An Adaptive Position Synchronization Controller Using Orthogonal Neural Network for 3-DOF Planar Parallel Manipulators

Chapter *in* Lecture Notes in Computer Science · July 2017

DOI: 10.1007/978-3-319-63315-2_1

3 authors:

Quang Dan Le
University of Ulsan

Hee-Jun Kang
University of Ulsan

Tien Dung Le
The University of Danang - University of Scien...

An Adaptive Position Synchronization Controller Using Orthogonal Neural Network for 3-DOF Planar Parallel Manipulators

Quang Dan Le[1], Hee-Jun Kang[2(✉)], and Tien Dung Le[3]

[1] Graduate School of Electrical Engineering, University of Ulsan, Ulsan, South Korea
ledanmt@gmail.com

[2] School of Electrical Engineering, University of Ulsan, Ulsan, South Korea
hjkang@ulsan.ac.kr

[3] The University of Danang - University of Science & Technology, Danang, Vietnam
ltdung@dut.udn.vn

Abstract. This paper proposes an adaptive position synchronization controller using orthogonal neural network for 3-DOF planar parallel manipulators. The controller is designed based on the combination of computed torque method with position synchronization technique and orthogonal neural network. By using the orthogonal neural network with online turning gains can overcome the drawbacks of the traditional feedforward neural network such as initial values of weights, number of processing elements, slow convergence speed and the difficulty of choosing learning rate. To evaluate the effectiveness of the proposed control strategy, simulations were conducted by using the combination of SimMechanics and Solidworks. The tracking control results of the parallel manipulators were significantly improved in comparison with the performance when applying non-synchronization controllers.

Keywords: Planar parallel manipulator · Position synchronization controller · Adaptive controller · An orthogonal neural network · Online self-turning

1 Introduction

The past decade has seen a steady increase of parallel manipulators in industrial production due to their advantages over serial manipulators such as high velocity, high accuracy, low moving inertia and strong structure. However, the parallel manipulators gradually reveal weaknesses because of their closed-chain structures. For example, they have complex kinematic, dynamic models and the limited workspace even with many singularities inside. Therefore, the parallel manipulator control needs more advanced technologies.

For parallel manipulator, the motion of end-effector based on the cooperation motion of each sub-manipulator with one another is called synchronization. The technical synchronization is based on cross-coupling error of each sub-manipulator with one another. This is different from the conditional non-synchronized controller

D.-S. Huang et al. (Eds.): ICIC 2017, Part III, LNAI 10363, pp. 3–14, 2017.
DOI: 10.1007/978-3-319-63315-2_1

that concerns only the convergence of each joint tracking error. The cross-error in controller was firstly defined by Koren [1] for CNC machine tool. Then, Koren and his cooperators applied this technique in mobile robot [2]. Upon the definition of cross-error, Lu Ren et al. defined a new cross-error for parallel manipulator [3, 4]. With inspiration synchronization, Liu and Chopra [5] and Rodriguez-Angeles and Nijmeijer [6] applied this technique in multi manipulator cooperated.

The adaptive methods for robot control are based on intelligent technique using neural network that increase wide variety. The Feedforward Neural Networks (FNN) used in adaptive control for robot [7–10] achieve high accuracy and can deal with uncertainties. Authors in [11, 12] combined fuzzy and neural network in control to compromise each other. The majority of these researchers used intelligence control in control systems. But the classical FNN have some drawbacks such as local minimum, initial values of weight, number of processing elements, slow convergence speed and the difficulty of choosing learning rate. In order to solve convergence speed problem, Chen and Manry replaced sigmoid active function by polynomial function [13]. Enrique Castillo and his cooperators used sensitivity analysis technique [14], while Karayiannis and Venetsanopoulos using the fast delta rule algorithm [15]. Yam et al. proposed linear algebraic method in [16] and Wang et al. presented the extreme learning machines in [17]. To resolve the remaining problems in conventional FNN, Yang and Tseng [18] presented Orthogonal Neural Network (ONN) based on feed-forward network and using polynomial function. This neural network handles most of the drawbacks in traditional FNN. Sher et al. [19] also showed comparison between the training method and the numerical method of ONN in function approximation. Stanisa and his group applied successfully ONN in anti-lock braking system and modeling of dynamical systems [20–22]. In addition, ONN was proved to have good results in robot controller by Timmis et al. [23] and Wang [24].

