Al-Khwarizmi Engineering Journal, Vol. 8, No. 1, P.P. 19-28 (2013)

Al-Khwarizmi Engineering Journal

Tuning PID Controller by Neural Network for Robot Manipulator Trajectory Tracking

Saad Zaghlul Saeed Al-Khayyt
Department of Mechatronics Engineering/ College of Engineering/ Mosul University
Email: alkhyaat@yahoo.com

(Received 7 May 2012; accepted 18 February 2013)

Abstract

Ziegler and Nichols proposed the well-known Ziegler-Nichols method to tune the coefficients of PID controller. This tuning method is simple and gives fixed values for the coefficients which make PID controller have weak adaptabilities for the model parameters variation and changing in operating conditions. In order to achieve adaptive controller, the Neural Network (NN) self-tuning PID control is proposed in this paper which combines conventional PID controller and Neural Network learning capabilities. The proportional, integral and derivative (K_P, K_I, K_D) gains are self tuned on-line by the NN output which is obtained due to the error value on the desired output of the system under control. The conventional PID controller in the robot manipulator is replaced by NN self tuning PID controller so as to achieve trajectory tracking with minimum steady-state error and improving the dynamic behavior (overshoot). The simulation results showed that the proposed controller has strong self-adaptability over the conventional PID controller.

Keywords: *PID controller, Neural Network, Self tuning controller, Robot manipulator, Trajectory tracking.*

1. Introduction

Robotics has become recently an interesting area of research, especially the robot manipulator control. Industrial robot manipulators of high accuracy require complicated methods of control. A robot manipulator is a nonlinear system with high coupling terms whose dynamics consists of uncertainty and is encountered with payload changes, friction and disturbance [1]. Applying a control technique is important to guarantee high efficiency and lower error for the motion of the robot.

Research of modern control theory is prompted recently and many methods have been proposed for the design of controllers. PID controller is considered the most control technique that is widely used in control applications. It provides robust and reliable performance for most systems if the coefficients are tuned properly [2]. Among the existing tuning techniques, the Ziegler-Nichols formula may be the most well-known method. But tuning is laborious and time-consuming, particularly for processes with serious nonlinearities. Therefore, this method usually needs retuning before being used to control industrial process [3]. However PID controllers cannot provide a general solution to all control problems when the processes are complex and time-variant, with delays and non-linearity, often with poorly defined dynamics and measurement noise [4]. Therefore, an operator is still needed to have control over the plant.

To enhance the capabilities of traditional PID tuning techniques, several methods such as embedding the Lyapunov stability criterion [5], the neural networks [6-12], the fuzzy logic controllers [13-15], and the Genetic algorithm (GA) [16,17] have been developed recently to tune the parameters of PID controllers. In these previous works, it was shown that better control performance can be achieved in comparison with the Ziegler-Nichols method. However if PID controller parameters are tuned off-line, these controllers will not maintain the desired control performance and stability during operation [18]. Thus on-line tuning is proposed in the works of [7,9,12] using multilayered neural network with model reference and also occasionally with estimator. Also for PID gains tuning of nonlinear multivariable systems, GAs offers a solution in combination with multilayered several neural networks for each design criterion in addition to the iterations involved in the algorithm [17].

Neural network (NN) has a strong self-adaptability, learning ability, and nonlinear mapping capability. To work on-line is actually a challenging task for strongly nonlinear systems and frequently considered important than off-line. In on-line procedure, data is simultaneously used to get current output and to optimize the parameters at the same time [19].

In order to achieve good control performance, the NN self tuning PID controller method is proposed. This method combines conventional PID controller and single layer NN. The NN tunes the K_P, K_I, and K_D gains of the PID controller in on-line procedure. By doing so, we overcame the tuning limitation of PID controller using classical tuning methods.

2. Neural Network PID Self Tuning

For undetermined system's model, experimental methods such as Ziegler-Nichols tuning rules may be used to design PID controller. Let the transfer function of a PID controller be written as

$$GC(s) = \mathrm{Kp}\ (1 + \frac{1}{s\ \tau_i} + s\ \tau_d) \qquad \ldots(1)$$

where K_P, τ_i , τ_d are the proportional gain, the integral time, and the derivative time, respectively [20]. Initially the derivative and integral terms are set to zero. The proportional gain is increased from zero to a critical gain value (K_{cr}) where the system exhibits sustained oscillations. Using the value of critical gain (K_{cr}) and the period of oscillation (P_{cr}), the value of parameters K_P, τ_i, τ_d are determined according to the formulas given in Table 1.

Table 1,
Ziegler-Nichols Tuning Rules Based on Critical Gain and Period of Oscillation [20].

