15th Triennial World Congress, Barcelona, Spain

PID AUTOTUNING USING NEURAL NETWORKS AND MODEL REFERENCE ADAPTIVE CONTROL [1]

K. Pirabakaran and V.M. Becerra

Department of Cybernetics
The University of Reading
Reading, RG6 6AY
United Kingdom

Abstract: This paper describes the application of artificial neural networks for automatic tuning of PID controllers using the Model Reference Adaptive Control approach. The effectiveness of the proposed method is shown through a simulated application. Copyright © 2002 IFAC.

Keywords: Autotuners, PID Control, Model Reference Adaptive Control, Neural Networks, Back–propagation.

1. INTRODUCTION

The PID controller is by far the most common control algorithm. Today, PID controllers can be found in virtually all control systems, with applications ranging from process conditions regulation to precision motion control for assembly and process automation. This is not surprising since the reliability of the PID controllers has been field proven by decades of successful applications (Kiong *et al.*, 1999). Their popularity is due to their robustness in a wide range of operating conditions, the simplicity of their structure as well as the familiarity of designers and operators with the PID algorithms. The importance of PID controllers cannot be undermined as they provide the engines to millions of control systems operating around the world.

Tuning a controller implies setting its adjustable parameters to appropriate values that provide good control performance. With all PID controllers, occasional retuning is needed: real-world processes change over time for a variety of reasons. PID tuning is often done manually and iteratively. The process operator or control engineer varies the parameters (in a principled or ad-hoc way) until acceptable process behavior to a small change in the setpoint is achieved. This can be a prolonged exercise.

Since the number of PID controllers that can be found in industry is so overwhelming, the tuning of such devices is an important issue. Indeed, this issue has received significant attention in the literature. Several methods have been proposed for PID auto-tuning. A subset of them can be found in the book by Astrom and Hagglund (1995). As the demand on control performance and process economy increased, and systems with more complex structure must be controlled, more sophisticated tuning methods are needed. Some authors have recently published methods for automatic tuning involving neural networks (Fujinaka *et al.*, 2000), (Rad *et al.*, 2000), (Konar *et al.*, 1995).

This paper describes the application of artificial neural networks for automatic tuning of PID controllers using the Model Reference Adaptive Control (MRAC) approach. The approach is to first construct a plant emulator, using a multi–layer perceptron (MLP) network. This emulator is then used together with an on-line trained neural network, which adjusts the PID parameters such that the error between the reference

[1] Work supported by the Research Endowment Trust Fund of the University of Reading

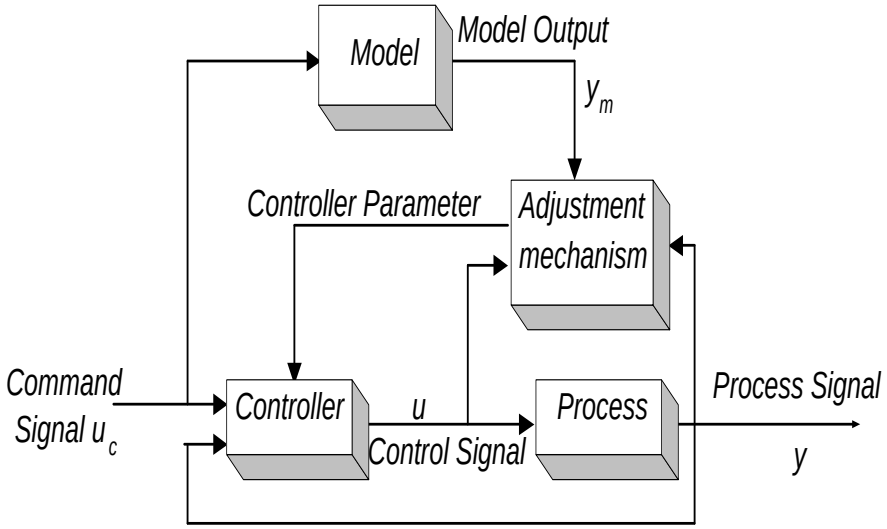

Fig. 1. A block diagram of a model reference adaptive system

model output and process output is reduced. The neural network tuner takes past plant output values, control signal and set point signal as inputs, and produces the required changes in the PID controller parameters.

This paper is organized as follows. In section 2, an overview of Model Reference Adaptive Control is presented. The proposed autotuning method is presented in section 3. Sections 4 and 5 present a case study based on a simulated two tank process model. The paper concludes with a discussion on the results and on the autotuning methods presented.

