Design of a Knowledge-Based Controller for Intelligent Control Systems

Kang G. Shin, *Senior Member, IEEE,* and Xianzhong Cui

Abstract—**A hierarchical knowledge-based controller is proposed to improve the performance of complex control systems, such as robots. Unlike parameter- and performance- adaptive controllers, this controller is designed only to modify the reference input of a lower-level servo controller. Because the internal parameters and structure of the lower-level controller are not affected, commercial servo controllers can be made to perform more sophisticated tasks than originally intended. The principle of the knowledge-based controller, modification of the reference input, knowledge representation, existence of the solution, and analyses of the controller's stability and tracking error are described in detail. A self-tuning multiple-step predictor is designed as part of the controller to eliminate the undesirable effects of system time delay. Both linear and nonlinear example control systems are tested via extensive simulations and have all shown promising performances.**

I. Introduction

Conventional control theory is developed based on mathematical models that describe the dynamic behavior of controlled systems. Usually, such a model consists of a set of linear or nonlinear differential/difference equations, most of which are derived under some forms of approximation and simplification. However, the complexity and model and/or parameter uncertainties of the controlled systems often make the controllers very complicated. On the other hand, human operators do not always handle the system control problem with a detailed mathematical model, but rather with a qualitative or symbolic description of the controlled system. This fact calls for the need of intelligent control (IC) for complex systems. An extensive survey of IC can be found in [1]. Most related IC work can be referred to as parameter-adaptive [2, 3], or performance-adaptive [4], [5]. The fundamental difference between these two lies in the goal of the knowledge base designed.

A typical structure of parameter-adaptive IC is presented in [2], [3], where IC is used to tune the parameters of a conventional controller as sketched in Fig. 1(a). In [2], a programmable logic controller (TI 565 PLC) performs the functions of a PID controller. Six different features are monitored, such as overshoot, rise time, settling time, etc. Two sets of production rules are established on the basis of the measurement of these features. One is called the *alarm rules* which will be fired when system instability is identified, and the other rules will be fired to re-tune the PID parameters. Another expert tuner for a PI controller is presented in [3]. Its main objectives are wind-up protection of the I part and tuning of PI gains. The transient response of a closed-loop system is characterized with nine categories including too_low_monotone, too_low_oscillatory, and so on. Moreover, the open-loop response is described with eight categories and

Manuscript received March 16, 1990; revised July 28, 1990. This work was supported in part by the National Science Foundation under Grant No. DMC-8721492.

The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122.

IEEE Log Number 9039996.

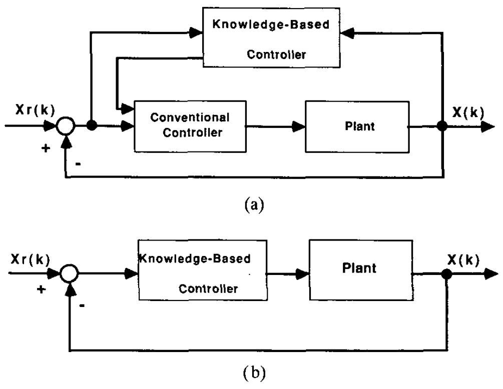


Fig. 1. Basic structures of intelligent controllers. (a) Parameter-adaptive intelligent controller. (b) Basic structures of intelligent controllers.

the nonlinearity with eight categories. Similarly to [2], system stability is monitored by the knowledge base itself.

The performance-adaptive IC's attempt to simulate human expert control or human cognitive ability directly. Their basic structure is shown in Fig. 1(b). An example of this structure, called a *cognitive controller,* is presented in [4]. Two types of rules are designed: *training rules* and *machine rules*. The training rules are represented by a set of production rules that directly map the position and velocity errors of the control object to the linear motion of the control mechanism, a joy stick. The machine rules are dynamic, learned from the accumulated experience of control, and implement the machine learning process. Using performance-adaptive IC, the principle of learning control can also be implemented [5]. In control theory, a learning controller is a trial-and-error mechanism that repeats a fixed procedure. The work in [5] replaced this fixed procedure by a knowledge-based controller such that the intermediate output errors are treated as knowledge. The error and error increment are quantified into eleven intervals, such as negative_big, positive_low_middle, etc. Based on this quantification, an 11×11 table is formed corresponding to 121 rules. Considering the global control performance, a set of rules is designed to determine the magnitude of reference input modification for different learning periods.

Though many IC schemes have been proposed, it is difficult to compare them, because of lack of theory and a universal performance criterion. For a successful design of IC, the following issues seem to be important:

1) What characteristics are used to express the performance of a system?
2) How to extract qualitative knowledge from quantitative sensor data?
3) How to quantify the result of qualitative reasoning into the

quantitative control signal of actuators?

4) How to analyze and guarantee the system stability?
5) How to implement the rules of learning from experience?

In contrast to the parameter-adaptive and performance-adaptive ICs, we propose a new hierarchical knowledge-based controller. In Section II, the basic principles of this controller are described. The characteristics of the lower-level subsystem and the modification of reference input are also presented in this section. A detailed description of the knowledge-based controller is given in Section III, including knowledge representation, existence of the solution, and inference process. The stability of the knowledge-based controller is analyzed in Section IV. In Section V, the procedure for designing a predictor is briefly discussed, and a detailed error analysis gives the lower and upper bounds of the trajectory tracking error when this controller is introduced. Finally, the simulation results in Section VI show the promising performances of the knowledge-based controller. The paper concludes with Section VII.

II. Design Principles and Characteristics

A. Basic Principles

A control system is evaluated by examining its response to some typical, preplanned trajectories, such as step, slope, parabola and/or sinusoidal signals. There are two ways to improve the performance of the control system. One is to set the desired trajectory as the system reference input, and redesign the internal structure of the servo controller so as to track the reference input precisely. For a complex control system, if this approach is used, the servo control level will become more complicated, and the fine tuning of the controller parameters will be extremely tedious (particularly for nonadaptive schemes). Moreover, there are some design trade-offs to consider, such as the one between rise time and maximum overshoot. The other way is to choose and adjust a reference input such that the controlled system tracks the preplanned trajectory. This forms a hierarchical structure, but requires little change in the internal structure of the servo control level. This is exactly what a hierarchical system is supposed to be; each level in the hierarchy is independent and does not affect the internal structures of other levels.

