2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Finite-Horizon Input-Constrained Nonlinear Optimal Control Using Single Network Adaptive Critics

Ali Heydari, *Student Member, IEEE,* and S. N. Balakrishnan, *Member, IEEE*

***Abstract*— A single neural network based controller called the Finite-SNAC is developed in this study to synthesize finite-horizon optimal controllers for nonlinear control-affine systems. For satisfying the constraint on the input, a non-quadratic cost function is used. Inputs to the neural network are the current system states and the time-to-go and the network outputs are the costates which are used to compute the feedback control. Convergence of the reinforcement learning based training method to the optimal solution, the training error and the network weights are provided. The resulting controller is shown to solve the associated time-varying Hamilton-Jacobi-Bellman (HJB) equation and provide the fixed-final-time optimal solution. Performance of the new synthesis technique is demonstrated through an attitude control problem wherein a rigid spacecraft performs a finite time attitude maneuver subject to control bounds. The new formulation has a great potential for implementation since it consists of only one neural network with single set of weights and it provides comprehensive *feedback solutions online* though it is trained offline.**

I. Introduction

There is a multitude of papers in the literature that use neural networks (NN) for the control of dynamical systems [1]-[4]. A few amongst them develop optimal control based on an approximate dynamic programming (ADP) formulation [3], [5]-[11]. Two classes of ADP based solutions, called the Heuristic Dynamic Programming (HDP) and the Dual Heuristic Programming (DHP) have emerged in the literature [3]. In HDP, the reinforcement learning is used to learn the cost-to-go from the current state while in the DHP, the derivative of the cost function with respect to the states, i.e. the costate vector is learnt by the neural networks [5]. The convergence proof of DHP for linear systems is presented in [6] and that of HDP for general case is presented in [7].

The implementation of the ADP learning is usually achieved through a dual network architecture called the Adaptive Critics (AC) [5], [8] . In the HDP class with ACs, one network, called the 'critic' network maps the input states to output the cost and another network called the 'action' network outputs the control with states of the system as its inputs [7]. In the DHP formulation, while the action network remains the same as with the HDP, the critic network outputs the costates with the current states as inputs.[8]-[9]. The single network adaptive critic (SNAC) [10] is shown to be able to eliminate the need for the second network and perform the DHP using only one network, resulting in a considerable decrease in the offline training effort and the simplicity of the online implementation through less required computational resources and storage memory. Note that these developments in the neural network literature have mainly addressed only the *infinite horizon* problems.

Finite-horizon optimal control is relatively more difficult. The difficulty is due to the time varying HJB equation resulting in *time-to-go dependent* optimal cost function and costates. If one were to use a shooting method, a two-point boundary value problem needs to be solved for each set of initial condition each time and it will provide only an open loop solution and only for one set of initial conditions. There is hardly any work in the neural network literature to solve this class of problems [11]-[12]. In this paper, a single neural network (Finite-SNAC) based solution is developed which embeds solutions to the HJB equation. Consequently, the offline trained network can be used to generate online feedback control. *Another major advantage is that this network provides optimal feedback solutions to any different final time as long as it is less than the final time for which the network is synthesized.*

In practical engineering problems, the designer faces constraints on the control effort. In order to facilitate the control constraint, a non-quadratic cost function [13], is used in this study.

Specifically, in this paper an ADP based NN controller for input-constrained finite-horizon optimal control for discrete-time input-affine nonlinear systems is developed. This is done through a SNAC scheme that uses the current states *and* the time-to-go as inputs. The scheme is DHP based. For the proof of convergence, proof of HDP for finite-horizon case is presented first. Then, it is shown that DHP has the same convergence result as HDP has and therefore, DHP also converges to the optimal solution. Finally, after presenting the convergence proofs of the training error and the network weights for the selected weight update law, the performance of the controller is evaluated with a spacecraft application in which a fixed final time attitude maneuver is carried out optimally.

Rest of the paper is organized as follows: the Finite-SNAC is developed in section II. Relevant convergence

This research was supported by a grant from the National Science Foundation.

A. Heydari is a PhD student at Mechanical & Aerospace Engineering Dept. of Missouri University of Science and Technology. (e-mail: ali.heydari@mail.mst.edu).

S. N. Balakrishnan is a Professor with Mechanical & Aerospace Engineering Dept. of Missouri University of Science and Technology. (e-mail: bala@mst.edu).

proofs are presented in section III. Numerical results and analysis from a spacecraft problem are presented in Section IV. Conclusions are given in Section V.

II. Theory of the Finite-SNAC

A single neural network (Finite-SNAC) that outputs the costates as a function of the current states and the time-to-go is used in this study. Its mapping is described in a functional form as

$$\lambda_{k+1} = NN(x_k, N-k, W),\ 0 \le k < N-1 \quad (1)$$

where $\lambda_{k+1} \in \mathbb{R}^n$ and $x_k \in \mathbb{R}^n$ denote the system costates at time $k+1$ and the states at time/stage k, respectively, and W denotes the network weights. n is the dimension of the state space. Note that for developing discrete control sets as a function of time-to-go, the specified final time is divided into N stages. Note that λ_{k+1} is a function of x_k and time-to-go $(N-k)$.

