled uncertainties,
- Simple to configure since it does not require a process model.
- Could track changes of process dynamics on-line.
- Has all the properties of PID control.

The chapter is organized as follows: section 1 gives a literature review and introduces a general overview of the control methodology used. The main contribution of this work is represented in section 2 where an adaptive RNN-PID controller is developed for Reference Model Control. Stability analysis of the designed controller is investigated via Lyapunov theory in section 3. Finally, discussion of simulation results and conclusions are given in section 4.

2. Adaptive RNN-PID design

The PID controller is one of the most useful and familiar controller used in industry. PI and PID controllers have been proven to be remarkably effective in regulating a wide range of processes. However, the PID controller may give low performance when dealing with

highly nonlinear and uncertain systems. The abilities of neural networks in dealing with nonlinear dynamical systems make them a viable alternative approach to deal with complex systems. RNNs will be used next to develop an adaptive PID controller which is robust to system parameters variation and uncertain dynamics.
In this section, we will develop an adaptive RNN based PID controller for nonlinear dynamic systems. The parameters of the proposed RNN-PID controller are adjusted on-line. A single-axis servomechanism is used as a case study to study the performance of the PID controller and validate the proposed adaptive control scheme.

2.1 Discrete-time PID controller

The design of the RNN based PID controller starts by deriving the discrete-time PID equation. From this difference equation the network can be designed accordingly. The general PID transfer function in the s-domain is given by

$$\frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d\left(\frac{s}{1+\tau s}\right). \tag{2.1}$$

For practical applications, an approximate derivative term is introduced to reduce the effect of measurement noise. Next, the discrete-time representation of the PID controller is obtained by mapping the transfer function from the s-domain to the z-domain using the bilinear transformation.

$$s \Leftrightarrow \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right), \tag{2.2}$$

The discrete-time control signal $u(n)$ is derived from the error signal $e(n)$ as follows.

$$u(n) = K_p e(n) + K_i v(n) + K_d w(n)\,, \tag{2.3}$$

where $v(n)$ is the integral term, and $w(n)$ is the derivative term.

$$\frac{V(s)}{E(s)} = \frac{1}{s}, \tag{2.4}$$

$$\frac{W(s)}{E(s)} = \frac{s}{1+\tau s}, \tag{2.5}$$

$$\frac{V(z)}{E(z)} = \frac{T}{2}\left(\frac{1+z^{-1}}{1-z^{-1}}\right), \tag{2.6}$$

The difference equation of the integral term $v(n)$ is

$$v(n) = v(n-1) + \frac{T}{2}[e(n) + e(n-1)]\,, \tag{2.7}$$

Similarly,

$$\frac{W(z)}{E(z)} = \frac{2(1-z^{-1})}{(2\tau+T)-(2\tau-T)z^{-1}}, \tag{2.8}$$

Defining

$$\alpha_0 = 2\tau + T \text{ and } \alpha_1 = 2\tau - T \tag{2.9}$$

The difference equation of the approximate derivative term is

$$\frac{\alpha_0}{2} w(n) = \frac{\alpha_1}{2} w(n-1) + e(n) - e(n-1), \tag{2.10}$$

This equation is written next in a modified form through a change of variables. Let's define $p(n) = \frac{\alpha_0}{2} w(n)$, then

$$p(n) = \frac{\alpha_1}{\alpha_0} p(n-1) + e(n) - e(n-1), \tag{2.11}$$

The PID derivative gain will be therefore changed accordingly

$$K_d w(n) = \left(\frac{2K_d}{\alpha_0} \right) p(n), \tag{2.12}$$

2.2 RNN controller design

Combining the derived discrete-time equations (2.5-2.12) of the PID controller, the corresponding recurrent neural network can be designed accordingly. The input to the network is the error signal and the output of the network is the control signal. There are several ways of designing the network architecture to represent the PID controller. In our approach, a partially connected recurrent neural network is used with a single hidden layer and three hidden neurons as shown in Fig. 2.1. The activation function is assumed to be linear. The feedback connections between the neurons in the hidden layer have one sampling time delay. The network parameters are clustered in three matrices W_{hi}, R_h, W_{ch}.

$$W_{hi} = \begin{bmatrix} 1 \\ \frac{T}{2} \\ 1 \end{bmatrix}, \; R_h = \begin{bmatrix} 0 & 0 & 0 \\ \frac{T}{2} & 1 & 0 \\ -1 & 0 & \frac{\alpha_1}{\alpha_0} \end{bmatrix}, \; W_{oh} = \begin{bmatrix} K_p & K_i & \frac{2K_d}{\alpha_0} \end{bmatrix}, \tag{2.13}$$

where, W_{hi} represents the RNN gains between the input layer and the hidden layer, W_{ch} represents the RNN gains between the hidden layer and the output layer, and R_h represents the RNN feedback gains between the neurons in the hidden layer. T is the sampling time, and K_p, K_i and K_d are the controller gains.

One of the major advantages of the designed network is the simplicity of the designed controller. The training procedure of the proposed controller is relatively easy due to the linearity of the activation functions.

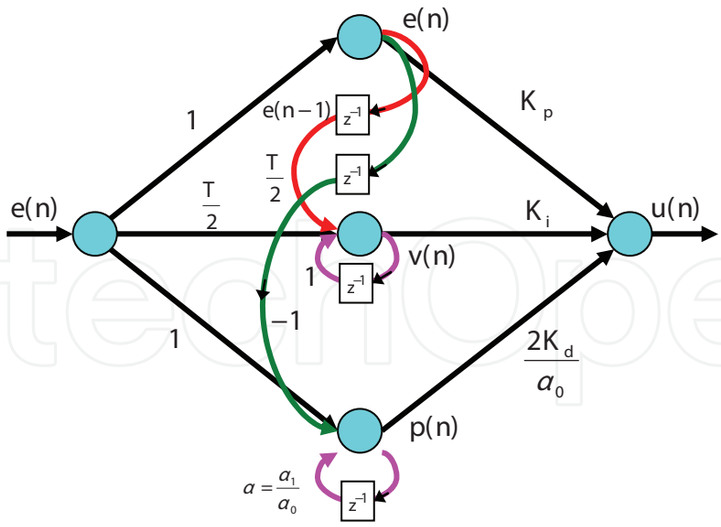


Fig. 2.1 Partially connected RNN-based PID controller

2.3 PID update equations

As it is shown in Fig. 2.1, the network includes four weights K_p, K_i, $\frac{2K_d}{\alpha_0}$, and $\frac{\alpha_1}{\alpha_0}$, which need to be tuned. The rest of the weights are fixed. The network output can be computed using the forward equations as follows

$$e(n) = O_1(n) \tag{2.14}$$

$$v(n) = O_2(n) = O_2(n-1) + \frac{T}{2}[e(n) + e(n-1)] \tag{2.15}$$

$$p(n) = O_3(n) = \alpha O_3(n-1) + e(n) - e(n-1) \tag{2.16}$$

$$u(n) = K_p O_1(n) + K_i O_2(n) + K_d^* O_3(n) \tag{2.17}$$

where the gains α and K_d^* are defined by

$$\alpha = \frac{\alpha_1}{\alpha_0},\ K_d^* = \frac{2K_d}{\alpha_0} \tag{2.18}$$

The training algorithm of the RNN-PID controller is based on the gradient descent method and uses the output error signal $e(n)$, which is the difference between the actual RNN output and the desired output.

$$\varepsilon(n) = d(n) - u(n) \tag{2.19}$$

For offline training, the data set $(\varepsilon(n), d(n))$ is generated from the simulated PID control system and is used to train the RNN. The performance index to be minimized is the sum of squares of errors $E_{sq}(n)$ of the training data set.

