01 Jan 2006

Neuroadaptive Model Following Controller Design for a Nonaffine UAV Model

Nishant Unnikrishnan

S. N. Balakrishnan
Missouri University of Science and Technology, bala@mst.edu

Neuroadaptive Model Following Controller Design for a Nonaffine UAV Model

Nishant Unnikrishnan, and S. N. Balakrishnan

Abstract— **This paper proposes a new model-following adaptive control design technique for nonlinear systems that are nonaffine in control. The adaptive controller uses online neural networks that guarantee tracking in the presence of unmodeled dynamics and/or parameter uncertainties present in the system model through an online control adaptation procedure. The controller design is carried out in two steps: (i) synthesis of a set of neural networks which capture the unmodeled (neglected) dynamics or model uncertainties due to parametric variations and (ii) synthesis of a controller that drives the state of the actual plant to that of a reference model. This method is tested using a three degree of freedom model of a UAV. Numerical results which demonstrate these features and clearly bring out the potential of the proposed approach are presented in this paper.**

I. Introduction

THE field of artificial neural networks and its application to control systems has seen phenomenal growth in the last two decades. The main philosophy that is exploited heavily in system theory applications is the universal function approximation property of neural networks [1]. Benefits of using neural networks for control applications include its ability to effectively control nonlinear plants while adapting to unmodeled dynamics (terms in the plant equations that were not considered during model synthesis) and time-varying parameters.

A paper by Narendra and Parthasarathy demonstrated the potential and applicability of neural networks for the identification and control of nonlinear dynamical systems [2]. Sanner and Slotine [3] developed a direct adaptive tracking control architecture with Gaussian radial basis function networks to compensate for plant nonlinearities. In 1996, Lewis et al. [4] proposed an online neural network that approximates unknown functions and it was used in designing a controller for a robot. More important, their theoretical development also provided a Lyapunov stability analysis that guaranteed both tracking performance as well as boundedness of weights. However, the applicability of this technique is limited to systems which could be expressed in the "Brunovsky form" [5] and which are affine in the control variable (in state space form).

A relatively simpler and popular method of nonlinear control design is the technique of dynamic inversion, which is essentially based on the philosophy of feedback linearization [5,6]. A drawback of this approach is its sensitivity to modeling errors and parameter inaccuracies. Important contributions have come from Calise and his co-workers in a number of publications (e.g. [7, 8]), who have proposed to augment the dynamic inversion technique with neural networks so that the inversion error is cancelled out.

There always has been a need to develop efficient control design techniques to address modeling errors issues. Almost all the techniques mentioned above are applicable only for certain classes of nonlinear systems (control-affine systems, SISO systems etc). In this context, a more powerful tool is one that can address nonlinear systems in a more general form. An approach was first presented in [9], where the authors followed a model-following approach. The idea was to design an 'extra control' online, which when added to a nominal controller (designed off-line), lead to overall stability and improved the overall performance of the plant.

An approach, by Lang and Stengel [10] suggested the transformation of the nonaffine state equation to an augmented form where the control variable appears in a linear form which can then be solved. A systematic procedure to implement this approach on plants where the control appears in a nonaffine fashion is given in [11].

The objective of this paper is to present an approach to design adaptive controllers for nonaffine systems with uncertain parameters or unmodeled dynamics (nonlinearities) in the state equations relying on the philosophy presented in [10]. The controller design is carried out in two steps: (*i*) synthesis of a set of neural networks which capture the unmodeled (neglected) dynamics due to neglected terms or due to uncertainties in the parameters and (*ii*) computation of a controller that drives the state of the actual plant to that of a reference model. The weight update rule for the online networks is derived using a Lyapunov-based approach, which guarantees both stability of the error as well as boundedness of weights.

In this study, numerical results based on the UAV model provided in [11] are provided. Simulations not only consider parametric uncertainties present in the system model but also account for modeling nonlinearities that were neglected during modeling or that manifested during plant operation.

S. N. Balakrishnan, Professor, Department of Mechanical and Aerospace Enginering, University of Missouri Rolla, MO 65409, USA. (Corresponding author; phone: 573-341-4675; e-mail: bala@ umr.edu).

Nishant Unnikrishnan, Ph. D. Student, Department of Mechanical and Aerospace Engineering, University of Missouri Rolla, MO 65409, USA.(e-mail: nu7v3@umr.edu).