The neural network $NN(.)$ in this study is selected to be of a form that is linear in the weights.

$$NN(x, N-k, W) \equiv W^T\phi(x, N-k) \quad (2)$$

where $\phi(.) \in \mathbb{R}^m$ is composed of m linearly independent basis functions and $W \in \mathbb{R}^{m\times n}$, where m is the number of neurons.

Dynamics of the nonlinear control-affine system is assumed as

$$x_{k+1} = f(x_k) + g(x_k)u_k \quad (3)$$

A non-quadratic cost function J is assumed to incorporate the input constraints [13]. It is given by

$$J = \frac{1}{2}x_N^T Q_f x_N + \sum_{i=0}^{N-1}\frac{1}{2}(x_i^T Q x_i + G(u_i)) \quad (4)$$

where $G(.) \in \mathbb{R}$ is defined as

$$G(v) \equiv \int_0^v \rho^{-1}(w)^T R dw \quad (5)$$

$\rho^{-1}(.)$ denotes the inverse of function $\rho(.)$ which is a bounded continuous one-to-one real-analytic integrable saturating function which passes through the origin, like for example, a hyperbolic tangent function. Note that $G(.)$ is a non-negative scalar and $\int_0^v \rho^{-1}(w)^T dw$ for $\rho^{-1}(w) \in \mathbb{R}^m$ is defined as

$$\int_0^v \rho^{-1}(w)^T dw \equiv \sum_{i=1}^m \int_0^{v_i} \rho_i^{-1}(w)\, dw \quad (6)$$

where subscript i in v_i and ρ_i denotes the ith element of the corresponding vector.

The network training target should be calculated using following two equations [11]:

$$\lambda_N^t = Q_f x_N \quad (7)$$

$$\lambda_{k+1}^t = Qx_{k+1} + \left(\frac{\partial(f(x_{k+1}) + g(x_{k+1})u_{k+1})}{\partial x_{k+1}}\right)^T \lambda_{k+2}$$
$$0 \le k < N-1 \quad (8)$$

In the SNAC training process, λ_{k+2} on the right hand side of (8) will be substituted by $NN(x_{k+1}, N-(k+1), W)$ as described in [10].

The SNAC training should be done in such a way which along with learning the target given in (8) for every state x_k and time k, *the final condition (7) is also satisfied.* In this study, this idea is incorporated by augmenting the training input-target pairs with the final stage costate. Define following augmented parameters:

$$\bar{\lambda} \equiv [\lambda_{k+1} \ \ \lambda_N] \quad (9)$$

$$\bar{\phi} \equiv [\phi(x_k, N-k) \ \ \phi(x_{N-1}, 1)] \quad (10)$$

Now, the network output and the target to be learned are

$$\bar{\lambda} = W^T\bar{\phi} \quad (11)$$

$$\bar{\lambda}^t \equiv [\lambda_{k+1}^t \ \ \lambda_N^t] \quad (12)$$

The training error is defined as

$$e \equiv \bar{\lambda} - \bar{\lambda}^t = W^T\bar{\phi} - \bar{\lambda}^t \quad (13)$$

In each iteration along with selecting a random state x_k, a random time k, $0 \le k < N-1$, is also selected and λ_{k+1}^t is calculated using (8) after propagating x_k to x_{k+1}. Then, to calculate λ_N^t through (7), another randomly selected state will be considered as x_{N-1} and propagated to x_N and fed to (7). Finally $\bar{\lambda}^t$ will be formed using (12). This process is depicted graphically in Fig. 1. In this diagram, the left column follows (8) and the right column follows (7) for the target calculations.

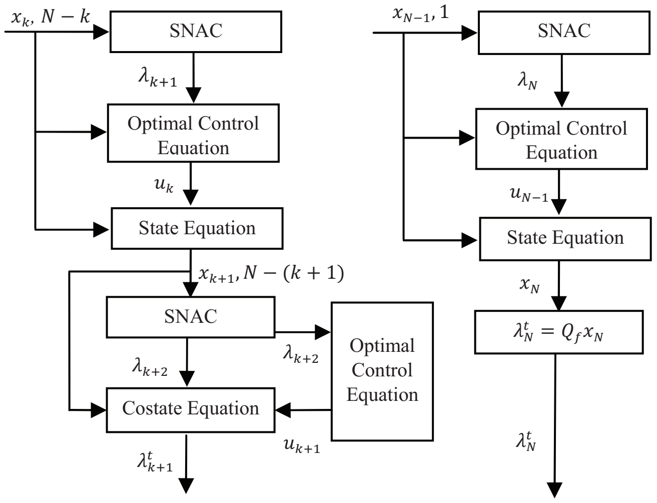

Fig. 1. Finite-SNAC training diagram

Having the input-target pair $\{[(x_k, N-k)\ (x_{N-1}, 1)], [\lambda_{k+1}^t\ \lambda_N^t]\}$ calculated, the network can be trained using some training method. In this study, the Galerkin method of approximation [14] is used. In this method, in order to find the unknown weight W one should solve the following set of linear equations.

