The SOF-PID Controller for the Control of a MIMO Robot Arm

Hassan B. Kazemian, Senior Member, IEEE

Abstract—The application of a self-organizing fuzzy proportional–integral–derivative (SOF-PID) controller to a multiple-input–multiple-output (MIMO) nonlinear revolute-joint robot arm is studied in this paper. The SOF controller is a learning supervisory controller, making small changes to the values of the PID gains while the system is in operation. In effect, the SOF controller replaces an experienced human operator, which otherwise would have readjusted the PID gains during the system operation. The three PID gains are tuned using classical tuning techniques prior to the application of the SOF controller at the supervisory level. Two trajectories of step input and path tracking were applied to the SOF-PID controller at the setpoint. For comparison purposes, the same experiments were repeated by using the self-organizing fuzzy controller (SOFC) and the PID controller subject to the same information supplied at the setpoint. For the step input, the SOF-PID controller produced a faster rise time, a smaller steady state error, and an insignificant overshoot than the SOFC and the PID controller. For the path tracking experiments, the output trajectories of the SOF-PID controller followed the required path closer and smoother than the output trajectories of the SOFC and the PID controller.

Index Terms—Performance index table, self-organizing fuzzy proportional–integral–derivative (SOF-PID) controller, Ziegler–Nichols method and robotics.

I. INTRODUCTION

Any industrial processes are controlled using proportional–integral–derivative (PID). There are commercially available PID modules that have switches for the operator to turn in order to set the values of each of the three PID gains. Despite of extensive use of PID in conventional control problems, its performance in industrial applications remains limited. For instance for a step input, the conventional tuning techniques for the PID gains $K_P$, $K_I$, and $K_D$ do not produce the desired output response and further reduction in the steady state error, overshoot and rise time is required. Retuning the PID gains may rectify some areas of the output response and deteriorates the other areas. For example, an increase in proportional gain $K_P$ improves the rise time but increases the possibility of the overshoot. A small rise in the integral gain $K_I$ decreases the steady state error but increases the possibility of instability. An increase in the derivative gain $K_D$, reduces the overshoot, however, it may cause fluctuations in the process output in the presence of high rates of change such as noise.

A fuzzy logic controller can be used at a supervisory level readjusting the PID gains at the actuator level, during the system operation. This master and slave control concept could be achieved using nonlearning or learning fuzzy controllers. The rule-based fuzzy PID controller is a nonlearning controller which has been studied in [1]–[7]. The approach in [1] utilizes classical tuning techniques to obtain nominal values for the PID gains. Then, by using an appropriate fuzzy matrix, the gains are modified slightly. Step input was applied to Tzafestas’s [1] technique and compared with conventional tuning techniques such as Ziegler–Nichols and Kalman. He concluded with an improvement in the transient and steady behavior of the closed loop system. Passino [2] argues a mixture of intelligent control at supervisory controller level in a form of fuzzy rule-based, readjusting the conventional PID gains, will have many industrial applications. Malki [3] introduced a fuzzy PID controller and applied it to a nonlinear flexible-joint robot arm. He concluded that the fuzzy PID controller produces a remarkable tracking performance better than PID controller, given the same information supplied at the setpoint. Li [4] designed a hybrid fuzzy proportional gain plus conventional integral-derivative (fuzzy P + ID). Li argued that the simulation results for time responses of step control, ramp tracking control and sinusoidal tracking control, demonstrate the effectiveness of the fuzzy P + ID in comparison with the conventional PID controller. Mudi [6] proposed an autonomous tuning scheme for PI and PD fuzzy controllers, where the output scaling factor is adjusted by fuzzy rules during the system operation. He applied the step input to the PI and PD-type fuzzy controllers and compared it with a conventional fuzzy controller. Mudi concludes for a step input a remarkable improvement is achieved in rise time, overshoot, settling time and steady state error. Golob [7] used three rule-based fuzzy controllers for each of the PID gains and applied the controllers to a magnetic suspension system. Golob’s conclusion is that the rule-based fuzzy controller gives a better performance output over a traditional linear PID controller.

The difficulty with the nonlearning rule-based fuzzy PID controller is that the rules are prewritten in the rule buffer block. These rules are the result of the experience gained by a human operator. The data base of the nonlearning rule-based fuzzy PID controller is specific to a known process, where all the nonlinearities within the system is taken into account in advance. Therefore, for every process control the experience of a human operator is fuzzified and a set of rules are obtained for the rule buffer block of the nonlearning fuzzy PID controller.

In contrast, the self-organizing fuzzy PID (SOF-PID) controller is a learning controller. The SOF-PID controller is the extension of the rule-based fuzzy PID controller with an additional learning mechanism. The SOF-PID controller generates...
its own control rule strategies automatically according to the changes encountered both at the setpoint and from the process under control, starting with no rules in the rule buffer block [8]. The self-organizing fuzzy controller (SOFC) as presented here, was initially introduced by Mamdani [9] and later on developed by Procyk [10]. As explained, the SOFC is a learning controller, which creates its own control rule strategy in response to a new experience and without the guidance of an experienced human operator. The work in [11]–[16] applied the SOFC to a revolute-joint robot arm, and concluded that the SOFC produces a better process output than the PID controller. After an extensive study of the SOFC and the PID controller, the obvious progression was to use the learning fuzzy controller as a supervisory controller to readjust the gains of the PID controller at actuator level. Since the PID controller is perceived to be one of the leading controllers in industrial processes, it is not necessary to rip out an existing system, such as a conventional PID, which works well and is already in operation in a large number of industrial applications. The aim of this paper is to consider how to compensate for the limitations of the PID controller by automatic means.