1-4244-0210-7/06/$20.00 ©2006 IEEE

The results obtained from this research bring out the potential of this approach. Rest of the paper is organized as follows. In Section 2, the control design technique and the two step process proposed in this paper are discussed. The neural network structure and the derivation of the weight update rule are also outlined in this section. Simulation studies on the UAV model and results are given in Section 3. Conclusions are drawn in section 4.

II. Control Design Procedure

The control design procedure proposed in this paper has two main components. In one part, the aim is to train the weights of the neural networks, so as to capture the unknown function that appears in the plant dynamics. In the other part, assuming a neural network approximation of the unknown algebraic function, the objective is to get the solution for the control variable that guarantees model-following. These two components of the control design procedure, starting with a mathematical description of the problem, are discussed in detail in the following subsections.

A. Problem Description

Let us consider a reference nonlinear system, which has the following representation

$$\dot{X}_d = f^*(X_d) \tag{1}$$

where $X_d \in R^n$ is the desired state vector. We assume that the order of the system n is known. The actual plant is assumed to have the following structure

$$\dot{X} = f(X,U) + d(X,U) \tag{2}$$

where $X \in R^n$ is the state of the actual plant and $U \in R^n$ is the controller (to be discussed later). The algebraic function $d(X,U) \in R^n$ arises because of two reasons: (*i*) there may be neglected algebraic terms in the model and (*ii*) the numerical values of the parameters may not perfectly represent the actual plant and this error results in unknown functions in the model. The controller U needs to be designed online such that the states of the actual plant follow the respective states of the reference model. In other words, the goal is to ensure that $X \to X_d$ as $t \to \infty$. As a means to achieve this, a prerequisite is to capture the unknown function $d(X,U)$, which is accomplished through a neural network approximation $\hat{d}(X,U)$ in this study. A needed intermediate step towards this is to define an 'approximate system' as follows

$$\dot{X}_a = f(X,U) + \hat{d}(X,U) + (X - X_a), \quad X_a(0) = X(0) \tag{3}$$

Through this artifice, we ensure that $X \to X_a \to X_d$ as $t \to \infty$. Obviously this introduces two tasks: (*i*) ensuring $X \to X_a$ as $t \to \infty$ and (*ii*) ensuring $X_a \to X_d$ as $t \to \infty$. We discuss these two steps separately in the following subsections. The reason of choosing an approximate system of the form in Eq. (3) is to facilitate meaningful bounds on the errors and weights.

B. Capturing the Unknown Function

First, we define $d(X,U) \equiv \left[d_1(X,U) \ \cdots \ d_n(X,U) \right]^T$, where $d_i(X,U), i = 1, \cdots, n$ is the i^{th} component of $d(X,U)$. In this study, each element of $d(X,U)$ is represented by a separate neural network. The idea is to have n neural networks for n independent channels, which facilitates easier mathematical analysis. Various researchers have worked on using one neural network to approximate the unknown nonlinearities using a single network in aerospace applications [7, 8]. Such an approach leads to one single network with a large number of hidden neurons in its structure. The approach of using a network for each state equation can be helpful in dealing with situations where the unknown terms in each state equation are of different magnitudes. *Polynomial basis neural networks* [12, 13] are used in this study for approximating each of the unknown functions $d_i(X,U)$. In order to form the vector of basis functions, the input data is first pre-processed. In our numerical experiments, we created C_i, $i = 1, \cdots, n$ which have a structure $C_i = \left[1 \ \ x_i \ \ x_i^2 \ \ \cdots \right]^T$ (the highest power depends on the application and was fixed by trial-and-error). The vector of basis functions is selected as

$$\Phi = kron\left(C_n, \cdots, kron\left(C_3, kron\left(C_1, C_2\right)\right), \cdots\right) \tag{4}$$

$kron(\square, \square)$ represent the "Kronecker product" and is defined as [12]

$$kron(Y,Z) = \left[y_1 z_1 \ \ y_1 z_2 \ \ \cdots \ \ y_n z_m \right]^T \tag{5}$$

where $Y \in R^n$ and $Z \in R^m$. The dimension of the neural network weight vector is same as the dimension of Φ.