$$E_{sq}(n) = \frac{1}{2}\sum_{n=1}^{N} \varepsilon(n)^2 \tag{2.20}$$

According to the gradient-descent method, the weights are updated by performing the following derivations.

$$K_p(n+1) = K_p(n) - \eta \frac{\partial E_{sq}(n)}{\partial K_p} \tag{2.21}$$

$$K_i(n+1) = K_i(n) - \eta \frac{\partial E_{sq}(n)}{\partial K_i} \tag{2.22}$$

$$K_d^*(n+1) = K_d^*(n) - \eta \frac{\partial E_{sq}(n)}{\partial K_d^*} \tag{2.23}$$

$$\alpha(n+1) = \alpha(n) - \eta \frac{\partial E_{sq}(n)}{\partial \alpha} \tag{2.24}$$

where η is the learning ratio. The chain rule is used to back propagate the error to find the terms to minimize the performance index. The derivative of the sum squares errors with respect to the network parameters can be written as

$$\frac{\partial E_{sq}(n)}{\partial K_p} = -\varepsilon(n)\frac{\partial u(n)}{\partial K_p} = -\varepsilon(n)O_1(n) \tag{2.25}$$

$$\frac{\partial E_{sq}(n)}{\partial K_i} = -\varepsilon(n)\frac{\partial u(n)}{\partial K_i} = -\varepsilon(n)O_2(n) \tag{2.26}$$

$$\frac{\partial E_{sq}(n)}{\partial K_d^*} = -\varepsilon(n)\frac{\partial u(n)}{\partial K_d^*} = -\varepsilon(n)O_3(n) \tag{2.27}$$

$$\frac{\partial E_{sq}(n)}{\partial \alpha} = -\varepsilon(n)\frac{\partial u(n)}{\partial \alpha} = -\varepsilon(n)K_d^*\frac{\partial O_3(n)}{\partial \alpha} \tag{2.28}$$

Based on the normal RTRL algorithm the derivative terms in equations 2.25-2.28 will be computed recursively at every time step and the network parameters are updated accordingly.

2.4 Adaptive PID controller

This section presents two direct and indirect adaptive control schemes for a PID controller using RNN. The first control scheme is based on one neural network to implement the RNN-PID, and the system Jacobian is computed by approximation. In the second control scheme, an RNN emulator is added to the system for the exact computation of the system Jacobian.

2.4.1 Adaptive PID controller without emulator

In the first control scheme, we consider only a RNN-PID controller in the system as shown in Fig. 2.2. According to the desired response of the system, the reference model is chosen to obtain the desired settling time and damping characteristics. The RNN-PID controller is trained first off-line and is placed next in series with the system for on-line tuning. The system output is fed-back to be compared with the reference signal and form the error $e_1(k)$. This error is the input signal to the RNN-PID controller. On the other hand the system output is compared with the reference model output to form the second error $e_2(k)$. By minimizing this error the system response will become closer to the model response. This minimization is done by tuning the RNN-PID parameters as discussed in the following section.

2.4.1.1 Update equations

The adaptation procedure of the PID controller is quite different from that done in off-line learning. Here the error which needs to be minimized is not immediately after the network. The objective here is to minimize the performance index function

$$I(n) = \frac{1}{2} e_2(n)^2 \tag{2.29}$$

To be able to do this minimization, we need to differentiate (2.29) with respect to the network parameters. By applying the chain rule

$$\frac{\partial I(n)}{\partial K_p} = \frac{\partial I(n)}{\partial y(n)} \times \frac{\partial y(n)}{\partial u(n)} \times \frac{\partial u(n)}{\partial K_p} = -e_2(n) J_p(n) \frac{\partial u(n)}{\partial K_p} \tag{2.30}$$

$$\frac{\partial I(n)}{\partial K_i} = \frac{\partial I(n)}{\partial y(n)} \times \frac{\partial y(n)}{\partial u(n)} \times \frac{\partial u(n)}{\partial K_i} = -e_2(n) J_p(n) \frac{\partial u(n)}{\partial K_i} \tag{2.31}$$

$$\frac{\partial I(n)}{\partial K_d^*} = \frac{\partial I(n)}{\partial y(k)} \times \frac{\partial y(n)}{\partial u(n)} \times \frac{\partial u(n)}{\partial K_d^*} = -e_2(n) J_p(n) \frac{\partial u(n)}{\partial K_d^*} \tag{2.32}$$

$$\frac{\partial I(n)}{\partial \alpha} = \frac{\partial I(n)}{\partial y(n)} \times \frac{\partial y(n)}{\partial u(n)} \times \frac{\partial u(n)}{\partial \alpha} = -e_2(n) J_p(n) \frac{\partial u(n)}{\partial \alpha} \tag{2.33}$$

where $J_p(n)$ is the system Jacobian. The terms $\frac{\partial u(n)}{\partial K_p}$, $\frac{\partial u(n)}{\partial K_i}$, $\frac{\partial u(n)}{\partial K_d^*}$, and $\frac{\partial u(n)}{\partial \alpha}$ can be calculated similar to equation 2.25-2.28 in section 2.3.

The major difference between the above difference equations and the off-line training equations is the multiplication with two additional terms which are $e_2(n)$ and $J_p(n)$. This is due to the fact that the error which needs to be minimized is not placed immediately after the network. Here the plant is placed between the error and the network. So the error should be back-propagated through the plant to reach to the networks parameters. This error back propagation through the plant requires the knowledge of the system Jacobian $J_p(n)$.

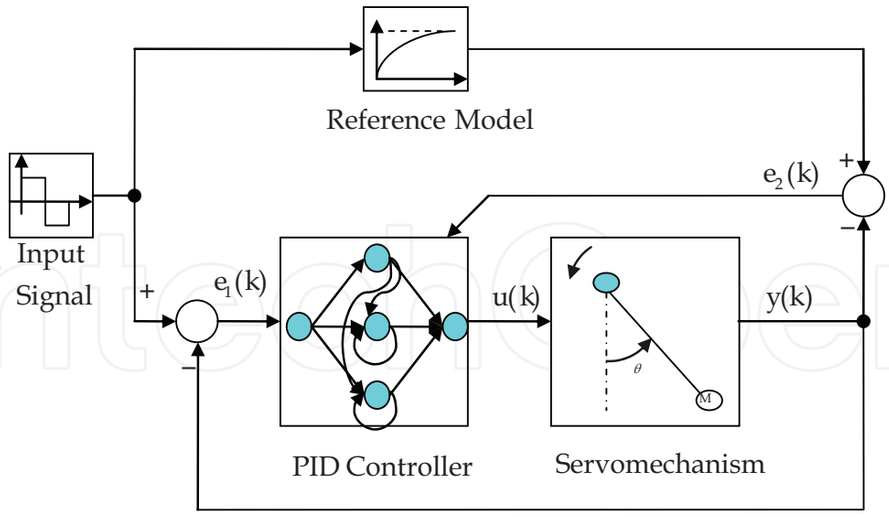


Fig. 2.2 Adaptive PID without emulator

2.4.1.2 System Jacobian

For MIMO system with n inputs and m outputs, the Jacobian is defined by the following matrix equation.

$$J_p = \frac{\partial y}{\partial u} = \begin{bmatrix} \frac{\partial y_1}{\partial u_1} & \cdots & \frac{\partial y_1}{\partial u_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial u_1} & \cdots & \frac{\partial y_m}{\partial u_n} \end{bmatrix} \tag{2.34}$$

where $y = [y_1 y_2 \cdots y_n]^T$ is the output vector and $u = [u_1 u_2 \cdots u_n]^T$ is the input vector to the system.