For the SOF-PID controller, the step input and the path tracking trajectories were applied to a multiple-input–multiple-output (MIMO) nonlinear revolute-joint robot arm, with presence of noise and time variant system. In the robot arm, the moment of inertia varies with time due to the movements of the links. The nonlinearities in a revolute-joint robot arm are due to backlash, friction, and motor characteristics. The same experiments were repeated using the SOFC and the PID controller, subject to the same information supplied at the setpoint. The results of the experiments for the SOF-PID controller were compared with the SOFC and the PID controller, in order to evaluate the suitability of the SOF-PID controller for dynamic system applications and also obtain some information about the tuning procedure.

Section II describes the structure of SOF-PID controller. Section III, outlines the characteristics of the nonlinear revolute-joint robot arm used in this work as a test bed to study the behavior of the SOF-PID controller, the SOFC and the PID controller. Section IV explains the experimental results of step input and path tracking trajectories applied to the SOF-PID Controller, and compares the results with the SOFC and the PID controller. Finally Section V discusses and concludes the findings of this work.

II. DEVELOPMENT OF THE SELF ORGANIZING FUZZY PID CONTROLLER

This section describes a detailed development of the single-input–single-output (SISO) SOF-PID controller. The block diagram of the SOF-PID controller is shown in Fig. 1. The broken lines in the block diagram show two controllers, the SOF at the supervisory controller level and the PID at the actuator level. The tuning procedure for the SOF-PID controller requires the following three stages. First, the three PID gains are tuned using a conventional tuning method. Second, the SOF at the upper supervisory level readjusts the proportional PID gains continuously to improve the process output response, while the system is in operation. Third, the corresponding readjusting values for the integral and derivative PID gains are calculated using Ziegler–Nichols tuning method, during the system operation.

In Fig. 1, an error (1) from the actuator level is fed into the supervisory level to enable the SOF to examine the process output and the setpoint behavior. There is also a proportional PID gain input from the PID controller block to the history of past states block via the PID input section block, to continuously update the values of the history of past states. The SOF makes small adjustments to the proportional PID gains, and feeds the results from the output section block into the PID controller block. The corresponding readjusting values of the integral and derivative PID gains are calculated using Ziegler–Nichols tuning method, and also fed from the output section block into the PID controller block. In the error input section block, the change in error is produced by using (2). This block fuzzifies the error and the change in error. In the process of fuzzification, scaling and quantization of the error and the change in error take place. The error scaling factor is denoted by $E_S$. However, two different values for the change in error scaling factors for the inference mechanism block and the performance index table block are used to obtain a good damping characteristic and a better convergence at the setpoint, denoted by $CES_\Sigma$ and $CES_P$, respectively. The fuzzified input error to the inference mechanism block and the performance index table block are represented by $FUZZ\cdot ERROR_i[-]$. The fuzzified input change in errors to the inference mechanism block and the performance index table block are denoted by $FUZZ\cdot C\cdot ERROR_i[-]$ and $FUZZ\cdot C\cdot ERROR_i[-]$, respectively, where $i$ is the number of links. In the PID input section block, the PID gain $K_P$ is also fuzzified. For fuzzification, the PID gain $K_P$ is scaled and quantized. The scaling factor for the proportional gain is $SK_P$:

1. $\text{error} (e_l) = \text{setpoint} (s_i) - \text{process~output} (p_{oi})$
2. $\text{change~in~error} (\Delta e_l) = \text{error} (e_l - 1) - \text{error} (e_l)$

where $l$ is sampling instant.

Fuzzification provides the conditions required for the PID gains to be between $-6$ and $+6$. The linguistic codes used are as follows:
TABLE I
A SET OF LINGUISTIC RULES