Controller	K_P	τ_i	τ_d
P	0.5 K_{cr}	∞	0
PI	0.45 K_{cr}	1/12 P_{cr}	0
PID	0.6 K_{cr}	0.5 P_{cr}	0.125 P_{cr}

Generally for robot arm, the only way to build a high-performance control system is to make use of feedback from joint sensors. Typically, this feedback is used to compute any servo error by finding the difference between the desired and the actual position and that between the desired and the actual velocity.

This paper proposes two inputs – single output self tuning of a PID controller as shown in Fig.1. The controller design uses the desired trajectory and the error as inputs to self tuning, and the NN output (y(t)) as output. The error signal (e) is defined as the difference between the desired position (θ_r) and actual position (θ) of joint angle for robot arm. The NN output is added to the conventional PID controller to adjust the gains of the PID controller on-line according to the change of the error signal. This ensures that the NN will provide the required change in the control signal (u) on-line. Now the control action of the PID controller after using the NN output (y(t)) can be described as:

$$u = K_{PNN}\ e(t) + K_{INN} \int e\ dt + K_{DNN} \frac{d\,e(t)}{dt} \qquad \ldots(2)$$

where K_{PNN}, K_{INN}, and K_{DNN} are the new gains of PID self tuning controller and equal to:
$K_{PNN} = y(t) \cdot K_P$, $K_{INN} = y(t) \cdot K_I$,
$K_{DNN} = y(t) \cdot K_D$.

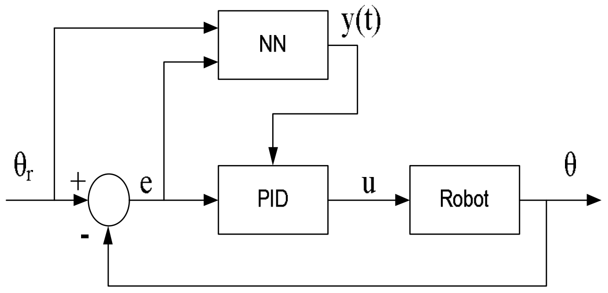


Fig. 1. Proposed Neural Network Self-Tuning PID Controller.

3. Adaptive Linear Network

The ADALINE has been and one of the most widely used neural networks found in practical applications. However, as single-layer linear networks are just as capable as multilayer linear networks. A linear neuron uses linear transfer function termed purelin which simply returning the value passed to it. The ADALINE network shown in Fig.2 below has one layer of S neurons connected to *R* inputs through a matrix of weights W [21]. The output is obtained as:

$$a = \text{purelin}(W \cdot P + b) = W \cdot P + b \quad \ldots(3)$$

where
a = network output vector of S rows,
W = weight matrix of S rows and P columns,
b = bias vector of S rows.

This neuron can be trained to learn an affine function of its inputs, or to find a linear approximation to a nonlinear function. The least mean square error algorithm adjusts the weights and biases of the ADALINE so as to minimize this mean square error.

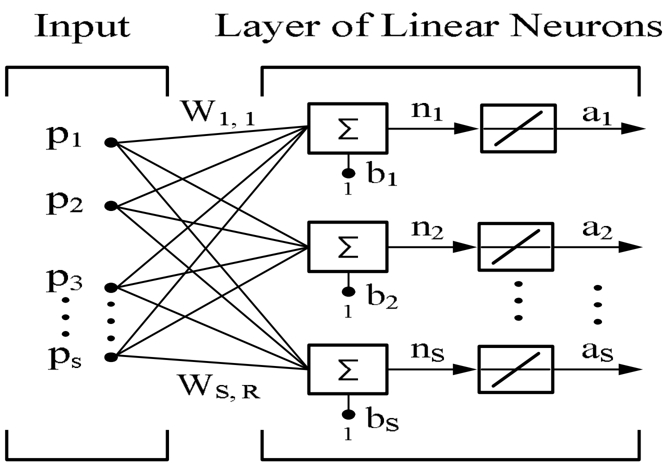


Fig. 2. The ADALINE Network Architecture.

Adaptive filtering is one of its major application areas. A tapped delay line can be combined with an ADALINE network to create the adaptive filter. This adaptive filter is used to predict the next value of a stationary random process, p(t). We use the network shown below to do this (Fig.3). The network changes the weights on each time step so as to minimize the error e(t). If this error is zero, then the network output a(t) is exactly equal to p(t), and the network did its prediction properly.