In this work, because we are dealing with only one input and one output, the system Jacobian is a scalar. One way to approximate this term is by taking the ratio of the difference between the current and previous input/output signals of the system. This approximation can be considered to be sufficiently precise if the sampling time is made sufficiently small.

$$J_p(n) = \frac{\partial y(n)}{\partial u(n)} \approx \frac{y(n) - y(n-1)}{u(n) - u(n-1)} \tag{2.35}$$

An alternative way to compute the Jacobian more accurately is by building an emulator of the system. This will be discussed in section 2.4.2.

With this approximation, the system Jacobian is used in (2.30)-(2.33) to perform the RNN-PID gains tuning and minimize the reference model error in the least squares sense.

2.4.1.3 Reference model difference equation

In the system simulation, the reference model is generated on-line within the control algorithm, so we need to find the difference equation for the desired model. Our desired model here is a second order model. With the general transfer function

$$G_{rm}(s) = \frac{{\omega_n}^2}{s^2 + 2\xi\omega_n s + {\omega_n}^2} \tag{2.36}$$

By considering $\tau^2 = \frac{1}{\omega_n^2}$

$$G_{rm}(s) = \frac{1}{\tau^2 s^2 + 2\tau\xi s + 1} \tag{2.37}$$

Considering the bilinear approximation rule $\left(s \to \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \right)$,

$$\frac{y_{rm}(z)}{r_m(z)} = \frac{1}{\tau^2\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right)^2 + 2\tau\xi\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right) + 1} \tag{2.38}$$

$$\frac{y_{rm}(z)}{r_m(z)} = \frac{1 + 2z^{-1} + z^{-2}}{\left(\frac{4\tau^2}{T^2} + \frac{4\tau\xi}{T^2} + 1\right) + \left(\frac{-8\tau^2}{T^2} + 2\right)z^{-1} + \left(\frac{4\tau^2}{T^2} - \frac{4\tau\xi}{T^2} + 1\right)z^{-2}} \tag{2.39}$$

The overall difference equation will be

$$y_{rm}(n) = -\frac{c_1}{c_0}y_{rm}(n-1) - \frac{c_2}{c_0}y_{rm}(n-2) + \frac{1}{c_0}\left[r_m(n) + 2r_m(n-1) + r_m(n-2)\right] \tag{2.40}$$

where r_m and y_{rm} are the input and output of the model and

$$\begin{aligned} c_0 &= \frac{4\tau^2}{T^2} + \frac{4\tau\xi}{T^2} + 1 \\ c_1 &= \frac{-8\tau^2}{T^2} + 2 \\ c_2 &= \frac{4\tau^2}{T^2} - \frac{4\tau\xi}{T^2} + 1 \end{aligned} \tag{2.41}$$

2.4.1.4 Simulation results with constant mass

To verify the performance of the adaptive RNN-PID controller and check whether it can force the servomechanism response to follow the reference model system, a multi-level step input signal is applied as shown in Figure 2.3.

Figure 2.3 shows the input signal and the system response before adaptation, and Figure 2.4 shows the response after 100 iterations. The sum of squares of errors is also illustrated in Figure 2.4. These Figures show clearly that the controller gains are adequately tuned to force the system output to follow closely the reference model output.

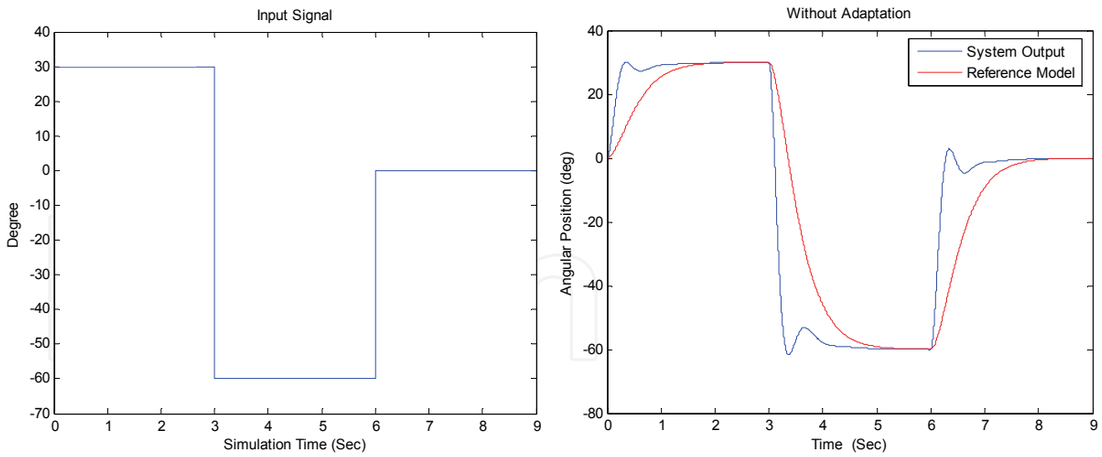


Fig. 2.3 Input signal and initial system response

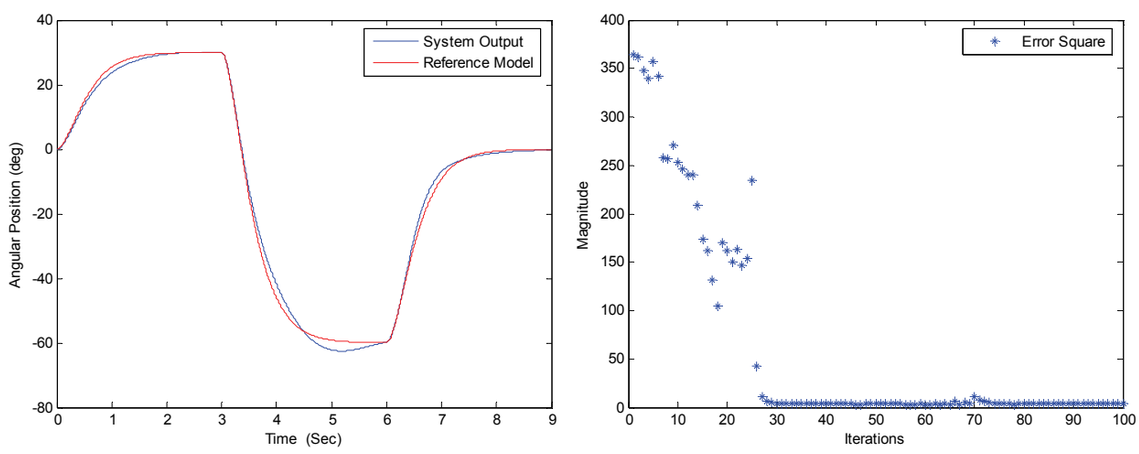


Fig. 2.4 System responses and sum of squares of errors after 100 iterations

The system parameters and are summarized in Table 2.1.

Simulation Parameters		
Simulation Time	t	9 sec
Sampling Time	T_S	0.001 sec
Reference Time Constant	τ_{rm}	0.2 sec
Damping Coefficient	ξ	1
Mass	M	2 Kg
Length	L	0.2 m
Damping Coefficient	B	1

Table 2.1 Simulation parameters for the adaptive PID controller with constant mass

The PID gains variation during the last cycle after 100 iterations is shown in Figure 2.5. It is shown that the gains, K_i, K_d*, andα, have stabilized to nearly constant values, while the

gain K_p is continuously tuned around an optimum value as the reference signal is changed.