<table>
<thead>
<tr>
<th>Rule</th>
<th>Linguistic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>IF E is NL and CE is NL or NM or NS then P is ZO</td>
</tr>
<tr>
<td>2.</td>
<td>IF E is NL and CE is ZO or PS or PM or PL then P is NL</td>
</tr>
<tr>
<td>3.</td>
<td>IF E is NM and CE is NL or NM or NS then P is ZO</td>
</tr>
<tr>
<td>4.</td>
<td>IF E is NM and CE is ZO or PS or PM or PL then P is NM</td>
</tr>
<tr>
<td>5.</td>
<td>IF E is NS and CE is NL or NM or NS then P is ZO</td>
</tr>
<tr>
<td>6.</td>
<td>IF E is NS and CE is ZO or PS or PM or PL then P is NS</td>
</tr>
<tr>
<td>7.</td>
<td>IF E is ZO and CE is NL or NM or NS or ZO or PS or PM or PL then P is ZO</td>
</tr>
<tr>
<td>8.</td>
<td>IF E is PS and CE is NL or NM or NS then P is PS</td>
</tr>
<tr>
<td>9.</td>
<td>IF E is PS and CE is ZO or PS or PM or PL then P is ZO</td>
</tr>
<tr>
<td>10.</td>
<td>IF E is PM and CE is NL or NM or NS or ZO then P is PM</td>
</tr>
<tr>
<td>11.</td>
<td>IF E is PM and CE is PS or PM or PL then P is PM</td>
</tr>
<tr>
<td>12.</td>
<td>IF E is PL and CE is NL or NM or NS or ZO then P is PL</td>
</tr>
<tr>
<td>13.</td>
<td>IF E is PL and CE is PS or PM or PL then P is ZO</td>
</tr>
</tbody>
</table>

Negative Large (NL) = -6 or -5;
Negative Medium (NM) = -4 or -3;
Negative Small (NS) = -2 or -1;
Zero (Z0) = 0;
Positive Small (PS) = +1 or +2;
Positive Medium (PM) = +3 or +4;
Positive Large (PL) = +5 or +6.

The performance index table block consists of a set of linguistic rules. In effect, it changes the values of \( K_{PI} \) slightly when it is observed that the process output is deflecting from the setpoint, during the system operation. The performance index table produces an appropriate gain correction \( P \) in order to readjust the value of the proportional PID gain. The performance index table for the SOF-PID controller is designed as such that if the process output follows or approaches the setpoint, then no action is taken and the value of the gain correction \( P \) is equal to zero. If the process output is deflecting from the setpoint, then some past values of the PID gains are responsible for the present poor performance and a gain correction \( P \) is selected. In Table I, by using (1) and (2) from the definitions of error \( (e_t) \) and change in error \( (\Delta e_t) \), a set of thirteen linguistic rules are obtained, [17]; and from these linguistic rules the performance index table is accomplished. The performance index table Table II, is used for all the experiments carried out with the SOF-PID controller.

The values of the gain correction \( P \) from Table II, are added to the values of the proportional PID gains \( K_{PI} \) from the history of past states block in the rule reinforcer block to produce new values for the rule buffer block. The history of past states block is a storage for the past fuzzified values of the proportional PID gains \( K_{PI} \), where \( I \) is the sampling instant, \( \eta \) represents number of samples before the present sample, and \( i \) is number of links. The number of samples \( \eta \) of the proportional PID gains \( K_{PI} \) in the history of past states block depends on the delay-in-reward (\( DIR \)), which in turn the DIR depends on the time-lag of the process. These past states could be obtained on the basis of first-in–first-out. The role of the rule reinforcer block is to manufacture and update new rules. If the gain correction \( P \) is zero from the performance index table, then no rule modification takes place. If \( P \) is not zero, then the rule modification takes place in the rule reinforcer block for sampling instant \( I \) using the following equation:

\[
K_{PI}(\text{reinforcer}) = K_{PI}(\text{rule buffer}) + P \tag{3}
\]

where \( K_{PI}(\text{rule buffer}) \) is the fuzzified proportional PID gain from the history of past states block, and \( K_{PI} \) is the new value of the proportional PID gain produced in the rule buffer block. In the experiments carried out, seven values of \( K_{PI}(\text{rule buffer}) \) were stored in the history of past states block. The rule buffer block stores all the appropriate values of the PID gain \( K_{PI} \) to be used in the inference mechanism block. The rules are the exact control rule strategy, which have been obtained from the learning section of the supervisory level. The \( K_{PI} \) values change continuously, according to new conditions encountered both at the setpoint and from the process under control.

From the inference mechanism block an output is obtained by combining the rules from the rule buffer block, and the \textsc{fuzz-Error}\( [\tilde{a}] \) and \textsc{fuzz-C-Error}\( [\tilde{a}] \) from the error input section block. This concept is known as the compositional rule of inference, [18]. The inference mechanism block is responsible for implication function and defuzzification. Implication function produces an output from the inference mechanism block, and this output is defuzzified to create an input for the output section block. There have been numerous experiments carried out with different implication functions and defuzzification methods for the proposed SOFC. Yamazaki [19], used the Max–Product implication function in conjunction with the center of gravity defuzzification method and concluded that the process output is smoother. Lembessis [20] combined the min implication function [21] with the mean of maxima defuzzification [22] method, and argued that this combination produces a faster convergence to the setpoint. Tanscheit [14] utilized these findings by Yamazaki and Lembessis, and concluded that in general in the control of a SISO revolute-joint robot arm, the min implication function with the mean of maxima produce better and smoother results for the process output. There were some initial experiments carried out in this research to apply the SOF-PID controller and the SOFC to a revolute-joint robot arm using the max–product implication function and the center of gravity defuzzification method, as well as the min implication function and the mean of maxima defuzzification method. The experimental results showed that the min implication function with the mean of maxima produce a faster convergence to the setpoint and a smoother process output. As a result, all the experiments outlined here are with the min implication function and the mean of maxima defuzzification method. An example of the min implication function and the mean of maxima defuzzification method for the SOF-PID controller is outlined below. For a step input during the system operation, assume the outputs from the error input section block are \textsc{fuzz-Error}\( [\tilde{a}] \) = 1.5 and \textsc{fuzz-C-Error}\( [\tilde{a}] \) = 1.0 for the 4 values of \( K_{PI} \) from the rule buffer block. The min implication function is demonstrated mathematically in equation (4) and graphically in Fig. 2, respectively. In Fig. 2, for each value of \( K_{PI} \) the membership function \( u(x) \) is plotted against the universe of discourse \( U_{PI} \).

