Adaptive Control of a Parallel Robot Using Fuzzy Inverse Model

*Ayman A. ALY, **Hidetoshi OHUCHI, and ***Ahmed A. Abo-Ismail

Abstract—The Stewart platform is one example of a parallel manipulator with high force to weight ratio and fine positioning accuracy far exceeding those of a conventional serial-link arm. It is basically a closed-link type robot manipulator having 6 degrees of freedom. In this paper the implementation of an adaptive control scheme based on fuzzy logic theory is used to control the motion of a Stewart platform manipulator. The inverse kinematics is analysed and six individual controllers are implemented in the actuators coordinates. An experimental study is conducted to evaluate the performance of the proposed control scheme implemented to control the manipulator to track a defined path. Experimental results show that the proposed control policy provides superior tracking capability as compared to the fixed-gain controllers.

Index Terms—Parallel robot, Stewart platform, Fuzzy inverse Model.

I. INTRODUCTION

Parallel manipulators can be found in many applications, such as flight simulators, adjustable articulated trusses, mining machines, pointing devices and walking machines. Recently, it has also been developed as a high-speed, high-precision, multi-DOF machining center.

Most of the 6 DOF parallel manipulators studied to date consist of six extensible limbs. These parallel manipulators possess the advantages of high stiffness, low inertia, lower accumulation of joint errors and large payload capacity. However, they suffer the problems of relatively small useful workspace and design difficulties. Furthermore, their direct kinematics is a very difficult problem.

Parallel manipulators are classified as planer, spherical, or spatial manipulator in accordance with their motion characteristics. As shown in Fig. 1, a parallel manipulator typically consists of a moving platform that is connected to fixed base by several limbs or legs. Typically, the number of the legs is equal to the number of the freedom such that every limb is controlled by one actuator and all the actuators can be mounted at or near the fixed base. For this reason, parallel manipulators tend to have a large load-carrying capacity. The researches in the parallel manipulators are mainly concerned in three studying areas: closed form solution for the forward and inverse kinematics, derivation of dynamics equations and designing an adequate controller [1] [2] [3].

In this paper, the inverse kinematics of the Stewart platform is analyzed on the base of geometric configuration and a 6-DOF trajectory tracking control system is implemented. Considering the robustness against the nonlinearity of the system parameters and the resultant accuracy with the supply pressure change, FMRLC (Fuzzy Model Reference Learning Control) scheme is adopted for tracking the given referred trajectory.

II. POSITION ANALYSIS OF THE STEWART PLATFORM

In Fig. 1, six identical limbs connect the moving platform to the fixed base by spherical joints at points $B_i$ and $A_i$, $i=1,2,…,6$, respectively. Each limb consist of an upper member and a lower member connected by a prismatic joint and therefore to control the location of the moving platform. There are 14 links connected by 6 prismatic joints and 12 spherical joints. Hence the number of DOF of the manipulator is

$$ F = \lambda(n - j - 1) + \sum_{i} f_i $$

$$ F = 6(14 - 18 - 1) + (6 + 3 * 12) = 12. \quad (1) $$

where $\lambda$ is the limbs number, $n$ is the links number, $j$ is the joints number and $f_i$ is the joint freedoms. However, there are 6 passive DOF associated with the six SPS limbs. Therefore, the moving platform poses a 6 DOF. Since the limbs are connected to the moving platform and the fixed base by spherical joints, no bending moments or twisting torques will transmit to the limbs. The force acting on each limb is directed along the longitudinal axis of the limb. Consequently, these limbs can be made of hollow cylindrical rods to produce a high-weight, high-stiffness, high-speed manipulator.
The length of the $i^{th}$ limb is obtained by taking the dot product of the vector $\overline{A_iB_i}$ with itself:

$$d_i^2 = [P + \bar{A} \bar{R}_b b_i - a_i] [P + \bar{A} \bar{R}_b b_i - a_i]^T$$

where $d_i$ denotes the length of the $i^{th}$ limb.