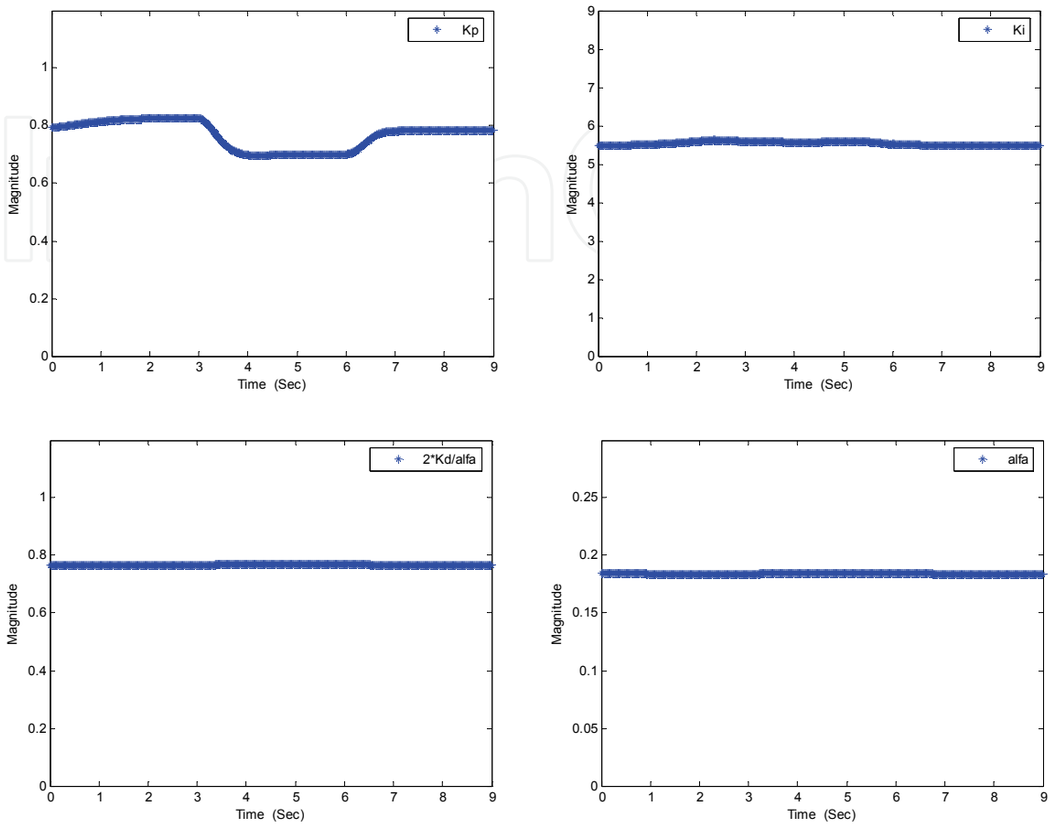


Fig. 2.5 PID gains variation for last iteration

2.4.2 Adaptive PID controller with emulator

Usually, back-propagating the error through the hybrid system results in some errors due to the approximation of the system Jacobian. To avoid these errors it is better to model the plant with a neural network. This neural network is called an emulator. By adding a neuro-emulator in parallel with the plant, the emulator will be trained to learn the dynamics of the system. So by having an RNN based plant, the system Jacobian can be computed more accurately. The emulator will be trained first off-line to make sure the RNN model is very close to the actual system. Next, the emulator will be trained on-line and will be used to compute the system Jacobian.

The servomechanism under consideration is of a second order type. The corresponding difference equation will include inputs and outputs delayed with two sampling times. We need therefore to construct a recurrent network which can memorize inputs and outputs up to two samples back in time. Figure 2.6 shows the proposed RNN to represent the pendulum system. In this partially connected network there are no recurrent connections between the output layer and the input layer. The only connections are between the output layer and the hidden layer and some internal connections within the hidden layers. The input signal includes only the current signal x(n) and the delayed signal x(n-1).

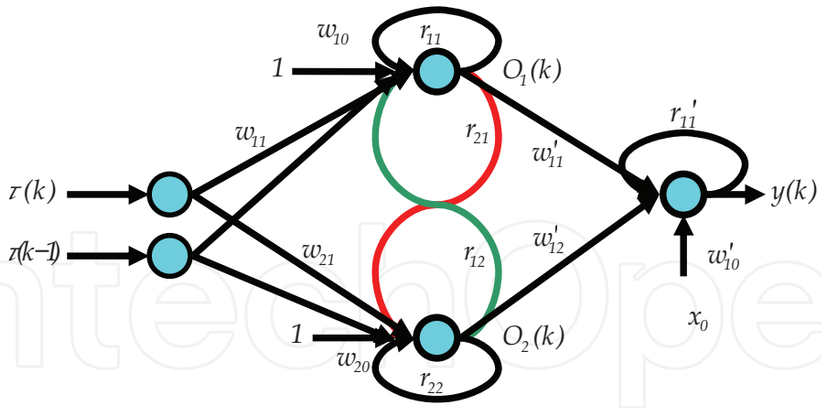


Fig. 2.6 RNN Architecture for Pendulum Identification

Figure 2.7 shows the general layout of the complete control system. As we can see from this figure the RNN based emulator is placed in parallel with the plant and is trained on-line to allow the computation of the system Jacobian.

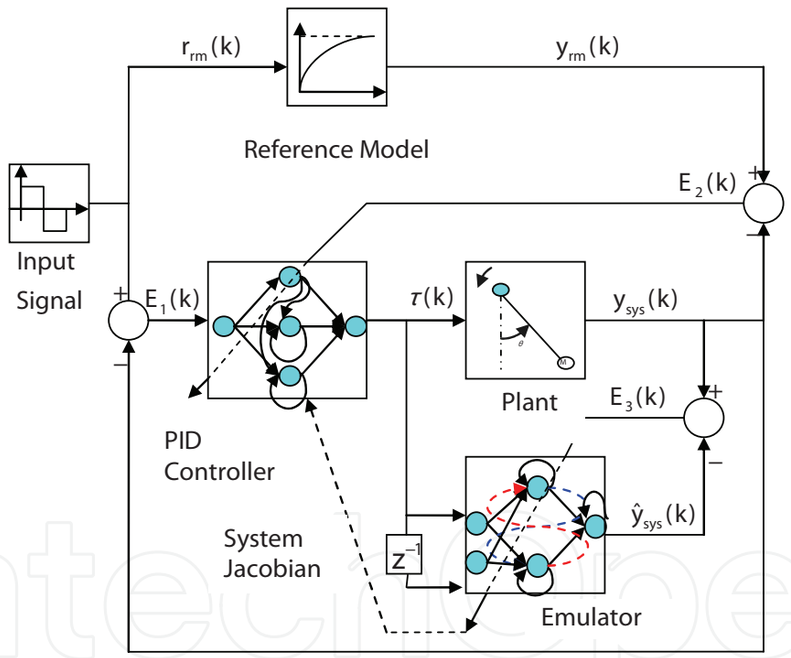


Fig. 2.7 Adaptive PID with RNN based emulator

2.4.3 Jacobian computation

For the exact calculation of the system Jacobian, there is a need to differentiate the plant output $y(k)$ with respect to the control signal $\tau(k)$. According to Figure 2.7, the forward equations of the emulator are written as

$$O_1(k) = w_{10} + w_{11}\tau(k) + w_{12}\tau(k-1) + r_{11}O_1(k-1) + r_{12}O_2(k-1) \tag{2.42}$$

$$O_2(k) = w_{20} + w_{21}\tau(k) + w_{22}\tau(k-1) + r_{21}O_1(k-1) + r_{22}O_2(k-1) \tag{2.43}$$

$$y(k) = w'_{10} + w'_{11}O_1(k) + w'_{12}O_2(k) + r'_{11}y(k-1) \tag{2.44}$$

By differentiating (2.44) with respect to $\tau(k)$

$$\frac{\partial y(k)}{\partial \tau(k)} = w'_{11}\frac{\partial O_1(k)}{\partial \tau(k)} + w'_{12}\frac{\partial O_2(k)}{\partial \tau(k)} + r'_{11}\frac{\partial y(k-1)}{\partial \tau(k)} \tag{2.45}$$