\[
u(\textsc{fuzz-Error}\{a\} \text{ AND } \textsc{fuzz-C-Error}\{b\})
\begin{cases}
\min(0.25, 0.50), \\
\min(0.75, 0.50)
\end{cases}
\min(0.0, 0.0), \min(0.0, 0.0)
\]

\[
u(x) = \{0.25, 0.50, 0.0, 0.0\}, \tag{4}
\]
Equations (5) and (6) and Fig. 3 present the calculations of the mean of maxima defuzzification method. The mean of maxima is defined [23] by taking the average between two elements in the universe of discourse which correspond to two largest values of the membership functions. \( U_{F_{\text{max}}} \) is the highest value of the membership function, and \( U_{F_{\text{max}-1}} \) is the second highest value. In this example in (6), both values are chosen, since only two rules are contributing in Fig. 3. If three rules were contributing, then two highest values of membership function would have been chosen

\[
\text{mean of maxima} \quad U_{P|L} = \frac{U_{F_{\text{max}}} + U_{F_{\text{max}-1}}}{2} \tag{5}
\]
\[
\text{mean of maxima} \quad U_{P|L} = \left[2 \theta + 0 \theta\right]/2 = 1.0 \tag{6}
\]

The output section block requires a nonfuzzy input signal to be fed into the PID controller. Here, the fuzzy signal \( U_{P|L} \) is dequantized and descaled. In the output section block, small changes are passed on to be added to the \( K_P \) gain in the PID controller block. The values of the proportional PID gains are slightly readjusted using the following equations:

\[
K_{P_{\text{after-apps}}} = K_{P_{\text{before-apps}}} + U_{P|L} \times K_1 \tag{7}
\]
\[
K_{I_{\text{after-apps}}} = K_{I_{\text{before-apps}}} + U_{P|L} \times K_2 \tag{8}
\]
\[
K_{D_{\text{after-apps}}} = K_{D_{\text{before-apps}}} + U_{P|L} \times K_3 \tag{9}
\]

where \( K_{P_{\text{after-apps}}} \), \( K_{I_{\text{after-apps}}} \) and \( K_{D_{\text{after-apps}}} \) on the left-hand side of (7)–(9) represent the PID gains after application of the supervisory level. \( K_{P_{\text{before-apps}}} \), \( K_{I_{\text{before-apps}}} \) and \( K_{D_{\text{before-apps}}} \) on the right-hand side represent the PID gains prior to the changes. \( K_1 \), \( K_2 \) and \( K_3 \) are descaled coefficients, which their values are determined by trial and error. \( K_1 \), \( K_2 \) and \( K_3 \) values depend on the fuzzification conditions implemented in the system; for example, \( NL = -6 \) or \(-5 \) and \( PL = +5 \) or \(+6 \). Because the values of \( K_P \), \( K_I \) and \( K_D \) change at different rates, three descaled coefficients are used. For instance, the range of variations are higher for \( K_P \) as opposed to \( K_I \) and \( K_D \). The descaled coefficients \( K_{1A} \), \( K_{2A} \), and \( K_{3A} \) in the output section block are selected to be different for each link. For a three-link revolute-joint robot arm, \( K_{1S} \), \( K_{2S} \), and \( K_{3S} \) are the descaled coefficients for the shoulder; \( K_{1A} \), \( K_{2A} \), and \( K_{3A} \) are for the arm; and \( K_{1H} \), \( K_{2H} \), and \( K_{3H} \) are for the hand. \( U_{P|L} \) is the fuzzified output from the inference mechanism block for the proportional gain, and similarly \( U_{I|T} \) for the integral gain and \( U_{D|O} \) for the derivative gain.