$$\langle e, \bar{\phi}\rangle = 0_{n\times m} \quad (14)$$

where $\langle X, Y\rangle = \int_\Omega XY^T dx$ is the defined inner product on the compact set Ω on $\mathbb{R}^n$ and $0_{n\times m}$ denotes an $n \times m$ matrix of elements equal to zero. Denoting the ith row of matrices e and $\bar{\phi}$ by e_i and $\bar{\phi}_i$, respectively, (14) leads to following equations

$$\langle e_i, \bar{\phi}\rangle = 0_{1\times m}\ \forall i,\ 1 < i < n \quad (15)$$

$$\langle e_i, \bar{\phi}_j\rangle = 0\ \forall i,j,\ 1 < i < n,\ 1 < j < m \quad (16)$$

Substituting e from (13) into (14) results in

$$\langle e, \bar{\phi}\rangle = W^T\langle\bar{\phi}, \bar{\phi}\rangle - \langle\bar{\lambda}^t, \bar{\phi}\rangle = 0 \quad (17)$$

or

$$W = \langle\bar{\phi}, \bar{\phi}\rangle^{-1}\langle\bar{\phi}, \bar{\lambda}^t\rangle \quad (18)$$

Eq. (18) is the desired weight update for the training process.

Finally, for use in a discrete problem, the integral used in the inner products in (18) is discretized by evaluating the inner products on p different points in a mesh covering the compact set Ω [12]. Denoting the distance between the mesh points by Δx, one has

$$\langle \bar{\phi}, \bar{\phi} \rangle = \lim_{\|\Delta x\| \to 0} \bar{\boldsymbol{\phi}} \bar{\boldsymbol{\phi}}^T \Delta x \quad (19)$$

$$\langle \bar{\phi}, \bar{\lambda}^t \rangle = \lim_{\|\Delta x\| \to 0} \bar{\boldsymbol{\phi}} \bar{\boldsymbol{\lambda}}^{t^T} \Delta x \quad (20)$$

where

$$\bar{\boldsymbol{\phi}} = [\bar{\phi}(x_1) \;\; \bar{\phi}(x_2) \;\; \ldots \;\; \bar{\phi}(x_p)] \quad (21)$$

$$\bar{\boldsymbol{\lambda}}^{\boldsymbol{t}} = [\bar{\lambda}^t(x_1) \;\; \bar{\lambda}^t(x_2) \;\; \ldots \;\; \bar{\lambda}^t(x_p)] \quad (22)$$

$\bar{\phi}(x_i)$ and $\bar{\lambda}^t(x_i)$ denote $\bar{\phi}$ and $\bar{\lambda}^t$ evaluated on the mesh point x_i, respectively.

Using (19) and (20), the weight update rule (18) is now simplified to the standard least square form as

$$W = (\bar{\boldsymbol{\phi}} \bar{\boldsymbol{\phi}}^T)^{-1} \bar{\boldsymbol{\phi}} \bar{\boldsymbol{\lambda}}^{t^T} \quad (23)$$

Note that for the inverse of the matrix $(\bar{\boldsymbol{\phi}} \bar{\boldsymbol{\phi}}^T)$ to exist, one needs the basis functions ϕ_i to be linearly independent and the number of mesh points p to be greater than or equal to half of the number of neurons m.

Though (23) looks like an one shot solution for the ideal NN weights, the training is an *iterative* process which needs selecting different random states from the problem domain and times and updating the network weights by repeated use of (23). The reason for the iterative nature of the training process is the reinforcement learning basis of ADP. To make it more clear, one should note that $\bar{\lambda}^t$ used in the weight update (23) is not the true optimal costate but its approximation with a current estimation of the ideal unknown weight, i.e. $\bar{\boldsymbol{\lambda}}^{\boldsymbol{t}}(W)$. Denoting the weights at the ith epoch of the weight update by $W^{(i)}$ results in the following iterative procedure as

$$W^{(i+1)} = (\bar{\boldsymbol{\phi}} \bar{\boldsymbol{\phi}}^T)^{-1} \bar{\boldsymbol{\phi}} \bar{\boldsymbol{\lambda}}^{\boldsymbol{t}} \left(W^{(i)}\right)^T \quad (24)$$

The weight training is started with an initial weight $W^{(0)}$ and iterated through (24) until the weights converge. The initial weight can be set to zero or can be selected based on the linearized solutions of the given nonlinear system.