$\frac{\partial O_1(k)}{\partial \tau(k)}$ and $\frac{\partial O_2(k)}{\partial \tau(k)}$ can be derived by differentiating (2.42) and (2.43) with respect to $\tau(k)$

$$\frac{\partial O_1(k)}{\partial \tau(k)} = w_{11} + w_{12}\frac{\partial \tau(k-1)}{\partial \tau(k)} + r_{11}\frac{\partial O_1(k-1)}{\partial \tau(k)} + r_{12}\frac{\partial O_2(k-1)}{\partial \tau(k)} \tag{2.46}$$

$$\frac{\partial O_2(k)}{\partial \tau(k)} = w_{21} + w_{22}\frac{\partial \tau(k-1)}{\partial \tau(k)} + r_{11}\frac{\partial O_1(k-1)}{\partial \tau(k)} + r_{12}\frac{\partial O_2(k-1)}{\partial \tau(k)} \tag{2.47}$$

To compute $\frac{\partial \tau(k-1)}{\partial \tau(k)}$ another approximation is used

$$\frac{\partial \tau(k-1)}{\partial \tau(k)} \approx \frac{\tau(k-1) - \tau(k-2)}{\tau(k) - \tau(k-1)} \tag{2.48}$$

Using (2.48), the terms in (2.46) and (2.47) are calculated recursively and are used to determinethe system Jacobian with a better approximation. The accuracy of the Jacobian depends on how good the emulator is trained. Consequently the overall performance of the controller will be improved.

With the RNN based emulator, the update equations are the same as those derived in the previous section. Because the controller and emulator are both tuned on-line during each iteration, it is better to make the emulator training procedure faster than the controller. This will help to increase the PID controller performance.

The flowchart of the program adaptive PID controller algorithm with an RNN based emulator is shown in Figure 2.8. The program starts with some initializations such as allocating random weights, learning ratio and momentum term. Initially $\tau(k)$ which is the control signal, is assumed to be zero. $\tau(k)$ is feeding system and emulator to generate $y_{sys}(k)$ and $\hat{y}_{sys}(k)$. Based on these two values $E_3(k)$ is calculated. The input signal is sent to the reference model to generate $y_{rm}(k)$. Based on the difference between $y_{rm}(k)$ and $y_{sys}(k)$, $E_2(k)$ is calculated. Finally $y_{sys}(k)$ is compared with the reference point to form $E_1(k)$. By finding all errors the RTRL adaptation mechanism will tune the parameters accordingly. After adaptation and before executing the program for the next sample , the old values are updated. This procedure is repeated till the total number of samples is reached. The whole procedure is again executed for the given number of iterations.

Figure 2.9 shows the initial system response before tuning and Figure 2.10 show the results after adaptation. Due to the better approximation of the Jacobian the controller tuning is faster. The simulation parameters are shown in Table 2.2. Since the control scheme is based on two separate RNN's that are tuned online the stability of overall system becomes a main issue. Each RNN with its inherent feed-back connections may cause one the networks to be unstable. Instability of one the networks make the whole control system to become unstable. The brief stability analysis of the designed controller will be discussed in section 3.

Simulation Parameters		
Simulation Time	t	9 sec
Sampling Time	T_s	0.001 sec
Reference Time Constant	τ_{rm}	0.2 sec
Damping Coefficient	ξ	1
Mass	M	2 Kg
Length	L	0.2 m
Damping Coefficient	B	1
Controller Parameters		
Controller Gain	K_p	7.2464
Controller Gain	K_i	18.8667
Controller Gain	K_d*	0.7668
Controller parameter	α	0.1826
Learning ratio	η	1e-5
Momentum term	ϑ	0.06
Emulator Parameters		
Learning ratio	η	0.09
Momentum term	ϑ	0