The technique for updating the weights of the neural networks (i.e. training the networks) for accurate representations of the unknown functions $d_i(X,U), i = 1, \cdots, n$ is discussed here. Define

$$e_{a_i} \equiv \left(x_{a_i} - x_i \right) \tag{6}$$

From Eqs. (2-3), equations for the i^{th} channel can be decomposed as

$$\dot{x}_i = f_i(X,U) + d_i(X,U) \tag{7}$$

$$\dot{x}_{a_i} = f_i(X,U) + \hat{d}_i(X,U) - e_{a_i} \tag{8}$$

Subtracting Eq.(7) from Eq.(8) and using the definition in Eq.(6) gives

$$\dot{e}_{a_i} = \hat{d}_i(X,U) - d_i(X,U) - e_{a_i} \tag{9}$$

From the universal function approximation property of neural networks [13], it can be stated that there exists an ideal neural network with an optimum weight vector W_i and basis function vector $\Phi_i(X)$ that approximates $d_i(X)$ to an

accuracy of ε_i; i.e.

$$d_i(X,U) = W_i^T \Phi_i(X,U) + \varepsilon_i \quad (10)$$

Let the actual weight of the function approximating network be $\hat{W}_i$. The approximated function can be expressed as

$$\hat{d}_i(X,U) = \hat{W}_i^T \Phi_i(X,U) \quad (11)$$

The objective now is to derive a weight update rule. On substituting Eqs. (10-11) in Eq.(9) we get

$$\dot{e}_{a_i} = \tilde{W}_i^T \Phi_i(X,U) - \varepsilon_i - e_{a_i} \quad (12)$$

where $\tilde{W}_i \equiv (\hat{W}_i - W_i)$ is the error between the ideal weight and actual weight of the neural network. Note that $\dot{\tilde{W}}_i = \dot{\hat{W}}_i$ since W_i is constant.

Next, a weight update law to be implemented online is derived. Define a series of Lyapunov functions L_i, $i = 1, \ldots, n$ such that

$$L_i = \frac{1}{2}\left(e_{a_i}^{\ 2}\right) + \frac{1}{2}\left(\tilde{W}_i^T \gamma_i^{-1} \tilde{W}_i\right) \quad (13)$$

where $\gamma_i > 0$ is the learning rate for the i^{th} network. A weight update rule shown in the following equation is used to update the function approximating network weights online.

$$\dot{\hat{W}}_i = -\gamma_i e_{a_i} \varphi_i(X) - \gamma_i \sigma_i \hat{W}_i \quad (14)$$

where σ is a sigma modification term used to prove a bound on network weights. Lyapunov derivative based proof of error convergence and stability of the weight update rule is shown in detail in [14]

C. Control solution

In this part, we assume that the neural network approximation of the unknown function $\hat{d}(X,U)$ is available. We attempt to drive $X_a \to X_d$ as $t \to \infty$ by enforcing the following second order error dynamics

$$(\ddot{X}_a - \ddot{X}_d) + K_d(\dot{X}_a - \dot{X}_d) + K_p(X_a - X_d) = 0 \quad (15)$$

with positive definite gain matrices K_d and K_p. This scheme allows the designer to shape the transient response of the error dynamics in Eq. (15).

The approach presented here is based on taking the time derivative of Eq. (3) thus obtaining an affine model where $\dot{U}$ appears in a linear fashion. This method of getting affinity in the derivative of the control is not new [10, 11]. On differentiating $\dot{X}_a$,we get

$$\ddot{X}_a = \left(\frac{\partial f}{\partial X}\right)\dot{X} + \left(\frac{\partial f}{\partial U}\right)\dot{U} + \left(\frac{\partial \hat{d}}{\partial X}\right)\dot{X} + \left(\frac{\partial \hat{d}}{\partial U}\right)\dot{U} + \dot{X} - \dot{X}_a \quad (16)$$

Next, substituting Eqs. (1), (3) and (16) in Eq.(15) leads to

$$\left(\left(\frac{\partial f}{\partial X}\right)\dot{X} + \left(\frac{\partial f}{\partial U}\right)\dot{U} + \left(\frac{\partial \hat{d}}{\partial X}\right)\dot{X} + \left(\frac{\partial \hat{d}}{\partial U}\right)\dot{U} + \dot{X} - \dot{X}_a - \ddot{X}_d\right) + \ldots$$
$$\ldots K_d\left(f(X,U) + \hat{d}(X) + (X - X_a) - f^*(X_d)\right) + K_p(X_a - X_d) = 0 \quad (17)$$

