A Single Network Approximate Dynamic Programming based Constrained Optimal Controller for Nonlinear Systems with Uncertainties

Jie Ding, S.N. Balakrishnan

Abstract—Approximate dynamic programming formulation implemented with an Adaptive Critic (AC) based neural network (NN) structure has evolved as a powerful alternative technique that eliminates the need for excessive computations and storage requirements needed for solving the Hamilton-Jacobi-Bellman (HJB) equations. A typical AC structure consists of two interacting NNs. In this paper, a novel architecture, called the Cost Function Based Single Network Adaptive Critic (J-SNAC) is used to solve control-constrained optimal control problems. Only one network is used that captures the mapping between states and the cost function. This approach is applicable to a wide class of nonlinear systems where the optimal control (stationary) equation can be explicitly expressed in terms of the state and costate variables. A non-quadratic cost function is used that incorporates the control uncertainties. Necessary equations for optimal control are derived and an algorithm to solve the constrained-control problem with J-SNAC is developed. Benchmark nonlinear systems are used to illustrate the working of the proposed technique. Extensions to optimal control-constrained problems in the presence of uncertainties are also considered.

Keywords: Approximate Dynamic Programming, Constrained Optimal Control, Nonlinear Control, Adaptive Critic, Cost Function Based Single Network Adaptive Critic, J-SNAC Architecture

I. INTRODUCTION

Feedback control is the preferred solution for many systems because of its beneficial properties like robustness with respect to noise and modeling uncertainties. It is well-known that a dynamic programming formulation offers the most comprehensive solution approach to nonlinear optimal control in a state feedback form (Lewis, 1992, Bryson et al., 1975). However, solving the associated Hamilton-Jacobi-Bellman (HJB) equation demands a large (rather infeasible) number of computations and storage space dedicated to this purpose. An innovative idea was proposed in (Werbos, 1992) to get around this numerical complexity by using an ADP formulation. The solution to the ADP formulation is obtained through a dual NN approach called the Adaptive Critic (AC). In one version of the AC approach, called the Heuristic Dynamic Programming (HDP), 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 the cost function to be minimized. Adaptive critic formulations can be found in many papers; some researchers have used ADP formulations to solve problems with finite state spaces in applications to behavioral and computer sciences and operations research and robotics (Barto, 1991, 2004, Powell, 2004, Bertsekas, 1996). Note that adaptive critics are also reinforcement learning designs (Barto, 1991, 2004). Their formulations employ primarily cost function based adaptive critics that we consider in this paper. There are also many papers in the literature that use system science principles to formulate their problems with applications to real-time feedback control of dynamic systems. In recent years, many researchers have paid more attention on ADP in order to obtain approximate solutions of the HJB equation (Al-Tamimi. et al. 2008, Balakrishnan et al. 2008, Li & Si, 2007, Werbos 2007). In (Padhi & Balakrishnan, 2004) a Single Network Adaptive Critic (SNAC) was implemented on a distributed parameter model for beaver population. Their SNAC used a critic network that mapped one-step ahead costates (at k+1) with states (at k) as inputs. Optimal control at k was computed from the critic outputs without the use of a separate control network. A real-life Micro-Electro-Mechanical-System (MEMS) problem was solved in (Padhi et al. 2004). In (Yadav et al. 2006), a suboptimal neurocontroller was obtained with SNAC to solve a heat transfer problem, and an online robust controller was developed to account for unmodeled dynamics and parametric uncertainties, where a weight update rule was presented that guarantees boundedness of the weights and eliminates the need for persistence of excitation (PE) condition to be satisfied. In (Chen et al. 2009), an aircraft control problem under nominal and damaged conditions was solved, and the convergence of the SNAC training was shown by reducing it to solving a set of nonlinear algebraic equations in weights. In (Ma et al., 2008) the SNAC based controllers for vibration isolation applications were developed.

