# Genetic Algorithm and Elman Network Used For Tuning The Parameters Of The PID Neural Controller Based Model Reference

**Mr. Ahmed Sabah Abdul Ameer Al-Araji**

Received on: 1/12 /2004
Accepted on: 28/ 4/2005

**Abstract**

A neural network-based self-tuning PID controller is presented. The scheme of the controller is based on using a modified Elman recurrent neural network as a self-tuner for (PID) controller. The proposed method has the advantage of not necessarily using a combined structure of identification and decision, common in a standard self-tuning controller, because it uses a genetic algorithm based model reference. The paper explains the algorithm for a general case, and then presents a specific application on non-linear dynamical plant.

**الخلاصة**

إن المسيطر PID الذي أساسه الشبكة العصبية ذات التنغيم التلقائي هو شبكة أيلمن المعدلـــة الراجعـــة استخدم ليكون المسيطر المنغم للعناصر PID .

وتمتلك الطريقة المقترحة فائدة بعدم استخدام الهيكلية الثابتة لعملية تنغيم عناصر المسيطر PID من حيث عملية التعريف للمنظومة، وذلك باستخدام الخوارزمية الجينية التي أساسها النموذج المرجع.

ويشرح هذا البحث الخوارزمية لحالة عامة ثم يطبق هذه الخوارزمية على منظومة ذات ديناميكية لا خطية.

## 1-Introduction

In recent years, there has been an increasing interest in developing alternative methodologies for design of the industrial PID control such as auto-tuning, self-tuning, pattern recognition, fuzzy logic, neural network and genetic algorithm. The reason of study by the researchers is motivated by simplicity to implement the PID control in the industrial environment, by easiness of utilization by engineers and process operators, and by acceptance in the industrial sector. Some approaches proposed in the literature for deriving PID controllers are using self-tuning

control techniques based on recursive parameter estimation, others are using automatic control techniques, or intelligent control techniques [1]. Despite the huge development in control theory, the majority of industrial processes are controlled by the well-established proportional-integral-derivative (PID) control. The popularity of PID control can be attributed to its simplicity and to its good performance in a wide range of operating conditions. However, PID controllers present as disadvantage the need of retaining whenever the processes are subjected to some kind of disturbance or when processes present complexities (non-linearities). So, over the last few years, significant development has been established in the process control area to adjust the PID controller parameters automatically, in order to ensure adequate servo and regulatory behavior for a closed-loop plant [2,3,4]. The self-tuning PID control algorithms in a particular case of the minimum variance controller and predictive design have shown several difficulties to the process engineers and researchers. These self-tuning control algorithms need information about the process to be controlled, utilize a parametric mathematical model and require an on-line identification procedure (complex engineering) [5,6]. The application of intelligent techniques to control systems has been a matter of wide study in recent years. These methods are used to solve complex problems that, in many cases, do not have an analytical solution. Neural networks (NN), due to their ability to learn, have become a powerful tool in the development of the control systems. In fact, a new branch in control theory has arisen nowadays: neuro control. This discipline studies the design of control systems aided by NN. Although in the process industry simple conventional controllers, such as the PID, have largely been extended, and show good performance for many tasks, when the plant or the process under control is complex or has high non-linearities, the control performance degrades notably [7]. The objective of this paper is to highlight a tuning method that utilizes a unified approach and yields consistent performance subject to that achievable over a wide range of process models. To achieve this objective, a Genetic Algorithm (GA) will be utilized. The GA approach is an intuitive and mature search and optimization technique based on the

principles of natural evolution and population genetics. In this way, genetic algorithms have been shown to be capable of locating high performance area in complex domains without experiencing the difficulties associated with high dimensionality or false optima, as may occur with gradient descent techniques. The (GA) has been recognized as a powerful tool in many control applications such as parameter identification and control structure design [8,9].

**The paper consists of the following sections:** Section two describes the use of Recurrent Neural Networks (RNNs)-type modified Elman network structure of the PID neural controller taught on-line by using genetic algorithm mechanism. Section three represents the core of the present paper and suggests using of self-tuning PID neural controller based model reference. Section four represents the proposed algorithm for self-tuning PID type modified Elman. Illustrative example that clarifies the features of the proposed strategy is given in section five, where the example is discussed in detail. Finally, section six contains the conclusions of the entire work.

## 2-Recurrent Neural Networks/Genetic Algorithm Learning

Recurrent neural networks (RNN) have one or more feedback connections, where each artificial neuron is connected to the others [10]. The RNN structures are suitable to channel equalization and multi-user detection applications, since they are able to cope with channel transfer functions that exhibit deep spectral nulls, forming optimal decision boundaries which are less computationally demanding than MLP networks for these applications [11]. Among the available recurrent networks, modified Elman networks, as shown in figure (1), are one of the simplest types that can be trained using genetic algorithm and it is used to minimize the oscillation or even instabilities to the training controller. The output of the jth context unit in the modified Elman network is given by:

$$h_j^o(k) = a h_j^o(k-1) + h_j(k-1) \qquad (1)$$

where $h_j^o(k)$ and $h_j(k)$ are respectively the output of the jth context unit and jth hidden unit and $a$ is the feedback gain of the self-connections. The value of $a$ adopted is the same for all self-connections and is not modified by the training algorithm. The value of $a$ is

between 0 and 1. A value of $a$ nearer to 1 enables the context unit to aggregate more pattern outputs. The input and output units interact with the outside environment, while the hidden and context units do not. The input units are only buffer units "Scales" while the output units are linear units, which sum the signals fed to them. The hidden units can have nonlinear such as sigmoidal activation functions. The context units are used only to memorize the previous activation of the hidden units and can be considered to function as one-step time delays. From the figure (1) it can be seen that the following equations:

$$h(k) = F\{V1U(k), V2h^{o}(k)\} \quad (2)$$

$$O(k) = Wh(k) \quad (3)$$

where V1,V2,W are weight matrices and F is a non-linear vector function. The multi-layered modified Elman neural networks shown in figure (1) are composed of many interconnected processing units called neurons or nodes.

where:

$V1$: Weight matrix of the input units.
$V2$: Weight matrix of the context units.
$W$ : Weight matrix.
$L$ : Denotes linear node.
$H$ :Denotes nonlinear node with sigmoidal function.

