Finite-Horizon Optimal Control Using Neural Networks with an Application to Orbit Transfer Problems

Ali Heydari

Missouri University of Science & Technology, Rolla, Missouri, 65409

A new controller is developed in this study, called Finite-SNAC, which embeds solutions to the Hamilton-Jacobi-Bellman equations for finite-time problems of control-affine nonlinear systems. This is a single neural network controller and the inputs to the network are the state vector along with the time-to-go and the output is the optimal costate vector to be used in calculating the optimal control vector. Convergence of the reinforcement learning based iterative method to the optimal solution along with the convergence of the training error and the network’s weights are proved. A fixed final time orbital spacecraft maneuver problem is solved and the results show the excellent potential of the proposed technique. A byproduct of the solution process is that the same network can be used to produce optimal feedback control over a range of initial conditions and final times.

I. Introduction

Among the first researches on using neural networks (NN) for control, one can cite Ref. 1 and 2, which demonstrated the huge potential capability of the NN for control. Within diverse developed applications of NN in this field, some researchers focused on optimal control of nonlinear systems. The results have been remarkable. Some of the developed optimal neurocontrollers are based on so called Approximate Dynamic Programming (ADP) proposed in Ref. 5. Two classes of the ADP based solutions, called Heuristic Dynamic Programming (HDP) and Dual Heuristic Programming (DHP) have emerged in the literature. In HDP, or cost function based reinforcement learning (RL), the RL is used to learn the cost-to-go for each state while in DHP, or costate based RL, the derivative of the cost function with respect to the states is being learned, i.e. the costate vector. The convergence proof of DHP for linear systems is presented in Ref. 10 and that of HDP for general case is presented in Ref. 11.

The implementation of the ADP is usually achieved through a dual network architecture called the Adaptive Critics (AC). In the HDP class with ACs, one network, called the ‘critic’ network maps the input states to the cost and another network called the ‘action’ network outputs the control with states of the system as its inputs. 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. Single network adaptive critic (SNAC) developed in Ref. 16 is shown to be able to eliminate the need for the second network and perform DHP using only one network. This results in a considerable decrease in the offline training effort and the simplicity of the online implementation through less required computational resources and storage memory. Similarly, the J-SNAC eliminates the need for the action network in an HDP scheme. Note that these developments in the neural networks literature have mainly addressed only the infinite horizon problems (regulation, tracking).

Finite-horizon optimal control is one of the difficult branches of optimal control theory. The difficulty is due to dealing with a time-varying Hamilton-Jacobi-Bellman (HJB) equation resulting in time-to-go dependent optimal cost and costate vector. In the finite-horizon case, the cost-to-go is not only a function of the current states, but also a function of how much time is left (time-to-go) to accomplish the goal. 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. There is hardly any work in the neural network literature to solve this class of problems.

The authors of Ref. 18 suggested using separate networks for learning the costate vector at different time steps for nonlinear discrete-time systems. Another method suggested in Ref. 19 and 20 is based on using a NN with time-varying weights whose weight should be calculated through interpolating among available weights for different time steps. 

1 Ph.D. Student, Dept. of Mechanical and Aerospace Eng., 400 W. 13th St. Rolla, MO 65409, ali.heydari@mail.mst.edu, AIAA Student Member.
steps. These approaches result in a large memory requirement due to needing to store different set of weights for different time steps.

The contribution of this paper is developing a single neural network (Finite-SNAC) controller which embeds solutions to the HJB equation without time-dependent weights. Consequently, the offline trained network can be used to generate online feedback control. Note 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.

The scheme is DHP based and for the proof of the validity of the proposed controller the convergence proof of HDP for finite-horizon case is presented which is based on the convergence proof of Ref. 11 for the infinite-horizon case with some modifications. Having proved the convergence of HDP, it is shown that DHP has the same convergence result as the HDP has and hence, DHP also converges to the optimal solution. Moreover, the convergence proofs of the training error and the network weights for the selected weight update law are presented.

For simulation study, a spacecraft rendezvous problem is selected, i.e. performing orbital maneuver and reaching the destination in a predetermined and fixed time. The performance of the proposed controller is analyzed through comparing the results with the optimal open loop numerical solution and performing some robustness analysis.

The rest of the paper is organized as follows: the Finite-SNAC is developed in section II. Relevant convergence theorems are presented in section III whose proofs are given in the Appendix. The orbital maneuver problem is discussed and simulated in section IV, and the conclusions are given in section V.

II. Theory of the Finite-SNAC

The discrete-time nonlinear input-affine system and the assumed quadratic cost function to be minimized are given below

\[
x_{k+1} = f(x_k) + g(x_k)u_k
\]

\[
J = \frac{1}{2} x_k^T Q x_k + \sum_{i=0}^{N-1} \frac{1}{2} (x_i^T Q x_i + u_i^T R u_i)
\]

where \(x_k \in \mathbb{R}^n\) and \(u_k \in \mathbb{R}^l\) denote the state and the control vectors at time step \(k\), respectively, \(n\) is the order of the system and \(l\) is the system’s number of inputs. \(f(.) \in \mathbb{R}^n\) and \(g(.) \in \mathbb{R}^{n \times l}\) are the system dynamics and \(Q_f \in \mathbb{R}^{n \times n}\) and \(Q \in \mathbb{R}^{n \times n}\) and \(R \in \mathbb{R}^{l \times l}\) are the penalizing matrices for the final states, states, and control vectors, respectively. \(Q_f\) and \(Q\) should be at least positive semi-definite while \(R\) has to be a positive definite matrix. Finally, \(N\) is the total (fixed) number of time steps and superscripted \(T\) denotes the transpose operation.