During the last few years, several methods for solving constrained control problems are found by Saberi et al. (1996), and Bernstein (1995). Successful NN controllers which are not based on optimal techniques have been presented in Lewis et al. (1999). S. E. Lyshevski showed the method to formulate constrained input in terms of a non-quadratic performance index in (S.E., Lyshevski, 1996, 1998). In (Adhyaru & Kar, 2008), a HJB equation based constrained optimal control algorithm is proposed for a bilinear system. In (Cheng and Lewis, 2006), fixed-final time
constrained input optimal control laws using neural networks to solve Hamilton-Jacobi-Bellman equations for
general affine in the input nonlinear systems are proposed.

Rest of the paper is organized as follows: In Section II, the
ADP equations are presented and in Section III, J-SNAC
approach which is a non-quadratic cost function is presented. An
online updated neural network is discussed in Section IV.
Conversion of solving a tracking problem into a regulator
problem is presented in Section V. Numerical results are
presented in section VI.

II. APPROXIMATE DYNAMIC PROGRAMMING

In this section, the principles of approximate (discrete)
dynamic programming, which both the AC and the J-SNAC
approaches rely upon, are described. An interested reader can
find more details about the derivations in (Balakrishnan et al.,
1996; Werbos, 1992). Note that a prime requirement to apply
the AC or the J-SNAC is to formulate the problem in
discrete-time. The control designer has the freedom to use
any appropriate discretization scheme. For example, one can
use the Euler approximation for the state equation and
Trapezoidal approximation for the cost function (Gupta,
1995). In a discrete-time formulation, we want to find an
admissible control \( U_k \) which causes the system given by

\[
X_{k+1} = F(X_k, U_k)
\]

(1)
to follow a trajectory from an initial point \( X_0 \) to a
final desired point \( X_N \) while minimizing a desired cost
function \( J \) given by

\[
J = \sum_{k=0}^{N-1} \Psi_k(X_k, U_k)
\]

(2)
where the subscript \( k \) denotes the time step. \( X_i \) and \( U_i \)
represent an \( n \times 1 \) state vector and an \( m \times 1 \) control vector
respectively, at time step \( k \). The functions \( F_i \) and \( \Psi_i \)
are assumed to be differentiable with respect to both \( X_i \) and \( U_i \).
Moreover, \( \Psi_k \) is assumed to be convex (e.g. a quadratic
function in \( X_i \) and \( U_i \)). One can notice that when \( N \to \infty \),
this cost function leads to a regulator (infinite time) problem. The
steps in obtaining optimal control are now described. The
cost function Eq.(2) is rewritten to start from time step \( k \)

\[
J_k = \Psi_k(X_k, U_k)
\]

(3)
The cost, \( J_k \), can be split into

\[
J_k = \Psi_k + J_{k+1}
\]

(4)
where \( \Psi_k \) and \( J_{k+1} = \sum_{k=k+1}^{N-1} \Psi_k \) represent the ‘utility
function’ at step \( k \) and the cost-to-go from time step \( k+1 \)
to \( N \), respectively. The \( n \times 1 \) costate vector at time step \( k \) is

\[
\lambda_k = \frac{\partial J_k}{\partial X_k}
\]

(5)
The necessary condition for optimality is given by

\[
\frac{\partial J_k}{\partial U_k} = 0
\]

(6)
Equation (6) can be further expanded as

\[
\frac{\partial J_k}{\partial U_k} = \frac{\partial \Psi_k}{\partial U_k} + \frac{\partial J_{k+1}}{\partial U_k} = \frac{\partial \Psi_k}{\partial U_k} + \left( \frac{\partial \lambda_{k+1}}{\partial U_k} \right)^T \frac{\partial \lambda_{k+1}}{\partial X_{k+1}}
\]

(7)