Solving for the control law $\dot{U}$ from Eq. (17), and substituting $f(X,U) + \hat{d}(X,U)$ for $\dot{X}$, we obtain

$$\dot{U} = \left(\left(\frac{\partial f}{\partial U}\right) + \left(\frac{\partial \hat{d}}{\partial U}\right)\right)^{-1} \begin{pmatrix} \ddot{X}_d + \dot{X}_a - \left(f + \hat{d}\right) - \left(\frac{\partial f}{\partial X}\right)\left(f + \hat{d}\right) - \left(\frac{\partial \hat{d}}{\partial X}\right)\left(f + \hat{d}\right) - \ldots \\ \ldots K_d\left(f(X,U) + \hat{d}(X) + (X - X_a) - f^*(X_d)\right) - K_p(X_a - X_d) \end{pmatrix} \quad (18)$$

Note that the $U \in R^n$ in this formulation.

III. Simulation Studies

The system considered in this study is a three degree of freedom model of an unmanned aerial vehicle (UAV).The system equations used in this study are taken from [11]. The system equations governing the UAV dynamics are

$$\begin{aligned} \dot{V} &= g\left(\frac{T - D}{W} - \sin(\gamma)\right) \\ \dot{\gamma} &= \frac{g}{V}\left(k_n n \cos(\mu) - \cos(\gamma)\right) \\ \dot{\chi} &= \frac{g\, k_n n \sin(\mu)}{V \cos(\gamma)} \end{aligned} \quad (19)$$

The state variables of the UAV under consideration here are the airspeed V, flight path angle γ, and flight path heading χ. The control inputs in the state model are the thrust applied T, load factor n, and bank angle μ. g represents the gravitational constant, W is the weight, and D is the drag force. The drag force is given by a simple model as shown below.

$$D = 0.5\, \rho V^2 S\, C_{D0} + \frac{2\, k\, k_n^2 n^2 W^2}{\rho V^2 S} \quad (20)$$

k_n is the load factor effectiveness coefficient and takes values such that $0 < k_n \le 1$. A value of k_n less than one indicates controller failure. If the controller is performing satisfactorily, $k_n = 1$. The parameters used in the model are given in Table 1 and have been taken directly from [11].

Table 1: UAV system parameters

Parameter	**Value**
Density ρ	1.2251 km/m^3
Weight W	14515 kg
Reference Area S	37.16 m^2
Maximum thrust T_{max}	113868.8 N
Maximum Load Factor $n_{\max}$	7
Induced Drag Coefficient k	0.1
Parasite Drag Coefficient C_{D0}	0.02

In [11] an adaptive controller for nonaffine plants (CNAP) was shown to be effective in capturing three parametric uncertainties: uncertainties in the parasite drag coefficient C_{D0}, in the induced drag coefficient k and in the load factor effectiveness coefficient k_n. In this study, an attempt was made to study the effectiveness of the proposed method to approximate system uncertainties (unmodeled nonlinear

terms **and** parametric uncertainties) and design a controller to make the UAV states track desired reference states. The added challenge in this work was the presence of unmodeled nonlinear terms in the system model. The term neglected in the plant equation for this purpose was the first term in Eq. (20) representing drag, namely: $0.5\rho V^2 S C_{D0}$. As in [11], parametric uncertainties were assumed to be present in the values of k and k_n and C_{D0} .

Define $X \equiv [V, \gamma, \chi]^T$, $U \equiv [T, n, \mu]^T$. Also $c_{11} \equiv (0.5 g \rho S C_{D0})/W$, $c_{12} \equiv g$, $c_{13} \equiv g/W$, $c_{14} \equiv (2gW^2)/(W\rho S)$, $c_{21} \equiv -g$, $c_{22} \equiv g$, $c_{31} \equiv g$. The state equations now become

$$\begin{aligned} \dot{x}_1 &= -c_{11}x_1^2 - c_{12}\sin(x_2) + c_{13}u_1 - (c_{14}\, k\, k_n^2\, u_2^2)/x_1^2 \\ \dot{x}_2 &= c_{21}(\cos(x_2)/x_1) + c_{22}(k_n\, u_2 \cos(u_3))/x_1 \\ \dot{x}_3 &= c_{31}(k_n\, u_2 \sin(u_3))/(x_1 \cos(x_2)) \end{aligned} \tag{21}$$

