Dynamic-Growing Fuzzy-Neural Controller, application to a 3PSP Parallel Robot

Mohsen Jalaeian-F., Mohammad-R. Akbarzadeh-T., Alirea Akbarzadeh, Damoon Azarpazhooh

Abstract— Due to the complexity of its dynamic equations, parallel robot control has become an active research area in recent years. In this application domain, model-based approaches are often time-consuming, and model-free controllers generally lack the desired performance for controlling such a fast machine. Here, a new adaptive intelligent controller is proposed for 3PSP parallel robots based on an auto-structuring fuzzy neural architecture. This control approach aims to reach high performance while maintaining overall stability. The proposed Dynamic Growing Fuzzy-Neuro Control (DGFNC) approach adds new rules more conservatively, hence pruning mechanism is omitted. Instead, an adaptive controller ‘adapts’ the system to parameter variations. Furthermore, a sliding mode nonlinear controller ensures system stability. This hybrid approach leads to less computation and faster response. The merits of DGFNC are illustrated by simulation results as applied to the 3PSP robot.

I. INTRODUCTION

Traditional controllers are usually complex in design; and relatively time-consuming because of involving complex mathematical equations. Moreover, if the plant is changed even insignificantly, these controllers must be designed anew. Thus in recent decades, scientists have taken an interest in developing intelligent controllers, and the new controllers are progressing at a phenomenal speed. The internal obscurity of the plant and external disturbance both lead to an inefficient control, so the adaptive controllers are the only choice to deal with these issues [1-3].

Neural networks have a special place in modeling the plant’s nonlinear dynamics, in both adaptive and online approaches [4, 5]. On the other hand, fuzzy systems are used in handling uncertainties in modeling, control and decision making [6, 7].

With the advanced progression of learning algorithms, Fuzzy Neural Network structure (FNN) has become one of the most efficient hybrid methods for adaptive control for its many qualities; namely fuzzy reasoning, neuron-learning and universal approximation (in FNN based structure) [8, 9]. FNN offers a favorable structure for approximation with finite nodes, when assisted by an appropriate learning algorithm.

This research, studies a new intelligent controller, and we apply an appropriate modification on this controller by considering the constraints of the plant, finally, the modified controller is used to control 3-PSP parallel robot.

The introduced controller divides learning algorithm to structure-learning and parameter-learning algorithms. The purpose of structure-learning algorithm is finding the optimal nodes quantity (R*) and it serves this purpose with two strategies node-adding and node-pruning, which modified by some changes to the dynamic node-adding. The end of Parameter-Learning Algorithm is finding the optimal value of FNN’s parameters which includes fuzzy system parameters (m, δ) and neural network’s parameter (ξ) in relation to the optimal determined structure [1]. This task is carried out using back-propagation learning algorithm. Finally, auto-structuring mechanism produces an optimal structure for FNN and adjusts its parameters at the same time, by considering the value and differentiation of error [1, 7].

The main advantages of this hybrid controller include relief from the complexity and hardships of designing the controller, offering a self-organizing controller, an optimal structure that provides optimal calculations, retaining high performance control, overall stability ensured by the supervisory controller-chosen as sliding-mode-, and finally being adaptive and model-free thereby eliminating the effect of modeling uncertainties on controller’s efficiency [1, 3].

Based on our extended experience with this robot control simulations we had the impression that the node-pruning mechanism is time-consuming and it fails to enhance the controller. This method merely balances the quantity of nodes along with the adding-mechanism, and if removed “The Curse of Dimensionality” (node redundancy) occurs. To cope with this problem node-adding must be done very strictly, that is node-adding criterion (Γn) must change to a dynamic threshold which can be applied by making Γn a function of R. Based on our experience, it can be realized that updating σ through parameter-learning mechanism has an insignificant effect on improvement of robots’ control, and since σ value is almost constant, an optimal predefined value can be assigned to σ in the process of adding a new
II. CONTROLLER

A. Controller description

Since the plant consists of three separate systems with similar dynamics; the dynamics is assumed here to have low interaction due to the robot’s parallel mechanical and loading design; and the control scheme is selected as Independent Joint Control thereby designing three parallel DGFNC MISO controllers are required.