2. MODEL REFERENCE ADAPTIVE CONTROL (MRAC)

Model reference adaptive control was originally proposed to solve a problem in which the specifications are given in terms of a reference model that tells how the process output ideally should respond to the command signal. This is one of the main approaches to adaptive control. The basic idea is illustrated in Figure 1, which is the original MRAC, proposed by Whitaker in 1958. The regulator can be thought of as consisting of two loops: an inner loop which is the ordinary feedback loop composed of the process and controller. The parameters of the controller are adjusted by the outer loop in such a way that the error e between the process output y and the model output y_m becomes small. The outer loop is thus the adaptation loop (Astrom and Wittenmark, 1995).

3. PID AUTOTUNING USING NEURAL NETWORKS AND MODEL REFERENCE ADAPTIVE CONTROL

This paper combines and extends of previous works on the subject of self tuning neural network PID controller by Fujinaka *et al.* (2000) and the work by the authors on autotuning of PID controllers using MRAC (Pirabakaran and Becerra, 2001). The algorithm presented by Fujinaka *et.al* (2000) achieves the tuning objective by minimizing the squared control error ($e_c^2 = (u_c - y)^2$, where u_c is the setpoint and y is the plant output), however, no performance specifications like natural frequency or damping ratio can be considered by using their approach.

The structure of the autotuning scheme described in this paper is shown in Figure 2, where the outputs of the neural network shown in Figure 3 are the controller parameter changes, (proportional ΔK_p, integral ΔK_i, and derivative ΔK_d gains) and the inputs are selected in a suitable way according to the specific problem.

This method calculates the controller parameters on–line. As a result, the training targets for the PID controller parameters are not available, so that supervised training, such as standard back–propagation, cannot be applied. Instead, the neural network weights are calculated to minimise a criterion using a gradient method. The criterion used is the minimisation of the squared model error:

$$J = \frac{1}{2}e^2 = \frac{1}{2}(y - y_m)^2 \qquad (1)$$

where y is the process output and y_m is the reference model output. The reference model transfer function is given by:

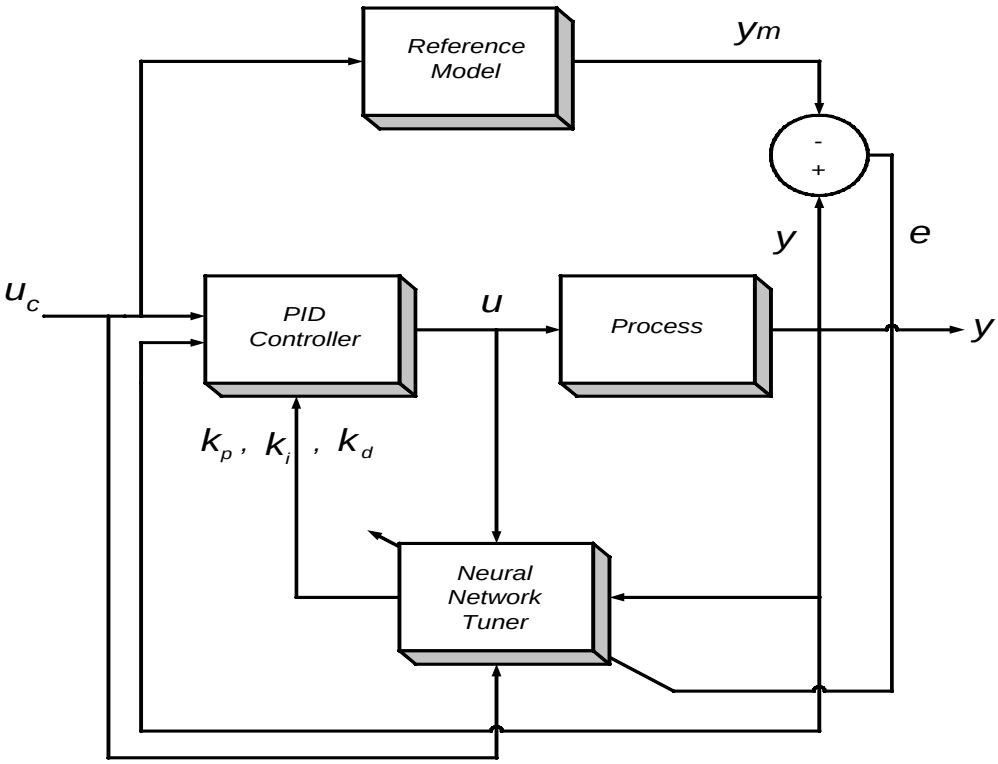

Fig. 2. Schematic diagram of autotuning PID controller

$$G_m(s) = \frac{Y_m(s)}{U_c(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \qquad (2)$$