For the inverse kinematics problem, the position vector $P$ and rotation matrix $\bar{A} \bar{R}_b$ of frame $B$ with respect to $A$ are given and the limb lengths $d_i$ are to be find. Expanding Equation (4) and by taking the square root we obtain

$$d_i = \pm \sqrt{P^T P + a_i^T a_i + 2P^T \bar{A} \bar{R}_b b_i - 2P^T a_i - 2a_i^T \bar{R}_b b_i}$$

This yields six equations describing the location of the moving platform with respect to the fixed base. Hence, corresponding to each given location of the moving platform, there are generally two possible solutions for each limb. However, the negative limb length is physically not feasible. When the solution of $d_i$ becomes a complex number, the location of the moving platform is not reachable.

IV. JACOBIAN ANALYSIS OF PARALLEL MANIPULATORS

The Jacobian analysis of parallel manipulators is much more difficult problem than the serial manipulators because there are many links that form a number of closed loops. An important limitation of the parallel manipulators is that singular configurations may exist within its workspace. Depending on which one is singular, a closed loop mechanism may be at a direct kinematics singular configuration, an inverse kinematics singular configuration, or both.

For the Stewart platform, the input vector is given by

$$\ddot{q} = [d_1, \ldots, d_6]^T,$$

and the output vector can be described by the velocity of the centroid P and the angular velocity of the moving platform.

$$\dot{X} = \begin{bmatrix} \dot{V}_p \\ \dot{\omega}_p \end{bmatrix}$$

The Jacobian matrix can be derived by formulating a velocity loop closure equation for each limb. Referring to Fig. 5.a loop-closure equation for the $i^{th}$ limb can be written as

$$\overline{OP} + \overline{PB_i} = \overline{OA_i} + \overline{A_iB_i}$$

III. INVERSE KINEMATICS

For the purpose of analysis, two Cartesian coordinate systems, frames $A(x,y,z)$ and $B(u,v,w)$ as shown in Fig. 1, are attached to the fixed base and moving platform, respectively. The transformation from the moving platform to the fixed base can be described by the position vector $P$ of the centroid $P$ and the rotation matrix $\bar{A} \bar{R}_b$ of the moving platform. Let $u, v,$ and $w$ be three unit vectors defined along the $u, v,$ and $w$ axes of the moving coordinate system; then the rotation matrix can be written as

$${}^A \bar{R}_b = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}$$

(2)

Note that the elements of $\bar{A} \bar{R}_b$ must satisfy the orthogonal conditions. As shown in Fig. 1, let $\overline{P_i} = [a_{ix}, a_{iy}, a_{iz}]^T$ and $\overline{b_i} = [b_{ix}, b_{iy}, b_{iz}]^T$ be the position vectors of points $A_i$ and $B_i$ in the coordinate frames $A$ and $B$, respectively. We can write a vector-loop equation for the $i^{th}$ limb of the manipulator as follows:

$$\overline{A_iB_i} = P + {}^A \bar{R}_b \overline{b_i} - \overline{a_i}$$

(3)
Differentiating equation with respect to time yields

\[ V_p + \omega_b \times b_i = d_i \omega_j \times s_i + d_i s_i \]  
\( i = 1, 2, \ldots, 6 \)

\[ (8) \]

where \( b_i \) and \( s_i \) denote the vector \( \overrightarrow{PB_i} \) and a unit vector along \( \overrightarrow{A_iB_i} \), respectively, and \( \omega_i \) denotes the angular velocity of the \( i^{th} \) limb with respect to the fixed frame \( A \). To eliminate \( \omega_i \), we dot-multiply both sides of the previous equation by \( s_i \):

\[ s_i \times V_p + (b_i \times s_i) \cdot \omega_i = d_i \]  
\( i = 1, 2, \ldots, 6 \)

\[ (9) \]

Equations written six times, once for each \( i = 1 \) to 6, yields six scalar equations, which can be assembled in matrix form:

\[ J_x X = J_q q \]  
\( (10) \)

where

\[ J_x = \begin{bmatrix} s_1^T (b_1 \times s_1)^T \\ s_2^T (b_2 \times s_2)^T \\ \vdots \\ s_6^T (b_6 \times s_6)^T \end{bmatrix} \]

\[ J_q = I \]  
(6X6 identity matrix)

Let: \( n_i = b_i + s_i \)  
\( (11) \)

Then \( n_i \) represents a vector that is normal to the plane containing points \( A_i, B_i, \) and \( P \).