At the supervisory level only, \( K_P \) is readjusted, and simultaneously, the corresponding readjusting values for \( K_I \) and \( K_D \) are calculated using the Ziegler–Nichols tuning method [24]. An experiment is carried out on the process using the Ziegler–Nichols tuning method, as follows. First, the process is controlled using proportional gain \( K_P \). The value of \( K_P \) is slowly increased until continuous oscillations is happened. At the time of oscillation the values of the gain \( K_O \) and the oscillation period \( T_O \) are noted. The value of the oscillation period \( T_O \) is initially determined off line and used as a fixed value on line. The method assumes that the proportional gain \( K_P \) is 60% of the gain at the time of oscillation, \( K_P = 0.6 \times K_O \). The integral time constant \( T_I \) is 50% of the oscillation period, \( T_I = 0.5 \times T_O \). Finally, the derivative time constant is 12.5% of the oscillation period, \( T_D = 0.125 \times T_O \). The Ziegler–Nichols method is based on the continuous systems and can also be used on the discrete cases for a fast sampling time. In a conventional PID controller, \( K_I = K_P/T_I \) and \( K_D = K_P/T_D \), the Ziegler–Nichols relationships could be used in these equations to produce two new equations, given as follows:

\[
K_I = 2K_P/T_O \tag{10}
\]
\[
K_D = K_P T_O / 8 \tag{11}
\]

In (10) and (11), \( U_{P|L} \) is used in place of \( K_P \), as \( U_{P|L} \) is the fuzzified output from the inference mechanism block. Substituting (10) and (11) into (8) and (9), the following equations are obtained:

\[
K_{P_{\text{after-apps}}} = K_{P_{\text{before-apps}}} + U_{P|L} \times K_1 \tag{12}
\]
\[
K_{I_{\text{after-apps}}} = K_{I_{\text{before-apps}}} + (2U_{P|L}/T_O) \times K_2 \tag{13}
\]
\[
K_{D_{\text{after-apps}}} = K_{D_{\text{before-apps}}} + (U_{P|L}/T_O / 8) \times K_3 \tag{14}
\]

The PID controller block feeds its output into the process. The PID controller output for a continuous (15) and a discrete (16) system is outlined as follows:

\[
u(t) = K_P e(t) + \frac{K_P}{T_I} \int_0^t e(t) \, dt + K_P T_D \frac{d}{dt}(t) \tag{15}
\]
\[
U(Z) = K_P + \frac{K_P T_Z}{T_I (Z - 1)} + K_P T_D (Z - 1) \tag{16}
\]
where \( u(t) \) is the controller output, \( e(t) \) is the error signal and \( T \) is the sampling period.

Alternatively, the aforementioned equation may be presented as

\[
U(Z) = K_P + \frac{K_f T Z}{(Z - 1)} + K_D \frac{Z - 1}{T Z}. \tag{17}
\]

Equation (17) can be approximated by the digital form

\[
u(kT) = K_P e(kT) + K_f T \sum_{l=0}^{k} e(lT) + \frac{K_D}{T} [e(kT) - e((k-1)T)] \tag{18}
\]

where \( k \) is the sample number (instant) and \( e(kT) - e((k-1)T) \) is the change in error.

In this paper, a revolute-joint robot arm is used to represent a process. It is called revolute-joint because the manipulator model describes the robot arm by its rotational characteristics.

### III. Kinematics and Dynamics of the Robot Arm

The mathematical model of a revolute-joint robot arm is taken as a nonlinear dynamic system and employed as a tool to study the behavior of the SOF-PID controller, the SOFC and the PID controller. The mathematical model outlines the robot arm by its rotational characteristics and comprises of three sections, the structure of the arm, the inverse arm and the forward arm [25]. The structure of the robot arm consists of two sections, the kinematics and the dynamics. The kinematics describe the relative positions between the links of the arm and give the axes of rotation for each of the joints [26]. The dynamics constitute the moment of inertia, the center of mass and the mass for each of the links [27]. The inverse arm is a set of equations which, when evaluated, yield the motor voltages required to produce particular accelerations. This is the inverse of a real arm which produces accelerations given the voltages. The forward arm is the process of applying voltages to each of the motors and calculating the movements of the joints in the robot arm.

The robot arm model can accommodate up to seven links and six joints. The seven links comprise of link 0 to link 6, and the six joints consist of joint 1 to joint 6. In the computer simulation, link 1 is taken as a SISO. A two-link and a three-link represent a MIMO, and link 0 is the static base. In Fig. 4, the Denavit–Hartenburg (D–H) [28] convention describes the kinematics of the links and joints as such that, link \( i \) rotates around the \( Z_{i-1} \) axis of link \( i - 1 \) when joint \( i \) turns. Similarly, link \( i + 1 \), rotates around \( Z_i \) at joint \( i + 1 \), etc. \( X_i \) is related to link \( i \) and points along the common normal of \( Z_i \) and \( Z_{i-1} \). The D–H representation of a link is based on four geometric parameters.
is the angle between links, measuring the joint angle from the \( X_{i-1} \) axis to the \( Z_i \) axis about the \( Z_{i-1} \) axis.

2) \( \alpha_i \) is the twist of the link, the angle between axes \( Z_{i-1} \) and \( Z_i \) about the \( X_i \) axis.

3) \( \ell_i \) is the length of the link, the shortest distance between the \( Z_{i-1} \) and \( Z_i \) axes.

4) \( d_i \) is the distance between the links, from the link \( i-1 \) to the link \( i \) along the \( Z_{i-1} \) axis.