The desired reference trajectories for the UAV states are $[x_{1d} \quad x_{2d}]^T = [300 \quad 0]^T$ and the heading angle is required to track the output of a reference model given by

$$\begin{aligned} \dot{x}_{m1} &= x_{m2} \\ \dot{x}_{m2} &= -x_{m1} - 1.4x_{m2} + r_\chi \\ x_{3d} &= x_{m1} \end{aligned} \tag{22}$$

where r_χ is a thirty degree heading angle doublet:

$$-r_\chi = \begin{cases} 0, & t \le 5 \\ 6(t-5), & 5 < t \le 10 \\ 30, & 10 < t \le 20 \\ 30(5-0.2t), & 20 < t \le 30 \\ -30, & 30 < t \le 40 \\ -30(9-0.2t), & 40 < t \le 45 \\ 0, & t > 45 \end{cases} \tag{23}$$

The simulation is carried out such that the parameters in the system model change after twenty seconds in the following manner

$$\begin{aligned} C_{D0}(t) &= 0.02(2 - e^{-(t-20)}) \\ k(t) &= 20(1 - e^{-(t-20)}) \\ k_n(t) &= 0.5(1 - e^{-(t-20)}) \end{aligned} \tag{24}$$

We assume that the changes in parameters given in Eq. (24) and the nonlinear term in the definition of drag ($-c_{11}x_1^2$) are unknown for purposes of controller design.

The approximate system is defined as follows

$$\begin{aligned} \dot{x}_{1a} &= -c_{12}\sin(x_2) + c_{13}u_1 - (c_{14}\, k\, k_n^2\, u_2^2)/x_1^2 + \hat{d}_1(X,U) + x_1 - x_{1a} \\ \dot{x}_{2a} &= c_{21}(\cos(x_2)/x_1) + c_{22}(k_n\, u_2 \cos(u_3))/x_1 + \hat{d}_2(X,U) + x_2 - x_{2a} \\ \dot{x}_{3a} &= c_{31}(k_n\, u_2 \sin(u_3))/(x_1 \cos(x_2)) + \hat{d}_3(X,U) + x_3 - x_{3a} \end{aligned} \tag{25}$$

Notice that the nonlinear term and the parametric uncertainty in the first equation of the state model is captured by the neural network $\hat{d}_1(X,U)$. Similarly, the terms that appear due to parametric uncertainties in the second and third equations of the state model are approximated by $\hat{d}_2(X,U)$ and $\hat{d}_3(X,U)$ respectively. Basis function neural networks were used in this study for approximating the unmodeled dynamics. For the first network, vectors C_{1i} , $i = 1,2$ which have a structure $C_{11} = [0.001 \;\; 0.001x_1 \;\; 0.001x_1^2]^T$ and $C_{12} = [0.001 \;\; 0.001u_2 \;\; 0.001u_2^2]^T$ were created. The vector of basis functions Φ was composed of all possible products of elements of C_i's . By using kronecker products to represent the neuron interactions, Φ was composed as $\Phi_1 = kron(C_{11}, C_{12})$ where $kron(*,*)$ represents the kronecker product. The neural network that approximated the uncertainties in the first state equation consisted of nine neurons. The second network had eight neurons and vectors C_{2i} , $i = 1,2,3$ were created to preprocess the data to form a set of polynomial basis. The vectors had a structure $C_{21} = [0.1 \;\; 0.01x_1]^T$, $C_{22} = [0.1 \;\; 0.01u_2]^T$, $C_{23} = [0.1 \;\; 0.01u_3]^T$. The basis function vector was synthesized to have a structure $\Phi_2 = kron(kron(C_{21}, C_{22}), C_{23})$. The basis function vector for the third network had a structure $\Phi_3 = kron(kron(kron(C_{31}, C_{32}), C_{33}), C_{34})$ where $C_{31} = [0.1 \;\; 0.02x_1]^T$, $C_{32} = [0.1 \;\; 0.02x_2]^T$, $C_{33} = [0.1 \;\; 0.01u_1]^T$ and $C_{34} = [0.1 \;\; 0.01u_2]^T$ .The third network was composed of sixteen neurons. Neural network training parameters used were $[p_1 \; p_2 \; p_3]^T = [10 \;\; 100 \;\; 500]^T$ and $[\gamma_1 \; \gamma_2 \; \gamma_3]^T = [100 \;\; 100 \;\; 100]^T$.