The optimal control equation can, therefore, be written as

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

(8)
The costate equation is derived in the following way

\[
\lambda_k = \frac{\partial \Psi_k}{\partial X_k} + \left( \frac{\partial \lambda_{k+1}}{\partial X_k} \right)^T \frac{\partial \lambda_{k+1}}{\partial X_{k+1}} + \frac{\partial \Psi_k}{\partial U_k} + \frac{\partial \lambda_{k+1}}{\partial U_k}
\]

(9)

Equations (1), (8) and (9) have to be solved simultaneously, along with appropriate boundary conditions
for the synthesis of optimal control. For the regulator
problems, the boundary conditions usually take the form: \( X_0 \) is fixed and \( \lambda_N \to 0 \) as \( N \to \infty \). If the state equation and cost
function are such that one can obtain explicit solution for the
control variable in terms of the state and the cost variables
from Eq.(8), J-SNAC is applicable.

III. J-SNAC SYNTHESIS WITH NON-QUADRATIC COST

In this section, the newly developed cost function based
single network adaptive critic (J-SNAC) technique is
discussed in detail. In the J-SNAC design, the critic network
captures the functional relationship between state \( X_k \) and cost
\( J_k \). However, the J-SNAC method is applicable only for problems where the optimal control Eq.(8) is explicitly
expressible for control variable \( U_k \) in terms of the state
variable \( X_k \) and cost \( J_{k-1} \), where such a restriction is not there
for the AC technique.

3.1 HJB Equation with Constraints on the Control Input

For the nonlinear system given by

\[
\dot{x} = f(x) + Bu, \quad x(t_0) = x_0
\]

(10)
where \( x \in \mathbb{R}^n \) and \( u \in \mathbb{R}^m \) are the state and control
vectors, \( f(\cdot): \mathbb{R}^n \to \mathbb{R}^n \) is the smooth mapping and \( B: \mathbb{R}^{n \times m} \).
The following cost function is commonly used in the design
of constrained controllers (Lyshesvski, 2001):

\[
J = \int_{t_0}^{t_f} \left[ \frac{1}{2} x^T Q x + \int (\phi^{-1}(u))^T R du \right] dt
\]

(11)
where \( Q \in \mathbb{R}^{n \times n} \) and \( R \in \mathbb{R}^{m \times m} \) are the diagonal weighting
matrices and \( \phi(\cdot) \) is the bounded, integrable, one-to-one,
real-analytic globally Lipschitz continuous function \( \phi: \mathbb{R}^m \to \mathbb{R}^m \).
The Hamilton-Jacobi functional equation for system (10) is

\[
\frac{\partial \psi}{\partial t} = \min_{\psi \in \psi(x)} \left[ \frac{1}{2} x^T Q x + \int (\phi^{-1}(u))^T R du + \frac{\partial \psi}{\partial \phi}(f(x) + Bu) \right]
\]

(12)
where \( V(\cdot) \) is the positive-definite, continuously differen
tiable (minimum-cost) return function, \( V(x_0) = \inf_{u \in U} f(x_0, u) > 0 \). By applying the cost function (11), control
law can be designed as

\[
u = -\phi \left( R^{-1}B^T \frac{\partial \psi}{\partial x} \right)
\]

(13)

In discrete-time form, the system and cost function are

\[
X_{k+1} = f(X_k) + BU_k
\]

(14)

\[
J = \sum_{k=0}^{N-1} \Psi_k
\]

(15)
where the utility function \( \Psi_k \) is given by

\[
\Psi_k = \left( \frac{1}{2} X_k^T Q X_k + M(U_k) \right) \Delta t
\]

(16)
where \( M(U_k) = \int f^T (\phi^{-1}(\phi)(\phi^{-1}(v))) R du \). By applying (8), we obtain

\[
\phi^{-1}(U_k) \Delta t R_w + \left( \frac{\partial \lambda_{k+1}}{\partial U_k} \right)^T \frac{\partial \lambda_{k+1}}{\partial X_{k+1}} = 0
\]