In the high-level controller's view, the lower-level subsystem is nothing but a mapping L_0 from the reference input, X_r, to the system output, X. There are two ways to improve the performance of the subsystem. One is to modify the map itself, e.g., some parameters or even the structure of the controller. This requires the high-level controller to know the detailed internal structure of the lower level. The other way is to modify the domain of the map only, (i.e., the reference input of the lower level) without requiring any detailed knowledge of the subsystem's structure. Considering the generality, and the inexactness of the structure of the lower-level subsystem, we have adopted the latter in this paper. This adoption also coincides with the principle of increasing intelligence with decreasing precision as we move up the levels of hierarchy.

It is assumed that the servo controller is designed independently of the high-level controller, and its dynamic structure and parameters are unknown to the high-level controller. In this paper, we shall design a knowledge-based controller (as a high-level controller) which modifies only the reference input to the subsystem as shown in Fig. 2. As a result, the internal structure and/or parameters of the (lower-level) servo controller are not required to be altered at all, thus imposing no constraints on the servo control level. This will, in turn, enable commercially designed servo controllers to perform more sophisticated tasks than originally intended.

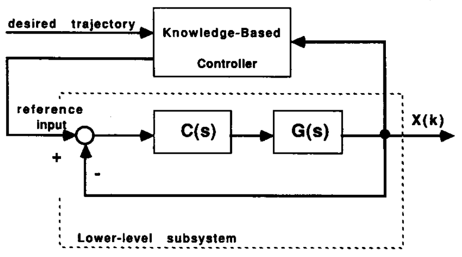


Fig. 2. Structure of the hierarchical knowledge-based controller.

B. Characteristics of the Subsystem and the Modification Process

To design a knowledge-based controller, one has to specify the input space of the knowledge base, or choose a typical representation of the system dynamic characteristics. The most commonly used components are output error and/or error increment, and the standard figures of step response. However, to express the system characteristics more directly and to eliminate the undesirable effects of time delay, we propose to use predicted system outputs from which suitable reference inputs are determined. Because the lower level of hierarchy is a well-designed closed-loop control system, designing an output predictor for such a system is not difficult.

Now, the problem is how to modify the reference input so as to make the system output track a desired trajectory. Using system output prediction, the desired performance can be achieved by iterative trial as is done in learning control. Note that learning control is usually used for a repetitive trajectory and needs a learning period during which an unacceptable output error could occur. By contrast, our knowledge-based controller is designed for an arbitrary trajectory and has to complete the iterative learning process in each sampling interval. The modification process is to 1) give a reference input, 2) compute a predicted system output, 3) calculate the predicted tracking error by comparing the prediction and the desired trajectory, and 4) modify the reference input based on this error. Note that actions taken in a control system are in general irrecoverable; that is, each reference input to the servo controller is the final decision at each sampling interval and cannot be undone. However, the combined prediction and modification allows us to analyze the "anticipated" consequence of each reference input, thereby at least partially solving the irrecoverable problem.

This modification process can be formalized as follows. Let $X_r^i(k)$ be the reference input, $X_d^*(k)$ the desired trajectory, and $\hat{X}^i(k+d/k)$ the d-step ahead prediction of the system output at time k, where the superscript i denotes the ith iteration. The reference input $X_r^i(k)$ is modified by

$$X_r^{i+1}(k) = X_r^i(k) + K_0^i(k)e^i(k+d), \qquad i = 0, 1, \cdots$$

where $e^i(k+d) = \hat{X}^i(k+d/k) - X_d^*(k+d)$, $X_r^0(k) = X_d^*(k)$, $K_0^i(k)$ is the learning gain at time k during ith iteration, and $K_0^0(k) = 0$. Then we get

$$X_r^{i+1}(k) = X_d^*(k) + \sum_{j=0}^{i} K_0^j(k)e^j(k+d).$$

Accurate tracking will be achieved by the iterative operation and the predictor, and this iterative operation must be completed in each sampling interval. To make the above modification feasible, the following conditions need to be met: 1) the iterative operation converges fast, and 2) the output prediction of the system is computable. Condition 1) is usually met because the lower level is a carefully designed control system, and $X_r^i(k)$ is near the optimal point. Condition 2) will be discussed in the following sections.

The lower-level subsystem is equipped with some well-designed servo controllers whose behavior is assumed to be linear. Then, following an argument similar to the one in [6], one can prove that there exists a learning gain $K_0^i(k)$ such that $e^i(k+d) \to 0$ as $i \to \infty$. Though such a $K_0^i(k)$ exists, due to lack of knowledge of the lower-level subsystem, it is not easy to calculate the gain accurately. Moreover, the parameters and/or model uncertainties are not even considered, thus necessitating design of a knowledge-based controller.

III. Description of the Knowledge-Based Controller

A. Knowledge Representation

Using the predictor, the subsystem performance is characterized by the predicted tracking error and the current reference input. Therefore, the space $\boldsymbol{E}$ of predicted tracking error forms the input space of the knowledge base. The goal of the knowledge-based controller is to implement the modification process discussed thus far. It is not difficult to express this process by a set of production rules. The possible actions that the knowledge-based controller can take include: increase the reference input, decrease the reference input, and keep the reference input unchanged. The problem is how much to increase/decrease and how to determine the bounds of the reference input. Because this scheme is based on the modification of reference input and the resulting predicted output, the internal structure and parameters of the lower-level subsystem are not affected. This property allows us to consider the predicted tracking error, but not its derivative, as the system characteristics or the input of the knowledge base, so as to simplify the design of production rules. The basic modification process can be represented by a decision tree as shown in Fig. 3. The ijth node is represented by $([a_j^i, b_j^i], c_j^i)$ where c_j^i is the quantity added to the reference input,

$$X_r^{i+1}(r) = X_d^*(k) + c_j^i,$$

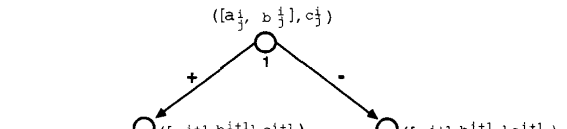
and $[a_j^i, b_j^i]$ is the interval to be searched, and $a_j^i < c_j^i < b_j^i$ for all i,j. By giving the reference input $X_r^i(k)$, at any node $([a_j^i, b_j^i], c_j^i)$, the interval $[a_j^i, b_j^i]$ will be split into two subintervals $[a_k^{i+1}, b_k^{i+1}] \equiv [a_j^i, c_j^i]$ and $[a_{k+1}^{i+1}, b_{k+1}^{i+1}] \equiv [c_j^i, b_j^i]$, which form the two successor nodes of $[a_j^i, b_j^i]$. During the ith iteration and at ijth node, let $e_j^i(k)$ denote the predicted tracking error resulting from $X_r^i(k)$:

