A Novel Multi-Agent Controller for Dynamic Systems based on Supervisory Loop Approach

Sukumar Kamalasadan, *Member, IAENG*

Abstract— **This paper presents a new intelligent agent supervisory loop based approach for dynamic system control. The scheme consists of three software agents that work in an autonomous manner for the precision control of a system that shows multiple modes and drastic parametric 'jumps'. The first two agents are based on artificial intelligent techniques (fuzzy systems and Radial Basis Function Neural Network (RBFNN)) that monitor a closed loop system using a supervisory loop concept. The third agent acts as a traditional Model Reference Adaptive Controller (MRAC) which controls the system dynamics and helps to keep the error inbound. The scheme efficiently alleviates the burden on soft computing techniques due to the strength of the MRAC and at the same time intelligently controls system when nonlinear and functional changes occur. Theoretical analysis and implementation results show that this approach has immense potential in modular agent based controller design.**

Index Terms— **Multi-Agent Controller, Supervisory Loop Principle, Dynamic System Control.**

I. Introduction

Controlling multimodal and 'jump' systems are extremely difficult due to drastic change in the plant dynamics. Several research works have their focus on controlling such systems and multiple model adaptive control has become one of the effective solutions [1], [2]. Research in MMAC to multi modal systems branches into two main directions. One deals with mathematical MMAC of deterministic and stochastic systems and the second one addresses newly emerging heuristic based MMAC using fuzzy systems and neural networks.

Fuzzy logic [3] based adaptive controller is shown to be effective for controlling dynamic systems [4]. A multiple modal method attempted in [5] developed a fuzzy model based adaptive control algorithm for a class of continuous-time nonlinear dynamic system that combines a set of linear fuzzy local model corresponding to the system operating points. Recently an important consideration of using fuzzy systems on an MMAC context is shown in [6]. In this, a set of models representing a nonlinear system during faults and in a fault free regime has been developed using a fuzzified system identifier.

Neural network based controllers are also shown to be really useful and efficient in nonlinear adaptive control. One such controller using neural networks and multiple models is proposed in [7]. A direct adaptive output feedback control design procedure for highly uncertain nonlinear systems that does not rely on state estimation is suggested in [8] and in [9] a multilayer perceptrons with long and short-term memories (LASTMs) for adaptive processing is discussed. Reference [10] studies the use of RBF neural networks as key components of adaptive controllers aimed at controlling flexible robotic arms.

It has been widely recognized that these soft computing topologies plays an important role in modern control techniques. However, implementation and issues related to tuning, controlling while switching and computational complexities are the critical factors in such control. Importance and design of intelligent agents in controlling sophisticated systems has been discussed in several works [11]-[13]. These techniques not only allow for modular design but will make the schemes suitable for practical implementation.

In this paper a novel concept of three intelligent agents to control complex and sophisticated systems is proposed. The first agent is a heuristic based multiple fuzzy reference model generator which moves the reference model, mapping the system auxiliary state when it shows multi modality. This agent is autonomous in nature as it works within its own domain taking auxiliary inputs and generating suitable reference model structure at every time instant. Compared to mathematical MMAC and controller reconfigurable systems [10], [14] - [16], this switching process doesn't show computational intensity. The second agent is a RBFNN based neural network controller that is used to augment the traditional MRAC in the presence of system functional uncertainty. Main consideration here is to use neural network to approximate inverse dynamics of the plant working in parallel with a linear adaptive control law. The advantage of such a feature when compared to a feedback error learning scheme is its ability for coordinated control, which provides a stable law during parameter and functional uncertainties. The third agent is the traditional MRAC which adaptively control the system, linearizes the parameters over a specific domain and forces the output or other plant variables to a suitable reference model structure. For validating the scheme, the controller is used for position tracking of a single link flexible manipulator at various conditions and the simulations are compared with dual agent schemes.

The rest of the paper is organized as follows. In section 2, the concept and mathematical preliminaries are addressed. Section 3 illustrates an application based on the proposed technique for

Manuscript received May 31 2006. This work was supported in part by the internal summer grant from the University of West Florida (UWF) and conducted at the Intelligent Systems and Control lab at UWF.

Sukumar Kamalasadan is with the Department of Engineering and Computer Technology, The University of West Florida, Pensacola, FL 3251 USA (phone: 850-857-6451; fax: 850-474-2804; e-mail: skamalasadan@ uwf.edu).

the position control of a robotic manipulator and section 4 discusses the simulation results. Section 5 concludes the paper.

II. Mathematical preliminaries

Fig. 1 shows a functional block diagram of the overall scheme. It consists of three agent based algorithms which are used to control systems showing multi-modality and sudden 'Jumps" and at the same time susceptible to un-modeled dynamics. In the presence of un-modeled dynamics and modal swings, the parallel controller with RBFNN augments the stable direct model reference adaptive controller and controls the system effectively. However when the system mode switches and the plant shows sudden 'Jumps' [17], the reference model structure needs to be changed. If these modal changes are known, a priori fuzzy inference engine can generate a multiple reference model structure.