As it can be seen, the net consists of three layers: An input layer (buffer layer as scales), a single hidden layer and a linear output layer. The neurons in the input layer simply store the scaled input values. The hidden layer neurons perform two calculations. To explain these calculations, consider the general j'th neuron in the hidden layer shown in figure (2). The inputs to this neuron consist of a ni–dimensional vector (ni is the number of the input nodes). Each of the inputs has a weight V1 and V2 associated with it. The first calculation within the neuron consists of calculating the weighted sum $net_j$ of the inputs as:

$$net_j = \sum_{i=1}^{ni} V1_{j,i} \times U_i + V2_{j,i} \times h_i^{o} \quad (4)$$

Then the output of the neuron $h_j$ is calculated as the continuous sigmoid function of the $net_j$ as:

$$h_j = \mathrm{H}(net_j) \quad (5)$$

$$\mathrm{H}(net_j) = \frac{2}{1 + e^{-net_j}} - 1 \quad (6)$$

Once the outputs of the hidden layer are calculated, they are passed to the output layer. In the output layer, a single linear neuron is used to calculate the weighted sum (neto) of its inputs (the output of the hidden layer as in equation (7)).

$$\mathrm{neto}_k = \sum_{j=1}^{nh} W_{k,j} \times h_j \quad (7)$$

where nh is the number of the hidden neuro (nodes) and $W_{kj}$ is the weight between the hidden neuron $h_j$ and the output neuron. The single linear neuron, then, passes the sum ($neto_k$) through a linear function of slope 1 (another slope can be used to scale the output) as:

$O_k = L(neto_k)$ where L(x)=x (8)

Thus the outputs at the output layer are Kp, Ki, & Kd which are denoted by O1, O2, & O3 respectively.

In this work as in [12], the GA with real coding rather than binary is used as follows: Each chromosome is considered as a list (or "vector") of the total weights of neural networks. The encoding is shown in figure (3) and the weights are read off the network in a fixed pre-defined order and placed in a vector. Each "gene" in the chromosome is a real number. To calculate the fitness of a given chromosome, the weights in the chromosome are assigned to the links in the corresponding modified Elman networks, the network is run on the training set, and an objective function is returned. An initial population of weights vectors was chosen to be 50 individuals, with each weight being between –1 and +1. The mutation operator adds a random value between –0.5 and +0.5 to the selected weight on the link. The crossover operator two mates vectors and exchanges the information by exchanging a subset of their components. The result is a new pair of vectors, each of which carries components from both of the parent vectors. The mean square of error (MSE) is used as an objective function to be minimized with the GA:

$$MSE = \sum_{k=1}^{Np} \frac{[y_{mr}(k) - y_p(k)]^2}{Np} \qquad (9)$$

where:

Yp(k) is the output of the plant at sample k.

Ymr(k) is the output of the linear model reference at sample k.

Np is the number of the training patterns.

Since the GA maximizes its fitness function, it is necessary therefore to map the objective function (MSE) to a fitness function. The objective-to-fitness transformation is of the form [12, 13,14].

$$fitness = \frac{1}{objectivefunction + m} \qquad (10)$$

Where $m$ is a constant chosen to avoid division by zero.

## 3-Self-Tuning PID Type Modified Elman Recurrent Neural Networks Controller

The control of nonlinear plants is considered in this section. The feedback neural controller is important because it

is necessary to stabilize the tracking error dynamics of the system when the output of the plant is drifted from the input reference. The approaches used to control the plant do not depend on the information available about the plant. The feedback neural controller is used having based on the minimization of the error between the model reference & the actual output plant in order to achieve good tracking of the reference signal and to use minimum effort. In direct model reference adaptive controller (MRAC) with parallel model reference used here for the feedback neural controller, the adjustable parameters of (PID) controller are adapted by genetic algorithm technique, so that, the output of the plant follows the output of the predefined desired model reference [15]. Thus, the integrated control structure that consists of the model reference and a self-tuning PID controller type modified Elman recurrent neural networks brings together the advantages of the neural model with the robustness of feedback. The general structure of the neural controller type can be given in the form of the block diagram shown in figure (4). The PID control configuration is illustrated in figure (5), where Kp is the proportional gain, Ki is an integral gain, and Kd is the derivative gain, which are adjusted to achieve the desired output. The proposed control structure for the self-tuning PID using GA is used to minimize the error function by adjusting the PID gain. The discrete-time version of PID controller is described by [16]:

$$U(k)=u(k-1)+Kp[e(k)-e(k-1)]+Ki\,e(k)+Kd[e(k)-2e(k-1)+e(k-2)] \quad (11)$$

Where Kp, Ki, & Kd denote the PID gains.

$$e(k)=ydes(k)-yp(k) \quad (12)$$

ydes(k) is a desired output.

yp(k) is an actual output.

In order to apply the proposed algorithm of the self-tuning PID neuron-controller, a cost function **MSE** should be minimized and it is defined as equation (9) in section two.

## 4-The Proposed Algorithm For Self-Tuning PID Type Modified Elman Recurrent Neural Networks Controller

The following genetic procedure is introduced for training the modified Elman recurrent neural network controller for the plant to track the reference model trajectory:

Step 1: Initialize the genetic operators: the crossover probability Pc, the

mutation probability Pm, the population size, and the maximum number of generations.