In this paper, an adaptive Position Synchronization Controller Using Orthogonal Neural Network (PSC-ONN) for 3-DOF planar parallel manipulator was presented. In this method, the PSC-ONN consists of computed torque control, error synchronization and ONN, The PSC-ONN deals with the existence of the uncertainty in robotic systems and obtain the higher tracking performance. In addition, the 3-DOF planar parallel manipulator simulation model was established using combination Solidworks and SimMechanics to verify effectiveness of the controller proposed.

The remainder of this paper is arranged as follows. Section 2 describes the dynamics of 3-DOF planar parallel manipulator. Section 3 presents the position synchronization controller. In Sect. 4 the combination of the PSC and the ONN is described for the proposed controller. Simulation results are shown in Sect. 5 and finally, some remarks are addressed in Sect. 6.

2 The 3-DOF Planar Parallel Manipulators

2.1 The Geometry of the 3-DOF Planar Parallel Manipulators

The 3-DOF planar parallel manipulator was depicted in Fig. 1. It includes three active joints, six passive joints and the end-effector. The link lengths of the parallel manipulator

$l_1 = A_iB_i$, $l_2 = B_iC_i$, $(i = 1,2,3)$, the end-effector $C_1C_2C_3$ is equilateral triangle with $l_3 = C_iP$ which is radius of the circle circumscribing the three vertices.

In Fig. 1 denotes: $\theta_a = [\theta_{a1}, \theta_{a2}, \theta_{a3}]^T$ is the vector of active joints angular, $\theta_p = [\theta_{p1}, \theta_{p2}, \theta_{p3}]^T$ is the vector of three important passive joints angular and $P = [x_P, y_P, \phi_P]^T$ is the vector of position of end-effector.

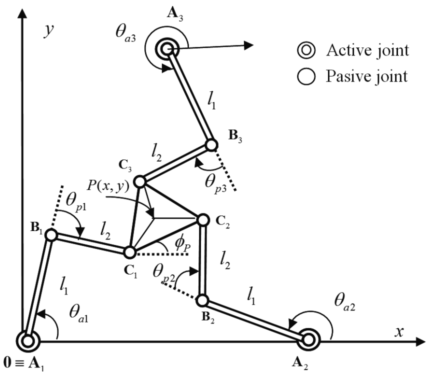


Fig. 1. The kinematics planar parallel manipulator 3-DOF

2.2 Dynamic Models of 3 DOF Planar Parallel Manipulators

Dynamic models of 3 DOF planar parallel manipulators are established using virtually cut and Lagrange-D'alamber principle. The virtually cut showed in Fig. 2, we derive the Lagrangian equation of the open-chain system and compute the joints torques needed to generate a given motion which satisfies the loop constraints.

Dynamic models of each planar serial 2 DOF manipulator and end-effector given by Largane without fiction and external distribution follow:

$$\frac{d}{dt}\left(\frac{\partial L_i}{\partial \dot{\theta}_i}\right) - \frac{\partial L_i}{\partial \theta_i} = \tau_{\theta_i} \tag{1}$$

$$\frac{d}{dt}\left(\frac{\partial L_i}{\partial \dot{X}}\right) - \frac{\partial L_i}{\partial X} = \tau_X \tag{2}$$

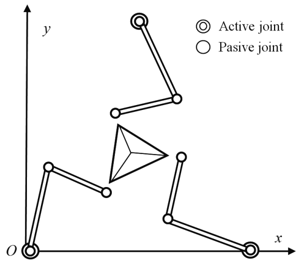


Fig. 2. The open-chain systems of planar parallel manipulator

where L_i the Lagrangian function for each serial manipulator *(i = 1,2,3)*.

$\theta_i = [\theta_{ai}, \theta_{pi}]^T$ is the angle joints vector. $\tau_{\theta_i} = [\tau_{ai}, \tau_{pi}]^T$ is the joint torque vector.
$X = [x_P, y_P, \phi_P]^T$ is the position of the end-effector on Cartesian.
$\tau_X = [\tau_{xP}, \tau_{yP}, \tau_{\phi P}]^T$ is the end-effector torque vector in each direction.