**Inverse kinematics singularities**

Inverse kinematics singularities cannot occur within the workspace of the manipulator since \( J_q \) is an identity matrix. However, inverse kinematics singularities can occur at the workspace boundary where one or more limbs are in fully stretched or retracted positions.

**V. DYNAMICS OF THE STEWART PLATFORM**

For the inverse kinematics, the time history of a desired trajectory is given and the problem to determine the actuator forces and/or torques required producing that motion. The time history of the moving platform can be described by the position vector of the centroid, \( P \), and three Euler angles, \( \phi, \theta, \) and \( \psi \). The velocity and the acceleration of the centroid \( P \) are obtained by taking the derivative of \( P \) with respect to time; that is, \( V_p = \dot{P} \) and \( \dot{V}_p = \ddot{P} \). Let the Euler angles be defined as rotation of \( \phi \) about the z-axis, followed by a second rotation of \( \theta \) about the rotated y-axis, and a third rotation of \( \psi \) about the rotated w-axis. Then the rotation matrix of the moving platform relative to the fixed base is given by:

\[ R_p = \begin{bmatrix} c\phi c\theta c\psi + s\phi s\psi & -c\phi c\theta s\psi - s\phi c\psi & c\phi s\theta \\ s\phi c\theta c\psi + c\phi s\psi & -s\phi c\theta s\psi + c\phi c\psi & s\phi s\theta & c\theta \\ -c\phi s\theta & s\phi s\theta & c\theta \end{bmatrix} \]  
\( (12) \)

The angular velocity of the moving platform, \( \omega_p \), written in terms of the Euler angles and the body-fixed \( \omega, V', \) and \( W'' \) unit vectors, is

\[ \omega_p = \dot{\phi} W + \dot{\theta} V' + \dot{\psi} W'' \]  
\( (13) \)

Expressing the previous equation in the fixed reference frame \( A \), we obtain

\[ \omega_p = \begin{bmatrix} \psi c\phi \theta - \dot{\theta} s\phi \\ \psi s\phi \theta + \dot{\theta} c\phi \\ \psi c\theta + \dot{\phi} \end{bmatrix} \]  
\( (14) \)

The angular acceleration of the moving platform is obtained by taking the derivatives of equation with respect to time:

\[ \ddot{\omega}_p = \begin{bmatrix} \psi c\phi \theta - \dot{\theta} s\phi & -\psi s\phi \theta - \dot{\theta} c\phi \psi & \psi s\phi \theta + \dot{\theta} c\phi \\ \psi s\phi \theta + \dot{\theta} c\phi & \psi c\phi \theta - \dot{\theta} s\phi \psi & -\psi c\phi \theta - \dot{\theta} s\phi \psi \\ \psi c\theta & -\psi s\theta & \psi c\theta + \dot{\phi} \end{bmatrix} \]  
\( (15) \)

Actually our study of the Stewart platform manipulator has focused on kinematics consideration only. The main reason for driving equations for the manipulator dynamics is when we are given a trajectory point in joint coordinates, joint velocities and joint acceleration, and we wish to find the required vector of joint-forces \( \tau \).

The dynamics of a manipulator is usually described by the equation:

\[ \tau = M(q) \ddot{q} + V(q, \dot{q}) + G(q) \]  
\( (16) \)

where \( M(q) \) is the \( n \times n \) mass matrix of the manipulator, \( V(q, \dot{q}) \) is a \( n \times 1 \) vector of centrifugal and Coriolis terms and \( G(q) \) is a \( n \times 1 \) vector of gravity terms. Each element of \( M(q) \) and \( G(q) \) is a complex function, which depends on \( q \), the position of all joints of the manipulator. Each elements of \( V(q, \dot{q}) \) is a complex function in both \( q \) and its derivatives. The vector \( \tau \) gives the forces and torques in the joints.