Step 2: Generate the initial population randomly.

Step 3: For each individual in the population, compute the objective function **MSE**, and then calculate the fitness function as in equation (10), where $m$ will be chosen as an input coefficient equal to 1.

Step 4: Put in descending orders all the chromosomes in the current population.

Step 5: Select individuals using hybrid selection method (Roulette Wheel plus deterministic selection). The real coded genetic operators of mutation and (13) over (single point) is applied.

Step 6: Stop if a maximum number of generations of genetic algorithms are achieved, otherwise increment the generations by one and go to Step 3.

## 5- Case Study

In this section, an example is taken to clarify the features of the neural controller explained in section three and apply the algorithm in section four.

**Mathematical Model of the CSTR System:**

In this example, the controller structure is applied to the Continuous Stirred Tank Reactor ***(CSTR)*** process that is described by the following two nonlinear ordinary differential equations [17]. Where $C_a(t)$ is the product (effluent) concentration, T(t) is the reactor temperature, q is the feed flow-rate (assumed to be constant), and $q_c(t)$ is the coolant flow-rate. The remaining model parameters are defined in nominal operating condition as given in appendix.

$$\frac{\partial C_a(t)}{\partial t} = \frac{q}{Vol}(C_{af} - C_a(t)) - K_o \times C_a(t) \times e^{\left(\frac{-E}{RT(t)}\right)}$$

$$\frac{\partial T(t)}{\partial t} = \frac{q}{Vol}(T_f - T(t)) + \frac{(-\Delta H) \times K_o \times C_a(t)}{\rho \times C_p} \times e^{\left(\frac{-E}{RT(t)}\right)}$$

$$\frac{\rho_c \times C_{pc}}{\rho_c \times C_{pc} \times Vol} \times q_c(t) \left(1 - e^{\left(\frac{-h_a}{q_c(t) \times \rho_c \times C_{pc}}\right)}\right) \times (T_{cf} - T(t))$$

The *(CSTR)* considered is one in which an irreversible exothermic reaction A B takes place, the heat of reaction is removed by a coolant medium that flows through a jacket around the reactor figure (6). The objective is to control $C_a(t)$ which can be done by introducing a coolant flow-rate $q_c(t)$ (the manipulated variable) so the temperature can be varied and hence the product concentration is controlled [17]. For the open loop, the response of the *CSTR* for step changes in the coolant flow-rate is shown in figures (7-a & b) respectively.

As shown both the damping and the steady-state gains of the plant vary considerably, depending on the set point, which gives an indication of the highly nonlinear dynamic behavior of the plant. From figures (7-a & b) the signals entering to or emitted from the network have been normalized to lie within (-1 & +1) in order to overcome numerical problems that is involved within real values. Scaling function has to be added at the neural network terminals to convert the scaled values to actual values and vice versa.

**Simulation Results:**

In this simulation, the proposed control scheme is applied to the CSTR model. The real-coded genetic algorithm is set to the following parameters:

Population size ($N_{POP}$) is equal to 50.

Crossover Probability (Pc) is equal to 0.8.

Mutation Probability (Pm) is equal to 0.05.

Maximum number of generations is 1500.

A continuous time model representation is adopted to be numerically solved using the Runge Kuta fourth order method where the time constant is equal to 1min and the simulation step size for this purpose h=0.1min. The training pattern (Np) used was taken as 350 as the desired trajectory. The modified Elman recurrent neural networks are used to minimize the performance error between the model reference and the actual output. The equation of the model reference is taken from [17] for more stability and without any oscillation in the response:

$$y_{mr}(k+1) = 0.3y_{mr}(k) + 0.7y_{des}(k+1) \quad (14)$$

Convergence is achieved when the performance error falls below a pre-specified value.

$$e_{mr}(k+1) = y_{mr}(k+1) - y_p(k+1) \quad (15)$$

where $e_{mr}(k)$ is the model reference error.

After training it can be observed that the actual output of the plant is following the desired trajectory (model reference) that can be shown as in figure (8), and the feedback control action as shown in figure (9). Moreover, the gains of the PID self-tuning neural controller as scale function are shown in figures (10-a, b, & c) Kp, Ki, & Kd respectively. The error between the desired output and the actual output of the plant is very small as shown in figure (11). Figure (12) describes the best objective function ***MSE***.

## 6- Conclusion

The structure of the modified Elman recurrent neural network as a self-tuning adaptive PID controller with genetic algorithm is shown as the proposed structure of controller and is successfully simulated to nonlinear system in the example. PID feedback controller with self-tuning neural to adjust the parameters (Kp, ki, Kd) of the controller is used. The output of the plant follows the output of the predefined and desired input "model reference", and genetic algorithm is used to give the controller minimum time and more stability and exclude oscillation with best parameters of the controller. The proposed control structure has shown the ability to minimize the error between the desired output model reference and the actual output of the plant as well as the control action, and has also shown excellent set point tracking, as it was clear when applied to the example.