From (1) and (2), using virtual displacement principle we have:

$$\left(\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_a}\right) - \frac{\partial L}{\partial \theta_a} - \tau_a\right)\delta\theta_a + \left(\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_p}\right) - \frac{\partial L}{\partial \theta_p} - \tau_p\right)\delta\theta_p + \left(\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{X}_P}\right) - \frac{\partial L}{\partial X_P} - \tau_X\right)\delta X_P = 0 \tag{3}$$

Rewrite the Eq. (3) with matrix form and actuators only have at active joints we obtain:

$$\left[\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_a}\right) - \frac{\partial L}{\partial \theta_a}, \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_p}\right) - \frac{\partial L}{\partial \theta_p}, \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{X}_P}\right) - \frac{\partial L}{\partial X_P}\right]\left[I, \frac{\partial \theta_p}{\partial \theta_a}, \frac{\partial X_P}{\partial \theta_a}\right]^T = \tau_a \quad (4)$$

Equation 4 can be rewritten:

$$W^T \tau = \tau_a \quad (5)$$

where $\tau = [\tau_a, \tau_p, \tau_X]^T \in \Re^{9\times1}$ is torque of robot system. $W = [I, \partial\theta_p/\partial\theta_a, \partial X_p/\partial\theta_a]^T \in \Re^{9\times3}$ is vector prismatic joint and velocity active joints. This is the similar to the work of Nakamura in [25].

The dynamic of the open-chain system can be arranged in the form

$$M_t(q)\ddot{q} + C_t(q, \dot{q})\dot{q} = \tau \quad (6)$$

where $q = [\theta_a, \theta_p, X_P]^T$ is the vector of joints and end-effector's position. $M_t(q) \in \Re^{9\times9}$ is the inertial matrix. $C_t(q, \dot{q}) \in \Re^{9\times9}$ is the Coriolis matrix.

Substituting (5) and (6) we have

$$W^T(M_t(q)\ddot{q} + C_t(q, \dot{q})\dot{q}) = \tau_a \quad (7)$$

From kinematic constraints in (5) we have:

$$\dot{q} = [I, \partial\theta_p/\partial\theta_a, \partial X_P/\partial\theta_a]^T \dot{\theta}_a = W\dot{\theta}_a \quad (8)$$

After derivative both side of (8), it leads to:

$$\ddot{q} = \dot{W}\dot{\theta}_a + W\ddot{\theta}_a \quad (9)$$

Now, by substituting (8) and (9) into (7) we obtain the dynamic model of the 3-DOF planar parallel manipulator in active joints space:

$$M\ddot{\theta}_a + C\dot{\theta}_a = \tau_a \quad (10)$$

where $M = W^T M_t \dot{W} \in \Re^{3\times3}$ is the inertial matrix of robot. $C = W^T M_t \dot{W} + W^T C_t W \in \Re^{3\times3}$ is the Croliolis matrix.

The Eq. (10) is dynamic model of robot 3-DOF planar parallel manipulator without friction and external disturbances in actuator at three active joints.

3 The Conventional Synchronization Controller

3.1 Problem Statements of the Conventional CTC

The dynamic model (10) did not contain the friction and external disturbances. However in real system robot, the uncertainties term always exists. General equation of robot can be written

$$M\ddot{\theta}_a + C\dot{\theta}_a + F_a + f = \tau_a \tag{11}$$

where F_a is friction at active joints. f is a load disturbance.

The computed torque controller (CTC) is used to control the robot which has dynamic model described by (10) follow a desired trajectory. The structure of the CTC is designed as:

$$\tau_a = M(q)\left[\ddot{\theta}_d + K_e(\theta_d - \theta) + K_v(\dot{\theta}_d - \dot{\theta})\right] + C(\theta, \dot{\theta})\dot{\theta} \tag{12}$$

where $\theta_d \in \Re^{3\times 1}$ is the desired manipulator trajectory, $K_e \in \Re^{3\times 3}$ and $K_v \in \Re^{3\times 3}$ are the proportional gain matrices.

The computed torque controller in (12) have drawbacks in real application such as requirement of an exact dynamic model of robot which usually impossible, lack of robustness to structured and unstructured uncertainties. In addition, the conventional CTC only concerns the position error of each joints and information from other joints is neglected while the uncertainties at each joint is different so they will make time convergence at each joint is different.