The second stage of the controller design was to ensure $X_a \to X_d$ where X_d was the vector of desired trajectories. This was achieved through a second order error equation as given by Eq. (15). On differentiating Eq. (3) to obtain an expression for $\ddot{X}_a$ in the error dynamic equation, it was seen that

$$\ddot{X}_a = \frac{\partial f}{\partial X}\dot{X} + \frac{\partial f}{\partial U}\dot{U} + \begin{bmatrix} \hat{W}_1^T(\partial\Phi_1/\partial X) \\ \hat{W}_2^T(\partial\Phi_2/\partial X) \\ \hat{W}_3^T(\partial\Phi_3/\partial X) \end{bmatrix}\dot{X} + \begin{bmatrix} \hat{W}_1^T(\partial\Phi_1/\partial U) \\ \hat{W}_2^T(\partial\Phi_2/\partial U) \\ \hat{W}_3^T(\partial\Phi_3/\partial U) \end{bmatrix}\dot{U} + \begin{bmatrix} \dot{\hat{W}}_1^T\Phi_1 \\ \dot{\hat{W}}_2^T\Phi_2 \\ \dot{\hat{W}}_3^T\Phi_3 \end{bmatrix} + \dot{X} - \dot{X}_a \tag{26}$$

Define $Q \equiv \begin{bmatrix} \hat{W}_1^T(\partial\Phi_1/\partial X) \\ \hat{W}_2^T(\partial\Phi_2/\partial X) \\ \hat{W}_3^T(\partial\Phi_3/\partial X) \end{bmatrix}$, $R \equiv \begin{bmatrix} \hat{W}_1^T(\partial\Phi_1/\partial U) \\ \hat{W}_2^T(\partial\Phi_2/\partial U) \\ \hat{W}_3^T(\partial\Phi_3/\partial U) \end{bmatrix}$, and $T \equiv \begin{bmatrix} \dot{\hat{W}}_1^T\Phi_1 \\ \dot{\hat{W}}_2^T\Phi_2 \\ \dot{\hat{W}}_3^T\Phi_3 \end{bmatrix}$.

Eq. (26) could now be rewritten as

$$\ddot{X}_a = \left(\frac{\partial f}{\partial X} + Q\right)\dot{X} + \left(\frac{\partial f}{\partial U} + R\right)\dot{U} + T + \dot{X} - \dot{X}_a \tag{27}$$

The expansion of Eq. (15) for this problem was

$$\left(\left(\frac{\partial f}{\partial X} + Q\right)\dot{X} + \left(\frac{\partial f}{\partial U} + R\right)\dot{U} + T + \dot{X} - \dot{X}_a - \ddot{X}_d\right) + K_d(\dot{X}_a - \dot{X}_d) + K_p(X_a - X_d) = 0 \tag{28}$$

Define $H \equiv \left(\frac{\partial f}{\partial X} + Q\right)\dot{X} + T + \dot{X} - \dot{X}_a$. Now, the control term could be solved by rearranging Eq. (28) and expressed as

$$\dot{U} = \left(\frac{\partial f}{\partial U} + R\right)^{-1}\left(\ddot{X}_d - H - K_d(\dot{X}_a - \dot{X}_d) - K_p(X_a - X_d)\right) \tag{29}$$

In this work,

$$\frac{\partial f}{\partial X}=\begin{bmatrix} \left(0.2c_{14}u_2^2\right) & \left(-c_{12}\cos(x_2)\right) & 0 \\ \left(-\frac{c_{21}\cos(x_2)}{x_1^2}-\frac{c_{22}u_2\cos(u_3)}{x_1^2}\right) & \left(-\frac{c_{21}\sin(x_2)}{x_1}\right) & 0 \\ \left(-\frac{c_{31}u_2\sin(u_2)}{\cos(x_2)x_1^2}\right) & \left(\frac{c_{31}u_2\sin(u_3)\sin(x_2)}{x_1\cos(x_2)^2}\right) & 0 \end{bmatrix} \quad (30)$$

and

$$\frac{\partial f}{\partial U}=\begin{bmatrix} (c_{13}) & \left(\frac{-0.2c_{14}u_2}{x_1^2}\right) & 0 \\ 0 & \left(\frac{c_{22}\cos(u_3)}{x_1}\right) & \left(-\frac{c_{22}u_2\sin(u_3)}{x_1}\right) \\ 0 & \left(\frac{c_{31}\sin(u_3)}{\cos(x_2)x_1}\right) & \left(\frac{c_{31}u_2\cos(u_3)}{x_1\cos(x_2)}\right) \end{bmatrix} \quad (31)$$