### Appendix

Nominal CSTR Operating Condition

| Para-meter | Description | Nominal Value |
|---|---|---|
| $q$ | Process flow-rate | $100\ \mathrm{l\,min^{-1}}$ |
| $C_{af}$ | Intel feed concentration | $1\ \mathrm{mol\,l^{-1}}$ |
| $T_f$ | Feed temperature | 350K |
| $T_{cf}$ | Inlet coolant temperature | 350K |
| vol | Reactor volume | 100 l |
| $h_a$ | Heat transfer coefficient | $7\times10^{10}\ \mathrm{cal\,min^{-1}.K^{-1}}$ |
| $k_o$ | Reaction rate constant | $7.2\times10^{10}\ \mathrm{min^{-1}}$ |
| $E/R$ | Activation energy | $9.95\times10^{3}\ \mathrm{K}$ |
| $\Delta H$ | Heat of reaction | $-2\times10^{5}\ \mathrm{cal\,mol^{-1}}$ |
| $\rho, \rho_c$ | Liquid densities | $1000\ \mathrm{g\,l^{-1}}$ |
| $C_p, C_{pc}$ | Specific heats | $1\ \mathrm{cal\,g^{-1}.K^{-1}}$ |
| $q_c$ | Coolant flow-rate | $103.41\ \mathrm{l.min^{-1}}$ |
| T | Reactor temperature | 440.2K |
| $C_a$ | Product concentration | $8.36\times10^{-2}\ \mathrm{mol\,l^{-1}}$ |

Note:

Process time constant= $\frac{\mathrm{Vol}}{q} = \frac{100\mathrm{l}}{100\mathrm{l.min^{-1}}} = 1\mathrm{min}$

## References

[1] L.S.Co, O.M.Al. "Design and Tuning of Intelligent and Self-Tuning PID Controller". Control Engineering International. University of Ceara, 2000.

[2] K.J. Astrom and B. Wittenmark, "Adaptive Control" Addison-Wesley Publishing Company, 1989.

[3] T.L. Chia, "Some basic approaches for self-tuning controllers" Control Engineering International, pp. 49-52, Dec. 1992.

[4] D.P. Kwok, P. Tam, C.K. Li. And P. Wang, "Linguistic PID controller" in Proc. of 11th IFAC world Congress, Tallinn, Estonia, USSR, 1990, vol.7, pp.192-197.

[5] L.S. Coelho, H. Simas, and A.A.R. Coelho, "Design issues and laboratory experiments in a self-tuning control teaching", in Proc. of 14th IFAC world congress, Beijing, China, vol. M, pp. 217-222, 1999.

[6] T. Yamanoto, S. Omatu, and M. Kaneda, ”A design method of self-tuning PID controllers”, in proc. of American control conference, Baltimore, Maryland, vol. 3, pp. 3263-3267, 1994.
[7] J. A. Mendez, L. Acosta, ”An application of a neural self-tuning controller to an over head crane”, Neural compute and Application. Vol. 8, pp. 143-150, 1999.
[8] T.O. Mahony and C.J. Downing, “Genetic algorithm for PID parameter optimization: minimizing error criteria” Cork Institute of Technology, 2000.
[9] Chipper field A.J., P.J. Flemming, & C.M. Fonscea, “Genetic algorithm for control system engineering“, proc. Adapt. Comp. in Eng. Design and Control, pp. 128-133, September 1994.
[10] Rodrigo C. & Raimundo S. ”An approximate minimum BER approach to multi-user detection using recurrent neural networks” IEEE, 2002.
[11] S. Haykin, “Neural Networks: A Comprehensive Foundation” 2nd Edition, Prentice-Hall, 1999.
[12] Al-Said I.A.M., ”Genetic Algorithm Based Intelligent Control”, Ph.D. Thesis, University of Technology, Jan. 2000.
[13] Jasim M., ”Evolutionary Techniques for Intelligent PID Controllers” M.Sc. Dissertation, University of Technology, March 2001.
[14] Pham D.T. and Xing L., ”Neural Networks for Identification, Prediction and Control”, Springer-Verlag, London Ltd, 3rd Printing, 1997.
[15] K. S. Narendra and S. Mukhopadhyay, “Adaptive control using neural networks and approximate models,” IEEE Trans. Neural Networks, vol. 8, no. 3, pp. 475-485, 1997.
[16] S. Omatu, M. Khalid, and R. Yusof, "Neuro-Control and its Applications". London: Springer-Velag, 1995.
[17] E. P. Nahas, M. A. Henson and D. E. Seborg, “Nonlinear internal model control strategy for neural network models,” Computers Chem. Eng., vol. 16, no. 12, pp. 1039- 1057, 1992.

