2013

Approximate dynamic programming based solutions for fixed-final-time optimal control and optimal switching

Ali Heydari

Department: Mechanical and Aerospace Engineering

APPROXIMATE DYNAMIC PROGRAMMING BASED SOLUTIONS FOR FIXED-FINAL-TIME OPTIMAL CONTROL AND OPTIMAL SWITCHING

by

ALI HEYDARI

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

MECHANICAL ENGINEERING

2013

Approved

S. N. Balakrishnan, Advisor
Jagannathan Sarangapani
Robert G. Landers
Douglas A. Bristow
Sanjay K. Madria

To my wife, Maryam

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following 8 articles:

Paper 1, Heydari, A. and Balakrishnan, S. N., "Finite-Horizon Control-Constrained Nonlinear Optimal Control Using Single Network Adaptive Critics," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 24, No. 1, pp. 145-157, 2013.

Paper 2, Heydari, A. and Balakrishnan, S. N., "Fixed-final-time Optimal Control of Nonlinear Systems with Terminal Constraints," accepted for publication in *Neural Networks*, 2013.

Paper 3, Heydari, A. and Balakrishnan, S. N., "Global Optimality of Approximate Dynamic Programming and its use in Non-convex Function Minimization," to be submitted to *Automatica.*

Paper 4, Heydari, A. and Balakrishnan, S. N., "Optimal Multi-therapeutic HIV Treatment Using a Global Optimal Switching Scheme," *Applied Mathematics and Computation*, Vol. 219, pp. 7872-7881, 2013.

Paper 5, Heydari, A. and Balakrishnan, S. N., "Optimal Switching and Control of Nonlinear Switched Systems Using Approximate Dynamic Programming," submitted to *IEEE Transactions on Neural Networks and Learning Systems*.

Paper 6, Heydari, A. and Balakrishnan, S. N., "Optimal Switching between Autonomous Subsystems," revision requested for publication in *Journal of the Franklin Institute*.

Paper 7, Heydari, A. and Balakrishnan, S. N., "Optimal Switching between Controlled Subsystems with Free Mode Sequence," to be submitted to *Neural Networks*.

Paper 8, Heydari, A. and Balakrishnan, S. N., "Optimal Switching of Nonlinear Systems with Modeling Uncertainty," to be submitted to *Journal of the Franklin Institute*.

ABSTRACT

Optimal solutions with neural networks (NN) based on an approximate dynamic programming (ADP) framework for new classes of engineering and non-engineering problems and associated difficulties and challenges are investigated in this dissertation. In the enclosed eight papers, the ADP framework is utilized for solving fixed-final-time problems (also called terminal control problems) and problems with switching nature. An ADP based algorithm is proposed in Paper 1 for solving fixed-final-time problems with soft terminal constraint, in which, a single neural network with a single set of weights is utilized. Paper 2 investigates fixed-final-time problems with hard terminal constraints. The optimality analysis of the ADP based algorithm for fixed-final-time problems is the subject of Paper 3, in which, it is shown that the proposed algorithm leads to the global optimal solution providing certain conditions hold. Afterwards, the developments in Papers 1 to 3 are used to tackle a more challenging class of problems, namely, optimal control of switching systems. This class of problems is divided into problems with fixed mode sequence (Papers 4 and 5) and problems with free mode sequence (Papers 6 and 7). Each of these two classes is further divided into problems with autonomous subsystems (Papers 4 and 6) and problems with controlled subsystems (Papers 5 and 7). Different ADP-based algorithms are developed and proofs of convergence of the proposed iterative algorithms are presented. Moreover, an extension to the developments is provided for online learning of the optimal switching solution for problems with modeling uncertainty in Paper 8. Each of the theoretical developments is numerically analyzed using different real-world or benchmark problems.

ACKNOWLEDGMENTS

First of all, I would like to thank God for blessing me with the bests in almost every aspect of my life, and would like to show respect toward the country of the United States of America for providing me with endless opportunities for prosperity and success. Also, I would like to thank my PhD advisor, Curators' Professor of Aerospace Engineering, Dr. S. N. Balakrishnan, for admitting me to his research group four years ago and exposing me to different research fields and topics. Besides different things he taught me, the motivation and confidence he gave me are the priceless assets of mine and I truly appreciate them. Moreover, I would like to thank Drs. Robert Landers and Douglas Bristow from Mechanical Eng. Dept. and Drs. Jagannathan Sarangapani and Sanjay Madria from Electrical and Computer Eng. and Computer Science Depts. for accepting to be on my defense committee. I would also like to acknowledge the great instructors that I had at Missouri S&T, including Drs. Balakrishnan (State Estimation), Sarangapani (Neural Networks for Control), and Landers (Manufacturing Processes), as well as Drs. Levent Acar (Linear Systems) and David Grow (Real Analysis). I learned a lot from all of my instructors, but, these five professors not only taught me the respective topics, but also provided me with models of excellence in effective teaching and communication with students. Last but not least, my greatest appreciation goes out to Maryam who accepted to be my partner in the journey of life, several years ago and provided me with an exemplary support throughout the difficult days of my education. It is not an exaggeration if I say without my wife's support this work would not have been possible.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Page

Paper 1

Paper 2

Paper 3

LIST OF TABLES

Page

1. INTRODUCTION

1.1. OVERVIEW

Optimal control of dynamical systems is, by nature, desirable as compared to any other control method. The reason lies in providing a solution which 'optimizes' a performance index. Features of such solutions are, for example, minimizing the consumed energy while fulfilling the goal (e.g., path planning for an aircraft), minimizing deviations from a set point (e.g., in the cruise phase of the flight when the aircraft is on auto-pilot), and minimizing terminal errors (e.g., smooth automatic landing of an aircraft on the runway). However, it is very challenging and sometimes impossible to find the optimal controller in a feedback form for problems with nonlinear dynamics and/or complexities including fixed-final-time, constraints on the control/states, and switching nature. Considering the excellent potential of approximate dynamic programming (ADP) framework in circumventing the problem of *curse of dimensionality* existing with the dynamic programming approach to optimal control problems [1]-[10], the ADP is used in this dissertation to solve the following two classes of problems.

1.1.1. Optimal Control with Fixed-final-time. Many control, guidance, and path planning problems are classified as 'terminal control' problems [11]. A terminal control problem is a *finite-horizon* problem with *soft* or *hard* constraints on the terminal states. In other words, in terminal control problems the goal is supposed to be achieved in a finite time. Examples of such problems are having an airplane to land at a given point on the runway, a missile to hit the target, or a spacecraft to maneuver and position strictly at a given point and time to dock with another spacecraft.

1.1.2. Optimal Control of Switching Systems. Switching systems are comprised of several subsystems or modes, in which at each time instant, only one of the subsystems is engaged, e.g., cars with manual transmission system. Many real-world problems, from aerospace field to chemical processes, are categorized as switching systems [12]-[16]. In such systems, the optimal control is not only a 'control input' to be applied on the system, but also a 'switching schedule' to switch between the subsystems at the 'best' times. The main issue is finding optimal switching instants, and once they are found, the problem reduces to a conventional optimal control problem.

1.2. BACKGROUND

Within the last two decades many researchers have focused on using ADP for solving different classes of problems emerging in different real-world systems [1]-[10]. ADP can be divided into two main classes, a) Heuristic Dynamic Programming (HDP) and b) Dual Heuristic Programming (DHP) [1]. 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-to-go function with respect to the states, i.e. the costate vector is learnt by the neural networks [3]. The convergence proof of DHP for linear systems is presented in [4] and that of HDP for general case is presented in [5]. While [3]-[10] deal with discrete-time systems, some researchers have recently focused on continuous time problems, [17]-[19].

ADP is usually carried out using a two-network synthesis called adaptive critic (AC) [2], [3]. In the heuristic dynamic programming (HDP) class with ACs, one network, called the 'critic' network, inputs the states to the NN and outputs the optimal cost and another network, called the 'action' network, outputs the control with states of the system as its inputs [5], [6]. In the dual heuristic programming (DHP) formulation, while the action network remains the same as the HDP, the critic network outputs the costates with the current states as inputs [2], [7], [8]. The single network adaptive critics (SNAC) architecture developed in [9] is shown to be able to eliminate the need for the second network and perform DHP using only one network. Similarly, the J-SNAC eliminates the need for the action network in an HDP scheme [10]. Note that these developments in the neural network literature have mainly addressed only *conventional* optimal control problems with *infinite-horizon*, i.e., regulator type problems.

1.2.1. Terminal Control Problems. One approach to solving terminal control problems of nonlinear systems is formulating the problem in an optimal control framework. For this class of problems, the Hamilton-Jacobi-Bellman (HJB) equation is very difficult to solve since the solution is time-dependent. An open loop solution is dependent on the selected initial condition (IC) and the time-to-go [20]. Available methods for solving terminal control problems can be classified as classical and intelligent control methods.

1.2.1.1. Classical approach to terminal control problems. One approach in classical methods is calculating the open loop solution through a numerical method, e.g., the shooting method and then using techniques like Model Predictive Control for closing the control loop as done in [21]. A limitation of this approach is the fact that it holds only for one set of specified IC and time-to-go. Another method, called the Approximate Sequence of Riccati Equation (ASRE) developed in [22] provides a closed form solution to the problem but again only for a pre-specified IC and time-to-go. This method is based on the calculation of a sequence of Riccati equations until they converge, and then using the converged result for control calculation. Finite-horizon State Dependent Riccati Equation (Finite-SDRE) method, developed in [23] and [24], offers a suboptimal closed form solution to this class of problems. Finite-SDRE provides solutions for different ICs and final times in real-time and shows a lot of potential in the applications, but can accommodate only soft terminal constraints.

Series-based solutions to the optimal control problem with hard terminal constraints were investigated in [25]-[27]. In [25], a closed form solution was found by using a Taylor series expansion of the cost-to-go function. Series-based methods are suitable for systems whose dynamics are given in a polynomial form and comprise only weak nonlinearities. The series can diverge for problems with a large nonlinearity. This limitation motivated the authors of [26] to propose a divide-and-conquer scheme. This scheme is based on determining some waypoints to split the main problem into several simpler problems for which the series based method does not produce significant midcourse errors. However, this method requires some extra numerical optimization to find suitable waypoints for each IC. Moreover, the number of required waypoints needs to be selected through trial and error in order to avoid divergence. The generating functions method proposed in [27] is a different series-based solution where the terminal constraint is a given point. In [28] a Generalized Hamilton-Jacobi-Bellman equation [29] was used with some modifications. Convergence of this method was proved for the unconstrained case in [29], but not for the constrained problem.

1.2.1.2. Intelligent approach to terminal control problems. The use of intelligent control for solving finite-horizon optimal control problems was considered in [30]-[36]. Authors of [30] developed a neurocontroller for a problem with state

constraints using AC scheme with time dependent weights. This controller is developed for an agile missile maneuver. It is however, a scalar problem wherein the final state and the control have a direct relationship. Hence, in a discrete formulation, the final state can be achieved from any state at the previous step. Continuous-time problems are considered in [31] where the time-dependent weights are calculated through a backward integration of the HJB equation. The finite-horizon problem with *unspecified terminal time* and a fixed terminal state was considered in [33]-[36]. The algorithms developed in these papers lead to an *infinite* sequence of controls. Therefore, the control needs to be applied for an infinite time horizon to optimize the cost-function and bring the states to the origin. To overcome this problem, the authors suggested truncating the control sequence in order to end up with the so called ϵ-optimal solution, which will hence, have a finite horizon. The truncation is done such that the remaining horizon is long enough in order for the cost-to-go truncation error to be less than a given $\epsilon > 0$. Moreover, the neurocontrollers developed in [33] and [34] can only control one IC, and once the IC is changed, the network needs to be re-trained to give the optimal solution for the new IC. The neurocontrollers in [34] and [35] require the system to be such that the state can be brought to the origin in one step, from any given state. Systems with invertible input gain matrices in a control-affine discrete-time form satisfy this requirement. A newly developed controller in [36] has removed the restrictions of fixed initial condition and being able to go to the origin in one step.

1.2.2. Optimal Switching Problems. Methods developed so far for finding the optimal solution to switching systems can be mainly divided into two groups; nonlinear programming based methods and discretization based methods.

1.2.2.1. Nonlinear programming based approach to switching problems. The first group is comprised of nonlinear programming based methods [37-42], in which through different schemes, the gradient of the cost with respect to the switching instants/points are calculated and then by using a nonlinear optimization method, e.g., steepest descent, the switching instants/points are adjusted to find the *local* optimum. It should be noted that in many existing papers, the sequence of active subsystems, called mode sequence, is selected a priori [37-41], and the problem reduces to finding the switching instants between the modes. In [42], the first and last subsystems are pre-

selected and a search is done initially to find all the possible mode sequences and for every such sequence, the optimal switching instants are calculated using nonlinear programming.

1.2.2.2. Discretization based approach to switching problems. The second group includes studies that discretize the problem in order to deal with a *finite* number of options. Having a finite number of candidate switching time sequences, authors of [43] utilize a direct search to evaluate the cost function for different randomly selected switching time sequences and select the best one in the sense of having less corresponding cost. In [44] the discretization of the state and input spaces is used for calculation of the value function for optimal switching through dynamic programming.

1.3. CONTENTS AND CONTRIBUTIONS OF THIS DISSERTATION

This dissertation is composed of eight research papers with the main theme of developing new ADP-based algorithms and methods for solving difficult problems in controls. Considering the literature survey presented in Subsection 1.2, the contents of the papers and their contribution as well as their comparison with the state-of-the-art are discussed in here.

1.3.1. Paper 1: Fixed-final-time Problems with Soft Terminal Constraints. A single neural network based solution with a single set of weights, called Finite-horizon Single Network Adaptive Critics (Finite-SNAC), is developed in Paper 1, to provide a comprehensive solution to fixed-final-time optimal control problems with input-affine nonlinear systems. The offline trained network can be used to generate *online feedback control* for different ICs. Furthermore, a major advantage of the proposed technique 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 [45], is used in this study. Specifically, in this paper an ADP based controller for control-constrained finite-horizon optimal control of 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.

Comparing the developed controller in this paper with the available controllers in the literature, the closest one is [30]. The difference between this study and the [30] is

developing a controller which uses only one network and one set of weights for the purpose, as well as providing comprehensive *convergence proofs*. Despite [31] and [32] the Finite-SNAC solves discrete-time problems and uses ADP to do so. Finally, [33]-[36] solves unspecified terminal time problems while Finite-SNAC solves the problems with given final times.

1.3.2. Paper 2: Fixed-final-time Problems with Hard Terminal Constraints.

The first part of the development in Paper 2 consists of formulating an approximate dynamic programming (ADP) based neurocontroller for fixed-final-time optimal control of systems with a *soft* terminal constraint. The main difference between Paper 1 and the proposed scheme in the first part of Paper 2 is the use of the cost-function based ADP, i.e., HDP. The development in Paper 1 is the costate based ADP, i.e., DHP. Another difference which leads to major changes in the training algorithm proposed in Paper 2 with respect to the one given in Paper 1, is using NNs with time varying weights in Paper 2 to accommodate the time-dependency of the solution. After discussing the solution to the problem with soft constraints, some modifications are performed in the network structure and the training algorithm to handle *hard* terminal constraints. These modifications are the main contributions of Paper 2. Another contribution of this study is proving the convergence of the network weights through a novel idea. It is done for the selected linear in the weights NN by showing that the successive approximation based weight update is a contraction mapping [46] within the compact domain of interest.

As compared with [30], the controller developed in Paper 2 can be used in a *multivariable* setting while the method presented in [30] is developed for the scalar dynamics of an agile missile. Neurocontrollers developed in [31]-[32] do not admit hard terminal constraint, which is the main contribution of this study. Finally, the method here does not have the restrictions in [33]-[36] as the need for truncating the control in order to end up with a finite-horizon solution and also the requirement of the terminal constraint being a ‘point’. The proposed technique can handle terminal constraints that are a point, a curve, or a surface which can be a nonlinear function of the state space elements. Moreover, the selected approach in this study directly results in a finite sequence of controls, hence, no truncation is required.

The trained (linear in the weights) NN in this Paper 2 offers a feedback solution though trained offline. Furthermore, notable features of the proposed technique include: a) Optimal control of any set of initial conditions in a compact set, as long as the resulting state trajectory lies within the domain on which the network is trained. b) Optimal control for any final time not greater than the final time for which the network is trained (Bellman principle of optimality [20]). c) Providing optimal control in a closed form versus the terminal surface/curve/point. Therefore, if, for example, the terminal point is changed, no retraining is needed for the network to give optimal solution for the new terminal point. Interested readers are referred to [47] for an application of the method developed in Paper 2 in solving spacecraft rendezvous problems.

1.3.3. Paper 3: Proof of Global Optimality of ADP for Fixed-final-time Problems. Despite much published literature on adaptive critics, there still exists an open question about the nature of optimality of the adaptive critic based results. Are they locally or globally optimal? A contribution of Paper 3 is in proving that the AC based solutions are globally optimal subject to the assumed basis functions. To help with the development of the proof, the ADP based algorithm for solving fixed-final-time problems developed in Paper 1 and Paper 2, included in this dissertation, is revisited first. Afterwards, a novel analysis is presented on global optimality of the result in Paper 3. It is shown that selecting any cost function with quadratic control penalizing term, if the sampling time used for discretization of the original continuous-time system is small enough, the resulting cost-to-go function will be *convex* versus the control at the current time and hence, the first order necessary optimality condition [48] will lead to the *global* optimal control. The second contribution of this paper is in showing that the ADP can be used for functional optimization, specifically, optimization of non-convex functions. Finally, through analytical and numerical discussions, it is shown that despite the gradient based methods, selecting any initial guess on the minimum and updating the guess using the control resulting from the actor, the states will move *directly* toward the global minimum, passing any existing local minimum in the path.

1.3.4. Papers 4 and 5: Switching Problems with Fixed Mode Sequence. All the cited methods in the literature of optimal switching *numerically* find the optimal switching time for *a specific initial condition*; each time the initial condition is changed,

new computations are needed to find the new optimal switching instants. If the function that describes the optimal cost-to-go for every given switching time sequence is known explicitly, then the optimal switching problem simplifies to minimization of the function with respect to the switching instants. However, even in the case of general linear subsystems with a quadratic cost function, this function is not available [38,49]. The main contributions of Paper 4 and 5 are developing algorithms for switching problems, respectively with autonomous and controlled subsystems, that learns the optimal cost-to-go as a function of current state and the switching instants. An ADP based scheme, in an HDP form is used to train an NN to learn the nonlinear mapping between the optimal cost-to-go and the switching instants. Once this function is learned, finding the optimal switching times reduces to minimization of an analytical function. Furthermore, a second NN is trained in Paper 5 along with to generate optimal control in a feedback form. Hence, once the optimal switching instants are calculated, one may use the control NN to generate the optimal control to be applied on the system.

As compared to available methods in the literature, the proposed technique has two advantages. They are: 1) the method developed in this paper gives *global* optimal switching instants versus local ones resulting from nonlinear programming based methods, 2) the learned function gives the optimal cost-to-go based on the switching instants for *a vast domain of initial conditions*; hence, optimal switching times for different initial conditions can easily be calculated using the *same* trained NNs. Moreover, once the optimal switching instants are calculated, the method developed in Paper 5 provides feedback optimal control, too. Convergence of the learning process is also provided.

1.3.5. Papers 6 and 7: Switching Problems with Free Mode Sequence. In Papers 6 and 7, two methods based on ADP are developed for solving optimal switching problems with *free mode sequence*, for autonomous and controlled subsystems, respectively. The idea is as simple as learning the optimal cost-to-go and the optimal control for different active modes. It is shown that having these functions the optimal mode can be found in a *feedback* form, i.e., as a function of the instantaneous state of the system and the remaining time. The real-time calculation of the optimal mode differentiates Papers 6 and 7 from Papers 4 and 5. Note that, after the training phase, the

methods presented in Papers 4 and 5 require an offline function minimization based on the selected initial condition, in order to end up with the optimal switching times.

The method developed in Papers 6 and 7 have several advantages over existing developments in the field: a) They provide *global* optimal switching (subject to the assumed neural network structure) unlike the nonlinear programming based methods which could provide only local optimal solution. b) The order of active subsystems and the number of switching are free. c) The neurocontroller determines optimal solution for *unspecified initial conditions*, without needing to retrain the networks. d) Once trained, the neurocontroller gives solution to *any other final time* as well, as long as the new final time is not greater than the final time for which the network is trained. e) The switching is scheduled in a feedback form, hence, it has the inherent robustness of feedback solutions in moderate disturbance rejection. f) The proposed method provides optimal *control* as well as optimal *switching* schedule for the control of the systems in Paper 7.

1.3.6. Paper 8: Switching Problems with Modeling Uncertainty. To the best of author's knowledge, the available switching developments in the literature require a *perfect model* of the system ahead of the implementation time, for calculation of the solution. In practice, however, modeling uncertainties are ubiquitous. This fact gives rise to the need for developing a scheme for *online* calculation of the optimal switching schedule based on the *actual* dynamics of the subsystems. This problem is investigated in Paper 8. In order to extend the switching scheme developed in Paper 6 to systems with modeling uncertainty, the idea proposed in Paper 6 for *finite-horizon* optimal switching is extended to *infinite-horizon* problems initially. Afterwards, an online training phase is proposed for capturing the effect of unmodeled dynamics on the cost-to-go approximator and also for identifying the unmodeled dynamics, motivated by the work in [50] for conventional optimal control problems. In other words, an NN is trained offline based on imprecise models of the subsystems and then it is utilized in the online operation of the system in which, the actual dynamics of the subsystems are captured and the network is re-trained based on the system's output to generate the optimal cost-to-go and hence, the optimal switching schedule. Besides solving the problems with modeling uncertainty, an important feature of this method is providing solution for different initial conditions.

Moreover, the mode sequence and the number of switching are subject to be determined optimally.

1.4. REFERENCES

[1] P. J. Werbos, "Approximate dynamic programming for real time control and neural modeling", in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. New York: Van Nostrand Reinhold, 1992, pp. 493-525.

[2] S. N. Balakrishnan and V. Biega, "Adaptive-critic-based neural networks for aircraft optimal control", *Journal of Guidance Control and Dynamics*, Vol. 19, No. 4, pp. 893-898, Jul.-Aug. 1996.

[3] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw*., Vol. 8, No. 5, pp. 997-1007, Sep. 1997.

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

[5] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf , "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Trans. Syst. Man Cybern. B*, Vol. 38, No. 4, pp. 943-949, Aug. 2008.

[6] T. Dierks and S. Jagannathan, "Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update," *IEEE Trans. Neural Netw. Learning Syst*., Vol. 23, No. 7, pp. 1118-1129, Jul. 2012.

[7] S. Ferrari, and R. F. Stengel, "Online adaptive critic flight control," *Journal of Guidance Control and Dynamics,* Vol. 27, No. 5, pp. 777-786, Sep.-Oct. 2004.

[8] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw*., Vol. 13, No. 3, pp. 764-773, May 2002.

[9] 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, No. 10, pp.1648–1660, Dec. 2006.

[10] J. Ding and S. N. Balakrishnan, "An online nonlinear optimal controller synthesis for aircraft with model uncertainties," in Proc. *AIAA Guidance, Navigation, and Control Conference*, 2010.

[11] A. E. Bryson, and Y. C. Ho, *Applied Optimal Control*, Taylor & Francis, New York, 1975, pp. 148–164.

[12] M. Rinehart, M. Dahleh, D. Reed, and I. Kolmanovsky, "Suboptimal control of switched systems with an application to the DISC engine," *IEEE Transactions on Control Systems Technology*, Vol. 16 (2), pp.189-201, 2008.

[13] K. Benmansour, A. Benalia, M. Djemaï, and J. de Leon, "Hybrid control of a multicellular converter," *Nonlinear Analysis: Hybrid Systems*, Vol. 1, No. 1, pp.16-29, 2007.

[14] E. A. Hernandez-Vargas, R. H. Middleton, P. Colaneri, F. Blanchini, "Dynamic optimization algorithms to mitigate HIV escape," Proc *IEEE Conference on Decision and Control*, pp.827-832, 2010.

[15] Z. Gong, C. Liu, E. Feng, L. Wang, Y. Yu, "Modelling and optimization for a switched system in microbial fed-batch culture," *Applied Mathematical Modelling*, Vol. 35, No. 7, pp.3276-3284, 2011.

[16] M. Soler, A. Olivares, and E. Staffetti, "Framework for aircraft trajectory planning toward an efficient air traffic management," *Journal of Aircraft*, Vol. 49 (1), pp.985-991, 2012.

[17] D. Vrabie, O. Pastravanu, F. Lewis, and M. Abu-Khalaf, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, Vol. 45, No. 2, pp. 477-484, Feb. 2009.

[18] K. Vamvoudakis and F. Lewis, "Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, Vol. 46, No. 5, pp. 878-888, May 2010.

[19] T. Dierks and S. Jagannathan, "Optimal control of affine nonlinear continuous-time systems" in Proc. *American Control Conf.*,2010, pp. 1568-1573, 2010.

[20] D. E. Kirk, *Optimal Control Theory: An Introduction*, Dover Publications, 2004 pp. 54-94,329-330.

[21] J. Liang, *Optimal Magnetic Attitude Control of Small Spacecraft*, PhD Thesis, Utah State University, Logan, 2005.

[22] T. Cimen and S. P. Banks, "Global Optimal Feedback Control for General Nonlinear Systems with Nonquadratic Performance Criteria," *Systems & Control Letters*, Vol. 53, 2004, pp. 327 – 346.

[23] A. Heydari and S. N. Balakrishnan, "Path Planning Using a Novel Finite-Horizon Suboptimal Controller," *Journal of Guidance, Control, and Dynamics*, in press, 2013.

[24] A. Heydari and S. N. Balakrishnan, "Closed-Form Solution to Finite-Horizon Suboptimal Control of Nonlinear Systems," revision requested for publication in *International Journal of Robust and Nonlinear Control*.

[25] S. R. Vadali and R. Sharma, "Optimal Finite-time Feedback Controllers for Nonlinear Systems with Terminal Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 4, 2006, pp. 921-928.

[26] R. Sharma, S. R. Vadali, and J. E. Hurtado, "Optimal Nonlinear Feedback Control Design Using a Waypoint Method," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 3, 2011, pp. 698-705.

[27] C. Park, V. Guibout, and D. J. Scheeres, "Solving Optimal Continuous Thrust Rendezvous Problems with Generating Functions," *Journal of Guidance, Control, & Dynamics*, Vol. 29, No. 2, 2006, pp. 321-331.

[28] M. Bando and H. Yamakawa, "New Lambert Algorithm Using the Hamilton–Jacobi–Bellman Equation," *Journal of Guidance, Control, & Dynamics*, Vol. 33, No. 3, 2010, pp. 1000-1008.

[29] R. Beard, *Improving the Closed-Loop Performance of Nonlinear Systems*, Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, USA, 1995.

[30] D. Han and S. N. Balakrishnan, "State-constrained agile missile control with adaptive-critic-based neural networks," *IEEE Trans. Control Systems Technology*, Vol. 10, No. 4, 2002, pp. 481-489.

[31] T. Cheng, F. L. Lewis, and M. Abu-Khalaf, "A neural network solution for fixed-final time optimal control of nonlinear systems," *Automatica*, Vol. 43, 2007, pp. 482-490.

[32] D. M. Adhyaru, I. N. Kar, and M. Gopal, "Fixed final time optimal control approach for bounded robust controller design using Hamilton-Jacobi-Bellman solution," *IET Control Theory Appl.*, Vol. 3, No. 9, pp. 1183-1195, Sep. 2009.

[33] F. Wang, N. Jin, D. Liu, and Q. Wei, "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw*., Vol. 22, No. 1, pp. 24-36, 2011.

[34] R. Song and H. Zhang, "The finite-horizon optimal control for a class of time-delay affine nonlinear system," *Neural Comput & Applic*, DOI 10.1007/s00521-011-0706-3, 2011.

[35] Q. Wei and D. Liu, "Finite horizon optimal control of discrete-time nonlinear systems with unfixed initial state using adaptive dynamic programming," *J Control Theory Appl*, Vol. 9, No. 3, pp. 381–390, 2011.

[36] Q. Wei and D. Liu, "An iterative ϵ-optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state," *Neural Networks*, Vol. 32, pp. 236-244, 2012.

[37] X. Xu and P. J. Antsaklis, "Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions," *International Journal of Control*, Vol. 75, No. 16, pp. 1406-1426, 2002.

[38] X. Xu and P. J. Antsaklis, "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Trans. on Automatic Control*, Vol. 49, No. 1, pp.2- 16, 2004.

[39] M. Egerstedt, Y. Wardi, and H. Axelsson, "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. on Automatic Control*, Vol. 51, No. 1, pp.110-115, 2006.

[40] M. Kamgarpoura and C. Tomlin, "On optimal control of non-autonomous switched systems with a fixed mode sequence," *Automatica*, Vol. 48, pp.1177–1181, 2012.

[41] R. Zhao and S. Li, "Switched system optimal control based on parameterizations of the control vectors and switching instant," Proc. *Chinese Control and Decision Conference*, pp. 3290-3294, 2011.

[42] J. Xu and Q. Chen, "Optimal control of switched hybrid systems," Proc. *8th Asian Control Conference*, Kaohsiung, Taiwan, 2011.

[43] R. Luus and Y. Chen, "Optimal switching control via direct search optimization," *Asian Journal of Control*, Vol. 6, No. 2, pp. 302-306, 2004.

[44] M Rungger and O. Stursberg, "A numerical method for hybrid optimal control based on dynamic programming," *Nonlinear Analysis: Hybrid Systems*, Vol. 5, No. 2, pp.254–274, 2011.

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

[46] H. Khalil, *Nonlinear Systems*, 3rd ed., Prentice Hall, New Jersey, 2002, pp. 653-656.

[47] A. Heydari and S. N. Balakrishnan, "Adaptive Critic Based Solution to an Orbital Rendezvous Problem," *Journal of Guidance, Control, and Dynamics*, in press, 2013.

[48] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999, pp. 1-187.

[49] W. Zhang, J. Hu, and A. Abate, "On the value functions of the discrete-time switched LQR problem," *IEEE Trans. on Automatic Control*, Vol. 54, No. 11, pp.2669-2674, 2009.

[50] N. Unnikrishnan, and S. N. Balakrishnan, "Dynamic reoptimization of a missile autopilot controller in presence of unmodeled dynamics," *Proc. AIAA Guidance, Navigation, and Control Conference*, pp.1-15, 2005.

PAPER

1. FINITE-HORIZON CONTROL-CONSTRAINED NONLINEAR OPTIMAL CONTROL USING SINGLE NETWORK ADAPTIVE CRITICS

Ali Heydari and S. N. Balakrishnan

ABSTRACT

To synthesize fixed-final-time control-constrained optimal controllers for discrete-time nonlinear control-affine systems, a single neural network based controller called the Finite-SNAC is developed in this study. 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 optimal feedback control. Control constraints are handled through a non-quadratic cost function. Convergence proofs of a) the reinforcement learning based training method to the optimal solution, b) the training error, and c) the network weights are provided. The resulting controller is shown to solve the associated time-varying Hamilton-Jacobi-Bellman equation and provide the fixed-final-time optimal solution. Performance of the new synthesis technique is demonstrated through different examples including 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

Among the multitude of researches in the literature that use neural networks (NN) for control of dynamical systems, one can cite [1]-[6]. A few amongst them develop neural network based optimal control based on an approximate dynamic programming (ADP) formulation [4], [7]-[17]. Two classes of ADP based solutions, called the Heuristic Dynamic Programming (HDP) and Dual Heuristic Programming (DHP) have emerged in the literature [4]. 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 [7]. The convergence proof of DHP for linear systems is presented in [8] and that of HDP for general case is presented in [9]. While [7]-[16] deal with discrete-time systems, some

researchers have recently focused on continuous time problems, [18]-[20].

Mechanism for ADP learning is usually provided through a dual network architecture called the Adaptive Critics (AC) [7], [11]. 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 [9], [10]. 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.[11]-[13]. The Single Network Adaptive Critic (SNAC) architecture developed in [14] 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 resulting simplicity makes it attractive for online implementation requiring less computational resources and storage memory. Similarly, the J-SNAC eliminates the need for the action network in an HDP scheme [15]. Note that these developments in the neural network literature have mainly addressed only the *infinite horizon* or regulator type problems.

Finite-horizon optimal control is relatively more difficult due to the time varying Hamilton-Jacobi-Bellman (HJB) equation resulting in a *time-to-go dependent* optimal cost function and costates. If one were to use a shooting method, a two-point boundary value problem (TPBVP) needs to be solved for each set of initial condition for a given final time and it will provide only an open loop solution. The authors of [21] developed a method which gives closed form solution to the problem but only for some pre-specified initial condition and time-to-go. Ref. [22] develops a dynamics optimization scheme which gives an open-loop solution, then, optimal tracking is used for rejecting the online perturbation and deviations from the optimal trajectory.

Using NN for solving finite-horizon optimal control problem is considered in [16], [23]-[28]. Authors of [16] used the AC's dual network scheme with time-dependent weights for solving the problem. Continuous-time problems are considered in [23] and [24] where the time-dependent weights are calculated through a backward integration. The finite-horizon problem with *unspecified terminal time* and a fixed terminal state is considered in [25]-[28]. In these researches the problem is called finite-horizon because the states are required to be brought to the origin using a *finite* number of steps, but, the

number of steps is not fixed which differentiates these works from the *fixed-final-time* problem investigated in this study.

In this paper, a single neural network based solution with a single set of weights, called Finite-horizon Single Network Adaptive Critics (Finite-SNAC), is developed which embeds solutions to the discrete-time HJB equation. Consequently, the offline trained network can be used to generate *online feedback control*. Furthermore, a major advantage of the proposed technique 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 [30], is used in this study.

Comparing the developed controller in this paper with the available controllers in the literature, the closest one is [16]. The difference between Finite-SNAC and the controller developed in [16] is using only one network and only one set of weights for the purpose. Despite [23] and [24] the Finite-SNAC solves discrete-time problems and uses ADP to do so. Finally, [25]-[28] solves unspecified terminal time problems while Finite-SNAC solves the problems with given and fixed final time.

Specifically, in this paper an ADP based controller for control-constrained finite-horizon optimal control of 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 the finite-horizon case is presented first. Then, it is shown that DHP has the same convergence result as the HDP, 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. The first example with a linear system allows easy comparison of the Finite-SNAC with known exact optimal results. The second example is a discrete-time nonlinear problem, and as the third example a more complex nonlinear spacecraft application, that is a fixed final time attitude maneuver, is carried out to show the applicability of Finite-SNAC to difficult engineering applications.

Rest of the paper is organized as follows: the Finite-SNAC is developed in Section II. Relevant convergence theorems are presented in Section III. Numerical results and analysis are presented in Section IV. Conclusions are given in Section V.

II. DEVELOPMENT OF FINITE-SNAC

A single neural network, called Finite-SNAC, is developed in this study which maps the relation between costates vector, used in the optimal control, and the states vector along with the time-to-go. The mapping is described in a functional form as

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

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. The dimension of the state space is given by n. 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 the time-to-go $(N-k)$.

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

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

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

Denoting the control vector by $u_k \in \mathbb{R}^l$, where l is the number of controls, the nonlinear control-affine system is assumed to be of the form

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

where $f: \mathbb{R}^n \to \mathbb{R}^n$ and $g: \mathbb{R}^n \to \mathbb{R}^{n \times l}$ are the system dynamics. A non-quadratic cost function J is used to incorporate the control constraints [30]. It is given by

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

where $G: \mathbb{R}^l \to \mathbb{R}$ is defined as

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

and $\rho^{-1}(.)$ denotes the inverse of function $\rho: \mathbb{R}^l \to \mathbb{R}^l$ which is a bounded continuous

one-to-one real-analytic integrable saturating function which passes through the origin, for example, a hyperbolic tangent function. Note that $G(.)$ is a non-negative scalar and $Q_f \in \mathbb{R}^{n\times n}, Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{l\times l}$ the penalizing matrices for the final states, states, and control vectors, respectively. Matrices Q_f and Q should be positive semi-definite or positive definite while R has to be a positive definite matrix.

The optimal cost-to-go at current state and time, denoted by $J^*(x_k, k)$, is given by solving the discrete-time HJB equation [29]

$$J^*(x_k, k) = \min_{u_k}\left(\frac{1}{2}(x_k^T Q x_k + G(u_k)) + J(x_{k+1}, k+1)\right),\ 0 \le k \le N-1. \quad (6)$$

The optimal control, u_k^*, is obtained from

$$u_k^* = \operatorname{argmin}_{u_k}\left(\frac{1}{2}(x_k^T Q x_k + G(u_k)) + J(x_{k+1}, k+1)\right), 0 \le k \le N-1. \quad (7)$$

Define $\lambda_k \equiv \frac{\partial J_k}{\partial x_k}$ as a column vector to get

$$u_k = -\rho(R^{-1} g(x_k)^T \lambda_{k+1}),\ 0 \le k \le N-1. \quad (8)$$

Replacing u_k in (6) by u_k^*, the HJB equation reads

$$J^*(x_k, k) = \frac{1}{2}(x_k^T Q x_k + G(u_k^*)) + J^*(x_{k+1}, k+1), 0 \le k \le N-1. \quad (9)$$

Taking the derivative of both sides of (9) with respect to x_k leads to the costate propagation equation which the network training targets, denoted by λ^t, are based on:

$$\lambda_{k+1}^t = Q x_{k+1} + \left(\frac{\partial(f(x_{k+1}) + g(x_{k+1})u_{k+1})}{\partial x_{k+1}}\right)^T \lambda_{k+2}, 0 \le k < N-1. \quad (10)$$

Note that

$$J^*(x_N, N) = \frac{1}{2} x_N^T Q_f x_N, \quad (11)$$

Hence,

$$\lambda_N^t = Q_f x_N. \quad (12)$$

In the training process, λ_{k+2} on the right hand side of (10) is obtained by using the same neural network as $NN(x_{k+1}, N-(k+1), W)$.

The network training should be done in such a way that along with learning the

target given in (10) for every state x_k and time k, *the final condition (12) is also satisfied at every step.* In this study, this idea is incorporated by augmenting the training input-target pairs with the final stage costate. Define the following augmented parameters:

$$\bar{\lambda} \equiv [\lambda_{k+1} \;\; \lambda_N], \tag{13}$$

$$\bar{\phi} \equiv [\phi(x_k, N-k) \;\; \phi(x_{N-1}, 1)]. \tag{14}$$

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

$$\bar{\lambda} = W^T \bar{\phi}, \tag{15}$$

$$\bar{\lambda}^t \equiv [\lambda_{k+1}^t \;\; \lambda_N^t]. \tag{16}$$

The training error is defined as

$$e \equiv \bar{\lambda} - \bar{\lambda}^t = W^T \bar{\phi} - \bar{\lambda}^t. \tag{17}$$

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 (10) after propagating x_k to x_{k+1}. Then, to calculate λ_N^t through (12), another randomly selected state will be considered as x_{N-1} and propagated to x_N and fed to (12). Finally $\bar{\lambda}^t$ will be formed as in (16). This process is depicted graphically in Fig. 1. In this diagram, the left side follows (10) and the right side follows (12) for network target calculations.