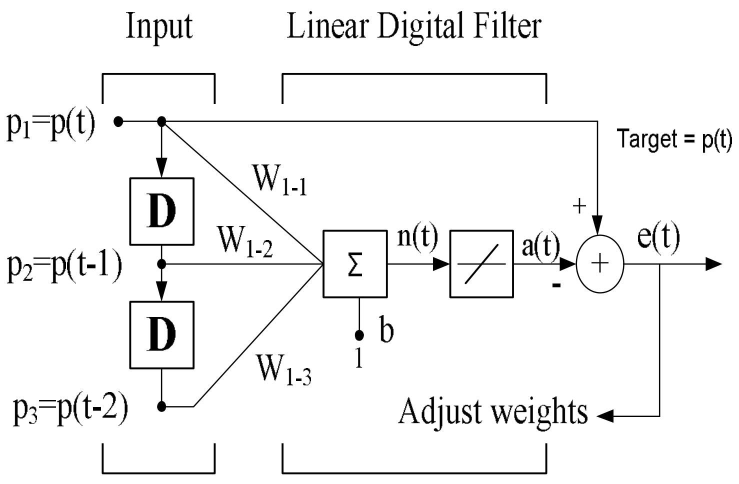


Fig. 3. Adaptive Filter Based on ADALINE Network [21].

In the proposed controller, the error signal (e(t)) is defined as the difference between the desired position (θ_r) and actual position (θ) of joint angle for robot arm. The network, once initialized and operating, adapts at each time step on-line to minimize the error and in a relatively short time is able to predict the required output (y(t)).

4. Manipulator's Dynamic Equations

The complete algorithm for computing joint torques from the motion of the joints is composed of two parts. First, link velocities and accelerations are iteratively computed from link 1 out to link n and the Newton−Euler equations are applied to each link. Second, forces and torques of interaction and joint actuator torques are computed recursively from link n back to link 1. The dynamic equation, which is a function of joints angle position (θ), can be written in the form [22]:

$$t = M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta) + F(\dot{\theta}) \qquad \ldots(4)$$

where M(θ) is the n x n mass matrix of the manipulator, $V(\theta,\dot{\theta})$ is an n x 1 vector of centrifugal and Coriolis terms, G(θ) is an n x 1 vector of gravity terms, $F(\dot{\theta})$ is an n x 1 vector of friction, and τ is an n x 1 vector of joints torque. For two-link planar manipulator (Fig.4), the above matrices become:

$$M(\theta)=\begin{bmatrix} (m_1+m_2)l_1^2+m_2\,l_2^2+ \\ 2m_2l_1l_2cos(\theta_2)\,m_2l_2^2+ \\ m_2l_1l_2cos(\theta_2) \\ m_2l_2^2+m_2l_1l_2cos(\theta_2)\quad m_2l_2^2 \end{bmatrix}$$

$$V(\theta(\dot{\theta})=\begin{bmatrix} -m_2l_1l_2\dot{\theta}_2^2\ sin(\theta_2)- \\ 2m_2l_1l_2sin(\theta_2)\,\dot{\theta}_1\dot{\theta}_2 \\ m_2l_1l_2\ sin(\theta_2)\,\dot{\theta}_1^2 \end{bmatrix}$$

$$G(q)=\begin{bmatrix} (m_1+m_2)gl_1cos(\theta_1)+ \\ m_2gl_2cos(\theta_1+\theta_2) \\ m_2\,g\,l_2cos(\theta_1+\theta_2) \end{bmatrix}$$

$$F(\dot{\theta})=\begin{bmatrix} \mu_k\dot{\theta}_1 & 0 \\ 0 & \mu_k\dot{\theta}_2 \end{bmatrix}$$

where m_1 & m_2 = Mass of link one and two respectively,
l_1 & l_2 = Length of link one and two respectively,
μ_k = Coefficient of Kinetic friction.

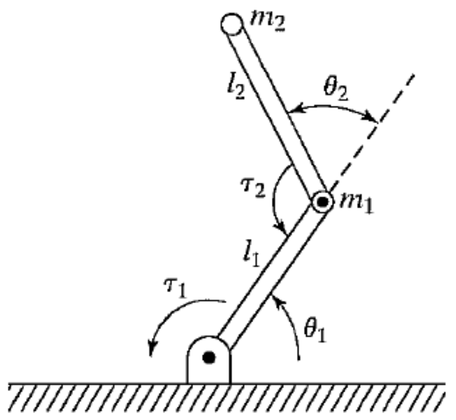


Fig. 4. Two-link Planar Manipulator with point Masses at Distal Ends Links.

5. Simulation and Results

Simulation results are obtained so as to test the proposed NN self tuning PID controller. We compared the results of PID classical tuning method and NN self tuning method in terms of overshoot, transient response, and steady state error. The tracking path from the initial position to the final position is considered in this paper. A tapped delay of 4 is used for all simulations. All simulations were presented using MATLAB and SIMULINK, which are used widely in control applications. The time step used in simulation is 0.001 second.