Once the network is trained, it can be used for optimal feed-back control in the sense that in the online implementation, the states and the time will be fed into the network to generate the optimal costate using (1) and the optimal control will be calculated as

$$u_k = -\rho(R^{-1} g(x_k)^T \lambda_{k+1}) \quad (25)$$

III. Convergence Proofs

Convergence proof for the proposed optimal controller is composed of three parts: first of all, one needs to show that the reinforcement learning, which the target calculation is based on, will result in the optimal target, then it needs to be shown that the weight update will force the error between the network output and the target to converge to zero and finally the network weights should be shown to converge.

A. Convergence of the algorithm to the optimal solution

The proposed algorithm for the Finite-SNAC training is DHP in which starting at an initial value for the costate vector one iterates to converge to the optimal costate. Denoting the iteration index by a superscript and the time index by a subscript, the learning algorithm for finite horizon optimal control starts with an initial value assignment to λ_k^0 for all k's, e.g. $\lambda_k^0 = 0 \;\forall k$, and repeating below three calculations for different i's from zero to infinity.

$$u_k^i = -\rho\left(R^{-1} g(x_k)^T \lambda_{k+1}^i\right) \quad (26)$$

$$\lambda_k^{i+1} = Q x_k + A\left(x_k, u_k^i\right)^T \lambda_{k+1}^i \quad (27)$$

$$\lambda_N^{i+1} = Q_f x_N \quad (28)$$

Eq. (28) is actually the final condition of the optimal control problem. Note that,

$$A\left(x_k, u_k^i\right) \equiv \frac{\partial\left(f(x_k) + g(x_k) u_k^i\right)}{\partial x_k} \quad (29)$$

$$\lambda_{k+1}^i \equiv \lambda^i(x_{k+1}) = \lambda^i\left(f(x_k) + g(x_k) u_k^i\right) \quad (30)$$

The problem is to prove that the iterative procedure results in the optimal value for the costate λ and control u. The convergence proof presented here is based on the convergence of HDP, in which the parameter subject to evolution is the cost function J whose behavior is much simpler to discuss as compared to that of the costate vector λ.

In the latter, the cost function J needs to be initialized, e.g. $J^0(x_k, k) = 0 \;\forall k$, and iteratively updated throught the following steps.

$$J^{i+1}(x_k, k) = \frac{1}{2}\left(x_k^T Q x_k + G(u_k^i)\right) + J^i(x_{k+1}, k+1) \quad (31)$$

$$u_k^i = \operatorname{argmin}_u \left(J^{i+1}(x_k, k)\right) = -\rho\left(R^{-1} g(x_k)^T \frac{\partial J_{k+1}^i}{\partial x_{k+1}}\right) \quad (32)$$

For finite horizon case, the final condition given below is satisfied at every iteration.

$$J^{i+1}(x_N, N) = \frac{1}{2} x_N^T Q_f x_N \quad (33)$$

Note that $J_k \equiv J(x_k, k)$ and

$$J_{k+1}^i \equiv J^i\left(f(x_k) + g(x_k) u_k^i, k+1\right) \quad (34)$$

In [7] the authors have proved that HDP for infinite-horizon regulation converges to the optimal solution. In this paper, that proof is modified to cover the case of constrained finite-horizon optimal control. For this purpose following four Lemmas are required of which three are cited from [7] with some modifications to handle the time dependency of the optimal cost function.

Lemma 1 [7]: Using any arbitrary control sequence of μ_k, and Λ^i defined as

$$\Lambda^{i+1}(x_k, k) = \frac{1}{2}\left({x_k}^T Q x_k + G(\mu_k)\right) + \Lambda^i\left(f(x_k) + g(x_k)\mu_k, k+1\right) \quad (35)$$

If $\Lambda^0(x_k, k) = J^0(x_k, k) = 0$ then $\Lambda^i(x_k, k) \geq J^i(x_k, k) \;\forall i$ where $J^i(x_k, k)$ is iterated through (31) and (32).

Proof: The proof is given in [7]

Lemma 2: If the system is controllable then $J^i(x_k,k)$, resulted from (31) and (32), is upper bounded by an existing bound $Y(x_k,k)$.

Proof: The proof is inspired by the proof of similar Lemma in [7], however, this is an important modification to deal with finite horizon problems. Let η_k be an arbitrary control. Let $Z^0(x_k,k) = J^0(x_k,k) = 0$, where Z^i is updated as

$$Z^{i+1}(x_k,k) = \frac{1}{2}\left(x_k{}^T Q x_k + G(\eta_k)\right) + Z^i(x_{k+1},k+1) \quad (36)$$

$$Z^{i+1}(x_N,N) = \frac{1}{2}x_N^T Q_f x_N \quad (37)$$

$$x_{k+1} = f(x_k) + g(x_k)\eta_k \quad (38)$$

Defining $Y(x_k,k)$ as

$$Y(x_k,k) = \frac{1}{2}x_N^T Q_f x_N + \sum_{n=0}^{N-k-1}\frac{1}{2}\left(x_{k+n}^T Q x_{k+n} + G(\eta_{k+n})\right) \quad (39)$$