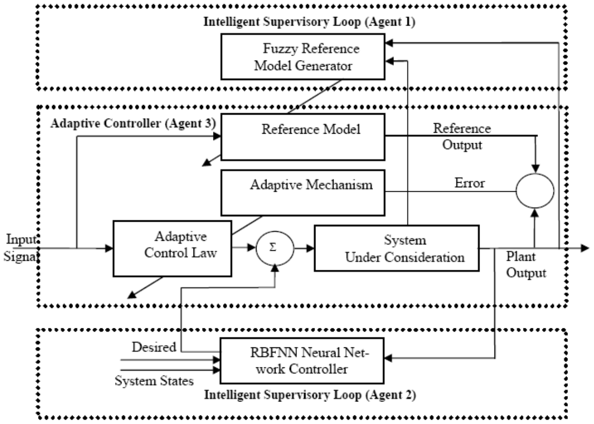
Intelligent Supervisory Loop (Agent 1)
Fuzzy Reference Model Generator
Adaptive Controller (Agent 3)
Reference Model
Reference Output
Adaptive Mechanism
Error
Input Signal
Adaptive Control Law
Σ
System Under Consideration
Plant Output
Desired
System States
RBFNN Neural Net-work Controller
Intelligent Supervisory Loop (Agent 2)

Fig.1. Functional Block Diagram with Intelligent Loops.

There are two intelligent agents based supervisory loops; the first one generates a changing reference model structure, and the second one augments the stable direct model reference adaptive controller. The third agent is the traditional MRAC algorithm itself. If the operation of a system under consideration can be modeled using one reference model, then the adaptive controller can be utilized for controlling the system to track the desired reference model output.

A. *The Design Concept*

The parallel controller structure is based on the fact that a neural network can be used to learn from the feedback error of the plant controller closed loop. The plant is controlled by an adaptive controller, which tracks the reference model output. This works in an autonomous domain by itself, looks at the output and provides required control action. In the event of output distortion, the RBFNN controller learns the system dynamics and augments the MRAC looking at system movements. This is in fact carried out continuously such that the output error is zero and precision tracking is achieved. When system mode 'jumps', the adaptive controller fails to perform with a single reference model as it is often stressed to its limit with only one reference model. Therefore, there is a need to generate reference model structures which is the process performed by the fuzzy agent.

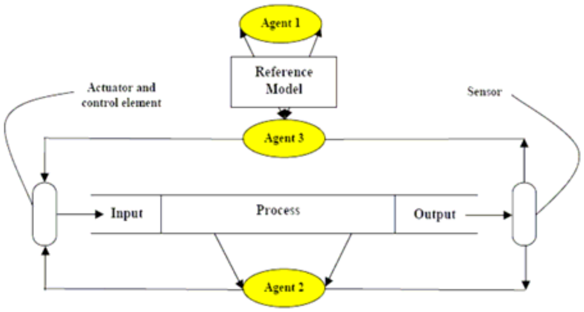

Fig.2. Functional Block Diagram with Intelligent Loops.

Fig. 2 shows the agent interactions and design method for the overall scheme. The first agent is in a hierarchical form indirectly monitoring the performance of the system through agent 3. In the event of drastic changes due to the system parameters sensed by the fuzzy system inputs, the inference engine selects the appropriate rules which in turn change the reference model structure. This enables for a smooth change in the reference model output that allows agent 3 to work effectively. Agent 2 which monitors the system output and auxiliary parameters augments agent 3 when there is a functional change in system output. In this context this architecture is a hybrid approach with hierarchal structure between agents 1 vs agent 3 and heterarchical arrangement agent 2 vs agent 3.

B. *Mathematical Formulation*

The nonlinear system to be controlled with input U and output yi can be represented as

$$\dot{x} = A_{i\,\mathrm{mod}\,e(t)} f(X) + b_{i\,\mathrm{mod}\,e(t)} g(X)U \qquad (1\text{-}2)$$
$$y_i(t) = h(X)$$

Where

$y_i(t)$ is the plant output at a specific mode,

U is the control input ,

X is the state vector where $[X_1(t)\ldots\ldots\ldots X_n(t)]^T$ Є R^n

$A_i(t)=[a_{1i},a_{2i},\ldots\ldots\ldots,a_{ni}]^T$ Є R^{nxn}

f (*), g(*) and h(*) are nonlinear functions and

b_i takes values from the set of H constant elements as indexed by subscripts i Є {1,2,……,H}.

Further, based on the change in values, let f(*) and g(*) be two nonlinear functions such that they are continuous in nature. First, assume that the system is operated at one mode without any mode switching. Under such a condition, (1) and (2) can be written as

$$\dot{x} = Af(X) + bg(X)U \qquad (3\text{-}4)$$
$$y(t) = h(X)$$

Assuming SISO control and a second order system, the above generic equation will be

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = F(x_1, x_2, U) \text{ or} \qquad (5\text{-}7)$$
$$\ddot{x}_1 = F(x_1, \dot{x}_1, U)$$

Assuming the function F is invertible, then

$$U = \overset{\neg}{F}[x_1, \dot{x}_1, \ddot{x}_1] \qquad (8)$$

Let U$_d$ be the desired value of the control, then

$$U_d = \overset{\neg}{F}[x_{1d}, \dot{x}_{1d}, \ddot{x}_{1d}] \tag{9}$$

If x$_1$=y, $U_d = \overset{\neg}{F}[y_d, \dot{y}_d, \ddot{y}_d]$ (10)

This means in the presence of a control U$_d$, the system can be effectively controlled through learning if it is invertible. An RBFNN controller is used for controlling the nonlinear system with unmodeled dynamics, which learns the system dynamics online and then generates the value of U$_d$. As it can be seen from (10), the inputs to RBFNN controller are the output of plant y and other desired states. The output and trajectory error e$_1$ is reduced by a gradient learning technique asymptotically and the system learns during the course of time, generating the value U$_d$.