The driving force for each link is an armature controlled dc motor. The voltage is applied at the input of the armature terminals and speed of rotation is produced at the output. A second order differential equation is used to represent the dynamics of a dc motor and load

\[
\frac{d^2 y}{dt^2} + f \frac{dy}{dt} + r(t) = r(t)u
\]

where \( u \) is the input to the process, \( y \) is the output from the process, \( f \) is the friction, and \( r(t) \) is the small friction values which varies with time. In the robot arm, the moment of inertia varies with time due to the movements of the links. The dc motor dynamics is a time variant system, which could represent small friction values and changes in the moment of inertia of the motor and load [29]. By varying the term \( r(t) \) which stands for small friction values, changes in the moment of inertia of the motor and load will take place. A sharp decrease or increase in the moment of inertia makes the system more difficult to control. The third order method of Runge–Kutta [30] is used to integrate the second order dynamic equation. To simulate the noise, a random number generator program is used to produce 4000 different numbers. This is based on a congruent linear random number generator, which gives a distribution close to a rectangular. In accordance with observations made with a practical system, the output is scaled to give a deviation of \( \pm 0.5 \) units which is added to the process output.

IV. IMPLEMENTATION OF THE SOF-PID CONTROLLER

The overall results of the computer simulation for the experiments carried out for the application of SOF-PID controller to a MIMO nonlinear revolute-joint robot arm are described in this section. In the computer simulation for the SOF-PID controller and the SOFC a maximum number of 6 runs per experiment was taken, which was more than sufficient to generate an optimum control rule strategy. The performance of the SOF-PID controller is compared with the SOFC and the PID controller. The experimental results of the step input and the path tacking trajectories are outlined in Sections IV-A and IV-B, respectively.

The PID gains \( K_P, K_I, \) and \( K_D \) are firstly tuned using a conventional tuning method, without the application of the SOF controller at the supervisory level. The experimental results presented in this paper are based on the following method. Firstly, a large value of \( K_P \) is selected and gradually reduced, in order to minimize the process output overshoot. Then, \( K_I \) and \( K_D \) are tuned and finally \( K_P \) is retuned to obtain the best possible process output response. Once the PID gains are tuned, the SOF supervisory controller readjusts the proportional PID gain and the Ziegler–Nichols formula calculates the corresponding readjusting values for the integral and derivative PID gains automatically, during the system operation.

A. Two-Input–Two-Output (TITO) and Three-Input–Three-Output (TTHO) Experiments for Step Inputs

Some of the step input experiences carried out for the SOF-PID controller are described here. The step input was taken as \( 50^\circ \) amplitude for the shoulder, arm, and hand. In general, a higher voltage was required for the shoulder than the arm as the shoulder carried the arm, hence, more weight was exerted on the shoulder. Similarly a higher voltage was required for the arm than the hand. Appropriate level of voltages were applied to each link to produce roughly the same rise time for the shoulder, arm, and hand.

In Fig. 5, an experiment with the following parameters is outlined below using the two SOF-PID controllers: at the supervisory level, \( ES = 0.35, CES_P = 4, CES_S = 12 \) and \( DIR = 7 \). At the supervisory level the shoulder descaled coefficients are \( K_{1S} = 0.5, K_{2S} = 0.05, K_{3S} = 0.12 \) and the arm descaled coefficients \( K_{1A} = 0.4, K_{2A} = 0.05, K_{3A} = 0.11 \). The sampling time is 12 ms, about 1/10 of mechanical time constant. The proportional gain scaling factor for the shoulder and arm is \( SK_P = 0.116 \). At the actuator level the PID gains are \( K_P[S] = 52, K_I[S] = 0.5, K_D[S] = 1.5 \) for the shoulder and \( K_P[A] = 30, K_I[A] = 0.5, K_D[A] = 1.0 \) for the arm. In Fig. 7, the sampling time is 12 ms, and about 1/10 of mechanical time constant. The proportional gain scaling factor for the shoulder and arm is \( SK_P = 0.0116 \). At the actuator level the PID gains are \( K_P[S] = 53, K_I[S] = 1.0, K_D[S] = 1.5 \) for the shoulder, \( K_P[A] = 52, K_I[A] = 0.5, K_D[A] = 1.5 \) for the arm, and \( K_P[H] = 50, K_I[H] = 0.5, K_D[H] = 1.0 \) for the hand.

The experiments for the SOFCs inherit all the parameters from the SOF-PID controller, except the ones outlined below. An experiment for the TITO SOFC with the following characteristics is shown in Fig. 6: the shoulder descaled coefficient
Fig. 5. Step response: using two SOF-PID controllers—run number 5. Scaling: X axis: 12 ms/sample Y axis: outputs—degrees.

Fig. 6. Step response: using two SOF-PID controllers—run number 5, TITO SOFC—run number 6, and two PID controllers. Scaling: X axis: 12 ms/sample Y axis: outputs—degrees.

Fig. 7. Step response: using three SOF-PID controllers—run number 4. Scaling: X axis: 12 ms/sample Y axis: outputs—degrees.

has a slower rise time to the hand output. The more weight is exerted on the links the slower the rise time becomes. By further re-tuning the scaling factors $ES$, $CES_P$, $CES_S$ and $SK_P$, the rise time for the shoulder, arm, and hand may be obtained at the same time.