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

$$\langle e, \bar{\phi} \rangle = 0_{n \times m}, \tag{18}$$

where $\langle X, Y \rangle = \int_\Omega XY^T dx$ is the defined inner product on the compact set Ω on $\mathbb{R}^n$ for the functions $X(x)$ and $Y(x)$, and $0_{n \times m}$ denotes an $n \times m$ null matrix. Denoting the ith row of matrices e and $\bar{\phi}$ by e_i and $\bar{\phi}_i$, respectively, (18) leads to following equations

$$\langle e_i, \bar{\phi} \rangle = 0_{1 \times m} \;\; \forall i, \; 1 \le i \le n, \tag{19}$$

$$\langle e_i, \bar{\phi}_j \rangle = 0 \;\; \forall i, j, \; 1 \le i \le n, \; 1 \le j \le m. \tag{20}$$

Substituting e from (17) into (18) results in

$$\langle e, \bar{\phi} \rangle = W^T \langle \bar{\phi}, \bar{\phi} \rangle - \langle \bar{\lambda}^t, \bar{\phi} \rangle = 0, \tag{21}$$

or

$$W = \langle \bar{\phi}, \bar{\phi} \rangle^{-1} \langle \bar{\phi}, \bar{\lambda}^t \rangle. \tag{22}$$

Eq. (22) is the desired weight update for the training process.

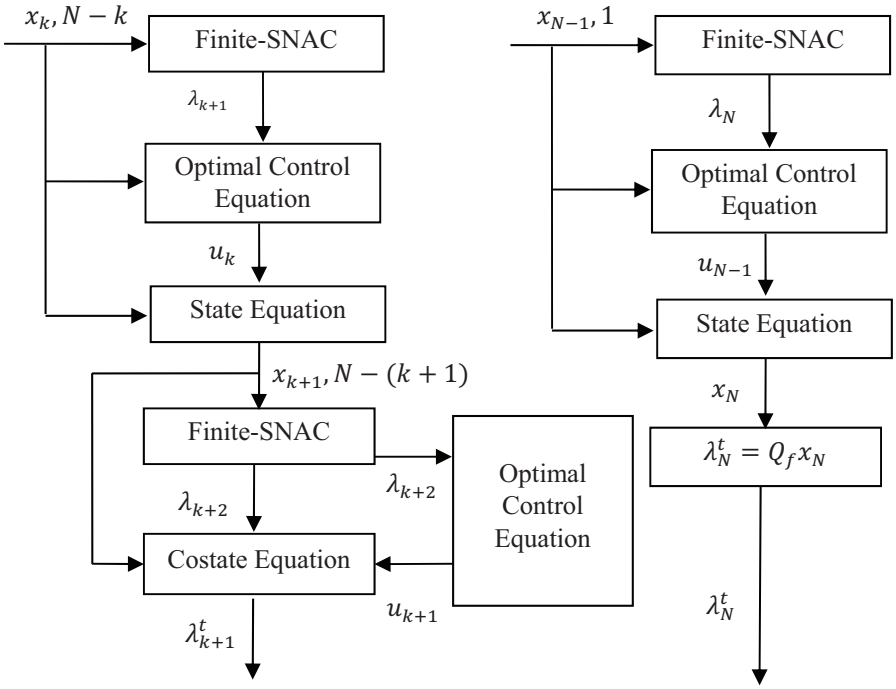

Fig. 1. Finite-SNAC training diagram

Finally, for use in discrete formulations, the integral used in the inner products in (22) is discretized by evaluating the inner products at p different points in a mesh covering the compact set Ω [23]. Denoting the distance between the mesh points by Δx, one has

$$\langle \bar{\phi}, \bar{\phi} \rangle = \lim_{\|\Delta x\| \to 0} \bar{\boldsymbol{\phi}} \bar{\boldsymbol{\phi}}^T \Delta x, \tag{23}$$

$$\langle \bar{\phi}, \bar{\lambda}^t \rangle = \lim_{\|\Delta x\| \to 0} \bar{\boldsymbol{\phi}} \bar{\boldsymbol{\lambda}}^{\boldsymbol{t}^T} \Delta x, \tag{24}$$

where

$$\bar{\boldsymbol{\phi}} = \left[\bar{\phi}(x_1) \;\; \bar{\phi}(x_2) \;\; \ldots \;\; \bar{\phi}(x_p) \right], \tag{25}$$

$$\bar{\boldsymbol{\lambda}}^{\boldsymbol{t}} = \left[\bar{\lambda}^t(x_1) \;\; \bar{\lambda}^t(x_2) \;\; \ldots \;\; \bar{\lambda}^t(x_p)\right], \tag{26}$$

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

Using (23) and (24), the weight update rule (22) is now simplified to the standard least square form as

$$W = (\bar{\boldsymbol{\phi}}\bar{\boldsymbol{\phi}}^T)^{-1}\bar{\boldsymbol{\phi}}\bar{\boldsymbol{\lambda}}^{\boldsymbol{t}^T}. \tag{27}$$

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 (27) looks like a one shot solution for the ideal NN weights, the training is an *iterative* process which consists of selecting different random states from the problem domain and times and updating the network weights by repeated use of (27). The reason for the iterative nature of the training process is the reinforcement learning basis of ADP. To understand it better, one should note that $\bar{\lambda}^t$ used in the weight update (27) 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 iteration of the weight update by $W^{(i)}$ results in the following iterative procedure as

$$W^{(i+1)} = (\bar{\boldsymbol{\phi}}\bar{\boldsymbol{\phi}}^T)^{-1}\bar{\boldsymbol{\phi}}\bar{\boldsymbol{\lambda}}^{\boldsymbol{t}}\left(W^{(i)}\right)^T. \tag{28}$$

The weight training is started with an initial weight $W^{(0)}$ and iterated 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.

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 using (8).

Remark 1: as seen in (7), the optimal control at time step k depends on the state at the next time step, i.e., u_k is implicitly a function of x_{k+1}, but it is explicitly a function of λ_{k+1} as can be seen from (8). Now, we seek to synthesize a feedback control u_k in terms of the current state x_k and therefore, we use SNAC to capture the mapping between x_k and λ_{k+1} through a reinforcement learning scheme.

Remark 2: The reinforcement learning given by (10) can be categorized as a value-iteration method of dynamic programming. In value-iteration scheme, one uses (6) for learning the cost-to-go [9], and the Finite-SNAC training uses the gradient of this equation with respect to the state vector, i.e., Eq. (10).

III. CONVERGENCE THEOREMS

Convergence theorems for the proposed optimal controller are composed of three parts: first, one needs to show that the reinforcement learning process, which the target calculation is based on, will result in the optimal target, then it needs to be shown that the weight update rule 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. It should be mentioned that the results in [10] provide the framework that we extend to the finite-horizon case and the DHP scheme in this paper.

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. The *iteration index is denoted 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_ks from zero to infinity where $0 \le k \le N$,

$$u_k^{i_k+1} = -\rho\left(R^{-1} g(x_k)^T \lambda_{k+1}^{i_{k+1}}\right), \ \ 0 \le k \le N-1, \tag{29}$$

$$\lambda_k^{i_k+1} = Q x_k + A\left(x_k, u_k^{i_k+1}\right)^T \lambda_{k+1}^{i_{k+1}}, \ \ 0 \le k \le N-1, \tag{30}$$

$$\lambda_N = Q_f x_N. \tag{31}$$

Note that $i_k$denotes the iteration number for stage k, $0 \le k \le N$. It needs the stage as a subscript since different stages may take different iterations to converge. For example in (30) the $(i_k+1)^{\text{th}}$ version of λ_k is being calculated based on $(i_{k+1})^{\text{th}}$ version of λ_{k+1}. Eq. (31) is the final condition of the optimal control problem. Note that,

$$A\left(x_k, u_k^{i_k+1}\right) \equiv \frac{\partial\left(f(x_k)+g(x_k)u_k^{i_k+1}\right)}{\partial x_k} \tag{32}$$

$$\lambda_{k+1}^{i_{k+1}} \equiv \lambda^{i_{k+1}}(x_{k+1}) = \lambda^{i_{k+1}}\left(f(x_k) + g(x_k)u_k^{i_k+1}\right). \tag{33}$$

The problem is to prove that this iterative procedure results in the optimal value for the costate λ and control u. The convergence proof presented in this paper 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.

$$J^{i_k+1}(x_k, k) = \frac{1}{2}\left(x_k^T Q x_k + G\left(u_k^{i_k+1}\right)\right) + J^{i_{k+1}}(x_{k+1}, k+1), 0 \le k \le N-1, \quad (34)$$

$$u_k^{i_k+1} = \text{argmin}_u \left(J^{i_{k+1}}(x_k, k)\right) = -\rho\left(R^{-1} g(x_k)^T \frac{\partial J_{k+1}^{i_{k+1}}}{\partial x_{k+1}}\right), \ 0 \le k \le N-1. \quad (35)$$

For the finite horizon case, the final condition is given by

$$J(x_N, N) = \frac{1}{2} x_N^T Q_f x_N. \quad (36)$$

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

$$J_{k+1}^{i_{k+1}} \equiv J^{i_{k+1}}\left(f(x_k) + g(x_k) u_k^{i_k+1}, k+1\right). \quad (37)$$

Convergence of the above mentioned reinforcement learning schemes are given below and their proofs are presented in the Appendix.

Theorem 1: HDP convergence: The sequence of J^{i_k} iterated through (34) to (36), in case of $J^0(x_k, k) = 0$ converges to the fixed-final-time optimal solution.

Theorem 2: DHP convergence: Denoting the states and the control vector with s and v, respectively, consider sequences $\lambda_k^{i_k}$ and $v_k^{i_k}$ defined by equations (38) to (40), where $s_{k+1} = f(s_k) + g(s_k) v_k^{i_k+1}$. Note that i_k is the index of iteration of the respective parameter at time step k, where k is the time index, and $A(s_k, v_k) \equiv \frac{\partial s_{k+1}}{\partial s_k} = \frac{\partial\left(f(s_k) + g(s_k) v_k^{i_k+1}\right)}{\partial s_k}$.

$$v_k^{i_k+1} = -\rho\left(R^{-1} g(s_k)^T \lambda_{k+1}^{i_{k+1}}\right), \quad (38)$$

$$\lambda_k^{i_k+1} = Q s_k + A\left(s_k, v_k^{i_k+1}\right)^T \lambda_{k+1}^{i_{k+1}}, \quad (39)$$

$$\lambda_N = Q_f s_N. \quad (40)$$

If $\lambda_k^0 = 0 \ \forall k$, then the sequence will converge to the optimal solution for the given nonlinear affine system, i.e. $\lambda_k^{i_k} \to \lambda_k^*$ and $v_k^{i_k} \to v_k^*$ as $i_k \to \infty$ for different ks where λ_k^* and v_k^* denote the corresponding optimal parameters.

Remark 3: In the learning laws given by (29) to (31) and also (34) to (36) it should be noted that the time index k is upper bounded by N, while i_k is the iteration index for the parameters at time step k.

B. Convergence of the Error of the Training Law and the 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 below theorems are given in the Appendix, as well.

Theorem 3: Training error convergence**:** The weight update (18) will force the error (17) to converge to zero as the number of neurons of the neural networks, m, tends to infinity.

Theorem 4: Neural network weight convergence: Assuming an ideal set of weights, denoted by W^*, where

$$\bar{\lambda}^t = \sum_{i=1}^{\infty} W_i^{*T} \bar{\phi}_i. \tag{41}$$

Then, using the weight update (18), one has $(W - W_{trunc}^*) \to 0$ where W_{trunc}^* is the truncated first m row of the ideal weight W^*.

IV. NUMERICAL ANALYSIS

In order to assess the performance of the proposed technique, three examples have been chosen: first is a linear system to motivate the developed technique, the second is a discrete-time nonlinear system, and the third is a complex nonlinear system to show its broad applicability.

A. Example 1

A first order linear system with a quadratic cost is selected for which the exact discrete-time optimal finite-horizon solution is known as given below:

$$x_{k+1} = x_k + u_k \,, \quad k = 0,1, \dots, N-1, \tag{42}$$

$$J = \frac{1}{2} x_N^2 + \sum_{k=0}^{N-1} \frac{1}{2} u_k^2. \tag{43}$$

Note that, the convergence results given in this paper are for a general form of penalizing

the control in the cost function, hence, the results hold for the quadratic cost function as well.

By solving discrete-time algebraic Riccati equation for the unknown P_k, where $\lambda_k = P_k x_k$, it is observed that the finite-horizon optimal solution for this system is given by

$$P_k = \frac{1}{N-k+1}, \tag{44}$$

hence,

$$u_k = -(R + B^T P_k B)^{-1} B^T P_k A x_k = -\frac{1}{N-k+2} x_k, \tag{45}$$

or equivalently

$$\lambda_{k+1} = \frac{1}{N-k+2} x_k. \tag{46}$$

For the comparison purpose, Finite-SNAC is used for solving this problem. Two separate set of basis functions are selected and the network is trained for both of them, separately. The first set of basis functions is selected as

$$\phi(x, \tau) = [x,\ \tau,\ x\tau,\ x/(\tau + 2)\ ]^T. \tag{47}$$

which contains the term $x_1/(\tau + 2)$ that was observed in Eq. (46) to be needed for the optimal costate calculation, note that τ denotes the time-to-go, i.e., $N - k$. Using this basis function set, the training was performed for 30 iterations where at each iteration 10 random states were selected from the interval $[-1\ \ 1]$ for the least squares operation given in (27). The resulting weight history versus the training iteration is given in Fig. 2. As seen in this figure, the weights have converged which is equivalent of convergence of the costates. The resulting weights are

$$W = [0.000,\ 0.000,\ 0.000,\ 1.000]^T, \tag{48}$$

which leads to the Finite-SNAC of the form

$$\lambda_{k+1} = \frac{1.000}{N-k+2} x_k, \tag{49}$$

and it is identical to the exact solution given by (46).

In many cases the required basis functions are not known and the control designer

needs to try different basis functions to end up with the best result. The second set of basis functions are selected in such a manner, i.e., not including $x/(\tau+2)$, as given below

$$\phi(x,\tau) = [x,\ \tau,\ x\tau,\ x\tau^2,\ x\tau^3,\ x\,exp(-\tau)\,]^T, \tag{50}$$

and the resulting weight evolution history is given in Fig. 3 which shows its convergence. Moreover, Fig. 4 depicts the network output versus different initial states at different training iterations, along with the optimal costate versus the state. As can be seen, the costate generated through the network is converging to the optimal costate as the number of iterations increases. Simulating the trained network using initial condition of $x = 1$ for $N = 10$, the cost-to-go difference between the exact optimal solution (46) and the trained network using the second set of basis functions turned out to be less than 0.001%. This comparison shows that the Finite-SNAC controller has been able to generate optimal control even if the basis functions do not include the desired functions given by the analytical optimal solution.

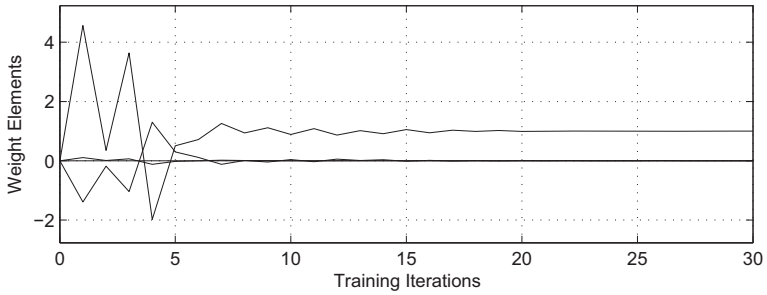

Fig. 2. History of weights for Example 1 with the first set of the basis functions.

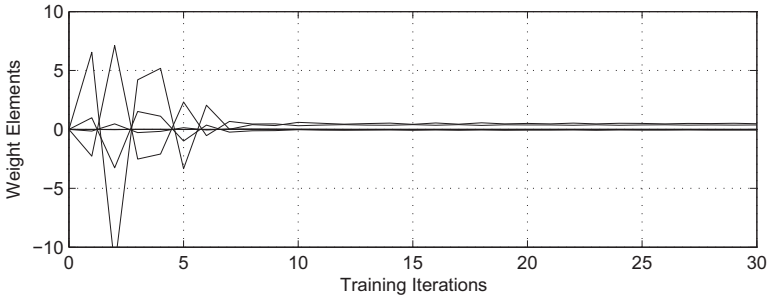

Fig. 3. History of weights for Example 1 with the second set of the basis functions.

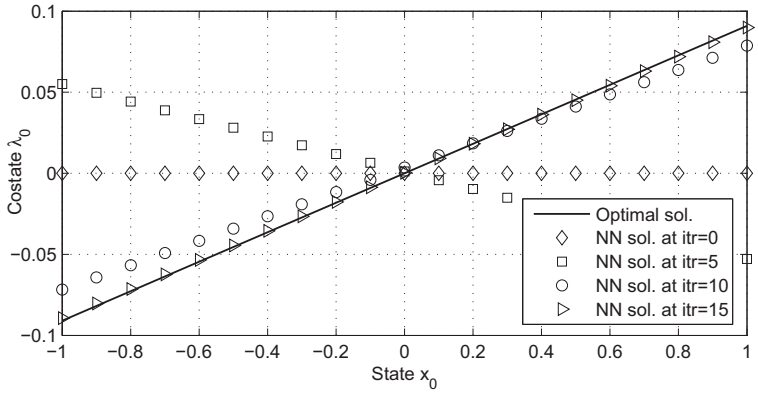

Fig. 4. Costate network output convergence with iterations.

B. Example 2

As the second simulation, the discrete-time nonlinear system simulated in [25] is adapted here.

$$x_{k+1} = f(x_k) + gu_k \tag{51}$$

where $x_k = \left[x_{1_k},\ x_{2_k}\right]^T,\ u_k = \left[u_{1_k},\ u_{2_k}\right]^T$, and

$$f(x_k) = \begin{bmatrix} 0.2x_{1_k}\exp(x_{2_k}^2) \\ 0.3x_{2_k}^3 \end{bmatrix},\quad g = \begin{bmatrix} -0.2 & 0 \\ 0 & -0.2 \end{bmatrix}. \tag{52}$$

Selected cost function for this system is

$$J = \frac{1}{2}x_N^T Q_f x_N + \sum_{k=0}^{N-1}\frac{1}{2}(x_k^T Q x_k + u_k^T R u_k), \tag{53}$$

where

$$Q = R = diag(10,\ 10),\ Q_f = diag(700,\ 700). \tag{54}$$

Denoting the inputs of the network by $x_i,\ i = 1,2,$ and τ, where τ is the normalized time-to-go (through dividing the time-to-go by the total number of time steps), the basis functions are selected as

$$\phi(x_1, x_2, \tau) = \\ [x_1, x_2, \tau, x_1x_2, x_1\tau, x_2\tau, x_1^2, x_2^2, x_1^3, x_2^3, x_1x_2^2, x_1^2x_2, x_1\,exp(-\tau)\,, x_2\,exp(-\tau)\,]^T. \tag{55}$$

The contributions of different network inputs in the basis functions are selected through some trial and error such that the network error is small. This selection leads to 14 basis functions.

Selecting $N = 4$, the network weights are initialized to zero and the training is performed for 10 iterations, where at each iteration, 200 states are randomly selected in the domain of interest of $x_i \in [-1, 1]$, $i = 1$ and 2. The convergence of the weights can be seen through Fig. 5 which shows the weights' evolution versus the training iterations.

The trained network is then used for online control starting with an initial condition of $x(t_0) = [1, -1]^T$. The results are shown in Fig. 6 using black plots. In order to evaluate the optimality of the results, the open-loop optimal solution to the problem is calculated using gradient based numerical methods and the results are shown in Fig. 6 using red plots. As seen in this figure, the Finite-SNAC result is very close to the optimal solution, while, Finite-SNAC is a closed-form solution and once trained offline, can be used for different initial conditions (as long as the state trajectory falls in the domain for which the network is trained) and different time-to-gos (as long as the new time-to-go is less than the one for which the Finite-SNAC is trained). To show this capability, the same trained network is used for controlling the same initial condition, but with shorter horizon of $N = 2$. The optimal numerical solution is also re-calculated for the new time-to-go and the results are depicted in Fig. 7, where the black plots denote the Finite-SNAC results and the red plots denote the open-loop numerical results. As seen, the Finite-SNAC is pretty accurate in generating the optimal control for the shorter horizon as well.

Finally, the same trained network is used for controlling another set of initial conditions, namely $x_0 = [1, 1]^T$. Note that each time that the initial condition changes or the horizon changes the numerical solution obtained using gradient based methods loses its validity and a new solution needs to be obtained. The Finite-SNAC results along with the new numerical optimal solution are depicted in Fig. 8. It can be seen that the Finite-SNAC's result is very close for the new initial condition too. These simulations show that the Finite-SNAC has a great potential for calculating the finite-horizon optimal solution in close-form and in real-time for different initial conditions and horizons.

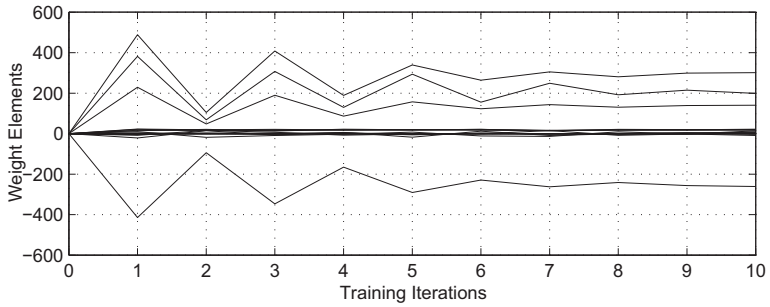

Fig. 5. Weight history versus training iteration for Example 2.

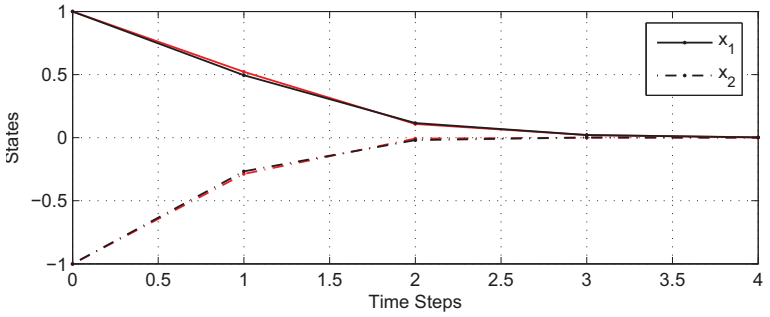

Fig. 6. State trajectories of Example 2 for initial condition of $[1, -1]^T$ and horizon of 4 time steps. Black plots denote the Finite-SNAC results and red plots denote the optimal open loop results.

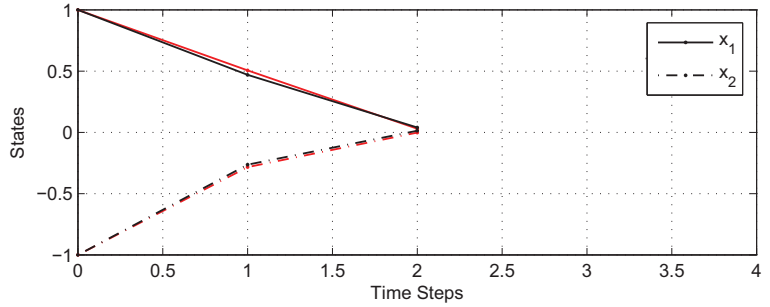

Fig. 7. State trajectories of Example 2 for initial condition of $[1, -1]^T$ and horizon of 2 time steps. Black plots denote the Finite-SNAC results and red plots denote the optimal open loop results.

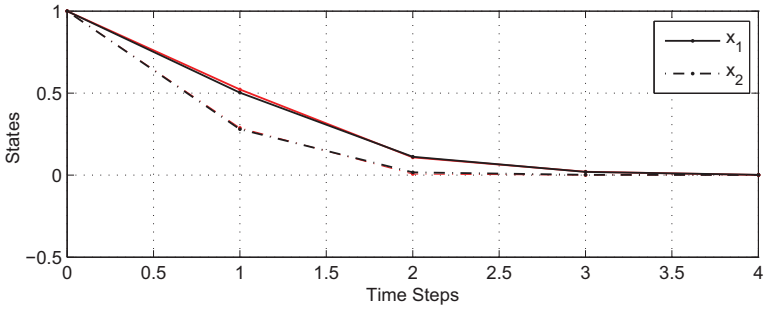

Fig. 8. State trajectories of Example 2 for initial condition of $[1,1]^T$ and horizon of 4 time steps. Black plots denote the Finite-SNAC results and red plots denote the optimal open loop results.

C. Example 3

As a real world application of the developed method, the problem of nonlinear satellite attitude control has been selected. Satellite dynamics are represented as [32]

$$\frac{d\omega}{dt} = I^{-1}(N_{net} - \omega \times I\omega), \tag{56}$$

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 $\times$ denotes the cross product of two vectors. The total torque, N_{net}, is composed of the control, N_{ctrl}, and the disturbance torques, N_{dist}.

$$N_{net} = N_{ctrl} + N_{dist} \tag{57}$$

The control torque is the torque created by the satellite actuators. Since control torques are bounded in practice, this problem is 'control-constrained'.

Following [33] and its order of transformation, the kinematic equation of the satellite is

$$\frac{d}{dt}\begin{bmatrix}\varphi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}1 & \sin(\varphi)\tan(\theta) & \cos(\varphi)\tan(\theta)\\ 0 & \cos(\varphi) & -\sin(\varphi)\\ 0 & \sin(\varphi)/\cos(\theta) & \cos(\varphi)/\cos(\theta)\end{bmatrix}\begin{bmatrix}\omega_X\\ \omega_Y\\ \omega_Z\end{bmatrix}, \tag{58}$$

where $\varphi, \theta,$ 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. Subscripts $X, Y,$

and Z denote the corresponding elements of the angular velocity vector ω.

The three Euler angles and the three elements of the angular velocity form the elements of the state space for the satellite attitude control problem and form the following state equation as

$$\dot{x} = f(x) + g(x)u, \tag{59}$$

where

$$f(x) \equiv \begin{bmatrix} M_{3\times1} \\ I^{-1}(N_{dist} - \omega \times I\omega) \end{bmatrix}, \tag{60}$$

$$g \equiv \begin{bmatrix} 0_{3\times3} \\ I^{-1} \end{bmatrix}, \tag{61}$$

$$x = [\varphi \quad \theta \quad \psi \quad \omega_X \quad \omega_Y \quad \omega_Z]^T, \tag{62}$$

$$u = [N_{ctrl_X} \quad N_{ctrl_Y} \quad N_{ctrl_Z}]^T, \tag{63}$$

and $M_{3\times1}$ denotes the right hand side of equation (58). The three-by-three null matrix is denoted by $0_{3\times3}$.

The moment of inertia matrix of the satellite is chosen as

$$I = \begin{bmatrix} 100 & 2 & .5 \\ 2 & 100 & 1 \\ .5 & 1 & 110 \end{bmatrix} kg.m^2. \tag{64}$$

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

The initial states are 60, -20, and -70 deg. for the Euler angles φ, θ, and ψ, respectively, and -0.001, 0.001, and -0.001 rad/s. for the angular rates around X, Y, and Z axes, respectively. The goal of the controller is to perform an attitude maneuver to bring the states to zero, in a fixed final time of 800 sec by minimizing the cost function in (4). The sampling time of 0.25 sec. is selected for discretizing the state equation using the Euler method and the 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

$$Q = diag(0.25\ \ 0.25\ \ 0.25\ \ 25\ \ 25\ \ 25), \tag{65}$$

$$Q_f = diag(10^3 \;\; 10^3 \;\; 10^3 \;\; 10^5 \;\; 10^5 \;\; 10^5), \tag{66}$$

$$R = 10^3 diag(25 \;\; 25 \;\; 25). \tag{67}$$

Note that the last three diagonal elements of matrix Q and Q_f correspond to the angular rates with a 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 go to close to 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 6 along with $x_i x_j$, $x_i^2\tau$, $x_i\tau^2$, $x_i\tau^3$, τ, τ^2, τ^3, and $x_i e^{-\tau}$ for $i, j = 1$ to 6 $\; i \neq j$, resulting in 60 neurons, where, x_i is the ith state element, i =1 to 6, and τ is the time-to-go normalized through dividing it by the total number of time steps. For the training process, *in each iteration*, 100 sets of initial states among a previously selected interval of states are randomly selected to form a mesh and the weight update rule (27) is used for training the neural network. The training is performed for 600 iterations, until the weights converge. Note that in this example, the state vector's dimension is large, i.e., it has 6 elements and only 100 random state is selected at each iteration for the training, but, as long as these 100 states are *randomly* selected *at each iteration*, where the number of iterations is 600, the network is provided with the chance of experiencing different combinations of the states (namely 600×100 different state vectors,) to learn the whole domain of interest.

Initially, the simulation is performed with no disturbance torque, as the developed theory of Finite-SNAC is based on, and the results are shown in Fig. 9 and Fig. 10. The Euler angles as seen in Fig. 9 have nicely converged close to zero in the given final time of 800 sec. Fig. 10 shows the applied control history, and as expected, it has not violated the control bounds.

As mentioned earlier in Example 2, the converged Finite-SNAC will give optimal solution for *any* initial condition set within the domain on which the training is performed and *any* final time less than or equal to the final time chosen for training. To

depict this capability, the new set of initial conditions of 30, -80 and 90 degrees for φ, θ, and ψ, respectively, is selected and the same trained network is used for performing this maneuver. The results are shown in Fig. 11 and Fig. 12, and show that the controller brings the states close to zero and does not violate the constraint on its limits.

To further demonstrate the versatility of the proposed controller, using the same trained network, the first attitude maneuver is performed with a *shorter* time-to-go, i.e. 500 sec. and the results are superimposed with previous results and shown in Fig. 13 and Fig. 14 in red. As can be seen, the controller has applied a different control sequence on the satellite to accomplish the final goal. The control torques are higher at first (as compared to the 800-second results), in order to accomplish the same mission in a shorter final time of 500 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 to the trained horizon by virtue of Bellman's principle of optimality [29].

Finally, in order to analyze the effect of external disturbances on the controller, the gravity gradient disturbance is modeled [32] and applied on the satellite and the results are shown using the red plots in Fig. 15 and Fig. 16, super imposed with the results of the (disturbance-less) first attitude control simulation. Note that even-though this method is not developed to measure and cancel the effect of the disturbance, the optimal control formulation and the generalization property of the neural networks are robust enough to be able to get an acceptable trajectory even in the presence of some unknown disturbances. This can be confirmed not only by looking at the Euler angles trajectory and their final values, but also by analyzing the applied disturbance torque in Fig. 17. Note that some big disturbance torques, as big as the actuators' saturation limit, are applied on the X and Y axes of the satellite. Comparing the applied control in the presence of the disturbance, with the case of no disturbance, in Fig. 16, shows that the controller has quickly switched the control toques on those two axes in order to compensate the effect of the excessive applied torque and accomplish the maneuver in the same fixed horizon.

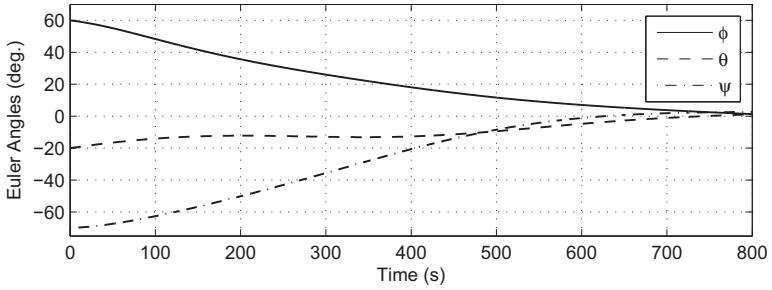

Fig. 9. Euler angles trajectories for the final time of 800 seconds.

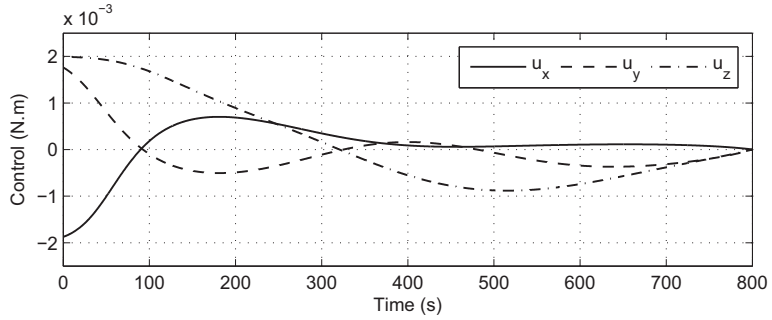

Fig. 10. Control histories for the final time of 800 seconds.

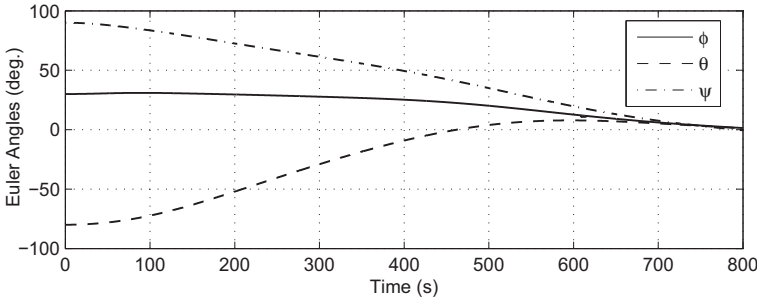

Fig. 11. Euler angles trajectories for a new set of initial conditions.

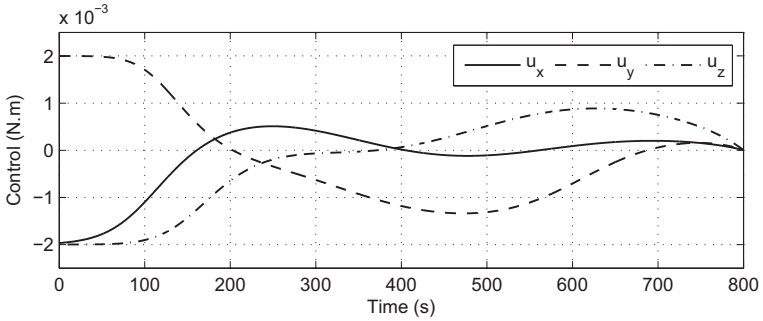

Fig. 12. Control histories for a new set of initial conditions.

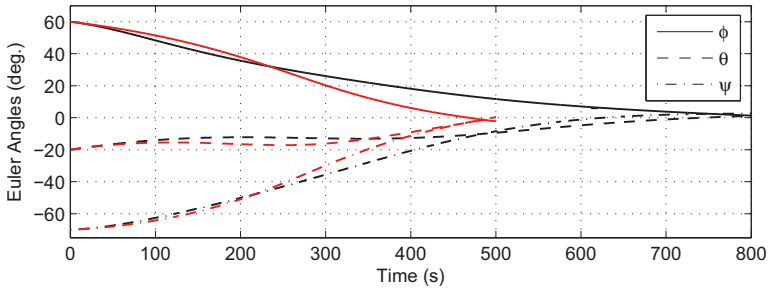

Fig. 13. Euler angles trajectories for different final times. Red and black plots denote the final times of 500 and 800 sec., respectively.

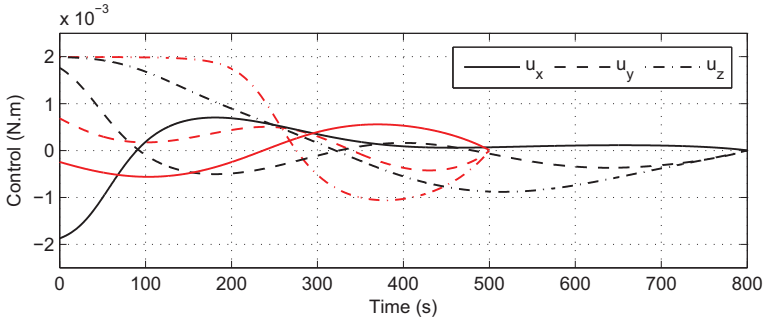

Fig. 14. Control histories for different final times. Red and black plots denote the final times of 500 and 800 sec., respectively.

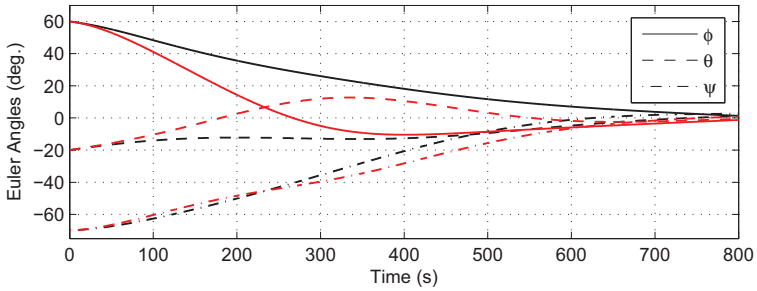

Fig. 15. Euler angles trajectories. Red and black plots denote the results with and without the presence of the disturbance, respectively.

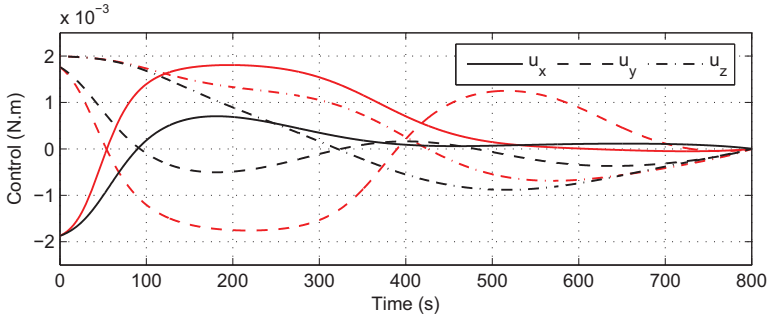

Fig. 16. Control histories. Red and black plots denote the results with and without the presence of the disturbance, respectively.

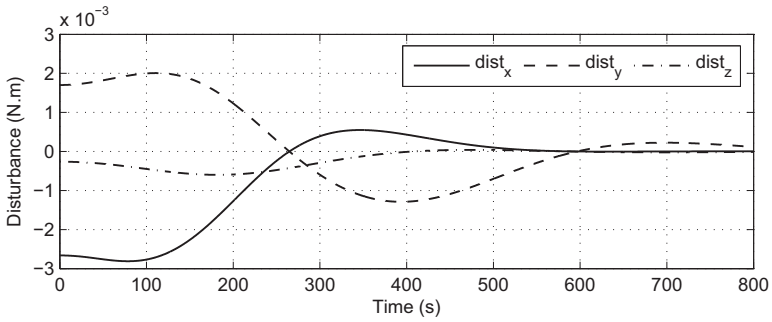

Fig. 17. Applied gravity gradient disturbance.

V. CONCLUSIONS

A finite-horizon optimal neurocontroller, that embeds 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 simulations for the linear example, the nonlinear example and for the nonlinear satellite control problem indicate that the developed method is very versatile and has a good potential for use in solving the optimal closed loop control of control-affine nonlinear systems.

ACKNOWLEDGEMENT

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

APPENDIX

The proofs of the theorems given in the paper are presented here.

A. Convergence of the Algorithm to the Optimal Solution: Proofs

In [9] 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 [9] with some modifications to handle the time dependency of the optimal cost function.

Lemma 1 [9]: Using any arbitrary control sequence of μ_k, and Λ^{i_k} defined as

$$\Lambda^{i_k+1}(x_k,k)=\frac{1}{2}\left(x_k^TQx_k+G(\mu_k)\right)+\Lambda^{i_{k+1}}(f(x_k)+g(x_k)\mu_k,k+1), \tag{68}$$

if $\Lambda^0(x_k,k)=J^0(x_k,k)=0$ then $\Lambda^{i_k}(x_k,k)\geq J^{i_k}(x_k,k)\ \ \forall i$ where $J^{i_k}(x_k,k)$ is iterated through (34) and (35).