$$e_j^i(k) \equiv e^i(k+d) = \hat{X}^i(k+d/k) - X_d^*(k+d).$$

Then, c_j^i is computed as

$$c_j^i = \begin{cases} b_j^i - (b_j^i - a_j^i)K, & \text{if } e_j^i(k) < 0 \\ a_j^i + (b_j^i - a_j^i)K, & \text{if } e_j^i(k) > 0 \end{cases}$$

and $0 < K < 1$ is a weighting factor that determines the step size of the iterative operation. a_0^0 and b_0^0 are the predesigned lower and upper bounds of the reference input modification, and usually $c_0^0 = 0$, i.e., at the beginning, the reference input is not modified.



$([a_j^i, b_j^i], c_j^i)$
1
+
-
2
$([a_k^{i+1}, b_k^{i+1}], c_k^{i+1})$
3
$([a_{k+1}^{i+1}, b_{k+1}^{i+1}], c_{k+1}^{i+1})$
+
-
+
-
$([a_{l+1}^{i+2}, b_{l+1}^{i+2}], c_{l+1}^{i+2})$
$([a_{l+3}^{i+2}, b_{l+3}^{i+2}], c_{l+3}^{i+2})$
$([a_l^{i+2}, b_l^{i+2}], c_l^{i+2})$
$([a_{l+2}^{i+2}, b_{l+2}^{i+2}], c_{l+2}^{i+2})$
• • •
• • •

Fig. 3. Decision tree.

B. Existence of the Solution

The basic forms of production rules are

IF $\{(e_j^i(k) < 0)$ AND $(|e_j^i(k)| > \varepsilon)\}$ THEN
$\{$increase c_j^i AND compute $X_r^{i+1}(k) = X_d^*(k) + c_j^i\}$.
IF $\{(e_j^i(k) > 0)$ AND $(|e_j^i(k)| > \varepsilon)\}$ THEN
$\{$decrease c_j^i AND compute $X_r^{i+1}(k) = X_d^*(k) + c_j^i\}$.
IF $\{|e_j^i(k)| \le \varepsilon\}$ THEN
$\{$set $X_r^{i+1}(k) = X_r^i(k)$ AND stop the iterative operation$\}$.

$\varepsilon > 0$ is the prespecified error tolerance. Because the amount of modification to the reference input is bounded, or $a_0^0 < c_j^i < b_0^0$, for all i,j, there may be a case that $|e_j^i(k)| > \varepsilon$ for all c_j^i To avoid this situation, the desired trajectory needs to be carefully designed. For example, when the desired trajectory is a step function and the system time delay is equal to two sampling intervals, at $k = 0$ the continuous system response cannot have a jump no matter how large the reference input is. A reasonable choice of ε is another way to prevent this problem. This existence problem can be monitored by adding, for example, the following rule into the knowledge base:

IF $\{((|c_j^i - b_0^0| < \delta)$ OR $(|c_j^i - a_0^0| < \delta))$ AND
$(|e_j^i(k)| > \varepsilon)\}$ THEN
$\{($change a_0^0 or b_0^0 automatically and continue the search$)$
OR (ask the operator for an adjustment) OR
(stop the iterative operation and choose c_j^i with the
smallest $e_j^i(k)$ as the best output)$\}$.

Suppose the weighting factor K is set too small or too large, then the search for c_j^i may take a very long time. This would not be acceptable if the required computation cannot be completed within one sampling interval. The case of the computation/search time exceeding one sampling interval is equivalent to having no solution. This case is monitored by:

IF $\{($the search time $> N)$ AND $(|e_j^i(k)| > \varepsilon)\}$ THEN
$\{($stop the iterative operation$)$ AND
$($choose c_j^i with the smallest $e_j^i(k)$ as the best output$)$
AND $($modify the weighting coefficient $K)\}$.

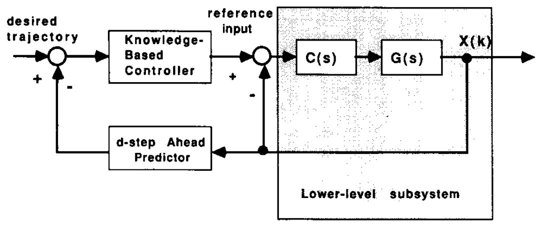


Fig. 4. Block diagram of the knowledge-based controller.

C. Inference Process

Based on the structure of the decision tree, one can see that the simplest inference process is similar to forward chaining, starting from the root node. However, it may be learned after a period of operation that, for example, a positive augment c_j^i is always needed. Once this fact is learned, the inference process can start from any node with $c_j^i > 0$ and go forward or backward, depending on the sign of predicted tracking error. Note that the backward search does not mean a reverse search, but rather intends to find a suitable node from which a forward search can begin. As soon as the forward search begins, the search process is not reversible.

IV. Stability Issues

It is easy to establish the stability of the lower-level subsystem for a fixed reference input, because it is a well-designed closed-loop control system. However, this does not imply the stability of the whole system. See Fig. 4 for a block diagram of the knowledge-based controller. Both system poles and zeros are affected by the presence of the knowledge-based controller and the predictor. If the transfer functions for all the blocks in Fig. 4 are given, one may be able to derive the conditions for system stability. But this is not the case in reality: $C(s)$ and $G(s)$ may not be known accurately, and the iterative learning with prediction and the knowledge-based controller do not form a simple feedback loop and cannot be expressed as simple mathematical transfer functions.