(17)
The constrained control is obtained as

\[
U_k = -\phi \left( (\Delta t R_w)^{-1} B^T \frac{\partial \lambda_{k+1}}{\partial X_{k+1}} \right)
\]
where it is assumed that \((\frac{\partial X_{k+1}}{\partial X_k})^{-1}\) exists. The costate equation can be obtained by applying Eq.(9) as

\[
\dot{\lambda}_k = \Delta Q_{W} X_k + [\partial X_{k+1}/\partial X_k]^{T} \Delta \lambda_{k+1}
\]  

(19)

3.2 Neural Network Training

In the J-SNAC synthesis with non-quadratic cost approach, the steps for the training the critic network, which captures the relationship between \(X_k\) and \(J_k\), are as follows (Fig. 1):

a. Input \(X_k\) to the critic network to obtain \(J_k = J_k^*\).

b. Calculate \(\lambda_k = \partial J_k / \partial X_k\), and calculate \(\lambda_{k+1}\) by Eq.(19).

c. Calculate \(U_k\), form the optimal control equation (18).

d. Use \(X_k\) and \(U_k\) to get \(X_{k+1}\) from the state Eq. (1).

e. Input \(X_{k+1}\) to the critic network to get \(J_{k+1}\).

f. Use \(X_k\), \(U_k\) and \(J_{k+1}\), to calculate \(J_k^*\) with Eq.(4).

g. Train the critic network by solving Eq.(A.7) for network weights. The details are given in Appendix.

h. Set \(k = k + 1\) and go to first step.

IV. DYNAMIC RE-OPTIMIZATION OF J-SNAC CONTROLLER

In this section, we consider the plant dynamics with parametric uncertainties or unmodeled nonlinearities. We discuss the dynamic re-optimization of the J-SNAC controller that is used to capture the uncertainty but are not considered in the system model used for controller design. This is achieved with the help of a virtual plant that is similar to the actual plant. The uncertainty approximation is achieved using an online neural network. Consider a general nonlinear system is given as

\[
X_{k+1} = f(X_k) + BU_k
\]  

(20)

where \(X \in R^n\) is the state vector and \(U \in R^m\) is the control vector. Let the actual plant have the structure

\[
X_{k+1} = f(X_k) + BU_k + d(X_k)
\]  

(21)

where the controller \(U\) will have to be re-optimized to optimize the plant performance with the unmodeled dynamics \(d(X_k)\) present. Since the term \(d(X_k)\) 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. Let \(X_r\) represent the vector of states of the virtual plant. The dynamics of this virtual plant is governed by

\[
x_{k+1} = f(x_k) + BU_k + \hat{d}(x_k) + K(x_k - x_{k+1})
\]  

(22)

where \(K > 0\) is a design parameter. We assume that we have all the actual plant states, \(X\), available for measurement at every step. The term \(\hat{d}(X_k)\) is the neural network approximation of the actual plant. Subtracting Eq.(22) from (21), by defining \(E_{x_k} \equiv X_k - x_{k+1}\), we obtain

\[
x_{k+1} - x_{k+1} = \hat{d}(X_k) - d(X_k) + K(x_k - x_{k+1})\quad\text{or}\quad E_{x_{k+1}} = \hat{d}(X_k) - d(X_k) - KE_{x_k}
\]

It can be seen that as \(\hat{d}(X_k) - d(X_k)\) approaches zero, the expression of \(E_{x}\) is exponentially stable, i.e. \(E_{x_k} \to 0\) as \(t \to \infty\). The J-SNAC dynamic re-optimization scheme is shown in Fig. 2.