Proof: The proof given in [9] is applicable here also.■

Lemma 2: The parameter $J^{i_k}(x_k,k)$ resulting from (34) and (35), is upper bounded by an existing bound $Y(x_k,k)$.

Proof: As explained in Subsection III.A, the learning rule (34) to (36) is implemented by randomly selecting a time index k within the bounds, at each iteration i_k, and updating $J^{i_k}(x_k,k)$ to $J^{i_k+1}(x_k,k)$. Hence, as long as the k selection is random, some ks might be selected more than the others, e.g., for some k the respective J might be iterated several times while for some other k, the respective J may not have yet been iterated and hence it is still zero because of being initialized at zero. However, note that even at consecutive iteration of the same time index k, parameters J_k and u_k will be updated, because the right hand side of Eq. (35) depends on x_{k+1} which itself is a function of u_k, hence, even if J_{k+1} is iterated only for i_{k+1} times and kept constant afterward, by each iteration of J_k using the same J_{k+1}, one ends up with new J_k and u_k until they converge. Let η_k be an arbitrary control and let $Z^0(x_k,k)=J^0(x_k,k)=0$, where Z^{i_k} is updated through random selection of time step k at each iteration, $0\leq k\leq N-1$, and using

$$Z^{i_k+1}(x_k,k)=\frac{1}{2}\left(x_k^TQx_k+G(\eta_k)\right)+Z^{i_{k+1}}(x_{k+1},k+1) \tag{69}$$

$$Z(x_N,N)=\frac{1}{2}x_N^TQ_fx_N \tag{70}$$

$$x_{k+1}=f(x_k)+g(x_k)\eta_k. \tag{71}$$

Defining $Y(x_k, k)$ as

$$Y(x_k, k) = \frac{1}{2} x_N^T Q_f x_N + \sum_{n=0}^{N-k-1} \frac{1}{2} (x_{k+n}^T Q x_{k+n} + G(\eta_{k+n})), \tag{72}$$

and subtracting (72) from (69) results in

$$Z^{i_k+1}(x_k, k) - Y(x_k, k) = Z^{i_{k+1}}(x_{k+1}, k+1) - \left(\frac{1}{2} x_N^T Q_f x_N + \sum_{n=1}^{N-k-1} \frac{1}{2} (x_{k+n}^T Q x_{k+n} + G(\eta_{k+n}))\right), \tag{73}$$

which is the equivalent of

$$Z^{i_k+1}(x_k, k) - Y(x_k, k) = Z^{i_{k+1}}(x_{k+1}, k+1) - Y(x_{k+1}, k+1), \tag{74}$$

and hence

$$Z^{i_{k+1}}(x_{k+1}, k+1) - Y(x_{k+1}, k+1) = Z^{i_{k+2}}(x_{k+2}, k+2) - Y(x_{k+2}, k+2), \tag{75}$$

where i_{k+2} denotes the iteration index of $Z(x_{k+2}, k+2)$ which has been used for calculation of $Z^{i_{k+1}}(x_{k+1}, k+1)$. The propagation of term $Z^{i_k+1}(x_k, k) - Y(x_k, k)$ from k toward N can be performed step by step as seen in (74) and (75). In this propagation, one of these two cases happens: a) at some time step, e.g., $\hat{k}$, the iteration index of $Z(x_{\hat{k}+1}, \hat{k}+1)$ used in the right hand side for calculation of $Z(x_{\hat{k}}, \hat{k})$ reaches zero, i.e., in

$$Z^{i_{\hat{k}}}(x_{k_1}, k_1) - Y(x_{\hat{k}}, \hat{k}) = Z^{i_{\hat{k}+1}}(x_{\hat{k}+1}, \hat{k}+1) - Y(x_{\hat{k}+1}, \hat{k}+1) \tag{76}$$

one has $i_{\hat{k}+1} = 0$, hence $Z^{i_{\hat{k}+1}}(x_{\hat{k}+1}, \hat{k}+1) = 0$, and

$$Z^{i_{\hat{k}}}(x_{k_1}, k_1) - Y(x_{\hat{k}}, \hat{k}) = 0 - Y(x_{\hat{k}+1}, \hat{k}+1) < 0. \tag{77}$$

b) the previous case doesn't happen, i.e., the propagation toward N is done completely without ending up with $Z^{i_{\hat{k}+1}}(x_{\hat{k}+1}, \hat{k}+1) = 0$ for some $\hat{k}$ and one has

$$Z^{i_N}(x_N, N) - Y(x_N, N) = \frac{1}{2} x_N^T Q_f x_N - \frac{1}{2} x_N^T Q_f x_N = 0. \tag{78}$$

From (77) and (78) one has

$$Z^{i_k}(x_k, k) \leq Y(x_k, k) \ \ \forall k. \tag{79}$$

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

$$J^{i_k}(x_k, k) \leq Y(x_k, k), \tag{80}$$

which proves Lemma 2.■

Lemma 3 [9]: If the optimal control problem can be solved, then there exists a least upper bound $J^*(x_k,k)$, $J^*(x_k,k) \le Y(x_k,k)$, which satisfies HJB equation (9) and $0 \le J^{i_k}(x_k,k) \le J^*(x_k,k) \le Y(x_k,k)$ where $Y(x_k,k)$ is defined in Lemma 2.

Proof: The proof is given in [9].■

Lemma 4 [9]: The sequence of $J^{i_k}(x_k,k)$ defined by HDP for every time step k, in case of $J^0(x_k,k) = 0$, is non-decreasing as i_k grows from 0 to infinity.

Proof: The proof is similar to [9], with some required modifications. Assume μ_k, be an arbitrary control and let $\Lambda^0(x_k,k) = J^0(x_k,k) = 0$, where Λ^{i_k} is updated through

$$\Lambda^{i_k+1}(x_k,k) = \frac{1}{2}\left(x_k^T Q x_k + G(\mu_k)\right) + \Lambda^{i_{k+1}}(x_{k+1},k+1). \tag{81}$$

From Lemma 1, we have

$$J^{i_k}(x_k,k) \le \Lambda^{i_k}(x_k,k), \forall k \tag{82}$$

Selecting $\mu_k = u_k^{i_k+2}$, and using $(i_{k+1}-1)^{\text{th}}$ version of $\Lambda(x_{k+1},k+1)$ in calculation of $\Lambda^{i_k}(x_k,k)$ leads to

$$\Lambda^{i_k}(x_k,k) = \frac{1}{2}\left(x_k^T Q x_k + G\left(u_k^{i_k+1}\right)\right) + \Lambda^{i_{k+1}-1}(x_{k+1},k+1). \tag{83}$$

From $\Lambda^0(x_k,k) = 0$, one has

$$J^1(x_k,k) - \Lambda^0(x_k,k) = \frac{1}{2}\left(x_k^T Q x_k + G(u_k^1)\right) + J^{i_{k+1}}(x_k,k) \ge 0. \tag{84}$$

Hence,

$$J^1(x_k,k) \ge \Lambda^0(x_k,k), \forall k. \tag{85}$$

To use method of induction, assume

$$J^{i_k}(x_k,k) \ge \Lambda^{i_k-1}(x_k,k), \forall k, \tag{86}$$

and subtract (83) from (34) to get

$$J^{i_k+1}(x_k,k) - \Lambda^{i_k}(x_k,k) = J^{i_{k+1}}(x_k,k) - \Lambda^{i_{k+1}-1}(x_k,k), \tag{87}$$

using assumption (86), one has

$$J^{i_k+1}(x_k,k) - \Lambda^{i_k}(x_k,k) \ge 0, \forall k, \tag{88}$$

hence,

$$J^{i_k+1}(x_k,k) \geq \Lambda^{i_k}(x_k,k), \forall k, \tag{89}$$

and by induction (89) is proved. Finally, inequalities (82) and (89) prove the lemma.■

Proof of Theorem 1: Using the results of Lemma 4 and Lemma 2 one has

$$J^{i_k}(x_k,k) \to J^{\infty}(x_k,k) \text{ as } i_k \to \infty, \forall k \tag{90}$$

for some $J^{\infty}(x_k,k)$, where

$$J^{\infty}(x_k,k) \equiv \frac{1}{2}x_N^T Q_f x_N + \sum_{n=0}^{N-k-1}\left(\lim_{i_{k+n}\to\infty}\frac{1}{2}\left(x_{k+n}^T Q x_{k+n} + G\left(u_{k+n}^{i_{k+n}}\right)\right)\right). \tag{91}$$

From Lemma 3

$$J^{\infty}(x_k,k) \leq J^*(x_k,k). \tag{92}$$

Since $J^{\infty}(x_k,k)$ satisfies the HJB equation and the finite-horizon final condition one has

$$J^{\infty}(x_k,k) = J^*(x_k,k) \tag{93}$$

which completes the proof.■

Proof of Theorem 2: This theorem is the DHP version of Theorem 1 and the result of Theorem 1 will be used for proving convergence of this algorithm. The idea is to use the method of induction to show that the evolution of the sequence in this algorithm is identical to that of the HDP one. The systems considered are the same and assume the same initial conditions. The states vector and control are denoted with another set of letters to provide the ability to compare them along the iterations in both of the algorithms.

From $J^0(x_k,k) = 0\ \forall k$ and $\lambda_k^0 = 0\ \ \forall k$ it follows that

$$\lambda_k^0 = \frac{\partial J^0(x_k,k)}{\partial x_k}\ \forall k. \tag{94}$$

One iteration of the HDP algorithm results in

$$u_k^0 = -\rho\left(R^{-1}g(x_k)^T\frac{\partial J^0(x_{k+1},k+1)}{\partial x_{k+1}}\right) \tag{95}$$

$$J^1(x_k,k) = \frac{1}{2}(x_k^T\,Q x_k + G(u_k^0)) + J^0(x_{k+1},k+1) \tag{96}$$

and one iteration of the DHP algorithm results in

$$v_k^0 = -\rho(R^{-1}g(s_k)^T\lambda_{k+1}^0) \tag{97}$$

$$\lambda_k^1 = Qs_k + A(s_k, v_k^0)^T\lambda_{k+1}^0. \tag{98}$$

If $x_k = s_k$, from (94), (95) and (97) it follows that

$$u_k^0 = v_k^0. \tag{99}$$

The derivative of (96) with respect to x_k is given by

$$\frac{\partial J^1(x_k,k)}{\partial x_k} = Qx_k + A(x_k, u_k^0)^T \frac{\partial J^0(x_{k+1},k+1)}{\partial x_{k+1}}. \tag{100}$$

Considering (94) and (99), and by comparing (100) with (98), it can be seen that

$$\lambda_k^1 = \frac{\partial J^1(x_k,k)}{\partial x_k}. \tag{101}$$

Now assume

$$x_k = s_k, \tag{102}$$

$$u_k^{i_k} = v_k^{i_k}, \tag{103}$$

$$\lambda_k^{i_k} = \frac{\partial J^{i_k}(x_k,k)}{\partial x_k}, \forall k, \tag{104}$$

and perform the i_kth iteration with both of the algorithms:

$$u_k^{i_k+1} = -\rho\left(R^{-1}g(x_k)^T \frac{\partial J^{i_{k+1}}(x_{k+1},k+1)}{\partial x_{k+1}}\right), \tag{105}$$

$$J^{i_k+1}(x_k,k) = \frac{1}{2}\left(x_k{}^T Q x_k + G\left(u_k^{i_k+1}\right)\right) + J^{i_{k+1}}(x_{k+1}, k+1), \tag{106}$$

$$v_k^{i_k+1} = -\rho\left(R^{-1}g(s_k)^T\lambda_{k+1}^{i_{k+1}}\right), \tag{107}$$

$$\lambda_k^{i_k+1} = Qs_k + A\left(s_k, v_k^{i_k+1}\right)^T\lambda_{k+1}^{i_{k+1}}. \tag{108}$$

The derivative of (106) with respect to x_k is

$$\frac{\partial J^{i_k+1}(x_k,k)}{\partial x_k} = Qx_k + A\left(x_k, u_k^{i_k+1}\right)^T \frac{\partial J^{i_{k+1}}(x_{k+1},k+1)}{\partial x_{k+1}}. \tag{109}$$

Again by comparing (109) with (108) and considering (102), (103) and (104), it can be

shown that

$$\lambda_k^{i_k+1} = \frac{\partial J^{i_k+1}(x_k,k)}{\partial x_k} \tag{110}$$

Hence, by iteratively using equations (34) to (36) for calculating $u_k^{i_k}$ and J^{i_k} and equations (38) to (40) for calculating $v_k^{i_k}$ and $\lambda_k^{i_k}$, it is proved that equations (103) and (104) are valid for all i_ks. Since $u_k^{i_k}$ and J^{i_k}, based on Theorem 1, converge to the optimal values as $i_k \to \infty$ for $0 \le k \le N-1$, $v_k^{i_k}$ and $\lambda_k^{i_k}$ will also converge to the optimal control and costates, and the proof is complete.■

B. Convergence of the Error of the Training Law and the Weights: Proofs

The proofs of Theorem 3 and 4 are inspired by [31], but, since the error equation and the dimension of the error are different compared to [31], the processes of the proofs are different and given below.

Proof of Theorem 3: Using Lemma 5.2.9 from [31], 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 $1 \le j \le m$ 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 $\psi(.)$ belongs to $span\{\bar{\phi}_j\}_1^\infty$ then

$$\psi = \sum_{j=1}^{\infty} \langle \psi, \bar{\phi}_j \rangle \bar{\phi}_j \tag{111}$$

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

$$\left\| \sum_{j=m+1}^{\infty} \langle \psi, \bar{\phi}_j \rangle \bar{\phi}_j \right\| < \epsilon \tag{112}$$

where $\|.\|$ denotes norm operation. From (18) one has

$$\langle e, \bar{\phi}_j \rangle = 0 \ \ \forall j, \ 1 < j < m \tag{113}$$

and from (17)

$$\langle e, \bar{\phi}_j \rangle = W^T \langle \bar{\phi}, \bar{\phi}_j \rangle - \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \tag{114}$$

which is equivalent to

$$\langle e, \bar{\phi}_j \rangle = \sum_{i=1}^{m} W_i^T \langle \bar{\phi}_i, \bar{\phi}_j \rangle - \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \tag{115}$$

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 $\{\bar{\phi}_j\}_1^\infty$

$$e = \sum_{j=1}^{\infty} \langle e, \bar{\phi}_j \rangle \bar{\phi}_j. \tag{116}$$

Inserting (115) into (116) results in

$$e = \sum_{j=1}^{\infty} \left(\sum_{i=1}^{m} W_i^T \langle \bar{\phi}_i, \bar{\phi}_j \rangle \bar{\phi}_j - \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \bar{\phi}_j \right). \tag{117}$$

But, from the weight update (113), the right hand side of (115) is also equal to zero. Applying this to (117) results in

$$e = \sum_{j=m+1}^{\infty} \left(\sum_{i=1}^{m} W_i^T \langle \bar{\phi}_i, \bar{\phi}_j \rangle \bar{\phi}_j - \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \bar{\phi}_j \right). \tag{118}$$

Due to the orthonormality of the basis functions, one has

$$\langle \bar{\phi}_i, \bar{\phi}_j \rangle = 0 \ \ \forall i \neq j. \tag{119}$$

Hence, (118) simplifies to

$$e = -\sum_{j=m+1}^{\infty} \langle \bar{\lambda}^t, \bar{\phi}_j \rangle \bar{\phi}_j. \tag{120}$$

Using (112) for $\psi = \bar{\lambda}^t$, as m increases, e decreases to zero.

$$\lim_{m \to \infty} \|e\| = 0 \tag{121}$$

This completes the proof.■

Proof of Theorem 4: The training error is defined as

$$e \equiv \bar{\lambda} - \bar{\lambda}^t. \tag{122}$$

Hence

$$e = \left(W^T - {W^*_{trunc}}^T \right) \bar{\phi} - \sum_{i=m+1}^{\infty} {W_i^*}^T \bar{\phi}_i. \tag{123}$$

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 (123) by $\bar{\phi}$ results in

$$\langle e, \bar{\phi} \rangle = \left(W^T - {W^*_{trunc}}^T \right) \langle \bar{\phi}, \bar{\phi} \rangle - \sum_{i=m+1}^{\infty} {W_i^*}^T \langle \bar{\phi}_i, \bar{\phi} \rangle. \tag{124}$$

The last term on the right hand side of the above equation vanishes due to the

orthonormality property of the basis functions. Considering $\langle \bar{\phi}, \bar{\phi} \rangle = I$, (123) simplifies to

$$\langle e, \bar{\phi} \rangle = W^T - {W^*_{trunc}}^T. \tag{125}$$

Examining (125) further, the weight update implies the left hand side to be zero, hence, using the weight update (18) one has $W \to W^*_{trunc}$. ■

REFERENCES

[1] P. J. Werbos, "Neural networks for control and system identification," in *Proc. IEEE Conf. Decision Control*, vol. 1, Tampa, FL, 1989, pp. 260-265.

[2] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4-27, Mar. 1990.

[3] P. J. Werbos, "Backpropagation through time: what it does and how to do it", *Proc. IEEE*, vol. 78, no. 10, pp. 1550-1560, Oct. 1990.

[4] P. J. Werbos, "Approximate dynamic programming for real time control and neural modeling", in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. New York: Van Nostrand Reinhold, 1992, pp. 493-525.

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

[6] F. L. Lewis, S. Jagannathan and A. Yesildire, *Neural network control of robot manipulators and nonlinear systems.* UK: Taylor & Francis,1999, pp. 173-411.

[7] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997-1007, Sep. 1997.

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

[9] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf , "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Trans. Syst. Man Cybern. B*, vol. 38, no. 4, pp. 943-949, Aug. 2008.

[10] T. Dierks and S. Jagannathan, "Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update," *IEEE Trans. Neural Netw. Learning Syst.,* vol. 23, no. 7, pp. 1118-1129, Jul. 2012.

[11] S. N. Balakrishnan and V. Biega, "Adaptive-critic-based neural networks for aircraft optimal control", *Journal of Guidance Control and Dynamics*, vol. 19, no. 4, pp. 893-898, Jul.-Aug. 1996.

[12] S. Ferrari, and R. F. Stengel, "Online adaptive critic flight control," *Journal of Guidance Control and Dynamics*, vol. 27, no. 5, pp. 777-786, Sep.-Oct. 2004.

[13] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13, no.3, pp. 764-773, May 2002.

[14] 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, no. 10, pp.1648–1660, Dec. 2006.

[15] J. Ding and S. N. Balakrishnan, "An online nonlinear optimal controller synthesis for aircraft with model uncertainties," in *Proc. AIAA Guidance, Navigation, and Control Conference*, 2010.

[16] D. Han and S. N. Balakrishnan, "State-constrained agile missile control with adaptive-critic-based neural networks," *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 4, pp. 481-489, Jul. 2002.

[17] M. Fairbank, E. Alonso, and D. Prokhorov, "Simple and fast calculation of the second-order gradients for globalized dual heuristic dynamic programming in neural networks," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 23, no. 10, pp. 1671-1676, Oct. 2012.

[18] D. Vrabie, O. Pastravanu, F. Lewis, and M. Abu-Khalaf, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, no. 2, pp. 477-484, Feb. 2009.

[19] K. Vamvoudakis and F. Lewis, "Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878-888, May 2010.

[20] T. Dierks and S. Jagannathan, "Optimal control of affine nonlinear continuous-time systems" in *Proc. American Control Conf.*,2010, pp. 1568-1573, 2010.

[21] T. Çimen and S. P. Banks, "Global optimal feedback control for general nonlinear systems with nonquadratic performance criteria," *Systems & Control Letters*, vol. 53, no. 5, pp. 327-346, Dec. 2004.

[22] V. Costanza and P. S. Rivadeneira, "Finite-horizon dynamic optimization of nonlinear systems in real time," *Automatica*, vol. 44, no. 9, pp. 2427-2434, Sep. 2008.

[23] 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. Neural Netw.*, vol. 18, no. 6, pp. 1725-1737, Nov. 2007.

[24] D. M. Adhyaru, I. N. Kar, and M. Gopal, "Fixed final time optimal control approach for bounded robust controller design using Hamilton-Jacobi-Bellman solution," *IET Control Theory Appl.*, vol. 3, no. 9, pp. 1183-1195, Sep. 2009.

[25] F.-Y. Wang, N. Jin, D. Liu, and Q. Wei, "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 24-36, Jan. 2011.

[26] Q. Wei and D. Liu, "An iterative ϵ-optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state," *Neural Networks*, vol. 32, pp. 236-244, Aug. 2012.

[27] D. Wang, D. Liu, and Q. Wei, "Finite-horizon neuro-optimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach," *Neurocomputing*, vol. 78, no. 1, pp. 14-22, Feb. 2012.

[28] R. Song and H. Zhang, "The finite-horizon optimal control for a class of time-delay affine nonlinear system," *Neural Computing & Applications,* Jul. 2011.

[29] D. E. Kirk, *Optimal control theory: an introduction*. New York: Dover Publications, 2004, pp. 54,75.

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

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

[32] J. R. Wertz, *Spacecraft attitude determination and control.* Netherlands: Kluwer Academic Publishers, 1978, pp. 521-570.

[33] P. H. Zipfel, *Modeling and simulation of aerospace vehicle dynamics.* Reston, VA: AIAA, 2000, pp.119-121.

2. FIXED-FINAL-TIME OPTIMAL CONTROL OF NONLINEAR SYSTEMS WITH TERMINAL CONSTRAINTS

Ali Heydari and S.N. Balakrishnan

ABSTRACT

A model-based reinforcement learning algorithm is developed in this paper for fixed-final-time optimal control of nonlinear systems with soft and hard terminal constraints. Convergence of the algorithm, for linear in the weights neural networks, is proved through a novel idea by showing that the training algorithm is a contraction mapping. Once trained, the developed neurocontroller is capable of solving this class of optimal control problems for different *initial conditions*, different *final times*, and different *terminal constraint surfaces* providing some mild conditions hold. Three examples are provided and the numerical results demonstrate the versatility and the potential of the developed technique.

I. INTRODUCTION

Many control, guidance, and path planning problems are classified as 'terminal control' problems [1]. A terminal control problem is a finite-horizon problem with *soft* or *hard* constraint on the terminal states. One approach to solving terminal control problems of nonlinear systems is formulating the problem in an optimal control framework. For this class of problems, the Hamilton-Jacobi-Bellman (HJB) equation is very difficult to solve since the solution is time-dependent. An open loop solution is dependent on the selected initial condition (IC) and the time-to-go [2]. Available methods in the literature can be classified as classical and intelligent control methods.

A. Classical Approaches to Terminal Control Problems

One approach in classical methods is calculating the open loop solution through a numerical method, e.g., the shooting method and then using techniques like Model Predictive Control (MPC) for closing the control loop as done in [3]. A limitation of this approach is the fact that it holds only for one set of specified IC and time-to-go. Another method, called the Approximate Sequence of Riccati Equation (ASRE) developed in [4] provides a closed form solution to the problem but again only for a pre-specified IC and time-to-go. This method is based on the calculation of a sequence of Riccati equations until they converge, and then using the converged result for control calculation. Finite-

horizon State Dependent Riccati Equation (Finite-SDRE) method, developed in [5], offers a suboptimal closed form solution to this class of problems. Finite-SDRE provides solutions for different ICs and final times in real-time and shows a lot of potential in the applications, but can accommodate only soft terminal constraints.

Series-based solutions to the optimal control problem with hard terminal constraints were investigated in [6-8]. In [6], a closed form solution was found by using a Taylor series expansion of the cost-to-go function. Series-based methods are suitable for systems whose dynamics are given in a polynomial form and comprise only weak nonlinearities. The series can diverge for problems with a large nonlinearity. This limitation motivated the authors of [7] to propose a divide-and-conquer scheme. This scheme is based on determining some waypoints to split the main problem into several simpler problems for which the series based method does not produce significant midcourse errors. However, this method requires some extra numerical optimization to find waypoints for each IC. Moreover, the number of required waypoints needs to be selected through trial and error in order to avoid divergence. The generating functions method proposed in [8] is a different series-based solution where the terminal constraint is a given point. In [9] a Generalized Hamilton-Jacobi-Bellman equation [10] was used with some modifications. Convergence of this method was proved for the unconstrained case in [10], but not for the constrained problem.

B. Intelligent Approaches to Terminal Control Problems

The use of intelligent control for solving finite-horizon optimal control problems was considered in [11-18]. Authors of [11] developed a neurocontroller for a problem with state constraints using ‘adaptive critics’ (AC) scheme [19,20] with time dependent weights. This controller is developed for an agile missile maneuver. It is however, a scalar problem wherein the final state and the control have a direct relationship. Hence, in a discrete formulation, the final state can be achieved from any state at the previous step. Continuous-time problems are considered in [12] where the time-dependent weights are calculated through a backward integration of the HJB equation. In [13] a single neural network (NN) with a single set of weights was proposed which takes the time-to-go as an *input* along with the states and generates the fixed-final-time optimal control for discrete-time nonlinear systems with *soft* terminal constraint. An adaptive critic based solution to

optimal *tracking* problems with fixed-final-time was developed in [14]. The neurocontroller provides the capability of tracking a single reference trajectory or a family of reference trajectories with *soft* terminal constraints.

The finite-horizon problem with *unspecified terminal time* and a fixed terminal state was considered in [15-18]. The algorithms developed in these papers lead to an *infinite* sequence of controls. Therefore, the control needs to be applied for an infinite time horizon to optimize the cost-function and bring the states to the origin. To overcome this problem, the authors suggested truncating the control sequence in order to end up with the so called ϵ-optimal solution, which will hence, have a finite horizon. The truncation is done such that the remaining horizon is long enough in order for the cost-to-go truncation error to be less than a given $\epsilon > 0$. Moreover, the neurocontrollers developed in [15,16] can only control one IC, and once the IC is changed, the network needs to be re-trained to give the optimal solution for the new IC. The neurocontrollers in [16,17] require the system to be such that the state can be brought to the origin in one step, from any given state. Systems with invertible input gain matrices in a control-affine discrete-time form satisfy this requirement. A newly developed controller in [18] has removed the restrictions of fixed initial condition and being able to go to the origin in one step.

C. Contributions of This Study

The first part of the development in this paper consists of formulating an approximate dynamic programming (ADP) based neurocontroller for fixed-final-time optimal control of systems with a *soft* terminal constraint. The closest neurocontrollers existing in the literature for such problems are [13,14]. The main difference between the cited references and the proposed scheme in this study is the use of the cost-function based ADP, known as Heuristic Dynamic Programming [20]. The developments in [13,14] are the costate based ADP, known as Dual Heuristic Programming. After discussing the solution to the problem with soft constraints, some modifications are performed in the network structure and the training algorithm to handle *hard* terminal constraints. These modifications are the main contributions of this paper. Another contribution of this study is proving the convergence of the network weights through a novel idea. It is done for the selected linear in the weights NN, by showing that the

successive approximation based weight update is a contraction mapping [21] within the compact domain of interest.

As compared with [11], the controller developed in here can be used in a multivariable setting while the method presented in [11] is developed for the scalar dynamics of an agile missile. Neurocontrollers developed in [12-14] do not admit hard terminal constraint, which is the main contribution of this study. Finally, the method here does not have the restrictions in [18] as the need for truncating the control in order to end up with a finite-horizon solution and also the requirement of the terminal constraint being a 'point'. The proposed technique can handle terminal constraints that are a point, a curve, or a surface which can be a nonlinear function of the state space elements. Moreover, the selected approach in this study directly results in a finite sequence of controls, hence, no truncation is required.

The trained (linear in the weights) NN in this work offers a feedback solution though trained offline. Furthermore, notable features of the proposed technique include: a) Optimal control of any set of initial conditions in a compact set, as long as the resulting state trajectory lies within the domain on which the network is trained. b) Optimal control for any final time not greater than the final time for which the network is trained (Bellman principle of optimality [2]). c) Providing optimal control in a closed form versus the terminal surface/curve/point. Therefore, if, for example, the terminal point is changed, no retraining is needed for the network to give optimal solution for the new terminal point.

The rest of this paper is organized as follows: The problem formulation is given in section II. Solution to the linear problem is discussed in section III. Solution for the nonlinear problem is developed in section IV. Analysis of results from numerical simulations is presented in section V, and the conclusions are given in section VI.

II. PROBLEM FORMULATION

Consider the nonlinear system presented in the control-affine form

$$\dot{x}(t) = f\big(x(t)\big) + g\big(x(t)\big)u(t), \tag{1}$$

where smooth functions $f: \mathbb{R}^n \to \mathbb{R}^n$ and $g: \mathbb{R}^n \to \mathbb{R}^{n\times m}$ represent the dynamics of the system. The system is assumed not to have a finite escape time. Vectors $x(t) \in \mathbb{R}^n$ and

$u(t) \in \mathbb{R}^m$ represent the state and control vectors, respectively. Integers m and n denote, respectively, the dimension of the state and control vectors. Initial conditions are given by $x(t_0) = x_0$ for some $x_0 \in \mathbb{R}^n$. Assume cost function

$$J = \phi\left(x(t_f)\right) + \frac{1}{2}\int_{t_0}^{t_f}\left(Q\big(x(t)\big) + u(t)^T R u(t)\right)dt, \tag{2}$$

where convex positive semi-definite functions $Q: \mathbb{R}^n \to \mathbb{R}$ and $\phi: \mathbb{R}^n \to \mathbb{R}$ penalize the states error during the horizon and at the final time, respectively. The positive definite matrix R penalizes the control effort. Since function $\phi(x(\,t_f))$ puts a penalty on the terminal state error, it is considered as a *soft* terminal constraint. Let the *hard* terminal constraint be given by $\psi\big(x(t_f)\big) = \psi_f$ for some smooth function $\psi: \mathbb{R}^n \to \mathbb{R}^l$ and $\psi_f \in \mathbb{R}^l$, where $l \leq n$. The problem is determining some control $u(t)$, $t \in [t_0, t_f)$ such that not only cost function (2) subject to state equation (1) is optimized, but also the hard terminal constraint is satisfied. The initial time and the fixed final time are denoted with t_0 and t_f, respectively.

III. SOLUTION TO LINEAR PROBLEMS

The controller developed in this study is motivated by the solution to the corresponding discrete-time problem with linear dynamics and a linear terminal constraint. Hence, it is instructive to study the solution process for the linear problem first. Assume the linear discrete-time dynamics

$$x_{k+1} = Ax_k + Bu_k,$$

with a quadratic cost function

$$J = \frac{1}{2}x_N^T S x_N + \frac{1}{2}\sum_{k=0}^{N-1}(x_k^T \bar{Q} x_k + u_k^T \bar{R} u_k), \tag{3}$$

and the linear hard terminal constraint $Cx_N = \psi_f$ for a given matrix $C \in \mathbb{R}^{l\times n}$. Subscripts denote the discrete time indices $k = 0,1,..,N$, where N denotes the final time-step. Constant matrices S and $\bar{Q}$ are assumed to be positive semi-definite and matrix $\bar{R}$ is assumed to be positive definite.

To ensure the satisfaction of the hard terminal constraint, one may augment cost function (3) by the term $v^T(Cx_N - \psi_f)$ where $v \in \mathbb{R}^l$ is a constant valued Lagrange multiplier [22]. The optimal cost-to-go at each instant may be denoted with $J_k^*(x_k)$ to

emphasize its dependency on the current state, x_k, and on the time-to-go, $N-k$. The solution to the problem is given by the discrete-time HJB equation (Bellman equation) [2]

$$J_N^*(x_N) = \frac{1}{2} x_N^T S x_N, \tag{4}$$

$$J_k^*(x_k) = \frac{1}{2}\left(x_k^T \bar{Q} x_k + {u_k^*}^T \bar{R} u_k^*\right) + J_{k+1}^*(x_{k+1}),\ k = 0,1,\dots,N-1, \tag{5}$$

$$u_k^* = -\bar{R}^{-1} B^T \frac{\partial J_{k+1}^*(x_{k+1})}{\partial x_{k+1}},\ k = 0,1,\dots,N-1, \tag{6}$$

where the optimal control at time k is denoted with u_k^*, and the gradient $\partial J_{k+1}^*(x_{k+1})/\partial x_{k+1}$ is formed as a column vector. Adapting the assumed form for the optimal cost-to-go of the respective continuous-time problem in [6] to the discrete-time problem at hand, the form

$$J_k^*(x_k) = \frac{1}{2} x_k^T P_k x_k + v^T W_k^T x_k + \frac{1}{2} v^T T_k v - v^T \psi_f \tag{7}$$

is selected for $J_k^*(x_k)$ and it is shown that it satisfies HJB equation (5) subject to final condition (4), where matrices $P_k \in \mathbb{R}^{n\times n}$, $W_k \in \mathbb{R}^{n\times l}$, and $T_k \in \mathbb{R}^{l\times l}$ are time-step dependent unknowns and vector v depends on the selected IC. It should be noted that the satisfaction of HJB equation (5) subject to final condition (4) provides the necessary and sufficient condition for optimality [2]. Therefore, any function $J: \mathbb{R}^n \to \mathbb{R}$ which satisfies HJB equation (5) along with final condition (4), will be the optimal cost-to-go function for the problem. In other words, even though the selected form given in (7) is just an assumption, the satisfaction of the sufficient condition leads to the optimality of the result. Moreover, the representation given in (7) is compatible with the assumed representation for the costate vector in [22].

Evaluating J_k^* given in Eq. (7) at $k+1$, leads to

$$J_{k+1}^* = \frac{1}{2} x_{k+1}^T P_{k+1} x_{k+1} + v^T W_{k+1}^T x_{k+1} + \frac{1}{2} v^T T_{k+1} v - v^T \psi_f. \tag{8}$$

Replacing x_{k+1} in (8) with $Ax_k + Bu_k^*$, and utilizing the result in equation (6), one has

$$u_k^* = -\bar{R}^{-1} B^T (P_{k+1}(Ax_k + Bu_k^*) + W_{k+1} v). \tag{9}$$

Eq. (9), after some algebraic manipulations, leads to

$$u_k^* = -Y^{-1} B^T (P_{k+1} A x_k + W_{k+1} v). \tag{10}$$

where $Y \equiv \bar{R} + B^T P_{k+1} B$. Therefore,

$$x_{k+1} = (A - BY^{-1}B^T P_{k+1} A)x_k - BY^{-1}B^T W_{k+1} v. \quad (11)$$

Substituting J_k^*, J_{k+1}^*, and u_k^*, in Eq. (5) using Eqs. (7), (8) (in which x_{k+1} is substituted using (11)), and (10), respectively, leads to

$$\begin{aligned}
&\frac{1}{2}x_k^T P_k x_k + v^T W_k^T x_k + \frac{1}{2}v^T T_k v - v^T \psi_f \\
&\qquad = \frac{1}{2}x_k^T \bar{Q} x_k + \frac{1}{2}x_k^T A^T P_{k+1} B Y^{-1} \bar{R} Y^{-1} B^T P_{k+1} A x_k \\
&+\frac{1}{2}v^T W_{k+1}^T B Y^{-1} \bar{R} Y^{-1} B^T W_{k+1} v + v^T W_{k+1}^T B Y^{-1} R Y^{-1} B^T P_{k+1} A x_k \\
&\quad +\frac{1}{2}x_k^T (A - BY^{-1}B^T P_{k+1} A)^T P_{k+1} (A - BY^{-1}B^T P_{k+1} A)x_k \\
&\qquad +\frac{1}{2}v^T W_{k+1}^T B Y^{-1} B^T P_{k+1} B Y^{-1} B^T W_{k+1} v \\
&-v^T W_{k+1}^T B Y^{-1} B^T P_{k+1} (A - BY^{-1}B^T P_{k+1} A)x_k + v^T W_{k+1}^T (A - BY^{-1}B^T P_{k+1} A)x_k \\
&\qquad -v^T W_{k+1}^T B Y^{-1} B^T W_{k+1} v + \tfrac{1}{2}v^T T_{k+1} v - v^T \psi_f. \quad (12)
\end{aligned}$$

Eq. (12) provides a relation between the time-varying unknown matrices P_k, W_k, and T_k. Note that this relation has to hold for every $x \in \mathbb{R}^n$ and every $v \in \mathbb{R}^l$, therefore, terms can be separated based on their dependency on x and on v. In other words, there are three set of terms; 1) terms which depend only on x, 2) terms which depend on both x and v, and 3) terms which only depend on v. Separating the three sets of terms leads to three equations. In order for the equations to hold for every x and every v, one can remove the x and v variables and force the unknown matrices to satisfy the equations. This process leads to the three difference equations given below

$$P_k = A^T P_{k+1} A + \bar{Q} - A^T P_{k+1} B(\bar{R} + B^T P_{k+1} B)^{-1} B^T P_{k+1} A, k = 0,1,..,N-1, \quad (13)$$

$$W_k = (A - B(\bar{R} + B^T P_{k+1} B)^{-1} B^T P_{k+1} A)^T W_{k+1}, k = 0,1,\dots,N-1, \quad (14)$$

$$T_k = T_{k+1} - W_{k+1}^T B(\bar{R} + B^T P_{k+1} B)^{-1} B^T W_{k+1}, k = 0,1,\dots,N-1, \quad (15)$$

The final conditions for the difference equations can be found using (7) and $Cx_N = \psi_f$ in Eq. (4). This process leads to $P_N = S$, $W_N = C^T$, and $T_N = 0$.

Denoting the l-dimensional null vector with 0_l, a necessary condition for optimality of J_k^* with respect to the Lagrange multiplier v, to enforce the hard terminal constraint, is

$$\partial J_k^*(x_k)/\partial v = 0_l, k = 0,1, \dots, N-1. \tag{16}$$

Using (7) in (16) provides

$$\partial J_k^*(x_k)/\partial v = W_k^T x_k + T_k v - \psi_f = 0,$$

therefore,

$$v = T_k^{-1}(\psi_f - W_k^T x_k), k = 0,1,..,N-1, \tag{17}$$

which gives the value of v, providing T_k is invertible for $k \neq N$. If T_k is not invertible the problem is called abnormal [1]. Note that T_N is singular, therefore, v cannot be calculated at the final time. But since v is a constant vector, one may calculate it at another time step, e.g., at $k = 0$. Finally, as expected, the 'difference' equations (13) to (15) correspond to similar 'differential' equations derived in [6]. Considering these results for the linear problem, in the next section the solution to the nonlinear problem is sought.

IV. APPROXIMATE DYNAMIC PROGRAMMING APPROACH TO NONLINEAR PROBLEMS

A RL scheme is proposed in an ADP framework [23] in this study as a solution technique to the terminal control problem. First we motivate the utilization of this approach for fixed-final-time optimal control with soft terminal constraint and then proceed to using it for the problems with hard terminal constraints.

A. Adaptive Critics for Optimal Control with Soft Terminal Constraint

In a finite-horizon dual network implementation of the ADP called adaptive critics (AC) for approximating the optimal control and the optimal cost-to-go, two NNs named actor and critic need to be trained. The NNs capture the mapping between a given state and the time-to-go (final time minus the current time,) as inputs, to the optimal control and optimal cost-to-go, respectively, as outputs. The first step is to discretize system (1) by selecting a small sampling time Δt. The discretization using Euler integrations leads to

$$x_{k+1} = \bar{f}(x_k) + \bar{g}(x_k)u_k, k = 0, 1, 2, \dots, N, \tag{18}$$

where $N = (t_f - t_0)/\Delta t$, $x_k \equiv x(k\Delta t + t_0)$, $\bar{f}(x) \equiv x + \Delta t f(x)$ and $\bar{g}(x) \equiv \Delta t g(x)$. The discretized cost function may be given by

$$J = \phi(x_N) + \frac{1}{2}\sum_{k=0}^{N-1}(\bar{Q}(x_k) + u_k^T \bar{R} u_k), \tag{19}$$

where $\bar{Q}(x) \equiv \Delta t Q(x)$, and $\bar{R} \equiv \Delta t R$. The terminal error penalizing function $\phi(.)$, i.e., the soft terminal constraint, remains intact in the discretization.