Suppose the prediction gives the true system output and let us consider the knowledge-based controller and the closed-loop subsystem. (The assumption of perfect output prediction is of course unrealistic and will be relaxed in our later discussions.) The knowledge-based controller can be viewed as a map $\boldsymbol{M}_0 : \boldsymbol{E} \rightarrow \boldsymbol{X}_R$, specified by all the production rules, where $\boldsymbol{E} \subset \boldsymbol{R}$ is the space of tracking error and $\boldsymbol{X}_R \subset \boldsymbol{R}$ the reference input space. The lower-level closed-loop subsystem is also a map, $\boldsymbol{L} : \boldsymbol{X}_R \rightarrow \boldsymbol{E}$, which is specified by the desired dynamic properties of the servo controller. Because $\boldsymbol{L}$ represents a well-designed controller and there exists a reference input at time k, $X_r^i(k) \in \boldsymbol{X}_R$, such that the trajectory tracking error $e^i(k+d) = 0$. Thus, it is reasonable to assume that $\boldsymbol{L}$ is a linear map. The properties of the map $\boldsymbol{M} \equiv \boldsymbol{L}\boldsymbol{M}_0 : \boldsymbol{E} \rightarrow \boldsymbol{E}$ depends mainly on the properties of the map $\boldsymbol{M}_0$. In fact, all the antecedents of production rules are established based on the output prediction. If the predictor gives the true output, then the properties of the invariant map $\boldsymbol{M} : \boldsymbol{E} \rightarrow \boldsymbol{E}$ is determined solely by the knowledge base.

For system stability, all production rules in the knowledge base must form a contraction map. More formally, we have the following theorem.

Theorem: Suppose 1) the output prediction of the lower-level subsystem is computable and the predictor gives the true output, and 2) $\boldsymbol{L} : \boldsymbol{X}_R \rightarrow \boldsymbol{E}$ of the lower-level closed-loop subsystem is a linear map. If the map $\boldsymbol{M}_0 : \boldsymbol{E} \rightarrow \boldsymbol{X}_R$ is given by the decision tree, then the composite map $\boldsymbol{M} \equiv \boldsymbol{L}\boldsymbol{M}_0 : \boldsymbol{E} \rightarrow \boldsymbol{E}$ is a contraction map.

Proof: See the Appendix.

At each node in the structure of decision tree, the rules always keep the search direction pointed to the node where the tracking error decreases. Because the iterative learning process is performed at each node, this is equivalent to that the iterative learning process decreases the tracking error. As mentioned in Section III, the inference process is not reversible, and thus, it is impossible to have an unstable system response.

V. Predictor Design and Error Analysis

As mentioned earlier, the lower-level subsystem is equipped with a servo controller which is assumed to have a linear response to the reference input. For such a linear system, there are several algorithms available to design a predictor. For convenience, we have chosen a self-tuning predictor among them. Its principle is briefly stated below (see [7] for a detailed account). The lower-level, closed-loop subsystem is represented by an ARMAX model:

$$\boldsymbol{A}(z^{-1})X(k) = \boldsymbol{B}(z^{-1})X_r(k-d_0) + \boldsymbol{C}(z^{-1})\xi(k) \tag{1}$$

where

$$\boldsymbol{A}(z^{-1}) = 1 + a_1 z^{-1} + \cdots + a_n z^{-n},$$
$$\boldsymbol{B}(z^{-1}) = b_0 + b_1 z^{-1} + \cdots + b_n z^{-n},$$
$$\boldsymbol{C}(z^{-1}) = 1 + c_1 z^{-1} + \cdots + c_n z^{-n}.$$

d_0 is an index representing the time delay, $X(k)$ and $X_r(k)$ are the output and the reference input of the subsystem, respectively. $\xi(k)$ is an uncorrelated random series with zero mean representing the modeling error and disturbances. Define the d-step ahead prediction error as

$$e_p(k+d) = X(k+d) - \hat{X}(k+d/k). \tag{2}$$

Substituting (2) into (1), and representing $\boldsymbol{A}(z^{-1})$, $\boldsymbol{B}(z^{-1})$ and $\boldsymbol{C}(z^{-1})$ as $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ for simplicity, we get

$$\boldsymbol{A}e_p(k) = \boldsymbol{B}X_r(k-d_0) - \boldsymbol{A}\hat{X}(k/k-d) + \boldsymbol{C}\xi(k). \tag{3}$$

Equation (3) can be viewed as a new system, in which the input is the prediction $\hat{X}(k/k-d)$, the output is the prediction error $e_p(k)$, $X_r(k-d_0)$ is the measurable noise and $\xi(k)$ is the unmeasurable noise. Define the cost function as $J = E[e_p^2(k+d)]$ and let

$$\boldsymbol{C} = \boldsymbol{E}_0\boldsymbol{A} + z^{-d_0}\boldsymbol{F},$$

where $\boldsymbol{E}_0$ and $\boldsymbol{F}$ are polynomials of z^{-1}, and $\deg(\boldsymbol{E}_0) = d-1$, $\deg(\boldsymbol{F}) = n-1$. By minimizing J, we get the optimal predictor

$$\hat{X}(k+d/k) = \frac{\boldsymbol{B}z^{-d_0}}{\boldsymbol{A}} X_r(k+d) + \frac{\boldsymbol{F}}{\boldsymbol{E}_0\boldsymbol{A}} e_p(k). \tag{4}$$

This is the prediction during the first iteration, i.e., $\hat{X}^1(k+d/k) \equiv \hat{X}(k+d/k)$. The prediction error corresponding to (4) is

$$e_p(k+d) = \boldsymbol{E}_0\xi(k+d). \tag{5}$$

All the parameters of the system and the predictor are unknown and estimated on-line by a recursive least square (RLS) algorithm, $\hat{X}(k+d/k)$ is then computed using the estimated parameters.

In the knowledge-based controller, for each computed $X_r^i(k)$, the corresponding output prediction $\hat{X}^i(k+d/k)$ should be computed. Note that only $\hat{X}^1(k+d/k)$ is computed by (4), while given $X_r^1(k) = X_d^*(k)$. The subsequent steps within this iteration are computed as

$$\hat{X}^i(k+d/k) = \hat{X}^1(k+d/k)\frac{X_r^i(k)}{X_d^*(k)}K_m^i(k), \quad i = 2, 3, 4, \cdots. \tag{6}$$

This formula is based on the assumption that the lower-level, closed-loop subsystem has a linear response to its reference input. For the case of $i > 1$, using (6) instead of (4) not only simplifies the computation, but also reduces the sensitivity of the iteration to some estimated parameters. In (6), $K_m^i(k)$ is the gain factor. A servo controller is usually designed with a unity gain with respect to its reference input, so it is reasonable to set $K_m^i(k) = 1$. However, $\hat{X}^1(k+d/k)$ is computed with the error (5); in case the prediction error increases after the iterative operation, the gain factor would not necessarily be one for $i > 1$ so as to compensate for the prediction error.