At first, the response of step desired joint angle position input is tested for single-link robot manipulator [23]. The dynamic equation is given as

$$\frac{d^2\theta}{d\,t^2}+2\,\frac{d\,\theta}{d\,t}+10\sin(\theta)=t \qquad \ldots(5)$$

Large overshoot, settling time, and steady state error are obtained for conventional PID controller as shown in Fig.5. PID gains are: K_P=45, K_I=65 and K_D= 4. The proposed NN self tuning improves the response through reducing the settling time, overshoot, and steady state error. The output of the NN is increased so as to reduce the error as shown in Fig.6. PID gains are: K_P=45 and K_D= 4. The transient mean square error (MSE) and root mean square (RMS) error are reduced from (0.03138) and (0.14349) to (0.00978) and (0.09887), respectively. The results of comparison are presented in Table 2.

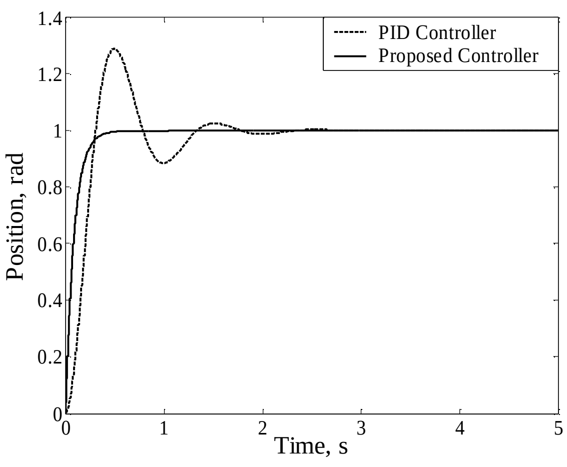


Fig. 5. Response for Step Input of Conventional PID and Proposed NN Self Tuning of PID.

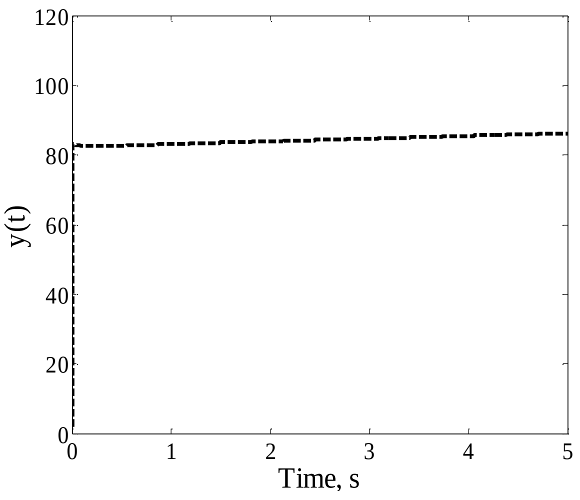


Fig. 6. Variation of NN Output (y(t)) for Desired Step Input.

Table 2,
Results of Comparison for Single-Link Robot Manipulator.

Controller	Delay time (sec.)	Rise time (sec.)	Maximum overshoot %	Settling time (sec.)	MSE (rad)	RMS (rad)
PID	0.1810	0.31	28.8	1.500	0.03138	0.14349
Proposed NN self tuning PID	0.0657	-	-	0.267	0.00978	0.09887

Then two-link robot arm is considered to follow a circular trajectory (Figs 7-10). The robot links' parameters are presented in Table 3. When the controller is PD only as in [24], the system has bad tracking. Therefore PID controller is used and the proportional gain (K_P) is increased to ten times. The PID gains are: K_P = [310 0; 0 450], K_I=[400 0; 0 635], and K_D = [60 0; 0 80]. The actual trajectories for link 1 and link 2 are shown in Figs 8.a and 9.a, respectively. The system has transient RMS error of (0.02445) for link 1 and (0.02333) for link 2, respectively as shown in Figs 8.c and 9.c.

The proposed NN self tuning control structure uses separate network for each link. This improves the obtained trajectory because any noise or disturbance if it is subjected to one link will not affect on the other. The results for the proposed NN self-tuning are shown in Figs 8.b and 9.b. The transient RMS tracking error is reduced to (0.02077) for link 1 and (0.037045) for link 2, respectively as shown in Figs 8.c and 9.c. Also the percentage error after 16 seconds of time is reduced from (1.17) to (0.014) and from (0.91) to (0.002) for link 1 and link 2, respectively. The desired circular trajectory performance of the proposed controller is shown in Fig.10a. The obtained trajectory after two cycles of learning is presented in Fig.10b. The simulation results are summarized in Table 4.