Even though Auto-Structuring Fuzzy Neural System as a controller has offered some advantages such as model-free designing, overall stability, optimal and self-organizing structure, it is not foolproof. Actually ASFNS take about 1.7 milliseconds (msec) in each run (hardware will be explained in simulation part) which is greater than that of parallel robot’s control loop, 1 msec. To resolve this problem, node pruning mechanism is omitted and node adding criterion is shaped as a function of R (Figure 1) instead of a pre-given constant. Since \( \delta \) does not a changed significantly during the running process, updating the \( \delta \) is discard and \( \delta \) is determined by a pre-given constant. A sliding surface is defined as:

\[
s = e^{(n-1)} + k_e e^{(n-2)} + \cdots + k_n \int_0^t e d \tau (2)
\]

Now for designing a sliding mode controller we have:

\[
\text{u}_{SMC} = u_E + u_H (3)
\]

The equivalent controller and the hitting controller are represented respectively as:

\[
u_E = h^e(-f_n(X_\cdot u) + x_e^{(n)}) + k_e e^{(n-1)} + \cdots (4)\]

\[
u_H = D_i \text{sgn}(s) (5)
\]

Where, the \( \text{sgn} . \) is the sign function. Substituting Eq. (2)-(5) in Eq. (1) yields:

\[
Ve^{n} + k_e e^{n-1} + \cdots (6)
\]

As it was mentioned in the beginning of the controller description, control scheme is Independent Joint Control; so three MISO controllers were utilized to control the plant. To check the stability of each controller, we use the Lyapunov function [1]:

\[
V_1(s) = \frac{1}{2} s^2 (7)
\]

Differentiating the Lyapunov function with respect to time and using Eq. (6), we have:

\[
\dot{V_1} + \text{sgn}(s) - \delta \leq -D_i \text{sgn}(s) \leq 0 (8)
\]

The most useful property of sliding-mode is its insensitivity to the variations of controlled system and external disturbance when trajectories are forced on sliding surface. Lyapunov theorem guarantees asymptotic stability. For more details see [1].

B. Sliding-Mode Controller

Consider the following equation as nth order nonlinear system:

\[
x^{(n)} = f(X, u) + \Delta f(X, u) + d = \{ f(X, u) - hu \} + hu + \delta = f_c(X, u) + hu + \delta (1)
\]

In which the state vector of system is \( x = [x', \ldots, x^n] \in \mathbb{R}^n; f(x) \) is an unknown continues function; \( h \) is a given constant and \( u \in \mathbb{R} \) is the input of system. The term of \( \delta \) comprises the internal system imperfection \( \Delta f(X, u) \) and the external disturbance \( (d) \).

If the state trajectory is \( x \) and the trajectory command is \( x_e \), then we define the tracking error as \( e = x_e - x \), and \( x_e \) is continues and available. We assume that \( \delta \) is bounded by \( |\delta| < D \), that \( D_i \) is a finite positive constant. A sliding surface is defined as:

\[
Ve^{n} + k_e e^{n-1} + \cdots (6)
\]

As it was mentioned in the beginning of the controller description, control scheme is Independent Joint Control; so three MISO controllers were utilized to control the plant. To check the stability of each controller, we use the Lyapunov function [1]:

\[
V_1(s) = \frac{1}{2} s^2 (7)
\]

Differentiating the Lyapunov function with respect to time and using Eq. (6), we have:

\[
\dot{V_1} + \text{sgn}(s) - \delta \leq -D_i \text{sgn}(s) \leq 0 (8)
\]

The most useful property of sliding-mode is its insensitivity to the variations of controlled system and external disturbance when trajectories are forced on sliding surface. Lyapunov theorem guarantees asymptotic stability. For more details see [1].

C. Fuzzy Neural Network

We used the feed forward FNN which contains three layers. The first layer is input layer and it is represented as:

\[
net_i^{[1]} = z_i, y_i^{[1]} = f^{[1]}(net_i^{[1]}) = net_i^{[1]}
\]

Where \( z_i \) represents the input of \( i^{th} \) node.
and $z_i = e^{(i)}$, $i = 1, 2, \ldots$.