Thus far, we have assumed that the predictor gives the true output, which is not realistic. The effect of prediction error on the tracking error is thus analyzed below. The tracking error is defined as

$$e_t(k) = X(k) - X_d^*(k). \tag{7}$$

As a result of the ith iteration, suppose the actual reference input becomes $X_r^i(k)$ and the d-step ahead prediction of the output is $\hat{X}^i(k+d/k)$. $\hat{X}^1(k+d/k)$ is computed by the self-tuning predictor (4) and $\hat{X}^i(k+d/k)$, $i > 1$, is computed by (6). When

$$|e^i(k+d)| = |\hat{X}^i(k+d/k) - X_d^*(k+d)| \le \varepsilon, \tag{8}$$

the iterative learning process is stopped, and the result is given as the actual reference input $X_r^f(k)$ and the corresponding output prediction $\hat{X}^f(k+d/k)$. Because the iteratively computed prediction error is $e_p^i(k) = X(k) - \hat{X}^i(k/k-d)$, using (7) we get

$$e_t(k) - e_p^i(k) = \hat{X}^i(k/k-d) - X_d^*(k). \tag{9}$$

Equation (8) can be used to convert (9) in the form $|e_t(k) - e_p^f(k)| \le \varepsilon$. Because $|e_t(k)| - |e_p^f(k)| \le |e_t(k) - e_p^f(k)|$ and $|e_t(k) - e_p^f(k)| = |e_p^f(k) - e_t(k)|$, we conclude

$$|e_p^f(k)| - \varepsilon \le |e_t(k)| \le |e_p^f(k)| + \varepsilon. \tag{10}$$

This formula gives the upper and lower bounds of tracking error when the knowledge-based controller is added. Specifically, it shows that the tracking error cannot be much less than the iteratively computed prediction error.

Obviously, the inaccurate prediction may degrade the performance of the knowledge-based controller. It can be seen from (5) that the prediction error of the self-tuning predictor is the moving average of a zero mean, uncorrelated random series of order $d-1$. Based on this observation, the subsequent steps of prediction are iteratively computed by (6). To reduce the tracking error, a sophisticated predictor needs to be designed as a part of the iterative operation. The other way is to change the gain factor $K_m^i(k)$ in (6) so as to compensate for the prediction error.

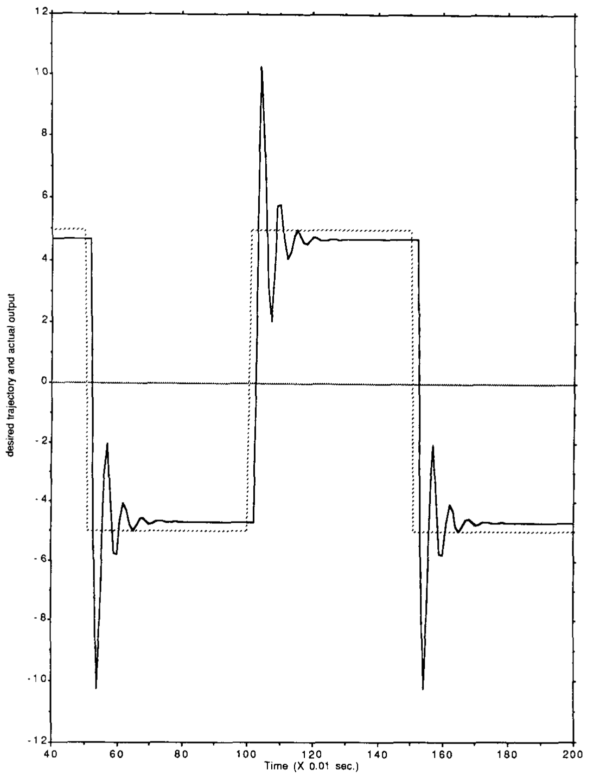


Fig. 5. Output response of a linear system without the knowledge-based controller. Dashed lines dashed lines show desired trajectory; solid lines show actual output.

VI. Simulation Results

Extensive simulations were carried out for two types of systems. First, we tested a linear open-loop system whose model is

$$y(k) = 0.45181\, y(k-1) + 0.47546\, y(k-2) - 0.04560\, u(k-1) - 0.00404\, u(k-2)$$

where y and u are the system output and control input, respectively. Using a proportional controller with $K_p = 20.9$, its closed-loop response is calculated and plotted in Fig. 5. The mean square tracking error is $MS = 4.55485$. By adding the knowledge-based controller, the performance is improved as shown in Fig. 6 with $MS = 1.69398$. The second system we tested is a two-link robot manipulator, a nonlinear open-loop system:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{11}$$

where $\boldsymbol{q}$ and $\boldsymbol{\tau}$ are the vectors of joint position and torque, respectively; $\boldsymbol{H}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ represent the Coriolis and centrifugal forces, and $\boldsymbol{G}(\boldsymbol{q})$ represent the gravitational force. Its configuration and dynamic and kinematic parameters are presented in Fig. 7. A controller is designed with the computed torque algorithm [8]. The proportional and derivative gains are $K_p = 986.96$ and $K_D = 62.83$, which correspond to $\zeta = 1.0$ and $f_n = 5$ Hz, respectively. The sampling frequency used is $f_c = 100$ Hz. For simplicity, the desired trajectory is specified