The error dynamics during system learning can be written as

$$e = F[x_1, \dot{x}_1, U] - F[x_{1d}, \dot{x}_{1d}, U_d] \tag{11}$$

Applying the Taylor series expansion and collecting the first order terms

$$\dot{e} \cong A(t)e + B(t)[U - Ud] + N(t)\varepsilon \tag{12}$$

Where

N(t) is the network variable and ε is the network approximation error.

$$A(t) = \frac{\partial F(x,U)}{\partial x^T} \mid X_d, U_d$$

$$B(t) = \frac{\partial F(x,U)}{\partial U^T} \mid X_d, U_d \tag{13-14}$$

When the RBFNN controller is able to learn the system characteristics successfully, then U=U$_d$ and the error dynamics will be

$$\dot{e} \cong A(t)e + N(t)\varepsilon \tag{15}$$

Considering ε is zero (perfect learning)

$$\dot{e} \cong A(t)e \tag{16}$$

It can be seen from (16) that the error derivative is dependent on A(t) which is a function of the states and control value. An adaptive control law is used to reduce this error asymptotically to zero.

The total control law is

$$U = U_{mr} + U_{nn} \tag{17}$$

Where

$$U_{mr} = \theta^T * \omega \tag{18}$$

$$\theta = \begin{bmatrix} k & \theta_0 & \theta_1 & \theta_2 \end{bmatrix}, \quad \omega = \begin{bmatrix} r \\ y_p \\ \omega_1 \\ \omega_2 \end{bmatrix}$$

$$U_{nn} = \sum_{j=1}^{node} (\exp-(d_j{}^2 / r_j{}^2)) * w_j \tag{19}$$

$d_j = \sqrt{\sum_{k=1}^{Nm} (X_k - C_{(j,k)})^2}$ is the distance and r$_j$, W$_j$ are radius and weights of each node respectively.

The neural network weights are adjusted in order to reduce the gradient of the output error so that nonlinear dynamics are kept bounded. The model reference adaptive controller identifies the plant parametric value online and effectively controls even in case of parameter drift. Moreover, the MRAC will stabilize the output trajectory with the desired value especially at initial stages, which allows the NN to learn the plant dynamics online. Added to all that, static gain designs (such as in linear controllers) are not needed if MRAC is used as a base line controller which is of great advantage especially when there are changing operating conditions.

In the presence of mode switching, the parameter vector can be represented by the triple {(A$_i$,b$_i$,c$_i$),…,(A$_H$,b$_H$,c$_H$)}, which changes its values depending on the modes of operation. Let the above set denote scheduled jump parameters for each mode specified by the parameter index i. The mode variable mode (t) takes the form mapped into any of the values in the domain i Є {1,2,…,H} and correspondingly A$_{mode(t)}$ and b$_{mode(t)}$ are time varying. The mapping of mode (t) is denoted by mode (t) = γ[X (t-d)] where d represents the time delay. The above-mentioned system is a spatial multimodal type because the dynamics are scheduled through a nonlinear mapping γ. As the system mode changes, system parameters 'jump' or move drastically, which in turn may cause unstable performance if a conventional adaptive controller is used. A structural change in the reference model is required if there is no explicit identification procedure available for the plant. This is accomplished through the fuzzy logic multiple reference model generation as described in intelligent agent #1, which in turn allows the direct model reference control law to operate effectively under different plant operating conditions. The design concepts of these agents are as follows.

Intelligent Agent #1: The theoretical basis of the fuzzy multiple reference model generator is as follows. Consider a fuzzy system output denoted by a function $f(\Omega)$ and represented as

$$f(\Omega) = \frac{\sum_{i=1}^{r} r_i \mu_i}{\sum_{i=1}^{r} \mu_i} \tag{20}$$

The above mentioned fuzzy system has r rules and μ_i is the membership function of the antecedent of ith rule given by the input Ω.

Where

$\Omega \in \Re^m$ is a vector containing the relevant auxiliary states

Assume that $\sum_{i=1}^{r} \mu_i \neq 0$ for all relevant auxiliary states and the parameter r$_i$ is the consequent of ith rule. Then the above-mentioned function can be written in the parameterized form as follows

$$f(\Omega) = \frac{\sum_{i=1}^{r} r_i \mu_i}{\sum_{i=1}^{r} \mu_i} = M^T P \vartheta = \Phi * \vartheta \tag{21}$$

Where

$$M = \begin{bmatrix} 1/\chi_1(\Omega) \\ 1/\chi_2(\Omega) \\ \vdots \\ 1/\chi_{m-1}(\Omega) \end{bmatrix} \quad \Omega \in \Re^m \quad , \quad P^T = \begin{bmatrix} p_{1,0} & \cdots & p_{1,m-1} \\ \vdots & \vdots & \vdots \\ p_{r,0} & \cdots & p_{m-1} \end{bmatrix}$$

$$\vartheta^T = \frac{\begin{bmatrix} \mu_1 & \ldots & \mu_r \end{bmatrix}}{\sum_{i=1}^{r} \mu_i} \text{ and } \Phi = M^T P$$

Thus the function approximation by fuzzy scheme is equal to the product of a parameter vector Φ and the weight matrix ϑ. Considering the reference model in state space as:

$$\dot{X}_m(t) = Am_{m(t)} X_m(t) + bm_{m(t)} S(t)$$

$$y_{mi}(t) = C^T X_m(t) \tag{22}$$

Where

$y_{mi}(t)$ is the reference model output and

S(t) is the command signal.