Subtracting (39) from (36) results in

$$Z^{i+1}(x_k,k) - Y(x_k,k) = Z^i(x_{k+1},k+1) - \left(\frac{1}{2}x_N^T Q_f x_N + \sum_{n=1}^{N-k-1}\frac{1}{2}\left(x_{k+n}^T Q x_{k+n} + G(\eta_{k+n})\right)\right) \quad (40)$$

which is the equivalent of

$$Z^{i+1}(x_k,k) - Y(x_k,k) = Z^i(x_{k+1},k+1) - Y(x_{k+1},k+1) \quad (41)$$

If $i \geq N-k-1$ then above equation results in

$$Z^{i+1}(x_k,k) - Y(x_k,k) = Z^{i-(N-k-1)}(x_N,N) - Y(x_N,N) \quad (42)$$

But the right hand side of (42) is

$$Z^{i-(N-k-1)}(x_N,N) - Y(x_N,N) = \frac{1}{2}x_N^T Q_f x_N - \frac{1}{2}x_N^T Q_f x_N = 0 \;\; if\; i > N-k-1 \quad (43)$$

$$Z^0(x_N,N) - Y(x_N,N) = 0 - Y(x_N) < 0 \;\; if\; i = N-k-1 \quad (44)$$

Hence, one has

$$Z^{i+1}(x_k,k) - Y(x_k,k) \leq 0 \;\; if\; i \geq N-k-1 \quad (45)$$

For the case of $i < N-k-1$ one has

$$Z^{i+1}(x_k,k) - Y(x_k,k) = Z^0(x_{k+i+1},k+i+1) - Y(x_{k+i+1},k+i+1) \quad (46)$$

But, $Z^0(x_{k+i+1},k+i+1) = 0$, hence,

$$Z^{i+1}(x_k,k) - Y(x_k,k) = 0 - Y(x_{k+i+1},k+i+1) < 0 \;\; if\; i < N-k-1 \quad (47)$$

In conclusion, (45) and (47) lead to

$$Z^i(x_k,k) \leq Y(x_k,k) \;\; \forall i \quad (48)$$

From Lemma 1 with $\mu_k = \eta_k$ one has $J^i(x_k,k) \leq Z^i(x_k,k)$, hence,

$$J^i(x_k,k) \leq Y(x_k,k) \quad (49)$$

which proves Lemma 2.

Lemma 3 [7]: If the system is controllable and the optimal control problem can be solved, then there exists a least upper bound $J^*(x_k,k)$, $J^*(x_k,k) \leq Y(x_k,k)$, which satisfies equation (31) when J^i and J^{i+1} are replaced by J^*,and $0 \leq J^i(x_k,k) \leq J^*(x_k,k) \leq Y(x_k,k)$ where $Y(x_k,k)$ is defined in Lemma 2.

Proof: The proof given is in [7].

Lemma 4 [7]: The sequence of J^i defined by HDP, in case of $J^0(x_k) = 0$, is non-decreasing.

Proof: The proof given is in [7].

Theorem 1: The sequence of J^i iterated through (31) to (33), in case of $J^0(x_k) = 0$ converges to the fixed final time optimal solution.

Proof: Using the results of Lemma 4 and Lemma 2 one has

$$J^i \to J^\infty \text{ as } i \to \infty. \quad (50)$$

From Lemma 3

$$J^\infty \leq J^* \quad (51)$$

Since J^∞ satisfies the HJB equation and the finite-horizon final condition one has

$$J^\infty = J^* \quad (52)$$

which completes the proof.

Now, we can proceed to the convergence proof DHP.

Theorem 2: The sequence of λ_k^i iterated through (26) to (28) for $k = 0,1,\dots,N$ providing $\lambda_k^0 = 0 \;\forall k$, converges to the optimal costate vector for the fixed final time problem as $i \to \infty$.

Proof: The idea is to use the method of induction to show that the evolution of the sequence in DHP is identical to that of HDP, i.e., at each learning iteration, we will have $\lambda_k^i = \frac{\partial J^i(x_k,k)}{\partial x_k} \;\forall k$, where λ_k^i is resulted from DHP and J^i is resulted from HDP. Since J^i, based on Theorem 1, converge to the optimal values as $i \to \infty$, λ_k^i will also converge to the optimal costate vector. The steps of the proof are skipped because of the page constraints.

B. *Convergence of the error of the weight update*

This step is to prove that the weight update rule makes the error between the network output and the target converge to zero and that the network weights themselves converge. The idea behind proofs of Theorem 3 and 4 are similar to [14], but, since the error equation and the dimension of the error are different compared to [14], the processes of the proofs are different and given below.

Theorem 3: Training error convergence
The weight update (14) will force the error (13) to converge to zero as the number of neurons of the neural networks, m, tends to infinity.