The second layer is the rules’ layer that means each node acts as a rule. For the kth node the reception and the activation functions are:

$$net_k^{[2]} = \sum_{i=1}^{n} \left( -\frac{(y_i^{[1]} - m_{ik})^2}{(\sigma_{ik})^2} \right)$$ (9)

$$y_k^{[2]} = f^{[2]}(net_k^{[2]}) = \exp(net_k^{[2]})$$ (10)

In which $m_{ik}$ and $\sigma_{ik}$ are the mean and standard deviation of the kth rule belonging to the ith input. The third layer is a simple node that is labeled by $\Sigma$, and computes the linear combinations of incoming inputs as:

$$net_0^{[3]} = \sum_k \xi_k y_k^{[2]}$$ (11)

$$y_0^{[3]} = f^{[3]}(net_0^{[3]}) = net_0^{[3]}$$ (12)

Where $\xi_k$ is the consequence weighting of the kth rule and $y_0^{[3]}$ is the output of FNN. Output of the FNN-based controller $u_{fn}$ with R rules can be written as:

$$u_{fn} = \sum_k \xi_k \prod_i \exp\left[\frac{-\left(\frac{z_i - m_{ik}}{\sigma_{ik}}\right)^2}{\left(\sigma_{ik}\right)^2}\right] = \sum_k \xi_k \Gamma_k$$ (13)

For more detail of parameter in vector form see [1].

D. Dynamic Growing mechanism

The objective of auto-structuring mechanism is finding appropriate nodes for the FNN. We used node-adding and node-pruning operations to find the optimum case node-adding operation is based on the maximum output of the existing nodes that is

$$\Gamma_{\max} = \max(\Gamma_k), k = 1, 2, \ldots, \hat{R}(t)$$

where $\hat{R}(t)$ is the number of FNN node at time t. For the network input at time t, if $\Gamma_{\max}(t) \leq \Gamma_{th}$ is satisfied, a new node is generated, where $\Gamma_{th}$ is updated by Dynamic Growing function. The parameters of a new generated node is set as $m_{ik} = z_i, \sigma_{ik} = \sigma_c$, for $i=1,2,\ldots,n$ and $\xi_{ik} = \xi_c$, where $\sigma_c$ is pre-specified constant and $\xi_c = 0$. By gradient-descent method, the adaptation laws are [1]:

$$\dot{\xi_k} = \eta_m \frac{\partial E}{\partial \xi_k}$$ (14)

$$\dot{\xi_k} = \eta_{\xi_k} \frac{\partial E}{\partial \xi_k}$$ (15)

Where $\eta_m$ and $\eta_{\xi_k}$ are positive constants.

III. 3PSP PARALLEL ROBOT

A. 3PSP Parallel Robot Structure

Our target plant is a type of parallel manipulator by 3 degrees of freedom named 3-PSP (Prismatic-Spherical-Prismatic). The 3-PSP structure is based on two rigid bodies; a movable platform (star shaped) and a fixed base, which are connected by three PSP legs. In the structure of each leg, there is an actuated prismatic joint and a passive spherical joint paired with a passive prismatic joint. This structure has three closed kinematic loops. Two prismatic joints are independent but the spherical joint is dependent. For more details about degrees of freedom see [11]. The manipulator has 15 joints; twelve are passive (nine revolute for three spherical and three upper prismatic) and three of them are active and driven by three independent actuators. It has been designed so that the first prismatic joint is actuated in each leg by connecting a motor to a ball screw, (for direct and inverse kinematic of 3-PSP see [11]). Figure 2, shows the virtual structure of the 3PSP manipulator. The PSP legs are completely identical, and three bars have been welded together in a way that the platform makes a star planar shape. This parallel structure has become fully symmetrical. These bars stand at 120 degrees against one another. For defining the position and orientation of the frame, we need to extend or descend each actuated prismatic joint. $\{T\}$ is attached to the center of the star shape platform by means of a fixed coordination frame and $\{B\}$ is attached to the fixed base. The base platform was chosen as an equilateral triangle for the sake of simplicity and symmetry of equations which creates a rather symmetrical shape for the 3-PSP robot. Since the manipulator is not to rotate about z-axis of the base frame, the prismatic actuators are welded to the fixed base.

The three degrees of freedom in 3-PSP manipulator are
produced by: Changing z-height of point P and also changing the orientation of the moving star about x and y axes of frame \{B\}.

The inverse Kinematics objective is obtaining the position of the actuators for a given pose of the end-effector (moving star shape). The control’s objective is computing the torque of motors for its determined positions which is obtained by inverse kinematic equations.

B. Plant Modeling and Simulation

We have previously modeled 3PSP parallel robot by some software such as Solid-works, Adams and Matlab of which we used the Matlab Model in this study.