Defining \(d(X) \equiv [d_1(X) \cdots d_n(X)]^T\), 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 ‘\(n\)’ NNs (one for each channel of the unmodeled dynamics) so as allow for simpler development and analysis. This is presented in Fig.3. Separating all channels, the state equations become

\[
x_{i_{k+1}} = f_i(X_k) + b_iU_k + d_i(X_k)
\]  

(23)

\[
x_{i_{k+1}} = f_i(X_k) + b_iU_k + \hat{d}_i(X_k) + e_{i_{k+1}}\]

(24)

where \(e_{i_{k+1}} \equiv x_i - x_{i_{k+1}}\). Subtracting Eq (24) from (23), we obtain

\[
x_{i_{k+1}} - x_{i_{k+1}} = \hat{d}_i(X_k) - d_i(X_k) - KE_{i_{k+1}}
\]

(25)

Let us assume that there exists a neural network with an optimum set of weights that approximates \(d(X)\) within a certain accuracy of \(e_i\). Thus we have

\[
d_i(X_k) = W_{iW_i}(X_k) + e_i.
\]  

(26)

Also \(\hat{d}_i(X_k) = W_{i}(X_k)\), where \(W_{i}(X_k)\) is the output of the actual neural network. \(W_{i}(X_k)\) represents the actual network weights. Substituting Eq.(26) into (25), we obtain

\[
x_{i_{k+1}} - x_{i_{k+1}} = W_{iW_i}(X_k) + e_i - W_{i}(X_k)\]

(27)

\[
e_i = W_{i}(X_k) + e_i - e_{i_{k+1}}
\]  

(28)

where \(\hat{W} = W - W_{i}\), is the difference between the optimal
weights of the neural network that represents \( d_i(X_k) \) and the actual network weights. More details on the update rule can be found in (Chen et al. 2009, Padhi 2007, Nishant 2006).

V. CONVERSION OF A TRACKING PROBLEM INTO A REGULATOR PROBLEM

With the knowledge of solving the regulator problem discussed above, it is easy to solve the tracking problem by converting it into a regulator problem. In order to define an appropriate cost function, we define \( e \equiv X - X_d \), where \( x_d \) is the reference input. Equations (10) and (11) then become

\[
\dot{e} = f(e + x_d) + Bu, \quad e(t_0) = x_0 - x_{d_0}
\]

\[
J = \frac{1}{2} e^T Q e + \int (\phi^{-1}(u) - u_d)^T R du
\]

where \( u_d \) is the desired steady state control. The HJB functional equation for the converted system (29) is

\[
-\frac{\partial V}{\partial t} = \min_{u \in U} \left\{ e^T Q e + \int (\phi^{-1}(u) - u_d)^T R du + \frac{\partial V}{\partial e} \left[ f(e) + Bu \right] \right\}
\]

where \( V(\cdot) \) is the positive-definite, continuously differentiable (minimum-cost) return function, \( V(e_0) = \inf_{u \in U} \{ f(e_0, u) \} > 0 \). In discrete-time form, the system and cost function is given by

\[
E_{k+1} = f(E_k) + BU_k
\]

\[
J = \sum_{k=0}^{N-1} \frac{1}{2} E_k^T Q E_k + M(U_k) \Delta t
\]

where \( M(U_k) = \int (\phi^{-1}(v) - u) R du \). The constrained control and costate equations are obtained as

\[
U_k = \phi \left( U_{d_k} - (\Delta t R_W)^{-1} B^T \frac{\partial J_k}{\partial U_{k+1}} \right)
\]

\[
\lambda_k = \Delta t Q_W E_k + [\beta E_{k+1}/\beta E_k]^T \lambda_{k+1}
\]

VI. NUMERICAL RESULTS

In this section, the application of the developed approach to an aircraft control problem is presented.