Proof: Using Lemma 5.2.9 from [14], assuming $\bar{\phi}$ to be orthonormal, rather than being linearly independent, does not change the convergence result of the weight update. Assume $\bar{\phi}$ is a matrix formed by m orthonormal basis functions $\bar{\phi}_j$ as its rows where $1 \leq j \leq m$ among the infinite number of orthonormal basis functions $\{\bar{\phi}_j\}_1^\infty$. The orthonormality of $\{\bar{\phi}_j\}_1^\infty$ implied that if a function $\psi \in span\{\bar{\phi}_j\}_1^\infty$ then

$$\psi = \sum_{j=1}^{\infty}\langle \psi, \bar{\phi}_j\rangle \bar{\phi}_j \quad (53)$$

And for any ϵ one can select m sufficiently large to have

$$\left\|\sum_{j=m+1}^{\infty}\langle \psi, \bar{\phi}_j\rangle \bar{\phi}_j\right\| < \epsilon \quad (54)$$

where $\|.\|$ denotes norm operation. From (14) one has

$$\langle e, \bar{\phi}_j\rangle = 0 \;\; \forall j,\; 1 < j < m \quad (55)$$

And from (13)

$$\langle e, \bar{\phi}_j \rangle = W^T \langle \bar{\phi}, \bar{\phi}_j \rangle - \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \quad (56)$$

which is equivalent to

$$\langle e, \bar{\phi}_j \rangle = \sum_{i=1}^{m} W_i^T \langle \bar{\phi}_i, \bar{\phi}_j \rangle - \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \quad (57)$$

where W_i is the ith row of weight matrix W.

On the other hand, one can expand the error e using the orthonormal basis functions $\{\bar{\phi}_j\}_1^\infty$.

$$e = \sum_{j=1}^{\infty} \langle e, \bar{\phi}_j \rangle \bar{\phi}_j \quad (58)$$

Inserting (57) into (58) results in

$$e = \sum_{j=1}^{\infty} \left(\sum_{i=1}^{m} W_i^T \langle \bar{\phi}_i, \bar{\phi}_j \rangle \bar{\phi}_j - \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \bar{\phi}_j \right) \quad (59)$$

But, from the weight update (55), the right hand side of (57) is also equal to zero. Applying this to (59) results in

$$e = \sum_{j=m+1}^{\infty} \left(\sum_{i=1}^{m} W_i^T \langle \bar{\phi}_i, \bar{\phi}_j \rangle \bar{\phi}_j - \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \bar{\phi}_j \right) \quad (60)$$

Due to the orthonormality of the basis functions, one has

$$\langle \bar{\phi}_i, \bar{\phi}_j \rangle = 0 \ \ \forall i \neq j \quad (61)$$

Hence, (60) simplifies to

$$e = -\sum_{j=m+1}^{\infty} \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \bar{\phi}_j \quad (62)$$

Using (54) for $\psi = \bar{\lambda}^t$, as m increases, e decreases to zero.

$$\lim_{m \to \infty} \|e\| = 0 \quad (63)$$

This completes the proof.

Theorem 4: Neural network weight convergence

Assuming an ideal set of weights, denoted by W^*, where

$$\bar{\lambda}^t = \sum_{i=1}^{\infty} W_i^{*T} \bar{\phi}_i \quad (64)$$

Then, using the weight update (14), one has $(W - W_{trunc}^*) \to 0$ where W_{trunc}^* is the truncated first m row of the ideal weight W^*.

Proof: The training error is defined as

$$e \equiv \bar{\lambda} - \bar{\lambda}^t \quad (65)$$

Hence

$$e = \left(W^T - {W_{trunc}^*}^T \right) \bar{\phi} - \sum_{i=m+1}^{\infty} W_i^{*T} \bar{\phi}_i \quad (66)$$

Note that $\bar{\phi}$ is a matrix formed by the first m orthonormal basis functions $\bar{\phi}_i$ as its rows, i.e. $1 \leq i \leq m$. The inner product of both sides of (66) by $\bar{\phi}$ results in

$$\langle e, \bar{\phi} \rangle = \left(W^T - {W_{trunc}^*}^T \right) \langle \bar{\phi}, \bar{\phi} \rangle - \sum_{i=m+1}^{\infty} W_i^{*T} \langle \bar{\phi}_i, \bar{\phi} \rangle \quad (67)$$

The last term on the right hand side of the above equation vanishes due to the orthonormality property of the basis functions. Considering $\langle \bar{\phi}, \bar{\phi} \rangle = I$, (66) simplifies to

$$\langle e, \bar{\phi} \rangle = W^T - {W_{trunc}^*}^T \quad (68)$$

Examining (68) further, the weight update implies the left hand side to be zero, hence, using the weight update (14) one has $W \to W_{trunc}^*$.