Table 2.2 Simulation parameters for the adaptive PID controller with emulator

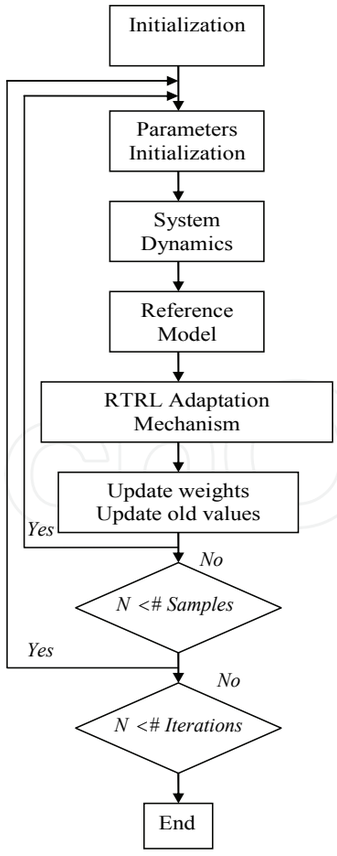


Fig. 2.8 Control program flowchart

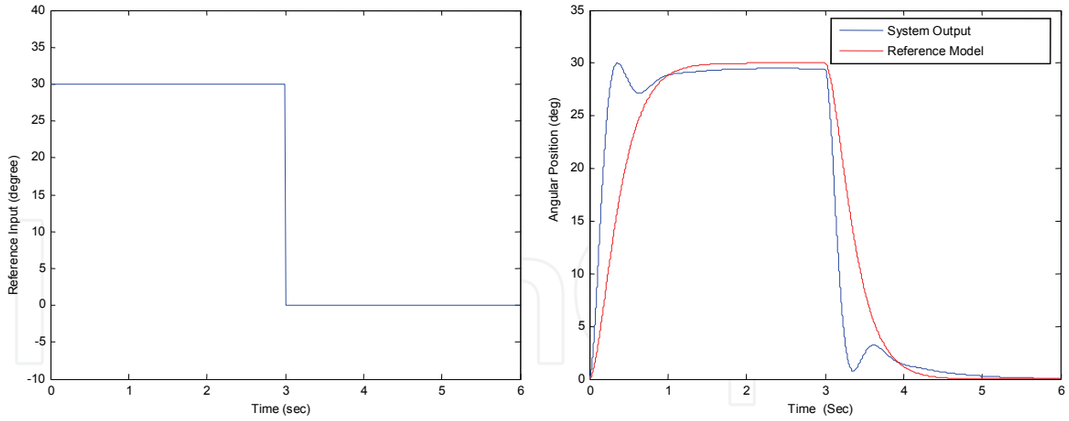


Fig. 2.9 Input signal and initial system response

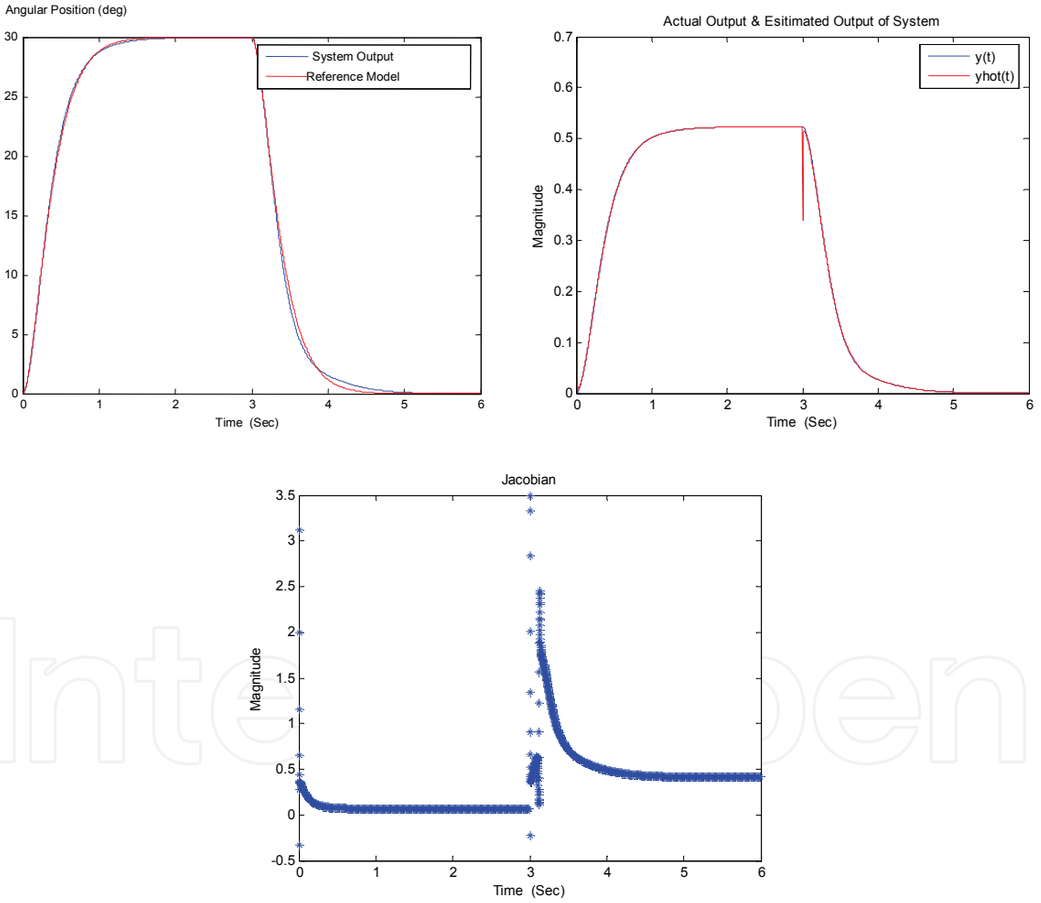


Fig. 2.10 Results after adaptation with $\tau_{rm} = 0.2$ sec

2.4.4 Robot arm control

A one-degree of freedom robot arm can be modelled as a pendulum system with a servomechanism at the joint. Controlling the robot arm position with variable load is a

challenging problem. In this section, the RNN based PID controller will be deployed to control the robot arm to follow a desired trajectory which consists of moving a load from one location to another location as shown in Figure 2.11.

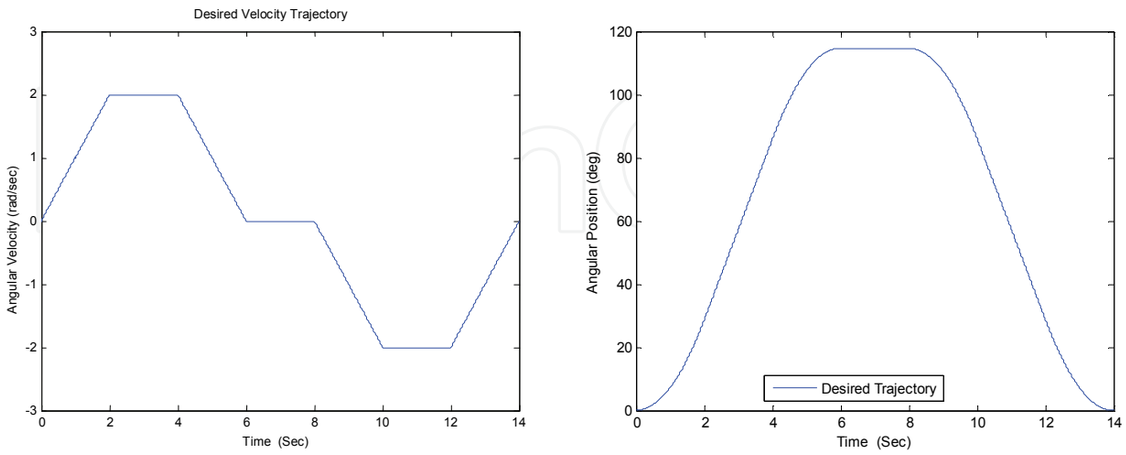


Fig. 2.11 Robot arm desired velocity and position

Assume that the robot arm starts to move from the 0^o initial position. During the first 2 seconds the arm velocity will increase with a constant slope in the positive direction. In the next two seconds the velocity becomes constant which means the arm will move at constant velocity in the same direction. In the third two seconds when the arm is close to the target, it needs to decelerate or decrease its velocity. In the next two seconds the arm should stop (velocity become zero) to pick an object. Then this procedure will be repeated in the opposite direction till reaching to the initial position. As it can be observed form Figure 2.12 which illustrates the system response without adaptation, at t=6 sec there is a disturbance applied to the system by picking up an object. It is assumed that the object mass is 10% of the initial system mass.

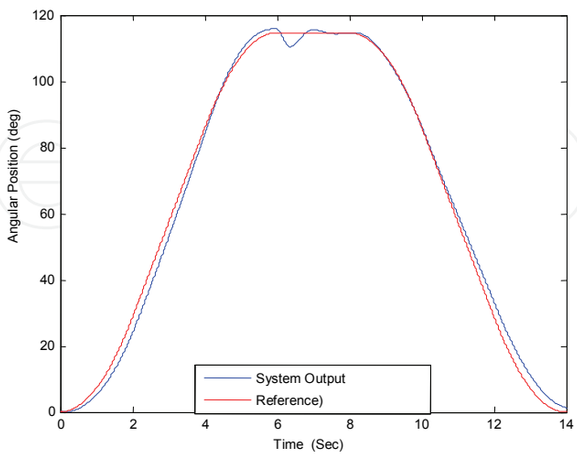


Fig. 2.12 Initial system response with disturbance

Our goal here is to verify whether the designed RNN based PID controller can tune its parameters to tolerate these disturbances or not. The tuning procedure is done on-line while

the arm is moving. If the controller is capable to control the arm for the desired trajectory with the consideration of mass disturbance, it means that our designed adaptive PID controller works fine. As it is shown in Figure 2.13 after 2 iterations the sum of squares of errors which was initially 1042 reaches to 0.092. The PID gains are successfully adapted in order to make the robot arm follow the desired trajectory.