It appears that the dynamic equations are less difficulty to obtain in Cartesian space than in joint space. This is a due to the lack of a closed form formulation of the forward kinematics. Thus, one seeks the equation
\[ J^T(P) \cdot \tau = M(P) \cdot \ddot{P} + V \cdot P + P + G(P) \quad (17) \]

The notations are the same as the previous equation but the joint coordinate's \( q \) have been replaced by Cartesian coordinates \( P \). As a consequence the joint force-vector \( \tau \) has been replaced by the force vector \( JT(P) \cdot \tau \), which is the Cartesian forces at the end-effector. By using the inverse kinematics we can get closed form solution, quickly and easy calculation as in Fig. 2.

\[ \text{Fig. 2 Joint-based control scheme.} \]

where \( P_{des} \) is the desired Cartesian space coordinate, \( L_{des} \) is the desired joint space coordinate, \( \delta L \) is the error between the desired and the actual position, \( \tau \) is the control signal to the robot, and \( L \) is the measured position.

The advantages with this control scheme are that the kinematics computations are performed outside the loop. This control scheme generally results in a controller running at a higher sampling frequency than Cartesian-based controller, which would, in general, increase the stability and disturbance rejection capabilities of the system.

VI. TRAJECTORY GENERATION

To make the manipulator move from one position to another in a smooth fashion we need some way to generate trajectories. The desired movements of a manipulator are usually described in a special robot programming language. In this language not only the desired position is given but also the desired speed, acceleration and deceleration, which is called a velocity profile, an example is shown in Fig. 3. These commands are given to the path planner, which computes a number of so called via points. These via points are points, which the manipulator tool-frame should pass through on its way to the desired position.

\[ \text{Fig. 3 Velocity Profile.} \]

The path-generator takes these via points and calculates the desired position for the position controller. The desired position is updated at a certain rate which is called the path update rate, which is show in Fig. 4.

\[ \text{Fig. 4 Path update rate.} \]

In Fig. 5 we have a diagram of how the complete trajectory generation scheme could look like.

\[ \text{Fig. 5 Complete trajectory generation scheme} \]

There are two schemes used for path generation: joint space scheme and Cartesian based scheme. In the joint space scheme a smooth function is found for the joints which moves through the \( v_{ia} \) points and end at the goal point. The time for each segment is the same for all joints which means they will reach the \( v_{ia} \) point at the same time. Hence, joint space schemes achieve the desired position and orientation at the \( v_{ia} \) points. In between via points the shape of the path, will rather simple in joint space, is complex if described in Cartesian space. Joint space schemes are usually the easiest to compute. Paths computed in joint space can ensure that via points are attained, even when these path points were specified by means of Cartesian frames. However, these spatial shape of the path taken by the end-effector is not a straight line through space, but rather, it is some complicated shape which depends on the particular kinematics of the manipulator.
In Cartesian based path generation schemes, the functions, which are splinted, to gather to form a trajectory are functions of time, which represent Cartesian variables. These paths can be planned directly from the user’s definition of path points without first performing inverse kinematics. However, Cartesian schemes are computationally expensive to execute since at run time, inverse kinematics must be solved at the path update rate. That is, after the path is generated in Cartesian space, as a last step the inverse kinematics calculation is performed to calculate desired joint positions. Because of the relatively easy calculations for the inverse kinematics of the Stewart platform a Cartesian based scheme could be used.

VII. DESIGN OF MODEL REFERENCE ADAPTIVE CONTROLLER

Due to the fact that the closed-form solution of forward kinematics is hard to obtain for Stewart platform structure, six individual controllers are designed in the actuators coordinates. By using the inverse kinematics analysis, a model reference learning control scheme is utilized.