Once the control variable in this problem $\dot{U}$ was obtained the actual control U could be obtained by a numerical integration procedure. In this work, Euler integration was performed to obtain the actual control signal U. The design parameters K_d and K_p were chosen as

$$K_d = diag[80 \quad 40 \quad 40], K_p = diag[80 \quad 40 \quad 40] \quad (32)$$

A time simulation of eighty seconds was carried out. Euler integraton with a step size of $\Delta t = 0.01$ was used for simulation purposes. Figure 1 is a plot containing errors in state variables and the control variables for the scheme if the approximations for uncertainties in the system equations were not carried out. Error in velocity is given in m/s, errors in flight path angle and in flight path heading angle are given in degrees. It can be seen that the state variable tracking is not satisfactory. The next three plots in Figure 1 are control trajectories. Thrust is plotted in KN, and bank angle is expressed in degrees. Figure 2 is an illustration of the errors in the state variables and control signals with the neuroadaptive scheme employed in control synthesis. The significant improvement in performance can be immediately noticed. Velocity tracking error does not exceed 40 m/s. Flight path angle and heading angle tracking errors are very small. Thus satisfactory tracking performance has been obtained. The small tracking error could be achieved due to the good function approximation properties of the three online neural networks attached to the approximate system dynamics and later on used for control synthesis. The last three plots in Figure 2 illustrate control histories with the neuroadaptive scheme. The three plots in Figure 3 show how well the neural networks capture the three unknown uncertainties/nonlinearities present in the actual plant equations.

IV. CONCLUSIONS

Dynamic systems and processes are difficult to model accurately. They may also change with time. It is essential that these unmodeled terms or changes in parameters are captured and are used to adapt the controller. A model-following adaptive controller using neural networks has been developed in this study for a fairly general class of nonlinear systems which are non-affine in the control variable. Although the idea of taking the time derivative of the state variable equation and obtaining a linear form for the derivative of the control is not new, there had been no attempt to couple this technique with adaptive control methods. This work uses a neuroadaptive scheme to account for parametric and unmodeled uncertainties (nonlinearities) and then in a separate step converts the nonaffine system to an affine in control system to solve for control. The nonlinear system for which the method is applicable is assumed to be of known order and has the same number of control inputs as there are states in the system, but it may contain unmodeled dynamics and/or parameter uncertainties. Tracking performance of the neuroadaptive control scheme has been shown through simulation studies on a three degree of freedom UAV model. The results of simulation studies performed show the potential of this technique.

ACKNOWLEDGMENT

This research was supported by NSF-USA grants 0201076 and 0324428.

REFERENCES

[1] Barto, A. G., Sutton, R. S., and Anderson, C. W., 1983, "Neuron-like Adaptive Elements that Can Solve Difficult Control Problems," IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-13, No. 5, pp. 834-846.

[2] Narendra, K. S. and Parthasarathy, K., 1990, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Transactions on Neural Networks, Vol. 1, No. 1, pp. 4-27.

[3] Sanner, R. M. and Slotine, J. J. E., 1992, "Gaussian Networks for Direct Adaptive Control", IEEE Transactions on Neural Networks", Vol. 3, No. 6, March, pp. 837-863

[4] Lewis, F. L., Yesildirek, A. and Liu, K., 1996, "Multilayer Neural Net Robot Controller with Guaranteed Tracking Performance", IEEE Transactions on Neural Networks", Vol. 7, No. 2, pp.388-399

[5] Khalil, H. K., 1996, " Nonlinear Systems", Prentice Hall Inc, New Jersey, USA.

[6] Slotine, J-J. E. and Li, W., 1991, "Applied Nonlinear Control", Prentice Hall.

[7] Kim, B. S. and Calise, A. J., 1997, "Nonlinear Filight Control using Neural Networks", AIAA Journal of Guidance, Control, and Dynamics, Vol. 20, No. 1, pp.26-33.

[8] Leitner, J., Calise, A. and Prasad, J. V. R., 1997, "Analysis of Adaptive Neural Networks for Helicopter Flight Controls", AIAA Journal of Guidance, Control, and Dynamics, Vol. 20, No. 5, pp.972-979.

[9] Balakrishnan, S. N. and Huang, Z., 2001, "Robust Adaptive Critic Based Neurocontrollers for Helicopter with Unmodeled Uncertainties", Proceedings of the 2001 AIAA conference on Guidance, Navigation and Control.

[10] Lane, S. H., and Stengel, R. F., 1988, "Flight Control Design Using Nonlinear Inverse Dynamics", Automatica, Vol. 24, No. 4, pp. 471-483.