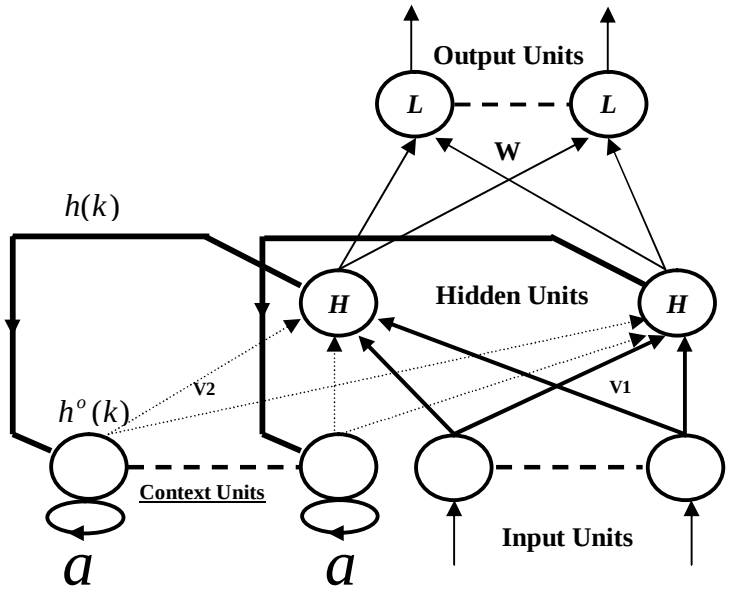


**Fig (1): The Modified Elman Recurrent Neural Networks**

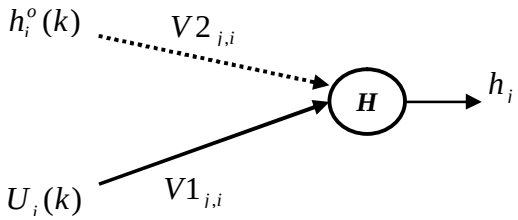


**Fig (2): Neuron j in the hidden layer.**

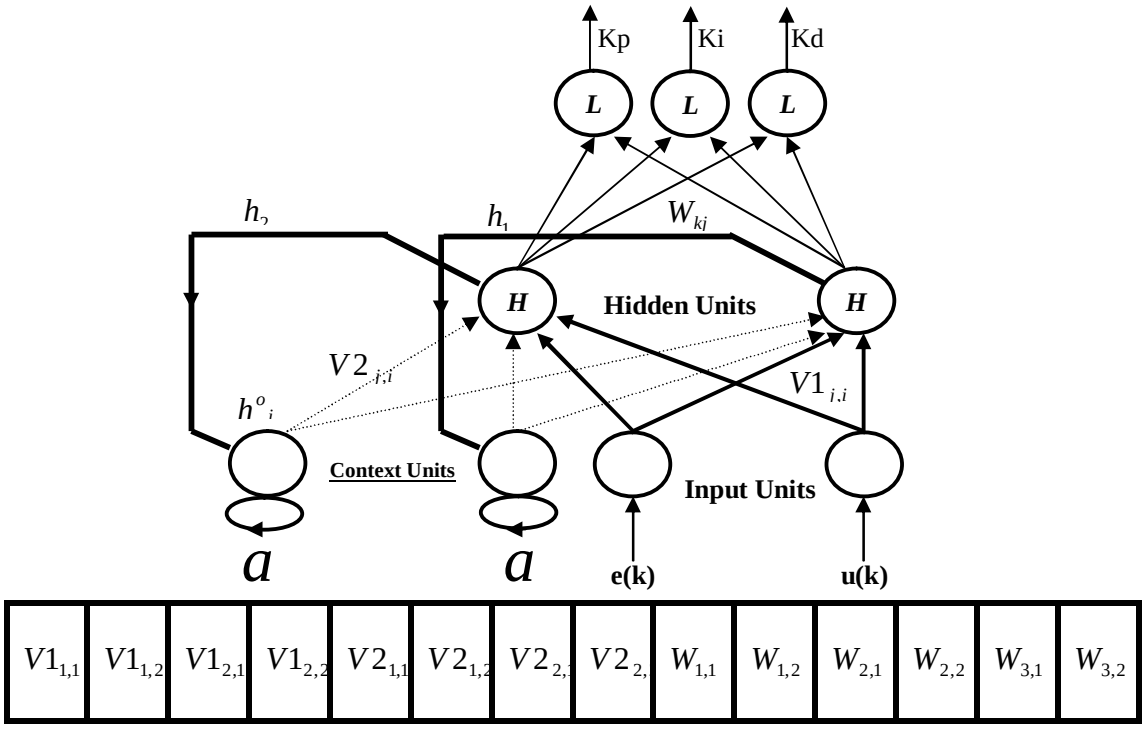


**Fig (3): The weights of modified Elman recurrent neural networks Encoding real-value to the chromosomes**

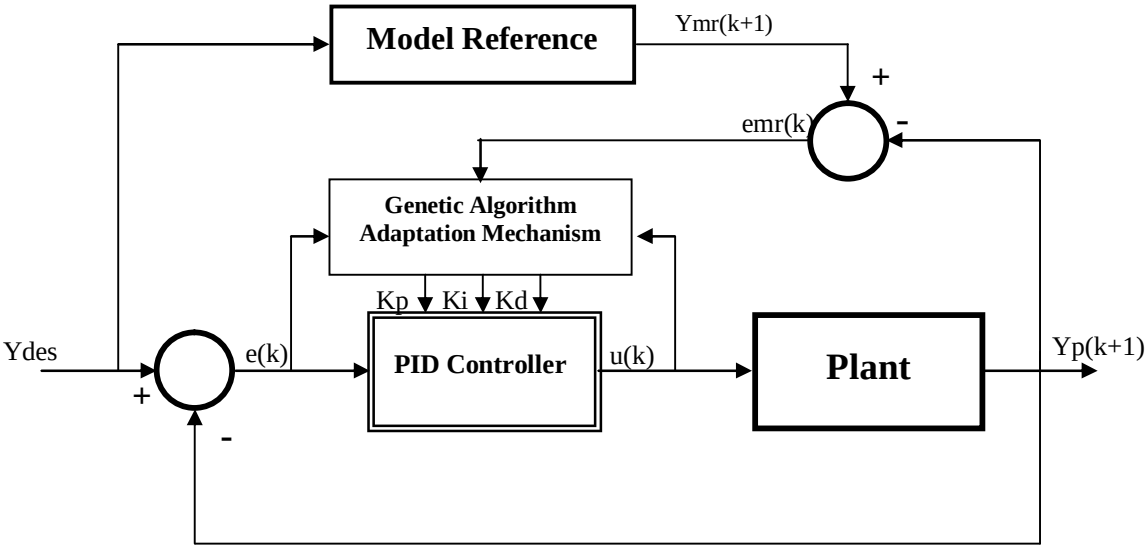