A direct tracking control architecture for class of continuous-time non-linear dynamic systems has been proposed [4]. The simulation results verified the effectiveness of the proposed control algorithm. In [5] a design method of Fuzzy control systems depending on trial and error has been presented, and effective and convenient support tools for the study and design of Fuzzy control systems have been introduced. A self-learning Fuzzy logic system for (MIMO) plants has been introduced [6], where a plant model is not required for training. Instead, training is guided by observations of plant responses to inputs.

J. R. Layne and K.M. Passino introduced a theoretical study for controlling cargo ship steering by using FMRLC that depends on changing the final controller output by adding direct correction for the final value based on defined model [7]. They extended their theoretical study for the applications to two degrees of freedom manipulator [8].

A learning control system is designed so that its “learning controller” has the ability to improve the performance of the closed loop system by utilizing feedback information from the plant. In this section we introduce the “fuzzy model reference learning controller” (FMRLC), which is a (direct) model reference adaptive controller. The term “learning” is used as opposed to “adaptive” to distinguish it from the approach to the conventional model reference adaptive controller for linear systems with unknown plant parameters.

The functional block diagram of the FMRLC is shown in Fig.6. It has three main parts: the fuzzy controller to be tuned, the reference model and the learning mechanism. We use discrete time signals since it is easier to explain the operation of the FMRLC for discrete time systems.

The FMRLC uses the learning mechanism to observe numerical data from a fuzzy control system. Using this numerical data, it characterizes the current performance of the fuzzy control system and automatically adjusts the fuzzy controller so that the system performance can meet some given objectives. Basically, the fuzzy control system loop (the lower part of Fig. 6) operates to make \( y(kT) \) track \( r(kT) \) by manipulating \( u(kT) \), while the upper-level adaptation control loop (the upper part of Fig.6) seeks to make the output of the plant \( y(kT) \) track the output of the reference model \( y^*(kT) \) by manipulating the fuzzy controller parameters.

The Fuzzy Controller and Fuzzy Reasoning

The fuzzy logic controller includes three important steps: Fuzzification, fuzzy reasoning (decision making) and defuzzification. The inputs to the fuzzy controller are the error \( e(kT) \) and change in error \( c(kT) \).

The basic operation of the inference process is to determine the values of the controller output based on the contributions of each rule in the rule base. One method of storing the rule base is the use of the Macvicar-Whelan control matrix (Table 1). This matrix is designed so that if the desired output is realized with zero change in error, then the output remains constant. However, if the output is different from the desired response, the rules produce an output signal based on the human knowledge of the operating system. Each element of the matrix describes a rule of the form:

\[
\text{If } e \text{ is } E^j \text{ and } c \text{ is } C^l \text{ Then } u \text{ is } U^m
\]  

where \( E^j \) is the \( j^{th} \) linguistic value associated with \( e(kT) \), \( C^l \) is the \( l^{th} \) linguistic value associated with \( c(kT) \) and \( U^m \) is the \( m^{th} \) linguistic value associated with \( u(kT) \).
<table>
<thead>
<tr>
<th>e(t)</th>
<th>N B</th>
<th>N S</th>
<th>Z O</th>
<th>P S</th>
<th>P B</th>
</tr>
</thead>
<tbody>
<tr>
<td>N B</td>
<td>N B</td>
<td>N B</td>
<td>N B</td>
<td>N S</td>
<td>Z O</td>
</tr>
<tr>
<td>N S</td>
<td>N B</td>
<td>N B</td>
<td>N S</td>
<td>Z O</td>
<td>P S</td>
</tr>
<tr>
<td>Z O</td>
<td>N B</td>
<td>N S</td>
<td>Z O</td>
<td>P S</td>
<td>P B</td>
</tr>
<tr>
<td>P S</td>
<td>N S</td>
<td>Z O</td>
<td>P S</td>
<td>P B</td>
<td>P B</td>
</tr>
<tr>
<td>P B</td>
<td>Z O</td>
<td>P S</td>
<td>P B</td>
<td>P B</td>
<td>P B</td>
</tr>
</tbody>
</table>

NB (Negative Big), NS (Negative Small), ZO (Zero), PS (Positive Small) and PB (Positive Big).