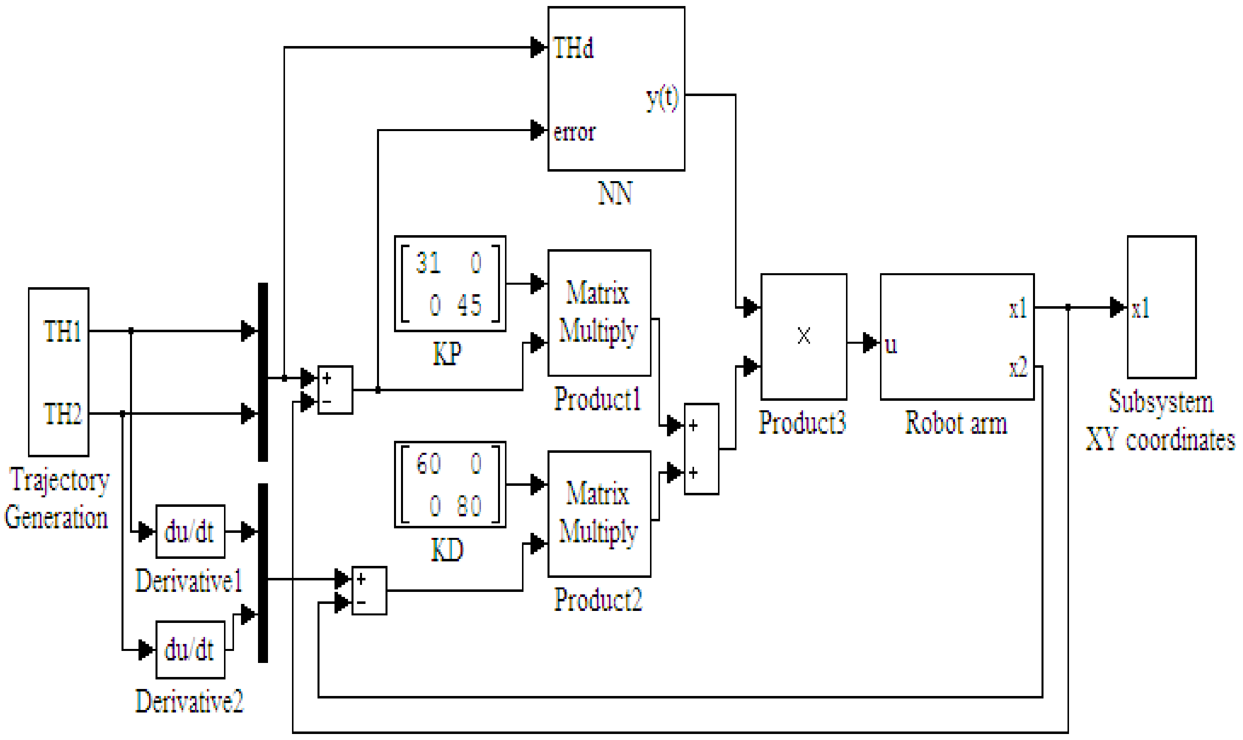


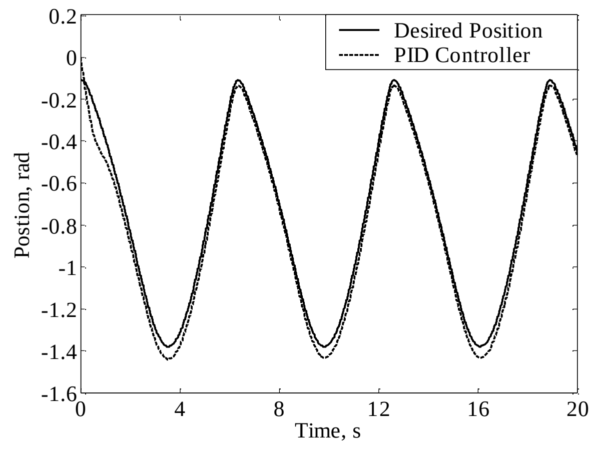
Fig. 7. Simulation Diagram for Robot Manipulator.

Table 3,
Robot's Links Parameters [24].

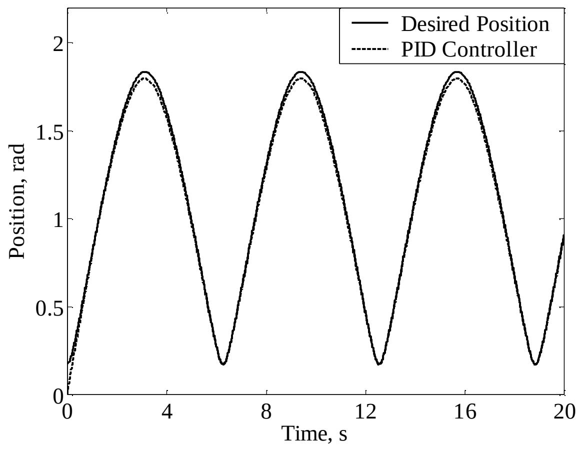
Link	**Mass (kg)**	**Length (m)**	**Friction, μ_k**	**K_P**	**K_D**
1	1	1	0.3	31	60
2	1	2	0.3	45	80

Table 4,
Results of Comparison for Two-Link Robot Manipulator.