Denoting the neural network mapping by \(NN(.)\), a single neural network called the Finite-SNAC in the below form is suggested to output the desired costate vector based on its inputs; the state vector and the time-to-go.

\[
\lambda_{k+1} = NN(x_k, N-k, W), \ 0 \leq k < N - 1
\]

where \(\lambda_{k+1} \in \mathbb{R}^n\) is the system costate vector at time step \(k + 1\) and \(W\) denotes the network weight matrix.

The neural network in this paper is selected to be linear in the weights,\(^21\). Hence,

\[
NN(x, N-k, W) = W^T \phi(x, N-k)
\]

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

The optimal cost, denoted by \(J^*\), is given by solving the discrete-time HJB equation,\(^22\).

\[
J^*(x_k, k) = \min_{u_k} \left( \frac{1}{2} (x_k^T Q x_k + u_k^T R u_k) + J(x_{k+1}, k + 1) \right), \ 0 \leq k < N - 1
\]

The optimal control, \(u_k^*\), is obtained from

\[
u_k^* = \arg\min_{u_k} \left( \frac{1}{2} (x_k^T Q x_k + u_k^T R u_k) + J(x_{k+1}, k + 1) \right), \ 0 \leq k < N - 1
\]

Define \(\lambda_k = \frac{\partial J_k}{\partial x_k}\) to get
Replacing $u_k$ in (5) by $u_k^*$, the HJB equation reads

$$J^*(x_k,k) = \frac{1}{2} (x_k^T Q x_k + u_k^T R u_k^*) + J^*(x_{k+1},k+1), \quad 0 \leq k < N-1$$  \hspace{1cm} (8)

The costate equation can be derived by taking the derivative of both sides of (8) with respect to $x_k$ as

$$\lambda_{k+1} = Q x_{k+1} + \left( \frac{\partial (f(x_{k+1}) + \theta(x_{k+1}) u_{k+1})}{\partial x_{k+1}} \right)^T \lambda_{k+2}, \quad 0 \leq k < N-1$$  \hspace{1cm} (9)

Note that

$$J^*(x_N,N) = \frac{1}{2} x_N^T Q_f x_N$$  \hspace{1cm} (10)

Hence,

$$\lambda_N = Q_f x_N$$  \hspace{1cm} (11)

Now, the network training target, denoted by $\tilde{\lambda}$, can be calculated using the following two equations.

$$\lambda_{k+1}^t = Q x_{k+1} + \left( \frac{\partial (f(x_{k+1}) + \theta(x_{k+1}) u_{k+1})}{\partial x_{k+1}} \right)^T \lambda_{k+2}, \quad 0 \leq k < N-1$$  \hspace{1cm} (12)

$$\lambda_N^t = Q_f x_N$$  \hspace{1cm} (13)

In the training process, $\lambda_{k+2}$ on the right hand side of (12) will be substituted by $NN(x_{k+1}, N-(k+1), W)$ as described in the regular SNAC training process in Ref. 16.

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-to-go will be fed into the network to generate the optimal costate vector and the optimal control will be calculated through (7).

It is imperative that the Finite-SNAC training should be done in such a way which along with learning the target given in (12) for every state $x_k$ and time $k$, the final condition (13) is also satisfied. In order to accomplish that objective, the training input-target pairs are augmented in such a way which the final condition is forced to be met in each learning iteration. To do so, the following augmented parameters are defined:

$$\tilde{\lambda} \equiv [\lambda_{k+1} \lambda_N]$$  \hspace{1cm} (14)

$$\tilde{\phi} \equiv [\phi(x_k, N-k) \phi(x_{N-1}, 1)]$$  \hspace{1cm} (15)

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

$$\tilde{\lambda} = W^T \tilde{\phi}$$  \hspace{1cm} (16)

$$\tilde{\lambda}^t \equiv [\lambda_{k+1}^t \lambda_N^t]$$  \hspace{1cm} (17)

Consequently, the training error is defined as

$$e \equiv \tilde{\lambda} - \tilde{\lambda}^t = W^T \tilde{\phi} - \tilde{\lambda}^t$$  \hspace{1cm} (18)

Now, in each iteration along with selecting a random state $x_k$ in the domain of interest, a random time $k$, $0 \leq k < N-1$, is also selected. Feeding $x_k$ and $N-k$ into (3) results in a costate vector which will be used for calculating $u_k$ through (7). Having $x_k$ and $u_k$ one can propagate $x_k$ to $x_{k+1}$ using (1). Having $x_{k+1}$ fed into (3) once
more, \( \lambda_{k+2} \) can be obtained. This value is needed for the target \( \lambda_{k+1}^N \) calculation through (12). Then, to calculate \( \lambda_N^t \) through (13), another randomly selected state will be considered as \( x_{N-1} \) and propagated to \( x_N \) using the similar process discussed above for propagating \( x_k \) to \( x_{k+1} \), and fed to (13). Finally \( \lambda^t \) will be formed using (17). This process is depicted graphically in Fig. 1. In this diagram, the left column follows (12) and the right column follows (13) for the target calculation.