We must choose initial values for each of the output membership functions. For example, for an output universe of discourse [-1, 1] we could choose triangular shaped membership functions with width of 0.4 and centers at zero.

The Reference Model
The goal of the FMRLC is to make the closed-loop system behave like a given “reference model”. Here we choose the output of the inverse kinematics to be the reference model scheme for the system. Then the system will try to follow the reference axis data as a reference model.

Learning Mechanism
The learning mechanism tunes the rule base of the direct fuzzy controller by changing the centers of the output membership functions so that the closed loop system behaves like the reference model. These rule base modifications are made by observing data from the controlled process, the reference model and the fuzzy controller. The learning mechanism consists of two parts: a “fuzzy inverse model” and a “knowledge base modifier”. The fuzzy inverse model performs the function of mapping \( y_i(kT) \) to change the process input \( u(kT) \). The knowledge base modifier performs the function of modifying the fuzzy controller’s rule base to affect the process inputs.

Fuzzy Inverse Model.
Similar to the fuzzy controller, the fuzzy inverse model shown in Fig. 6 produces \( p(kT) \) by using the fuzzy inference mechanism with the rule of Table 1.

Given that \( y_i \) and \( y_c \) are inputs to the fuzzy inverse model, the rule base for the fuzzy inverse model contains rules of the form:

If \( y_e \) is \( Y_e^j \) and \( y_c \) is \( Y_c^l \) Then \( p \) is \( P^m \)  \hspace{1cm} (19)

Where \( Y_e^j \) and \( Y_c^l \) denote linguistic values and \( P^m \) denotes the linguistic value associated with the \( m^{\text{th}} \) output fuzzy set.

The value of \( p(kT) \) gives the necessary changes in the plant input to reduce the plant error.

Knowledge Base Modifier
Given the necessary changes in the input to the plant, to force the error \( y_e \) to zero, the knowledge base modifier changes the rule base of the fuzzy controller so that the control action \( u(kT) \) will be modified by the action of \( p(kT) \).

By modifying the fuzzy controller’s knowledge base, we may force the fuzzy controller to produce a desired output. Let \( b_m \) denote the center of the symmetric membership function associated with \( T^m \) and the knowledge base modification is performed by shifting the values of \( b_m \):

\[
b_m(kT) = b_m(kT - T) + p(kT)
\]  \hspace{1cm} (20)

This modification affects only on the centers of membership functions that are used in the current fuzzy reasoning.

VIII. EXPERIMENTAL RESULTS
The real time implementation was carried out using the sampling time \( T \) of 2msec and the same reference trajectory input with 10kg payload and 2.5MPa supply pressure. The robot limbs are hydraulic cylinders of 20mm inners diameter, 15mm rods diameter and 90mm full strokes. The control algorithm is implemented by a personal computer, based on a 1.66 GHz Pentium processor.

An experimental test was conducted, at first, by using a fixed gain classical controller (PID) as follows,

\[
u[kT] = K_p \cdot e[kT] + K_i \cdot \sum_{j=0}^{i=k} e[jT] + K_d \cdot c[kT]
\]  \hspace{1cm} (21)

where the controller parameters \( \{K_p, K_i, K_d \} \) were adjusted manually to get faster response and smaller overshoots with zero steady state error as possible. The method of design that we work with each axis whenever the other axis is off and we try to choose the gains of the PID controller to get the best result as possible. When the whole system works the gains values slightly changed due to the coupling effect.

The reference input of the manipulator at the teaching center point (TCP) was a step input with amplitude of 10mm. Fig. 7.a shows the response of an axis based on PID controller, while the other axes were off. The system response was coincident with the reference after 0.33sec.