The transfer function equivalent will then be

$$W_{m_i}(t) = y_{m_i}(t)/S(t) = K_{mi} * Z_{mi} / R_{mi} \tag{23}$$

Where

K_{mi} is the gain, Z_{mi} is the zero polynomial and R_{mi} is the poles.

From (23), it can be seen that the numerator and the denominator is a function of state variables X and the location of these poles and zeros are further influenced by the modes of operation of the plant. In order to include these modal transitions, (23) has to be combined with (21). This can be rewritten as

$$W_{m_i}(s) = f(\Omega) * (K_{mi} * Z_{mi} / R_{mi}) \tag{24}$$

Oscillations in the system dynamics can thus be mapped through auxiliary states to the changes in system polynomial roots or the poles/zero combinations. Considering the above facts reference model transfer function can be a function of the fuzzy logic output

$$y_{mi}(t) = (M_i^T P_i \vartheta_i) * W_{mi} * S(t) \underline{\underline{\Delta}} (\Phi_i * \vartheta_i) * W_{mi} * S(t) \tag{25}$$

For a constant command signal (25) can be rewritten as

$$y_{m_i}(t) = \nu(\Phi_i, \vartheta_i, W_{m_i}) \tag{26}$$

Where

$y_{m_i}(t)$ is the reference model output for the ith mode

Φ_i is the parameter vector developed by the fuzzy system depending on the system operating points

ϑ_i is the membership function weights, ν is the coefficient index and W_{mi} is the corresponding reference model transfer function.

Clearly, the membership function weights act as a performance index function in modifying the reference model output. Based on each system modal transition, the parameter vector $\Phi * \vartheta$, which is the fuzzy system output, changes. Subsequently the reference model output moves such that the closed loop system is stable with its roots in the left half s plane. This movement in the reference model secures the system from becoming unstable.

Intelligent Agent #2: The RBFNN structure is used along with a MRAC to capture the functional non-linearity through online learning. Steps involved in the RBFNN design are as follows.

A. Selection of the Number of Nodes and Centers

The proposed growing RBFNN will start with three nodes. The centers for these nodes will be initialized to zeros except for the last one, which will be set to 0.1 in order to effectively calculate the radii with these centers. Thus the RBFNN is novice about the system that is being approximated or controlled. Further each time when a new input is passed on by the input layer, the distance and the outputs are calculated as

$$Oj = \exp(-\|(\Theta_k - C_{(j,k)})\| / r_j^2) \tag{27}$$

Where

j=1: number of nodes

k=1: number of input vector elements.

B. Distance Measurement

The distance measurement is Euclidean in nature based on the norm. At first the distance between each input vector element is calculated with all the centers and finally the norm is found to get the distance with respect to each node for all the developed nodes. This can be represented as

$$d_j = \sqrt{\sum_{k=1}^{Ni} (\Theta_k - C_{(j,k)})^2} \tag{28}$$

Where

N_i is number of input vector elements.

C. Calculation of the Radius

The radius is calculated from each center using

$$r = for\left\{ j = 1 : n \left[\max(dj = \sqrt{\sum_{k=1}^{Ni} (X_k - C_{(j,k)})^2}) \right] \right\} * 0.6 \tag{29}$$

Where

n is the number of nodes.

D. Selection of Basis Function

Most commonly used basis function is the Gaussian function expressed as

$$O_j = \exp(-d_j^2 / r_j^2) \tag{30}$$

Where

O is the network output.

This function has been chosen due to the fact that it gives most weight to points that are close to the nodal center and exponentially less weight to the points, which move, farther from the node centers.

E. Weight Selection and Online Adjustment

In order to sufficiently capture any non-linear function, the weights must be chosen correctly. To get the correct weights, the neural networks are ‘trained’ to learn the function, which is being approximated. However to train the weights, data regarding the inputs and the outputs need to be collected. The weights are initially adjusted to zeros. This means at the first instant the output obtained from the neural network will be a random value. At each instant when a new value of the input/output vector is formulated, the weights are updated. The updated law is based on a gradient descent technique,

$$e_t = r - y$$
$$W_j{}^* = W_j + U_p * e_t * d_j \tag{31}$$

Where

$W_j{}^*$ is the new value of the weights
W_j the old value
U_p is an update rate which is pre-defined number and
e_t is the output error

Intelligent Agent #3: The third agent will be a mathematical based algorithm on the MRAC. Details of the MRAC derivations are described in the appendix.

III. Application to Robotic Manipulator Position Tracking

The developed scheme (viz., NNPFAC) is tested for the control of position tracking on a single link manipulator dynamic nonlinear model. Several cases have been used for testing the ability of the controller. In the following analysis, two important cases are demonstrated. In the first case, the scheme is compared with two intelligent agents acting on the system without the neural network agent termed as FMRMAC. In the second case, the scheme is compared with two agents without the fuzzy logic agent termed as NNPAC.