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

\[
\langle e, \tilde{\phi} \rangle = 0_{n \times m} \quad \text{(19)}
\]

where \( \langle X, Y \rangle = \int_\Omega XY^T d\Omega \) is the defined inner product on the compact set \( \Omega \) on \( \mathbb{R}^n \) and \( 0_{n \times m} \) denotes an \( n \times m \) matrix of elements equal to zero. Denoting the \( i \)th row of matrices \( e \) and \( \tilde{\phi} \) by \( e_i \) and \( \tilde{\phi}_i \), respectively, equation (19) leads to following equations

\[
\langle e_i, \tilde{\phi}_i \rangle = 0_{1 \times m} \quad \forall i, \quad 1 \leq i \leq n \quad \text{(20)}
\]

\[
\langle e_i, \tilde{\phi}_j \rangle = 0 \quad \forall i, j, \quad 1 \leq i \leq n, \quad 1 \leq j \leq m \quad \text{(21)}
\]

Substituting \( e \) from (18) into (19) results in

\[
\langle e, \tilde{\phi} \rangle = W^T \langle \tilde{\phi}, \tilde{\phi} \rangle - \langle \lambda^t, \tilde{\phi} \rangle = 0 \quad \text{(22)}
\]

or

\[
W = \langle \tilde{\phi}, \tilde{\phi} \rangle^{-1} \langle \tilde{\phi}, \lambda^t \rangle \quad \text{(23)}
\]
Eq. (23) is the desired weight update for the training process.

Finally, for the implementation in the discrete-time problem, the integral used in the inner products in (23) can be discretized by evaluating the arguments of the inner products at \( p \) different points in a mesh covering the compact set \( \Omega \). Denoting the distance between the mesh points by \( \Delta \), one has

\[
\langle \phi, \phi \rangle = \lim_{\|\Delta x\| \to 0} \phi \phi^T \Delta x
\]

(24)

\[
\langle \phi, \phi' \rangle = \lim_{\|\Delta x\| \to 0} \phi \phi'^T \Delta x
\]

(25)

where

\[
\phi \equiv \begin{bmatrix} \phi(x_1) & \phi(x_2) & \ldots & \phi(x_p) \end{bmatrix}
\]

(26)

\[
\phi' \equiv \begin{bmatrix} \phi(x_1) & \phi(x_2) & \ldots & \phi(x_p) \end{bmatrix}
\]

(27)

and \( \phi(x_i) \) and \( \phi'(x_i) \) denote matrices \( \phi \) and \( \phi' \) evaluated on the mesh point \( x_i \), respectively.

Using (24) and (25), the weight update (23) can be simplified to the standard least square solution nicely as

\[
W = (\phi\phi^T)^{-1}\phi\phi'^T
\]

(28)

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

Though (28) looks like a one shot solution for the ideal NN weights, the training is an iterative process which needs selecting different random states and times and updating the weights through solving (28) successively. 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 \( \lambda' \) used in the weight update (28) is not the true optimal costate and is a function of current estimation of the ideal unknown weight, i.e. \( \lambda' = \lambda'(W) \). Denoting the weights at the \( i \)th epoch of the weight update by \( W^{(i)} \) the iterative procedure is given by

\[
W^{(i+1)} = (\phi\phi^T)^{-1}\phi\phi'^T(W^{(i)})^T
\]

(29)

Note that

\[
\lim_{i \to \infty} W^{(i)} = W^*
\]

(30)

where \( W^* \) denotes the optimal NN weights in the sense that it generates the optimal costate vector. For this purpose, one starts with an initial weight \( W^{(0)} \) and iterates through (28) 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. Proofs of the assertions made are given next.

### III. Convergence Theorems

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 selected weight update will force the error between the network output and the target to converge to zero and finally that the network weights will converge.

**A. Convergence of the algorithm to the optimal solution**

The proposed algorithm for Finite-SNAC training is based on Dual Heuristic Programming (DHP) or the costate based RL in which starting at an initial value for the costate vector one iterates to converge to the optimal costate (reinforcement learning).

Denoting the iteration index by superscript and the time index by subscript, DHP for finite horizon optimal control starts by an initial value assignment to \( \lambda_k^0 \) 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.
\[ u^i_k = -R^{-1}g(x_k)^T\lambda^i_{k+1} \] (31)

\[ \lambda^{i+1}_k = Qx_k + A(x_k, u^i_k)^T\lambda^i_{k+1} \] (32)

\[ \lambda^i_N = QfN \] (33)

which the last equation is actually the final condition of the optimal control problem, and

\[ A(x_k, u^i_k) \equiv \frac{\partial (f(x_k) + g(x_k)u^i_k)}{\partial x_k} \] (34)

\[ \lambda^{i+1}_{k+1} \equiv \lambda^i(x_{k+1}) = \lambda^i(f(x_k) + g(x_k)u^i_k) \] (35)

The problem is proving that the iterative procedure results in the optimal costate vector \( \lambda^* \) and the optimal control vector \( u^* \). The convergence proof presented in this paper is based on the convergence of Heuristic Dynamic Programming (HDP) or the cost function based RL, in which the parameter subject to evolution is the cost function \( J \) whose behavior is much simpler to discuss compared to that of the costate vector \( \lambda \).