which has unit steady state gain, natural frequency ω_n and damping ratio ζ. The reference model determines the desired output for a given input. The model error is calculated using the reference model output and the process output and then the model error is used to update the neural networks weights. In order to update the weights, the back–propagation algorithm was modified to consider the above criterion.

The velocity form of the PID control algorithm in discrete-time was used:

$$\begin{aligned} u(n) = u(n-1) &+ K_p(e(n) - e(n-1)) \\ &+ K_i e(n) \\ &+ K_d(e(n) - 2e(n-1) + e(n-2)) \end{aligned} \qquad (3)$$

where K_p, K_i, and K_d are proportional, integral and derivative gains, respectively, $u(n)$ denotes the plant input at nT, and e is the control error. Here T is the sampling interval. In order to adjust K_p, K_i, and K_d, a three layered neural network with sigmoid activation functions in the hidden layer and the linear activation functions in the output layer was used. Each layer consists of N_1, N_2, and N_3 neurons, where N_1 and N_2 can be selected by trial and error. N_3=3, corresponds to the number of PID gains. The cost function to be minimised by the back–propagation method is defined to be the squared model error (1). The connection weights at the output layer and hidden layer are updated by the following equations.

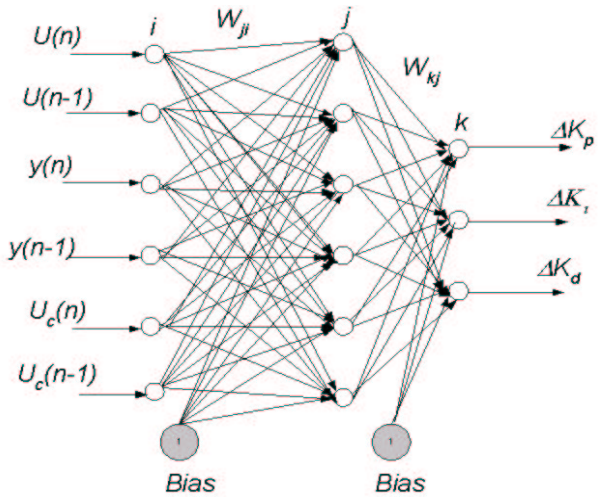

Fig. 3. Neural network for autotuning

$$\begin{aligned} \Delta w_{kj}(n+1) = \eta\delta_k o_j + \alpha\Delta w_{kj}(n) \\ + \beta\Delta w_{kj}(n-1) \end{aligned} \qquad (4)$$

$$\begin{aligned} \Delta w_{ji}(n+1) = \eta\delta_j o_i + \alpha\Delta w_{ji}(n) \\ + \beta\Delta w_{ji}(n-1) \end{aligned} \qquad (5)$$

where k=1,2,3; j=1,2,....N_2; i=1,2,...N_1, η is the learning rate, α and β are the momentum terms and w_{ji} denotes the connection weight from input node i to neuron j, w_{kj} denotes the connection weight from neuron j to output neuron k. The local gradient $\delta_j(n)$ for hidden neuron j is given by:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial o_j(n)} \frac{\partial o_j(n)}{\partial net_j(n)} \tag{6}$$

$$\delta_j(n) = \frac{\partial E(n)}{\partial o_j(n)} f_j'\left(net_j(n)\right) \tag{7}$$

Hence

$$\delta_j = f'(net_j) \sum_{k=1}^{3} e_k(n) f'(net_k) w_{kj} \tag{8}$$

The local gradient of the output neurons, δ_k, is given by:

$$\delta_k = -\frac{\partial E}{\partial net_k} \tag{9}$$

From equation (1), the model error is given by:

$$E = \frac{1}{2} e_k(n+1)^2 = \frac{1}{2}(y_m(n+1) - y(n+1))^2 \tag{10}$$

Therefore,

$$\frac{\partial E}{\partial net_k} = -e(n+1)\frac{\partial y(n+1)}{\partial net_k} \tag{11}$$