By selecting 'linear in the weights' networks, the expressions for the actor (control) and the critic (cost), can be written as

$$u_k(x_k) = V_k^T \sigma(x_k), k = 0,1, \dots, N-1, \tag{20}$$

$$J_k(x_k) = W_k^T \rho(x_k), k = 0,1, \dots, N. \tag{21}$$

where $J_k(x_k)$ denotes the *approximate* optimal cost-to-go at current state x_k and time-to-go $N - k$, and $u_k(x_k)$ represents the *approximate* optimal control at current state x_k and time-to-go $N - k$. Matrices $V_k \in \mathbb{R}^{p \times m}$ and $W_k \in \mathbb{R}^q$ are the unknown weights of the actor and the critic networks at time step k, respectively. The selected basis functions are given by $\sigma: \mathbb{R}^n \to \mathbb{R}^p$ and $\rho: \mathbb{R}^n \to \mathbb{R}^q$ for p and q being positive integers denoting the number of neurons in the respective network. Note that the weight matrices are time-dependent, in order to accommodate the time-dependency of the approximated $J_k(x_k)$ and $u_k(x_k)$. For simplicity in the notation, argument x_k is omitted from $J_k(x_k)$ and $u_k(x_k)$ in some places.

The discrete-time HJB equation for the nonlinear system and non-quadratic cost function terms is given by

$$J_N^*(x_N) = \phi(x_N),\ J_k^*(x_k) = \frac{1}{2}(\bar{Q}(x_k) + u_k^*(x_k)^T \bar{R} u_k^*(x_k)) + J_{k+1}^*(x_{k+1}^*),$$

$$k = 0,1, \dots, N-1, \tag{22}$$

$$u_k^*(x_k) = -\bar{R}^{-1}\bar{g}(x_k)^T \left.\frac{\partial J_{k+1}^*}{\partial x_{k+1}}\right|_{x_{k+1}^*},\ k = 0,1, \dots, N-1. \tag{23}$$

where $x_{k+1}^* \equiv \bar{f}(x_k) + \bar{g}(x_k)u_k^*(x_k)$ and gradient $\partial J_{k+1}^*/\partial x_{k+1}$ is formed as a column vector. The reinforcement learning scheme can be derived from the HJB equation for learning these unknowns for the fixed-final-time problem once (23) is replaced with

$$u_k^{i+1}(x_k) = -\bar{R}^{-1}\bar{g}(x_k)^T \left.\frac{\partial J_{k+1}^*}{\partial x_{k+1}}\right|_{x_{k+1}^i}, k = 0,1,\dots,N-1, \quad (24)$$

where $x_{k+1}^i \equiv \bar{f}(x_k) + \bar{g}(x_k)u_k^i(x_k)$ and superscript i denotes the index of iteration. The iteration starts with an initial guess on $u_k^0(x_k)$ and $u_k^{i+1}(x_k)$ is calculated based on $u_k^i(x_k)$ using (24) for each selected $k = 0,1,\dots,N-1$. The converged value of $u_k^i(x_k)$ is denoted with $u_k^*(x_k)$ and is used in (22) for calculating $J_k^*(x_k)$. Note that in a dual network AC scheme for *finite horizon* optimal control, 'learning' takes place in the iteration based controller synthesis, as seen in (24). Once state-control relationship is learned, optimal cost-to-go is obtained as a 'mapping' process as in (22).

Considering (21), one has $\partial J_{k+1}/\partial x_{k+1} = \nabla\rho(x_{k+1})^T W_{k+1}$ where the ∇ operator denotes the gradient of a function with respect to x. Note that $\left(\nabla\rho(x_{k+1})\right)^T$ is denoted with $\nabla\rho(x_{k+1})^T$, which is an $n \times q$ matrix. Using (20) and (21) in (22) and (24) leads to the weight update equations for the actor and the critic.

$$W_N^T\rho(x_N) = \phi(x_N), \quad (25)$$

$$V_k^{i+1^T}\sigma(x_k) = -\bar{R}^{-1}\bar{g}(x_k)^T\nabla\rho\left(\bar{f}(x_k) + \bar{g}(x_k)V_k^{i^T}\sigma(x_k)\right)^T W_{k+1},$$

$$k = 0,1,\dots,N-1, \quad (26)$$

$$W_k^T\rho(x_k) = \frac{1}{2}\bar{Q}(x_k) + \frac{1}{2}\sigma(x_k)^T V_k\bar{R}V_k^T\sigma(x_k) + W_{k+1}^T\rho\left(\bar{f}(x_k) + \bar{g}(x_k)V_k^T\sigma(x_k)\right),$$

$$k = 0,1,\dots,N-1. \quad (27)$$

The training should be done in a backward fashion from $k = N$ to $k = 0$. At each time step k, starting with an initial guess on V_k^0, one iterates using (26) until the iteration converges to some V_k. It will then be used in (27) along with W_{k+1}, which is already learned in the previous stage, to calculate W_k in one shot. This process is detailed in the algorithm given below.

Algorithm 1

Step 1: Train W_N such that $W_N^T\rho(x_N) \cong \phi(x_N)$ for different randomly selected $x_N \in \Omega$, where compact set Ω denotes the domain of interest.

Step 2: Set $k = N-1$.

Step 3: Set $i = 0$, and select a guess on V_k^0.

Step 4: Train V_k^{i+1} such that

$$V_k^{i+1^T}\sigma(x_k) \cong -\bar{R}^{-1}\bar{g}(x_k)^T\nabla\rho\left(\bar{f}(x_k) + \bar{g}(x_k)V_k^{i^T}\sigma(x_k)\right)^T W_{k+1}$$

for different randomly selected $x_k \in \Omega$.

Step 5: Set $i = i + 1$. Repeat Step 4 until $\|V_k^{i+1} - V_k^i\|$ converges to a small value for different x_ks.

Step 6: Set $V_k = V_k^i$.

Step 7: Train W_k such that

$$W_k^T\rho(x_k) \cong \frac{1}{2}\bar{Q}(x_k) + \frac{1}{2}\sigma(x_k)^T V_k\bar{R}V_k^T\sigma(x_k) + W_{k+1}^T\rho\left(\bar{f}(x_k) + \bar{g}(x_k)V_k^T\sigma(x_k)\right),$$

for different $x_k \in \Omega$.

Step 8: Set $k = k - 1$. Go back to Step 3 until $k = 0$.

Remark 1: In Algorithm 1, the number of iterations is time-step dependent, i.e., each time-step k may be iterated for different number of iterations until the corresponding weights converge. Once the weights converge, the iteration index i resets to zero and the iteration starts for another time-step.

Remark 2: In Steps 1, 4, and 7 of Algorithm 1, the method of least squares, detailed in Appendix A, can be used in order to find the unknown weight versus the given parameters.

Remark 3: The solution developed here is for *fixed-final-time* problems. Another set of terminal control problems are *free-final-time* problems. However, free-final-time problems can be transformed to a fixed-final-time problem by changing the independent variable, providing the new independent variable changes monotonically with time and has a fixed final value. Once transformed to a fixed-final-time problem, the method developed here can be used for solving the problem. Interested readers are referred to [24] for an automatic landing problem of an aerial vehicle in which the touch-down time is free, but, the downrange (the travelled distance along the runway) is fixed and monotonically changes with time. It is shown that the change of independent variable from time to the downrange leads to a fixed-final-time problem.

B. Adaptive Critics for Optimal Control with Hard Terminal Constraint

In this section, the network structure and the training algorithm are modified to learn the optimal solution subject to the hard terminal constraint, i.e., condition $\psi(x_N) = \psi_f$ is enforced. This is the main contribution of this work, compared with the available methods in the literature, including [12-14]. Motivated by the linear solution, where the time-step dependent matrices used in (7) are calculated first and used in the subsequent calculation of v, the actor and critic networks are trained to approximate the optimal control and cost-to-go, respectively, based on *a given* vector v. In other words, the networks approximate $u_k^*(x_k, v)$ and $J_k^*(x_k, v)$. Once these relationships are learned, the necessary condition given in (16) is enforced to find the optimal value for v. Afterwards, the optimal v will be fed to the networks to generate the optimal solution. For this purpose, the following modified network structures are proposed:

$$u_k(x_k, v) = V_k^T \sigma(x_k) + \bar{V}_k^T \theta(x_k, v),\ k = 0,1,\dots,N-1, \tag{28}$$

$$J_k(x_k, v) = W_k^T \rho(x_k) + \bar{W}_k^T \eta(x_k, v) - v^T \psi_f,\ k = 0,1,\dots,N, \tag{29}$$

where $V_k \in \mathbb{R}^{p\times m}$ and $W_k \in \mathbb{R}^q$ are prior network weights, and the new weights $\bar{V}_k \in \mathbb{R}^{r\times m}$ and $\bar{W}_k \in \mathbb{R}^s$ are the weights of the augmented terms to the actor and the critic networks at time step k, respectively. The new basis functions are given by $\theta: \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^r$ and $\eta: \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^s$ for r and s being positive integers denoting the number of neurons in the respective augmented networks.

The selected form for approximate optimal cost-to-go (29) is motivated by the assumed representation for the cost-to-go in the linear problem, i.e., equation (7). The first term in the right hand side of (29) is motivated by the existence of $(1/2)x_k^T P_k x_k$ in (7). The second term in the right hand side of (29) is motivated by the terms $v^T W_k^T x_k + (1/2)v^T T_k v$ in (7). Considering this analogy, the basis functions $\eta(.,.)$ may be selected such that

$$\eta(x, 0) = 0, \forall x \in \mathbb{R}^n \text{ and } \eta(0, v) \neq 0, \exists v \in \mathbb{R}^l. \tag{30}$$

One natural selection for the basis functions is to form them as polynomials made up of different combinations of the elements of the network inputs. This design is selected in some of the simulation studies in this paper. In such a design, the conditions given by

(30) mean that a) all the basis functions need to have an element of v in them, and b) there has to be some basis functions which does not contain any elements of x. Given characteristics (30), $\theta(.,.)$ should be selected such that

$$\theta(x,0)=0,\forall x\in\mathbb{R}^n, \tag{31}$$

because it is supposed to be used in learning a term which is proportional to the gradient of $\eta(.,.)$ with respect to x.

Using (28) and (29) in the learning equations given by (22) and (24) leads to the new weight update equations for the actor and critic as

$$W_N^T\rho(x_N)+\bar{W}_N^T\eta(x_N,v)-v^T\psi_f=\phi(x_N), \tag{32}$$

$$V_k^{i+1^T}\sigma(x_k)+\bar{V}_k^{i+1^T}\theta(x,v)=-\bar{R}^{-1}\bar{g}(x_k)^T\left(\nabla\rho\left(x_{k+1}^i\right)^TW_{k+1}+\nabla\eta\left(x_{k+1}^i,v\right)^T\bar{W}_{k+1}\right), \tag{33}$$

$$W_k^T\rho(x_k)+\bar{W}_k^T\eta(x_k,v)=\frac{1}{2}\bar{Q}(x_k)+\frac{1}{2}u_k^T\bar{R}u_k+W_{k+1}^T\rho(x_{k+1})+\bar{W}_{k+1}^T\eta(x_{k+1},v), \tag{34}$$

in which parameters x_{k+1}^i, x_{k+1} and u_k, based on the weights, are given below

$$x_{k+1}^i=\bar{f}(x_k)+\bar{g}(x_k)\left(V_k^{i^T}\sigma(x_k)+\bar{V}_k^{i^T}\theta(x_k,v)\right),$$

$$u_k=V_k^T\sigma(x_k)+\bar{V}_k^T\theta(x_k,v),$$

$$x_{k+1}=\bar{f}(x_k)+\bar{g}(x_k)\left(V_k^T\sigma(x_k)+\bar{V}_k^T\theta(x_k,v)\right).$$

Note that the difference between x_{k+1} and x_{k+1}^i is that in the calculation of the former the converged values of V_k and $\bar{V}_k$ are used, whereas the latter is based on the current version, actually, ith version of V_k and $\bar{V}_k$.

In order to derive the independent weight update rules, starting with (32), separating terms with and without the dependence on v leads to two equations. They are equation (25) and

$$\bar{W}_N^T\eta(x_N,v)=v^T\psi(x_N). \tag{35}$$

This can be confirmed by evaluating (32) at $v=0$, considering (30), also noting that equation (32) needs to be valid for all $x,v\in\mathbb{R}^n$, hence, it needs to be valid for $v=0$, as well, which leads to (25). Due to constraint $\psi(x_N)=\psi_f$, equation (35) follows from the

remaining terms in (32), i.e., the v-dependent terms. Equations (25) and (35) provide us with the weights W_N and $\bar{W}_N$.

Following the same scheme mentioned above for finding a separate weight update laws for W_N and $\bar{W}_N$, separate weight updates can be found for V_k, $\bar{V}_k$, W_k, and $\bar{W}_k$, $k = 0,1,..,N-1$, also. This is done through separating terms *with and without dependence on* v. This separation was done for the linear case too, where separating terms in the form of $x^T(...)x$ from those of form $v^T(...)x$ led to (13) and (14). In here, an easy way to separate the v-independent terms from both sides of equations (33) and (34) is setting $v = 0$ because of (30) and (31). Note that, these equations need to hold for every $x, v \in \mathbb{R}^n$, hence, they need to be valid for $v = 0$, $x \in \mathbb{R}^n$, as well. Doing so simplifies (33) and (34) to (26) and (27), respectively. As for the weights of the v-dependent basis functions, one may train the v-independent weights, i.e., V_k and W_k, and then bring all the v-independent terms to the right hand sides of equations (33) and (34). This approach leads to the following equations to be used for learning $\bar{V}_k$ and $\bar{W}_k$

$$\bar{V}_k^{i+1^T}\theta(x_k, v) = -\bar{R}^{-1}\bar{g}(x_k)^T\left(\nabla\rho\left(x_{k+1}^i\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^i, v\right)^T\bar{W}_{k+1}\right) - V_k^T\sigma(x_k), \quad (36)$$

$$\bar{W}_k^T\eta(x_k, v) = \frac{1}{2}\bar{Q}(x_k) + \frac{1}{2}u_k^T\bar{R}u_k + W_{k+1}^T\rho(x_{k+1}) + \bar{W}_{k+1}^T\eta(x_{k+1}, v) - W_k^T\rho(x_k). \quad (37)$$

Note that x_{k+1}^i will now be different compared to (33), in the sense that it will be based on the converged V_k instead of V_k^i, i.e.,

$$x_{k+1}^i = \bar{f}(x_k) + \bar{g}(x_k)\left(V_k^T\sigma(x_k) + \bar{V}_k^{i^T}\theta(x_k, v)\right).$$

In summary, one learns V_k and W_k first, and then, uses them in the learning of $\bar{V}_k$ and $\bar{W}_k$. Algorithm 2 gives the training/learning process for finding V_k, W_k,$\bar{V}_k$, and $\bar{W}_k$, $\forall k$.

Algorithm 2

Step 1: Using Algorithm 1, find optimal weights W_k, $k = 0,1,\dots,N$ and V_k, $k = 0,1,\dots,N-1$.

Step 2: Train $\bar{W}_N$ such that $\bar{W}_N^T\eta(x_N, v) \cong v^T\psi(x_N)$ for different $x_N \in \Omega$ and $v \in \bar{\Omega}$. Set Ω denotes the compact set representing the domain of interest and $\bar{\Omega} \in \mathbb{R}^n$ is a compact set assumed the optimal v to belong to.

Step 3: Set $k = N-1$.

Step 4: Set $i = 0$, and select a guess on $\bar{V}_k^0$.

Step 5: Train $\bar{V}_k^{i+1}$ such that

$$\bar{V}_k^{i+1^T}\theta(x_k, v) \cong -\bar{R}^{-1}\bar{g}(x_k)^T\left(\nabla\rho\left(x_{k+1}^i\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^i, v\right)^T \bar{W}_{k+1}\right) - V_k^T\sigma(x_k),$$

for different randomly selected $x_k \in \Omega$ and $v \in \bar{\Omega}$, where $u_k^i = V_k^T\sigma(x_k) + \bar{V}_k^{i^T}\theta(x_k, v)$ and $x_{k+1}^i = \bar{f}(x_k) + \bar{g}(x_k)u_k^i$.

Step 6: Set $i = i + 1$. Repeat Step 5 until $\|\bar{V}_k^{i+1} - \bar{V}_k^i\|$ converges to a small value for different x_ks and vs.

Step 7: Set $\bar{V}_k = \bar{V}_k^i$.

Step 8: Train $\bar{W}_k$ such that

$$\bar{W}_k^T\eta(x_k, v) \cong \frac{1}{2}\bar{Q}(x_k) + \frac{1}{2}u_k^T\bar{R}u_k + W_{k+1}^T\rho(x_{k+1}) + \bar{W}_{k+1}^T\eta(x_{k+1}, v) - W_k^T\rho(x_k),$$

for different $x_k \in \Omega$ and $v \in \bar{\Omega}$, where $u_k = V_k^T\sigma(x_k) + \bar{V}_k^T\theta(x_k, v)$ and $x_{k+1} = \bar{f}(x_k) + \bar{g}(x_k)u_k$.

Step 9: Set $k = k - 1$. Go back to Step 4 until $k = 0$.

Remark 4: As detailed in Algorithm 2, one needs to select a domain to which vector v is supposed to belong, i.e., $\bar{\Omega}$. To have an estimate, one can solve for the linearized solution first or select a large domain and proceed.

Remark 5: In Steps 2, 5, and 8 of Algorithm 2, the method of least squares, detailed in Appendix A, can be used in order to find the unknown weight versus the given parameters.

Weight updates (26) and (36) utilized in Algorithm 2 are converging successive approximations. In order to prove the convergence, the following theorem whose proof is given in Appendix B, is presented.

Theorem 1: Let the basis functions used in the actor and critic networks be smooth in domains Ω and $\bar{\Omega}$. There exists some sampling time Δt to be used in the discretization of the smooth continuous dynamics given in (1) which using any sampling time smaller than that, the iterations on V_k^i and $\bar{V}_k^i$ in Algorithm 2 converge for any finite initial guess on $V_k^0 \in \mathbb{R}^{p\times m}$ and $\bar{V}_k^0 \in \mathbb{R}^{r\times m}$, $k = 0,1, \dots, N-1$.

In Theorem 1 the role of the sampling time in discretization of a continuous-time system is emphasized. It is worthwhile to discuss this issue in more details. Let's consider

the case of soft terminal constraint, for simplicity. Substituting (20) and (21) in optimal control equation (23), leads to

$$V_k^T \sigma(x_k) = -\bar{R}^{-1} \bar{g}(x_k)^T \nabla \rho \left(\bar{f}(x_k) + \bar{g}(x_k) V_k^T \sigma(x_k) \right)^T W_{k+1}. \tag{38}$$

Optimal weights V_k at each time instant k can be calculated from solving nonlinear equation (38), without using the iteration given in (26). Eq. (38) is actually m equations for $p \times m$ unknown elements of V_k. Following the remedy of evaluating Eq. (38) for p many random x_ks, similar to what is explained in the least squares process in Appendix A, one can end up with enough number of equations to find the $p \times m$ unknowns. However, there is no analytical solution to set of nonlinear equations (38) in general. Therefore, one needs to resort to numerical methods for solving the set of equations. Theorem 1 proves that for any given smooth dynamics and smooth basis functions, if the sampling time is small enough, the iterations given in (26) converge to the solution to the nonlinear equation (38). This convergence is proved for any initial guess on V_k^0 and any selected weight matrix R. However, if the sampling time is fixed, certain conditions need to hold on R and $g(x)$ in order for the iterations to converge. These conditions can be easily derived from the proof of Theorem 1.

In solution to linear problems, discussed in Section III, a similar issue is observed. To see this fact, one may consider optimal control equation (23) which leads to Eq. (9). This equation is the equation corresponding to (38). Similar to (38), the unknown, u_k^* in here, exists in both sides of the equation. However, equation (9) is linear and the analytical solution can be calculated, which is given by (10). If solution (10) was not available, one could use the following iterations, starting with any initial guess u_k^0, to find u_k^*.

$$u_k^{i+1} = -\bar{R}^{-1} B^T \left(P_{k+1} \left(A x_k + B u_k^i \right) + W_{k+1} v \right). \tag{39}$$

Following the idea presented in proof of Theorem 1, it is straightforward to show that (39) is a contraction mapping, and hence u_k^i converges to the solution of (9), providing the sampling time is small enough. Therefore, as long as one can solve the set of equations (9) in the linear problem, or the set of equations (38) in the NN-based solution to the nonlinear problem, no iterations and hence, no condition on the sampling time is

required. In practical implementation however, since the training is being done offline, one can always adjust the sampling time such that the convergence is achieved.

Considering Theorem 1, and the fact that the weight update rule was derived from the HJB equation (22) and (23) the converged weights satisfy

$$W_N^T \rho(x_N) + \bar{W}_N^T \eta(x_N, v) - v^T \psi_f = \phi(x_N) + \epsilon_N,$$

$$W_k^T \rho(x_k) + \bar{W}_k^T \eta(x_k, v) - v^T \psi_f = \frac{1}{2}\bar{Q}(x_k) + \frac{1}{2}u_k^T \bar{R} u_k + J_{k+1}(x_{k+1}) + \epsilon_k,$$

$$V_k^T \sigma(x_k) + \bar{V}_k^T \theta(x_k, v) = -\bar{R}^{-1} \bar{g}(x_k)^T \frac{\partial J_{k+1}}{\partial x_{k+1}} + \bar{\epsilon}_k,$$

where $\epsilon_k,\ k = 1,2,..,N$, and $\bar{\epsilon}_k,\ k = 1,2,\dots,N-1$, are the critic and actor network reconstruction errors, respectively. Using the Galerkin method of approximation [10] which simplifies to least squares for this problem [13,12] it has been shown that the reconstruction errors can be made arbitrarily small once the number of basis functions becomes large [10]. In the ideal case when $p, q, r, s \to \infty$ which results in $\epsilon_k = \bar{\epsilon}_k = 0, \forall k$, the generated J_k and u_k through the NNs in (28) and (29) satisfy the HJB equation (22) and (23) along with its final condition. Selecting v such that the necessary condition (16) is satisfied, the generated control u_k will, hence, be the optimal control at time k and J_k will be the optimal cost-to-go. In implementation, however, the number of basis functions is finite and the resulting solution is an approximation of the optimal solution.

B.1. Calculating optimal Lagrange multiplier

Once the network weights are trained, one needs to calculate the optimal v and feed it to the networks in order for the networks to output optimal results. The optimal v is such that the necessary optimality condition given in (16) is satisfied. Taking the gradient of J_k, given by (29), with respect to v and using it in (16) leads to

$$(\partial \eta(x_k, v)/\partial v)^T \bar{W}_k - \psi_f = 0_l. \tag{40}$$

The foregoing algebraic equation needs to be solved online based on the given IC to calculate the optimal v. Note that v is a constant vector depending on IC x_0. Therefore, in case of selecting rich basis functions to avoid numerical errors due to the network reconstruction errors, one needs to solve (40) only once for the selected IC. Errors may occur, however, because a finite number of basis functions may not be able to completely

and accurately capture the underlying nonlinear relationships. Therefore, in applications, it may be desirable to compute the values of v along the trajectory at some intervals. Differing values of v were observed in [6] with a Taylor series solution to the problem. The known singularity of (16) exists at the final time for the linear problems [1,6] due to the singularity of T_N in (17). If the values of v grow too large, then updating should stop close to the final time.

C. Adaptive Critics for Terminally Constrained Problems without State Penalizing Terms

Not having the state penalizing terms ($Q(.) = 0$ and $\phi(.) = 0$) simplifies Algorithm 2 considerably. The network weights V_k and W_k vanish and only $\bar{V}_k$ and $\bar{W}_k$ remain. This can be confirmed by looking at the weight update rules (25)-(27) where setting $\phi(x_N) = \bar{Q}(x_k) = 0$ gives the solution $V_k = W_k = 0$, $\forall k$. Hence, in such a case, the actor and critic networks simplify to

$$u_k(x_k, v) = \bar{V}_k^T \theta(x_k, v),\ k = 0,1, \dots, N-1, \tag{41}$$

$$J_k(x_k, v) = \bar{W}_k^T \eta(x_k, v) - v^T \psi_f,\ k = 0,1, \dots, N. \tag{42}$$

The training algorithm will be the same as Algorithm 2, except that one skips Step 1 in the algorithm and sets $V_k = W_k = 0$, $\forall k$, in the rest of the steps.

V. NUMERICAL ANALYSIS

A. Example 1: Scalar Problem with Soft Terminal Constraint

As the first example, a nonlinear scalar system given below is selected for simulating the 'soft' terminal constraint controller discussed in section IV.A.

$$\dot{x} = x^2 + u,$$

where $x \in \mathbb{R}$ and $u \in \mathbb{R}$. Notation x^2 denotes 'square of x' and should not be mistaken with the iteration index, as used in Eq. (24). The problem is defined as brining the given state to close to the origin in 1 s, i.e., $t_0 = 0$ and $t_f = 1\ s$. Cost function (2) with the terms given below is selected

$$\phi\left(x(t_f)\right) = 100\left(x(t_f)\right)^2,\ Q(x) = 0,\ R = 1.$$

In this example the following basis functions are used:

$$\rho(x) = \sigma(x) = [1, sin(x), cos(x), sin(2x), cos(2x), \dots, sin(kx), cos(kx)]^T,$$

where k is the highest order trigonometric function used in the basis functions. Note that the basis function selection in the adaptive critics design is an important step. Typically, one selects some set of basis functions, trains the network based on that, and evaluates the result to observe if the selected basis function is rich enough. If not, a richer set is selected.

Selecting the sampling time of $\Delta t = 0.005\, s$, the continuous dynamics is discretized to 200 steps. Parameter k in the basis functions, is selected equal to 5. This leads to 11 basis functions. The least squares process in Algorithm 1 is done using 50 random states at each iteration where $\Omega = \{x \in \mathbb{R}: -2 \leq x \leq 2\}$. The learning iterations were observed to converge in less than 5 iterations. Once the network is trained, it is used for controlling different initial conditions $x(t_0) \in \{-1.5, -0.5, 1, 2\}$. The resulting state histories are given in Fig. 1. For the comparison purpose, the optimal open loop numerical solutions for each one of the initial conditions, calculated separately using the direct method of optimization, are also plotted in this figure. As seen, the results of the developed neurocontroller are quite accurate through laying over the optimal results.

In order to evaluate the performance of the controller in providing optimal solution for shorter final times, the same NN is used for controlling initial condition $x(t_0) = 2$, but with different final times $t_f \in \{1, 0.75, 0.5, 0.25\}\ s$. The resulting state histories are plotted in Fig. 2. Comparing the AC based solutions with the optimal numerical solutions shows the versatility of the controller in approximating the optimal solutions to the problems with shorter horizons. Note that, once the optimal weights are calculated for time-indices $k = 0$ to $k = N$, the optimal weights for the shorter horizon of $k = 0, 1, \dots, N_1$, where N_1 is an integer smaller than N, are the weights with time indices of $k = N - N_1$ to $k = N$, based on Bellman principle of optimality [2].

Finally, in order to check the effect of using a less rich set of basis functions, two other sets of basis functions are selected with $k = 2$ and $k = 3$. After training the NNs with the new basis functions, their result in controlling initial condition $x(t_0) = 2$ are depicted in Fig. 3. In this figure, the resulting state history for the case of $k = 5$ and the optimal numerical result are also plotted. As seen, as the order of the basis functions increases, the results converge to the optimal solution. This analysis shows that selecting

$k = 2$ or $k = 3$ does not lead to a rich set of basis functions, compared with the case of $k = 5$.

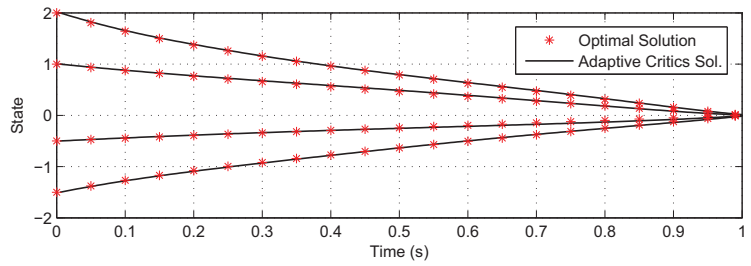

Fig. 1. State histories for different initial conditions (Example 1).

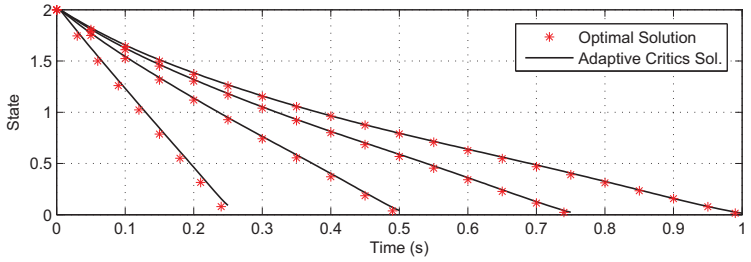

Fig. 2. State histories for different final times (Example 1).

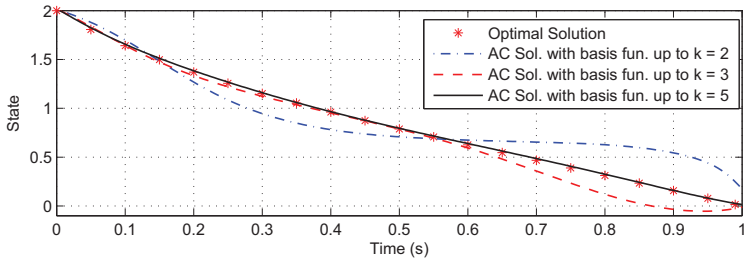

Fig. 3. State histories for different basis functions selections (Example 1).

B. Example 2: Second Order Problem with Hard Terminal Constraint

The second example is a benchmark nonlinear system, namely, Van der Pol's oscillator:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = (1 - x_1^2)x_2 - x_1 + u$$

where the subscripts on the state vector denotes the respective element of the matrix. This example is a problem with terminal hard constraint. The nonlinear terminal constraint is selected as the curve given by $\psi(x(t_f)) \equiv x_1^2(t_f) + x_2(t_f) = 1$, which leads to v being

a scalar. The sampling time is selected as $\Delta t = 0.01\ s$ with $t_0 = 0$, and $t_f = 1\ s$. This leads to $N = 100$, therefore, there will be 100 weight matrices for the actor and 100 weight matrices for the critic to be trained.

Cost function (2) with $\phi(x(t_f)) = 0$, $Q(x) = 0$, and $R = 1$ is selected. Because of the selected cost function terms, the neurocontrollers will be of the forms given by (41) and (42). Let the vector whose elements are all the *non-repeating* polynomials made up through multiplying the elements of vector X by those of vector Y be denoted with $X \otimes Y$. In this example the following basis functions are used:

$$\eta(x, v) = \left[v^2, (v \otimes x)^T, \left(v \otimes (x \otimes x)\right)^T, \left(v \otimes \left(x \otimes (x \otimes x)\right)\right)^T\right]^T,$$

$$\theta(x, v) = \left[v, (v \otimes x)^T, \left(v \otimes (x \otimes x)\right)^T\right]^T.$$

This selection leads to 10 neurons for the critic and 6 neurons for the actor. Therefore, the total number of weight elements will be $N(10 + 6) = 1600$.

The least squares in Algorithm 2 is carried out using 200 random states at each iteration where $\Omega = \{x \in \mathbb{R}^3 : -2 \leq x_i \leq 2, i = 1,2\}$ and $\bar{\Omega} = \{v \in \mathbb{R} : -10 \leq v \leq 10\}$. The learning process converged in less than 5 iterations as seen in Fig. 4 which denotes the evolution of the weights of the actor during the training iterations. Note that only some of the actor weights, namely $\bar{V}_0$, $\bar{V}_{50}$ and $\bar{V}_{99}$ are selected and presented in this figure, to avoid having too many plots in the figure. The resulting optimal weights are given in Fig. 5. This figure presents the value of each single weight element as a function of time. In other words, the evolution of the optimal weight matrices versus $t \in [t_0, t_f)$ is depicted. Once the network is trained, it is used for controlling a variety of ICs where x_1 varies between -1.5 and 1.5 in steps of 0.5, and x_2 varies between -1 and 2 in steps of 1. This selection leads to 28 different ICs. During the simulation, at each time-step, necessary condition (16) is enforced through updating scalar v by solving a single linear algebraic equation generated from (40). The resulting state trajectories are given in Fig. 6. As can be seen, the neurocontroller has done an excellent job in satisfying the final constraint through shaping the state trajectories to land on the curve representing the terminal nonlinear constraint (denoted by a thick blue plot in this figure.)

To evaluate the performance of the trained network for controlling the states in bringing them to the terminal curve at *different final times*, nine separate simulations are done for the different horizons of 1, 0.9, 0.8,…, 0.1 s. The resulting state trajectories for the given IC of $x(t_0) = [1, -1]^T$ are depicted in Fig. 7. This figure shows that the controller has perfectly controlled different problems with different final times. Finally, the same IC with the final time of 1 s is simulated for different ψ_fs of 2, 1, 0, -1, and -2 where $x_1^2(t_f) + x_2(t_f) = \psi_f$ represents the terminal constraint curve. The resulting state trajectories, given in Fig. 8, demonstrate the performance of the same trained network in solving the problems with shifted terminal curves or different given terminal states.

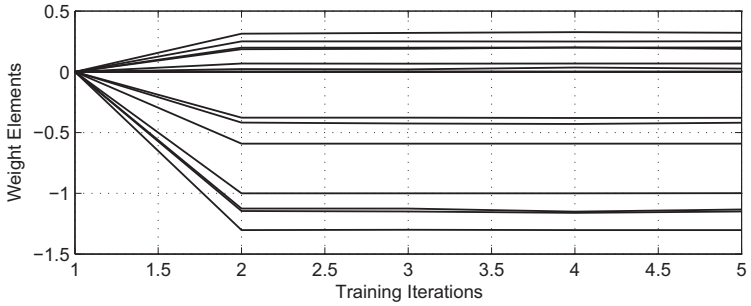

Fig. 4. Evolution of the actor weights versus training iterations (Example 2).

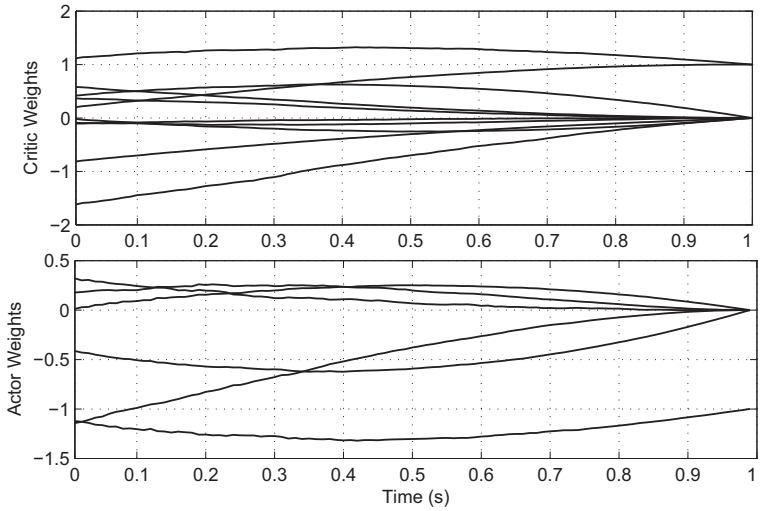

Fig. 5. Optimal weight histories (Example 2).

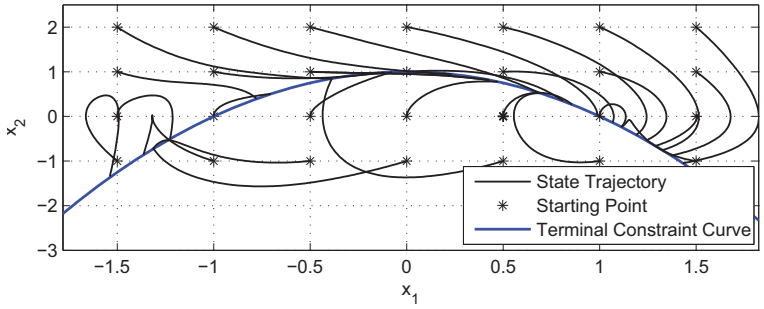

Fig. 6. State trajectories for different initial conditions (Example 2).

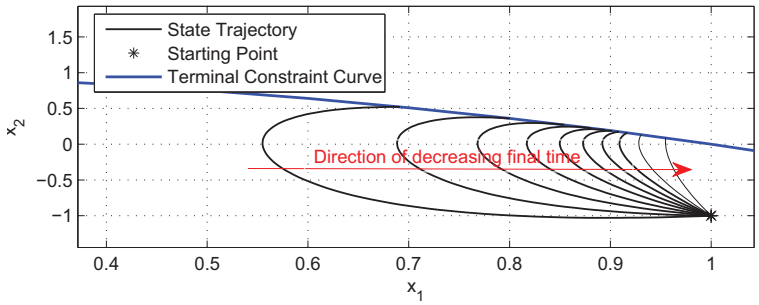

Fig. 7. State trajectories for different horizons (Example 2).

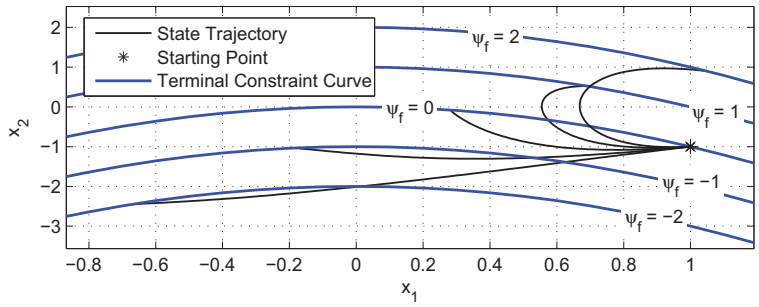

Fig. 8. State trajectories for different terminal curves (Example 2).

C. Example 3: Real-world Problem with Hard Terminal Constraint

As the last example, a practical problem is selected to show the applicability of the method to real world problems with hard terminal constraints. The selected problem is the detumbling of a rigid spacecraft in a given time, as investigated in [6]. In other words, a rigid spacecraft is tumbling with some initial angular velocities about each one of its three perpendicular axes. The controller needs to damp the angular velocities and the selected terminal state enforces the terminal angular velocities to be zero. The equations of motion are given by

$$\dot{x} = \mathbb{I}^{-1}(\mathbb{I}x \times x + u),$$

where $x \in \mathbb{R}^3$, $\mathbb{I} \in \mathbb{R}^{3\times3}$, and $u \in \mathbb{R}^3$ are the rigid body angular velocities, the moment of inertia of the rigid body, and the applied mechanical torque, respectively. Sign '×' denotes matrix cross product. The design parameters are selected to be the same with [6] in order to be able to compare the results with the series-based solution given to the same problem in the mentioned reference. Hence, $t_0 = 0$, $t_f = 2$ s, and $\mathbb{I} = diag(86.24,85.07,113.59)$ kgm^2.