A. *Single Link Flexible Robotic Manipulator Dynamic Model*

The robotic arm has the position zero x and y axes denoted as x_0 and y_0 as shown in fig. 3. Depending on the angular movement based on the required trajectory, there will be deflection, and the arm settles to new axes denoted as x_1 and y_1 at every time instant. This angular deflection is represented by the angle θ. The movement of the arm basically depends on the mass and the weight of the link, and the respective dynamic model shows the equations along with the parameter values. Further, the arm tip will have the effect of the tip load which is varied at time instant denoted as the Load Torque. Due to this, physically the arm tip may not track the trajectory, which is denoted by the angle α. In order to create this effect in the robotic manipulator dynamic model, the load torque is changed at different time instants online while the controller value in the form of the input voltage to the manipulator changes at each instant based on the output error.

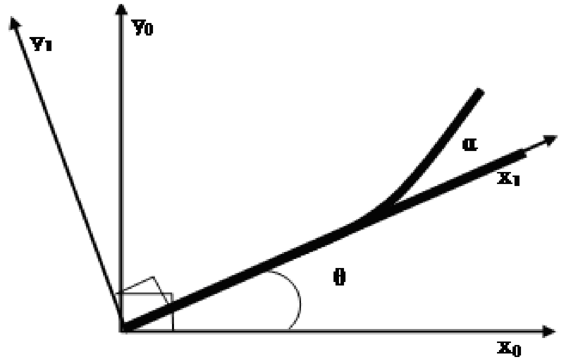

Fig. 3. Single Link Flexible Robotic Manipulator Model.

The important physical parameter, which relates the mass and the weight of the manipulator, is its angular position θ and the angular velocity. This along with the vibration mode constitutes the basic mathematical equations. However, the velocity and vibration mode in turn depend on the physical properties of the link such as mass, gravity, moment of inertia, elasticity and the length of the link. One of the sub-models is the DC motor model, which is used to drive the link with its torque and speed. The physical parameters involved in this modeling are the DC motor torque, resistance, inductance, gear ratio and efficiency. Thus, the complete model consists of five first order differential equations.

B. *Physical Details and Mathematical Equations*

Given the system dynamics [18], the robotic link's dynamics can be represented as below

$$y(1) = \beta(t), y(2) = \dot{\beta}(t), y(3) = \phi(t), y(4) = \dot{\phi}(t) \tag{32-35}$$

The variables y(1)- y(4) are robotic link's angular position, link's angular velocity, link's vibration mode, link's vibration mode first derivative respectively. Differentiating we get

$$\dot{y}(1) = \dot{\beta}(t), y(2) = \dot{\beta}(t), \dot{y}(3) = \dot{\phi}(t), y(4) = \dot{\phi}(t) \tag{36-39}$$

Emerging from above the complete model of a single link manipulator is as follows.

$$\dot{y}(1) = y(2)$$
$$\dot{y}(2) = \frac{1}{r_1 r_9 - r_5{}^2}\left[r_1(\tau + r_2 - r_3 - r_4) - r_5(r_6 + r_7 - r_8)\right]$$
$$\dot{y}(3) = y(4)$$
$$\dot{y}(4) = \frac{1}{r_1 r_9 - r_5{}^2}\left[r_9(r_6 + r_7 - r_8) - r_5(\tau + r_2 - r_3 - r_4)\right] \quad \text{and}$$
$$\dot{y}(5) = \frac{1}{L_a}\left[V_{in} - R_a y(5) - nK_b y(2)\right] \tag{40-44}$$

Where

$$\tau = n\eta\left(K_m y(5) - nK_d y(2) - T_f\right)$$
$$r_1 = \rho\zeta_1,\ r_2 = \frac{L^2}{2}\left[g\rho\cos(y(1))\right]$$
$$r_3 = 2\rho\zeta_1 y(3)y(4)y(2)$$
$$r_4 = g\rho y(3)\zeta_3\sin(y(1))$$
$$r_5 = \rho\zeta_2,\ r_6 = \rho y(3)y(2)^2\zeta_1$$
$$r_7 = g\rho\cos(y(1))\zeta_3$$
$$r_8 = EIy(3)\zeta_4\ \ r_9 = \frac{L^3}{3}\rho + \rho y(3)^2\zeta_1$$

The values of the parameters are given in Table I and the corresponding nomenclature is shown below.

TABLE I
PARAMETRIC VALUES OF A SINGLE LINK MANIPULATOR

R_a=8.33	n=65.5	ρ=0.1658
L_a=0.00617	η=0.66	ζ_1=0.4661541842
K_b=0.039534088	E=68.9e^9	ζ_2=0.3835861510

R_a and L_a : D.C motor resistance and inductance respectively.

K_m : D.C motor torque constant.

K_b : Back e.m.f. constant.

T_f and K_d : Torque friction and Damping coefficient respectively.

n and η : Gear ratio and gear efficiency respectively.

E : Young's modulus of elasticity.

I : Link's area moment of inertia with respect to its neutral axis.

L : Length of the link.

g : Earth's gravity.

ρ : Mass per unit length of the link.

$\zeta_1, \zeta_2, \zeta_3$ and ζ_4 : Link's area moment of inertia w. r. to all other axes.

IV. Simulation Results and Discussion

In each of these cases, tables below show variations in the load torque. Further, the output angular position for the desired time span is plotted and combinations of one intelligent agent with MRAC along with the reference models output. Relevant graphs for each of the cases are also shown. The fuzzy switching scheme in this example consists of two inputs, one output and twenty-five rules. Initially, a rule base has been developed based on the performance of the system under study. The rule base for this case study as shown in Table II consists of five input linguistic term membership function in the form of Very Small (VS), Small (S), Medium (M), Large (L) and Very Large (VL).