![Fig. 4. F-16 Short-Period Dynamics](image)

6.1 Example: F-16 Short-Period Dynamics Flight Control

6.1.1 Problem Description and Optimality Conditions

The dynamic model is given by (Young et al. 2007)

\[
\dot{q} = \begin{bmatrix} -1.019 & 1 \\ 0.8223 & -1.0774 \end{bmatrix} \begin{bmatrix} q \end{bmatrix} + \begin{bmatrix} 0 \\ -0.1756 \end{bmatrix} \delta_e + f(q, \delta_e) + d(q, \delta_e)
\]

In this problem, the control objective is to adjust \( \delta_e \) so that \( \alpha \) tracks commanded reference input

\[
r(t) = 0.2 \left[ \frac{1}{1 + e^{-t}} - 0.5 \right],
\]

where \( d(q) = -1.28 \alpha^3 \quad 0 \).

The discretized equations can be written as

\[
E_{k+1} = E_k + \Delta t \left[ f(E_k + X_{d_k}) + BU_k + d \right]
\]

where \( \sigma = 0.25, C_0 = 0.1, h = 0.14, a_0 = 0 \). Note that \( d(\alpha) \) is an unmatched uncertainty and \( f(\alpha, \delta_e) \) is a matched uncertainty. By defining \( X = [x_1, x_2]^T = [\alpha, q]^T \), \( U = \delta_e, d = [d_1, d_2]^T = BF(\alpha, \delta_e) + d(\alpha) \), the system dynamics is given by

\[
\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -1.019 \\ 0.8223 \end{bmatrix} \begin{bmatrix} x_1 \end{bmatrix} + \begin{bmatrix} 0 \\ -0.1756 \end{bmatrix} \begin{bmatrix} u \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}
\]

Since we are dealing with a tracking problem, by defining \( E = X - X_d \), the optimal control problem can be formulated to drive \( E \to 0 \) with the system dynamics and the cost, \( J_c \) as

\[
\dot{E} = f(E + X_d) + BU, \quad E(t_0) = X_0 - X_{d_0}
\]

\[
J_c = \int \frac{1}{2} E^T Q_W E + \int (\phi^{-1}(U_{\text{max}}) - U_d)^T R_W du
\]

where \( X_d \) is the reference input, \( U_d \) is the desired steady state control. \( Q_W \geq 0 \) and \( R_W > 0 \) are weighting matrices for state and control respectively. \( U_{\text{max}} \) is the control constraint. In this problem, we select the constrained control \( |U| \leq U_{\text{max}} = 6 \) deg. The state equation and cost function are discretized as follows:

\[
E_{k+1} = E_k + \Delta t \left[ f(E_k + X_{d_k}) + BU_k \right]
\]

\[
J = \sum_{k=0}^{N-1} \frac{1}{2} E_k^T Q E_k + \int (\phi^{-1}(U_{\text{max}}) - U_d)^T R_W du
\]

The constrained control and costate equations can be obtained as follows:

\[
U_k = U_{\text{max}} \phi \left( U_{d_k} - (\Delta t R_W)^{-1} B^T \frac{\partial J_k}{\partial U_{k+1}} \right)
\]

\[
\lambda_k = \Delta t Q_W E_k + [\beta E_{k+1}/\beta E_k]^T \lambda_{k+1}
\]

where \( \Delta t = 0.005, Q_W = diag(1, 1) \) and \( R_W = 0.005 \). The Lipschitz continuous function \( \phi \) is selected as \( \phi(\cdot) = \text{tanh}(\cdot) \). In J-SNAC synthesis, the cost \( J_c \) is a function of the network weights \( W^k \) and states error \( E_k \), which is given by Eq.(B.2) as \( J_c = \tilde{W}^k \phi(E_k) \). In this problem, the network weights were initialized at zero. The basis functions \( \phi(E) \) are selected as \( [E_1, E_2, E_1^2, E_2^2, E_1 E_2]^T \) and \( m \), the number of sets in Eq.(A.7) is selected as 20. Note that there is only one set of point at every time step \( k \) for online training. However, in order to implement Least Squares Method given in Appendix, \( m \) different sets of point are needed at every time step \( k \). Therefore, \( m-1 \) sets of points, which are within 5% percent difference from the true point value, are generated every time. These points are different from but close to the true point.