Using the chain rule,

$$\frac{\partial y(n+1)}{\partial net_k} = \frac{\partial y(n+1)}{\partial u(n)} \frac{\partial u(n)}{\partial o_k} \frac{\partial o_k}{\partial net_k} \tag{12}$$

So that the local gradient δ_k can be written as:

$$\delta_k = e(n+1)\frac{\partial y(n+1)}{\partial u(n)} \frac{\partial u(n)}{\partial o_k} \frac{\partial o_k}{\partial net_k} \tag{13}$$

Furthermore, from equation (3)

$$\frac{\partial u(n)}{\partial o_k} = \begin{cases} e(n) - e(n-1), \\ e(n), \\ e(n) - 2e(n-1) + e(n-2), \end{cases} \tag{14}$$

where k=1,2 or 3, $o_1 = K_p$; $o_2 = K_i$; $o_3 = K_d$.

The calculation of the plant jacobian $\partial y(n+1)/\partial u(n)$ is described in section 5.

4. THE TWO TANK PROCESS MODEL

In order to test the autotuning techniques introduced in this paper, a nonlinear dynamic model of a two tank process has been used. This system is illustrated in Figure 4. The liquid level control system considered here consists of two tanks, which are connected by a short pipe. The amount of liquid flowing into the tanks is regulated by a control valve. The control problem is to maintain the desired liquid level (h_2) in second tank by adjusting the valve opening percentage (V). A

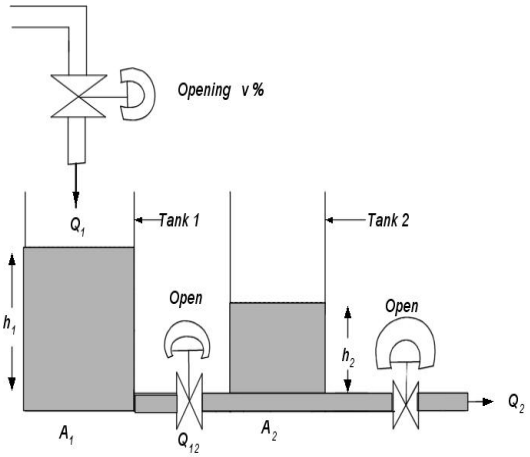

Fig. 4. Two tank process

model of this system is given by the following set of equations:

$$\begin{aligned} Q_1 &= \frac{CV}{100} \\ Q_2 &= K_2\sqrt{h_2} \\ Q_{12} &= K_1\sqrt{h_1 - h_2} \\ \frac{dh_1}{dt} &= \frac{(Q_1 - Q_2)}{A_1} \\ \frac{dh_2}{dt} &= \frac{(Q_{12} - Q_2)}{A_2} \end{aligned} \tag{15}$$

where the manipulated variable V is the valve opening in percentage, Q_1, Q_2, and Q_{12} are volumetric flows in cm^3 /s, h_1 is the liquid level in tank 1 in cm, h_2 is the liquid level in tank 2 in cm, A_1 is the base area of tank 1 in cm^2, A_2 is the base area of tank 2 in cm^2. The measured output is h_2, the liquid level in tank 2. The parameters of the process are: A_1=289 cm^2 ; A_2=144 cm^2; $c = 280\text{cm}^2\text{s}^{-1}.\%^{-1}$; $K_1 = 30\text{cm}^{5/2}\text{s}^{-1}$; $K_2 = 30\text{cm}^{5/2}\text{s}^{-1}$.

5. IMPLEMENTATION AND RESULTS

This section illustrates the application of the developed neural network based autotuning techniques (see section 3) to the two tank level control system described in section 4.

The system Jacobian $\partial y(n+1)/\partial u(n)$ is not available as the true plant parameters are assumed to be unknown. Thus, an emulator is introduced that mimics the input/output map of the plant. By using this emulator to provide an approximation to the system Jacobian, the modified back-propagation method described in section 3 can be applied to adjust the neural network tuner weights. The PID gains are adjusted by the neural network. The Jacobian of the level control