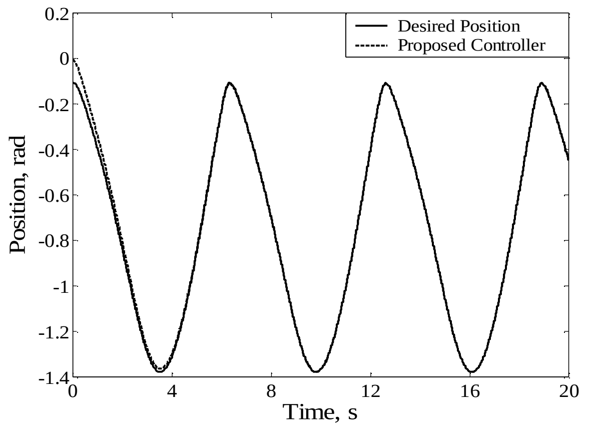
Controller	**Performance (rad)**	**Link 1**	**Link 2**
	RMS error	0.02445	0.02333
PID	Error after 16 sec.	0.01614	0.01521
	Error after 16 sec. %	1.17	0.91
	RMS error	0.02077	0.037045
Proposed NN self tuning PID	Error after 16 sec.	1.9202e-04	3.7218e-05
	Error after 16 sec. %	0.014	0.002