This paper thus aims to design an adaptive control scheme for uncertain robot planar parallel manipulator based on CTC, synchronization error, and orthogonal neural network such that the system output will follow the desired trajectories.

3.2 The Position Synchronization Controller

In this part, the position synchronization controller (PSC) based on synchronization error and cross-coupling error is presented. The position tracking error of each active joints is defined:

$$e_i(t) = \theta_i^d(t) - \theta_i(t), \quad i = 1, 2, \ldots, n \tag{13}$$

where $\theta_i^d(t)$ denotes the desired position of the each active joint. In the position synchronization, not only the position error $e_i(t) \to 0$ but also aimed to regulate motion relationship among multiple active joints during the tracking $e_1(t) = e_2(t) = \ldots = e_n(t)$.

The synchronization error are defined [2] of a subset of all possible pairs of two active joints from total of n joints and this case is three active joints in the following $\varepsilon_1(t) = e_1(t) - e_2(t)$, $\varepsilon_2(t) = e_2(t) - e_3(t)$, $\varepsilon_3(t) = e_3(t) - e_1(t)$.

The goal of synchronization error can be achieved if $\varepsilon_i(t) = 0$ for all active joints. Unlike conditional controller non-synchronize that concerns the convergence of position tracking error, in synchronize controller consider error between each other active joints. Specifically, the controller is to control $e_i(t) \to 0$ and at the same time.

The cross-coupling error was defined in [4]

$$e_i^* = e_i + \beta \int_0^t (\varepsilon_i - \varepsilon_{i-1}) dw \tag{14}$$

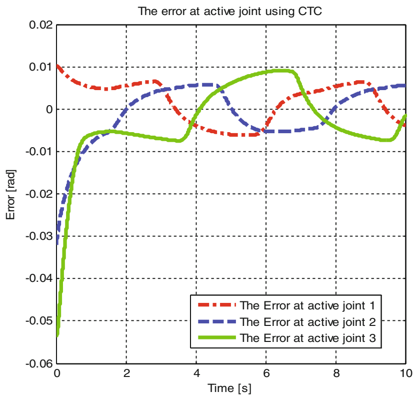
where β is positive constant. w is variable from time zero.

In this paper, the position synchronization controller is defined:

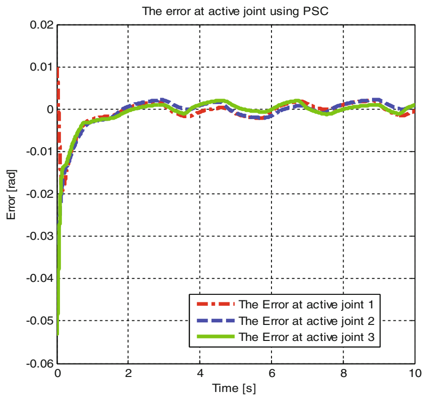
$$\tau_a = M(q)\left[\ddot{\theta}_d + K_e e + K_v \dot{e}\right] + C(\theta, \dot{\theta})\dot{\theta} + K_s s(t) + K_c r(t) \tag{15}$$

where K_s is positive matrix. ε is synchronization error matrix. $s(t) = \varepsilon_i - \varepsilon_{n+1-i}. r(t) = e_i^* + \Gamma \dot{e}_i^*$.

The friction at each active joint is $F_i = 0.4\,\mathrm{sign}\,(\dot{\theta}_{ai}) + 0.5\dot{\theta}_{ai}$. The results using CTC and the position synchronization controller (PSC) showed on Fig. 3. We can see, the error when using PSC achieves smaller error than of CTC. Figure 3 was also presented how errors reduced when using the principle of synchronization errors. The error at each joint promote each other convergence to zero at same time.



a) The error of active joints when using CTC



b) The error of active joints when using PSC

Fig. 3. The result of tracking errors in the cases of using CTC and PSC

4 The Proposed Adaptive Synchronization Controller

4.1 The Orthogonal Neural Network

In this part, the orthogonal neural network was based on FNN and polynomial for active function [18, 19] is showed. According to theory of orthogonal function, an arbitrary function $f(x), \mathrm{f} : [\mathrm{a}, \mathrm{b}] \rightarrow \Re$ will have an orthogonal polynomial.

$$F_n(x) = w_1\phi_1(x) + w_2\phi_2(x) + \ldots + w_n\phi_n(x) \tag{16}$$

Such that

$$\lim_{n \to \infty} \int_a^b (f(x) - F_n(x))^2 dx = 0 \tag{17}$$

where

$$\int_a^b \phi_i(x)\phi_j(x)dx = \begin{cases} 0\, i \neq j \\ A_i\, i = j \end{cases} \tag{18}$$

$$w_i = \int_a^b f(x)\phi_i(x)dx/A_i \quad i = 1, 2, \ldots, n \tag{19}$$

$\{\phi_1(x), \phi_2(x), \ldots\}$ is an orthogonal set.