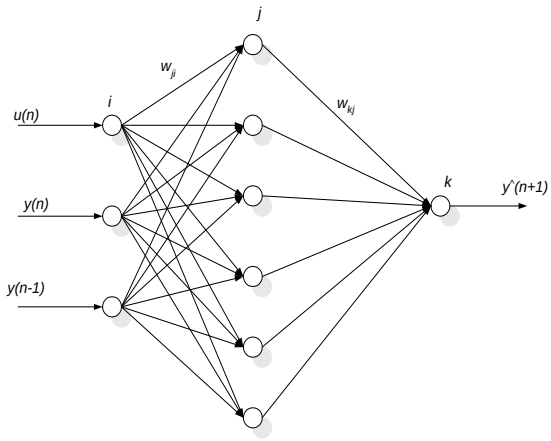

Fig. 5. Neural network emulator

system is derived by using the emulator shown in Figure 5. The emulator used is a non–linear ARX model. The input layer of the emulator consists of three units, which are connected to six units in the hidden layer. The output layer consists of one unit, representing the estimated value. Its mapping can be written as follows:

$$\hat{y}(n+1) = f(u(n), y(n), y(n-1)) \qquad (16)$$

The neural network emulator was trained off–line using the Levenberg Marquadt's algorithm. Two thousand training data samples were used to train the emulator. Using this emulator, the system Jacobian is calculated as follows:

$$\frac{\partial \hat{y}(n+1)}{\partial u(n)} = \sum_{j=1}^{6} w_{1j} o_j (1-o_j) w_{j1} \qquad (17)$$

Here the suffix 1 in denotes the neuron unit corresponding to $u(n)$ at the input layer. The range of summation over j is in accordance with the structure of the connection between the hidden layer and output layer.

To illustrate, a simulation has been carried out using the following values for the reference model parameters, sampling time, learning rates and momentum parameters, respectively: ω_n=0.02 rad/s, ζ=0.7; T=3 s, η=10^{-4}, $\alpha = 2.0 \times 10^{-7}$, β=10^{-7}. The reference signal u_c during the autotuning period was generated as follows:

$$u_c(t) = u_{sq}(t) + 0.6\sin(0.02t) + 5 \qquad (18)$$

where $u_{sq}(t)$ is a square wave with amplitude 0.6 and period 500 s.

Figure 6 shows the reference model output and processes output before, during and after the tuning period, and also the oscillations during the tuning period. Notice that the system follows quite closely the reference model output after the autotuning is carried out. The step response is shown before and after the tuning. Figure 7 shows the evolution the controller parameters. It can be seen that how the controller parameters change with time and eliminate the model error.

6. CONCLUSIONS

A new autotuning technique was proposed in which a neural network is used to tune on–line the gains of a PID controller, based on Model Reference Adaptive Control concepts. A neural network tuner is trained on–line in order to make the system behave like the reference model. The on–line training was carried out using a modified back–propagation method. This requires the use of an emulator, which is trained off–line, to obtain an approximation to the plant Jacobian. The use of neural networks allows nonlinearities in the controlled system to be considered for tuning purposes. Furthermore, the use of the Model Reference Adaptive Control approach allows desired performance measures, such as natural frequency and damping ratio, to be specified. The effectiveness of the developed techniques has been demonstrated by means of a simulated two tank process.

Further work includes the following aspects. In the method described in section 3, the stability of the closed loop system is not guaranteed during the auto–tuning period. Lyapunov theory can be used to derive an adaptation algorithm with guaranteed stability. From Figure 7 it is possible to note that although the controller parameters (K_p, K_i, K_d) have not converged when the adaptation stops, the scheme clearly makes the closed loop system follow the reference model very closely. It is then important to investigate the conditions for the convergence of the controller parameters.

7. REFERENCES

Astrom, K. and T. Hagglund (1995). *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America. Research Triangle Park, USA.

Astrom, K. J. and B. Wittenmark (1995). *Adaptive Control, 2nd ed*. Addison–Wesley. Reading, Mass.

Fujinaka, T., Y. Kishida, M. Yoshioka and S. Omatu (2000). Stabilization of double inverted pendulum with self-tuning Neuro-PID. In: *IEEE International Conference on Neural Networks*. Vol. IV. Como, Italy. pp. 345–348.

Kiong, T. K., W. Q. Guo and H. C. Chieh (1999). *Advances in PID control*. Springer. London.

Konar, A. F., S. A. Harp and T. Samad (1995). Neuro-PID controller. *US Patent and Trademark Office*. USP: 5,396,415.

Pirabakaran, K. and V. M. Becerra (2001). Automatic tuning of PID controllers using model reference adaptive control techniques. In: *IECON'01 27th Annual Conference of the IEEE Industrial Electronics Society*. Denver, Colorado, USA. pp. 736–740.