TABLE II
RULE BASE

		a₁				
Load Torque	ω_n	VS	S	M	L	VL
	VS	S	S	M	L	S
	S	S	S	M	L	S
	M	L	S	M	L	S
	L	S	S	M	L	VL
	VL	S	S	M	L	VL

The inputs are auxiliary parameters such as the tip load and a_1. The output values are the changes needed in the natural frequency of the second order reference model structure ω_n which is generated by the rule base. The rules have been developed analyzing the simulated response of the system. Implication method used is based on 'min' operation and centroid defuzzification method has been employed to generate crisp values. The reference model representation has the following specific form,

$$W_m(s) = \frac{\omega_n{}^2}{s^2 + 2\varsigma\omega_n + \omega_n{}^2} \quad (45)$$

In this example, the value of ς is set to 0.7. The value of ω_n will be evaluated at every time instant depending on the two auxiliary inputs mentioned above. By simulating and studying the process it was found that for the range of load torque [0, 24], the closed loop system response was best by keeping the range of ω_n [0, 24]. Considering the approach for following manipulator tip load, the fuzzy mapping is therefore, used for generating multiple reference models.

A. *Case 1: Tip Load Pattern 1*

In this test, the tip load of the manipulator is changed at different time instants as in Table III. Demonstration of this example is to show the ability of the design in the presence of unmodeled dynamics. The comparison is with a technique viz., FMRMAC in which only two agents (agents#1 and agent #3) has been used. At the load torque variation, as shown in Table III, the FMRMAC went unstable and the proposed scheme successfully controlled the plant. The position output comparison is shown in fig. 4. The effectiveness of the controller to track the position output in presence of un-modeled dynamics and the failure of FMRMAC is notable from the figure.

TABLE III
TIP LOAD VARIATION (CASE 2)

Time Range (sec)	0-1	1-3	3-5	5-7	7-8
Load Torque (Nm)	0	14	0	22	12

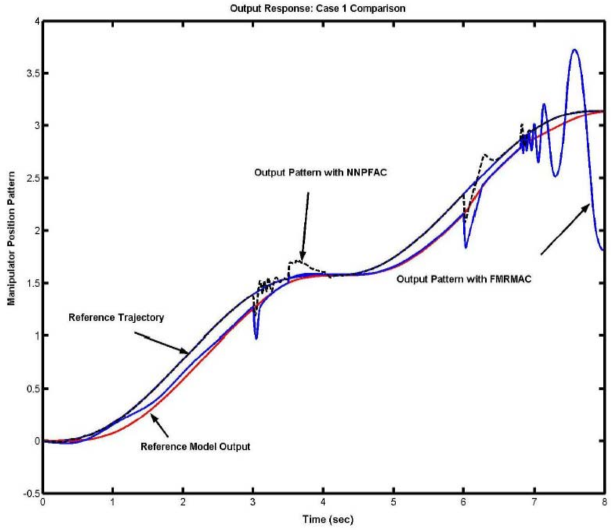

Fig.4. Output pattern comparison Case 1.

B. *Case 2: Tip Load Pattern 2*

In Case 2, the tip load of this manipulator is changed at different time instants as in Table IV. Moreover, between 6 to 6.5 seconds, the frictional load torque is augmented with a value of 0.03. This case is to demonstrate the ability of the algorithm and the ineffectiveness of a technique (viz., NNPAC algorithm in which agent#2 and agent #3 has been used) when the reference model structure is not the suitable one. Here, the w_n values for a suitable second order reference model structure when used with the NNPAC algorithm is kept as three.

TABLE IV
TIP LOAD VARIATION (CASE 2)

Time Range (sec)	0-3	3-3.5	3.5-6	6-6.8	6.8-8.0
Load Torque (Nm)	0	15	0	22	0

The output performance when compared to the NNPAC algorithm is as shown in fig. 5. As it can be seen the NNPAC becomes no longer stable when a different reference model is chosen. Thus it is apparent that the issue of reference model selection is critical for performance of the controller.

C. Case3: Tip Load Pattern 3

In Case 3 the manipulator tip load is changed at different time instant as in Table V. The ability of proposed controller with changes in the reference model to control this case when compared to the NNPAC algorithm is shown in fig. 6. Both the controllers perform well and the proposed scheme showing better tracking ability when compared to the NNPAC algorithm. The position error pattern is as shown in fig. 7, which clearly indicates this effect. The main factor that contributes to this effect is due to the reference model structure modification using fuzzy generator as opposed to a static one. It is worth noting that the value of the w_n for suitable second order reference model structure when used with NNPAC algorithm is kept as seven.

TABLE V
TIP LOAD VARIATION (CASE 3)

Time Range (sec)	0-3	3-3.5	3.5-6	6-6.8	6.8-8.0
Load Torque (Nm)	0	15	0	20	0

V. CONCLUSION

In this paper, a novel concept of intelligent agent supervisory loop controller is proposed for dynamic systems. The proposed scheme consists of three autonomous agents of which two are based on artificial intelligent techniques. These agents are autonomous and modular in nature and perform the control action effectively. The importance of the proposed scheme is its uniqueness in design and effectiveness in controlling complex and dynamic systems. Moreover, due to the hybrid design approach, the scheme is feasible and can be implemented using network and embedded control. Technically, it reduces the burden in traditional control and performs while the system is drastically changing or swings within uncertain modes. The simulation results conducted on several practical system models shows that this approach has immense potential in modular agent based practical control design. The scheme performed better than only one intelligent agent working in tandem with the traditional MRAC. It also holds the strength of an online RBFNN structure which reduces the size and offline training requirements for NN control.