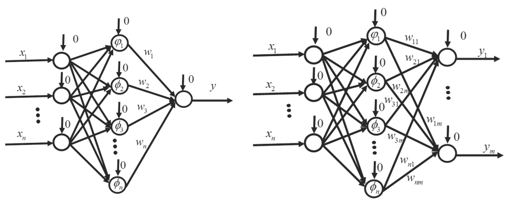
In case there is a function with m variables. Orthogonal function set will be $\{\Phi_1(x), \Phi_2(x), \ldots\}$. Each of orthogonal function is defined as

$$\Phi_i(X) = \phi_{1i}(x_1)\phi_{2i}(x_2)\ldots\phi_{mi}(x_m) \tag{20}$$

where $X = [x_1, x_2, \ldots, x_n]^T$ is input vector.

There are orthogonal functions such as Fourier series, Bessel function, Legendre, and Chebyshev polynomial.

Orthogonal neural network based on feedforward network with one hide layer showed in Fig. 4 by Tseng and Chen. In Fig. 4, (a) is multi input-one input and (b) is multi input-multi output. The connection between input layer and hide layer with the weight is one and the bias is zero. The $\phi_i (i = 0, 1, \ldots, n)$ is orthogonal function. The weight between is w_{ij} (with i is number orthogonal function and j is number output). The output of ONN can be showed below:



a) b)

a) single output *b) Multi output*

Fig. 4. Orthogonal neural network.

$$y = F(X) = \sum_{i=1}^{n} w_i(t)\Phi_i(X) = W^T(t)\Phi \tag{21}$$

4.2 The Proposes Adaptive Synchronization Controller

According to the nice result of the Eq. (15), the propose controller adding ONN for estimating uncertainties for the planar parallel manipulator is expressed as follow:

$$\tau_a = M(q)\left[\ddot{\theta}_d + K_e(e) + K_v(\dot{e})\right] + C(\theta, \dot{\theta})\dot{\theta} + K_s s(t) + K_c r(t) + f_{ONN} \tag{22}$$

where f_{ONN} is output of ONN. The ONN which is based on feedforward network has three layers. The block diagram of the proposed adaptive position synchronization controller using orthogonal neural network is illustrated in Fig. 5.

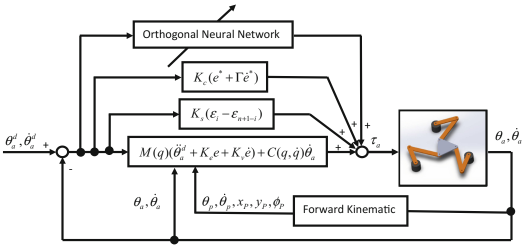


Fig. 5. The block diagram of the proposed controller

In the Eq. (22), the sliding function is defined as:

$$s = \dot{e} + \Gamma e + \Lambda \varepsilon \tag{23}$$

The Input Layer: The input vector of the ONN is denoted by

$$X = [x_1, x_2, \ldots, x_9]^T = [e_1, \dot{e}_1, \varepsilon_1, e_2, \dot{e}_2, \varepsilon_2, e_3, \dot{e}_3, \varepsilon_3]^T \tag{24}$$

where $e_i = \theta_{ai}^d - \theta_{ai}, \dot{e}_i = \dot{\theta}_{ai}^d - \dot{\theta}_{ai}, \varepsilon_i = e_i - e_{i+1}, i = 1, 2, 3$. The input must be inside [−1, 1]. The limitation of input inside [a,b] we can be transformed as:

$$t_i = \frac{2}{b-a}x_i - \frac{b-1}{b-a}, \; t_i \in [-1, 1] \tag{25}$$

The Hidden Layer: The hidden layer using Chebyshev polynomial of the first kind for orthogonal function expressed follow:

$$\phi_i(x) = \frac{(-2)^i i!}{(2i)!}\sqrt{1-x^2}\frac{d^i}{dx^i}(1-x^2)^{i-1/2} \; \text{i = 0,1,2,...,n} \tag{26}$$

The equation can be rewritten $\phi_0(x) = 1, \;\; \phi_1(x) = x, \;\; \phi_2(x) = 2x^2 - 1, \;\; \ldots,$ $\phi_{n+1}(x) = 2x\phi_n(x) - \phi_{n-1}(x)$. In this paper, n = *10* is chosen.