Rad, A. B., T. W. Bui, Y. Li and Y. K. Wong (2000). A new on line PID tuning method using neural networks. In: *IFAC Digital Control, Past, Present and Future*. Terrassa, Spain. pp. 443–448.

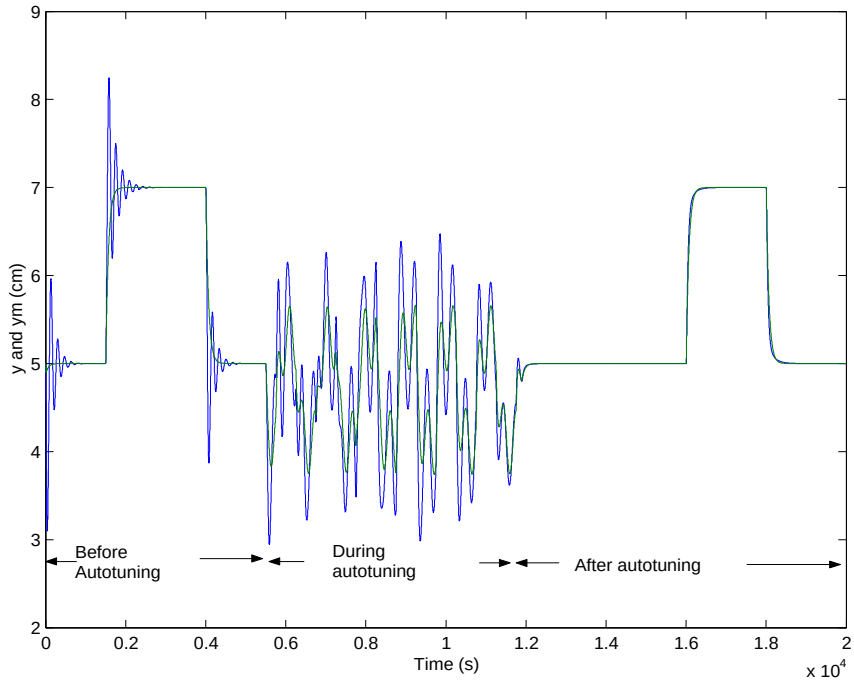

Fig. 6. Process and reference model outputs before, during and after the autotuning period

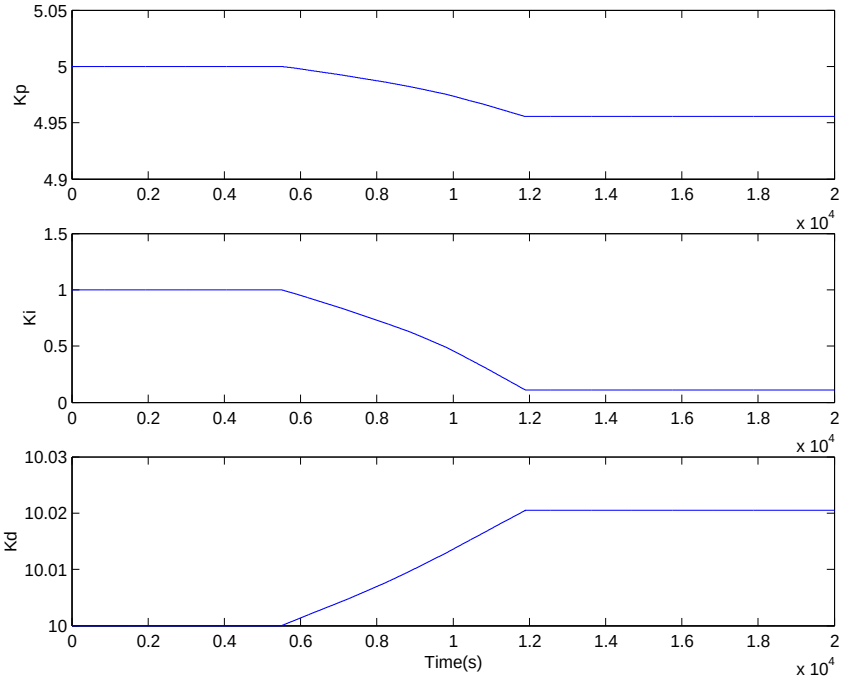

Fig. 7. Controller parameter values before, during and after the autotuning period.