In Fig. 7.b the test was repeated for the same axis while the same inputs were fed to all of the other axes. It is noticed that the system response reached the steady state after about 0.7sec and small overshoot appeared. As we tried to improve the response, the confliction effect of the gain changes limited the ability of improving. For example, if we want to increase the response speed by increasing the proportional or the derivative gains, the stability gets worse, and if we want to improve it by increasing the integral gain the speed becomes slower and so on. It is clear the difference between the two responses is according to the axes coupling effects and the change of the supply pressure.

The system response based on FMRLC is introduced in Fig. 8 for the same axis under the same reference input. As before, the other axes are off in Figs. 8. (a) and (b). It is noticed the controller has the ability to provide robustness with faster response (the response arrived at the steady state region in 0.18sec).

The pressure changes are shown in Figs 9 and 10 with the same conditions of Fig. 7. When only one axis was driven...
based on PID controller, the supply pressure change was within 0.5MPa and when all axes were on it was more than 1.0MPa. While, with FMRLC (when only one axis was on) the pressure change was within 0.38MPa and when all axes were on it was about 0.76MPa.

Although, the inertia difference of the moving parts and the large pressure change happened, FMRLC can regulate the system to keep the response behave like the model. This is due to the fact that the learning ability of the controller can eliminate the error between the plant response and the model response. In Fig.11, the modification by the adaptation signals \( \pi(kT) \) is shown for the case of Fig.8 (b), where the adaptation continued until each of the axes responses became coincident with the reference inputs.
For the testing of motion control, an example of circle reference input to the TCP with 40 mm diameter was applied. The TCP response and the axes responses based on the proposed controller is presented in Figs 9 and 10 respectively. It is clear the ability of the controller to force the system motion to follow the model reference trajectory as a desired behave.

IX. CONCLUSION

The control problem of an electro-hydraulic six axis motion base system is studied in this paper. A self-learning intelligent controller based on fuzzy logic and control knowledge has been given. It is shown that the intelligent fuzzy controller has a good learning effect and a robust control performance. The main advantage of the proposed controller:

- The control algorithm does not need a precise model of the plant and it is easy to implement even on small microprocessor systems.
- The proposed algorithm has superior response characteristics compared with the traditional PID control.
- Through the experiments using the servo valve controlled cylinder system, the effectiveness of the proposed control was confirmed.

The successful application of the adaptive inverse model control on the hydraulic power system is a breakthrough in this field. Extending this proposed policy for including the force control loops is promise study.

REFERENCES

Ayman A. Aly is the head of Mechatronics Section at Taif University, Saudi Arabia since 2008. Prior to joining Taif University, he is also one of the team who established the “Mechatronics and Robotics Engineering” Educational Program in Assiut University in 2006. He was in the Managing and implementation team of the Project “Development of Mechatronics Courses for Undergraduate Program” DMCUP Project- HEEP Grant A-085-10 Ministry of Higher Education – Egypt, 2004-2006.

In additions to 5 text books, Ayman A. Aly is the author of more than 75 scientific papers in Refereed Journals and International Conferences. He supervised some of MSc. and PhD. Degree Students. His main areas of research are Robust and Intelligent Control of Mechatronics systems, Automotive control systems, Thermofluid systems modeling and simulation.

Ahmed Abo-Ismail is the Provost “Vice President for Education and Academic Affairs” at Egypt-Japan University of Science and Technology. Prior to joining E-JUST, Professor Abo -Ismail served as the Dean of Faculty of Engineering in Assiut University from 2003 to 2006 and from 2000 to 2003 he was the Vice-Dean for Graduate studies and Research. Professor Abo-Ismail established the “Mechatronics and Robotics Engineering” Educational Program in Assiut University in 2006.

Professor Abo-Ismail’s main areas of research are Robust Control, Intelligent Fuzzy Control, Variable Structure Control, Design and Control of Smart Grippers, Design and Control of Intelligent Piezoelectric Actuators, Development and control of Versatile Soft- Gripper, Intelligent Mechatronics System, Intelligent Control of a Flexible Robot Arm, VR -Hepatic Surgical Simulators, Smart sensors and Land mines detections and Surgical Manipulators and Assistive devices.