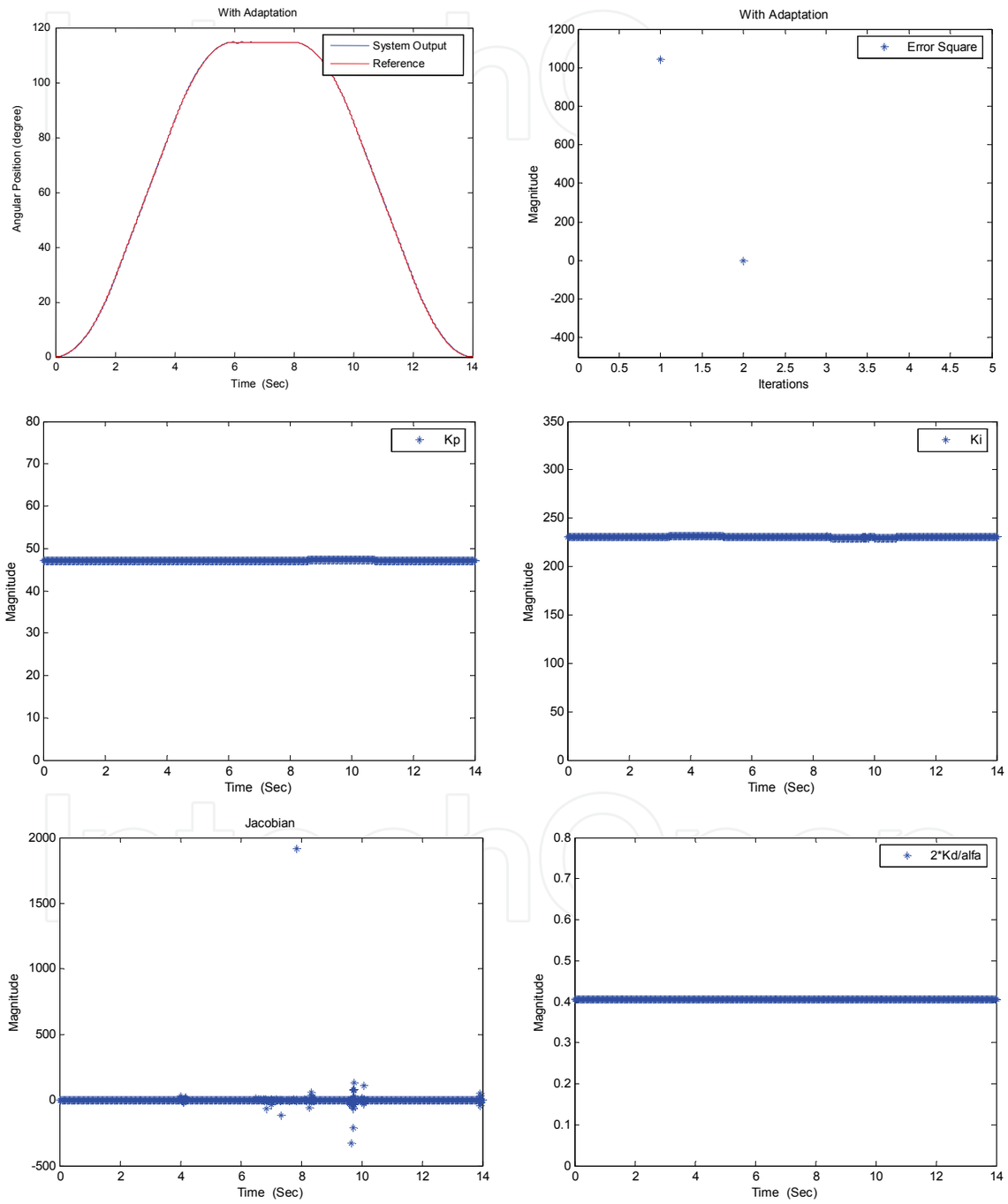


Fig. 2.13 Results after adaptation

3. Stability analysis

Global Asymptotic Stability (GAS) is a desired goal in designing any control system. However, this desired goal may not be easily achieved for systems involved with RNN. The inherent feedback properties of RNN make the analysis of this kind of systems complex. Several researches have been done to derive necessary and sufficient conditions for stability of RNN. Suykens (Suykens 2002) derived a necessary and sufficient condition for global stability of a specific class of RNN. The weak point of his criterion was due to the elimination of biases in the stability analysis. Barabanov and Prokhorov (Barbanov & Prokhorov 2002) observed that ignoring biases not only severely limits the mapping capabilities of RNN but also almost always results in extremely conservative stability criterion. They used the Linear Matrix Inequality (LMI) approach to derive a sufficient condition for the absolute stability of a given RNN. Their criterion was more useful compared to (Suykens 2002) due to the consideration of biases but still was not efficient. The derivation could not confirm the stability of many stable systems which are actually globally stable. Barabanov and Prokhorov later on proposed a new method of stability by approximating Lyapunov surface (Barbanov & Prokhorov 2003). The new method which is based on the reduction of a dissipativity domain can be applied to all bounded and differentiable systems. The proposed method can give the largest space of stable RNN parameters compared to all the previous studies.

The designed RNN-PID controller in this work is fairly easy for stability analysis because of using linear functions in the hidden layer. In this section we will put the RNN-PID controller dynamics into a state equation to derive the stability criterion.

Denote the output of the controller (control signal) as $y(k)$ and the input to the controller (error) $u(k)$. Define the output of the hidden layer as the state vector (shown in Figure 3.1)

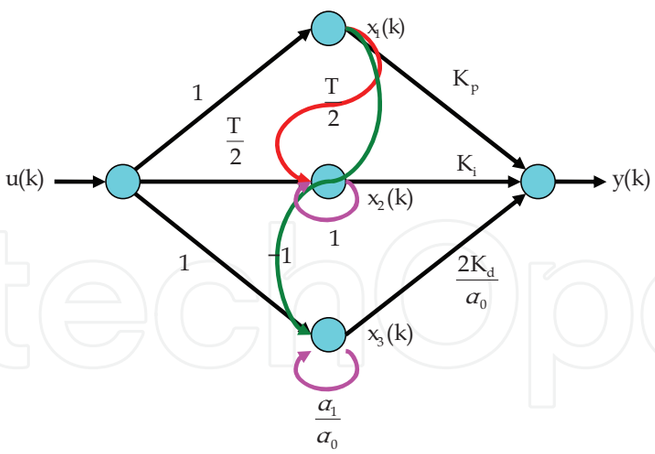


Fig. 3.1 RNN based PID controller in state space form

$$X(k) = \begin{bmatrix} x_1(k) & x_2(k) & x_3(k) \end{bmatrix}^T \tag{3.1}$$

Hence the controller dynamics can be written as

$$\begin{aligned} X(k+1) &= AX(k) + Bu(k) \\ y(k) &= CX(k) \end{aligned} \tag{3.2}$$

Where

$$A=\begin{bmatrix} 0 & 0 & 0 \\ \frac{T}{2} & 1 & 0 \\ -1 & 0 & \frac{\alpha_1}{\alpha_0} \end{bmatrix},\ B=\begin{bmatrix} 1 \\ \frac{T}{2} \\ 1 \end{bmatrix},\ C=\begin{bmatrix} K_p & K_i & \frac{2K_d}{\alpha_0} \end{bmatrix} \tag{3.3}$$

System (3.2) is considered to be slowly varying if $\|\Delta u(k)\|$ is sufficiently small. Considering the slow change in the rate of $u(k)$ then the above system would be globally asymptotically stable if and only if the eigenvalues satisfy the following condition

$$|\lambda_i|<1,\ \ i=1,2,3 \tag{3.4}$$

Where λ_i is the i^{th} eigenvalue of A. The eigenvalue of A are $\lambda_1=0,\lambda_2=1,\lambda_3=\frac{\alpha_1}{\alpha_0}$. Since one of the eigenvalue is located on the unit circle then the designed controller cannot be asymptotically stable. However the controller can be stable in the sense of Lyapunov if and only if

$$-1\le\frac{\alpha_1}{\alpha_0}\le 1 \tag{3.5}$$

Substituting for α_0 and α_1 from equation (2.9) the stability criterion for the controller can be written as

$$-T-2\tau\le -T+2\tau\le T+2\tau \tag{3.6}$$

4. Conclusion

This work investigates the application of artificial neural networks for system identification and control of nonlinear dynamic systems with a special focus on Recurrent Neural Networks (RNNs). This work mainly focused on developing a RNN-based adaptive PID controller. The corresponding algorithms are developed and tested.

The adaptive controller was designed to compensate the drawbacks of the conventional PID controller in controlling nonlinear dynamic systems. Two major control approaches have been verified. First, when the plant is known and we have some information about it in advance. The second control approach assumes a completely unknown plant. The comprehensive control approach for the second case contains two RNNs. One of them acts as a controller and the other one as an emulator. It has been shown that the control system with two networks is more efficient, reliable and accurate. However, with the RNN it has been observed that the system becomes more sensitive which needs careful tuning of the learning ratio and momentum terms.