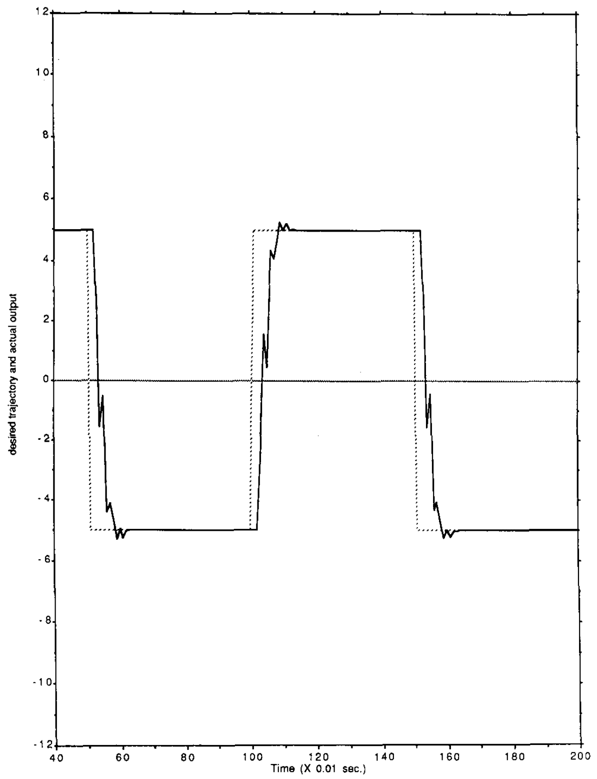
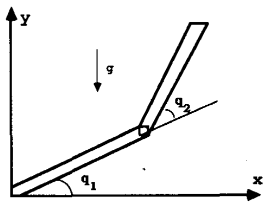


Fig. 6. Output response of a linear system with the knowledge-based controller. Dashed lines dashed lines show desired trajectory; solid lines show actual output.



	link 1	link 2
lerngth L_i	1.0 m	1.0 m
mass center L_{ci}	0.5 m	0.5 m
mass m_i	20 kg	10 kg
moment of intertia I_i	0.8 kgms^2	0.2 kgms^2

Fig. 7. Two-link robot manipulator and its parameters.

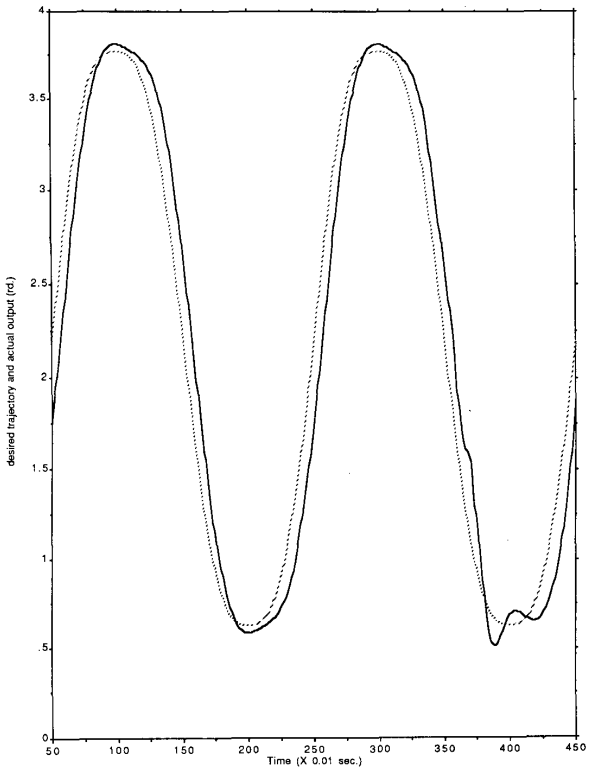


Fig. 8. Output response of the first link using inaccurate inertia matrix without the knowledge-based controller. Dashed lines show desired trajectory; solid lines show actual output.

in joint space. To achieve desired performance, the computed torque algorithm requires the accurate values of each term in (11). If inaccurate values of the inertia matrix are used in the computed torque algorithm, the performance is degraded as shown in Fig. 8 with mean square tracking error $MS = 0.08543$. When the knowledge-based controller is added, the performance is improved as shown in Fig. 9 with $MS = 0.00932$. For comparison, with accurate values of the inertia matrix, the closed-loop response of the first link are plotted in Figs. 10 and 11. The corresponding mean square tracking errors are $MS = 0.05742$ without the knowledge-based controller and $MS = 0.00179$ with it. One can see that the performance is also improved. Moreover, the simulation results of the second link are similar to that of the first link.

VII. Conclusion

A knowledge-based controller is proposed as a new architecture of IC and analyzed in detail. Its basic principle is to modify the reference input of the lower-level subsystem so as to track a predesigned trajectory accurately, and to leave the internal structure and/or parameters of the subsystem unaffected. With the concept of iterative learning control, the knowledge base is simple to design and the stability of the overall system is guaranteed. By using a d-step ahead predictor, the undesirable effect of system time delay is eliminated and each reference input is analyzed in advance. This, in turn, solves the irrecoverable control problem such that commercially designed controllers can be made to perform more sophisticated tasks than originally intended.

The proposed knowledge-based controller is simulated extensively, and the results are quite promising. Our immediate future work is to extend this scheme to the problem of coordinating multiple systems.

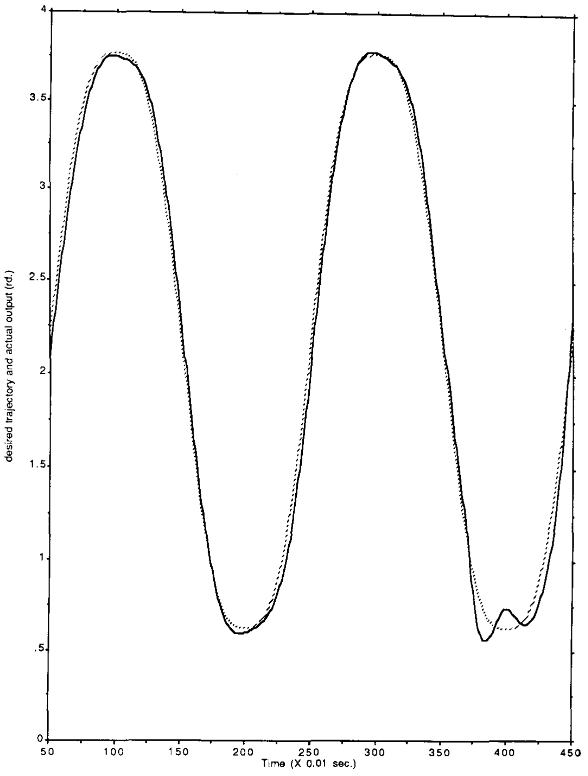


Fig. 9. Output response of the first link using inaccurate inertia matrix with the knowledge-based controller. Dashed lines show desired trajectory; solid lines show actual output.