Cost function (2) with the terms given below is selected

$$\phi\left(x(t_f)\right) = 0,\ Q(x) = x^T x,\ R = diag(1,1,1).$$

The terminal constraint is selected as $\psi\left(x(t_f)\right) \equiv x(t_f) = [0,0,0]^T$, which leads to $v \in \mathbb{R}^3$. In this example the following basis functions are used:

$$\rho(x) = \left[\ (x \otimes x)^T, \left(x \otimes (x \otimes x)\right)^T\right]^T,$$

$$\eta(x,v) = [\ (v \otimes v)^T, (v \otimes x)^T,]^T,$$

$$\sigma(x) = [x^T, (x \otimes x)^T]^T.$$

$$\theta(x,v) = v.$$

Selecting the sampling time of $\Delta t = 0.02\,s$ the continuous dynamics is discretized to 100 steps. The least squares in Algorithm 2 are done using 500 random states at each iteration where $\Omega = \{x \in \mathbb{R}: -2 \le x \le 2\}$ and $\overline{\Omega} = \{v \in \mathbb{R}: -2 \times 10^4 \le v \le 2 \times 10^4\}$. The learning iterations were observed to converge in around 4 iterations. Once the networks are trained, they are used for controlling the IC of $x_0 = [-0.4,0.8,2]^T$ rad/s, as simulated in Ref. [6]. The resulting state and control histories are given in Figs. 9 and 10, respectively. For comparison, the optimal open loop solution, calculated in [6], and the series based solution developed in the same reference are plotted in these figures. As seen in Fig. 9, the adaptive critics solution developed in this study has done a nice job in providing a solution which is very close to the optimal numerical solution, while the series based solution is not as accurate as the adaptive critics solution. This fact can be seen in Fig. 10, where the deviations of the series based solution form the optimal control

can be easily observed, while the adaptive critics solution has accurately generated the optimal solution. Comparing the adaptive critics solution with the open loop optimal solution it should be noted that the adaptive critics solution has the advantages of providing *feedback* solution to different initial conditions, different terminal points, and different final-times, as examined in Examples 1 and 2. While, each time that one of these parameters changes, the open loop numerical solution loses its validity and a new solution has to be calculated.

In these simulations, for the best result, vector v was updated at each time-step through solving a set of three linear algebraic equations resulting from (40). The resulting histories for v are plotted in Fig. 11. As seen in this figure, the elements of vector v have been almost constant during the simulation. The changes in the elements are due to the numerical error in approximating the optimal cost-to-go and optimal control using a finite number of basis functions.

This example shows that the developed method has a promising performance in real-world engineering problems. The real-time computational burden of the method is limited to the evaluation of the NN outputs given the inputs, and solving the algebraic equation (40), in real-time. Note that, Eq. (40) is not required to be solved at every single time-step. It can be solved every, for example, ten time steps in order to update v. Doing so further decreases the real-time computational load. As for the storage requirement, it should be noted that for online calculation of the control, one needs matrices $\bar{W}_k$, V_k, and $\bar{V}_k$. The number of elements of these matrices is 27 for the selected basis functions. Considering $N = 100$, the total number of 2700 real numbers are required to be stored for online control. This number for a solution to the linearized problem, which is only valid in a close vicinity of the origin by definition, is 2100, to store matrices P_k, W_k, and T_k defined in Eq. (7). Comparing 2700 with 2100, it can be seen that the storage requirement of the developed method is not prohibitive, compared to the solution to the linearized problem.

VI. CONCLUSIONS

A new algorithm was developed in the framework of approximate dynamic programming for finding fixed-final-time optimal control of nonlinear systems subject to

terminal nonlinear constraints. Convergence of the network weights was proved. From the numerical results, it can be concluded that the developed technique is quite versatile.

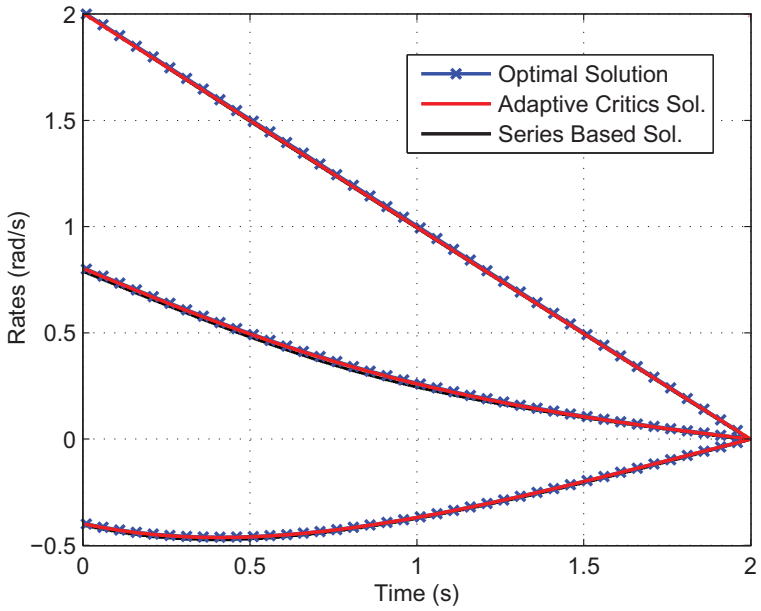

Fig. 9. State histories (Example 3).

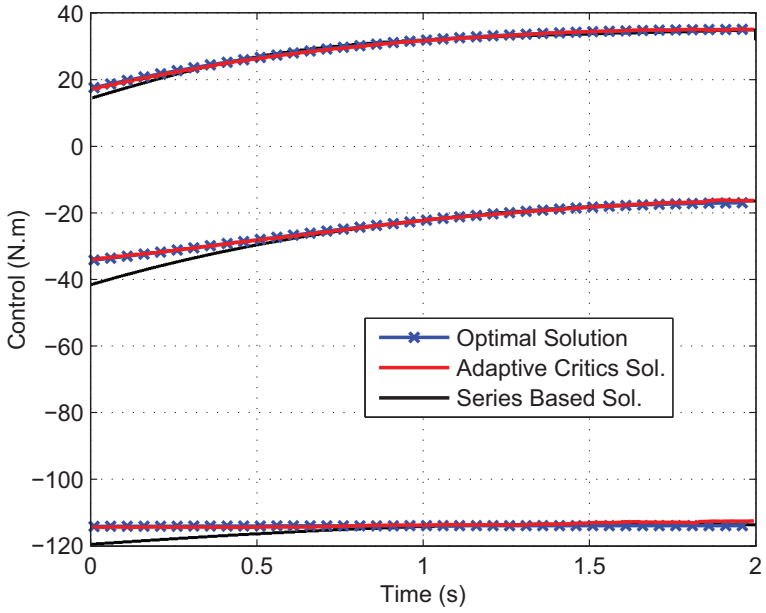

Fig. 10. Control histories (Example 3).

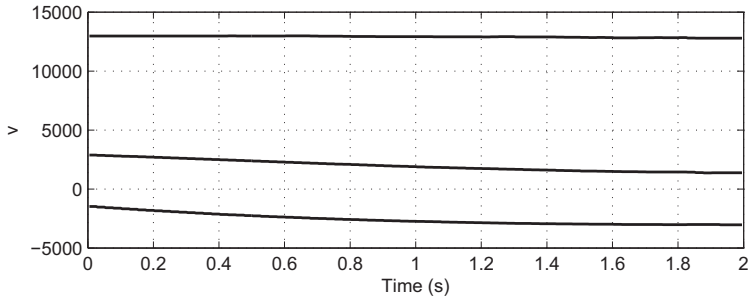

Fig. 11. Lagrange multiplier histories (Example 3).

ACKNOWLEDGEMENT

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

APPENDIX A

Equations (26), (27), (36), and (37), used in Algorithms 1 and 2, give the weight update rules for V_k, W_k, $\bar{V}_k$ and $\bar{W}_k$, respectively. The least squares method can be used for rewriting these equations such that the unknown weights are explicitly given in terms of known parameters. In this appendix, the process for finding such an equation for $\bar{V}_k$ is explained and one can easily find the corresponding equation for the other weight matrices.

To perform least squares for the weight update of $\bar{V}_k$, instead of one random x and random v, n random xs and n random vs denoted with $x^{(i)}$ and $v^{(i)}$, respectively, where $i \in \{1,2,\dots,n\}$, are selected. Denoting the right hand side of equation (36) resulting from each one pair of $x^{(i)}$ and $v^{(i)}$ with $\mathcal{Y}\left(x^{(i)}, v^{(i)}\right)$, the objective is finding $\bar{V}_k$ such that it solves

$$\begin{cases} \bar{V}_k^T \theta\left(x^{(1)}, v^{(1)}\right) = \mathcal{Y}\left(x^{(1)}, v^{(1)}\right) \\ \bar{V}_k^T \theta\left(x^{(2)}, v^{(2)}\right) = \mathcal{Y}\left(x^{(2)}, v^{(2)}\right) \\ \vdots \\ \bar{V}_k^T \theta\left(x^{(n)}, v^{(n)}\right) = \mathcal{Y}\left(x^{(n)}, v^{(n)}\right) \end{cases}. \tag{A.1}$$

Define

$$\boldsymbol{\theta} \equiv \left[\theta\left(x^{(1)}, v^{(1)}\right) \;\; \theta\left(x^{(2)}, v^{(2)}\right) \;\; \dots \;\; \theta\left(x^{(n)}, v^{(n)}\right)\right],$$

$$\boldsymbol{\mathcal{Y}} \equiv \left[\mathcal{Y}\left(x^{(1)}, v^{(1)}\right) \;\; \mathcal{Y}\left(x^{(2)}, v^{(2)}\right) \;\; \dots \;\; \mathcal{Y}\left(x^{(n)}, v^{(n)}\right)\right].$$

Using the method of least squares, the solution to the system of linear equations (A.1) is given by

$$\bar{V}_k = (\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\boldsymbol{y}^T \tag{A.2}$$

Note that for the inverse of matrix $(\boldsymbol{\theta}\boldsymbol{\theta}^T)$, which is an $r \times r$ matrix, to exist, certain conditions need to hold. These conditions are a) the elements of the basis functions $\theta(.,.)$ need to be linearly independent, and b) n needs to be greater than or equal to the number of the elements of the basis functions, i.e., $n \geq r$.

APPENDIX B

Proof of Theorem 1: The proof of convergence for $\bar{V}_k^i$ is detailed here and that of V_k^i is skipped, since it is straight forward by following the line of proof given for $\bar{V}_k^i$. The iteration performed on $\bar{V}_k^i$, given in (36) and repeated here, is a successive approximation to find a fixed point of a function

$$\bar{V}_k^{i+1^T}\theta(x_k, v) = -\bar{R}^{-1}\bar{g}(x_k)^T\left(\nabla\rho\left(x_{k+1}^i\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^i, v\right)^T \bar{W}_{k+1}\right) - V_k^T\sigma(x_k),$$

In other words, there exists a function $\mathcal{F}$: $\mathbb{R}^{r\times m} \to \mathbb{R}^{r\times m}$ such that (36) is of the form

$$\bar{V}_k^{i+1} = \mathcal{F}(\bar{V}_k^i). \tag{B.1}$$

Therefore, the problem simplifies to whether (B.1) is a contraction mapping [21]. Since $\mathbb{R}^{r\times m}$ with the 2-norm denoted by $\|.\|$ is a Banach space, iterations given by (B.1) converges to some $\bar{V}_k = \mathcal{F}(\bar{V}_k)$ if there is a $0 \leq c < 1$ such that for every U_1 and U_2 in $\mathbb{R}^{r\times m}$, the following inequality holds [21]

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \leq c\|U_1 - U_2\|. \tag{B.2}$$

Function $\mathcal{F}(.)$ can be formed by converting (36) to a least squares form discussed in Appendix A. Rewriting Eq. (A.2), given in Appendix A with the notations defined therein, leads to

$$\mathcal{F}\left(V_k^i\right) \equiv$$

$$\times (\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\begin{bmatrix} \left(-\bar{R}^{-1}\bar{g}\left(x_k^{(1)}\right)^T\left(\nabla\rho\left(x_{k+1}^{\bar{V}_k^i,(1)}\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^{\bar{V}_k^i,(1)}, v^{(1)}\right)^T \bar{W}_{k+1}\right) - V_k^T\sigma\left(x_k^{(1)}\right)\right)^T \\ \left(-\bar{R}^{-1}\bar{g}\left(x_k^{(2)}\right)^T\left(\nabla\rho\left(x_{k+1}^{\bar{V}_k^i,(2)}\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^{\bar{V}_k^i,(2)}, v^{(2)}\right)^T \bar{W}_{k+1}\right) - V_k^T\sigma\left(x_k^{(2)}\right)\right)^T \\ \vdots \\ \left(-\bar{R}^{-1}\bar{g}\left(x_k^{(n)}\right)^T\left(\nabla\rho\left(x_{k+1}^{\bar{V}_k^i,(n)}\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^{\bar{V}_k^i,(n)}, v^{(n)}\right)^T \bar{W}_{k+1}\right) - V_k^T\sigma\left(x_k^{(n)}\right)\right)^T \end{bmatrix}, \tag{B.3}$$

where $x_{k+1}^{\bar{V}_k^i,(j)} \equiv \bar{f}\left(x_k^{(j)}\right) + \bar{g}\left(x_k^{(j)}\right)\left(V_k^T\sigma\left(x_k^{(j)}\right) + \bar{V}_k^{iT}\theta\left(x_k^{(j)}, v^{(j)}\right)\right)$. One has

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \leq \sqrt{n}\|(\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\|$$

$$\times \|\bar{R}^{-1}\bar{g}\left(x_k^{(\ell)}\right)^T\left(\nabla\rho\left(x_{k+1}^{U_1,(\ell)}\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^{U_1,(\ell)}, v^{(\ell)}\right)^T \bar{W}_{k+1}\right) -$$

$$\bar{R}^{-1}\bar{g}\left(x_k^{(\ell)}\right)^T\left(\nabla\rho\left(x_{k+1}^{U_2,(\ell)}\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^{U_2,(\ell)}, v^{(\ell)}\right)^T \bar{W}_{k+1}\right)\|, \tag{B.4}$$

where $\ell \in \{1,2,\dots,n\}$ is such that

$$\ell = argmax_{i\in\{1,2,\dots,n\}}\|\bar{R}^{-1}\bar{g}\left(x_k^{(i)}\right)^T\left(\nabla\rho\left(x_{k+1}^{U_1,(i)}\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^{U_1,(i)}, v^{(i)}\right)^T \bar{W}_{k+1}\right)$$

$$-\bar{R}^{-1}\bar{g}\left(x_k^{(i)}\right)^T\left(\nabla\rho\left(x_{k+1}^{U_2,(i)}\right)^T W_{k+1} + \nabla\eta\left(x_{k+1}^{U_2,(i)}, v^{(i)}\right)^T \bar{W}_{k+1}\right)\|.$$

In inequality (B.4), the following norm inequality is used

$$\left\|\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}\right\| \leq \sqrt{n}\|y_\ell\|, \tag{B.5}$$

where y_is are real-valued row-vectors and $\ell = \underset{i\in\{1,2,\dots,n\}}{argmax}\|y_i\|$. Inequality (B.4) leads to

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \leq \sqrt{n}\|(\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\|\left\|\bar{R}^{-1}\bar{g}\left(x_k^{(\ell)}\right)^T\right\|$$

$$\|W_{k+1}^T\left(\nabla\rho\left(x_{k+1}^{U_1,(\ell)}\right) - \nabla\rho\left(x_{k+1}^{U_2,(\ell)}\right)\right) + \bar{W}_{k+1}^T\left(\nabla\eta\left(x_{k+1}^{U_1,(\ell)}, v^{(\ell)}\right) - \nabla\eta\left(x_{k+1}^{U_2,(\ell)}, v^{(\ell)}\right)\right)\|. \tag{B.6}$$

The smoothness of basis functions $\rho(.)$ and $\eta(.,.)$ leads to the smoothness of $\nabla\rho(.)$ and $\nabla\eta(.,.)$. Therefore, functions $\nabla\rho(.)$ and $\nabla\eta(.,v)$, $\forall v \in \bar{\Omega}$, are Lipschitz

continuous on compact set Ω with respect to x (uniformly in v) [25]. In other words, for every x_1 and x_2 in Ω and every $v \in \bar{\Omega}$, one has $\|\nabla\rho(x_1) - \nabla\rho(x_2)\| \leq k_\rho\|x_1 - x_2\|$ and $\|\nabla\eta(x_1, v) - \nabla\eta(x_2, v)\| \leq k_\eta\|x_1 - x_2\|$ for some non-negative constants k_ρ and k_η independent of v. Using the Lipschitz continuity of $\nabla\rho(.)$ and $\nabla\eta(., v)$, inequality (B.6) provides

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \leq \sqrt{n}\|(\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\| \left\|\bar{R}^{-1}\bar{g}\left(x_k^{(\ell)}\right)^T\right\|$$

$$\times \left(k_\rho\|W_{k+1}\| \left\|\left(x_{k+1}^{U_1,(\ell)} - x_{k+1}^{U_2,(\ell)}\right)\right\| + k_\eta\|\bar{W}_{k+1}\| \left\|\left(x_{k+1}^{U_1,(\ell)} - x_{k+1}^{U_2,(\ell)}\right)\right\|\right), \tag{B.7}$$

and substituting $x_{k+1}^{U_1,(\ell)}$ and $x_{k+1}^{U_2,(\ell)}$ by their values leads to

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \leq \sqrt{n}\|(\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\| \left\|\bar{R}^{-1}\bar{g}\left(x_k^{(\ell)}\right)^T\right\|$$

$$\times \left(k_\rho\|W_{k+1}\| + k_\eta\|\bar{W}_{k+1}\|\right) \left\|\bar{g}\left(x_k^{(\ell)}\right)\right\| \|\theta\left(x_k^{(\ell)}, v^{(\ell)}\right)\|\|(U_1 - U_2)\| \tag{B.8}$$

Define

$$c \equiv \sqrt{n}\|(\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\| \left\|\bar{R}^{-1}\bar{g}\left(x_k^{(\ell)}\right)^T\right\|$$

$$\times \left(k_\rho\|W_{k+1}\| + k_\eta\|\bar{W}_{k+1}\|\right) \left\|\bar{g}\left(x_k^{(\ell)}\right)\right\| \|\theta\left(x_k^{(\ell)}, v^{(\ell)}\right)\|, \tag{B.9}$$

which in terms of the continuous-time problem parameters, one has

$$c = \sqrt{n}\|(\boldsymbol{\theta}\boldsymbol{\theta}^T)^{-1}\boldsymbol{\theta}\| \left\|R^{-1}g\left(x_k^{(\ell)}\right)^T\right\|$$

$$\times \left(k_\rho\|W_{k+1}\| + k_\eta\|\bar{W}_{k+1}\|\right) \left\|\Delta t g\left(x_k^{(\ell)}\right)\right\| \left\|\theta\left(x_k^{(\ell)}, v^{(\ell)}\right)\right\|. \tag{B.10}$$

The defined c simplifies inequality (B.8) to (B.2). One can always select sampling time Δt, in the discretization of the continuous dynamics (1), small enough such that condition $0 \leq c < 1$ is satisfied. The reason is the fact that a smaller Δt directly results in a smaller $\|\Delta t g\left(x_k^{(\ell)}\right)\|$. Note that continuity of $g(.)$ and $\theta(.,.)$ in their domains (which follows from their smoothness) results in being bounded in the compact sets Ω and $\bar{\Omega}$ [26], hence, the $x_k^{(\ell)}$ dependent terms in (B.10) are upper bounded. Moreover, terms $\|W_{k+1}\|$ and $\|\bar{W}_{k+1}\|$ are already calculated in the previous time-step, therefore they are finite. Note that as

$\Delta t \to 0$, these two weights stay finite, because, they are the weights of NNs which approximate the summation given in (19), whose limit as $\Delta t \to 0$ is the integral given in (2). Since the horizon is finite, using finite controls, the cost-to-go will always stay finite, except for systems with finite scape time which are ruled out of the investigation. This completes the proof of convergence of $\bar{V}_k^i$ to $\bar{V}_k$ for $0 \leq k < N-1$ using any sampling time smaller than the one for which $0 \leq c < 1$. ■

REFERENCES

[1] Bryson, A. E., and Ho, Y. C., *Applied Optimal Control*, Taylor & Francis, New York, 1975, pp. 148–164.

[2] Kirk, D. E., *Optimal Control Theory: An Introduction*, Dover Publications, 2004 pp. 54-94,329-330.

[3] Liang, J., *Optimal Magnetic Attitude Control of Small Spacecraft*, PhD Thesis, Utah State University, Logan, 2005.

[4] Cimen, T., and Banks, S. P. "Global Optimal Feedback Control for General Nonlinear Systems with Nonquadratic Performance Criteria," *Systems & Control Letters*, Vol. 53, 2004, pp. 327 – 346.

[5] Heydari, A. and Balakrishnan, S. N., "Approximate closed-form solutions to finite-horizon optimal control of nonlinear systems," *American Control Conference*, Montreal, Canada, June 2012, pp. 2657-2662.

[6] Vadali, S. R., and Sharma, R., "Optimal Finite-time Feedback Controllers for Nonlinear Systems with Terminal Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 4, 2006, pp. 921-928.

[7] Sharma, R., Vadali, S. R., and Hurtado, J. E., "Optimal Nonlinear Feedback Control Design Using a Waypoint Method," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 3, 2011, pp. 698-705.

[8] Park, C., Guibout, V., and Scheeres, D. J., "Solving Optimal Continuous Thrust Rendezvous Problems with Generating Functions," *Journal of Guidance, Control, & Dynamics*, Vol. 29, No. 2, 2006, pp. 321-331.

[9] Bando, M., and Yamakawa, H., "New Lambert Algorithm Using the Hamilton–Jacobi–Bellman Equation," *Journal of Guidance, Control, & Dynamics*, Vol. 33, No. 3, 2010, pp. 1000-1008.

[10] Beard, R., *Improving the Closed-Loop Performance of Nonlinear Systems*, Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, USA, 1995.

[11] Han, D. and Balakrishnan, S. N., "State-constrained agile missile control with adaptive-critic-based neural networks," *IEEE Trans. Control Systems Technology*, Vol. 10, No. 4, 2002, pp. 481-489.

[12] Cheng, T., Lewis, F. L., and Abu-Khalaf, M., "A neural network solution for fixed-final time optimal control of nonlinear systems," *Automatica*, Vol. 43, 2007, pp. 482-490.

[13] Heydari, A., and Balakrishnan, S. N., "Finite-Horizon Control-Constrained Nonlinear Optimal Control Using Single Network Adaptive Critics," *IEEE Trans. Neural Netw*. *Learning Syst.*, Vol. 24, No. 1, 2013, pp. 145-157.

[14] Heydari, A., and Balakrishnan, S. N., "Fixed-final-time Optimal Tracking Control of Input-affine Nonlinear Systems," submitted to *Neurocomputing*.

[15] Wang, F., Jin, N., Liu, D., and Wei, Q., "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw*., vol. 22 (1), pp. 24-36, 2011.

[16] Song, R., and Zhang, H., "The finite-horizon optimal control for a class of time-delay affine nonlinear system," *Neural Comput & Applic*, DOI 10.1007/s00521-011-0706-3, 2011.

[17] Wei, Q., and Liu, D., "Finite horizon optimal control of discrete-time nonlinear systems with unfixed initial state using adaptive dynamic programming," *J Control Theory Appl*, vol. 9 (3), pp. 381–390, 2011.

[18] Wei, Q., and Liu, D., "An iterative ϵ-optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state," *Neural Networks*, vol. 32, pp. 236-244, 2012.

[19] Balakrishnan, S.N., and Biega, V., "Adaptive-critic based neural networks for aircraft optimal control", *Journal of Guidance, Control & Dynamics*, Vol. 19, No. 4, 1996, pp. 893-898.

[20] Prokhorov, D. V., and Wunsch, D. C., "Adaptive critic designs," *IEEE Trans. Neural Networks*, Vol. 8, No. 5, 1997, pp. 997-1007.

[21] Khalil, H., *Nonlinear Systems*, 3rd ed., Prentice Hall, New Jersey, 2002, pp. 653-656.

[22] Bryson, A. E., *Dynamic Optimization*, Addison Wesley Longman, California, 1999, pp. 243-256.

[23] Werbos, P. J., "Approximate dynamic programming for real-time control and neural modeling". In White D. A., & Sofge D. A. (Eds.), *Handbook of Intelligent Control*, Multiscience Press, 1992.

[24] Heydari, A., and Balakrishnan, S. N., "Optimal Online Path Planning for Approach and Landing Guidance," in Proc. *AIAA Atmospheric Flight Mechanics Conference*, Portland, OR, 2011, AIAA-2011-6641.

[25] Marsden J. E., Ratiu T., and Abraham R., *Manifolds, Tensor Analysis, and Applications*, 3rd ed. Springer-Verlag Publishing Co., New York, 2001, p. 74.

[26] Trench, W. F., *Introduction to Real Analysis*, available online at http://ramanujan.math.trinity.edu/ wtrench/texts/trench_real_analysis.pdf, 2012, p. 313.

3. GLOBAL OPTIMALITY OF APPROXIMATE DYNAMIC PROGRAMMING AND ITS USE IN NON-CONVEX FUNCTION MINIMIZATION

Ali Heydari and S. N. Balakrishnan

ABSTRACT

This study investigates the global optimality of approximate dynamic programming (ADP) based solutions using neural networks for optimal control problems with fixed final time. Issues including whether or not the cost function terms and the system dynamics need to be convex functions versus their respective inputs are discussed and sufficient conditions for global optimality of the result are derived. Next, a new idea is presented to use ADP with neural networks for optimization of non-convex smooth functions. It is shown that any initial guess leads to direct movement toward the proximity of the global optimum of the function. This behavior is in contrast with gradient based optimization methods in which the movement is guided by the shape of the local level curves. Illustrative examples are provided with single and multi-variable functions that demonstrate the potential of the proposed method.

I. INTRODUCTION

In the last two decades, approximate dynamic programming (ADP) has been shown to have great promise in solving optimal control problems with neural networks (NN) [1-15]. In the ADP framework, the solutions are obtained using a two-network synthesis called adaptive critics (ACs) [2-4]. In the heuristic dynamic programming (HDP) approach with ACs, one network, called the 'critic' network, maps the input states to output the cost-to-go and another network, called the 'action' network, outputs the control with states of the system as its inputs [4,5]. In the dual heuristic programming (DHP) formulation, the action network remains the same as in the HDP, however, the critic network outputs the costates with the current states as inputs [2,6,7]. The computationally effective single network adaptive critics (SNAC) architecture consists of one network only. In [8], the action network was eliminated in a DHP type formulation with control being calculated from the costate values. Similarly, the J-SNAC [9] eliminates the need for the action network in an HDP scheme. Note that the developments in [1-9] are for *infinite-horizon* problems.

The use of ADP for solving *finite-horizon* optimal control problems was considered in [10-15]. Authors of [10] developed a time-varying neurocontroller for solving a scalar problem with state constraints. In [11] a single NN with a single set of weights was proposed which takes the time-to-go as an *input* along with the states and generates the fixed-final-time optimal control for discrete-time nonlinear multi-variable systems. An HDP based scheme for optimal control problems with *soft* or *hard* terminal constraints was presented in [12]. Finite-horizon problems with *unspecified* terminal times were considered in [13-15].

Despite much published literature on adaptive critics, there still exists an open question about the nature of optimality of the adaptive critic based results. Are they locally or globally optimal? A major contribution of this study is in proving that the AC based solutions are globally optimal subject to the assumed basis functions. To help with the development of the proof, the ADP based algorithm for solving fixed-final-time problems developed in [11,12] is revisited first. After describing the algorithm, a novel analysis is presented on global optimality of the result. It is shown that selecting any cost function with quadratic control penalizing term, if the sampling time used for discretization of the original continuous-time system is small enough, the resulting cost-to-go function will be convex versus the control at the current time and hence, the first order necessary optimality condition [16] will lead to the *global* optimal control. The second major contribution of this paper is in showing that the ADP can be used for functional optimization, specifically, optimization of non-convex functions. Finally, through numerical simulations, two examples with varying complexities are presented and the performance of the proposed method is investigated. It is shown that despite the gradient based methods, selecting any initial guess on the minimum and updating the guess using the control resulting from the actor, the states will move *directly* toward the global minimum, passing any possible local minimum in the path.

The rest of this paper is organized as follows: The problem formulation is given in section II. The ADP-based solution is discussed in section III. The supporting theorems and analyses are presented in section IV. The use of the method in static function optimization is discussed in section V, and the conclusions are given in section VI.

II. PROBLEM FORMULATION

Let the control-affine dynamics of the system be given by

$$\dot{x}(t) = f\big(x(t)\big) + g\big(x(t)\big)u(t) \tag{1}$$

where $f: \mathbb{R}^n \to \mathbb{R}^n$ and $g: \mathbb{R}^n \to \mathbb{R}^{n\times m}$. Positive integers n and m, respectively, denote the dimension of the state and the control vectors. The selected cost function, J is fairly general but quadratic in control:

$$J = \psi\left(x\big(t_f\big)\right) + \int_{t_0}^{t_f}\left(Q\big(x(t)\big) + u(t)^T R u(t)\right)dt, \tag{2}$$

where positive semi-definite smooth functions $Q: \mathbb{R}^n \to \mathbb{R}$ and $\psi: \mathbb{R}^n \to \mathbb{R}$ penalize the states and positive definite matrix R penalizes the control effort. The initial and final time are denoted with t_0 and t_f, respectively. Discretizing the time horizon to N time steps using sampling time Δt leads to the discrete-time dynamics and cost function as

$$x_{k+1} = \bar{f}(x_k) + \bar{g}(x_k)u_k,\ k \in K, \tag{3}$$

$$J = \psi(x_N) + \textstyle\sum_{k=0}^{N-1}(\bar{Q}(x_k) + u_k^T \bar{R} u_k), \tag{4}$$

where $K \equiv \{0, 1, 2, \dots, N-1\}$, $N \equiv \big(t_f - t_0\big)/\Delta t$, $x_k \equiv x(k\Delta t + t_0)$, and $\bar{f}(x) \equiv x + \Delta t f(x)$, $\bar{g}(x) \equiv \Delta t g(x)$, $\bar{Q}(x) \equiv \Delta t Q(x)$, and $\bar{R} \equiv \Delta t R$. The problem is defined as finding a control history $u(t) \in \mathbb{R}^m$, $t \in [t_0, t_f)$, such that cost function (4) is minimized subject to the dynamics given in (3).

Assumption 1: The dynamics of the system do not have finite escape times. Also, the functions $f(x)$ and $g(x)$ are smooth in x.

Remark 1: In order to use ADP, the continuous-time problem is discretized. Moreover, the assumption that discrete-time system (3) is obtained through discretizing a continuous-time problem is utilized in convergence analysis of the algorithm.

III. APPROXIMATE DYNAMIC PROGRAMMING BASED SOLUTION

In this section, an ADP scheme called AC is used for solving the optimal control problem in terms of the network weights and selected basis functions. The method is adopted from [11,12]. In this scheme, two networks called critic and actor are trained to approximate the optimal cost-to-go and the optimal control, respectively. It should be

noted that the optimal cost-to-go at each instant is a function of the current state, x_k, and the current time, k, therefore, it is denoted with $J_k^*(x_k)$, i.e.,

$$J_k^*(x_k) = \psi(x_N) + \sum_{\hbar=k}^{N-1}\left(\bar{Q}(x_\hbar) + u_\hbar^T \bar{R} u_\hbar\right). \tag{5}$$

The solution to the problem is given by the Bellman equation [17] as

$$J_N^*(x_N) = \psi(x_N), \tag{6}$$

$$J_k^*(x_k) = \bar{Q}(x_k) + u_k^*(x_k)^T \bar{R} u_k^*(x_k) + J_{k+1}^*(x_{k+1}^*),\ k \in K \tag{7}$$

$$u_k^*(x_k) = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T\ \nabla J_{k+1}^*(x_{k+1}^*),\ k \in K. \tag{8}$$

where $x_{k+1}^* = \bar{f}(x_k) + \bar{g}(x_k)u_k^*(x_k)$ and gradient $\nabla J_{k+1}^*(x_{k+1}^*) \equiv \partial J_{k+1}^*(x_{k+1})/\partial x_{k+1}$ is evaluate at x_{k+1}^*. Note that $\nabla J_{k+1}^*(x_{k+1}^*)$ is a column vector.

An iterative learning scheme can be derived from Bellman equation for learning the solution to the fixed-final-time problem once Eq. (8) is replaced with [12]

$$u_k^{i+1}(x_k) = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u_k^i(x_k)),\ k \in K. \tag{9}$$

Superscript i denotes the index of iteration which starts with an initial guess on $u_k^0(x_k)$, $k \in K$. The converged value of $u_k^i(x_k)$ in (9) is denoted with $u_k^*(x_k)$ and used in (7). Note that in a dual network AC scheme for finite horizon optimal control, 'iterations' takes place in the training of the actor, as seen in (9). Once state-control relationship is learned, the optimal cost-to-go is obtained in a 'one-shot' process as given in (7).

Denoting the approximated optimal cost-to-go and the approximated optimal control with $J_k(x_k)$ and $u_k(x_k)$, respectively, and selecting linear in the weights NNs, the expressions for the actor (control) and the critic (cost), can be written as

$$u_k(x) = V_k^T \sigma(x),\ k \in K \tag{10}$$

$$J_k(x) = W_k^T \phi(x),\ k \in K \cup \{N\} \tag{11}$$

where $V_k \in \mathbb{R}^{p\times m}$ and $W_k \in \mathbb{R}^q$ are the unknown weights of the actor and the critic networks at time step k, respectively, and the selected smooth basis functions are given by $\sigma: \mathbb{R}^n \to \mathbb{R}^p$ and $\phi: \mathbb{R}^n \to \mathbb{R}^q$ where p and q denote the number of neurons. The idea is using Eqs. (6), (7), and (9) to find the NN weights. Note that, once $J_{k+1}^*(.)$ is known one can use (9) to find $u_k^*(.)$ and then (7) gives $J_k^*(.)$. Therefore, starting with (6) to find

$J_N^*(.)$, all the unknowns can be calculated in a backward in time fashion, i.e., from $k = N-1$ to $k = 0$. The learning process for calculating weights V_k and W_k, $\forall k$, is detailed through Algorithm 1. Note that $\nabla\phi(x) \equiv \partial\phi(x)/\partial x$ is a column vector.

Algorithm 1

Step 1: Randomly select n state vectors $x_N^{[j]} \in \Omega, \forall j \in \mathcal{J} \equiv \{1,2,..,n\}$, where n is a selected large positive integer, and Ω denotes a compact subset of $\mathbb{R}^n$ representing the domain of interest.

Step 2: Find W_N such that $W_N^T\phi\left(x_N^{[j]}\right) \cong \psi(x_N^{[j]}), \forall j \in \mathcal{J}$.

Step 3: For $k = N-1$ to $k = 0$ repeat

{

Step 4: Set $i = 0$ and select a guess on $u_k^{0,[j]} \in \mathbb{R}^m, \forall j \in \mathcal{J}$.

Step 5: Randomly select n state vectors $x_k^{[j]} \in \Omega, \forall j \in \mathcal{J}$.

Step 6: Set $x_{k+1}^{i,[j]} = \bar{f}\left(x_k^{[j]}\right) + \bar{g}\left(x_k^{[j]}\right)u_k^{i,[j]}, \forall j \in \mathcal{J}$.

Step 7: Set $u_k^{i+1,[j]} = -\frac{1}{2}\bar{R}^{-1}\bar{g}\left(x_k^{[j]}\right)^T\nabla\phi\left(x_{k+1}^{i,[j]}\right)^T W_{k+1}, \forall j \in \mathcal{J}$.

Step 8: Set $i = i+1$ and repeat Step 7, until $\left\|u_k^{i+1,[j]} - u_k^{i,[j]}\right\| < \beta, \forall j \in \mathcal{J}$, where $\beta > 0$ is a preset tolerance. Denote the converged value of $u_k^{i,[j]}$ with $u_k^{[j]}, \forall j$.

Step 9: Find V_k such that $V_k^T\sigma\left(x_k^{[j]}\right) \cong u_k^{[j]}, \forall j \in \mathcal{J}$.

Step 10: Find W_k such that

$$W_k^T\phi\left(x_k^{[j]}\right) \cong \bar{Q}\left(x_k^{[j]}\right) + u_k^{[j]^T}\bar{R}u_k^{[j]} + W_{k+1}^T\phi\left(\bar{f}\left(x_k^{[j]}\right) + \bar{g}\left(x_k^{[j]}\right)u_k^{[j]}\right), \forall j \in \mathcal{J}.$$

}

In Steps 2, 9, and 10 of Algorithm 1, the method of Least Squares, explained in the Appendix, can be used for finding the unknown weights.

Remark 2: The capability of *uniform approximation* of neural networks [18,19] indicates that once the network is trained for a large enough number of samples distributed evenly throughout the domain of interest, the network is able to approximate the output for any new sample of the domain with a bounded approximation error. This error bound can be

made arbitrarily small once the network is rich enough. For the linear in the weight neural networks selected in this study and the polynomial basis function utilized in the numerical examples, Weierstrass approximation theorem [20] proves a similar uniform approximation capability.

IV. SUPPORTING THEOREMS AND ANALYSES

A. Convergence Analysis

Theorem 1: The iterative relation given by Eq. (9), with any initial guess for $u_k^0 \in \mathbb{R}^m$, $\forall k \in K$, converges, providing the sampling time Δt selected for discretization of continuous-time dynamics (1) is small enough.