**Fig (4): The General Proposed of Neural Networks Controller**

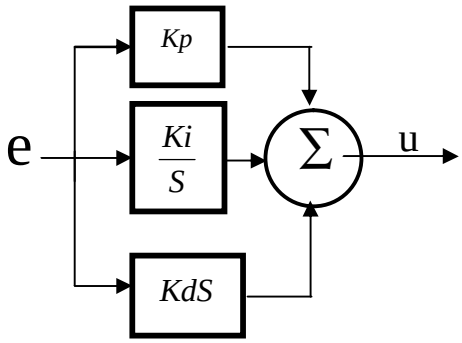


**Fig (5): General Configuration of PID controller**

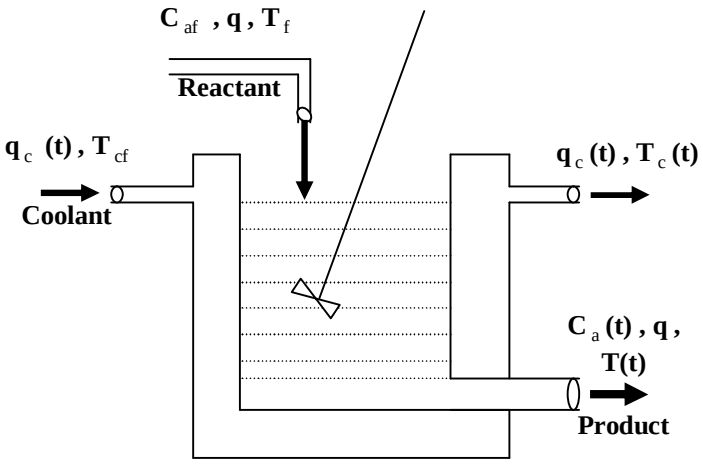


**Fig (6): CSTR with cooling jacket**

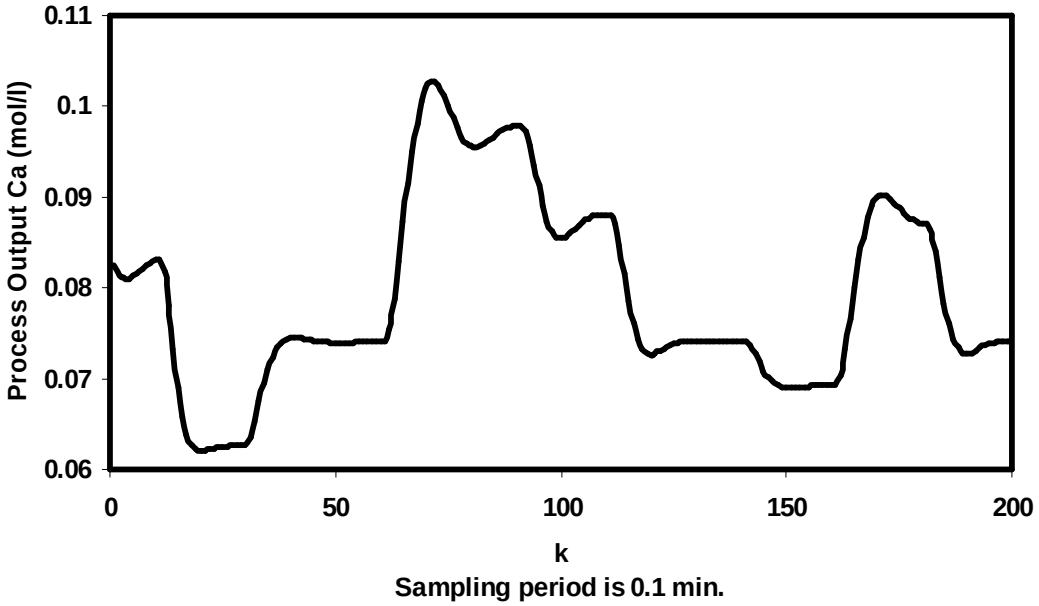


**Fig (7-a): The open loop response of the CSTR**

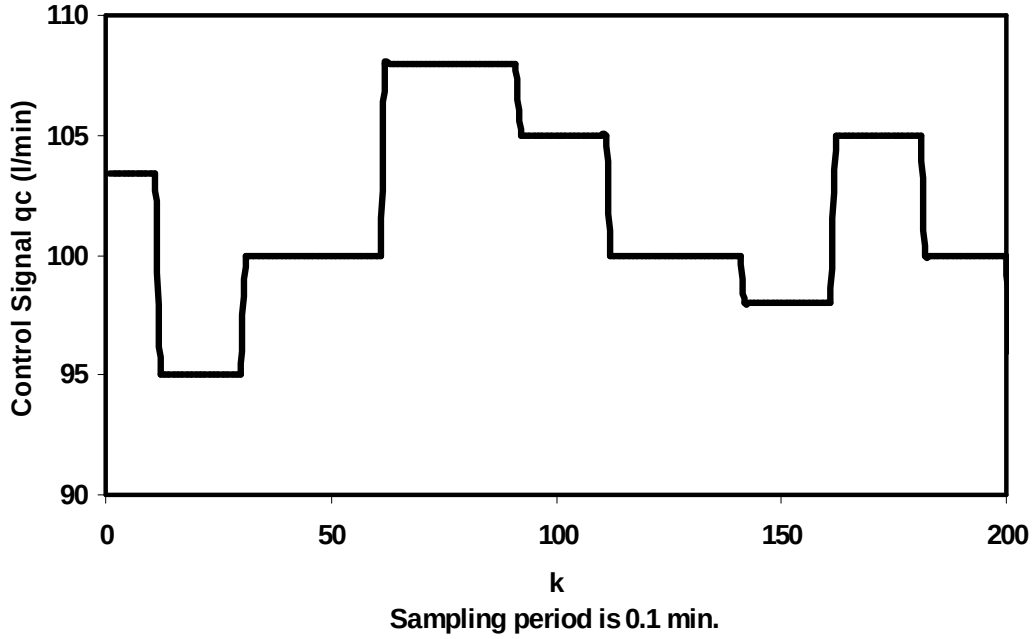