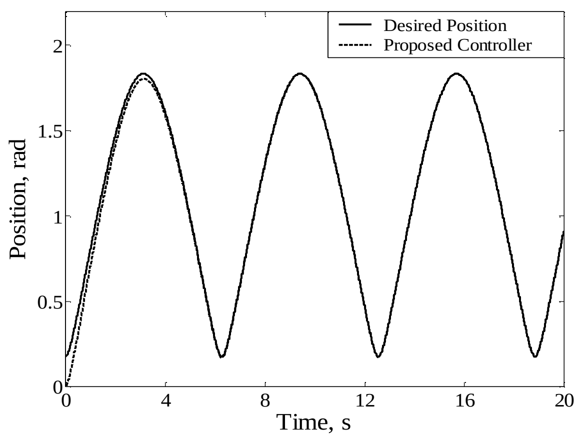
a



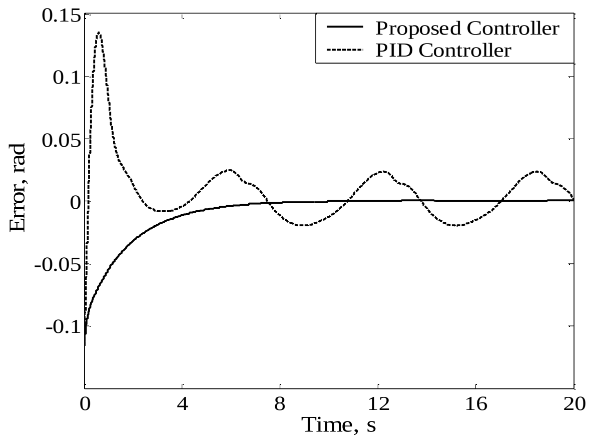
a



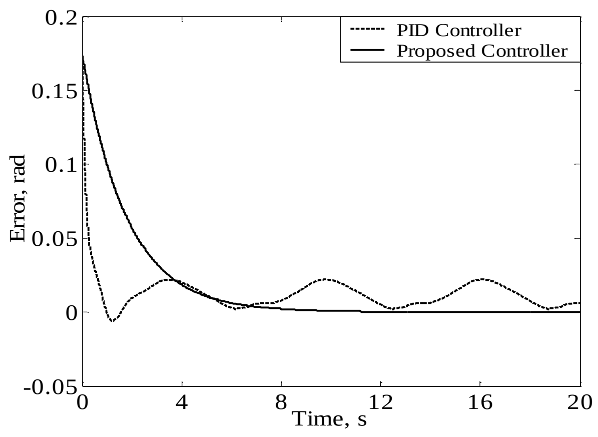
b



b



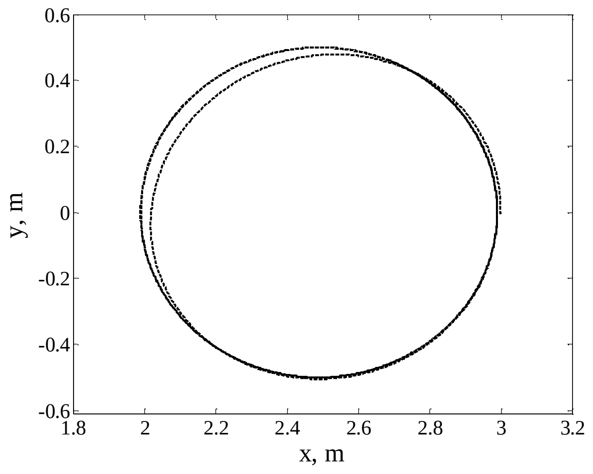
c



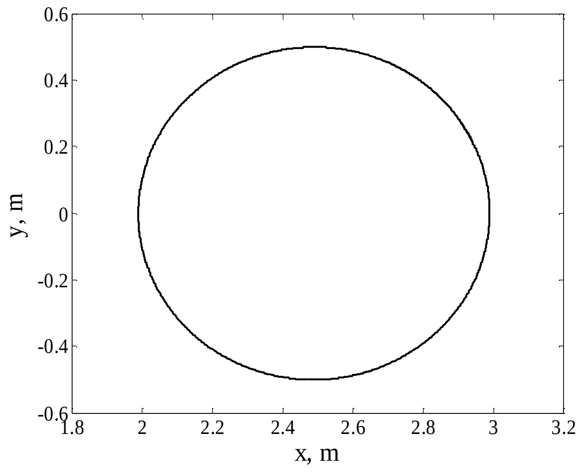
c

Fig. 8. Link 1 Performance for Desired Trajectory.

Fig. 9. Link 2 Performance for Desired Trajectory.



a



b

Fig. 10. Circular trajectory.
a-Trajectory for 20 sec.; b- Trajectory after 2 Cycles of Learning

6. Conclusion

Neural network self tuning PID controller provides a different way to approach a control problem. This method focuses on what the system should do rather than trying to model how it works. One can concentrate on solving the problem rather than trying to model the system mathematically, if that is even possible.

The application of a NN tuning to nominal controller designed using Ziegler-Nichols tuning rules is studied in this paper. It is shown that the addition of a NN tuning improves the response of a nominal PID controlled system through gradual increase in the controller gains. The advantage of the NN self tuning is its adaptively due to on-line training which provides auto-tuning and reducing the amount of tuning required. At the beginning of simulation, the error is maximum because the NN is not learned yet. But after 1 second of time, the error is reduced quickly and the robot starts follow the desired trajectory efficiently.

Simulation studies show that the proposed NN self tuning provides excellent tracking and robustness in the presence of system nonlinearity, since the NN is trained on-line, therefore any changes in system parameters will cause the NN to provide the appropriate change in the control signal to resume the effect. On the other side, we proved that the proposed controller is more efficient to control the robot manipulator to follow the desired trajectory compared to classical tuning method of PID controller.

7. References

[1] Spong, M. W. and Vidiasagar, M., "Robot modeling and Control", Wiley, New York, 1989, pp.231.
[2] Tao Ai, Jun-qi Yu, Yan-feng Liu, and Jiang Zhou, "Study on Neural Network Self-tuning PID Control for Temperature of Active Solar House Heating System", IEEE 2nd International Workshop on Intelligent Systems and Applications, 2010, pp.1-4.
[3] Gawthrop, P. J. and Nomikos, P.E., "Automatic tuning of commercial PID controllers for single-loop and multiloop applications", IEEE Control Systems Magazine, Vol. 10, 1990, pp.34-42.
[4] Jing-Chug Shen and Huann-Keng Chiang, "PID Tuning Rules for Second Order Systems", IEEE 5th Asian Control Conference, Vol.1, 2004, pp.472-477.
[5] Han-Pang Huang and Ching-Hung Lin, "A Stable On-Line Self-Tuning Optimal PID Controller for A Class of Unknown System", Asian Journal of Control, vol. 9, no. 2, pp. 151-162, 2007.
[6] Al-Zohairy, T. A., "Adaptive Control OF Nonlinear Multivariable Dynamical Systems Using RBF Neural Network", Canadian Journal on Automation, Control and Intelligent Systems, Vol.2, No.1, 2001, pp.9-25.
[7] Pirabakaran, K. and Becerra, V.M., "PID Autotuning Using Neural Networks and Model Reference Adaptive Control", 15th Triennial World Congress, Barcelona, Spain, 2002.
[8] Gao Zhenhai and Zhu Bo, "Vehicle Lane Keeping of Adaptive PID Control with BP

Neural Network 'Self-Tuning", IEEE Proceedings on Intelligent Vehicles Symposium, 2005, pp.84-87.