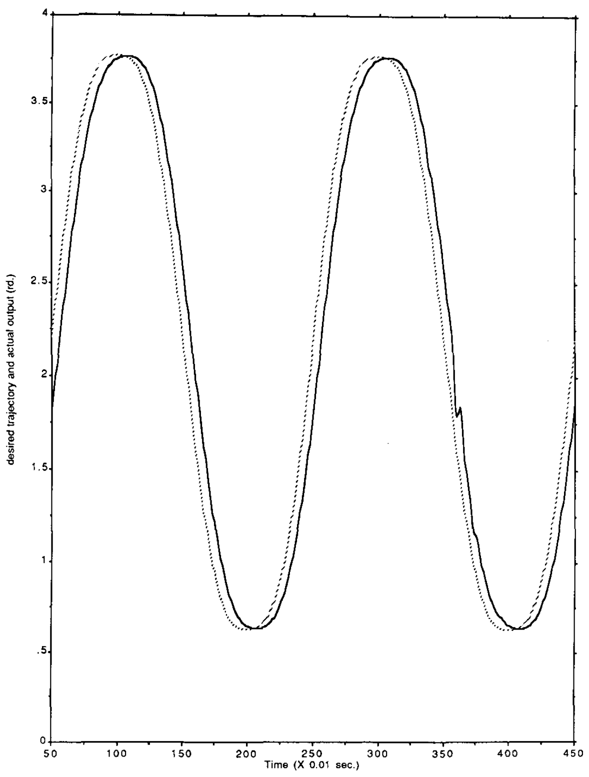


Fig. 10. Output response of the first link using accurate inertia matrix without the knowledge-based controller. Dashed lines show desired trajectory; solid lines show actual output.

Appendix
Proof of Theorem

The basic production rules are in the form of

$$\begin{aligned}
&\text{IF } \{\text{by the reference input } X_r^i(k),\\
&\quad\text{the predicted tracking error } e_j^i(k) < 0\}\\
&\text{THEN } \{\text{increase } c_j^i \text{ to get } c_{k+1}^{i+1}, c_{k+1}^{i+1} > c_j^i\},\\
&\text{IF } \{\text{by the reference input } X_r^i(k),\\
&\quad\text{the predicted tracking error } e_j^i(k) > 0\}\\
&\text{THEN } \{\text{decrease } c_j^i \text{ to get } c_k^{i+1}, c_k^{i+1} < c_j^i\},
\end{aligned}$$

where $e_j^i(k)$ is the predicted tracking error resulting from the augment c_j^i at the jth node of the ith level in the decision tree. These rules associate each predicted tracking error with a specified value of c_j^i. From a new augment c_k^{i+1} or c_{k+1}^{i+1}, the predicted tracking error $e_k^{i+1}(k)$ and $e_{k+1}^{i+1}(k)$ are then computed.

Since the decision tree is searched downward after finding a starting node in the tree, we want to show that this search process has the following property:

$$d(c_l^{i+2}, c_k^{i+1}) < d(c_k^{i+1}, c_j^i). \tag{12}$$

for all $i, j, k, l = 0, 1, 2, \cdots$, and $d(,)$ is a metric on X_R.

Referring to Fig. 3, we get

$$\begin{aligned}
c_k^{i+1} &= a_k^{i+1} + (b_k^{i+1} - a_k^{i+1})K = a_j^i + (c_j^i - a_j^i)K\\
c_{k+1}^{i+1} &= b_{k+1}^{i+1} - (b_{k+1}^{i+1} - a_{k+1}^{i+1})K = b_j^i - (b_j^i - c_j^i)K\\
c_l^{i+2} &= a_l^{i+2} + (b_l^{i+2} - a_l^{i+2})K = a_j^i + (c_k^{i+1} - a_j^i)K\\
c_{l+1}^{i+2} &= b_{l+1}^{i+2} - (b_{l+1}^{i+2} - a_{l+1}^{i+2})K = c_j^i - (c_j^i - c_k^{i+1})K\\
c_{l+2}^{i+2} &= a_{l+2}^{i+2} + (b_{l+2}^{i+2} - a_{l+2}^{i+2})K = c_j^i + (c_{k+1}^{i+1} - c_k^i)K\\
c_{l+3}^{i+2} &= b_{l+3}^{i+2} - (b_{l+3}^{i+2} - a_{l+3}^{i+2})K = b_j^i - (b_j^i - c_{k+1}^{i+1})K.
\end{aligned}$$

The metric defined on X_R is given by

$$d(c_k^{i+1}, c_j^i) \equiv |c_k^{i+1} - c_j^i| = |(a_j^i - c_j^i)(1-K)|$$

$$d(c_{k+1}^{i+1}, c_j^i) \equiv |c_{k+1}^{i+1} - c_j^i| = |(b_j^i - c_j^i)(1-K)|$$

$$d(c_l^{i+2}, c_k^{i+1}) \equiv |c_l^{i+2} - c_k^{i+1}| = |(a_j^i - c_j^i)(1-K)K| \tag{13}$$

$$d(c_{l+1}^{i+2}, c_k^{i+1}) \equiv |c_{l+1}^{i+2} - c_k^{i+1}| = |(c_j^i - a_j^i)(1-K)^2| \tag{14}$$

$$d(c_{l+2}^{i+2}, c_{k+1}^{i+1}) \equiv |c_{l+2}^{i+2} - c_{k+1}^{i+1}| = |(c_j^i - b_j^i)(1-K)^2| \tag{15}$$

$$d(c_{l+3}^{i+2}, c_{k+1}^{i+1}) \equiv |c_{l+3}^{i+2} - c_{k+1}^{i+1}| = |(b_j^i - c_j^i)(1-K)K|. \tag{16}$$