**Fig (7-b): The step changes in the coolant flow-rate**

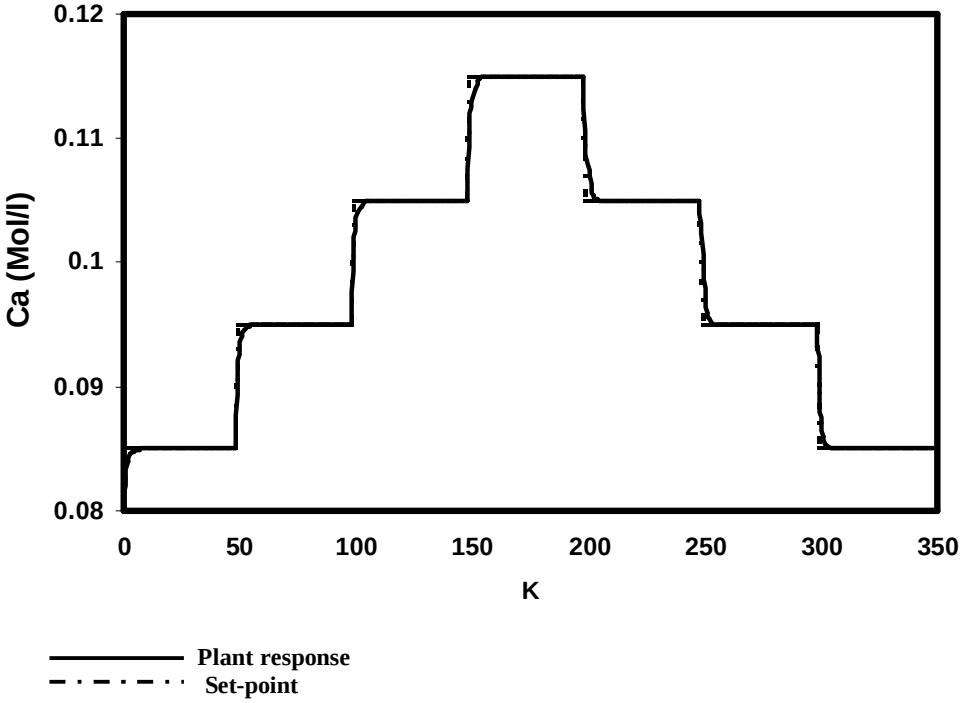


**Fig (8): The response of the plant & the set-point**

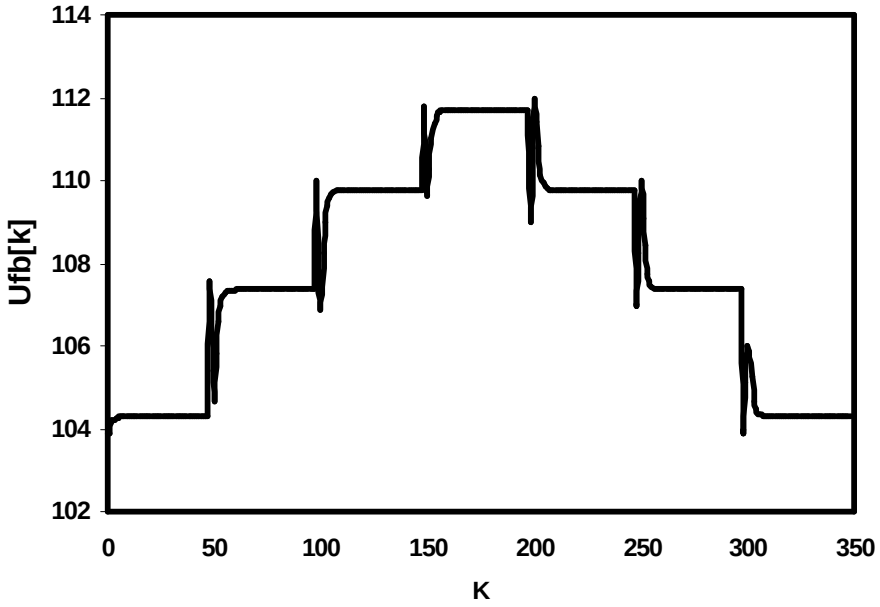


**Fig (9): The feedback control signal of the PID self-tuning controller**

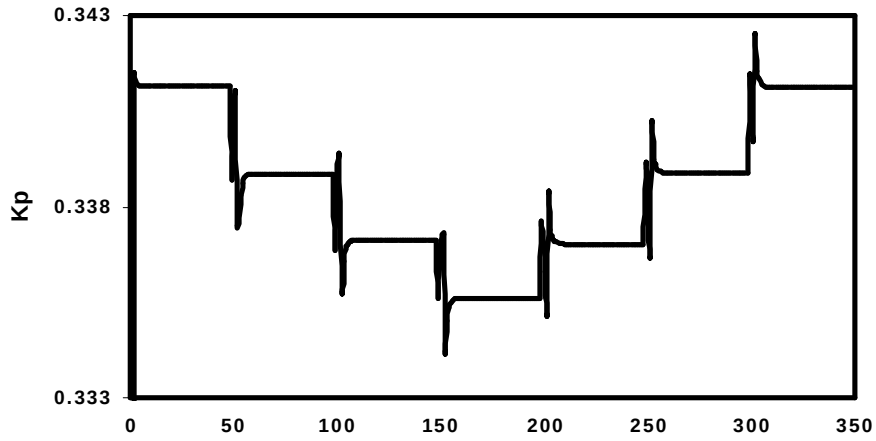


**Fig (10-a): Kp gain of the PID Controller**

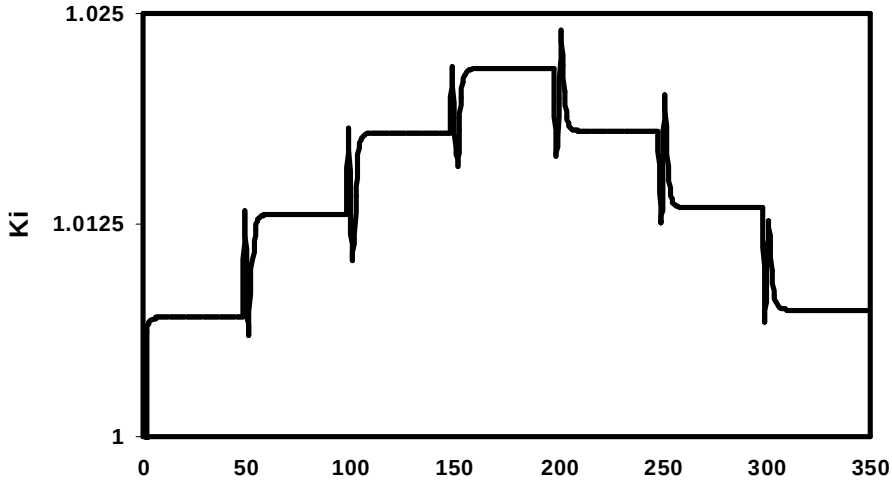