The Output Layer: The weight matrix connection between the hidden with output layer is showed by:

$$W = [w_1, w_2, \ldots, w_n]^T \tag{27}$$

The output of ONN expressed by:

$$y = W^T \Phi \tag{28}$$

where $\Phi = \{\Phi_1(\mathrm{X}), \Phi_2(\mathrm{X}), \ldots, \Phi_n(\mathrm{X})\}$ with $\Phi_i(\mathrm{X})$ showed on (20).

In the proposed controller, the weights of the ONN are adjusted based on the gradient descent method. Although in this case is multiple outputs showed in Fig. 4b. However, each of single-output neural networks has its independent weights, their respective weights can be separately trained. The cost function given by

$$E = \frac{1}{2}e^2 = \frac{1}{2}(y - \hat{y})^2 \in \Re \tag{29}$$

where e is the learning error. $\hat{y}$ is the output of ONN. y is the actual output.

The ONN's weight update law for the instantaneous gradient descent algorithm is given as $\dot{W} = \delta e \Phi$ which δ is the learning rate. e is the learning error.

5 The Result Simulation

To illustrate the effectiveness of the adaptive position synchronization controller using ONN for 3-DOF planar parallel manipulator in this paper, simulation studies were conducted on Matlab-Simulink and mechanical model of the parallel manipulator was built on Solidworks and exported to SimMechainics. By this way, the mechanical model is almost the same with the real model and the forward dynamics don't need to be used. The parallel robot manipulator has full parameter exported from Solidworks designed.

The parameter of 3-DOF planar parallel manipulator were presented in Table 1, where l_i is length of link i^{th}, l_{ci} is distance from the joint to the center of mass for link i^{th}, m_i is mass of link i^{th}, and I_i is inertial of link i^{th} (i = 1,2,3). The gains of the conventional CTC were chosen as $K_e = 20 \times I_{3\times3}$, $K_v = 50 \times I_{3\times3}$ in which $I_{3\times3}$ is the identity matrix with dimension equal to 3×3. The learning rate of FNN is $\eta = \mu = 0.2$ and the number of hide neurons is $n = 10$. The synchronization rate $\beta = 0.2$, $K_s = 220 \times I_{3\times3}$, $K_c = 0.8 \times I_{3\times3}$, $\Gamma = 0.4 \times I_{3\times3}$.

Table 1. Parameter of the manipulator

Links	l_i (m)	L_{ci} (m)	m_i (kg)	I_i (kg.m^2)
Active link	$l_1 = 0.4$	0.2	5.118	9139×10^{-6}
Passive link	$l_2 = 0.6$	0.3	7.39	26763×10^{-6}
End-effector	$l_3 = 0.2$	–	1.84	3170×10^{-6}

In ONN, the learning rate is $\eta = 0.2$ and $\Lambda = 0.5 \times I_{3\times3}$. The number of Chebyshev polyniminal was chosen $n = 10$.

The input trajectories of end-effector given: $x(t) = 0.49 + 0.03\ \cos\left(\pi(t)/3\right)$, $y(t) = 0.37 + 0.03\ \sin\left(\pi(t)/3\right)$, $\phi_P(t) = \pi/12$. After computed inverse kinematic, the desired angular is given at each active joints of planar parallel manipulator $\theta^d_{ai} = [\theta^d_{a1}, \theta^d_{a2}, \theta^d_{a3}]^T$. The uncertainties are given by addition the friction each active joints as the following $f_{ai} = 0.4sign(\dot{\theta}_{ai}) + 0.5\dot{\theta}_{ai}$, i $= 1, 2, 3$.