IV. Simulations

For demonstration of the new synthesis technique, the problem of nonlinear satellite attitude control has been selected. Satellite dynamics can be represented as [15]

$$\frac{d\omega}{dt} = I^{-1}(N_{net} - \omega \times I\omega) \quad (69)$$

where I, ω, and N_{net} are inertia tensor, angular velocity vector of the body frame with respect to inertial frame and the vector of the total torque applied on the satellite, respectively. The selected satellite is an inertial pointing satellite; hence, one is interested in its attitude with respect to the inertial frame. All vectors are represented in the body frame and the sign × denotes cross product of two vectors.

The total torque is composed of control and the disturbance torques. The control torque is the torque created using satellite actuators. Since control torque is limited in practice, this problem is 'input-constrained'.

Following [16] and its order of transformation, the kinematic equation of the satellite is

$$\frac{d}{dt}\begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 & \sin(\varphi)\tan(\theta) & \cos(\varphi)\tan(\theta) \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi)/\cos(\theta) & \cos(\varphi)/\cos(\theta) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (70)$$

where φ, θ, and ψ are the three Euler angles describing the attitude of the satellite with respect to x, y, and z axes of the inertial coordinate system, respectively. The subscript x, y, and z denote the corresponding elements of the vector ω.

To form the state space equation of satellite attitude problem, one can choose the three Euler angles and the three elements of the angular velocity as the states and form the following state space equation as

$$\dot{x} = f(x) + g(x)u \quad (71)$$

where

$$f(x) \equiv \begin{bmatrix} M_{3\times1} \\ I^{-1}(N_{gg} - \omega \times I\omega) \end{bmatrix} \quad (72)$$

$$g \equiv \begin{bmatrix} 0_{3\times3} \\ I^{-1} \end{bmatrix} \quad (73)$$

$$x = [\varphi \quad \theta \quad \psi \quad \omega_x \quad \omega_y \quad \omega_z]^T \quad (74)$$

$$u = \begin{bmatrix} N_{ctrl_x} & N_{ctrl_y} & N_{ctrl_z} \end{bmatrix}^T \quad (75)$$

$M_{3\times1}$ denotes the right hand side of equation (70) and $0_{3\times3}$ denotes a three-by-three null matrix.

A. Numerical Results

The moment of inertia matrix of the satellite is chosen as

$$I = \begin{bmatrix} 100 & 2 & .5 \\ 2 & 100 & 1 \\ .5 & 1 & 110 \end{bmatrix} kg.m^2 \quad (76)$$

The different moments around different axes and also the non-zero off-diagonal elements result in some gravity gradient disturbance torque acting on the satellite.

The initial states are selected based on initial Euler angles of 60, -20, and -70 deg. and zero angular rates. The mission of the controller is to perform an attitude maneuver to bring the states to zero, in a fixed final time of 800 sec. A saturation limit of $\pm 0.002\ N.m$ is selected for the actuators.

The orbit for the satellite is assumed circular with a radius of 20,000 km, and an inclination of 90 degrees.

The state and control weight matrices are selected as

$$Q = diag(1 \ 1 \ 1 \ 100 \ 100 \ 100) \quad (77)$$

$$Q_f = 4000\, Q \quad (78)$$

$$R = diag(10^5 \ 10^5 \ 10^5) \quad (79)$$

Note that the last three diagonal elements of matrix Q and Q_f correspond to the angular rates with the unit of radians per second and are set to higher values relative to the first three elements. This is because the objective in this study is to force the angles along with the rates to reach zero and higher weights on angular rates helps this process. Moreover, higher values for Q_f compared to Q are to stress the importance of minimizing the terminal errors. A tangent hyperbolic function describes the saturating function $\rho(.)$ used in the performance index (4) and is scaled to reflect the actuator bounds.

The network weights are initialized to zero and the basis functions are selected as polynomials x_i, x_i^2, x_i^3 for i =1 to 7 along with $x_i x_j$, $x_i^2 x_7$, $x_i x_7^2$, $x_i x_7^3$ and $x_i e^{-x_7}$ for $i, j = 1$ to 6 $i \neq j$, resulting in 60 neurons, where, x_i is the ith network input. Note that x_7 is the fed normalized time-to-go and its contribution in the basis functions are selected through some trial and error such that the network error is as small as possible. For the training process, in each Epoch, 50 initial states among a previously selected interval of states are randomly selected to form a mesh and the weight update (23) is used for training the neural network. The training is performed for 600 Epochs, until the weights converge.

The simulation results are shown in Fig. 2 and Fig. 3 by the black plots. The Euler angles as seen in Fig. 2 have nicely converged close to zero in the fixed final time of 800 sec. Fig. 3 shows the applied control history and as expected it has not violated the control bounds.

To demonstrate the versatility of the proposed controller, using the same trained network, the same attitude maneuver is performed with a *shorter* time-to-go, i.e. 400 sec. and the results are superimposed with previous results and shown in Fig. 2 and Fig. 3 using blue plots. As can be seen, the controller has applied another control sequence on the satellite with more saturation at first in order to accomplish the same mission in a shorter time-to-go of 400 sec. This illustrates the power of the Finite-SNAC technique that the same controller will be optimal for all of the final times less than or equal that horizon by virtue of the Pontryagin's principle of optimality.