[11] Boskovic, J. D., Chen, L., and Mehra, R. K., 2004, "Adaptive Control Design for Nonaffine Models Arising in Flight Control", Journal of Guidance Control and Dynamics, Vol. 27, No. 2, pp. 209-217.

[12] Ham, F. M. and Kostanic, I., 2001, "Principles of Neurocomputing for Science and Engineering", Mc Graw Hill, Inc.

[13] Hassoun, M. H., 1995, "Fundamentals of Artificial Neural Networks", MITPress, Cambridge, MA.

[14] Unnikrishnan, N. and Balakrishnan, S. N., "Neuroadaptive Model Following Controller Design for a Nonaffine UAV model", submitted to Journal of Guidance Control and Dynamics.

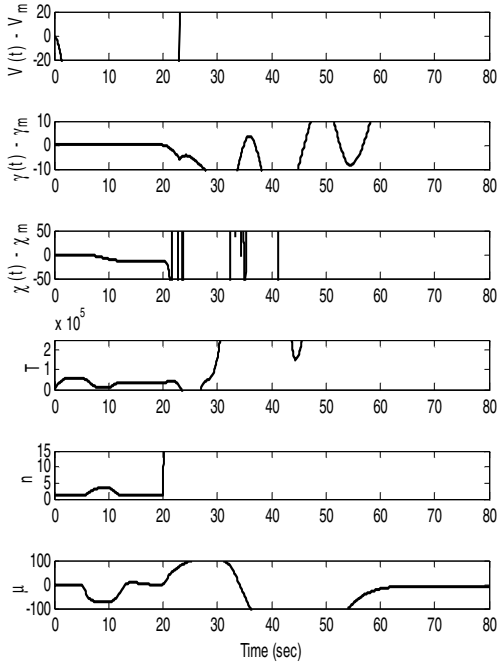

Figure 1: Tracking errors in state variables and control signals (nominal control scheme)

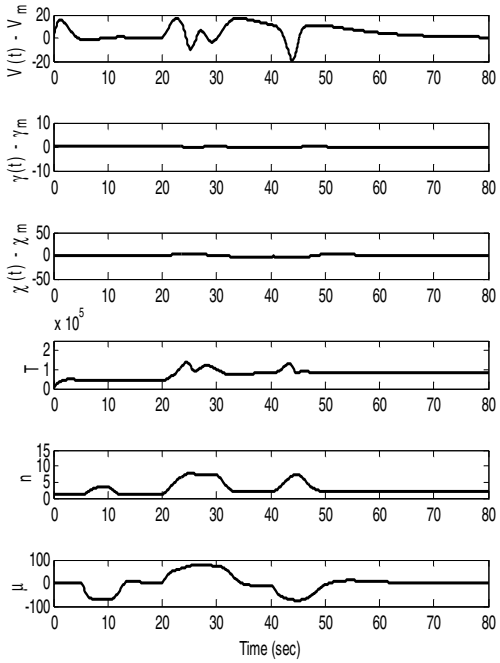

Figure 2: Tracking errors in state variables and control signals (neuroadaptive control scheme)

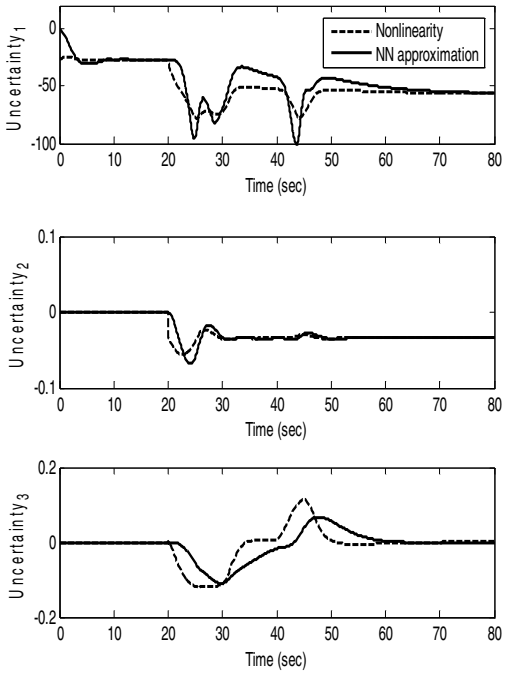

Figure 3: Uncertainty approximations in state equations