6.1.2 Uncertainty Estimation

In this study, the uncertainty network structure is \( \tilde{d}_i = W_i^T \Phi, \quad i = 1, 2, 3 \). \( \tilde{d}_1 \) is to approximate \( d(\alpha) \), \( \tilde{d}_2 \) is to approximate \( f(\alpha, \delta_e) \). The design parameter \( K \) is selected as 10. \( W_1 \) and \( W_2 \) are both initialized as 27×1 zero vectors. The basis functions are selected as \( \Phi = \text{kronek}(c_1, c_2, c_3) \), where \( c_1 = [1 \sin(\alpha) \cos(\alpha)]^T \), \( c_2 = [1 \sin(\alpha) \cos(\alpha)]^T \), \( c_3 = [1 \tanh(\delta_e) \delta_e]^T \) and \( \text{kronek} \) denotes Kronecker product. The discretized equations can be written as

\[
E_{k+1} = E_k + \Delta t \left[ f(E_k + X_{d_k}) + BU_k + d \right]
\]
Expression for optimal control is the same as Eq.(43). The costate equation though changes to
\[ \lambda_k = \Delta t Q_{kk} E_k + \left[ \frac{\partial f_k}{\partial E_k} \right]^T \lambda_{k+1} \] (46)
where \( f_k \) represents the expression on the right hand side of Eq.(45). During each iteration of the simulation, the critic network is updated. The online training is carried out using \( E_k \) as the input and the new target cost \( f_k^t \) as the output.

### 6.1.3 Analysis of Results

Actual and reference state histories are shown in Fig. 5 and Fig. 6. It can be seen that both \( x_1(t) \) and \( x_2(t) \) track their desired histories eventually. True and estimated uncertainty histories are shown in Fig. 7 and Fig. 8. It can be seen that the estimated uncertainties nicely and quickly track the true uncertainties. Fig. 9 shows the history of the constrained control \( U \), which is within the control constraint, \( |U| \leq U_{\text{max}} = 6 \text{ deg} \).

![Fig. 5. Reference and Actual Angle of Attack (AOA) Histories](image)

![Fig. 6. Reference and Actual Pitch Rate Histories](image)

![Fig. 7. True and Estimate Uncertainty Histories – AOA](image)

VII. CONCLUSION

In this paper, an online cost function based single network adaptive critic (J-SNAC) technique has been presented to solve nonlinear constrained control problems with model uncertainties. A non-quadratic cost function is used that incorporates the control constraints. Necessary equations for optimal control are derived and an algorithm to solve the constrained-control problem with J-SNAC is developed. Extensions to optimal control-constrained problems in the presence of uncertainties are also considered. A flight control of F-16 short-period dynamics is used to illustrate the working of the proposed technique.

ACKNOWLEDGEMENT

Support for this study from NASA under Grant No: ARMD NRA NNH07ZEA001N-IRAC1 and the National Science Foundation are gratefully acknowledged. The views of the authors do not necessarily represent the views of the NSF and NASA.

APPENDIX

This appendix is derived from the convergence proof by Al-Tamimi et al. (2008). The difference is that the action network is eliminated in this study. Consider a discrete nonlinear control-affine system
\[ X_{k+1} = f(X_k) + BU_k \] (A.1)
where \( X_k \) is an \( n \times 1 \) vector, \( U_k \) is an \( m \times 1 \) vector, \( f(\cdot) \) can be a nonlinear function of the states, \( B \) is a constant \( n \times m \) matrix. \( f_k \) is a function of the network weights \( \Phi \) and states \( X_k \)
\[ f_k = \Phi^T \phi(X_k) \] (A.2)
\[ \lambda_k = \partial \left( \Phi^T \phi(X_k) \right) / \partial X_k \] (A.3)
In \( i \)th iteration, the control is computed as
\[ u_k^i = -R^{-1} B \left( \frac{\partial f_k}{\partial X_k} \right)^T \left( \frac{\partial f_k}{\partial X_k} - Q X_k \right) \] (A.4)
The cost relationship at stages \( k \) and \( k+1 \) is given by
\[ J_k^e = (X_k^e)^T X_k^e + u_k^e R u_k^e + J_{k+1} \] (A.5)