Proof: Let the right hand side of (9) be denoted with function $\mathcal{F}$: $\mathbb{R}^m \to \mathbb{R}^m$ where

$$\mathcal{F}(u) = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u) \tag{12}$$

The proof is complete if it is shown that the relation given by the successive approximation

$$u^{i+1} = \mathcal{F}(u^i) \tag{13}$$

is a contraction mapping [21]. Since $\mathbb{R}^m$ with 2-norm denoted with $\|.\|$ is a Banach space, the iterations given by (13) converges to some $u_k = \mathcal{F}(u_k)$ if there is a $0 \le \rho < 1$ such that for every u and v in $\mathbb{R}^m$, the following inequality holds

$$\|\mathcal{F}(u) - \mathcal{F}(v)\| \le \rho\|u - v\|. \tag{14}$$

By Eq. (12) one has

$$\|\mathcal{F}(u) - \mathcal{F}(v)\| \le$$

$$\left\|\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u) - \frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)v)\right\| \tag{15}$$

The optimal cost-to-go function, $J_{k+1}^*(x_{k+1})$, is smooth versus its input, x_{k+1} since functions $\phi(.)$, $Q(.)$, $f(.)$, and $g(.)$ are smooth. By considering Eq. (6), since $\phi(x_N)$ is a smooth function, function $J_N^*(x_N)$, and hence $\nabla J_N^*(x_N)$, are smooth. Smoothness of $\bar{g}(x_{N-1})$ and $\nabla J_N^*(x_N)$ leads to the smoothness of optimal control function $u_{N-1}^*(x_{N-1})$, by (8), which along with the smoothness of $\bar{Q}(x_{N-1})$ lead to a smooth $J_{N-1}^*(x_{N-1})$, by (7). Repeating this argument backward from $N-1$ to $k+1$, it follows

that $J_{k+1}^{*}(x_{k+1})$ is a smooth function. This smoothness leads to the Lipschitz continuity of $\nabla J_{k+1}^{*}(x_{k+1})$ in the domain of interest Ω [22]. In other words, there exists some positive real number ρ_J such that for every x_1 and x_2 in Ω, one has $\|\nabla J_{k+1}^{*}(x_1) - \nabla J_{k+1}^{*}(x_2)\| \leq \rho_J \|x_1 - x_2\|$. Using this characteristic, inequality (15) can be written as

$$\|\mathcal{F}(u) - \mathcal{F}(v)\| \leq \rho_J \|\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T\| \|\bar{g}(x_k)\| \|(u - v)\| \tag{16}$$

By defining

$$\rho \equiv \rho_J \|\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T\| \|\bar{g}(x_k)\|$$

which is equivalent of

$$\rho \equiv \rho_J \|\frac{1}{2}R^{-1}g(x_k)^T\| \|\Delta t g(x_k)\| \tag{17}$$

one can select the sampling time Δt in discretization of the continuous-time dynamics (1) small enough such that the condition $0 \leq \rho < 1$ is satisfied. Note that the continuity of $g(.)$ in its domain result in being bounded in compact set Ω [23], hence, the state-dependent terms in (17) are upper bounded. This completes the proof of existence of a fixed point, denoted with u_k, and the convergence of u_k^i to u_k, as $i \to \infty$, $\forall u_k^0 \in \mathbb{R}^m$ and $\forall k \in K$ using successive approximation given by (9). ■

In Theorem 1 the role of the sampling time in discretization of a continuous-time system is emphasized. It is worthwhile to discuss this issue in detail. Substituting (11) in optimal control equation (8) leads to

$$u_k^{[j]} = -\frac{1}{2}R^{-1}g\left(x_k^{[j]}\right)^T \nabla\phi\left(f\left(x_k^{[j]}\right) + g\left(x_k^{[j]}\right)u_k^{[j]}\right)^T W_{k+1}, \forall j \in \mathcal{J}, \tag{18}$$

which is the same as the iterative equation given in Step 6 of Algorithm 1 except that $u_k^{i,[j]}$ and $u_k^{i+1,[j]}$ on both sides are replaced with the converged value, i.e., $u_k^{[j]}$. Optimal control $u_k^{[j]}$, $\forall k \in K$ and $\forall j \in \mathcal{J}$, can be calculated by solving the set of m nonlinear equations given by (18) for the m unknown elements of $u_k^{[j]}$, without using the iteration given in Step 6 of Algorithm 1. However, there is no analytical solution to the set of nonlinear equations (18) in general. Therefore, one needs to resort to numerical methods for solving the set of equations. Theorem 1 proves that for any given smooth dynamics

and smooth basis functions, if the sampling time is small enough, the iterations given in Step 6 of Algorithm 1 converge to the solution to the nonlinear equation (18). However, if the sampling time is fixed, certain conditions on the dynamics or the cost function terms need to hold in order for the iterations to converge. These conditions can be easily derived from the proof of Theorem 1, i.e., from Eq. (17).

In the solution to linear problems with quadratic cost function terms, a similar issue is observed. To observe this, one may consider optimal control equation (8). The cost-to-go function, for the linear problem, is assumed to be of form $J_k = \frac{1}{2}x_k^T P_k x_k$ for some $P_k \in \mathbb{R}^{n\times n}$. Considering this assumption, optimal control equation (8) reads

$$u_k = -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}x_{k+1} = -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}(\bar{A}x_k + \bar{B}u_k), \tag{19}$$

where the continuous problem given by $\dot{x}(t) = Ax(t) + Bu(t)$ and $J = x\big(t_f\big)^T Sx\big(t_f\big) + \int_{t_0}^{t_f}\big(x(t)^T Qx(t) + u(t)^T Ru(t)\big)\,dt$ is discretized to $x_{k+1} = \bar{A}x_k + \bar{B}u_k$ and $J = x_N^T Sx_N + \sum_{k=0}^{N-1}(x_k^T\bar{Q}x_k + u_k^T\bar{R}u_k)$. Similar to (18), unknown u_k exists in both sides of equation (19). However, equation (19) is linear and the analytical solution can be calculated as

$$u_k = -(2\bar{R} + \bar{B}^T P_{k+1}\bar{B})^{-1}\bar{B}^T P_{k+1}\bar{A}x_k. \tag{20}$$

If solution (20) was not available, one could use the following iterations, starting with any initial guess u_k^0, to find u_k

$$u_k^{i+1} = -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}(\bar{A}x_k + \bar{B}u_k^i)\ . \tag{21}$$

Following the idea presented in proof of Theorem 1, it is straightforward to show that (21) is a contraction mapping if the sampling time used for discretization of the original continuous-time linear problem is small enough. Hence u_k^i converges to the solution of (19). Another way of investigating the effect of the sampling time in the convergence of Eq. (21) is considering the evolution of u_k^i during the *iterations* as the evolution of the state vector of a discrete-time system versus *time*. In other words, one can look at Eq. (21) as a discrete-time system with the state vector at 'time' i being denotes with u_k^i and the constant control to the system being x_k. Then Eq. (21) can be written as

$$u_k^{i+1} = \mathcal{A}u_k^i + \mathcal{B}x_k \tag{22}$$

where the superscripts denote the 'time' index and

$$\mathcal{A} \equiv -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}\bar{B} = -\frac{1}{2}\Delta t R^{-1}B^T P_{k+1}B \tag{23}$$

and

$$\mathcal{B} \equiv -\frac{1}{2}\bar{R}^{-1}\bar{B}^T P_{k+1}\bar{A}.$$

Selecting any bounded u_k^0, the system given in (22) is stable for a bounded x_k and converges to a steady state u_k as $i \to \infty$, providing the eigenvalues of $\mathcal{A}$ are inside the unit circle around the origin [24]. Considering (23), it is straight forward to show that for any given R, B, A, Q, and S, (where the last three matrices affect the equation through P_{k+1}), selecting small enough sampling time Δt, the eigenvalues of $\mathcal{A}$ can be made arbitrarily close to the origin. Therefore, utilizing small enough sampling time, the eigenvalues of $\mathcal{A}$ can be brought into the unit circle and hence, the system can be made stable. This stability leads to the convergence of iterations given by (21).

B. Global Optimality Analysis

The objective is finding the 'global' optimal control, not a local optimum. Let the cost-to-go given the state x_k, time k, and the control u_k at the current time and utilizing the optimal control for $k+1$ to $N-1$ be denoted with $J_k(x_k, u_k)$. In other words,

$$J_k(x_k, u_k) = \bar{Q}(x_k) + u_k^T \bar{R} u_k + J_{k+1}^*\big(\bar{f}(x_k) + \bar{g}(x_k)u_k\big). \tag{24}$$

This cost-to-go should be differentiated from the 'optimal' cost-to-go, which is only a function of time and current state, as given in Eq. (5). In other words, $J_k^*(x_k) = \min_{u_k \in \mathbb{R}^m}\big(J_k(x_k, u_k)\big)$. The global optimal control is given by

$$\begin{aligned} u_k^* &= \operatorname{argmin}_{u_k \in \mathbb{R}^m}\big(J_k(x_k, u_k)\big) \\ &= \operatorname{argmin}_{u_k \in \mathbb{R}^m}\Big(\bar{Q}(x_k) + u_k^T \bar{R} u_k + J_{k+1}^*\big(\bar{f}(x_k) + \bar{g}(x_k)u_k\big)\Big) \\ &= \operatorname{argmin}_{u_k \in \mathbb{R}^m}\Big(u_k^T \bar{R} u_k + J_{k+1}^*\big(\bar{f}(x_k) + \bar{g}(x_k)u_k\big)\Big) \end{aligned} \tag{25}$$

As shown in the proof of Theorem 1, the optimal cost-to-go function $J_{k+1}^*(x_{k+1})$ is smooth versus x_{k+1}, therefore, the smoothness of $J_k(x_k, u_k)$ with respect to u_k follows from (24). If function $J_k(x_k, u_k)$ is 'convex' in u_k, then its global optimum could be

found by comparing its value at its sole critical point as well as at the boundaries. Smoothness of the function and its unboundedness at the boundaries of $\mathbb{R}^m$ leads to the only candidate being the point at which the gradient vanishes. Therefore, if the function is convex, the global optimum is given by

$$\partial J_k(x_k, u_k)/\partial u_k = 0 \implies u_k^* = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \nabla J_{k+1}^*(x_{k+1}) \tag{26}$$

However, if the function is not convex, Eq. (26) which the Adaptive Critics are based on, can lead to a 'local' minimum. Considering (25), even though the first term subject to minimization is convex in u_k, the second term, i.e., $J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u_k)$, may not be convex in u_k. In other words, satisfying Eq. (26) is a 'necessary' condition for global optimality [16]. This fact leads to the problem that there might be more than one u_k which satisfy (26). For example, the iteration given by (9) may converge to some u_k which satisfies (26), but it is not the global optimum.

Considering the result given in Theorem 1, this concern is addressed. Note that, once it is shown that (9) is a contraction mapping in Theorem 1, the 'uniqueness' of the converged value follows [21]. In other words, if (9) is a contraction mapping, there cannot be any other u_k which satisfy (26), except the one which can be found using the iteration given by (9). This result might be unexpected because of claiming the global minimum without assuming convexity of $J_{k+1}^*(x_{k+1}, .)$ or even assuming convexity of cost function terms $Q(.)$ or $\psi(.)$. It should be noted that even if the cost-function terms are convex in their arguments, because of the arbitrary dynamics of the system, the cost-to-go may not be convex in control. Interested reader are referred to Mangasarian Sufficient Condition Theorem [25] which requires the cost function terms as well as the dynamics functions to be convex in x and u, and also requires that the resulting optimal costates being non-negative for the entire horizon, in order to conclude the 'global' optimality of the solution to the optimal control problem. As for ADP, the authors of [26] claim that the ADP is susceptible is getting stuck in local optimums.

Note that we are interested in minimization of $F(u_k) \equiv u_k^T \bar{R} u_k + J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u_k)$. Theorem 2 proves that, providing a certain condition holds, function $F(u_k)$ is convex. Therefore, Eq. (26) gives the global optimal solution. For this theorem, the following Lemma is required. Before proceeding to the lemma, it should be noted that

function $F: \mathbb{R}^m \to \mathbb{R}$ is *convex* in convex space $\overline{\Omega}$ if $F_{uu}(u) \equiv \partial^2 F(u)/\partial u^2 \geq 0, \forall u \in \overline{\Omega}$. If $F_{uu}(u) > 0, \forall u \in \overline{\Omega}$, then the function is called *strictly convex*. Moreover, if there exists some positive real number c such that $F_{uu}(u) \geq cI, \forall u \in \overline{\Omega}$, then the functions is called *strongly convex*, where the $m \times m$ identity matrix is denoted with I, cf. [27], and notations $F_{uu}(u, a) \geq 0$ and $F_{uu}(u, a) > 0$, respectively, mean that $F_{uu}(u, a)$ is positive semi-definite and positive-definite.

Lemma 1: Define $F(u, a) \equiv af_1(u) + f_2(au)$, for twice differentiable functions $f_1: \mathbb{R}^m \to \mathbb{R}$ and $f_2: \mathbb{R}^m \to \mathbb{R}$, where a is a positive real number. Assume $f_1(.)$ is a strongly convex function in a convex set Ω_1 containing the origin. Then, there exists some positive real number a_0 which for any $a \in (0, a_0)$ function $F(x, a)$ is strictly convex with respect to x in any bounded and convex set $\Omega_2 \subset \Omega_1$.

Proof: Because of the twice differentiability of $f_1(.)$ and $f_2(.)$, function $F(u, a)$ is also twice differentiable with respect to u. If there exists some a_0 which for every $a \in (0, a_0)$ one has $F_{uu}(u, a) > 0$ for every u in a convex domain, then $F(., a)$ is strictly convex in that domain and the proof is complete [27]. One has

$$F_{uu}(u, a) = af_{1_{uu}}(u) + a^2 f_{2_{uu}}(au). \tag{27}$$

The strong convexity of $f_1(.)$ leads to the existence of some positive real number c_1 such that

$$f_{1_{uu}}(u) - c_1 I > 0, \forall u \in \Omega_1. \tag{28}$$

On the other hand, $f_{2_{uu}}(.)$ in any convex and bounded set $\Omega_2 \subset \Omega_1$ is bounded, because of the twice differentiability of $f_2(.)$. Therefore, denoting the smallest eigenvalue with $\lambda(.)$, there exists some positive real number c_2 such that

$$\min_{u \in \Omega_2} \lambda\left(f_{2_{uu}}(u)\right) \geq -c_2. \tag{29}$$

Considering (28) and (29), there always exists some a_0 which for any $a \in (0, a_0)$ one has $F_{uu}(u, a) > 0, \forall u \in \Omega_2$. The reason is, because of the symmetricity of matrices $f_{1_{uu}}$ and $f_{2_{uu}}$ one has [28]

$$\min_{u \in \Omega_2} \lambda(F_{uu}(u, a)) \geq a \min_{u \in \Omega_2} \lambda\left(f_{1_{uu}}(u)\right) + a^2 \min_{u \in \Omega_2} \lambda\left(f_{2_{uu}}(au)\right). \tag{30}$$

Inequality (28) leads to

$$\min_{u\in\Omega_2} \lambda\left(f_{1_{uu}}(u)\right) > c_1. \tag{31}$$

Using (29) and (31) in (30) leads to

$$\min_{u\in\Omega_2} \lambda(F_{uu}(u,a)) > ac_1 - a^2c_2. \tag{32}$$

Therefore, selecting $a_0 = c_1/c_2$, one has $\min_{u\in\Omega_2} \lambda\left(F_{uu}(u,a)\right) > 0$, if $a \in (0, a_0)$. This shows that $F_{uu}(u,a)$ is positive definite, $\forall u \in \Omega_2$, and $\forall a \in (0, a_0)$, hence, proves the lemma. ■

In order to better understand the result given in Lemma 1, the following example is helpful.

Example 1: Let $f_1(u) = u^2$ and

$$f_2(u) = 2.1333u^4 + 0.9333u^3 - 2.1333u^2 - 0.9333u + 1.$$

Fig. 1 depicts the shapes of functions $f_1(.)$ and $f_2(.)$. As seen in the figure, $f_1(.)$ is a convex function, but, $f_2(.)$ is non-convex. Let $F(u,a) \equiv af_1(u) + f_2(au)$. Fig. 2 presents the shape of the resulting $F(u,a)$ for the three values of $a = 10$, 1, and 0.1. This figure shows that as a becomes smaller, function $F(u,a)$ changes toward becoming a convex function. For example, $F(u,10)$ is non-convex while $F(u,0.1)$ is a convex function. This behavior is compatible with the result given in Lemma 1.

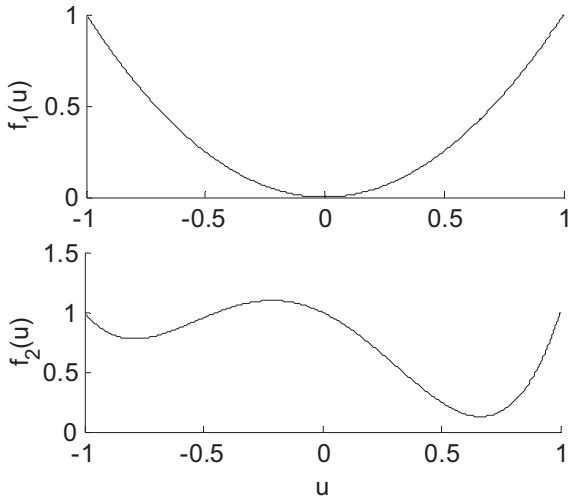

Fig. 1: Functions $f_1(u)$ and $f_2(u)$.

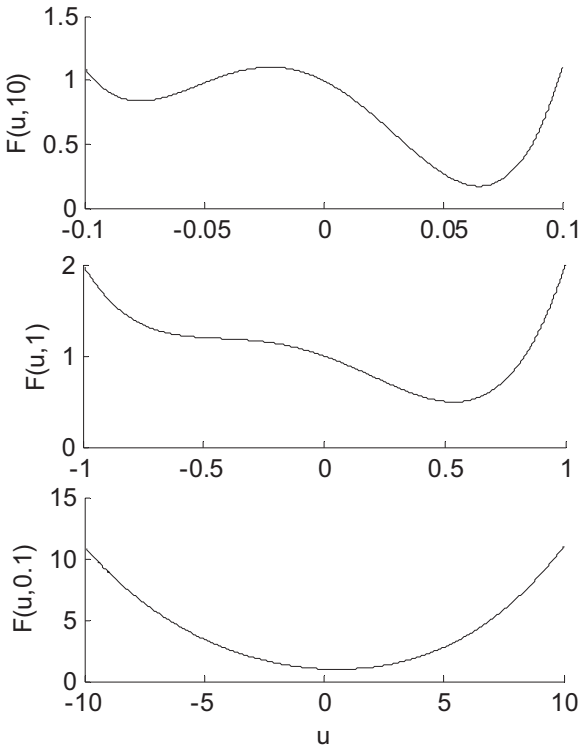

Fig. 2: Function $F(u,a)$ for $a = 10$, 1, and 0.1.

Theorem 2: There exists some sampling time Δt_0 for discretization of the original continuous problem, that selecting any sampling time smaller than that leads to the strict convexity of the cost-to-go function given in (24) with respect to the control in any bounded and convex subset of $\mathbb{R}^m$.

Proof: Rewriting $F(u_k) \equiv \frac{1}{2}u_k^T \bar{R} u_k + J_{k+1}^*(\bar{f}(x_k) + \bar{g}(x_k)u_k)$ in terms of the parameters given in the continuous-time problem (before discretization) one has

$$F(u_k, \Delta t) \equiv \Delta t u_k^T R u_k + J_{k+1}^*(x_k + \Delta t f(x_k) + \Delta t g(x_k)u_k) \tag{33}$$

Defining $f_1(u_k) \equiv u_k^T R u_k$ and $f_2(\Delta t u_k) \equiv J_{k+1}^*(x_k + \Delta t f(x_k) + \Delta t g(x_k)u_k)$, the strong convexity and twice differentiability of $f_1(.)$ follows from $f_{1_{u_k u_k}}(u_k) = R > 0$. The smoothness of $J_{k+1}^*(.)$, derived in the proof of Theorem 1, leads to the twice differentiability of $f_2(.)$. Comparing $F(u_k, \Delta t)$ defined in (33) with $F(x,a)$ defined in Lemma 1, Δt has the same role that a has in Lemma 1. Therefore, there exists some small

sampling time, which for any sampling time smaller than that, the cost-to-go is strictly convex with respect to the control in any selected bounded and convex subset of $\mathbb{R}^m$. ■

While the focus in this study is on optimal control problems with finite-horizon, it can be seen that the result given in Theorem 2 holds for the use of ADP for infinite horizon problems as well. Therefore, the application of the optimality condition leads to global optimal control for such problems, also.

In order to better understand the effect of sampling time in making the cost-to-go a convex function with respect to the control, the following example is helpful. In this example, terminal cost $\psi(.)$ is selected as a non-convex function, and the relation between the global minimum of $\psi(.)$ and that of the optimal control problem is investigated.

Example 2: Let the fixed final time cost function with the horizon equal to Δt be given by

$$J = c\psi\big(x(\Delta t)\big) + \int_0^{\Delta t} u(t)^T u(t)\, dt \tag{34}$$

for some positive real number c. The discretized cost function will be

$$J = c\psi(x_1) + \Delta t u_0^T u_0. \tag{35}$$

Assume the dynamics be given by the single integrator $\dot{x} = u$ discretized to $x_1 = x_0 + \Delta t u_0$. For simplicity assume that x_k and u_k are scalar. Let $c\psi(x)$ be the non-convex function plotted in Fig. 3.

Let the selected initial condition be x_0 shown in the figure. As seen, the global minimum of $\psi(.)$ is at x^*, while the selected x_0 is on a downslope toward a local minimum. Denote the cost-to-go of applying no control and staying at x_0 with J_0. Assume that the optimal solution, with corresponding optimal cost-to-go of J^*, leads to moving toward the left, i.e., toward the global minimum of $\psi(.)$. The cost-to-go difference denoted with ΔJ and defined as $J^* - J_0$, needs to be negative, otherwise J^* is not optimal. However, looking at Fig. 3, it can be seen that unless x_1, i.e., the terminal point, is on the left side of $\bar{x}$, ΔJ will not be negative. The reason is, if x_1 is somewhere between x_0 and $\bar{x}$, then we have spent some control cost and have ended up at a place with more terminal cost compared to staying at x_0 and spending no control cost. But, in order to end up at some point on the left side of $\bar{x}$, we need a control which satisfies

$|u_0| > a/\Delta t$, because of the selected dynamics, where $a \equiv |x_0 - \bar{x}|$. In this case, the control cost will be $\Delta t u_0^2 = (a/\Delta t)^2 \Delta t$. But, the best thing which can be done in terms of having less terminal cost is reaching point x^*, with the reward equal to b, given in the figure. Therefore, in moving toward left one has

$$\Delta J > (a/\Delta t)^2 \Delta t - b = a^2/\Delta t - b. \tag{36}$$

As $\Delta t \to 0$, regardless of how deep the global optimum is, i.e., how large parameter c in the cost function is, which results in large b, there always exists some Δt which for any sampling time smaller than that, one has $\Delta J > 0$. Therefore, the global optimal solution to the cost function is not toward the left of x_0.

The global optimal solution is moving toward the right side of x_0. The reason is, $\partial\psi(x_0)/\partial x_0 = -d < 0$, for some positive d. By definition, if x_1 is selected close to x_0, one has $\partial\psi(x_0)/\partial x_0 \cong -\bar{b}/\bar{a}$. Therefore, $\bar{b} \cong d\bar{a}$. Denoting the cost-to-go of moving toward the right by $\bar{J}^*$, the cost difference $\Delta J \equiv \bar{J}^* - J_0$ has to be negative in order to move toward right, for example to the point denoted with x_1. The control for this move is $\bar{a}/\Delta t$, and the control cost is $(\bar{a}/\Delta t)^2 \Delta t = \bar{a}^2/\Delta t$. Hence, $\Delta J = \bar{a}^2/\Delta t - \bar{b}$. Considering $\bar{b} \cong d\bar{a}$, one has

$$\Delta J = \bar{a}^2/\Delta t - d\bar{a}. \tag{37}$$

Looking at (37), regardless of how small Δt is, there always exists some small $\bar{a}$ for which one has $\Delta J < 0$. Such $\bar{a}$ is given by $\bar{a} < \Delta t d$. Therefore, the global minimum of the *optimal control problem* is toward the right of point x_0, despite the fact that the global minimum of the *terminal cost term* is to the left of x_0.

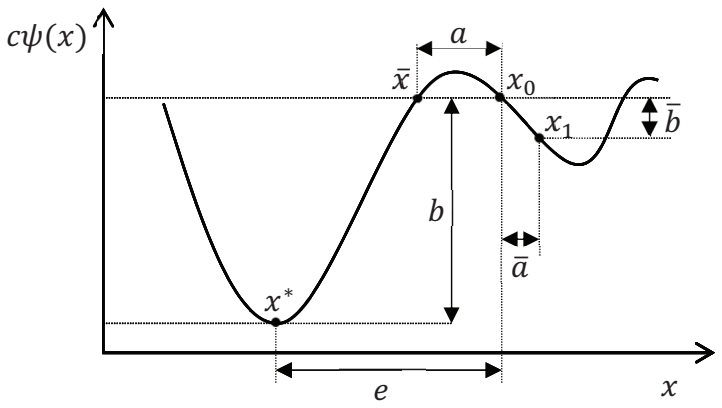

Fig. 3: Non-convex function $c\psi(x)$ versus x.

It should be noted that in here, we limited the horizon to be only one time-step. Assuming a constant sampling time Δt and increasing the number of time-steps N, the results change toward coinciding the global optimum of the optimal control problem with that of the non-convex $\psi(x)$. To see this, let the sampling time be small but, constant and the number of time-steps N be the design parameter. The cost function will be

$$J = \psi(x_N) + \Delta t \sum_{k=0}^{N-1} u_k^T u_k \tag{38}$$

Considering the distance between x_0 and x^*, one policy to get to x^* from x_0 is applying constant control $\bar{u} = -e/(N\Delta t)$ for N time steps to have $x_N = x^*$, based on the selected dynamics. The control cost for such $\bar{u}$ will be

$$\Delta t \sum_{k=0}^{N-1} \bar{u}_k^T \bar{u}_k = e^2/(N\Delta t) \tag{39}$$

Therefore, denoting the cost-to-go of going from x_0 to x^* in N times steps using constant control $\bar{u}$ with J^*, one has

$$\Delta J \equiv J^* - J_0 = e^2/(N\Delta t) - b. \tag{40}$$

Considering Eq. (40), for any selected Δt, there always exists some large enough N for which, ΔJ is negative. As $N \to \infty$, one has $\Delta J \to -b$. Note that going to any other $x_N \neq x^*$, denoting $\psi(x_N) - \psi(x_0)$ with h, regardless of what constant or non-constant control is being selected, the cost-to-go difference (the cost-to-go of going to x_N minus J_0) will be, at the best case scenario, equal to $-h$. Considering the fact that regardless of how small or large $c > 0$ is, one has $b > h$, the 'global' optimal solution to the optimal

control problem will be having $x_N = x^*$, as $N \to \infty$. This result confirms that the global optimum of the optimal control problem coincides with the global optimum of the terminal cost term $\psi(.)$ once the number of time steps goes to infinity.

The important fact which leads to this result, is the control cost being in a quadratic form, therefore, we can always select a larger number of time-steps in order to split the control and end up with less control cost. Note that a control equal to $\bar{u}$ leading to the control cost $\Delta t\bar{u}^2$, once split to N equal parts will have the control cost of $N\Delta t(\bar{u}/N)^2 = \Delta t\bar{u}^2/N$. Therefore, for a fixed Δt, as $N \to \infty$ the control cost vanishes.

Finally, it should be noted that for the NNs trained based on Algorithm 1 to provide the global optimal solution, beside the global optimality of the input-target training pairs, another condition is also required. The condition is that the network training law needs to avoid getting stuck in local minimums, in learning the mapping between the input and the target. One way of fulfilling this requirement is using linear in the weight NNs, as in this study, and using the method of Least Squares (explained in the Appendix) for finding the weight. Note that lease squares problems are convex [27], hence, their solution is the *global* optimum.

V. NON-CONVEX FUNCTION OPTIMIZATION

One of the applications of the global optimality results given in this study is using ADP for finding the global optimum of smooth but possibly non-convex functions. In other words, the 'optimal control' tool can be used for 'convex or non-convex function optimization'. Considering nonlinear programming based optimization methods [17,16], for optimizing function $\psi(x)$, one selects an initial guess, denoted with x_0, and uses the update rule

$$x_{k+1} = x_k + \tau u_k, \tag{41}$$

where $\tau \in \mathbb{R}\backslash\{0\}$ is the update rate. Parameter u_k is calculated based on the gradient/Hessian of the function subject to minimization at point x_k in gradient based methods of optimization. Update rule (41) should be repeated until the minimum is reached. It is well-known that nonlinear programming based methods are susceptible to getting stuck in local minima. Therefore, they are suitable for optimization of smooth convex functions, not for general smooth functions.

Looking at (41) it resembles the discretized version of single integrator dynamics $\dot{x} = u$, which is

$$x_{k+1} = x_k + \Delta t u_k. \tag{42}$$

Selecting cost function

$$J = c\psi\left(x(t_f)\right) + \int_{t_0}^{t_f} u(t)^T R u(t)\, dt, \tag{43}$$

which is discretized to

$$J = c\psi(x_N) + \Delta t \sum_{k=0}^{N-1} u_k^T u_k, \tag{44}$$

minimizing the cost function for large c results in approximately optimizing $\psi(.)$. In other words, solving the optimal control problem defined by cost function (44) subject to dynamics (42) for large $c > 0$ results in the (approximate) global minimum of function $\psi(.)$, given by the state at the final time, i.e., x_N. Therefore, the ADP method described earlier may be used for minimization of non-convex smooth functions.

Remark 3: Even-though it is more intuitive to assume the terminal state penalizing term in the cost function being positive semi-definite in the ADP problems, they are not required to be so for the ADP to provide the optimal solution. Hence, in the development in this section, no assumption on positive semi-definiteness of function $\psi(.)$ is made.

The simplicity of the dynamics given in (42) and the lack of presence of state penalizing term $Q(x)$ in cost function (44) provide an interesting feature for the optimal control problem. Defining the costate vector $\lambda_k \equiv \nabla J_k^*(x_k)$, the costate equation, resulting from taking the derivative of the cost-to-go recursive equation given by (6) and (7) with respect to x, is

$$\lambda_N = \partial\psi(x_N)/\partial x_N, \tag{45}$$

$$\lambda_k = \partial\bar{Q}(x_k)/\partial x_k + A_k^T \lambda_{k+1},\ k \in K, \tag{46}$$

where $A_k \equiv \partial x_{k+1}/\partial x_k$. The optimal control will then be given by

$$u_k^* = -\frac{1}{2}\bar{R}^{-1}\bar{g}(x_k)^T \lambda_{k+1},\ k \in K. \tag{47}$$

Considering dynamics (42) and cost function (44) for the problem at hand, the costate equation simplifies to

$$\lambda_k = \lambda = \partial\psi(x_N)/\partial x_N, \ \forall k \in K \cup \{N\}, \tag{48}$$

in other words, the costate vector will be a constant vector. Therefore, once the optimal costate vector is found, the optimal control also will be constant and given by $u_k^* = u^* = -\lambda$. This results in considerable computational simplicity, because, the approximate global minimum x_N, can then be simply calculated as

$$x_N = x_0 - N\Delta t\lambda. \tag{49}$$

Note that, at the global minimum of a smooth function whose minimum is not at the boundaries of its domain, the gradient has to be zero. Therefore, if the global minimum is given by x_N, it is needed to have $\partial\psi(x_N)/\partial x_N = 0$. Considering (48), this condition leads to $\lambda = 0$, and therefore $u^* = 0$ and $x_N = x_0$, which is of course not the desired solution. Solving the defined optimal control problem will not give the solution of $u^* = 0$, because it obviously does not optimize the selected cost function, unless the selected initial condition x_0 coincides with the global optimum of $\psi(.)$. In general, the optimal control solution gives a non-zero control, which means that $\lambda = \partial\psi(x_N)/\partial x_N \neq 0$. Therefore, this method will not give the exact global optimum of $\psi(.)$, however, it will provide an approximation of the global optimum. The reason is, looking at (49) parameter N is a design parameter. Selecting a fixed and finite initial condition x_0, the method will provide a finite x_N, otherwise the solution is not the optimal solution to the control problem. However, N can be selected arbitrarily large. Therefore, the following property holds

$$\lim_{N\to\infty} \lambda = 0. \tag{50}$$

In other words, as the number of time steps, denoted with N grows for a given sampling time Δt, the optimal costate vector λ has to converge to zero, in order to cancel the effect of the growth of N and to lead to a finite x_N. The global optimality of the ADP result along with the property given in (50) show that if the number of time steps goes to infinity, x_N will converge to a point which has two features: 1) it globally optimizes the selected cost function, 2) function $\psi(.)$ at this point has the slope of zero, that is $\partial\psi(x)/\partial x = 0$. Selecting large $c > 0$, the only point which has these two features is the global optimum of $\psi(.)$.

Once the optimal control problem is solved, looking at the value of λ provides us with an insight into 'how optimal the result will be'. As discussed above, as λ goes to zero, the result will be closer to the global minimum of $\psi(.)$. It should be noted that this method is not limited to the case of one-dimensional x, and can be used for $x \in \mathbb{R}^n$, i.e., optimizing multi-variable functions. However, it is applicable to the problems which the global optimum is finite and an estimate of the domain of interest, containing the global optimum exists, in order to be used in Algorithm 1 for training the networks.

Further analyzing the property of constant costate vector, given in Eq. (48), provides some interesting result given in Theorem 3.

Theorem 3: The optimal cost-to-go function $J_k^*(x)$, $\forall k \in K$, resulting from minimizing cost function (44) subject to dynamics (42) is a strongly convex function with respect to x in the domain of interest. More specifically, $J_k^*(x)$ is a paraboloid with a unique minimum.

Proof: Considering (49) and the fact that $\lambda = \nabla J_k^*(x)$, $\forall k$, one has

$$\nabla J_k^*(x) = (x - x_N)/(N\Delta t), \forall k \in K, \forall x \in \Omega. \tag{51}$$

Calculating the gradient of (51) leads to the Hessian of $J_k^*(.)$

$$J_{k_{xx}}^*(x) = 1/(N\Delta t)I > 0, \forall k \in K, \forall x \in \Omega. \tag{52}$$

The abovementioned relations proves that $J_k^*(.)$ is strongly convex in Ω [27]. Eq. (51) provides the shape of function $J_k^*(.)$. It says that function $J_k^*(.)$ is such that its gradient at each point x is proportional to the distance between x and x_N. Also, it says that the function has a unique point with zero gradient located at x_N. Such a function is a paraboloid centered at x_N. It can also be observed by integrating Eq. (51). From the convexity of $J_k^*(x)$ it follows that the paraboloid opens upward, i.e., point x_N is its 'minimum'. ■

Note that the convexity result given in Theorem 3, for the problem defined in this section, does not assume any condition on the size of the selected sampling time, despite the result given in Theorem 2. However, in order to use the ADP based algorithm discussed in this study for solving the problem, the condition of small enough sampling time is required to guarantee the convergence of the algorithm (Theorem 1).

VI. NUMERICAL ANALYSIS

In order to numerically analyze the global optimality of ADP scheme and its use in static minimization, a simple way is selecting a cost function with a non-convex terminal cost term and evaluating the performance of the ADP in providing the global optimum. To this end, two separate examples are selected; a single variable example and a multi-variable benchmark example, namely Rosenbrock/Banana function.

A. Optimizing a Single Variable Function

The function subject to minimization is a single variable non-convex function which has a local minimum as well as a global minimum. The function is

$$\psi(x) = 2.1333x^4 + 0.9333x^3 - 2.1333x^2 - 0.9333x + 1.$$

The function is plotted in Fig. 4. The global minimum is at $x = 0.67$, while the local minimum is at $x = -0.79$. The cost function given in Eq. (43) and discretized to (44) is selected with $c = 100$, in order to put a heavy weight on minimization of $\psi(.)$ as opposed to the control cost. The selected dynamics is of form (42) and the sampling time is selected as $\Delta t = 0.0001$, where $t_0 = 0$ and $t_f = 1\ s$, therefore, $N = 10^4$.

The basis functions are selected as polynomials given below

$$\phi(x) = [1, x, x^2, \dots, x^7],\ \sigma(x) = [1, x, x^2, \dots, x^6].$$

The least squares, as explained in the Appendix, is carried out over 100 points from the domain of interest selected as $-2 \le x \le 2$. Fig. 5 shows the history of the converged weights of the critic network. Once the training is done, the approximated optimal cost-to-go function $J_0(x)$ is given by the critic network as

$$J_0(x) = W_0^T \phi(x) = -.4276 \times 10^{-10} x^7 - .1554 \times 10^{-8} x^6 - .4842 \times 10^{-7} x^5 - .5592 \times 10^{-5} x^4 - .5019 \times 10^{-4} x^3 + .5050\, x^2 - .6973\, x + 21.77.$$

Neglecting terms with coefficients of order 10^{-5} and less, the approximated $J_0(x)$ is a parabola, which confirms the results given in Theorem 3. This result can also be observed through looking at the shape of $J_0(x)$ as plotted versus x in Fig. 6, without neglecting any term. The roots of the derivative of $J_0(x)$ turned out to be 0.6905, 71.223, 19.681$\pm$78.812i, -71.211$\pm$45.904i, where $i \equiv \sqrt{-1}$. Neglecting the roots which are out of the problem domain and the imaginary roots, the remained root is 0.6905. Hence, the minimum of $J_0(x)$ is very close to the global optimum of $\psi(x)$, as expected.

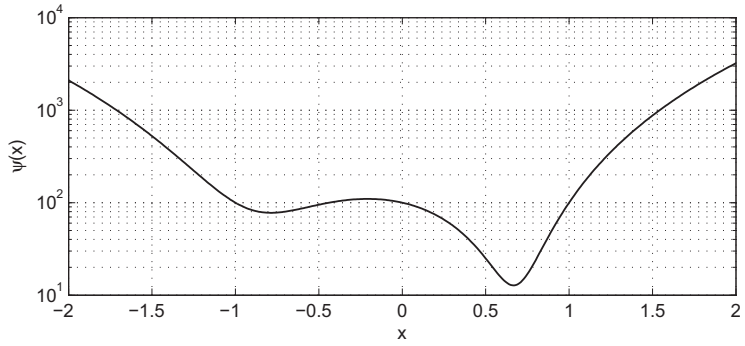

Fig. 4: Function subject to minimization, $\psi(x)$ versus x.

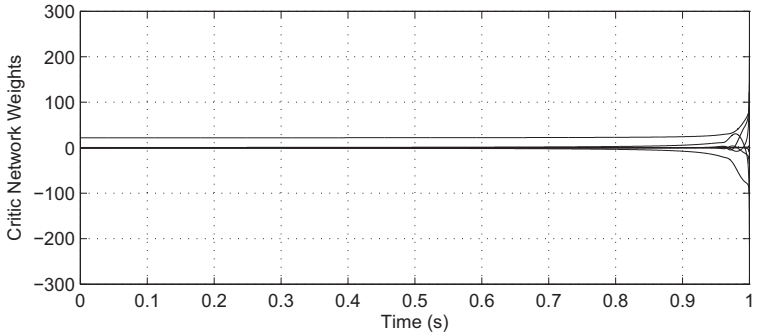

Fig. 5: Weight of the critic network versus time.

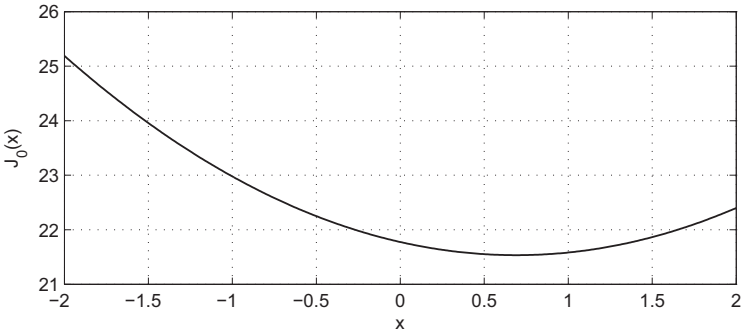

Fig. 6: Optimal cost-to-go versus x.

Five different initial conditions $x_0 \in \{-2, -1, 0, 1, 2\}$ are selected and each one is separately simulated using the dynamics given in Eq. (42), where the control is provided by the actor network. The resulting state and control histories are presented in Figs. 7 and 8, respectively. As seen in Fig. 7, the ADP scheme has resulted in $x_N \cong 0.700$ for different initial conditions, which is very close to the global optimum of $\psi(x)$, that is 0.67. It should be noted that some of the initial conditions are selected to be on the left

side of the local minimum of $\psi(x)$. It shows that the method does not go towards the local minimum. The controller, for such initial conditions, has passed the local minimum and has reached to the proximity of the global minimum.

Considering the applied control histories given in Fig. 8, the controls have been constant in the majority of the time as expected based on Eq. (48). However, some anomalies are observed at the end of the horizon, which could be due to the numerical errors of using NNs with finite number of basis functions. Instead of propagating the initial conditions x_0s to the terminal points x_N using the dynamics given in (42), one can use the relation given in (49) to fine x_N in one shot. Doing so for the selected five initial conditions leads to the terminal points x_N varying between 0.678 and 0.717 which still are very close to the global optimum.