APPENDIX

Any certainty equivalent adaptive control law can be used as a base control in this case. We use a direct MRAC approach as descried in [19]. Consider an unknown linear SISO plant described by,

$$x_p = A_p x_p + b_p u, y_p = h_p^T x_p \qquad (46\text{-}47)$$

Where

u is the input, y_p is the plant output, x_p is the *nth-order* state vector, A_p is an *n* x *n matrix*

Let us consider a reference model represented as

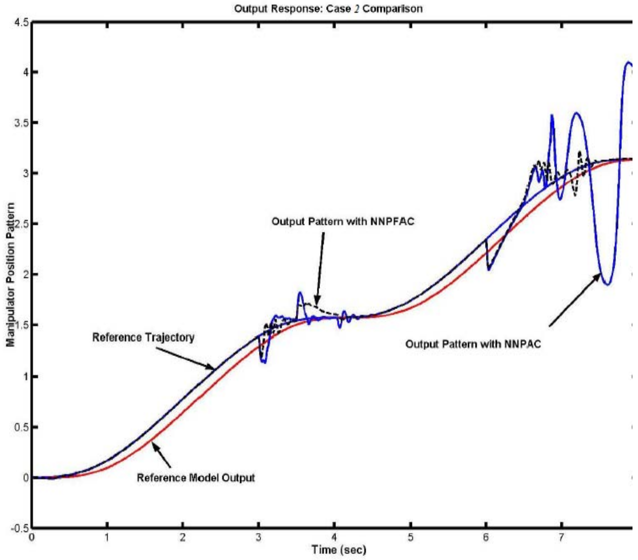

Fig.5 Output pattern comparison Case 2.

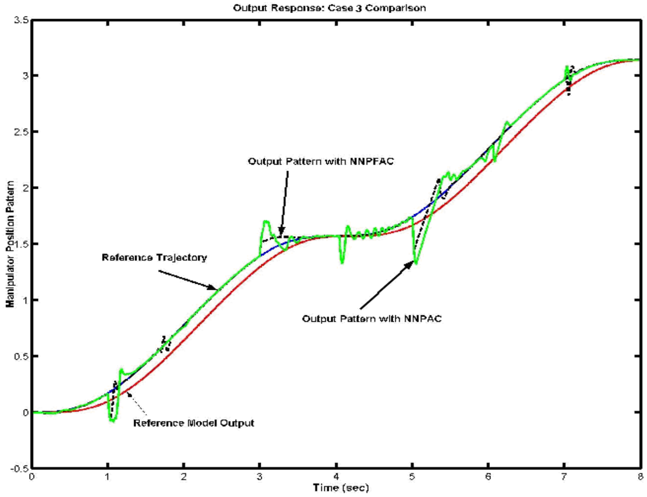

Fig.6. Output pattern comparison Case 3.

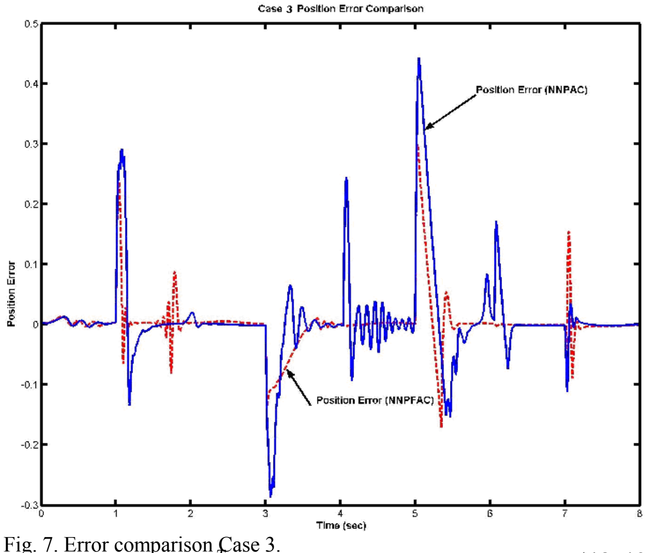

Fig. 7. Error comparison Case 3.

$$x_m = Ax_m + br, y_m = h^T x_m \qquad (48\text{-}49)$$

The aim is to find u such that the output error

$$e = y_p - y_m \tag{50}$$

tends to zero asymptotically for arbitrary initial conditions. The control structure for such a scheme is

$$u = \Theta^T \omega \tag{51}$$

Where

$\Theta = \begin{bmatrix} k & \theta_0 & \theta_1^T & \theta_2^T \end{bmatrix}^T$ is the control parameter vector and

$\omega = \begin{bmatrix} r & y_p & \omega_1^T & \omega_2^T \end{bmatrix}^T$ is the regressor vector.

The regression vectors are updated online based on the following equation

$$\omega_1 = \Lambda\omega_1 + Lu$$

$$\omega_2 = \Lambda\omega_2 + Ly_p \tag{52-53}$$

Where

Λ is a stable matrix of order $(n-1)\times(n-1)$ such that the determinant $|sI - \Lambda| = Z_m(s)$ and the vector L is defined as $L^t = [0 \;.....0 \; 1]$.