Using the two SOF-PID controllers and three SOF-PID controllers, the rise time for the shoulder, arm and hand are much faster than the TITO SOFC and THITHO SOFC, and the two PID controllers and three PID controllers, see Figs. 6 and 8. In Figs. 6 and 8, the continuous lines represent the SOF-PID controllers, the broken lines represent the SOFCs and the broken dotted lines represent the PID controllers. Careful tuning of the shoulder descaled coefficients $K_{1S}$, $K_{2S}$, $K_{3S}$, and the arm descaled coefficients $K_{1A}$, $K_{2A}$, $K_{3A}$, and the hand descaled coefficients $K_{1H}$, $K_{2H}$, $K_{3H}$ may produce a faster rise time. The overshoot for the two SOF-PID controllers and three SOF-PID controllers is nonexistence, very similar to the TITO SOFC and THITHO SOFC. The overshoot for the two PID controllers and three PID controllers is noticeably higher than the two SOF-PID controllers and three SOF-PID controllers, and the TITO SOFC and THITHO SOFC. The steady-state error is improved for the SOF-PID controllers compared with the SOFCs and the PID controllers. In Fig. 6, the steady-state errors for the two-link robot arm are about, 1.3% for the two SOF-PID controllers, 1.8% for the TITO SOFC,
Fig. 8. Step response: using three SOF-PID controllers—run number 4, THITHO SOFC—run number 5, and three PID controllers. Scaling: X axis: 12 ms/sample, Y axis: outputs—degrees.

and 2.6% for the two PID controllers. In Fig. 8, the steady state errors for the three-link robot arm are about, 1.5% for the three SOF-PID controllers, 2.0% for the THITHO SOFC, and 3.0% for the three PID controllers.

B. TITO and THITHO Path Tracking Experiments

For the two-link revolute-joint robot arm, the experiments incorporated the simultaneous operation of two individual SOF-PID controllers, one controlling the shoulder movement and the other controlling the arm movement. For the three-link robot arm, three individual SOF-PID controllers were used, one controlling the shoulder movement, one controlling the arm movement and one controlling the hand movement. Each SOF-PID controller regards its joint as a single-input single-output system, learning its rules in the face of cross-coupling effects experienced by the other system.

For the two-link revolute-joint robot arm, the tip of the arm link was assumed to track the square. For the three-link robot arm, the tip of the hand traced the square. $\theta_1$, $\theta_2$ and $\theta_3$ were initially at rest. For the two-link, $\theta_1$ was increased from 0° to 45°, while $\theta_2$ was kept at zero. From then the two joints $\theta_1$ and $\theta_2$ were moved to draw the required square with 20 cm on each side, on the X–Y plane. For the three-link robot arm the shoulder joint angle $\theta_1$ was increased from 0° to 45°, while $\theta_2$ and $\theta_3$ were kept at zero. Two joint angles were varied at any one time using kinematic transformations and joint angles manipulations, to draw a two-dimensional square with 30 cm on each side, on the X–Y plane.

In Fig. 9, a path tracking experiment with the following parameters is outlined, using the two SOF-PID controllers: at the supervisory level, $ES = 0.5$, $CES_P = 6$, $CES_S = 15$ and $DHR = 7$. At the supervisory level, the shoulder descaled coefficients are $K_{1S} = 0.5$, $K_{2S} = 0.05$, $K_{3S} = 0.12$, and the arm descaled coefficients are $K_{1A} = 0.4$, $K_{2A} = 0.05$, $K_{3A} = 0.11$. For the path tracking, a smaller sampling time of 6 ms is used which produced a smoother output response. The proportional gain scaling factor for the shoulder and arm is $SK_P = 1.2$. At the actuator level the PID gains are $K_P[S] = 4$, $K_I[S] = 1.5$, $K_D[S] = 1.25$ for the shoulder, and $K_P[A] = 3$, $K_I[A] = 1.0$, $K_D[A] = 1.0$ for the arm. In Fig. 12, the same parameters are used for the three SOF-PID controllers subject to the same information supplied at the set points, with the following changes in the parameters of the descaled coefficients and the PID gains: at the supervisory level, the shoulder descaled coefficients are $K_{1S} = 0.6$, $K_{2S} = 0.06$, $K_{3S} = 0.12$, the arm descaled coefficients are $K_{1A} = 0.5$, $K_{2A} = 0.05$, $K_{3A} = 0.11$, and the hand descaled coefficients are $K_{1H} = 0.4$, $K_{2H} = 0.05$, $K_{3H} = 0.11$: the proportional gain scaling factor for the shoulder, arm and hand is $SK_P = 1.4$. At the actuator level, the PID gains are $K_P[S] = 5$, $K_I[S] = 2.0$, $K_D[S] = 1.5$ for the shoulder, $K_P[A] = 4$, $K_I[A] = 1.5$, $K_D[A] = 1.25$ for the arm and $K_P[H] = 3$, $K_I[H] = 1.0$, $K_D[H] = 1.0$ for the hand.