Kinematic and dynamic equations of 3PSP are obtained and described [11];

IV. SIMULATION

In this work, we applied DGFNC controller to the 3PSP robot’s model (Figure 3-6) using MATLAB software with SIMULINK tools on a PC powered by Intel core2duo 2.2 GHz CPU.

DGFNC can start with zero number of nodes in hidden layer. In this case, at preliminary iteration the number of nodes grows up, but at a steady state the number stops changing. Figure 6 displays this pattern and figure 3 shows the desired and actual position trajectory of end-effector (desired trajectory is a circle and z-axis scaled 1e-3 at the figure).

This control scheme is implemented as an independent joint control, that is, the controller excites motors on the actuated joints and the feedback is received from motors encoder. As actuators are actually AC-servo motors, in these simulations we did not consider the dynamic of the motors and motor drivers, because they were a lot faster than the structure’s dynamic. Desired and actual trajectories of joints angles (when controller start with R0=0) are shown in figure 4. Although there is undershoot at the beginning of tracking (zoomed at figure 5), this error can be solved completely by initializing R (number of neurons) at first. This undershoot occurs because actual angles are significantly different from desired angles. It is clear that the value of error is more than the amount that DGFNC with very low number of neurons, can compensate for. This issue was solved by initializing DGFNC with the certain number of trained neurons, for instance, R initialized by 6 and the \( m \), \( \sigma \) and \( \zeta \) matrices initialized by last trained values.

V. CONCLUSION

As mentioned in this study, a major setback for designing practical controllers is the speed of calculations. In other words, time-consuming controllers are unusable in real control experiments. The most important output of this work is to design an auto-structuring controller with optimal and swift calculations. We introduced a new intelligent controller, DGFNC, based on some modification on ASFNS which eventually produced a dynamic mechanism for node-adding. Unlike ASFNS, DGFNC does not lose the trained nodes through pruning-mechanism, but rather improves their learning. The outstanding merits of DGFNC can be described as the optimal and self-organizing structure, brief calculation (a major decrease in time of control loop), as well as improved stability and high-efficiency. Finally, DGFNC is applied to the control of a special fast parallel robot, 3PSP, and simulation results verify the advantages of this new intelligent controller.

ACKNOWLEDGMENT

The authors would like to gratefully acknowledge the hard work of our teammates in Parallel Robots Group of Ferdowsi University of Mashhad, we enjoyed working with them.
REFERENCES

[1] Kuo-Hsiang Cheng, “Auto-structuring fuzzy neural system for intelligent control” Journal of the Franklin Institute 346 (2009) 267–288
[2] L.X. Wang, Adaptive Fuzzy Systems and Control: Design and Stability Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1994.
[3] J.H. Park, S.H. Huh, S.H. Kim, S.J. Seo, G.T. Park, Direct adaptive controller for nonaffine nonlinear systems using self-structuring neural networks, IEEE Trans. Neural Netw. 16 (2005) 414–422.
[4] K. Funahashi, "On the approximate realization of continuous mapping by neural networks," Neural Networks, vol. 2, pp. 183-192, 1989.
[5] K.Hornik, M.Stinchcombe, H.White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1999) 359–366.
[6] L. X. Wang and J.M.Mendel, “Analysis and design of fuzzy logic controller,” USC SIPI Rep, 184, 1991
[7] L. X. Wang and J.M.Mendel, “Backpropagation fuzzy systems as nonlinear dynamic system identifiers,” in Proc, IEEE 1992 Int. Conf. Fuzzy Systems (San Diego, CA) Mar 1992, pp. 1409-1418.
[8] L.X.Wang, J.M.Mendel, Fuzzy basis functions, universal approximation, and orthogonal least squares learning, IEEE Trans. Neural Netw. 3 (1992) 807–814.
[9] F.Da, W.Song, Fuzzy neural networks for direct adaptive control, IEEE Trans. Ind. Electron. 50 (2003) 507–513.
[10] J. P. Merlet, “Parallel Robots”, 2nd ed., Kluwer Academic Publishers, London, 2000.
[11] P. Mahmoodnia and A. R. Akbarzadeh Tootoonchi, “Close form Solutions for Inverse and Direct Position Analysis of a Special 3-PSP Parallel Manipulator,” ISME conference, 2009.