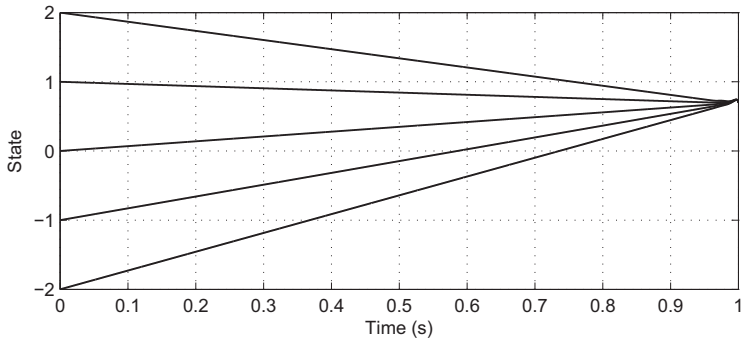

Fig. 7: State histories for different initial conditions $x_0 \in \{-2,-1,0,1,2\}$

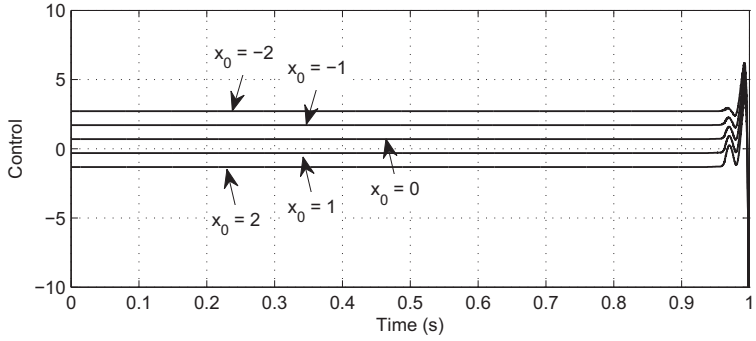

Fig. 8: Control histories for different initial conditions $x_0 \in \{-2,-1,0,1,2\}$

An interesting feature of developed ADP scheme is providing optimal solution for every shorter horizon as well, without needing to retrain the networks. This is due to Bellman's principle of optimality [17] which says that every optimal policy for a given

horizon remains optimal for the shorter horizons as well. Let the shorter horizon be $t_f = 0.5$ s. Note that having the optimal control for $t \in [0,1)$, the optimal control for the horizon of $t \in [0,0.5)$ is given by utilizing the network weights corresponding to the last 0.5 s of the original horizon. In other words, if W_0 to W_{10000} give the optimal critic weighs for $t \in [0,1)$, then, W_{5000} to W_{10000} give the optimal weighs for $t \in [0,0.5)$. Simulating the same five initial conditions using the shorter horizon of $t_f = 0.5$ s leads to the state and control histories given in Figs. 9 and 10, respectively. The resulting terminal points are still 0.700, close to the global optimum of $\psi(x)$. Comparing the applied control histories given in Fig. 10 for the shorter horizon, $t_f = 0.5$ s, with the applied control histories resulted from the longer horizon of $t_f = 1$ s, given in Fig. 8, it can be observed that the actor has smartly applied almost twice larger controls in order to get to point x_N in half the time that the controls given in Fig. 8 were applied.

The plot of the optimal cost-to-go function, for the time-to-go of 0.5 s, which is given by $W_{5000}^T \phi(x)$, is supper-imposed on the plot of the same function for the time-to-go of 1 s, which is given by $W_0^T \phi(x)$, in Fig. 11. The minimum of the new parabola has slightly moved, from 0.690 in $t_f = 1$ s case to 0.695 in $t_f = 0.5$ s case. This observation is compatible with the theoretical result given in section V, i.e., as N becomes larger, the results get closer to the global minimum of $\psi(.)$. Considering the mathematical formula for the new cost-to-go function, given below, shows that the new parabola has almost twice the slope that the previous parabola had, in order to generate twice larger controls for the shorter horizon.

$$W_{5000}^T \phi(x) \cong 1.0203x^2 - 1.4086x + 22.0202, \text{ for } t_f = 0.5 \text{ s.}$$

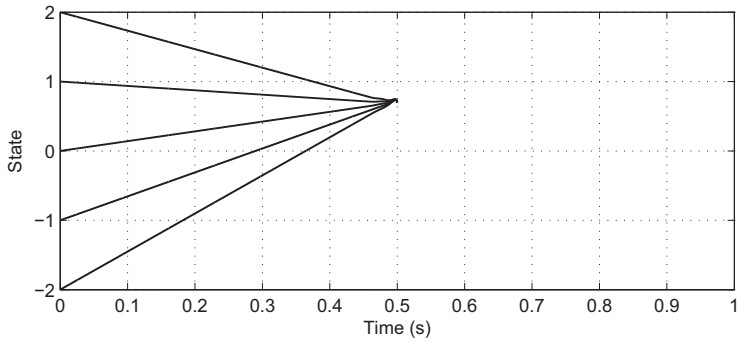

Fig. 9: State histories for different initial conditions $x_0 \in \{-2, -1, 0, 1, 2\}$, with $t_f = 0.5$.

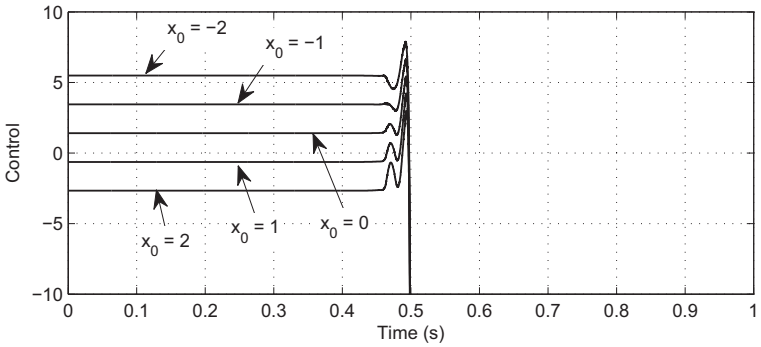

Fig. 10: Control histories for different initial conditions $x_0 \in \{-2, -1, 0, 1, 2\}$, with $t_f = 0.5$.

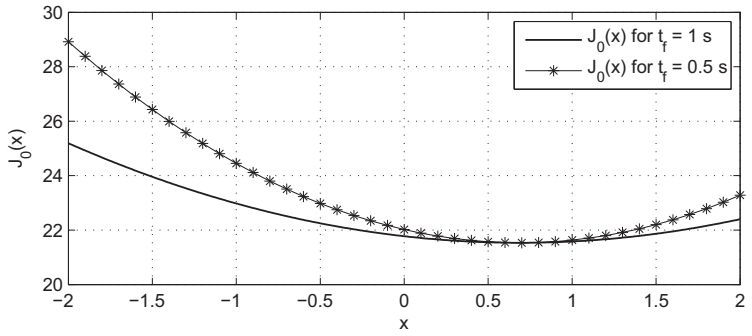

Fig. 11: Optimal cost-to-go versus x, for different t_fs.

It is noteworthy that as discussed in Example 1 in section IV, if the horizon is shortened too much without refining the sampling time, the method may fail to lead to the global optimum of $\psi(.)$. This is showed in Fig. 12, where, the very short horizon of $t_f = 0.01$ s with the sampling time used before, is utilized for propagating the five initial conditions. As seen in this figure, some of the initial conditions are led to the proximity of the global minimum of $\psi(.)$, while, some other are absorbed by the local minimum of the function at $x = -0.79$. Re-training the network for the short horizon of $t_f = 0.01$ s but with smaller sampling time is seen to solve this issue and present the same behavior observed in Figs. 7 and 9.

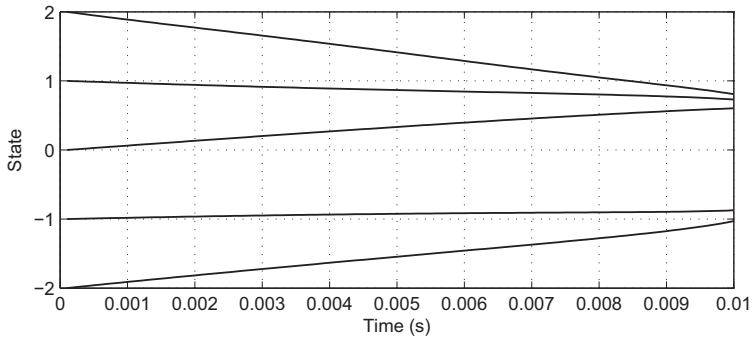

Fig. 12: State histories for different initial conditions $x_0 \in \{-2, -1, 0, 1, 2\}$, with $t_f = 0.01$.

B. Optimizing a Multi-variable Function

For the second example, a benchmark optimization problem is selected, namely, the Rosenbrock (Banana) function [29].

$$\psi(X, Y) = (1 - X)^2 + (Y - X^2)^2$$

The Rosenbrock function is a non-convex function with two independent variables (See Fig. 13). It has a global minimum at $(X, Y) = (1,1)$. Defining the state vector as $x \equiv [X, Y]^T$, the ADP scheme is utilized for solving the optimal control problem with the cost function given in (44) and the discretized dynamics (42). The design parameters are selected as $c = 5$, $t_0 = 0$, $t_f = 10$ s, $\Delta t = 0.005$, hence, $N = 2000$. Moreover, the basis functions are selected as

$$\phi(x) = [1, X, Y, XY, X^2, Y^2, \dots, X^8, Y^8]$$
$$\sigma(x) = [1, X, Y, XY, X^2, Y^2, \dots, X^7, Y^7]$$

The least squares method is done using 100 randomly selected points from the domain of interest $\Omega \equiv [-2,2] \times [-2,2]$, which is a typical region selected for this benchmark problem [30]. Once the network training is carried out, the approximated optimal cost-to-go function $J_0(X, Y)$ is observed to be given by the paraboloid

$$J_0(X, Y) = W_0^T \phi(x) \cong 0.0498X^2 + 0.0482Y^2 - 0.0012XY - 0.0972X - 0.1002Y - 0.1107,$$

where the higher order terms with the coefficient less than 10^{-5} are skipped. Function $J_0(X, Y)$, without neglecting the higher order terms, is plotted in Fig. 14. Finding the roots of the jacobian of $J_0(X, Y)$, which leads to the extrema of $J_0(X, Y)$, gives the only real

root of $(X, Y) = (0.988, 1.048)$. It is seen that the sole minimum of the cost-to-go function is very close to the global minimum of the Rosenbrock function.

In order to analyze the performance of the developed method in global minimization of the cost function, 25 different initial conditions $x_0 \in \{-2, -1, 0, 1, 2\} \times \{-2, -1, 0, 1, 2\}$ are selected and separately simulated using the controls provided by the actor network. The resulting state trajectories are presented in Fig. 15. In this figure, the level curves of the Rosenbrock function are also plotted. It can be seen, that each one of the initial conditions has been directly led toward the global minimum, regardless of the orientation of the level curves at the points. This behavior shows the advantage of the method over, for example, steepest descent method, in which, the states move in the direction that is perpendicular to the local level curves [17,16].

The state trajectories are super-imposed on the level curves of the optimal cost-to-go function $J_0(x_0)$ in Fig. 16. In this figure, however, one can see that the directions of movement of the trajectories are (almost) perpendicular to the local level curves of the cost-to-go function. This shows the analogy between the developed method and the steepest descent method, with the difference that the ADP algorithm follows the gradient of the cost-to-go function, not that of $\psi(.)$. It should be noted that each u_k, for different ks, will be selected to be perpendicular to the level curves of the respective $J_k(.)$, while, only those of $J_0(.)$ are plotted in Fig. 16. However, the level sets of the rest of $J_k(.)$s also were observed to have the same shape, i.e., circles/ellipses centered at a point close to the global minimum point of $\psi(.)$, as expected based on Theorem 3.

VII. CONCLUSIONS

The performance of approximate dynamic programming in finding the global optimal solution to the fixed-final-time control problem was investigated. A sufficient condition for global optimality of the result, regardless of the convexity or non-convexity of the functions representing the dynamics of the system or the state penalizing terms in the cost function, was derived. Moreover, an idea was presented in converting a static function optimization to an optimal control problem and using ADP for approximating the global minimum of the selected convex or non-convex function. Numerical results showed that the proposed method results in a trajectory which directly goes to the proximity of the global minimum, regardless of the shape of the local level curves. This

is a promising feature which differentiates the method from many nonlinear programming based optimization methods.

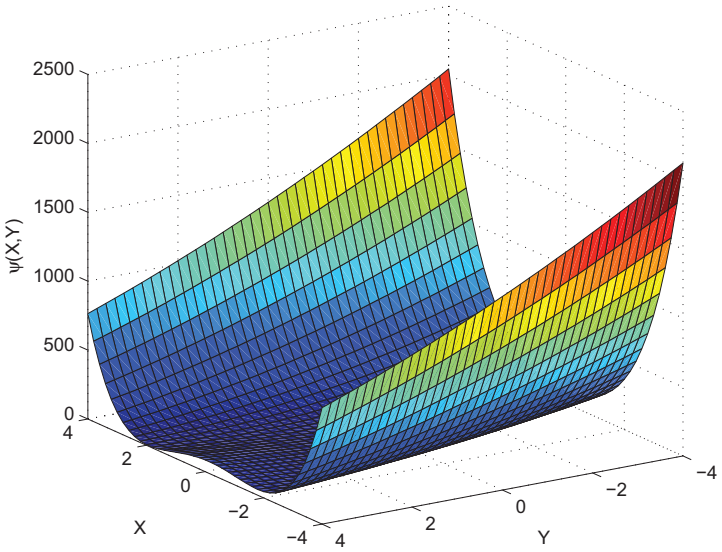

Fig. 13: Rosenbrock function subject to minimization versus the inputs X and Y.

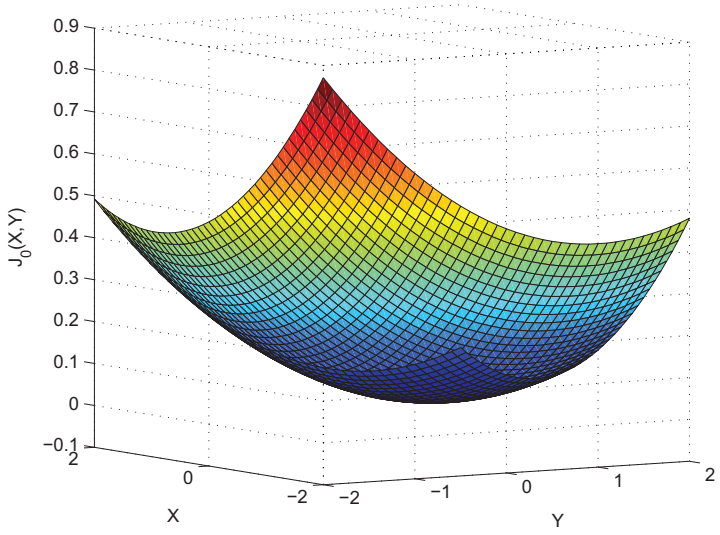

Fig. 14: Optimal cost-to-go versus X and Y.

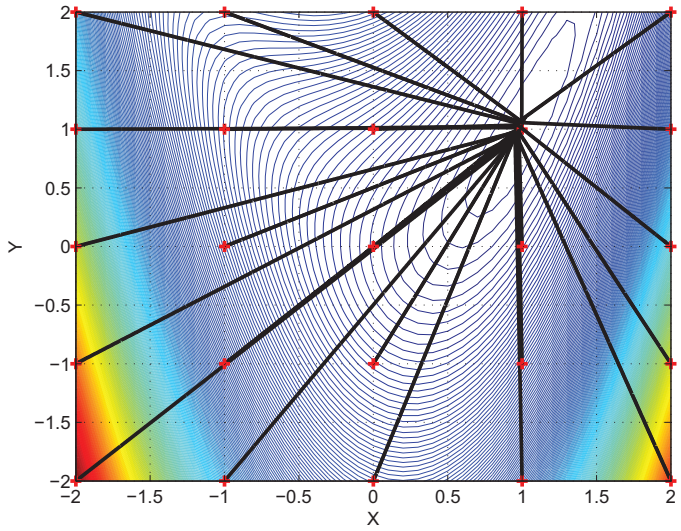

Fig. 15: Level curves of the Rosenbrock function and state trajectories for different initial conditions $x_0 \in \{-2,-1,0,1,2\} \times \{-2,-1,0,1,2\}$. The red plus signs denote the initial point of the respective trajectory.

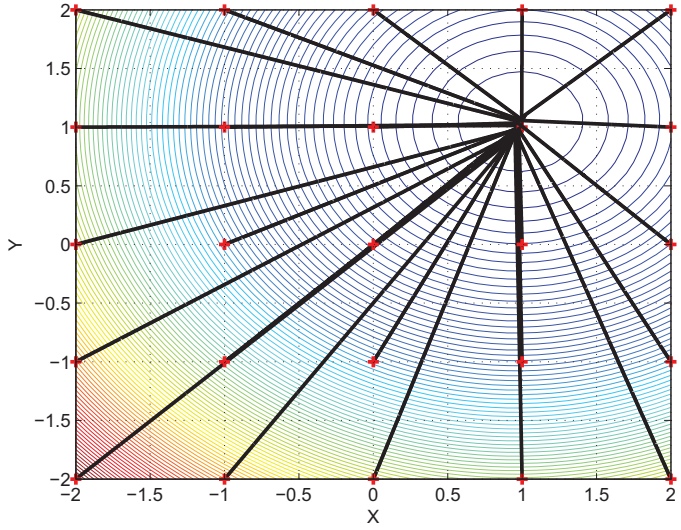

Fig. 16: Level curves of the optimal cost-to-go and state trajectories for different initial conditions $x_0 \in \{-2,-1,0,1,2\} \times \{-2,-1,0,1,2\}$. The red plus signs denote the initial point of the respective trajectory.

ACKNOWLEDGEMENT

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

APPENDIX

In Steps 2, 9, and 10 of Algorithm 1, the least squares method can be used for calculating V_k and W_k. For example, considering Step 9 of the algorithm, the objective is finding V_k such that it solves

$$\begin{cases} {V_k^j}^T \sigma\left(x_k^{[1]}\right) = u_k^{[1]} \\ {V_k^j}^T \sigma\left(x_k^{[2]}\right) = u_k^{[2]} \\ \vdots \\ {V_k^j}^T \sigma\left(x_k^{[n]}\right) = u_k^{[n]} \end{cases}. \tag{53}$$

Define

$$\boldsymbol{\sigma} \equiv \left[\sigma\left(x_k^{[1]}\right) \quad \sigma\left(x_k^{[2]}\right) \quad \ldots \quad \sigma\left(x_k^{[n]}\right)\right],$$

$$\boldsymbol{u} \equiv \left[u_k^{[1]} \quad u_k^{[2]} \quad \ldots \quad u_k^{[n]}\right].$$

Using the method of least squares, the solution to system of linear equations (53) is given by

$$V_k = (\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\boldsymbol{u}^T \tag{54}$$

Note that for the inverse of matrix $(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)$, which is a $p \times p$ matrix, to exist, one needs the basis functions $\sigma(.)$ to be linearly independent and n to be greater than or equal to the number of the basis functions.

REFERENCE

[1] 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.

[2] S. N. Balakrishnan, and V. Biega, "Adaptive-critic based neural networks for aircraft optimal control", *Journal of Guidance, Control & Dynamics*, Vol. 19, No. 4, 1996, pp. 893-898.

[3] D. V. Prokhorov, and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Networks*, Vol. 8, No. 5, 1997, pp. 997-1007.

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

[5] T. Dierks, B. T. Thumati, and S. Jagannathan, "Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence," *Neural Networks*, vol. 22, pp. 851-860, 2009.

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

[7] G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13 (3), pp. 764-773, 2002.

[8] 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.

[9] J. Ding and S. N. Balakrishnan, "An online nonlinear optimal controller synthesis for aircraft with model uncertainties," in Proc. *AIAA Guidance, Navigation, and Control Conference*, 2010.

[10] D. Han and S. N. Balakrishnan, "State-constrained agile missile control with adaptive-critic-based neural networks," *IEEE Trans. Control Systems Technology*, Vol. 10, No. 4, 2002, pp. 481-489.

[11] A. Heydari and S. N. Balakrishnan, "Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 24 (1), 2013, pp. 145-157.

[12] A. Heydari and S. N. Balakrishnan, "Fixed-final-time Optimal Control of Nonlinear Systems with Terminal Constraints," submitted to *Neural Networks*.

[13] F. Wang, N. Jin, D. Liu, and Q. Wei, "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw.*, vol. 22 (1), pp. 24-36, 2011.

[14] R. Song and H. Zhang, "The finite-horizon optimal control for a class of time-delay affine nonlinear system," *Neural Comput & Applic*, DOI 10.1007/s00521-011-0706-3, 2011.

[15] Q. Wei, and D. Liu, "An iterative ϵ-optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state," *Neural Networks*, vol. 32, pp. 236-244, 2012.

[16] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999, pp. 1-187.

[17] D. E. Kirk, *Optimal control theory: an introduction*. New York: Dover Publications, 2004, pp. 54,75,332.

[18] K. Homik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

[19] J. G. Attali, and G. Pages, "Approximations of Functions by a Multilayer Perceptron: a New Approach," *Neural Networks*, vol. 10 (6), pp. 1069-1081, 1997.

[20] M. Stone, and P. Goldbart, *Mathematics for Physics - A Guided Tour for Graduate Students*, Cambridge, England, Cambridge University Press, p. 70, 2009.

[21] H. Khalil, *Nonlinear Systems*, 3rd ed., Prentice Hall, New Jersey, 2002, pp.653-656.

[22] J. E. Marsden, T. Ratiu, and R. Abraham, *Manifolds, Tensor Analysis, and Applications*, 3rd ed. Springer-Verlag Publishing Co., New York, 2001, p. 73.

[23] W. F. Trench, *Introduction to Real Analysis*, available online at http://ramanujan.math.trinity.edu/ wtrench/texts/trench_real_analysis.pdf, 2012, pp. 313.

[24] C.-T. Chen, *Linear System, Theory and Design*, Oxford University Press, New York, 3rd Edition, 1999, p. 131.

[25] B. Chachuat, *Nonlinear and Dynamic Optimization: From Theory to Practice*, EPFL, 2007, pp.120-121.

[26] T. T. Shannon, and G. G. Lendaris, "A new hybrid critic-training method for approximate dynamic programming," Proceedings of *International Society for the System Sciences*, ISSS'2000. Available online at: web.pdx.edu/~tads/papers/20060.pdf.

[27] S. Boyd, and L. Vandenberghe*, Convex Optimization*, Cambridge University Press, 2009, pp.4-7,71,459.

[28] J. R. Magnus, and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*, 3rd ed. John Wiley & Sons, 2007, p. 231.

[29] M. French, Class notes of Applied Optimization course, Department of Mechanical Engineering Technology, Purdue University, http://www2.tech.purdue.edu/Met/Courses/MET503/.

[30] M. Molga, and C. Smutnicki, "Test functions for optimization needs," *Computer and Information Science*, 2005, pp.1-43.

4. OPTIMAL MULTI-THERAPEUTIC HIV TREATMENT USING A GLOBAL OPTIMAL SWITCHING SCHEME

Ali Heydari and S. N. Balakrishnan

ABSTRACT

The problem of multi-therapeutic HIV treatment is posed in this study as a switching problem to find the optimal switching time between the different therapies. To solve the optimal switching problem with nonlinear subsystems an algorithm is developed for learning the cost-to-go as a function versus *different switching times* and *different initial conditions*. Once the function is obtained in a closed form, finding optimal switching time for every given initial condition reduces to a function optimization. Through numerical simulations of a model for the HIV problem, the proposed algorithm is shown to be a useful tool for solving this class of problems.

I. INTRODUCTION

Modeling and control of HIV infection has been investigated by different researchers and several models have been developed in the literature for describing the progress of the HIV disease [1-8]. Optimal control methods have been shown in numerical simulations to offer promising treatment for HIV [9-14]. There are different therapies for HIV infection, but it is known that switching therapeutic options are useful for better control of the HIV progress [14,15] due to certain factors including the existing mutations between the viral strain types and their vulnerabilities toward each specific therapy. References [10-13] investigated optimal control of HIV with a single therapy, with the view of controlling the dosage/efficiency of the medicine in order to optimize the performance index. In [14] however, different therapies were considered but with fixed medicine dosages. The problem was defined as finding the optimal switching time between the therapies. By assuming different therapies, the evolution of the disease under each therapy, modeled by a scalar linear model for each viral strain type, was considered as a *mode*. Afterward, the optimal time for switching between the modes, to have the best result at the end of a specified time period, was sought. Hence, the problem reduced to formulating a scheme to find optimal switching time for a *switching system*.

Switching systems, appearing in different fields [14,16-20], are comprised of subsystems with different dynamics one of which is active at each time instant. Hence,

controlling these processes involves not only applying a proper control to the system, but also involves *decision making* to determine *when* to switch and *what* mode to switch to. Note that if a fixed mode sequence is assumed, the main problem is to find optimal switching instants, and once they are found, the problem reduces to a conventional optimal control problem to find the control.

The developed methods for solving switching problems can be classified into two categories; in the first category, through different schemes the gradient of the cost with respect to the switching instants are calculated and used in a nonlinear programming method to find the local optimal switching times/points [21-28]. In many of these developments the sequence of active subsystems, called mode sequence, is selected a priori [21-27], and the problem is finding the switching instants between the modes. A depth search was done in [28] after freezing the first and last subsystems, to find the entire possible mode sequences and for every such sequence, the optimal switching instants are calculated using nonlinear programming. Using the iterative solution to the nonlinear optimization problem along with ideas from model predictive control, the authors of [25] developed the so-called crawling window optimal control scheme for the optimal switching problem.

The second category includes studies that discretize the switching problem to deal with a *finite* number of options. An optimization scheme was developed in [14] to find the optimal mode sequence and switching time for positive linear systems with a *pre-fixed* initial condition. Some remedies were suggested to decrease the computational load of the proposed algorithm which grows exponentially with the growth of the number of time steps and the number of modes. Authors of [29] utilized a direct search to evaluate the cost function for different randomly selected switching time sequences. In [30] the discretization of the state and input spaces was used for calculation of the value function for optimal switching through dynamic programming. As for the intelligent methods to the problem, genetic algorithm and neural networks were used in [31] and [32], respectively, to find the optimal switching for a preselected initial condition.

All the cited methods require a large amount of computations to numerically find the optimal switching time for an *a priori selected initial condition*; each time the initial condition is changed, a new set of computations needs to be performed to find the

corresponding optimal switching instants. In [24] the switching parameter was found as the local optimum in the sense that it minimizes the worst possible cost for all trajectories starting in the selected set of initial states, in order to extend the validity of the results for different initial conditions within a pre-selected set. Also, the derivative of the switching parameters with respect to the initial conditions was sought through sensitivity analysis.

If the function giving the optimal performance index for every given switching time sequence is known explicitly, then the problem simplifies to optimization of the function with respect to the switching instants. However, even for the general linear subsystems with a quadratic cost function, this function is not available [22,33].

The main contribution of this paper is presenting an algorithm by considering the optimal HIV treatment problem as a forced switching system with autonomous subsystem dynamics and learning the performance index as a function of current state and the switching instants, i.e., the desired function. Forced switching systems are those in which the switching between the modes is directly controlled and dictated. For existing methods on optimal switching of systems with autonomous subsystem dynamics, one may refer to [14,23-25]. The approach formulated in this paper is motivated by studies in intelligent control as in [34-36]. This method involves training a neural network (NN) to learn the nonlinear mapping between the optimal cost-to-go and the switching instants. Once this function is learned, finding the optimal switching times simplifies to minimization of an analytical function. As compared to available methods in the literature for the HIV treatment and generally for switching systems, the proposed technique has two advantages. They are: 1) the method developed in this paper gives *global* optimal switching instants versus local ones resulting from nonlinear programming based methods, 2) the learned function provides the optimal cost based on the switching instants for a vast domain of *initial conditions*. Therefore, optimal switching times for different initial conditions can be easily calculated using the *same* trained NN. This means that switching instants for HIV therapies with different modes is provided for a vast number of initial states through our method.

The rest of this paper is organized as follows: Modeling and formulation of the HIV treatment problem are given in section II, the switching problem in the general form is detailed in section III and the proposed solution is described in section IV. An example

HIV infection control problem is solved in section V and the optimal switching scheme is applied for finding the optimal drug prescription. Conclusions are presented in section VI.

II. MODELING AND FORMULATION OF THE HIV TREATMENT PROBLEM

Any model used for describing the evolution of the HIV infection [4-8] and control of the disease should at least incorporate three variables [12]; they are the number of uninfected target cells ($CD4^+$ T cells), denoted with T, the number of target cells infected by the virus v_i, denoted with T_i, and the population of v_i, denoted with V_i. The subscript i corresponds to the viral strain type. In this study, two viral strains, denoted with v_1 and v_2, are considered, hence, $i \in \{1,2\}$. In other words, the case of co-infection by different viral strain types is considered. Different therapies will have different effectiveness in controlling each strain. Moreover, due to the mutative nature of the strains, each strain can generate the other strain by mutation.

The model given in [8,12] which is the multi-virion type version of the ones in [4,6] is adapted here.

$$\begin{aligned}
\dot{T} &= s - d_T T - \beta_1 V_1 T - \beta_2 V_2 T \\
\dot{T}_1 &= (1-\mu_1)\beta_1 V_1 T + \mu_2 \beta_2 V_2 T - \delta T_1 \\
\dot{T}_2 &= (1-\mu_2)\beta_2 V_2 T + \mu_1 \beta_1 V_1 T - \delta T_2 \\
\dot{V}_1 &= p_{1_j} T_1 - c V_1 \\
\dot{V}_2 &= p_{2_j} T_2 - c V_2
\end{aligned} \tag{1}$$

where the constants used in the model are defined in Table 1. The listed values in the table are in the range given in [7,8].

Assuming two different therapies for the HIV infection, following [14], different viral production rate are assumed under different therapies. Indexing the therapies by a and b, the selected values for the parameters are

$$p_{1_a} = 2\ day^{-1},\ p_{1_b} = 1\ day^{-1},\ p_{2_a} = 1\ day^{-1},\ p_{2_b} = 2\ day^{-1}.$$

hence, therapy a performs better in controlling virion v_2 while therapy b is better for control of v_1. Considering the definition of each parameter used in the model (1), it is

straight forward to see the effect of each term on the right hand side of the differential equations given in the model for different states. For example, the infection of the healthy/uninfected target cells by each virus is proportional to the infection rate constant β_i, the total number of uninfected cells denoted with T and the concentration of the respective virus, denoted with V_i. Hence, the terms $-\beta_1 V_1 T$ and $-\beta_2 V_2 T$ are listed in the right hand side of the differential equation for $\dot{T}$, corresponding to the reduction of T due to the infection of healthy cells by v_1 and v_2, respectively.

As seen in (1), the dynamics are autonomous with two modes corresponding to the evolution of the disease under the two therapies. The problem is defined as finding the optimal time for switching between the autonomous modes to have the highest number of uninfected target cells at the end of a specified time period. In the next section, the problem is posed as an optimal switching problem with autonomous subsystems and in the subsequent section, the solution method is developed.

III. GENERAL PROBLEM FORMULATION

A switched autonomous system can be represented by the set of M subsystems

$$\dot{x}(t) = f_i\big(x(t)\big) \tag{2}$$

where $f_i: \mathbb{R}^n \to \mathbb{R}^n$, $i \in I \equiv \{1, 2, \dots, M\}$, and n denotes the dimension of the state vector $x(t)$. *Optimal solution* is defined as designing a switching sequence that allows the operation of the system, from the initial time t_0 to the final time t_f, i.e., $t \in [t_0, t_f]$, to switch between different subsystems such that the performance index, defined as the cost or revenue, given below is optimized.

$$J = \psi\left(x\big(t_f\big)\right) + \int_{t_0}^{t_f} Q\big(x(t)\big)\, dt \tag{3}$$

Table 1: The parameters of the HIV model.

Constant	Description	Value
s	Production rate of healthy cells.	$10\ mm^{-3}day^{-1}$
d_T	Death rate of healthy cells.	$0.01\ day^{-1}$
β_i	Rate of infection of healthy cells by virion v_i, $i \in \{1,2\}$.	$3 \times 10^{-5}\ mm^{-3}day^{-1}$
μ_i	Rate of mutation of virion v_i to v_j, $i, j \in \{1,2\}, i \neq j$.	1×10^{-4}
δ	Death rate of infected cells.	$0.24\ day^{-1}$
p_{i_j}	Production rate of virion i under therapy j, $j \in \{a, b\}$.	In the range 1 to 2 day^{-1}
c	Death rate of virus.	$0.24\ day^{-1}$

Convex functions $Q: \mathbb{R}^n \to \mathbb{R}$ and $\psi: \mathbb{R}^n \to \mathbb{R}$ correspond to the cost during the time period and at the end, respectively. A switching sequence in time interval $[t_0, t_f]$ can be defined as [22]

$$s = \big((t_0, i_0),\ (t_1, i_1), \dots, (t_K, i_K)\big)$$

where $t_0 \le t_1 \le \cdots \le t_K \le t_f$, $0 \le K < \infty$ and $i_k \in I$, for $k = 1,2, \dots, K$. In this notation, (t_k, i_k) means that the system switches from subsystem i_{k-1} to i_k at time t_k. Following [21-27], the order of the active subsystems is frozen, i.e., mode sequence, in this study and therefore, the problem is to find optimal switching times. Since the mode sequence is pre-selected, the unknowns in this problem will be the switching time sequence given by

$$\tau = (t_0,\ t_1,\ t_2, \dots,\ t_K)$$

where $t_0 \le t_1 \le \cdots \le t_K \le t_f$, and integer K denotes the number of switching, $0 \le K < \infty$.

IV. COST-TO-GO FUNCTION APPROXIMATION

In this section, two algorithms are proposed for learning the performance index, or cost-to-go of the system for a given switching sequence and different initial conditions. A NN is trained using the proposed algorithm as a universal function approximator.

A. Cost-to-go Approximation for a Conventional Problem

Assumption of a pre-selected switching time sequence reduces the switching system to a conventional time-varying system where different modes are active at different fixed switching times. The developed algorithm is motivated by the notion of adaptive critics (AC) developments in implementation of Heuristic Dynamic Programming (HDP) [35], where the critic network learns the cost-to-go and the actor learns the optimal control. In this study the actor is skipped, and the critic is utilized to learn the cost-to-go at each time step k, and state vector x_k for the nonlinear time-varying system. Discretizing the system in (2) by selecting a sampling time Δt results in discrete-time dynamics of the subsystems

$$x_{k+1} = \bar{f}_i(x_k),\ \ k = 0, 1, 2, \dots, N,\ \ i \in I$$

where $N = (t_f - t_0)/\Delta t$, $x_k = x(k\Delta t + t_0)$, and $\bar{f}_i(x_k) \equiv x + \Delta t f_i(x)$ if Euler integration is used. The discretized cost (or revenue) function is given by

$$J = \psi(x_N) + \sum_{k=0}^{N-1} \bar{Q}(x_k) \tag{4}$$

where $\bar{Q}(x_k) \equiv \Delta t Q(x)$. Note that the cost-to-go, defined as the cost incurred by the propagation of the states from current time to the final time, at each time step k depends on the current state x_k and the time-to-go $N - k$, i.e.,

$$J_k(x_k) = \psi(x_N) + \sum_{i=k}^{N-1} \bar{Q}(x_i) \tag{5}$$

Selecting a linear in the parameter NN as the function approximator, the expressions for the critic (cost-to-go approximator), can be written as

$$J_k(x_k) = W_k^T \phi(x_k), \quad k = 0,1, \dots, N$$

Vector $W_k \in \mathbb{R}^m$ is the unknown weights of the network at time step k, and the basis functions are given by $\phi: \mathbb{R}^n \to \mathbb{R}^m$ for m being a positive integer denoting the number of neurons.

Considering cost function (5), it can be seen that the cost-to-go satisfies the following recurrence relation

$$J_N(x_N) = \psi(x_N),\ J_k(x_k) = \bar{Q}(x_k)\ + J_{k+1}(x_{k+1}),\ k = 0,1, \dots, N-1 \tag{6}$$

which will be used for learning the cost-to-go. The training process may be detailed through Algorithm 1.

Algorithm 1

Step 1: Train the network weight W_N such that

$$W_N^T \phi(x_k) = \psi(x_k) \tag{7}$$

for different $x_k \in \Omega$ where Ω denotes the domain of interest.

Step 2: Set $k = N - 1$.

Step 3: Randomly select state vector $x_k \in \Omega$.

Step 4: Set $x_{k+1} = \bar{f}_j(x_k)$ (where j denotes the index of the active subsystem at time k based on the pre-selected switching time sequence.)

Step 5: Train the network weight W_k such that

$$W_k^T \phi(x_k) = \bar{Q}(x_k) + W_{k+1}^T \phi\left(\bar{f}_j(x_k)\right) \tag{8}$$

Step 6: Repeat steps 3 to 5 until W_k converges for different random $x_k \in \Omega$.

Step 7: Set $k = k - 1$ and go to Step 3, until $k = 0$.

As for the NN training method, in steps 1 and 5 of Algorithm 1, one may use the least squares for finding the unknown W_k in one shot, i.e., in the Batch training scheme. Appendix A discusses this process.

Remark 1: The only requirement for ending up with the cost-to-go estimator of the *switching system* with frozen switching times is utilizing the respective *active subsystem* at each time index k in the process of propagation of x_k to x_{k+1}, in order to be used in the weight update equation of W_k.

B. Cost-to-go Approximation for the Switching Problem

As seen in the previous subsection, using a fixed switching time sequence, the cost-to-go function can be found using Algorithm 1. In this subsection, the network structure and the training algorithm is modified to find the optimal switching times. This process is described in the next paragraph and the resulting algorithm is presented subsequently.

In order to find the optimal switching times, the critic network is trained to approximate the cost-to-go (or revenue) for every given switching time sequences. Denoting the cost-to-go for a set of initial conditions x_0 and a switching time sequence τ as $J_0(x_0, \tau)$, for different τs and different x_0s function $J_0(x_0, \tau)$ is learned. Since the mapping functions are selected to be analytical, finding the optimal τ for a given x_0 simplifies to finding the minimum of $J_0(x_0, \tau)$ with respect to τ. For example, if the problem involves only one switch, then the only unknown of the optimal switching sequence $\tau = (t_0, t_1)$ is t_1, and finding it is as simple as calculating the roots of the derivative of $J_0(x_0, \tau)$ with respect to t_1 and comparing the value of $J_0(x_0, \tau)$ at those roots along with the value at the boundary points to find the global optimum. As mentioned earlier, even for linear systems with a quadratic cost function, function $J_0(x_0, \tau)$ is not known in a closed form, i.e., with respect to x_0 and τ [22,33].

Here, Algorithm 1 is modified to learn $J_0(x_0, \tau)$. For this purpose, the following modified network structure is proposed:

$$J_k(x_k, \tau) = W_k^T \phi(x_k, \tau) \quad k = 0,1, \dots, N$$

The inputs to the NN are selected as the current state *and the given switching time sequence* on which the cost-to-go is dependent. In order to accommodate the extra input, the new basis function is $\phi: \mathbb{R}^n \times \mathbb{R}^K \to \mathbb{R}^m$, where, the switching time sequence τ is considered as a K-vector $[t_1, t_2, \dots, t_K]^T$. By using this structure for the critic network along with the recurrence relation (6), Algorithm 2 gives the training process to find W_k, $\forall k$.

Algorithm 2

Step 1: Using least squares find W_N such that

$$W_N^T \phi(x_k, \tau) = \psi(x_k) \tag{9}$$

for different random $x_k \in \Omega$ and different random switching time sequences.