[9] Cordova, H., Wijaya, A. F., "Self-Tuning PID Neural Network Controller to Control Nonlinear Ph Neutralization in Waste Water Treatment", The Journal for Technology and Science, vol. 18, no. 3, 2007.

[10] Jiangjiang Wang, Chunfa Zhang, and Youyin Jing, "Adaptive PID Control with BP Neural Network Self-Tuning in Exhaust Temperature of Micro Gas Turbine", IEEE Conference on Industrial Electronics and Applications, 2008, pp.532-537.

[11] Qing-song Lin, Yu-fei Yao and Jun-xiao Wang, "Simulation and Application of Neural Network PID Auto-tuning Controller in Servo-system", IEEE 2nd International Workshop on Database Technology and Applications, 2010, pp.1-4.

[12] Hendookolaei, A. and Ahmadi, N. A., "PID Controller with Neural Auto Tuner Applied in Drum Type Boilers", Canadian Journal on Electrical and Electronics Engineering, vol. 3, no. 6, 2012.

[13] Copeland, R. P. and Rattan, K. S., "A Fuzzy Supervisor for PD Control of Unknown Systems", IEEE International Symposium on Intelligent Control, 1994, pp.22-26.

[14] Algreer, M.M.F. and Kuraz, Y.R.M., "Design Fuzzy Self Tuning of PID Controller for Chopper-Fed DC Motor Drive", Al-Rafidian Engineering, Vol.16, No.2, 2008, pp.54-66.

[15] Ataei, M. and Shafiei S. E., "Sliding Mode PID-Controller Design for Robot Manipulators by Using Fuzzy Tuning Approach", 27th Chinese Control Conference, Kunming, China, 2008, pp.170-174.

[16] CHIA-JU WU, "Genetic Tuning of PID Controllers Using a Neural Network Model: A Seesaw Example", Journal of Intelligent and Robotic Systems, Vol.25, 1999, pp.43-59.

[17] Azevedo, Ana B. and Ruano, Antonio E., "Neural Networks PID Autotuning with On-Line Adaptation", 16th IMAGS world congree, 2000.

[18] Kui Zhang and Xinyan An, "Design of multivariable self-tuning PID controllers via Quasi-Diagonal Recurrent wavelet Neural Network", 2nd International Conference on Intelligent Human Machine Systems and Cybernetics, Vol.2, 2010, pp.95-99.

[19] Selami, B. and Musa, A., "A New RBF Network Based Sliding-Mode Control of Nonlinear Systems", International Multi conference on Computer Science and Information Technology, Vol.4, 2009, pp.11-16.

[20] Ogata, K., "Modern Control Engineering" , Prentic-hall, Inc., 2002.

[21] Howard Demnth and Mark Beale, "Neural Network Toolbox", The Math Works, Inc., 2008, pp.327-337.

[22] Craige, John J., "Introduction to Robotics: Mechanics and Control", Prentice-Hell, Inc., 2005, pp.180.

[23] Lehuy, H. and Sybille, G., "MATLAB Help ver.7", Math works, Inc., 2004.

[24] Heredia, A. and Wen Yu, "A High-Gain Observer-Based PD Control for Robot Manipulator", American Control Conference, Chicago, USA, Vol.4, 2000, pp.2518-2522.

موالفة المسيطر التناسبي-التكاملي-التفاضلي بالشبكة العصبية لتتبع مسار ذراع روبوت

سعد زغلول سعيد الخياط
قسم هندسة الميكاترونكس/ كلية الهندسة/ جامعة الموصل
البريد الألكتروني: alkhyaat@yahoo.com

الخلاصة

اقترح الباحثان Ziegler وNichols طريقة لتوليف معاملات المسيطر PID. الطريقة بسيطة و تعطي قيم ثابتة للمعاملات مما يجعل المسيطر PID ضعيفا في التكيف للتغير في خواص المنظومة و ظروف التشغيل. لكي ينجز مسيطر متكيف، تم اقتراح في هذا البحث موالفة المسيطر PID بالشبكة العصبية (NN) و الذي يجمع المسيطر PID التقليدي مع قابلية التعلم للشبكة العصبية. ان معاملات المسيطر PID و هي المكاسب K_P، K_I ، K_D يمكن ان تولف أنيا بواسطة إخراج الشبكة العصبية الذي ينتج تبعا للخطأ في إخراج المنظومة تحت السيطرة. تم استبدال المسيطر PID التقليدي لذراع الروبوت بالمسيطر المقترح لكي ينجز تتبع المسار بأقل خطأ و تحسين التصرف الديناميكي (تجاوز الحد). استخدمت المماثلة عبر الحاسوب الآلي و أظهرت النتائج ان المسيطر المقترح يمتلك تكيفا ذاتيا متفوقا على المسيطر PID التقليدي.