For the path tracking experiments, the SOFC utilizes all the parameters used in the SOF-PID controller, except the ones given later. In Fig. 10, the result of a TITO SOFC experiment in tracking a two-dimensional square is described, with the following descaled coefficient values: the descaled coefficient for the shoulder $K_S = 2$, and for the arm $K_A = 1$. A similar experiment for the THITHO SOFC with different values of the descaled coefficients are outlined in Fig. 13: $K_S = 2$, $K_A = 2$ and $K_H = 1$. The performance index table of [19] is also used.

The application of two SOF-PID controllers to control a two-link revolute-joint robot arm to track a path of two-dimensional square, produces the maximum deviation of about 1.5 cm from the X axis and 1.3 cm from the Y axis. Using the three SOF-PID controllers, the maximum deviation from the required square is about 2.4 cm from the X axis and 2.3 cm from the Y axis, higher than the two SOF-PID controllers. For the two-link robot arm, the time taken from the starting point $\theta_1 = \theta_2 = 0$ to the top right corner of the required square was about 1 second. The time required to draw the two-dimensional square was approximately 4 s, a total of 816 samples. For the three-link robot arm, the time taken from the starting point to the top right corner of the square was about 1 s, and 6 s for the completion of the 30 cm $\times$ 30 cm square, a total of 1140 samples.

In the path tracking experiments, the two SOF-PID controllers trace the required square closer and smoother than
the TITO SOFC and the two PID controllers. The average maximum deviation from the ideal square for each controller is: approximately 1.4 cm for the two SOF-PID controllers, 1.6 cm for the TITO SOFC and 2.5 cm for the two PID controllers, Figs. 9 –11, respectively. The performance of the three SOF-PID controllers in tracing a two-dimensional square, is also better than the THITHO SOFC and the three PID controllers. The average maximum deviations are about 2.4 cm for the three SOF-PID controllers, 2.6 cm for the THITHO SOFC, and 3.2 cm for the three PID controllers, Figs. 12–14, respectively.

V. CONCLUSION

The purpose of conducting step input experiments for the SOF-PID controller were to accomplish a good damping around the setpoint, to eliminate the overshoot as the setpoint getting closer, and to minimize the steady-state error. It is concluded that for the step input experiments, the multi-individual SOF-PID controllers demonstrate an improvement in the process output response. The rise time is faster using the multi-individual SOF-PID controllers as opposed to using the MIMO SOFC and the multi-individual PID controllers. Overshoot is practically nil for the multi-individual SOF-PID
controllers and the MIMO SOFC. The multi-individual PID controllers exhibit some overshoot in the presence of high rates of change such as noise, due to its derivative action. The steady-state error is smaller for the multi-individual SOF-PID controllers in comparison with the MIMO SOFC and the multi-individual PID controllers.

For the path tracking experiments, the multi-individual SOF-PID controllers were able to supply the motors with appropriate signals which were of sufficient magnitude to enable the trajectories to follow the path closer and smoother than the MIMO SOFC and the multi-individual PID controllers, in the presence of noise and time variant dynamics. In general, the SOF-PID controller is superior in tracing a path and adjusting itself to continuous changes both at the setpoint and from the process, in comparison with the SOFC and the PID controller.

For the step input and path tracking trajectories, with careful tuning of the supervisory controller parameters $E_S$, $C_{ES_F}$, $C_{ES_S}$, $SK_p$, $DIR$, $K_1$, $K_2$, and $K_3$, the multi-individual SOF-PID controllers produce a smoother process output than the MIMO SOFC and the multi-individual PID controllers in controlling a nonlinear revolute-joint robot arm. However, further research may be required to reduce the number of tuning parameters. At the supervisory level, the min implication function may be an expression which helped to improve this paper.

Finally, a linguistic self organizing process controller was introduced here, designed to be used at the supervisory level and improves the performance of the process output significantly. Finally, an introduction of noise to the system for the multi-individual SOF-PID controllers, produces less disturbance in the process output response when it is compared with the MIMO SOFC and the multi-individual PID controllers.

**ACKNOWLEDGMENT**

The author would like to thank all referees for their valuable suggestions which helped to improve this paper.

**REFERENCES**

Hassan B. Kazemian (M’88–SM’99) received the B.Sc. degree in engineering, the M.Sc. degree in control systems engineering, and the Ph.D. degree in learning fuzzy controllers, from <AUTHOR: PLEASE DETAIL WHICH UNIVERSITIES> Universities in England, U.K., in 1985, 1987, and 1998, respectively.

He is currently a Senior Lecturer at London Guildhall University, London, U.K. He was with Ravensbourne College University, U.K. as a Senior Lecturer for eight years. His previous lecturing experience was at the University of East London, U.K., University College, Northampton, U.K., Newham College, U.K., and secondment to Ford Motor Company, U.K. <PLEASE PROVIDE CITIES FOR ALL PAST AFFILIATIONS> His research interests include fuzzy control, networks, Bluetooth, digital video, and robotics.

Dr. Kazemian is a member of the Institution of Electrical Engineers, U.K., and a Chartered Engineer in the U.K.