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

\[ J^{i+1}(x_k, k) = \frac{1}{2}x_k^TQx_k + u^i_k^TRu^i_k + f(x_k)u^i_k \] (36)

\[ u^i_k = \text{argmin}_u \left( J^{i+1}(x_k, k) \right) = -R^{-1}g(x_k)^T \frac{\partial J^{i+1}_{k+1}}{\partial x_k} \] (37)

For finite horizon case, the below final condition also should be satisfied in the HDP iterations.

\[ J^{i+1}(x_N, N) = \frac{1}{2}x_N^TQfN \] (38)

Note that \( J_k \equiv J(x_k, k) \) and \( J^{i+1}_k \equiv J^i(f(x_k) + g(x_k)u^i_k, k + 1) \).

**Theorem 1: HDP Convergence**

The sequence of \( J^i \) defined by HDP, i.e. (36) to (38), in case of \( J^0(x_k) = 0 \) converges to the fixed final time optimal solution.

**Theorem 2: DHP Convergence**

Denoting the states and the control vector with new letters \( s \) and \( v \), respectively, consider sequences \( \lambda^i_k \) and \( v^i_k \) defined by equations (39) to (41), where the state equation is \( s_{k+1} = f(s_k) + g(s_k)v^i_k \). Note that \( i \) is the index of iteration and \( k \) is the time index, and \( A(s_k, v_k) \equiv \frac{\partial s_{k+1}}{\partial s_k} = \frac{\partial f(s_k)}{\partial s_k} + \frac{\partial g(s_k)}{\partial s_k} v_k. \)

\[ v^i_k = -R^{-1}g(s^i_k)^T\lambda^i_{k+1} \] (39)

\[ \lambda^{i+1}_k = Qs^i_k + A(s^i_k, v^i_k)^T\lambda^i_{k+1} \] (40)

\[ \lambda^i_N = QfN \] (41)

If \( \lambda^0_k = 0 \) \( \forall k \), then the sequences \( \lambda^i_k \) and \( v^i_k \) will converge to the optimal solution for the given nonlinear affine system, i.e. \( \lambda^i \to \lambda^* \) and \( v^i_k \to v^*_k \) as \( i \to \infty \) where \( \lambda^* \) and \( v^* \) denote the optimal parameters.

The proofs of the above theorems are given in the Appendix.
B. Convergence of the training error and the network weights

After having proved that the training target converges to the optimal solution, the next step is proving the ability of the weight update law to force the error between the network output and the target to converge to zero and the convergence of the weights. The proofs of the theorems are given in the Appendix.

**Theorem 3: Training error convergence**

The weight update (19) will force the error (18) to converge to zero as the number of neurons of the neural networks, $m$, goes to infinity.

**Theorem 4: Neural network weight convergence**

Assume that an ideal set of weights exist, denoted by $W^*$, where

$$
W^* = \sum_{i=1}^{\infty} W_i^T \phi_i
$$

Then, using the weight update (19), one has $W - W_{\text{trunc}}^* \rightarrow 0$ where $W_{\text{trunc}}^*$ is the truncated first $m$ row of the ideal weight $W^*$.

IV. Application of the Controller to the Spacecraft Orbital Maneuver

A. Modeling the spacecraft maneuver

Denoting the displacement vector of the center of mass of a rigid spacecraft from the center of the orbital frame in the destination orbit by $\delta r \equiv [X \ Y \ Z]^T$ where $X$, $Y$ and $Z$ are the components of the vector $\delta r$ in the orbital frame, the non-dimensionalized equation of motion of a spacecraft in the gravity field is given as:

$$
\dot{X} - 2Y + (1 + X)(\frac{1}{r^3} - 1) = u_x
$$

$$
\dot{Y} + 2\dot{X} + Y(\frac{1}{r^3} - 1) = u_y
$$

$$
\dot{Z} + \frac{1}{r^3} Z = u_z
$$

where the dots denote the time-derivative of the parameters, $u_x$, $u_y$, and $u_z$ denote the three components of the non-dimensionalized total force exerted on the spacecraft and $r \equiv \sqrt{(1 + X)^2 + Y^2 + Z^2}$.

Having the equation of motion, one can derive the state equation of the orbital maneuver problem. Selecting the state vector as $x = [X \ Y \ \dot{X} \ \dot{Y} \ \dot{Z}]^T$ and the control vector as $u = [u_x \ u_y \ u_z]^T$, the state equation of the orbital maneuver problem reads:

$$
\frac{d}{dt}\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6
\end{bmatrix} = \begin{bmatrix}
x_4 \\
x_5 \\
x_6 \\
2x_5 + (1 + x_1)(1/r^3 - 1) \\
-2x_4 - x_2(1/r^3 - 1) \\
-x_3(1/r^3)
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
0 \\
u_1 \\
u_2 \\
u_3
\end{bmatrix}
$$

where the components of the state vector are denoted by $x_i$, $1 \leq i \leq 6$, and those of the control vector are denoted by $u_i$, $1 \leq i \leq 3$. This state equation is in the nonlinear input-affine form, suitable for the Finite-SNAC synthesis.

Now, the problem is to apply some optimal control to force the states $x_i$, $1 \leq i \leq 6$, to go to zero in a predetermined and fixed final time. Convergence of the states to zero is equivalent of performing the orbital maneuver and locating at the destination point in the dictated time.

B. Numerical Analysis

The problem is going from the source orbit to the destination orbit, e.g. to meet a space station in the destination orbit. The specifications of the source and destination orbits are given in Table 1 and the mean anomaly of both of the maneuvering and the destination spacecrafts are selected zero at the start of the maneuver. In order to have a three-dimensioned maneuver, the orbits are selected to be in different planes.
Table 1. The spec. of the destination and source orbits

<table>
<thead>
<tr>
<th>The Characteristic</th>
<th>Source Orbit</th>
<th>Destination Orbit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orbit Semi-major axes</td>
<td>10,000 km</td>
<td>10,000 km</td>
</tr>
<tr>
<td>Right Ascension of the Ascending Node</td>
<td>15 deg.</td>
<td>0 deg.</td>
</tr>
<tr>
<td>Inclination</td>
<td>85 deg.</td>
<td>90 deg.</td>
</tr>
<tr>
<td>Eccentricity</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

For the given characteristics, the initial condition is calculated resulting in the normalized initial states of

\[ x_0 = [0.0 \ 0.2588 \ 0.0341 - 0.0038 \ 0.0842 \ 0.0226]^T \]  

(45)

The fixed final time is selected as 3 time units. The time unit is \( t = 1.58 \times 10^3 \text{sec} \) and the reference length is \( R = 10,000 \text{ km} \).\(^{25} \). In this condition, each control unit will be \( R/t^2 = 0.004 \text{ m/sec}^2 \). Note that, this implementation is based on having continuous thrust for actuation. The weight matrices are selected through trial and error as

\[ Q = \text{diag}(10 \ 10 \ 10 \ 1 \ 1 \ 1) \]

\[ Q_f = \text{diag}(2000 \ 2000 \ 2000 \ 200 \ 200 \ 200) \]  

(46)

\[ R = \text{diag}(1 \ 1 \ 1) \]

The values of the elements of \( Q_f \) are selected much higher than those of \( Q \) in order to force the error of the states at the final time to be small.

The basis functions selected for the neural network are following polynomials of the combinations of the network inputs: \( x_i, x_i^2, x_i^3 \) for \( i = 1,2,...,7 \) and \( x_i x_j, x_i x_j x_l, x_i x_j x_l^2, x_i e^{-x_j}, x_i x_j e^{-x_l} \) for \( i, j = 1,2,...,6 \) and \( i \neq j \). Note that \( x_i \) is the \( i \)th input of the neural network and \( x_l \) 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.

This selection leads to the number of basis functions being equal to 96. The network is trained for 200 epochs with number of states selected at each iteration equal to 300 for creating a mesh over the region of interest as explained in the weight update discussion at the end of section II. For this simulation states are selected in such a way which each element belongs to the interval of \((-0.3, 0.3)\). The training took 75 seconds in a typical desktop computer and the weights converged as shown in Fig. 2.

After the convergence of the training, the network was used for simulating the spacecraft maneuver. Histories of the trajectory of the position elements and the velocity elements of the state vector are given in Fig. 3 and 4, respectively, and the control histories are shown in Fig. 5. In order to be able to evaluate the performance of the controller, the numerical solution to the TPBVP has been calculated through a long iterative process for the given initial condition and the results are depicted using red plots in Fig. 3 to 5, while black plots denote those of the Finite-SNAC. As seen through the plots, the proposed controller has been able to force the states to converge to the origin in the fixed final time of 3 units and the states trajectory are close to those of the optimal open loop numerical solution. The cost-to-go of the Finite-SNAC turned out to be only 3% more than the open loop optimal solution while the open loop solution is only applicable to the preselected initial condition and the fixed time-to-go and also suffers from the open loop nature in comparison to the Finite-SNAC.

To evaluate the robustness of the proposed controller under the presence of a disturbance force, the same trained network is used for simulating the same maneuver, where a constant force equal to 0.1 unit is applied in the \( \theta \) direction. The state trajectories of the Finite-SNAC and the open loop solution are shown in Fig. 6. As seen, the open loop solution has not been able to stabilize the system under the disturbance, while the Finite-SNAC has been able to accomplish the maneuver, with a small steady state error. To analyze the disturbance rejection capability of the Finite-SNAC, the control histories of the cases of with and without disturbance of the proposed controller are superimposed and shown in Fig. 7, where the red plots denote the same controller’s result simulated with the disturbance. Interestingly, the value of \( u_y \) in the most part of the simulation is shifted by a value around the negative of the disturbance force to cancel the effect of the applied disturbance on the \( \theta \) direction. This verifies the nice disturbance rejection of the developed controller. Note that this behavior is because of the close-loop nature of the proposed controller, otherwise, the controller is not theoretically designed to evaluate and cancel any disturbance.
To show another capability of the finite-horizon controller, the same network without retraining is used for another maneuver with the same initial condition but with less time-to-go, i.e. 2 time units. The trajectories of the position elements of the state vector and the control histories for the shorter horizon are superimposed on the result of the previous simulation of the Finite-SNAC controller and shown in Fig. 8 and 9, respectively. In these two figures, the black plots denote the results of the maneuver with time-to-go of 3, and the red plots denote those of the maneuver with time-to-go of 2 time units.

As seen in Fig. 9, the controller has applied a different control history on the spacecraft to accomplish the same maneuver in a shorter time. This shows that the network has learned the time-dependency of the optimal control in finite-horizon problems. Fig. 8 shows that the new applied control has been able to perform the maneuver by forcing the spacecraft to travel on another trajectory to do the same maneuver in a shorter time. This different trajectory can be well seen in Fig. 11 which shows the three-dimensional trajectories between the source and the destination orbits for both of the maneuvers.

The developed controller is able to control different initial conditions as long as the resulting state trajectory belongs to the domain on which the network is trained. To evaluate the performance of the controller on different initial conditions, another source orbit is selected with the specifications of semi-major axis of 11000 km, right ascension of -15 degrees and inclination of 95 degrees which leads to the new set of initial conditions given below for the simulation using the same trained network.

$$\mathbf{x}_0 = [0.0 \ -0.2847 \ -0.0625 \ -0.0502 \ -0.0803 \ 0.0215]^T$$

(47)

The resulted position trajectory, called maneuver 2, is super imposed with that of resulted from the previous initial conditions, called maneuver 1, and showed in Fig. 10. As seen, the new maneuver is accomplished in the fixed set time, as well, confirming the applicability of the controller for different initial conditions. In Fig. 12, the three-dimensional trajectories of the maneuvers with different source orbits are shown.

V. Conclusions

A single network neurocontroller, called Finite-SNAC, has been developed in this study. Theorems for convergence of weights and to optimal control have been provided. The proposed NN controller has shown to be able to solve finite-horizon optimal control problems for discrete-time nonlinear control-affine systems. The simulation results show that with a small increase in the number of neurons of the network compared to the previous methods, the controller is able to achieve the desired results without using time varying weights. Moreover, since Finite-SNAC embeds the HJB solutions, the finite-horizon controller once trained for a horizon, will remain optimal for any other horizon shorter than the one it’s trained for. This feature makes the Finite-SNAC extremely versatile for real-life applications.

Figure 2. The histories of some of the elements of the weights matrix during the training iterations
Figure 3. The trajectories of the position elements of the state vector for the time-to-go of 3 units

Figure 4. The trajectories of the velocity elements of the state vector for the time-to-go of 3 units

Figure 5. The histories of the applied controls for the time-to-go of 3 units

Figure 6. The states trajectories for two controllers under the presence of the disturbance force
Figure 7. The Finite-SNAC’s applied controls for the cases of with and without the disturbance force

Figure 8. The trajectories of the position elements of the state vector; Finite-SNAC with different final times

Figure 9. The histories of the applied controls; Finite-SNAC with different final times

Figure 10. The trajectories of the position elements of the state vector for different initial conditions
Figure 11. The three-dimensional trajectories of the maneuvers with different time-to-go’s

Figure 12. The three-dimensional trajectories of the maneuvers with different source orbits
Appendix

A. Convergence of the algorithm to the optimal solution: Proofs

In Ref. 11 the authors have proved that the HDP for infinite-horizon regulation converges to the optimal solution. Here, we modify their proof to cover the case of finite-horizon optimal control. For this purpose, the following four Lemmas are required of which three are cited from Ref. 11 with some modifications to handle the time dependency of the optimal cost function.

**Lemma 1.** Using any arbitrary control sequence of \( \mu_k \in \mathbb{R}^l \), and \( \lambda_k \in \mathbb{R}^m \) defined as

\[
A^{i+1}(x_k, k) = \frac{1}{2}(x_k^T Q x_k + \mu_k^T R \mu_k) + A^i(F(x_k) + g(x_k) \mu_k, k + 1)
\]  

(48)

If \( A^0(x_k, k) = J^0(x_k, k) = 0 \) then \( A^i(x_k, k) \geq J^i(x_k, k) \) \( \forall i \) where \( J^i(x_k, k) \) is iterated through HDP, i.e. (36) and (37).

**Proof:** The proof given in Ref. 11 for infinite horizon case, is applicable here also.

**Lemma 2:** The \( \lambda_k(x_k) \) resulting from HDP, is upper bounded by an existing bound \( X(x_k, k) \).

**Proof:** This proof is inspired by the proof of a similar Lemma in Ref. 11, however, this is an important modification to deal with finite-horizon problems. Let \( \eta_k \in \mathbb{R}^m \) be an arbitrary control. Let \( Z^0(x_k, k) = J^0(x_k, k) = 0 \), where \( Z^i \in \mathbb{R} \) is updated as

\[
Z^{i+1}(x_k, k) = \frac{1}{2}(x_k^T Q x_k + \eta_k^T R \eta_k) + Z^i(x_{k+1}, k + 1)
\]  

(49)

\[
Z^{i+1}(x_N, N) = \frac{1}{2} x_N^T Q f x_N
\]  

(50)

\[
x_{k+1} = f(x_k) + g(x_k) \eta_k
\]  

(51)

Defining the \( Y(x_k, k) \) as

\[
Y(x_k, k) = \frac{1}{2} x_k^T Q f x_N + \sum_{n=0}^{N-k-1} \frac{1}{2}(x_{k+n}^T Q x_{k+n} + \eta_{k+n}^T R \eta_{k+n})
\]  

(52)

Subtracting (52) from (49) results in

\[
Z^{i+1}(x_k, k) - Y(x_k, k) = Z^i(x_{k+1}, k + 1) - \left( \frac{1}{2} x_{k+n}^T Q f x_N + \sum_{n=1}^{N-k-1} \frac{1}{2}(x_{k+n}^T Q x_{k+n} + \eta_{k+n}^T R \eta_{k+n}) \right)
\]  

(53)

which is equivalent of

\[
Z^{i+1}(x_k, k) - Y(x_k, k) = Z^i(x_{k+1}, k + 1) - Y(x_{k+1}, k + 1)
\]  

(54)

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

\[
Z^{i+1}(x_k, k) - Y(x_k, k) = Z^{i-(N-k-1)}(x_N, N) - Y(x_N, N)
\]  

(55)

But the right hand side of (55) is

\[
Z^{i-(N-k-1)}(x_N, N) - Y(x_N, N) = \frac{1}{2} x_N^T Q f x_N - \frac{1}{2} x_N^T Q f x_N = 0 \text{ if } i > N - k - 1
\]  

(56)

\[
Z^0(x_N, N) - Y(x_N, N) = 0 - Y(x_N) < 0 \text{ if } i = N - k - 1
\]  

(57)

Hence, one has

\[
Z^{i+1}(x_k, k) - Y(x_k, k) \leq 0 \text{ if } i \geq N - k - 1
\]  

(58)
For the case of \( i < N - k - 1 \) one has
\[
Z^{i+1}(x_k, k) - Y(x_k, k) = Z^0(x_{k+i+1}, k+i+1) - Y(x_{k+i+1}, k+i+1)
\]
(59)
But, \( Z^0(x_{k+i+1}, k+i+1) = 0 \), hence,
\[
Z^{i+1}(x_k, k) - Y(x_k, k) = 0 - Y(x_{k+i+1}, k+i+1) < 0 \text{ if } i < N - k - 1
\]
(60)
In conclusion, (58) and (60) lead to
\[
Z^i(x_k, k) \leq Y(x_k, k) \quad \forall i
\]
(61)
From Lemma 1 with \( \mu_k = \eta_k \) one has \( f^i(x_k, k) \leq Z^i(x_k, k) \), hence,
\[
f^i(x_k, k) \leq Y(x_k, k)
\]
(62)
which proves Lemma 2.

**Lemma 3**, \(^{11}\): If the optimal control problem can be solved, then there exists a least upper bound \( f^*(x_k, k) \), \( f^*(x_k, k) \leq Y(x_k, k) \), which satisfy HJB equation (8), and \( 0 \leq f^i(x_k, k) \leq f^*(x_k, k) \leq Y(x_k, k) \) where \( Y(x_k, k) \) is defined in Lemma 2.

**Proof:** The proof is given in Ref. 11, where the upper boundedness for the finite-horizon case is given in Lemma 2.

**Lemma 4**, \(^{11}\): The sequence of \( f^i \) defined by HDP, in case of \( f^0(x_k) = 0 \), is non-decreasing.

**Proof:** The proof is given in Ref. 11.

**Proof of Theorem 1:** Similar to Ref. 11, but for finite-horizon case, using the results of Lemma 4 and Lemma 2 one has
\[
f^i \rightarrow f^\infty \text{ as } i \rightarrow \infty.
\]
(63)
From Lemma 3
\[
f^\infty \leq f^*
\]
(64)
Because \( f^\infty \) satisfies the HJB equation (8) and the finite-horizon final condition one has
\[
f^\infty = f^*
\]
(65)
which completes the proof.

**Proof of Theorem 2:** This theorem is DHP version of Theorem 1 and the result of Theorem 1 will be used for proving the convergence of DHP. The idea here is to use induction to show that the evolution of the sequence in DHP is identical to that of HDP. The systems are the same with the same initial conditions. The state vector and control are denoted with another set of letters to provide the ability of comparing them with the HDP approach along the iterations.

From \( f^0(x_k, k) = 0 \) and \( \lambda^0_k = 0 \ \forall k \) it follows that
\[
\lambda^0_k = \frac{\partial f^0(x_k, k)}{\partial x_k} \forall k
\]
(66)
One iteration of the HDP results in
\[
u^0_k = -R^{-1} g(x_k)^T \frac{\partial f^0(x_{k+1}, k+1)}{\partial x_{k+1}}
\]
(67)
\[
f^1(x_k, k) = \frac{1}{2} (x_k^T Q x_k + u^0_k^T R u^0_k) + f^0(x_{k+1}, k+1)
\]
(68)
and one iteration of DHP results in

\[
v_k^0 = -R^{-1} g(s_k)^T \lambda_{k+1}^0
\]

(69)

\[
\lambda_k^1 = Qs_k + A(s_k, v_k^0)^T \lambda_{k+1}^0
\]

(70)

If \(x_k = s_k\), from (66), (67) and (69) it follows that

\[
u_k^0 = v_k^0
\]

(71)

The derivative of (68) with respect to \(x_k\) is

\[
\frac{\partial f^i(x_k,k)}{\partial x_k} = Qx_k + A(x_k, u_k^i)^T \frac{\partial f^i(x_k+1,k+1)}{\partial x_{k+1}}
\]

(72)

Considering (66) and (71), and comparing (72) with (70) result in

\[
\lambda_k^1 = \frac{\partial f^i(x_k)}{\partial s_k}
\]

(73)

Now assume

\[
x_k = s_k
\]

(74)

\[
u_k^i = v_k^i
\]

(75)

\[
\lambda_k^i = \frac{\partial f^i(x_k)}{\partial s_k}
\]

(76)

and perform \(i\)th iteration of HDP and DHP

\[
u_k^i = -R^{-1} g(x_k)^T \frac{\partial f^i(x_k+1)}{\partial x_{k+1}}
\]

(77)

\[
f^{i+1}(x_k,k) = \frac{1}{2} \left( x_k^T Q x_k + u_k^T R u_k \right) + f^i(x_{k+1}, k + 1)
\]

(78)

\[
v_k^i = -R^{-1} g(s_k)^T \lambda_{k+1}^i
\]

(79)

\[
\lambda_{k+1}^i = Qs_k + A(s_k, v_k^i)^T \lambda_{k+1}^i
\]

(80)

The derivative of (78) with respect to \(x_k\) is

\[
\frac{\partial f^{i+1}(x_k,k)}{\partial x_k} = Qx_k + A(x_k, u_k^i)^T \frac{\partial f^i(x_k+1,k+1)}{\partial x_{k+1}}
\]

(81)

Again comparing (81) with (80) and considering (74), (75) and (76) results in

\[
\lambda_{k+1}^i = \frac{\partial f^{i+1}(x_k,k)}{\partial s_k}
\]

(82)

American Institute of Aeronautics and Astronautics
Hence, using iterative equations (36) to (38) for calculating \( u_k^i \) and \( f^i \) and equation (39) to (41) for calculating \( v_k^i \) and \( \lambda_k^i \), it is proved that equation (75) and (76) are valid for all \( i \). Since \( u_k^i \) and \( f^i \), based on Theorem 1, converge to the optimal values, \( v_k^i \) and \( \lambda_k^i \) will also converge to the optimal control and costates, and the proof is complete. □

B. Convergence of the training error and the network weights: Proofs

The main idea behind proofs of Theorem 3 and 4 are from Ref. 23, but, since there are some differences between the error equation and its dimension in this paper compared to Ref. 23 the proofs are different and given here.

**Proof of Theorem 3:** Using Lemma 5.2.9 from Ref. 23, assuming \( \Phi \) being orthonormalized, rather than being linearly independent, does not change the convergence result of the weight update. Assume \( \Phi \) is a matrix formed by \( m \) orthonormal basis functions \( \phi_j \) as its rows where \( 1 \leq j \leq m \) among the infinite number of orthonormal basis functions \( \{\phi_j\}_1^\infty \). The orthonormality of \( \{\phi_j\}_1^\infty \) implied that if a function \( \psi(.) \) belongs to \( \text{span}\{\phi_j\}_1^\infty \) then

\[
\psi = \sum_{j=1}^\infty \langle \psi, \phi_j \rangle \phi_j
\]  

and for any \( \epsilon \) one can select \( m \) sufficiently large to have

\[
\| \sum_{j=m+1}^\infty \langle \psi, \phi_j \rangle \phi_j \| < \epsilon
\]

From (19) one has

\[
\langle e, \phi_j \rangle = 0 \quad \forall j, \quad 1 < j < m
\]

and from (18)

\[
\langle e, \phi_j \rangle = W^T \langle \phi, \phi_j \rangle - \langle \lambda, \phi_j \rangle
\]

which is equivalent of

\[
\langle e, \phi_j \rangle = \sum_{i=1}^m W_i^T \langle \phi_i, \phi_j \rangle - \langle \lambda, \phi_j \rangle
\]

where \( W_i \) is the \( i \)th row of weight matrix \( W \).

On the other hand, one can expand the error \( e \) using the orthonormal basis functions \( \{\phi_j\}_1^\infty \).

\[
e = \sum_{j=1}^\infty \langle e, \phi_j \rangle \phi_j
\]

Inserting (87) into (88) results in

\[
e = \sum_{j=m+1}^\infty \langle e, \phi_j \rangle \phi_j
\]

But, from the weight update (85), the right hand side of (87) is also equal to zero. Applying this to (89) results in

\[
e = \sum_{j=m+1}^\infty \langle \lambda, \phi_j \rangle \phi_j
\]

Due to the orthonormality of the basis functions, one has

\[
\langle \phi_i, \phi_j \rangle = 0 \quad \forall i \neq j
\]

Hence, (90) simplifies to

\[
e = -\sum_{j=m+1}^\infty \langle \lambda, \phi_j \rangle \phi_j
\]

Using (84) for \( \psi = \lambda \), as \( m \) increases, \( e \) decreases to zero.
This completes the proof.

**Proof of Theorem 4:** The training error is defined as

$$e \equiv \bar{x} - \bar{x}^T$$

(94)

Hence,

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

(95)

Note that $\bar{\phi}$ is a matrix formed by 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 (95) by $\bar{\phi}$ results in

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

(96)

The last term in the right hand side of above equation vanishes due to orthonormality of the basis functions. Considering $\langle \bar{\phi}, \bar{\phi} \rangle = I$, (95) simplifies to

$$\langle e, \bar{\phi} \rangle = W^T - W_{\text{trunc}}^*$$

(97)

Looking at (97), the weight update implies the left hand side to be zero, hence, using the weight update (19) one has $W \rightarrow W_{\text{trunc}}^*$.

**Acknowledgments**

Dr. S.N. Balakrishnan, my advisor, is acknowledged for his guidance and support in this research.

**References**


American Institute of Aeronautics and Astronautics