**Fig (10-b): Ki gain of the PID Controller**

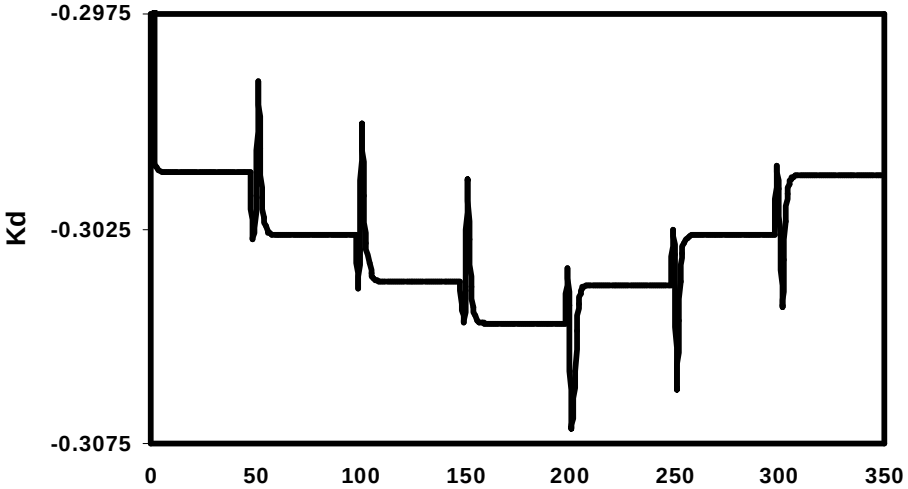


**Fig (10-c): Kd gain of the PID Controller**

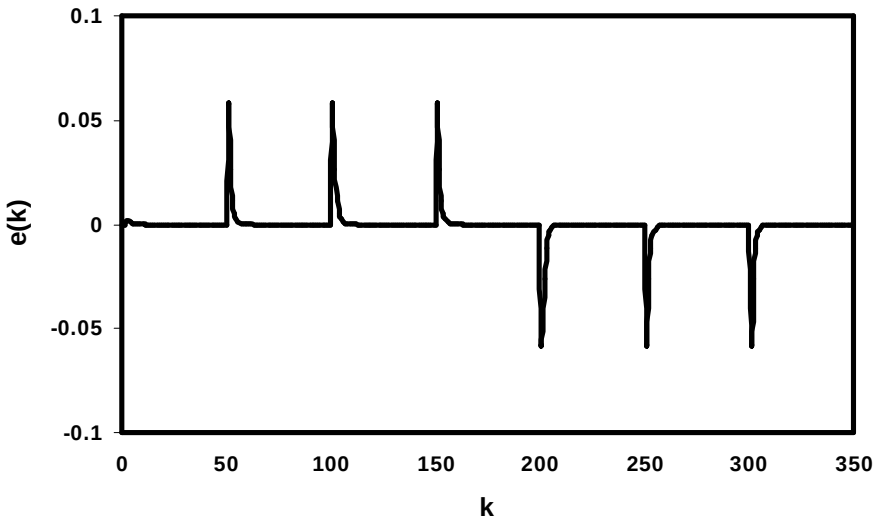


**Fig (11): Output error between the set point desired & the actual output**

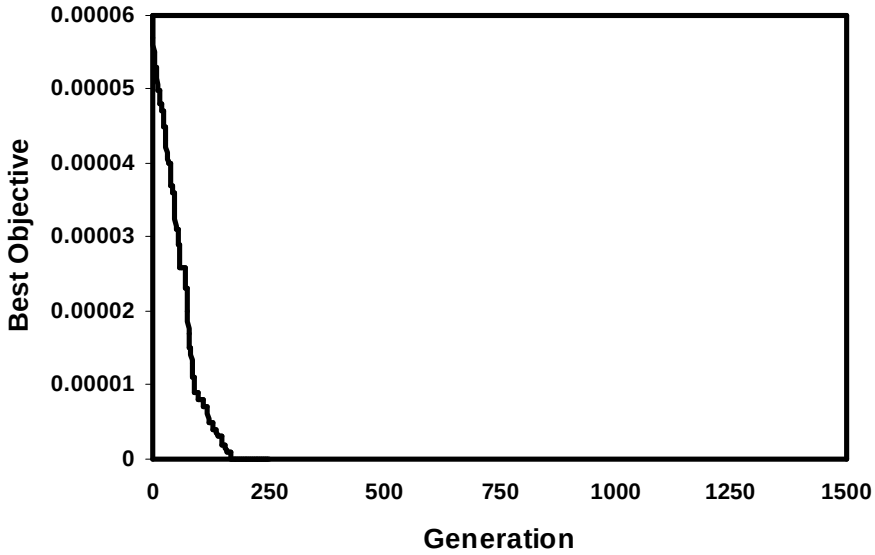


**Fig (12): The best mean square error**