In this research, the significant contribution is in the development of the RNN based controller with the self-tuning ability. Controlling unknown complex systems is a challenging problem, especially when the black-box system is highly nonlinear and the output is contaminated with disturbances. The authors have shown the power of RNNs to overcome these difficulties.

In the tuning process of the RNN-based controller with emulator, it is found that increasing the learning ratio and momentum term minimizes the error faster, but may deteriorate the network stability.

5. References

Sigeru Omatu, Marzuki Khalid and Rubiyah Yusof, (1995), Neuro-Control and its applications, Springer, TJ217.5.053

L. C. Jain and L. R. Medsker, (1999), Recurrent Neural Networks: Design and Applications, CRC Press, ISBN 0849371813.

Narendra, M. and K. Parthasarathy, (1990), "Identification and Control of Dynamical Systems using Neural Networks," IEEE Trans. on Neural Networks, Vol. 1, No. 1, pp. 4-27.

G. V. Puskorius and L. A. Feldkamp, (1993) "Automotive Engine Idle Speed Control with Recurrent Neural Networks '' Research Laboratory, Ford Motor Company, pp 48124,

George Saikalis, (2001), "A Neural Network Control by Adaptive Interaction," in Proceedings of the American Control Conference, Arlington, VA, June 25-27, 2001, pp. 1247-1252.

Wei-Der Chnag, Rey-Chue Hwang, Jer-Guang Hsieh, (2003), "A multivariable on-line adaptive PID controller using auto-tuning neurons," Engineering Applications of Artificial Intelligence, Vol.16, No.1, pp. 57-63, February 2003.

Tae-Yong Kuc, Woong-Gie Han, (2000) "An Adaptive PID learning control of Robot manipulators," Automatica, Vol. 36, pp. 717-725, May 2000.

Phlippe Ranger and Andre Desbiens, (2003), "Improved Back stepping-based adaptive PID control," in Proceedings of the 4th International Conference on Control and Automation ICCA'03, pp. 123-127.

G.P.Liu, "Nonlinear Identification and Control, (2001), A neural Network Approach". Springer, TJ213 .L522.

Danilo P. Mandic and Jonathon A. Chambers, (2001)," Recurrent Neural Networks for Prediction", Chichester, New York, John Wiley & Sons, Q325.5 .M36.

R. Dhaouadi and R. Jafari, "Adaptive PID Neuro-Controller for a Nonlinear Servomechanism," (2007), Proc. of the 2007 IEEE International Symposium on Industrial Electronics (ISIE07), Vigo, Spain, June 4-7, 2007.

Ronald J. Williams and Jing Peng, (1990), "An Efficient Gradient-Based Algorithm for on-line Training of Recurrent Network Trajectories," College of Computer Science Northeastern University, Boston, MA 02115, pp 490-501. 3

Ronald J. Wiliams and David Zipser, (1989), "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," MIT Press Journals, Neural Computation, Summer 1989, Vol. 1, No. 2, pp. 270-280, (doi: 10.1162/neco.1989.1.2.270).

Gaviphat Lekutai, (1997), "Adaptive Self-Tuning Neuro Wavelet Network Controllers," Doctorial thesis, Electrical Engineering Department, Blacksburg, Virginia.

Astrom K.J. and B. Wittenmark, (1989), "Adaptive Control" Addison Wesley, USA,

Nishikawa, Y., N. Sanomiya, T. Ohta, and H. Tanaka, (1984), "A method for Auto-Tuning of PID control parameters," Automatica, Vol. 20, pp. 321-332.

Feng Lin, Robert D. Brandt, and George Saikalis, (2000), "Self-Tuning of PID Controllers by Adaptive Interaction," in Proceedings of the American Control Conference, Chicago, Illinois, June 2000.

Shen Dongkai and Wang Zhanlin, "An Adaptive Controller Based On Neural Networks for Motor-Drive Load Simulator", College of Automation Science and Electrical Engineering, Beijing University of Aeronautics and Astronautics Beijing 10083, PR. China.

Wiliam H. Press, Saul A. Teukolsky, Wiliam T. Vetterling and Brain P. Flannery, (2002), "Numerical Recipe in C", Cambridge Pres.

A. Scottedward Hodel and Chales E.Hall, (2001), "Variable-Structure PID Control to Prevent Integrtator Windup", IEEE Transaction on Industrial Electronics, Vol.48, No.2, April 2001.

Haykin, S., (1994), Neural Networks A Comprehensive Foundation. Upper Saddle River, NJ: Prentice Hall,.

Hassoun, M. H., (1995), Fundamentals of Artificial Neural Networks. Cambridge, MA: MIT Press.

Smith, Bradley R., (1997), "Neural Network Enhancement of Closed-Loop Controllers for Ill-Modeled Systems with Unknown Nonlinearities," PhD Dissertation, URN: etd-111597- 81423, Mechanical Engineering, Virginia Tech. University.

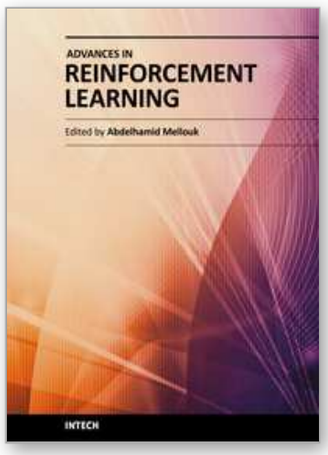
Tanomaru, J. and S. Omatu, (1991), "On the Application of Neural Networks to Control and Inverted Pendulum: an Overview," Proceedings of the 30th SICE Annual Conference, pp. 1151-1154.

Greene, M.E. and H. Tan, (1991)," Indirect Adaptive Control of a Two-Link Robot Arm Using Regularization Neural Networks," Proceedings of the International Conference on Industrial Electronics, Control and Instrumentation, Vol.2, pp. 952-956 .

N. E. Barbanov and D. V. Prokhorov, (2002), "Stability Analysis of Discrete-Time Recurrent Neural Networks," IEEE Transactions on Neural Networks, Vol. 13, No.2 March.

N. E. Barbanov and D. V. Prokhorov, (2003), "A New Method for Stability of Nonlinear Discrete-Time Systems," IEEE Transactions on Automatic Control, Vol. 48, No.12 Dec. 2003.

John A. K. Suykens, Joos P. L. Vandewalle and Bart L. R. De Moor, (1996), "Artificial Neural Networks for Modeling and Control of Nonlinear Systems," Kluwer Academic Publishers, Boston.



Advances in Reinforcement Learning
Edited by Prof. Abdelhamid Mellouk

ISBN 978-953-307-369-9
Hard cover, 470 pages
Publisher InTech
Published online 14, January, 2011
Published in print edition January, 2011

Reinforcement Learning (RL) is a very dynamic area in terms of theory and application. This book brings together many different aspects of the current research on several fields associated to RL which has been growing rapidly, producing a wide variety of learning algorithms for different applications. Based on 24 Chapters, it covers a very broad variety of topics in RL and their application in autonomous systems. A set of chapters in this book provide a general overview of RL while other chapters focus mostly on the applications of RL paradigms: Game Theory, Multi-Agent Theory, Robotic, Networking Technologies, Vehicular Navigation, Medicine and Industrial Logistic.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Reza Jafari and Rached Dhaouadi (2011). Adaptive PID Control of a Nonlinear Servomechanism Using Recurrent Neural Networks, Advances in Reinforcement Learning, Prof. Abdelhamid Mellouk (Ed.), ISBN: 978-953-307-369-9, InTech, Available from: http://www.intechopen.com/books/advances-in-reinforcement-learning/adaptive-pid-control-of-a-nonlinear-servomechanism-using-recurrent-neural-networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen