01 Jan 2005

# Dynamic Re-Optimization of a MEMS Controller in Presence of Unmodeled Uncertainties

Nishant Unnikrishnan

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

Venkat Durbha

## Recommended Citation

N. Unnikrishnan et al., "Dynamic Re-Optimization of a MEMS Controller in Presence of Unmodeled Uncertainties," *Proceedings of the 44th IEEE Conference on Decision and Control*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2005.
The definitive version is available at https://doi.org/10.1109/CDC.2005.1583378

Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

# Dynamic Re-optimization of a MEMS Controller in Presence of Unmodeled Uncertainties

Nishant Unnikrishnan, Venkat Durbha, S. N. Balakrishnan, *Member, IEEE*

*Abstract*— **Online trained neural networks have become popular in recent years in designing robust and adaptive controllers for dynamic systems with uncertainties in their system equations because of their universal function approximation property. This paper discusses a technique that dynamically reoptimizes a Single Network Adaptive Critic (SNAC) based optimal controller in the presence of unmodeled uncertainties. The controller design is carried out in two steps: (*i*) synthesis of a set of online neural networks that capture the uncertainties in the plant equations on-line (*ii*) re-optimization of the existing optimal controller to drive the states of the plant to a desired reference by minimizing a predefined cost function. The neural network weight update rule for the online networks has been derived using Lyapunov theory that guarantees stability of the error dynamics as well as boundedness of the weights. This approach has been applied in the online re-optimization of a micro-electromechanical device controller and numerical results from simulation studies are presented here.**

## I. Introduction

Many difficult real-life control design problems can be formulated in the framework of optimal control theory. It is well-known that the dynamic programming formulation offers the most comprehensive solution to compute nonlinear optimal control in a state feedback form [1]. An innovative idea was proposed in [2] to get around the numerical complexity of solving the associated Hamilton-Jacobi-Bellman (HJB) equation by using an 'Approximate Dynamic Programming (ADP)' formulation. In one version of this approach, called the Dual Heuristic Programming (DHP), one network (called the action network) represents the mapping between the state and control variables while a second network (called the critic network) represents the mapping between the state and costate variables.

A significant improvement to the adaptive critic architecture was proposed by [3]. It is named Single Network Adaptive Critic (SNAC) because it uses only the critic network instead of the action-critic dual network set up in typical adaptive critic architecture. SNAC is applicable to a large class of problems for which the optimal control (stationary) equation is explicitly solvable for control in terms of state and costate variables. SNAC eliminates the iterative training loops between the action and critic networks and consequently there are computational savings.

In the recent past, there has been a lot of interest in the use of neural networks for direct closed loop controller design that guarantee desired performance in presence of uncertainties and unmodeled dynamics [4]. An adaptive optimal controller that makes use of online neural networks to approximate parametric/unmodeled nonlinear uncertainties for general control affine systems of the form $\dot{X} = f(X) + g(X)U$ is discussed in this work. The weight update rule used in the neural networks in this work is very similar to the Lyapunov based weight update rule used in Leitner's work [4]. The uniqueness of the method proposed in this work is that the online function approximating network can be used to re-optimize in real time an existing Single Network Adaptive Critic [3] based optimal controller that has already been designed for a nominal system. This method is also unique in that unmatched uncertainties can be dealt with. Section 2 discusses approximate dynamic programming and the Single Network Adaptive Critic technique for optimal control design. Section 3 outlines the online approximation of system uncertainties and the Lyapunov based online weight update rule used in this work. Online re-optimization of the SNAC controller is discussed in section 4. Simulation studies are performed and results are given in section 5. The MEMS example considered here involves the presence of an unmatched parametric uncertainty in the system model that causes unmodeled nonlinearities to be present in the tracking error equations. The objective is dynamic reoptimization of the existing SNAC controller designed for a nominal MEMS model. Conclusions are drawn in section 6. Convergence proof of the SNAC technique for control affine linear systems has been given in the appendix.

## II. Approximate Dynamic Programming

### *A. Outline*

In this section, we attempt to outline the principles of approximate (discrete) dynamic programming, on which the SNAC approach is based on. An interested reader can find

Nishant Unnikrishnan: Ph.D. student, Department of Mechanical and Aerospace Engineering, University of Missouri, Rolla, USA. (E-mail: nu7v3@umr.edu).
Venkat Durbha: Masters student, Department of Mechanical and Aerospace Engineering, University of Missouri, Rolla, USA. (E-mail: vpdkc7@umr.edu)
Dr. S. N. Balakrishnan: Professor of Aerospace Engineering, Department of Mechanical and Aerospace Engineering, University of Missouri, Rolla, USA. (Tel: 1(573)341-4675, Fax: 1(573)341-4607, E-mail: bala@umr.edu).

This research was supported by NSF grants 0201076 and 0324428.

more details about the derivations in [5]. In discrete-time formulation, the aim is to find an admissible control $U_k$, which causes the system described by the *state equation*

$$X_{k+1} = F_k(X_k, U_k) \tag{1}$$

to follow an admissible trajectory that optimizes a sensible performance index $J$ given by

$$J = \sum_{k=1}^{N-1} \Psi_k(X_k, U_k) \tag{2}$$

where, the subscript $k$ denotes the time step. $X_k$ and $U_k$ represent the $n\times 1$ state vector and $m\times 1$ control vector, respectively, at time step $k$. One can notice that when $N\to\infty$, this leads to the *infinite time* problem. First, the cost function from time step $k$ is denoted as

$$J_k = \sum_{\tilde{k}=k}^{N-1} \Psi_{\tilde{k}}(X_{\tilde{k}}, U_{\tilde{k}}) \tag{3}$$

Then $J_k$ can be rewritten as

$$J_k = \Psi_k + J_{k+1} \tag{4}$$

where $\Psi_k$ and $J_{k+1} = \sum_{\tilde{k}=k+1}^{N-1} \Psi_{\tilde{k}}$ represent the *utility function* at time step $k$ and the *cost-to-go* from time step $k+1$ to $N$, respectively. The *costate* vector is defined as

$$\lambda_k = \frac{\partial J_k}{\partial X_k} \tag{5}$$

For *optimal control (stationary) equation*, the necessary condition for optimality is given by

$$\frac{\partial J_k}{\partial U_k} = 0 \tag{6}$$

The *optimal control equation* can be written as

$$\left(\frac{\partial \Psi_k}{\partial U_k}\right) + \left(\frac{\partial X_{k+1}}{\partial U_k}\right)^T \lambda_{k+1} = 0 \tag{7}$$

Using Eq.(7), *on the optimal path,* the costate equation can be expressed as

$$\lambda_k = \left(\frac{\partial \Psi_k}{\partial X_k}\right) + \left(\frac{\partial X_{k+1}}{\partial X_k}\right)^T \lambda_{k+1} \tag{8}$$

Eqs.(1), (7) and (8) have to be solved simultaneously, along with appropriate boundary conditions, for the synthesis of optimal control.

### *B. Single Network Adaptive Critic(SNAC)*

The SNAC technique retains all powerful features of the dual network Adaptive Critic (AC) methodology, while eliminating the action network completely. Details of the AC methodology have been provided in [5]. Note that in SNAC design, the critic network captures the functional relationship between $X_k$ and $\lambda_{k+1}$, whereas in AC design the critic network capture the relationship between $X_k$ and $\lambda_k$. Note that the SNAC method is valid only for the class of problems where the optimal control equation Eq.(7) is explicitly solvable for control variable $U_k$ in terms of the state variable $X_k$ and costate variable $\lambda_{k+1}$. Details regarding the neural network training and convergence checks can be obtained from [3].

## III. Neural Network Based Adaptation And Online Weight Update Rule

In this section, a novel technique that is used to capture parametric uncertainties or unmodeled nonlinearities that may be present in the plant dynamics but are not considered in the system model used for controller design is briefly discussed. This method has been elaborated in [6]. The uncertainty approximation is achieved using an online neural network in each system equation. Consider a general nonlinear system with the following structure

$$\dot{X} = f(X) + g(X)U \tag{9}$$

The control vector drives the system from an initial point to a final desired point optimizing a sensible performance index J given by

$$J = \int_0^\infty \Psi(X,U)dt \tag{10}$$

It is assumed that a pre-designed SNAC optimal control $U$ is available to drive the modeled system along a desired trajectory. Let the actual plant have the structure

$$\dot{X} = f(X) + g(X)U + d(X) \tag{11}$$

where the controller $U$ will have to be re-optimized to optimize the plant performance with the unmodeled dynamics $d(X)$ present. Since the term $d(X)$ in the plant equation is unknown, the first step in controller re-optimization is to approximate the uncertainty in the plant equation. For this purpose, a virtual plant is defined. The dynamics of this virtual plant is governed by

$$\dot{X}_a = f(X) + g(X)U + \hat{d}(X) + X - X_a, \qquad X_a(0) = X(0) \tag{12}$$

We assume that we have all the actual plant states, $X$, available for measurement at every step. The term $\hat{d}(X)$ is the neural network approximation of the unmodeled dynamics of the system which is a function of the actual plant state. Subtracting Eq.(12) from Eq. (11) we have $\dot{X} - \dot{X}_a = d(X) - \hat{d}(X) - X + X_a$ or $\dot{E}_a = d(X) - \hat{d}(X) - E_a$ where $E_a \equiv X - X_a$. It can be seen that as $d(X) - \hat{d}(X)$ approaches zero, the expression becomes an exponentially stable differential equation, *i.e.* $E_a \to 0$ as time $t\to\infty$. Defining $d(X) \equiv [d_1(X)\ldots d_n(X)]^T$, where $d_i(X)$ denotes the unmodeled dynamics in the differential equation for the $i^{th}$ state of the system. The approach in this study is to have '$n$' neural networks (one for each component of the unmodeled dynamics) so as allow for simpler development and analysis. Let us assume that there exists a neural network with an optimum set of weights $W_i$ and a basis function vector, $\varphi_i(X)$ that approximates $d_i(X)$ within a certain accuracy $\varepsilon_i$. Thus we have

$$d_i(X) = W_i^T \varphi_i(X) + \varepsilon_i \tag{13}$$

Also $\hat{d}_i(X)=\hat{W}_i^T\varphi_i(X)$, where $\hat{W}_i^T\varphi_i(X)$ is the output of the actual neural network. $\hat{W}_i$ represents the actual network weights. A stable weight update rule is

$$\dot{\hat{W}}_i=\Gamma_i e_{a_i} p_i \varphi_i(X) \tag{14}$$

which can be obtained from a quadratic Lyapunov function of the form $L_i=(1/2)e_{ai}^2 p_i+(1/2)(\tilde{W}_i^T\Gamma_i^{-1}\tilde{W}_i)$ where $e_{a_i}\equiv x_i-x_{a_i}$ and $\tilde{W}_i\equiv W_i-\hat{W}_i$ .

## IV. Dynamic Re-optimization of The SNAC Controller

In this section we discuss how the state and costate equations get updated online that help in reoptimizing the critic network. The steps of this process are detailed below.

It can be seen in subsection II.B that the main components of the SNAC controller design architecture are the Critic Network, the optimal control equations, the state equations and the costate equations. For implementation purposes, the state equation Eq. (9) is rewritten in a discrete form as

$$X_{k+1}=X_k+\Delta t(f(X_k)+g(X_k)U_k) \tag{15}$$

where $\Delta t$ is the time step of integration.

The performance index Eq. (10) can be expressed as

$$J=\sum_{k=1}^{\infty}\Psi_k(X_k,U_k) \tag{16}$$

In a discrete form, the actual plant equation Eq. (11) is written as

$$X_{k+1}=X_k+\Delta t(f(X_k)+g(X_k)U_k+d(X_k)) \tag{17}$$

In this study Euler integration scheme has been followed for implementation purposes.

The critic network is trained to represent the mapping between $X_k$ and $\lambda_{k+1}$ for the cost function given by Eq. (16) subject to the nominal state equation Eq. (15). The actual plant is given by Eq. (17) where the uncertainty $d(X_k)$ is present in the system dynamics. The critic network has not been trained with the actual state equation and hence is not the optimal critic for the actual plant. On close examination it can be seen that the costate equations will have to be modified so that an online training routine can help the critic capture the optimal relation between $X_k$ and $\lambda_{k+1}$. The uncertainty in the actual plant dynamics is captured by the online neural network and is represented by $\hat{d}(X_k)$ . It should be noted here that the inputs to the neural network are the states of the actual plant which we assume are readily available for measurement at every time step. In a discrete format, the weight update rule of the network that approximates the plant uncertainty is expressed as

$$\hat{W}_{i,k+1}=\hat{W}_{i,k}+\Delta t(\Gamma_i e_{a_{i,k+1}} p_i \varphi_i(X_k)) \tag{18}$$

Revisiting the costate equation (Eq. (8)), it can be seen that there is a term that involves $\left(\partial X_{k+1}/\partial X_k\right)$. An essential part of the actual plant equation is the uncertainty $d(X_k)$ . This term will have to be incorporated into the costate equation to ensure optimality of the costate. On replacing $d(X_k)$ with $\hat{d}(X_k)$ in Eq. (17) and using it in the costate equation, the new costate equation can be written as

$$\lambda_k=\left(\frac{\partial\Psi_k}{\partial X_k}\right)+\left[1+\Delta t\left(\frac{\partial f(X_k)}{\partial X_k}\right)+\Delta t\left(\frac{\partial g(X_k)}{\partial X_k}\right)U_k+\Delta t\left(\frac{\partial \hat{d}(X_k)}{\partial X_k}\right)\right]^T\lambda_{k+1} \tag{19}$$

The uncertainty approximation $\hat{d}(X_k)$ is given by $\hat{W}_k^T\varphi(X_k)$ (output of the online neural network). The partial derivative term $\left(\partial\hat{d}(X_k)/\partial X_k\right)$ can be written as $\hat{W}_k^T\frac{\partial\varphi(X_k)}{\partial X_k}$. Since the basis functions, $\varphi(X_k)$ are chosen by the control designer, the partial derivative of the basis functions can be calculated offline. This ensures that the costate equation gets updated online as the online neural network approximates the uncertainty. The reoptimization scheme is represented in Figure 1.

The steps for dynamic (online) critic re-optimization are as follows:

1. For each step $k$ , follow the steps below:
   - Input $X_k$ to the critic network to obtain $\lambda_{k+1}=\lambda_{k+1}^a$
   - Calculate $U_k$ , form the optimal control equation since $X_k$ and $\lambda_{k+1}$ are known.
   - Get $X_{k+1}$ from the state Eq.(30) using $X_k$ and $U_k$
   - Get $\hat{d}(X_k)$ as the output of the online neural network
   - Input $X_{k+1}$ to the critic network to get $\lambda_{k+2}$
   - Using $X_{k+1}$ and $\lambda_{k+2}$ , calculate $\lambda_{k+1}^t$ from the updated costate Eq.(32)
2. Train the critic network for $X_k$; the output being corresponding $\lambda_{k+1}^t$ .
3. Update time step $k$ to $k+1$
4. Continue steps 1-3.

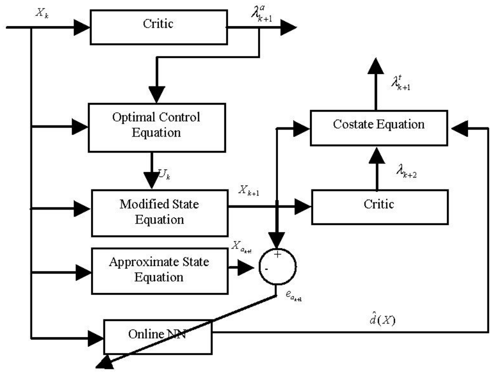

Figure1: Dynamic Reoptimization of SNAC Controller

## V. Simulation Study: A MEMS Actuator

### *A. Problem Description and Optimality Conditions*

The problem considered in this study is a MEMS device, namely electrostatic actuator [7]. This problem was chosen to prove that this technique is applicable for complex engineering systems of practical significance.

The governing equations are given by

$$\dot{Q} - \frac{1}{R}\left(V_{in} - \frac{Qg}{\varepsilon A}\right) = 0$$
$$m\ddot{g} + b\dot{g} + k(g - g_0) + \frac{Q^2}{2\varepsilon A} = 0 \tag{20}$$

where $Q$ denotes the charge, $g$ the gap between the plate and the base ($g_0 = 1\mu m$), and $\dot{g}$ represents the rate of change of the gap when the plate moves. $V_{in}$ is the input voltage that is used to move the plate to the desired position. The system parameters for the MEMS actuator are available in [7]. Defining the state variable $Z = [z_1\ z_2\ z_3]^T = [Q\ g\ \dot{g}]^T$, and substituting system parameters, Eq.(20) can be written as

$$\begin{aligned} \dot{z}_1 &= 1000V_{in} - 10z_1z_2 \\ \dot{z}_2 &= z_3 \\ \dot{z}_3 &= -\left(z_1^2/200\right) - \left(z_3/2\right) - z_2 + 1 \end{aligned} \tag{21}$$

The function of the control input in this problem is to bring the plate to some desired position, i.e. the gap $g$ has to be maintained at some desired value. The desired value of the gap was set as $0.5\,\mu m$. An optimal controller was designed to drive the plate to the desired value. At the equilibrium point, $z_2 = 0.5,\ \dot{Z} = 0$. Solving Eq.(21) for $z_1, z_3$ and $V_{in}$ the values of the states at the equilibrium (operating) point were obtained as $Z_0 = [10\quad 0.5\quad 0]^T$ and the associated steady state controller value was given as $V_{in_0} = 0.05$. Next, the deviated state was defined as $X = [x_1\ x_2\ x_3]^T \equiv Z - Z_0$ and the deviated control as $u \equiv V_{in} - V_{in0}$. In terms of these variables, the error dynamics of the system were

$$\begin{aligned} \dot{x}_1 &= -5x_1 - 100x_2 - 10x_1x_2 + 1000u \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= -0.1x_1 - 0.5x_3 - x_2 - 0.005x_1^2 \end{aligned} \tag{22}$$

Next, an optimal regulator problem was formulated to drive $X \to 0$ with a cost function, J as

$$J = \frac{1}{2}\int_0^\infty \left(X^T Q_w X + R_w u^2\right) dt \tag{23}$$

with $Q_w \geq 0$ and $R_w > 0$ the weighting matrices for state and control respectively. The state equation and cost function were discretized as follows

$$\begin{bmatrix} x_{1_{k+1}} \\ x_{2_{k+1}} \\ x_{3_{k+1}} \end{bmatrix} = X_k + \Delta t \begin{bmatrix} -5x_{1_k} - 100x_{2_k} - 10x_{1_k}x_{2_k} + 1000u_k \\ x_{3_k} \\ -0.1x_{1_k} - 0.5x_{3_k} - x_{2_k} - 0.005x_{1_k}^2 \end{bmatrix} \tag{24}$$

$$J = \sum_{k=1}^{N\to\infty} \frac{1}{2}\left(X_k^T Q_W X_k + R_W u_k^2\right)\Delta t \tag{25}$$

Next, using $\Psi_k = (X_k^T Q_W X_k + R_W u_k^2)\Delta t/2$ in Eqs.(7) and (8), the optimal control and costate equations were obtained as follows

$$u_k = -R_w^{-1}\frac{\lambda_{1_{k+1}}}{R} \tag{26}$$

$$\lambda_k = \Delta t\, Q_w X_k + \left[\frac{\partial F_k}{\partial X_k}\right]^T \lambda_{k+1} \tag{27}$$

In Eq. (27) $F_k$ represents the expression on the right hand side of Eq. (24). For this problem we chose $\Delta t = 0.001$, $Q_w = I_3$ and $R_w = 1$. In the SNAC synthesis, we chose three sub-networks each having a 6-1 structure for the critic network. In each network, we selected hyperbolic tangent functions and a linear function as activation functions for the hidden layer and output layer respectively. The SNAC based control law was able to make the actuator plate optimally track the desired reference.

*B. Unmatched Uncertainty*

In order to test the reoptimization scheme, the value of permittivity $\varepsilon$ was changed from $1C^2/N\,\mu m^2$ to $0.5\,C^2/N\,\mu m^2$. This change was assumed to be an unknown uncertainty that would introduce additional nonlinearities to the nominal model represented by Eq. (21). The aim of the reoptimization scheme was then to reoptimize the existing optimal SNAC controller (nominal controller) to make the actuator plate optimally track the desired reference position. The system model now became

$$\begin{aligned} \dot{z}_1 &= 1000V_{in} - 20z_1z_2 \\ \dot{z}_2 &= z_3 \\ \dot{z}_3 &= -\left(z_1^2/100\right) - \left(z_3/2\right) - z_2 + 1 \end{aligned} \tag{28}$$

It is assumed that only the nominal model Eq. (21) is known and Eq. (28) is unknown. Eq. (28) can be expressed in terms of the known model with the uncertainty lumped up in each state equation as shown in Eq. (29).

$$\begin{aligned} \dot{z}_1 &= 1000V_{in} - 10z_1z_2 + d_1(Z) \\ \dot{z}_2 &= z_3 + d_2(Z) \\ \dot{z}_3 &= -\left(z_1^2/100\right) - \left(z_3/2\right) - z_2 + 1 + d_3(Z) \end{aligned} \tag{29}$$

where the unknown terms $d_1 \equiv -10z_1z_2,\ d_2 \equiv 0,\ d_3 \equiv -\left(z_1^2/100\right)$. At the equilibrium point, $z_2 = 0.5,\ \dot{Z} = 0$. Solving Eq.(29) for $z_1, z_3$ and $V_{in}$, the values of the states at equilibrium are

$$\begin{aligned} \overline{z}_2 &= 0.5 \\ \overline{z}_3 &= d_2 \\ \overline{z}_1 &= \left(\left(-(\overline{z}_3/2) - \overline{z}_2 + 1 + d_3\right)200\right)^{1/2} \end{aligned} \tag{30}$$

and the associated steady state control value is

$$\overline{V}_{in} = \left(10z_1z_2 - d_1\right)/1000 \tag{31}$$

In order to approximate the unknown nonlinear terms in the state equations we introduce a virtual plant as explained in Section 3. The structure of the virtual plant used in this problem was

$$\begin{aligned} \dot{z}_{1_a} &= 1000V_{in} - 10z_1z_2 + \hat{d}_1(Z) + z_1 - z_{1_a} \\ \dot{z}_{2_a} &= z_3 + \hat{d}_2(Z) + z_2 - z_{2_a} \\ \dot{z}_{3_a} &= -\left(z_1^2/100\right) - \left(z_3/2\right) - z_2 + 1 + \hat{d}_3(Z) + z_3 - z_{3_a} \end{aligned} \tag{32}$$

In Eq. (32) $\hat{d}_i$ represents the output of the online neural network that approximates the uncertainty in the $i^{th}$ state equation of the system.

Basis function neural networks were used in this study for approximating the unmodeled dynamics. Vectors $C_i$, $i=1,2,3$ which have a structure $C_i=[1\ \ z_i]^T$ were created. The vector of basis functions $\Phi$ was composed of all possible products of elements of $C_i's$. By using kronecker products to represent the neuron interactions, $\Phi$ was composed as $\Phi = kron(kron(C_1,C_2),C_3)$ where $kron(*,*)$ represents the kronecker product. During simulation $\hat{d}_i, i=1,2,3$ were used in Eqs. (30) and (31) to replace $d_i, i=1,2,3$.

The discretized error equations are

$$\begin{bmatrix} x_{1_{k+1}} \\ x_{2_{k+1}} \\ x_{3_{k+1}} \end{bmatrix} = X_k + \Delta t \begin{bmatrix} 1000(\overline{V}_{in}+u_k)-10(\overline{z}_1+x_{1_k})(\overline{z}_2+x_{2_k})+d^e_{1_k}(X_k) \\ (\overline{z}_3+x_{3_k})+d^e_{2_k}(X_k) \\ -\left((\overline{z}_1+x_{1_k})^2/200\right)-\left((\overline{z}_3+x_{3_k})/2\right)-(\overline{z}_2+x_{2_k})+d^e_{3_k}(X_k) \end{bmatrix} \quad (33)$$

where $d_i^e, i=1,2,3$ represent the uncertainties that appears in the error equation because of the nonlinearities in the plant equations. A virtual set of error equations were defined that were used to approximate the uncertainties in the error equation, Eq. (33). These virtual set of error equations were given by

$$\begin{bmatrix} x^a_{1\,k+1} \\ x^a_{2\,k+1} \\ x^a_{3\,k+1} \end{bmatrix} = X^a_k + \Delta t \begin{bmatrix} 1000(\overline{V}_{in}+u_k)-10(\overline{z}_1+x_{1_k}) \ \ldots \\ (\overline{z}_3+x_{3_k})+\hat{d}^e_{2_k}(X_k)+x_{2_k}-x^a_{2\,k} \\ -\left((\overline{z}_1+x_{1_k})^2/200\right)-\left((\overline{z}_3+x_{3_k})/2\right) \ \ldots \\ \ldots \ (\overline{z}_2+x_{2_k})+\hat{d}^e_{1_k}(X_k)+x_{1_k}-x^a_{1\,k} \\ \ldots \ -(\overline{z}_2+x_{2_k})+\hat{d}^e_{3_k}(X_k)+x_{3_k}-x^a_{3\,k} \end{bmatrix} \quad (34)$$

In Eq. (34), $\hat{d}_i^e, i=1,2,3$ represent approximations of the unknown terms in the error equations Eq. (33). The terms denoted by $\hat{d}_i^e$ are outputs of online neural networks which have the same structure as the networks described earlier (to approximate uncertainties in the plant equation). Vectors $C_i^e$, $i=1,2,3$ which have a structure $C_i^e=[1\ \ x_i]^T$ were created. The vector of basis functions $\Phi^e$ was $\Phi^e = kron(kron(C_1^e,C_2^e),C_3^e)$. The outputs of the online neural networks that approximate uncertainties in the error equation were expressed as $\hat{d}^e_{i_k} = \hat{W}^{eT}_{i_k}\Phi^e_i(X_k)$. These outputs were used to replace the uncertainties denoted by $d_i^e$ in Eq. (33). Next, using $\Psi_k = (X_k^T\, Q_W\, X_k + R_W\, u_k^2)\,\Delta t/2$ the optimal control and costate equations was obtained. The expression for optimal control is the same as Eq. (26). The costate equation now accommodates the change in the state equation.

During each iteration of the simulation, the critic network mentioned in subsection V. (A) was updated. This update was based on the new costate equation. The online training was carried out using the error vector ($X_k$) at that instant as the input and the new target costate ($\lambda_{k+1}$) as the output. This training was performed online made use of only one epoch of the Levenberg-Marquardt back-propagation scheme available in the Neural Network Toolbox of MATLAB v 7.0 for training/ reoptimizing the network at each time step.

## VI. Results

Simulation studies were performed for twenty seconds. The actuator position has been plotted in Figure 2. It details the nominal state trajectory (state trajectory of the plant if uncertainties were not accounted for), state trajectory affected by online reoptimization of the critic network and optimal state trajectory if the uncertainty were known and accounted for when the optimal controller was designed. The last mentioned state trajectory was simulated for comparison purposes. It can be seen in Figure 2 how well the reoptimized controller drives the actuator to the desired position of $0.5\,\mu m$. Figure 3 illustrates nominal control, reoptimized control, and optimal control if uncertainties were known and modeled in the system equations *apriori*. The error state uncertainty approximations carried out by the corresponding networks for the three error state equations are shown in Figure 4.

## VII. Conclusions

There has not been any concerted effort to dynamically re-optimize existing controllers in the presence of uncertainties. In this study, a scheme to re-optimize a pre-designed optimal SNAC controller for control affine systems in the presence of unmodeled/parametric uncertainties has been developed. This methodology has been simulated and results have been shown for the tracking control of a MEMS actuator. The results shown are promising. A set of online neural networks capture plant uncertainty. This information about the nonlinearity/uncertainty is used to update the costate equation in the SNAC architecture to further train /adapt the critic network to optimize itself in presence of the unmodeled term. This method is unique in that unmatched uncertainties and nonlinearities can be compensated for as is shown in the example illustrated in Section 5.

## Appendix: Proof of convergence of SNAC

Let $g(.)$ represent the function approximated by the critic neural network.

$$\lambda_{k+1} = g(x_k) \quad (1a)$$

The control is then given by,

$$u_k = -R^{-1}b^T g(x_k) \quad (2a)$$

Using Eq(2a) in state propagation equation we get

$$x_{k+1} = Ax_k - bR^{-1}b^T g(x_k) \quad (3a)$$

Let $W_i$ and $W_o$ represent the weight matrices of the input and output layers of a two layer neural network respectively. Let $\varphi(.)$ be the activation function of the hidden layer. $g(.)$ can be expressed as $g(x) = W_o^T\varphi\left(W_i^T x\right)$.

Let $g_{n+1}(x_k)$ represent the desired output of the network in (n+1)th iteration. $g_{n+1}(x_k)$ can be expressed as

$$g_{n+1}\left(x(k)\right)=Qx_n\left(k+1\right)+A^T g_n\left(x_n\left(k+1\right)\right) \tag{4a}$$

To prove that SNAC training converges, it is sufficient to prove that $g_n(.)$ is a contraction mapping.

From Eq. (4a) we can write

$$g_n\left(x(k)\right)=Qx_{n-1}\left(k+1\right)+A^T g_{n-1}\left(x_{n-1}\left(k+1\right)\right) \tag{5a}$$

Subtracting Eq. (5a) from Eq. (4a) we get

$$\begin{aligned}\left(g_{n+1}\left(x(k)\right)-g_n\left(x(k)\right)\right)&=Q\left(x_n\left(k+1\right)-x_{n-1}\left(k+1\right)\right)+\dots\\ \dots\ &A^T\left(g_n\left(x_n\left(k+1\right)\right)-g_{n-1}\left(x_{n-1}\left(k+1\right)\right)\right)\end{aligned} \tag{6a}$$

Taking norm on both sides of Eq (6a) yields

$$\begin{aligned}\left\|\left(g_{n+1}\left(x(k)\right)-g_n\left(x(k)\right)\right)\right\|&\le\left\|Q\left(x_n\left(k+1\right)-x_{n-1}\left(k+1\right)\right)\right\|+\dots\\ \dots\ &\left\|A^T\left(g_n\left(x_n\left(k+1\right)\right)-g_{n-1}\left(x_{n-1}\left(k+1\right)\right)\right)\right\|\end{aligned} \tag{7a}$$

On substituting the state propagation equation, Eq. (3a) in the first term on the right hand side of Eq. (7a), the first term gets modified to $\left\|QbR^{-1}b^T\left(g_n\left(x_k\right)-g_{n-1}\left(x_k\right)\right)\right\|$.

Now, consider the second term on the right hand side of Eq. (7a). The term, $\left\|\left(g_n\left(x_n\left(k+1\right)\right)-g_{n-1}\left(x_{n-1}\left(k+1\right)\right)\right)\right\|$

$$\le\left\|g_{n-1}\left(x_n\left(k+1\right)\right)-g_{n-1}\left(x_{n-1}\left(k+1\right)\right)\right\|+\left\|g_n\left(x_n\left(k+1\right)\right)-g_{n-1}\left(x_n\left(k+1\right)\right)\right\| \tag{8a}$$

Consider the first term in the above inequality, $\left\|g_{n-1}\left(x_n\left(k+1\right)\right)-g_{n-1}\left(x_{n-1}\left(k+1\right)\right)\right\|$

$$\le\left\|W^T_{o_{n-1}}\right\|\left\|\left(\varphi\left(W^T_{i_{n-1}}x_n\left(k+1\right)\right)-\varphi\left(W^T_{i_{n-1}}x_{n-1}\left(k+1\right)\right)\right)\right\| \tag{9a}$$

As $\varphi(.)$ is a Lipschitz function, $\varphi(x)-\varphi(y)\le\eta(x-y)$ where, $\eta$ is any positive number. Using this relation, Eq. (9a)

$$\le\left\|W^T_{o_{n-1}}\right\|\left\|\eta\right\|\left\|W^T_{i_{n-1}}\right\|\left\|bR^{-1}b^T\right\|\left\|\left(g_n\left(x(k)\right)-g_{n-1}\left(x(k)\right)\right)\right\|$$

The second term in Eq. (8a) represents the change in the network output due to change in weights. Let the change of weights in the input layer be $\Delta w_i$ and that in output layer be $\Delta w_o$. It can be seen that, $\left\|g_n\left(x_n\left(k+1\right)\right)-g_{n-1}\left(x_n\left(k+1\right)\right)\right\|$

$$=\left\|\left(W^T_{o_n}\varphi\left(W^T_{i_n}x_n\left(k+1\right)\right)-W^T_{o_{n-1}}\varphi\left(W^T_{i_{n-1}}x_n\left(k+1\right)\right)\right)\right\|$$

$$\le\left\|W^T_{o_{n-1}}\right\|\left\|\eta\right\|\left\|\left(\Delta w_{i_{n-1}}\right)^T\right\|\left\|\left(x_n\left(k+1\right)\right)\right\|+\left\|\left(\Delta w_{o_{n-1}}\right)^T\right\| \tag{10a}$$

Based on backpropagation algorithm, we get

$$\left\|\left(\Delta w_{o_{n-1}}\right)^T\right\|\le\left\|k_1\right\|\left\|\left(g_n\left(x(k)\right)-g_{n-1}\left(x(k)\right)\right)\right\|$$

$$\left\|\left(\Delta w_{i_{n-1}}\right)^T\right\|\le\left\|k_2\right\|\left\|\left(g_n\left(x(k)\right)-g_{n-1}\left(x(k)\right)\right)\right\|$$

where $k_1=\alpha\varphi'(.)y_i(n)$ and $k_2=\alpha W^T{}_{0_{n-1}}$

Therefore, the second term in Eq. (8a),

$$\begin{aligned}\left\|g_n\left(x_n\left(k+1\right)\right)-g_{n-1}\left(x_n\left(k+1\right)\right)\right\|&\le\left\|W^T_{o_{n-1}}\right\|\left\|\eta\right\|\left\|k_2\right\|\left\|\left(g_n\left(x(k)\right)-g_{n-1}\left(x(k)\right)\right)\right\|\dots\\ \dots+&\left\|k_1\right\|\left\|\left(g_n\left(x(k)\right)-g_{n-1}\left(x(k)\right)\right)\right\|\end{aligned} \tag{11a}$$

Eq. (7a) can be rewritten as

$$\begin{aligned}&\left\|\left(g_n\left(x_n(k+1)\right)-g_{n-1}\left(x_{n-1}(k+1)\right)\right)\right\|\le\left\|W^T_{o_{n-1}}\right\|\left\|\eta\right\|\left\|k_2\right\|\left\|\left(g_n(x(k))-g_{n-1}(x(k))\right)\right\|\dots\\ &\dots+\left\|k_1\right\|\left\|\left(g_n(x(k))-g_{n-1}(x(k))\right)\right\|+\left\|W^T_{o_{n-1}}\right\|\left\|\eta\right\|\left\|W^T_{i_{n-1}}\right\|\left\|bR^{-1}b^T\right\|\dots\left\|\left(g_n(x(k))-g_{n-1}(x(k))\right)\right\|\\ &=\left\|\rho\right\|\left\|\left(g_n\left(x(k)\right)-g_{n-1}\left(x(k)\right)\right)\right\| \qquad ,\rho<1\end{aligned} \tag{12a}$$

Now, we can write

$$\begin{aligned}&\left\|\left(g_{n+1}\left(x(k)\right)-g_n\left(x(k)\right)\right)\right\|\le\left\|QbR^{-1}b^T\left(g_n\left(x_k\right)-g_{n-1}\left(x_k\right)\right)\right\|+\dots\\ &\dots\ \left\|A^T\right\|\left\|\rho\right\|\left\|\left(g_n\left(x(k)\right)-g_{n-1}\left(x(k)\right)\right)\right\|\le\Psi\left\|\left(g_n\left(x_k\right)-g_{n-1}\left(x_k\right)\right)\right\|\end{aligned} \tag{13a}$$

The above is a contraction mapping if $\left\|\Psi\right\|<1\ \ \forall n$. This condition can be met by selecting appropriate values of $R$ and $\alpha$.

## References

[1] Bryson, A. E. and Ho, Y. C., "Applied Optimal Control", *Taylor and Francis, 1975.*

[2] Werbos, P. J., "Approximate Dynamic Programming for Real-time Control and Neural Modeling". In White D.A., & Sofge D.A (Eds.), *Handbook of Intelligent Control, Multiscience Press,* 1992.

[3] Padhi, R., "Optimal Control of Distributed Parameter Systems Using Adaptive Critic Neural Networks", 2001, *Ph.D. Dissertation,* University of Missouri Rolla.

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

[5] Balakrishnan, S. N., and Biega, V., Adaptive-Critic Based Neural Networks for Aircraft Optimal Control", *Journal of Guidance, Control and Dynamics, Vol. 19, No. 4, July- Aug. 1996, pp. 893-898.*

[6] Padhi, R., Unnikrishnan, N and Balakrishnan, S. N., 'Model Following Robust Neuro-Adaptive Control Design for Non-square, Non-affine Systems', *Proceedings of the 2004 ASME International Mechanical Engineering Congress and R&D Expo,* November 13-19, Anaheim, California, USA

[7] Senturia., S. D., "Microsystem Design", *Kluwer Academic Publishers, 2001.*

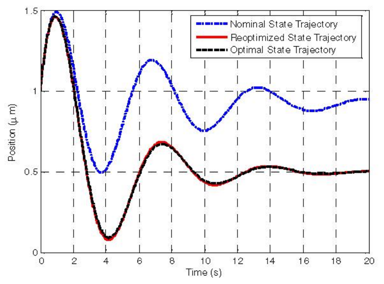

Figure 2: Actuator Position vs. Time

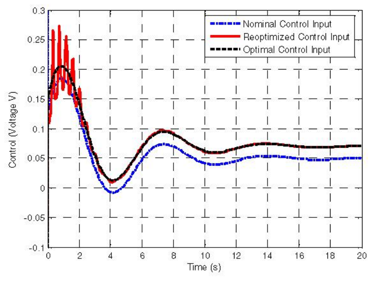

Figure 3: Control Voltage vs. Time

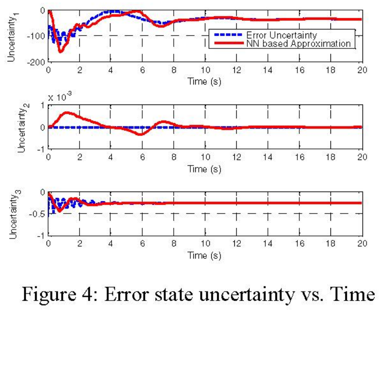

Figure 4: Error state uncertainty vs. Time