Suppose $0.5 < K < 1$, then at node 2, from (13) and (14) we get $d(c_l^{i+2}, c_k^{i+1}) > d(c_{l+1}^{i+2}, c_k^{i+1})$. Taking the larger of $d(c_l^{i+2}, c_k^{i+1})$ and $d(c_{l+1}^{i+2}, c_k^{i+1})$ and comparing it with $d(c_k^{i+1}, c_j^i)$, we get

$$d(c_l^{i+2}, c_k^{i+1}) < d(c_k^{i+1}, c_j^i), \tag{17}$$

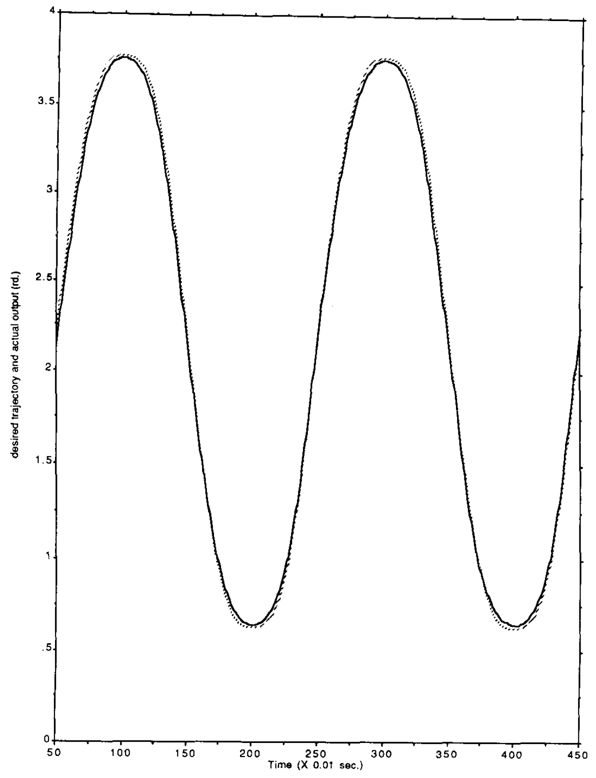


Fig. 11. Output response of the first link using accurate inertia matrix with the knowledge-based controller. Dashed lines show desired trajectory; solid lines show actual output.

because $(1-K)K < (1-K)$. Similarly, at node 3, from (15) and (16) we get $d(c_{l+3}^{i+2}, c_{k+1}^{i+1}) > d(c_{l+2}^{i+2}, c_{k+1}^{i+1})$. Taking the larger of $d(c_{l+3}^{i+2}, c_{k+1}^{i+1})$ and $d(c_{l+2}^{i+2}, c_{k+1}^{i+1})$ and comparing it with $d(c_{k+1}^{i+1}, c_j^i)$, we get

$$d(c_{l+3}^{i+2}, c_{k+1}^{i+1}) < d(c_{k+1}^{i+1}, c_j^i), \tag{18}$$

because $(1-K)K < (1-K)$. Both (17) and (18) show that (12) holds.

For the case of $0 < K \leq 0.5$ (12) can be proved similarly.

Because $L : X_R \to E$ is a linear map, the property of (12) in X_R is preserved under the map L and has the form of

$$d_1(e_l^{i+2}, e_k^{i+1}) < d_1(e_k^{i+1}, e_j^i), \tag{19}$$

for all $i, j, k, l = 0, 1, 2, \cdots$ and $d_1(,)$ is a metric on E. By (19), $M : E \to E$ is a contraction map, and the theorem is proved.

References

[1] J.R. James and G.J. Suski, "A survey of some implementations of knowledge-based systems for real-time control," in *Proc. IEEE Int. Conf. Decision Contr.,* 1988, pp. 580–585.

[2] K.L. Anderson, G.L. Blankenship, and L.G. Lebow, "A rule-based adaptive PID controller," in *Proc. IEEE Int. Conf. Decision Contr.,* 1988, pp. 564–569.

[3] B. Porter, A.H. Jones and C.B. McKeown, "Real-time expert controller for plants with actuator non-linearities," in *Proc. IEEE The 2nd Int. Symp. Intelligent Contr.,* 1987, pp. 171–177.

[4] S. Lee and M.H. Kim, "Cognitive control of dynamic systems," in *Proc. IEEE The 2nd Int. Symp. Intelligent Contr.,* 1987, pp. 455–460.

[5] Z. Geng and M. Jamshidi, "Expert self-learning controller for robot manipulator," *Proc. IEEE Int. Conf. on Decision Contr.,* 1988, pp. 1090–1095.

[6] M. Togai and O. Yamano, "Learning control and its optimality: Analysis and its application to controlling industrial robots," *Proc. IEEE Int. Conf. on Robotics Automat.,* Apr. 1986, vol. 1, pp. 248–253.

[7] R.M.C.D. Keyser and A.R.V. Cauwenberghe, "A self-tuning multistep predictor application," *Automatica,* vol. 17, no. 1, pp. 167–174, 1981.

[8] H. Asada and J.J. Slotine, *Robot Analysis and Control.* New York: Wiley, 1986.

Kang G. Shin (S'75–M'78–SM'83) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea in 1970, and both the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY in 1976 and 1978, respectively.

He is Professor of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan, which he joined in 1982. From 1978 to 1982 he was on the faculty of Rensselaer Polytechnic Institute, Troy, NY. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley, and International Computer Science Institute, Berkeley, CA.

Dr. Shin has been very active and authored/coauthored more than 180 technical papers in the areas of fault-tolerant computing, distributed real-time computing, computer architecture, and robotics and automation. In 1987, he received the IEEE TRANSACTIONS ON AUTOMATIC CONTROL Outstanding Paper Award for a paper on robot trajectory planning. In 1989, he also received the Research Excellence Award from The University of Michigan. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer, called HARTS, to validate various architectures and analytic results in the area of distributed real-time computing. He was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS and the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on Real-Time Systems. He is currently a Distinguished Visitor of the Computer Society of the IEEE and an Area Editor of *International Journal of Time-Critical Computing Systems,* and chairs the IEEE Technical Committee on Real-Time Systems.

Xian-Zhong Cui graduated from North China Institute of Electric Power, Baoding, China, in 1976. He received the first M.S. degree in power plant automation from Electric Power Research Institute (EPRI), Beijing, China, in 1982, and the second M.S. degree in electrical engineering-system from the University of Michigan, Ann Arbor, MI, in 1989. Since 1987, he has been working toward the Ph.D. degree and as a research assistant in the Department of Electrical Engineering and Computer Science, the University of Michigan.

From 1976 to 1980, he served Tainjin Power Plant Construction Company as a technician of instrument and control. From 1982 to 1986, he worked for EPRI as an electrical engineer for power plant automation. He spent 1986–1987 as a visiting researcher in the University of Michigan. His research interests include intelligent control, neural networks, robot control systems, and process control of power plants and electric power systems.