Substituting Eq. (A.2) into Eq. (A.5), we obtain
\[
\begin{align*}
\hat{J}_k^{e+1} = & \frac{1}{2} (X_k^{e+1})^T (X_k^{e+1} + u_k^{e+1} R u_k^{e+1}) + \hat{J}_k^{e+1} \phi(X_{k+1})
\end{align*}
\] (A.6)

Equation (A.6) is linear in \( \hat{J}_k^{e+1} \) with \( m \) unknowns, where \( m \) is number of elements of vector \( \phi(X_k) \). Taking the transpose of Eq.(A.6), and selecting \( m \) sets of states \( X_k \) called \( X_k^{(1)} \) to \( X_k^{(m)} \), it ends up with \( m \) equations with \( m \) unknowns:
\[
\begin{align*}
\phi(X_k^{(j)}) \hat{J}_k^{e+1} = & \frac{1}{2} (X_k^{(j)})^T (X_k^{(j)} + u_k^{(j)} R u_k^{(j)}) + \phi(X_{k+1}^{(j)}) \hat{J}_k^{e+1} \\
\phi(X_k^{(m)}) \hat{J}_k^{e+1} = & \frac{1}{2} (X_k^{(m)})^T (X_k^{(m)} + u_k^{(m)} R u_k^{(m)}) + \phi(X_{k+1}^{(m)}) \hat{J}_k^{e+1}
\end{align*}
\] (A.7)

where \( j = 1, 2, \ldots, m \).
\[
\begin{align*}
u_k^{(j)} & = U_k^{(j)} (X_k^{(j)}) = -R^{-1} B^T A^{-1} \left( \frac{∂^2 \psi}{∂ x}\right) - Q_k(X_k^{(j)}) \\
X_k^{(j)} & = f(X_k^{(j)}) + B u_k^{(j)}
\end{align*}
\] (A.8)

Equations (A.7) can be rewritten as
\[
\Phi(X_k) \hat{J}_k^{e+1} = RHS(X_k, \hat{W})
\] (A.10)

where the RHS is the \( m \times 1 \) vector composed of all the RHS of Eq.s (A.7) and the \( m \times m \) matrix \( \Phi(X_k) \) is given by
\[
\Phi(X_k) = \begin{bmatrix}
\phi(X_k^{(1)})^T & \cdots & \phi(X_k^{(m)})^T
\end{bmatrix} \begin{bmatrix}
X_k^{(1)} \\
\vdots \\
X_k^{(m)}
\end{bmatrix}
\] (A.11)

By using Eq.(A.11) in (A.10) leads to a recursive relationship for the network weights as
\[
\hat{J}_k^{e+1} = \Phi(X_k)^{-1} RHS(X_k, \hat{W})
\] (A.12)

For the inverse \( \Phi(X_k)^{-1} \) to exist, \( X_k^{(j)} \)'s should not be identical and the elements of vector \( \phi(X_k) \) should be linearly independent. One can select more than \( m \) minimum required sets of states, and formulate a recursive relationship for the over-defined system of equations. In fact, it is advisable to use a large number of data sets since it will better enable the network to capture the long term behavior or the system evolution for many different initial conditions and may help with faster convergence. In this case, the unique solution of the least squares minimization problem is simply:
\[
\hat{W}_k^{e+1} = (\Phi(X_k)^T \Phi(X_k))^{-1} \Phi(X_k)^T RHS(X_k, \hat{W})
\] (A.13)

REFERENCES