Further, the control signal u is structured as

$$k = -\mathrm{sgn}(K_p)er \;\; , \; \theta = -\mathrm{sgn}(K_p)ey_p \tag{54-55}$$

$$\theta_1^T = -\mathrm{sgn}(K_p)e\omega_1^T \quad \theta_2^T = -\mathrm{sgn}(K_p)e\omega_2^T \tag{56-57}$$

The above equations (51 - 57) are used in the control process.

ACKNOWLEDGMENT

The author would like to thank the advising assistance received from Dr. Adel A. Ghandakly, Professor and Chair at the California State University at Chico. Also the thoughtful comments and suggestions by anonymous reviewers and Dr. Abhiman Hande at the University of Texas at Dallas are greatly appreciated.

REFERENCES

[1] D.T. Magill, “Optimal Adaptive Estimation of Sampled Stochastic Processes”, IEEE Trans. on Automatic Control, vol. AC-1965, pp.434-439, Oct.1965.

[2] K.S. Narendra and J. Balakrishnan, “Adaptive Control Using Multiple Models”, IEEE Trans. on Automatic Control,vol.42, no. 2, pp.171-187, Feb.1997.

[3] T. Takagi, and M. Sugeno, “Fuzzy Identification of Systems and its Applications to Modeling and Control”, IEEE Trans. System Man Cybernet., vol. 15, no.1, pp.116-132, Jan. 1992.

[4] J.T. Spooner and K. M. Passino, “Stable Adaptive Control Using Fuzzy Systems and Neural Networks”, IEEE Trans. on Fuzzy Systems, vol. 4, no.3, pp.339-359, Aug. 1996.

[5] W. Shyong and C. J.Sun, “Fuzzy Model based Adaptive Control for a Class of Nonlinear Systems”, IEEE Trans. on Fuzzy systems, vol.9 no.3, pp.413-425, June 2001.

[6] A. Ichtev, “Multiple Fuzzy Models for Fault Tolerant Control”, International Journal of Fuzzy Systems, vol.5, no.1, pp.31-39, March 2003.

[7] L. Chen, K. S. Narendra, “Nonlinear Adaptive Control using Neural Networks and Multiple Models”, Automatica vol. 37, no.8, pp. 1245-1255, Aug. 2001.

[8] A J. Calise, N. Hovakimyan, M. Idan “Adaptive output feedback control of nonlinear systems using neural networks”, Automatica, vol. 37, no.8, pp. 1201- 1211, Aug. 2001.

[9] J. T. Lo, D. Bassu, “Adaptive Multilayer Perceptrons With Long-and Short-Term Memories”, IEEE Transactions on Neural Networks, vol. 13, no.1, pp 22-33, Jan. 2002.

[10] R. Li, Y. Zhang, Numerically Robust Implementation of Multiple-Model Algotithm, IEEE Trans. On Aerospace and Electronic Sysems, vol. 36, no. 1, pp.266-278, Jan. 2000.

[11] J. I. Arciniegas, A. H. Eltimsahy, and K. J. Cios, “Neural-networks-based adaptive control of flexible robotic arms”, Neurocomputing, vol. 17, no. 3-4, pp. 141-157, Nov.1997.

[12] S. Sri Ewari Devi, V. Ramachandran, Agent Based Control For Embedded Applications, Centre For Professional Development Education, Anna University, Chennai, 600 025

[13] D. B. Lange, Mobile Objects and Mobile Agents: The Future of Distributed Computing? ECOOP Proceedings 1987-2000

[14] K. S. Narendra, C. Xiang, “Adaptive Control of Discrete-Time Systems Using Multiple Models,” IEEE Trans. on Automatic Control, vol.45, no. 9, pp.1669-1685, Sept. 2000.

[15] P. S. Maybeck and R. D. Stevens, “ Reconfigurable Flight Control Via Multiple Model Adaptive Control Methods”, IEEE Trans. on aerospace and electronic systems, vol. 27, no.3, pp.470-479, May 1991.

[16] S. Kanev and M. Vergaegen, “Controller Reconfiguration for Non-Linear Systems,” Control Engineering Practice, vol.8, no.11, pp.1223-1235, Nov. 2000.

[17] S.G. Fabri and V. Kadirkamanathan, Functional Adaptive Control: An Intelligent Systems Approach, Springer-Verlag, 2001, ch 1,7-9.

[18] D. K. Wedding, Artificial Intelligent Control of Flexible Robotic Manipulators, Ph.D. Dissertation, The University of Toledo, Toledo, OH (1999)

[19] P. A. Ioannou and J. Sun, Robust Adaptive Control, Upper Saddle River, NJ: Prentice-Hall, 1996, ch.2-6.

S. Kamalasadan (M’ 06) received his Ph.D. in Electrical Engineering from the University of Toledo, Ohio in 2004, M.Eng in Electrical Power Systems Management, from the Asian Institute of Technology, Bangkok Thailand in 1999 and B Tech. degree in Electrical and Electronics from the University of Calicut, India in 1991. He is currently working as an Assistant Professor at the University of West Florida. He has held industrial positions for six years and has teaching and research experience for four years. He has won several awards including the faculty summer research award at UWF, Outstanding teaching award from the graduate student association University of Toledo, 2001- 2002 and from the college of engineering, 2002-2003. His research interests include Intelligent and Autonomous Control, Power Systems dynamics, Stability and Control, AI applications in control of Dynamic Systems and Real-time Embedded Systems Applications. He is a technical reviewer of IEEE Transactions of Power Systems, IEEE transactions of Fuzzy systems, IEEE transactions of education and several other international journals and conferences.