In order to analyze the effect of external disturbances on the controller, the gravity gradient disturbance is modeled [15] and applied on the satellite and the results are shown using red plots in the same figures. Note that even-though this method is not developed to measure and cancel the effect of the disturbance, the feedback form of the controller is robust enough to be able to get an acceptable trajectory even in the presence of unknown disturbances.

V. Conclusions

A finite-horizon optimal neurocontroller, that embeds the solution to finite-horizon HJB equation, has been developed in this study. The developed neurocontroller has been shown to solve finite-horizon input-constrained optimal control problem for discrete-time nonlinear control-affine systems. Convergence proofs have been given. The numeric simulation from a satellite control problem indicate that the developed method is very versatile and has a good potential for use in solving for optimal closed loop control of control-affine nonlinear systems..

References

[1] K.S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks*, vol. 1 (1), pp. 4-27, 1990.

[2] P. J. Werbos, "Backpropagation through time: what it does and how to do it", in *Proc. of the IEEE*, vol. 78 (10), pp. 1550-1560, 1990.

[3] P. J. Werbos, "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.

[4] D. P. Bertsekas, J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Proc. IEEE Conference on. Decision and Control*, pp. 560-564, 1995

[5] D.V. Prokhorov and D.C. II Wunsch, "Adaptive critic designs," *IEEE Trans. on Neural Networks*, vol. 8 (5), pp. 997-1007, 1997.

[6] X. Liu and S. N. Balakrishnan, "Convergence analysis of adaptive critic based optimal control," in *Proc. American Control Conf.*, Chicago, USA, 2000, pp. 1929-1933.

[7] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf , "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Trans. On Systems, Man, and Cybernetics—Part B*, vol. 38, pp. 943-949, 2008.

[8] S. N. Balakrishnan, and V. Biega, "Adaptive-critic based neural networks for aircraft optimal control", *J. of Guidance, Control and Dynamics*, vol. 19 (4), pp. 893-898, 1996.

[9] S. Ferrari, and R. F. Stengel, "Online adaptive critic flight control, *J. of Guidance, Control and Dynamics*, vol. 27 (5), pp. 777-786, 2004.

[10] R. Padhi, N. Unnikrishnan, X. Wang, and S. N. Balakrishnan, "A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems," *Neural Networks*, vol. 19, pp.1648–1660, 2006.

[11] D. Han and S. N. Balakrishnan, "State-constrained agile missile control with adaptive-critic-based Neural Networks," *IEEE Trans. on Control Systems Technology*, vol. 10 (4), pp. 481-489, 2002.

[12] T. Cheng, F. L. Lewis, and M. Abu-Khalaf, "Fixed-final-time-constrained optimal control of nonlinear systems using Neural Network HJB approach," *IEEE Trans. on Neural Networks*, vol. 18 (6), pp. 1725-1737, 2007.

[13] S. E. Lyshevski, "Optimal control of nonlinear continuous-time systems: Design of bounded controllers via generalized nonquadratic functionals," in *Proc. American Control Conf.*, 1998, pp. 205–209.

[14] R. Beard, "Improving the closed-loop performance of nonlinear systems," Ph.D. Thesis, Rensselaer Polytechnic Institute, USA, 1995.

[15] J. R. Wertz, *Spacecraft Attitude Determination and Control*, Reidel, 1978.

[16] P. H. Zipfel, *Modeling and Simulation of Aerospace Vehicle Dynamics*, AIAA, 2000.

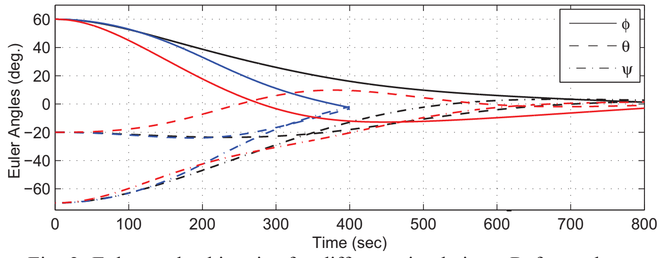

Fig. 2. Euler angles histories for different simulations. Refer to the text for color coding.

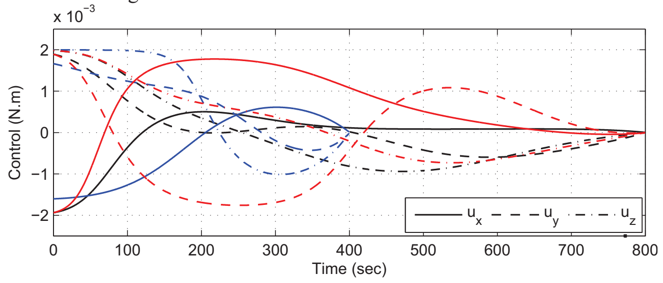

Fig. 3. Control histories for different simulations. Refer to the text for color coding.