In Figs. 6, 7, 8 and 9, the performance of our proposed controller was compared with adaptive CTC using FNN (CTC-FNN) and adaptive PSC using FNN (PSC-FNN). Firstly, the feature of synchronization was clearly shown in Figs. 6, 7 and 8. The convergence of error at joint one includes two steps. (1) The error at joint one equals to the error at other ones. (2) The error at each joint approaches to zero together. As the results can be seen in Figs. 7, 8 and 9, PSC-FNN and PSC-ONN showed the same position error trend, while CTC-FNN has the reverse position error trend. The reason is that due to the constraints of joint two and joint three as shown in Figs. 7 and 8. The position error of joint one using PSC-FNN and PSC-ONN has to change adaptively to the error of joint two and joint three. However, although PSC-ONN has the convergence slower than PSC-FNN and CTC-FNN in the first 2 s, it approaches to zero faster after that. The results of tracking trajectory were presented in Fig. 9.

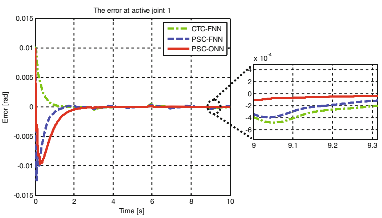


Fig. 6. The error of active joint 1

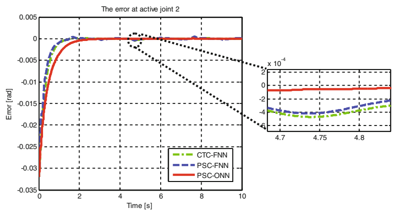


Fig. 7. The error of active joint 2

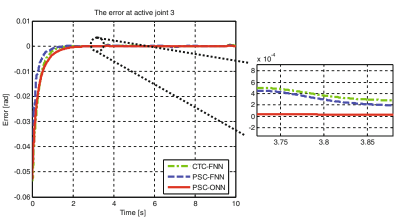


Fig. 8. The error of active joint 3

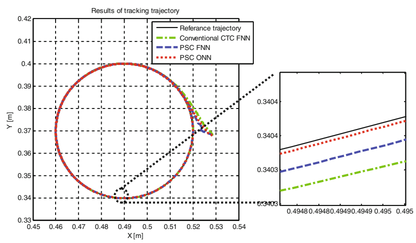


Fig. 9. The result of tracking trajectory

6 The Conclusion

In this paper, an adaptive position synchronization controller using orthogonal neural network for 3-DOF planar parallel manipulators was presented. The proposed controller is composed of the computed torque control method with position synchronization control algorithm and orthogonal neural network. The orthogonal neural network was trained online during the trajectory tracking control based on slide mode function and online learning method to deal with the existence of the uncertainties. To illustrate the effectiveness of the proposed strategy, simulations were conducted in order to compare the proposed controller with adaptive computed torque controller using traditional FNN and adaptive position synchronization controller using traditional FNN. The simulation results shows that the joint errors converges quickly to zero almost at the same time and the proposed controller has the better tracking performance than CTC and PSC control algorithm using traditional FNN.

Acknowledgement. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A3B03930496).

References

1. Koren, Y.: Cross-coupled biaxial computer control for manufacturing systems. J. Dyn. Syst. Meas. Control **102**, 265 (1980)
2. Feng, L., Koren, Y., Borenstein, J.: Cross-coupling motion controller for mobile robot. IEEE Control Syst. **13**, 35–43 (1993)
3. Ren, L., Mills, J.K., Sun, D.: Controller design applied to planar parallel manipulator for trajectory tracking robot. In: Proceeding of the 2005 IEEE International Conference on Robotics and Automation (2005). doi:10.1109/ROBOT.2005.1570242
4. Ren, L., Mills, J.K., Sun, D.: Experimental comparison of control approaches on trajectory tracking control of a 3 DOF parallel robot. IEEE Trans. Control Syst. Technol. **15**(5), 982–988 (2007)
5. Liu, Y.-C., Chopra, N.: Controlled synchronization of heterogeneous robotic manipulators in the task space. IEEE Trans. Robot. **28**, 268–275 (2012)
6. Rodriguez-Angeles, A., Nijmeijer, H.: Mutual synchronization of robot via estimated state feedback: a cooperative approach. IEEE Trans. Control Syst. Technol. **12**(4), 542–554 (2004)
7. Adaptive robot control using neural network: Sad, M., Bigras, P., Dessaint, L.-A., AI-Haddad, K. IEEE Trans. Ind. Electron. **41**, 176–181 (1994)
8. He, W., Chen, Y., Yin, Z.: Adaptive neural network control of an uncertain robot with full-state constraints. IEEE Trans. Cybern. **46**, 620–629 (2016)
9. Li, X., Cheah, C.C.: Adaptive neural network control of robot based on a unified objective bound. IEEE Trans. Control Syst. Technol. **22**, 1032–1043 (2014)
10. Le, Q.D., Kang, H.-J., Le, T.D.: Adaptive extended computed torque control of 3 DOF planar parallel manipulators using neural network and error compensator. In: Huang, D.-S., Han, K., Hussain, A. (eds.) ICIC 2016. LNCS, vol. 9773, pp. 437–448. Springer, Cham (2016). doi:10.1007/978-3-319-42297-8_41

11. Gao, Y., Er, M.J., Yang, S.: Adaptive control of robot manipulators using fuzzy neural networks. IEEE Trans. Ind. Electron. **48**, 1274–1278 (2001)
12. Chen, C.-S.: Dynamic structure neural fuzzy networks for robust adaptive control of robot manipulators. IEEE Trans. Ind. Electron. **55**, 3402–3414 (2008)
13. Chen, M.S., Manry, M.T.: Conventional modeling of the multilayer perception using polynominal basis functions. IEEE Trans. Neural Netw. **4**, 14–16 (1992)
14. Castillo, E., Guijarro-Berdinas, B., Fontenla-Romero, O., Alonso-Betanzos, A.: A verry fast learning method for neural networks based on sensitivity analysis. J. Mach. Learn. Res. **7**, 1159–1182 (2006)
15. Karayiannis, N.B., Venetsanopoulos, A.N.: Fast learning algorithm for neural networks. IEEE Trans. Circ. Syst. **39**(7), 453–474 (1992)
16. Yam, Y.F., Chow, T.W.S., Leung, C.T.: A new method in determining the initial weights of feedforward neural networks. Neuralcomputing **16**, 23–32 (1997)
17. Wang, S., Chung, F.-L., Wang, J., Wu, J.: A fast learning method for feedforward neural networks. Neuralcomputing **149**, 295–307 (2015)
18. Yang, S.-S., Tseng, C.-S.: An orthogonal neural network for function approximation. IEEE Trans. Syst. Man Cybern. **26**(5), 779–785 (1996)
19. Sher, C.F., Tseng, C.-S., Chen, C.-S.: Properties and performance of orthogonal neural network in function approximation. Int. J. Intell. Syst. **16**, 1377–1392 (2001)
20. Peric, S.L., Antic, D.S., Milovanovic, M.B., Mitic, D.B., Milojkovic, M.T., Nikolic, S.S.: Quasi-sliding mode control with orhtogonal endocrine neural network-based estimator applied in anti-lock braking system. IEEE/ASME Trans. Mechatron. **21**(2), 754–764 (2016)
21. Milojkovic, M.T., Antic, D.S., Milovanovic, M.B., Nikolic, S.S., Peric, S.L., Almawlawe, M.: Modeling of dynamic systems using othogonal endocrine adaptive neural-fuzzy inference systems. J. Dyn. Syst. Meas. Control **137**(9), DS-15-1098 (2015)
22. Milojkovic, M.T., Nikolic, S.S., Dankovic, B., Antic, D., Jovanovic, Z.: Modeling of dynamical systems based on almost orthogonal polynominal. Math. Comput. Model. Dyn. Syst. **16**(2), 133–144 (2010)
23. Timmis, J., Neal, M., Thorniley, J.: An adaptive neuro-endocrine system for robotic systems. In: IEEE Workshop on Robotic Intelligence in Informationally Structured Space, pp. 129–136 (2009). doi:10.1109/RIISS.2009.4937917
24. Wang, P.: Control of robot manipulators based on legendre orthogonal neural network. Appl. Mech. Mater. **427–429**, 1089–1092 (2013)
25. Nakamura, Y., Ghodoussi, M.: Dynamics computation of close-link robot mechanisms with no redundant and redundant actuators. IEEE Trans. Robot. Autom. **5**, 294–302 (1989)