Step 2: Set $k = N - 1$.

Step 3: Randomly select both state vector $x_k \in \Omega$ and switching time sequence τ.

Step 4: Set $x_{k+1} = \bar{f}_j(x_k)$ (where j denotes the index of the active subsystem at time k based on the selected switching time sequence τ in Step 3.)

Step 5: Using least squares find W_k such that

$$W_k^T \phi(x_k, \tau) = \bar{Q}(x_k) + W_{k+1}^T \phi\left(\bar{f}_j(x_k), \tau\right) \tag{10}$$

Step 6: Repeat steps 3 to 5 until W_k converges for different random $x_k \in \Omega$ and switching time sequences.

Step 7: Set $k = k - 1$ and go to Step 3, until $k = 0$.

Remark 2: Comparing Algorithm 2 with Algorithm 1, it can be observed that the only modification is selecting different random τs at each iteration and selecting the state equation for propagation of x_k to x_{k+1} based on the currently selected τ.

Remark 3: As compared to the cited methods in the introduction, the advantages of the method presented here are twofold: 1) global optimal switching time sequence is obtained

rather than local ones given by [21-28], 2) the network provides the optimal switching times for *every other initial conditions as well* while studies [14, 21-32] calculate optimal solutions for only one pre-selected initial condition.

C. A Transformation for the Switching Problem

At the switching instants between the subsystems, the weight history W_k may not be smooth. In order to facilitate this issue with the selected controller design, a transformation as used in [22] for a different purpose, is carried out. To motivate the basic idea, let us assume that the number of subsystems is two with one switching instant at t_1. Define a new independent variable $\hat{t} \in [0\ 2]$ as

$$t = \begin{cases} t_0 + (t_1 - t_0)\hat{t} & if\ 0 \le \hat{t} < 1 \\ t_1 + (t_f - t_1)(\hat{t} - 1) & if\ 1 \le \hat{t} \le 2 \end{cases}$$

one has $t = t_0$ if $\hat{t} = 0$, $t = t_1$ if $\hat{t} = 1$, and $t = t_f$ if $\hat{t} = 2$. Using the new independent variable $\hat{t}$, the state equations given in (2) can be expressed as

$$x'(\hat{t}) = (t_1 - t_0) f_1\big(x(\hat{t})\big)$$

$$x'(\hat{t}) = (t_f - t_1) f_2\big(x(\hat{t})\big)$$

where x' denotes the derivative of x with respect to $\hat{t}$. The performance index (3) converts to

$$J = \psi\big(x(2)\big) + (t_1 - t_0)\int_0^1 Q\big(x(\hat{t})\big)\, d\hat{t} + (t_f - t_1)\int_1^2 Q\big(x(\hat{t})\big)\, d\hat{t} \qquad (11)$$

As can be seen, the benefit of the transformed time is the fact that the switching always happens at a fixed transformed time of $\hat{t} = 1$. Note that the actual switching time is still free and given by t_1. This feature gives Algorithm 2 the capability of generating a desired form in the history of the weights at the time instant of $\hat{t} = 1$ to account for the subsystem switching at $t = t_1$. It should be noted that the transformation of the independent variable does not incur any change in the main problem, i.e., the solutions to the transformed problem and the original problem are identical.

For problems having more switches or more subsystems, e.g., number of switching equal to K, this solution can be extended where $\hat{t} \in [0\ \ K + 1]$ and the switches happen at $\hat{t} = 1,2,3, \dots, K$. For implementation of Algorithm 2, after performing this time

transformation, one needs to discretize the transformed performance index given in (11) and compare it with the general form of the performance index given in (4), in order to find the state penalizing term $\bar{Q}(x_k)$ for use in Algorithm 2.

V. NUMERICAL ANALYSIS

In this section, the proposed method for solving optimal switching problem of autonomous systems is applied to the HIV treatment problem defined in Section II, i.e*., optimal switching between the therapies to have the highest number of uninfected cells at the end of an specific time period*. The time period considered in this paper is three months and it is assumed that in the beginning of the period, the disease condition/progression is monitored at a clinical visit and an optimal prescription in the sense of optimal drug switching is to be prescribed for the next ninety days. Without loss of generality, assume that the drugs are supposed to switch not more than once. Later, the extension of the method for the cases of more than one switching will be discussed. Moreover, the sequence of the therapies, i.e., the mode sequence, is assumed to be given.

A. Utilizing the Optimal Switching Method for Solving the Problem

Considering the model given by (1), the performance index to be *maximized* is defined as

$$J = T_N$$

i.e., the number of healthy cells at the end of the horizon. The time from $t \in [0, 90]$, with the unit of day, is transformed to $\hat{t} \in [0, 2]$, as explained in subsection IV.C, the problem is discretized with a sampling time of $\Delta\hat{t} = 0.02$, resulting in 100 time steps.

Assuming a mode sequence of $\{a, b\}$, since there is only one switching, the switching sequence simplifies to one unknown, namely, the switching time denoted with t_1. Consequently, the NN will have six inputs with five of them being the elements of the state vector and the sixth being the switching time t_1.

An important step in the design is the selection of the basis functions. The well-known Weierstrass approximation theorem [37] proves that any continuous function on a closed and bounded interval can be uniformly approximated on that interval by polynomials to any degree of accuracy. Assuming the cost-to-go is a continuous function of states and switching times, the basis functions are selected as polynomials $T^i T_1^j T_2^k V_1^l V_2^m t_1^n$ where $i, j, k, l, m \in \{0,1,2,3\}$, $i + j + k + l + m \leq 3$ and $n \in \{0,1, \dots ,4\}$.

Note that as mentioned in the theorem, the accuracy can be adjusted by selection of order of the polynomials. In here, the mentioned orders are selected which showed good accuracy as will be seen in the rest of this section. This selection results in 280 neurons. Since the whole training is done offline, the number of neurons is not a barrier in the solution process.

The least squares method as described in Algorithm 2 is used in training. 1000 random state value and switching times are selected for training each W_k, $k = 0,1, \dots, N$. The domain of interested Ω for the network training is selected as $T \in [0, 1000]$, $T_i \in [0, 100]$, $V_i \in [0,100]$, for $i = 1,2$. During training, it was observed that normalizing the network inputs to belong to the interval $[0, 1]$ or even [0, 10] facilitates easier function approximation of the network with the selected basis functions. Therefore, the network inputs are normalized through dividing the state element T by 500, the rest of the state elements by the constant 10, and the switching time t_1 by constant 90 (to vary between 0 and 1). Notice that this is not a required step. The training process took 31 seconds using MATLAB 2010 on a machine with Intel Core 2 Duo 2.66 GHz and 2 GB of RAM.

The resulting weight evolution versus the time is depicted in Fig. 1. As expected, there is a corner in the history of some of elements of the weight matrix at the switching time which is fixed at the transformed time step 50. Moreover, the weight values at the final time are all zero except for the one corresponding to the normalized T, which has the final value of 500, since T was normalized through division by the same number. Hence, one will have $J_N(x_N, t_1) = W_N^T \phi(x_N, t_1) = T_N$, as expected.

The first simulation was performed with the initial conditions

$$T(0) = 1000\,, \qquad T_1(0) = 5, \qquad T_2(0) = 25, \qquad V_1(0) = 10, \qquad V_2(0) = 50$$

denoted with IC 1, where the unit of the numbers are mm^{-3}. The performance index for different switching times is calculated through exhaustive simulations of the model, i.e., simulating the system for the period of 90 days with every possible switching time, and the results are depicted in Fig. 2 through the dash-dot plots. Using the trained network, for this initial conditions, function $J_0(x_0, \tau)$ is approximated by $W_0^T \phi(x_0, \tau)$ given in terms of t_1 as

$$J(\tau) = -644.3t_1^4 + 1714t_1^3 - 1704t_1^2 + 719.0t_1 + 844.2 \qquad (12)$$

which is plotted in Fig. 2. It can be observed in Fig. 2 that the NN approximator has been able to approximate the performance index with *very good accuracy*. From exact results, the optimal switching time was found to be the 40th day. The maximum of function given by (12) gives the optimal switching time of 39th day which is quite close to the exact optimal switching time. Fig. 3 depicts the histories of the healthy cells, infected cells, and the viral population versus time for the case of therapy switching at 40th day. As seen, therapy a has done a good job in controlling virion v_2 and its respected infected cells, but has not been able to do much for the virion v_1, as expected, because of the values assigned to p_{i_j} parameters in the model. Interestingly, the controller has waited until a suitable time to switch to therapy b. The switching time is suitable when the population of the cells infected by v_2, i.e., variable T_2, decreases till some value which switching to therapy b (which is not the ideal therapy for the viral strain type v_2,) does not void the effect of the initial treatment. After the switching, therapy b has controlled virion v_1 and infected cells T_1, such that the infected cells and the virion population are regulated to zero and the healthy cells has started to grow after the initial decay due to the disease.

The biggest advantage of this method, i.e., approximating the function $J_0(x_0, t_1)$ versus different initial condition x_0 and switching time t_1, is being able to find the optimal switching time for *different initial conditions.* However, it should be noted that the initial condition should be such that the resulting trajectory falls in the domain for which the network is trained. To illustrate this capability, another set of initial conditions, denoted with IC 2 is selected. Parameter values for IC 2 are:

$$T(0) = 800\,, \qquad T_1(0) = 10, \qquad T_2(0) = 0, \qquad V_1(0) = 1, \qquad V_2(0) = 0$$

Feeding the new initial condition to *the same trained network*, gives the function

$$J(\tau) = 33.62t_1^4 - 57.56t_1^3 + 45.22t_1^2 - 47.72t_1 + 915.4$$

which is plotted in Fig. 4. The exact performance index versus switching times are calculated through a new set of exhaustive simulations for the new initial condition and the result is depicted in the same figure using dash-dot plots, which is observed to be very close to the approximated value. One can see that for the patient with IC 2, the best prescription is switching to therapy b immediately by skipping therapy a.

As the last simulation, the initial conditions

$$T(0) = 800\,, \qquad T_1(0) = 10, \qquad T_2(0) = 5, \qquad V_1(0) = 1, \qquad V_2(0) = 100$$

denoted with IC 3, are selected. Feeding IC 3 to the neural network gives

$$J(\tau) = -151.2t_1^4 + 470.4t_1^3 - 509.9t_1^2 + 207.1t_1 + 863.6$$

which is plotted in Fig. 5 along with the exact performance calculated using another exhaustive simulation. Again, the results are quite similar and the optimal switching day is found to be the 30th day which is exactly the same as the result obtained using exhaustive simulations.

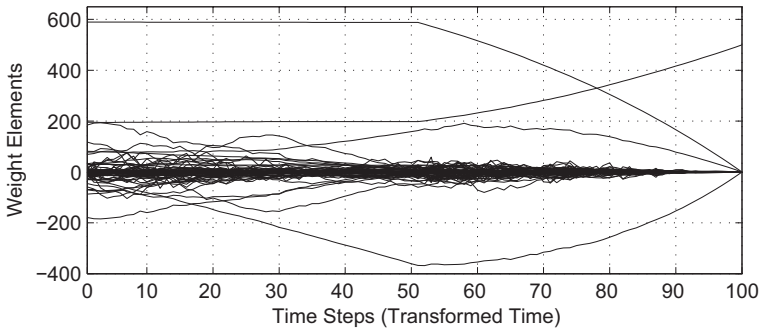

Fig. 1. Weight history versus transformed time.

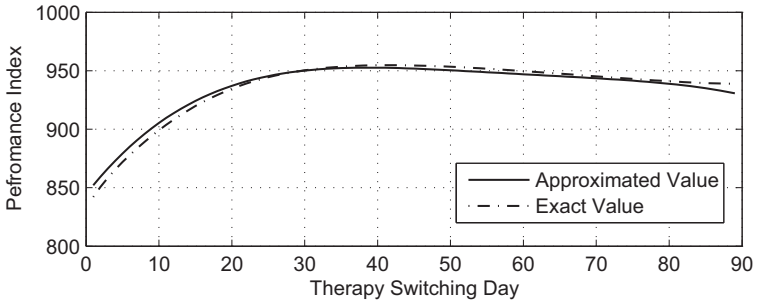

Fig. 2. Performance index versus switching time for IC 1.

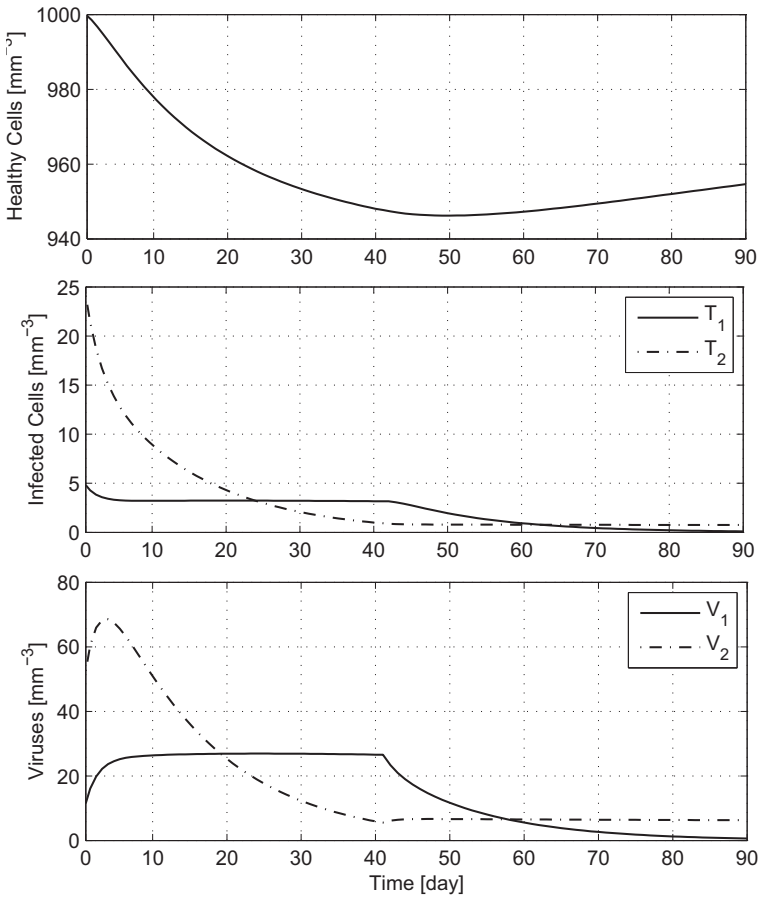

Fig. 3. Evolution of disease parameters for IC 1.

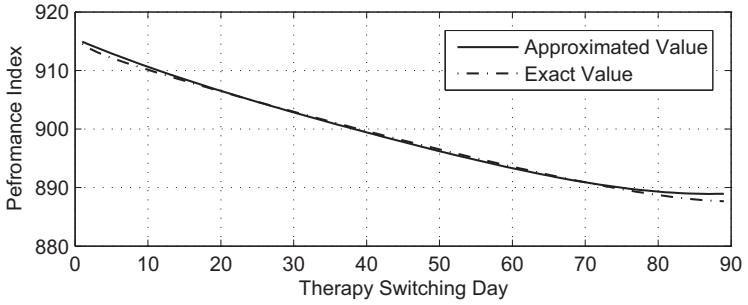

Fig. 4. Performance index versus switching time for IC 2.

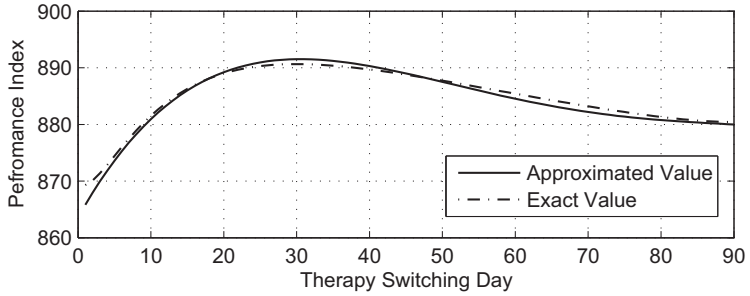

Fig. 5. Performance index versus switching time for IC 3.

In this example, we assumed a fixed mode sequence and selected the maximum number of switching to be one but this technique can easily be generalized. As for the number of switching in a problem, one can define more switching which results in more number of unknowns. Of course, the network basis functions need to be rich enough to capture the performance index function which instead of a curve will be a hyper-plane depending on the number of unknowns.

VI. CONCLUSIONS

A new approach was developed for solving HIV problem. The developed method was applied to an illustrative optimal drug switching in the HIV infection control problem with excellent results. Compared to existing techniques, the developed method is quite versatile in providing optimal switching in real-time for *different initial conditions*. The proposed scheme can also be utilized for determining optimal drug switching times for multi-therapeutic treatment of any other disease, as long as models for the evolution of the disease under each therapy are available.

APPENDIX

In Algorithms 1 and 2, equations (7), (8), (9), and (10) give the weight update rules for the weights. The least squares method can be used for rewriting this equation such that W_k is explicitly given based on the known parameters. In this appendix, the process for finding such an equation for W_k is explained. To perform least squares for the weight update of W_k, n random states and n random switching sequence denoted with $x^{(i)}$ and $\tau^{(i)}$, respectively, where $i \in \{1,2,..,n\}$, are selected. Denoting the right hand side of the equations resulting from each one pair of $x^{(i)}$ and $\tau^{(i)}$ with $\mathcal{Y}\left(x^{(i)},\tau^{(i)}\right)$, the objective is finding W_k such that it solves

$$\begin{cases} W_k^T \phi\left(x^{(1)}, \tau^{(1)}\right) = \mathcal{Y}\left(x^{(1)}, \tau^{(1)}\right) \\ W_k^T \phi\left(x^{(2)}, \tau^{(2)}\right) = \mathcal{Y}\left(x^{(2)}, \tau^{(2)}\right) \\ \vdots \\ W_k^T \phi\left(x^{(n)}, \tau^{(n)}\right) = \mathcal{Y}\left(x^{(n)}, \tau^{(n)}\right) \end{cases} \tag{13}$$

Define

$$\boldsymbol{\phi} \equiv \left[\phi\left(x^{(1)}, \tau^{(1)}\right) \;\; \phi\left(x^{(2)}, \tau^{(2)}\right) \;\; \ldots \;\; \phi\left(x^{(n)}, \tau^{(n)}\right)\right]$$

$$\boldsymbol{\mathcal{Y}} \equiv \left[\mathcal{Y}\left(x^{(1)}, \tau^{(1)}\right) \;\; \mathcal{Y}\left(x^{(2)}, \tau^{(2)}\right) \;\; \ldots \;\; \mathcal{Y}\left(x^{(n)}, \tau^{(n)}\right)\right]$$

Using the method of least squares, solution to the system of linear equations (13) is given by

$$W_k = (\boldsymbol{\phi}\boldsymbol{\phi}^T)^{-1}\boldsymbol{\phi}\boldsymbol{\mathcal{Y}}^T$$

Note that for the inverse of matrix $(\boldsymbol{\phi}\boldsymbol{\phi}^T)$, which is a $m \times m$ matrix, to exist, one needs the basis functions $\phi(.,.)$ to be linearly independent and n to be greater than or equal to the number of the basis functions.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Wang, X., Elaiw, A., Song, X., "Global properties of a delayed HIV infection model with CTL immune response," *Applied Mathematics and Computation*, vol. 218 (18), pp.9405–9414, 2012.

[2] Rivadeneira, P.S., Moog, C.H., "Impulsive control of single-input nonlinear systems with application to HIV dynamics," *Applied Mathematics and Computation*, vol. 218 (17), pp.8462–8474, 2012.

[3] Naresh, R., Tripathi, A., and Sharma, D., "A nonlinear HIV/AIDS model with contact tracing," *Applied Mathematics and Computation*, vol. 217 (23), pp.9575–9591, 2011.

[4] Perelson, A., Nelson, P., "Mathematical analysis of HIV-1 dynamics in vivo," *SIAM Review*, vol. 41, pp. 3–44, 1999.

[5] Vergu, E., Mallet, A., Golmard, J. L., "A modeling approach to the impact of HIV mutations on the immune system," *Computers in Biology and Medicine*, vol. 35, pp. 1–24, 2005.

[6] Xia, X., "Modeling of HIV infection: vaccine readiness, drug effectiveness and therapeutical failures," *Journal of Process Control*, vol. 17, pp. 253–260, 2007.

[7] Hadjiandreou, M., Conejeros, R., Vassiliadis, V., "Towards a long-term model construction for the dynamic simulation of HIV infection," *Mathematical Bioscience and Engineering*, vol. 4, pp. 489–504, 2007.

[8] Ferreira, J., Middleton, R., "A preliminary analysis of HIV infection dynamics," Proc. *Irish Signals and Systems Conference*, Galway, Ireland, 2008.

[9] Kwon, H. D., Lee, J., Yang, S. D., "Optimal control of an age-structured model of HIV infection," *Applied Mathematics and Computation*, vol. 219 (5), pp.2766–2779, 2012.

[10] de Souza, J., Caetano, M., and Yoneyama, T., "Optimal control theory applied to the anti-viral treatment of AIDS" , Proc. *IEEE Conference on Decision and Control,* Sydney, Australia, 2000, pp.4839-4844.

[11] Yadav, V., Balakrishnan, S. N., "Optimal impulse control of systems with control constraints and application to HIV treatment," Proc. *American Control Conference,* Minneapolis, MN, June 2006, pp. 4824-4829.

[12] Kwon, H. D., "Optimal treatment strategies derived from a HIV model with drug-resistant mutants," *Applied Mathematics and Computation*, vol. 188, pp. 1193–1204, 2007.

[13] Banks, H. T.,Kwon, H. D., Toivanen, J. A., Tran, H. T., "A state-dependent Riccati equation-based estimator approach for HIV feedback control, *Optimal Control Application and Methods*, vol. 27, pp. 93–121, 2006.

[14] Hernandez-Vargas, E., Colaneri, P., Middleton, R., and Blanchini, F., "Discrete-time control for switched positive systems with application to mitigating viral escape*," Int. J. Robust and Nonlinear Control*, vol. 21, pp. 1093–1111, 2011.

[15] Martinez-Cajas, J. L., Wainberg, M. A., "Antiretroviral therapy: optimal sequencing of therapy to avoid resistance" *Drugs*, vol. 68, pp. 43–72, 2008.

[16] Soler, M., Olivares, A., and Staffetti, E., "Framework for aircraft trajectory planning toward an efficient air traffic management," *Journal of Aircraft*, vol. 49 (1), pp.985-991, 2012.

[17] Rinehart, M., Dahleh, M., Reed, D., Kolmanovsky, I., "Suboptimal control of switched systems with an application to the DISC engine," *IEEE Transactions on Control Systems Technology*, vol. 16 (2), pp.189-201, 2008.

[18] Benmansour, K., Benalia, A., Djemaï, M., and de Leon, J., "Hybrid control of a multicellular converter," *Nonlinear Analysis: Hybrid Systems*, vol. 1 (1), pp.16-29, 2007.

[19] Hernandez-Vargas, E.A., Middleton, R.H., Colaneri, P., Blanchini, F., "Dynamic optimization algorithms to mitigate HIV escape," Proc *IEEE Conference on Decision and Control*, pp.827-832, 2010.

[20] Gong, Z., Liu, C., Feng, E., Wang, L., Yu, Y., "Modelling and optimization for a switched system in microbial fed-batch culture," *Applied Mathematical Modelling*, vol. 35 (7), pp.3276-3284, 2011.

[21] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions," *International Journal of Control*, vol. 75 (16), pp. 1406-1426, 2002.

[22] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Trans. on Automatic Control*, vol. 49 (1), pp.2- 16, 2004.

[23] Egerstedt, M., Wardi, Y., and Axelsson, H., "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. on Automatic Control*, vol. 51 (1), pp.110-115, 2006.

[24] Axelsson, H., Boccadoro, M., Egerstedt, M., Valigi, P., and Wardia, Y., "Optimal mode-switching for hybrid systems with varying initial states," *Nonlinear Analysis: Hybrid Systems*, vol. 2 (3), pp.765–772, 2008.

[25] Ding, X., Schild, A., Egerstedt M., and Lunze J., "Real-time optimal feedback control of switched autonomous systems," Proc. *IFAC Conference on Analysis and Design of Hybrid Systems*, pp.108-113, 2009.

[26] Kamgarpoura, M., and Tomlin, C., "On optimal control of non-autonomous switched systems with a fixed mode sequence," *Automatica*, vol. 48, pp.1177–1181, 2012.

[27] Zhao, R., and Li, S., "Switched system optimal control based on parameterizations of the control vectors and switching instant," Proc. *Chinese Control and Decision Conference*, pp. 3290-3294, 2011.

[28] Xu, J., and Chen, Q., "Optimal control of switched hybrid systems," Proc. *8th Asian Control Conference*, Kaohsiung, Taiwan, 2011.

[29] Luus, R., and Chen, Y, "Optimal switching control via direct search optimization," *Asian Journal of Control*, Vol. 6 (2), pp. 302-306, 2004.

[30] Rungger, M., and Stursberg, O., "A numerical method for hybrid optimal control based on dynamic programming," *Nonlinear Analysis: Hybrid Systems*, vol. 5 (2), pp.254–274, 2011.

[31] Sakly, M., Sakly, A., Majdoub, N., and Benrejeb, M., "Optimization of switching instants for optimal control of linear switched systems based on genetic algorithms," Proc. *IFAC Int. Conf. Intelligent Control Systems and Signal Processing*; Istanbul, 2009.

[32] Long, R., Fu, J., Zhang, L., "Optimal control of switched system based on neural network optimization," Proc. *Int. Conference on Intelligent Computing*, pp.799-806, 2008.

[33] Zhang, W., Hu, J., and Abate, A., "On the value functions of the discrete-time switched LQR problem," *IEEE Trans. on Automatic Control*, vol. 54 (11), pp.2669-2674, 2009.

[34] Prokhorov, D.V., and Wunsch, D.C. II, "Adaptive critic designs," *IEEE Trans. Neural Networks*, Vol. 8, No. 5, 1997, pp. 997-1007.

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

[36] Ding, J., and Balakrishnan, S. N., "Approximate dynamic programming solutions with a single network adaptive critic for a class of nonlinear systems," *J Control Theory Appl, vol. 9* (3), pp. 370–380, 2011.

[37] Stone, M., and Goldbart, P., *Mathematics for Physics - A Guided Tour for Graduate Students*, Cambridge, England, Cambridge University Press, p. 70, 2009.

5. OPTIMAL SWITCHING AND CONTROL OF NONLINEAR SWITCHING SYSTEMS USING APPROXIMATE DYNAMIC PROGRAMMING

Ali Heydari and S.N. Balakrishnan

ABSTRACT

The problem of optimal switching and control of switching systems with nonlinear subsystems is investigated in this study. An approximate dynamic programming based algorithm is proposed for learning the optimal cost-to-go function based on the switching instants and the initial conditions. The global optimal switching times for every selected initial condition are directly found through minimization of the resulting function. Once the optimal switching times are calculated, the same neurocontroller is used to provide optimal control in a feedback form. Proof of convergence of the learning algorithm is presented. Three illustrative numerical examples are given to demonstrate the versatility and accuracy of the proposed technique.

I. INTRODUCTION

From aerospace field to chemical processes, many examples exist in engineering that can be categorized as switching systems [1-5], in which subsystems with different dynamics exist and at each time instant, one of them is active. Hence, controlling these processes involves not only applying a stabilizing control to the system, but also making decisions on *when* to switch and *what* mode to switch to. Optimal switching and control of a switching system is a challenging problem and some methods have been developed to address the problem [6-18]. The main issue is to find optimal switching instants, and once they are found, the problem reduces to a conventional optimal control problem.

Methods developed so far for finding the optimal switching instants can be mainly divided to two groups: the first group comprises of nonlinear programming based methods [6-13], in which through different schemes, the gradient of the cost with respect to the switching instants/points are calculated and then by using a nonlinear optimization method, e.g., steepest descent, the switching instants/points are adjusted to find the *local* optimum. It should be noted that in many existing papers, the sequence of active subsystems, called mode sequence, is selected a priori [6-12], and the problem reduces to finding the switching instants between the modes. In [13], the first and last subsystems are pre-selected and a depth search is done initially to find all the possible mode

sequences and for every such sequence, the optimal switching instants are calculated using nonlinear programming. Iterative solution to a nonlinear optimization problem is suggested in [10] and using the combination of this control approach with ideas from model predictive control, the authors developed the so-called crawling window optimal control scheme.

The second group includes studies that discretize the problem in order to deal with a *finite* number of options. Having a finite number of candidate switching time sequences, authors of [14] utilize a direct search to evaluate the cost function for different randomly selected switching time sequences and select the best one in the sense of having less corresponding cost. In [15] the discretization of the state and input spaces is used for calculation of the value function for optimal switching through dynamic programming.

In [16] genetic algorithm is used to find the optimal switching times. A neural network (NN) is used for solving the optimal switching problem for a pre-specified initial condition in [17]. A hierarchical decomposition is proposed in [18], with the lower-level being the time-driven dynamics and the higher-level being the event-driven dynamics representing the mode switching.

All the cited methods numerically find the optimal switching time for *a specific initial condition*; each time the initial condition is changed, new computations are needed to find the new optimal switching instants. In order to extend the validity of the results for different initial conditions within a pre-selected set, in [9] the switching parameter is found as the local optimum in the sense that it minimizes the worst possible cost for all trajectories starting in the selected initial states set. Also, the derivative of the switching parameters with respect to the initial conditions is sought through sensitivity analysis.

Note that that switching systems can be classified into externally switched systems (ESS) and internally switched systems (ISS) [7]. An ESS is one in which the switching between the modes is directly controlled and dictated, like changing the gears in a manual transmission car, i.e., in ESS switching is forced. In ISS, however, the switching happens once the states reach a certain condition, called a switching condition, e.g., automatic change of the gears in an automatic transmission in a car. In the other words, ISS is a system with autonomous switching. In the cited papers, [9,10,17,18] deal with ISS, [6,8,11,12,13,14,16] consider ESS, and [7,15] consider both. The authors of [7]

developed their method for ESS to find the desired switching times, and then, they extend their work to ISS through forcing the state to satisfy the switching condition at the dictated switching times.

Recently, the authors of this study proposed a NN based scheme in [20] for optimal switching of systems with *autonomous* dynamics, i.e., where the subsystems do not admit control inputs. Switching with *controlled* subsystem, which is the subject of this study, makes the problem much more complicated due to the inter-coupling between the effect of *switching* to different modes and *inputting* different controls once each mode is active.

Within the last two decades, approximate dynamic programming (ADP) has shown a lot of promise in obtaining solutions to conventional optimal control problems with NN as the enabling structure [21-34]. ADP is usually carried out using a two-network synthesis called adaptive critics (ACs) [22-24]. In the heuristic dynamic programming (HDP) class with ACs, one network, called the 'critic' network, inputs the states to the NN and outputs the optimal cost and another network, called the 'action' network, outputs the control with states of the system as its inputs [24,25]. In the dual heuristic programming (DHP) formulation, while the action network remains the same as the HDP, the critic network outputs the costates with the current states as inputs [22,26,27]. The single network adaptive critics (SNAC) architecture developed in [28] is shown to be able to eliminate the need for the second network and perform DHP using only one network. Similarly, the J-SNAC eliminates the need for the action network in an HDP scheme [29]. Note that the developments in [21-28] are for *infinite-horizon* problems. The use of ADP for solving *finite-horizon* optimal control of conventional problems was considered in [30-34]. Authors of [30] developed a time-varying neurocontroller for solving a problem with state constraints. In [31] a single NN with a single set of weights was proposed which takes the time-to-go as an *input* along with the states and generates the fixed-final-time optimal control for discrete-time nonlinear systems. Finite-horizon problems with *unspecified* terminal times were considered in [32-34].

As for optimal control of *switching* problems, if the function that describes the optimal cost for every given switching time sequence is known explicitly, then the

problem simplifies to minimization of the function with respect to the switching instants. However, even in the case of general linear subsystems with a quadratic cost function, this function is not available [7,19]. The main contribution of this paper is developing an algorithm for ESS-type problems that learns the optimal cost as a function of current state and the switching instants. An ADP based scheme, in an HDP form is used to train an NN to learn the nonlinear mapping between the optimal cost-to-go and the switching instants. Once this function is learned, finding the optimal switching times reduces to minimization of an analytical function. Furthermore, a second NN is trained along with to generate optimal control in a feedback form. Hence, once the optimal switching instants are calculated, one may use the control NN to generate the optimal control to be applied on the system.

As compared to available methods in the literature, the proposed technique has two advantages. They are: 1) the method developed in this paper gives *global* optimal switching instants versus local ones resulting from nonlinear programming based methods, 2) the learned function gives the optimal cost based on the switching instants for *a vast domain of initial conditions*; hence, optimal switching times for different initial conditions can easily be calculated using the *same* trained NNs. Moreover, once the optimal switching instants are calculated, this method provides feedback optimal control, too. Convergence of the learning process is also provided.

The rest of this paper is organized as follows: The problem formulation is given in section II, the proposed solution is described in section III, numerical analysis are given in section IV, and some conclusions are made in section V.

II. PROBLEM FORMULATION

A control-affine switching system can be represented by a set of M subsystems given by

$$\dot{x}(t) = F_{j(t)}\big(x(t)\big) + G_{j(t)}\big(x(t)\big)u(t), \tag{1}$$

where $F_j: \mathbb{R}^n \to \mathbb{R}^n$ and $G_j: \mathbb{R}^n \to \mathbb{R}^{n\times m}$, $\forall j \in \mathcal{J} \equiv \{1, 2, \dots, M\}$. Positive integers n and m denote the dimension of the state vector $x(t)$, and the control vector $u(t)$, respectively. Moreover, switching function $j: [t_0, t_f] \to \mathcal{J}$ returns the index of active subsystem at time $t \in [t_0, t_f]$. The technique developed in this study requires that the

given systems are in a control-affine form. If they are not control-affine, some mathematical construct (for example, defining a new control) is needed to convert the given system to a control-affine system [35]. The controller design process of switching systems includes not only selecting a history of control input $u(t)$, but also a switching function $j(t)$ that allows the operation of the system to switch between different subsystems. Following [7] the switching function can be given by a switching sequence as

$$s = \big((t_0, j_0),\ (t_1, j_1), \dots, (t_K, j_K)\big) \tag{2}$$

where $t_0 \le t_1 \le \cdots \le t_K \le t_f$, $0 \le K < \infty$ and $j_k \in \mathcal{J}$, for $k = 1,2, \dots, K$. In this notation, (t_k, j_k) means that the system switches from subsystem j_{k-1} to j_k at time t_k, and K denotes the number of switching. Considering switching sequence s, switching function $j(t)$ is given by $j(t) = j_k$ where k is such that $t_k \le t < t_{k+1}$, $k \in \{0,1, \dots, K\}$.

The problem of optimal control of switching systems can be defined as determining a switching sequence s (which leads to a switching function $j(t)$) and a control $u(t)$, $t \in [t_0, t_f]$, such that the cost function given below is minimized.

$$J = \psi\left(x(t_f)\right) + \tfrac{1}{2}\int_{t_0}^{t_f}\left(Q\big(x(t)\big) + u(t)^T R u(t)\right) dt \tag{3}$$

Convex positive semi-definite functions $Q: \mathbb{R}^n \to \mathbb{R}$ and $\psi: \mathbb{R}^n \to \mathbb{R}$ penalize the states and positive definite matrix R penalizes the control effort. Following [6-12] in this paper, we freeze the order of the active subsystems, i.e., the mode sequence, and work on finding the optimal switching times. Since the mode sequence is pre-selected, the unknowns in this problem will be the switching time sequence and the optimal control. A switching time sequence is given by

$$\tau = (t_0,\ t_1,\ t_2, \dots,\ t_K) \tag{4}$$

where $t_0 \le t_1 \le \cdots \le t_K \le t_f$ and switching happens at t_ks, $k \in \{0,1, \dots, K\}$. Note that even for linear subsystems with quadratic cost functions, available methods in the literature give only a *local optimal* solution to this problem and only for a *single* set of initial conditions [6,7].

Assumption 1: There exists a piecewise continuous optimal control solution $u^*(t)$ to the problem where the discontinuous points of $u^*(t)$ are limited to the switching times.

Assumption 2: The dynamics of the subsystems do not have finite escape times. Also, the functions representing the dynamics are smooth versus state and control vectors x and u.

III. APPROXIMATE DYNAMIC PROGRAMMING APPROACH

An approximate dynamic programming framework is used in this study as a solution technique to the optimal switching problem. First we motivate the utilization of this approach for conventional optimal control problems and then proceed to using it for switching systems.

A. Adaptive Critics for Conventional Optimal Control

In order to motivate the idea of using ADP for solving the switching problem, in this subsection it is assumed that the switching time sequence is fixed and the unknown is the optimal control $u^*(t)$. This assumption reduces the switching problem to a conventional optimal control with a given cost function, wherein different subsystems are active at different 'given' time periods [7]. In other words, the switching system simplifies to a system with time-varying dynamics. In the HDP scheme [23] with ADP, two NNs named *actor* and *critic* can be trained for approximating the optimal control and the optimal cost-to-go. Extending the idea of HDP to problems with finite-horizon cost function, the optimal control and the optimal cost-to-go are functions of the time-to-go (final time minus the current time) as well as the states [31]. Therefore, the actor and the critic are trained to capture the mapping between a given state and the time-to-go as inputs to the optimal control and the optimal cost-to-go as outputs for the actor and the critic, respectively.

Considering a fixed switching time sequence, switching system (1) simplifies to time-varying system

$$\dot{x}(t) = f(x(t), t) + g(x(t), t)u(t), \tag{5}$$

where $f(x, t) \equiv F_{j(t)}(x)$ and $g(x, t) \equiv G_{j(t)}(x)$. Note functions $f(x, t)$ and $g(x, t)$ are smooth with respect to x, but, they can have discontinuity with respect to t at the fixed

t_ks, due to the switching between the modes. Discretizing the system in (5) by selecting a sampling time Δt results in discrete-time dynamics of the subsystems

$$x_{k+1} = \bar{f}(x_k, k) + \bar{g}(x_k, k)u_k,\ k = 0, 1, 2, \dots, N, \tag{6}$$

where k denotes the time index, $N = (t_f - t_0)/\Delta t$, $x_k = x(k\Delta t + t_0)$, and $\bar{f}(x, k) \equiv x + \Delta t f(x, k\Delta t + t_0)$, $\bar{g}(x, k) \equiv \Delta t g(x, k\Delta t + t_0)$ if forward-in-time Euler integration is used.

Let the cost function be discretized as

$$J = \psi(x_N) + \frac{1}{2}\sum_{k=0}^{N-1}(\bar{Q}(x_k) + u_k^T \bar{R} u_k) \tag{7}$$

where $\bar{Q}(x) \equiv \Delta t Q(x)$, and $\bar{R} \equiv \Delta t R$.

Remark 1: Dynamic Programming [36] which is the back-bone of the method developed here, gives solution to *discrete-time* problems. Therefore, the continuous problem is discretized. Moreover, the assumption that discrete-time system (6) is obtained through discretizing a 'continuous-time' problem is utilized in convergence analysis of the developed algorithm in this paper. Note that almost all physical systems have subsystems with continuous-time dynamics; therefore, this assumption does not impose a limitation on the obtained results for such systems.

Let the optimal control and the optimal cost-to-go be denoted with superscripted '$*$' notation. The optimal cost-to-go at each instant may be denoted with $J_k^*(x_k)$ to emphasize its dependency on the current state, x_k, and on the left time, $N - k$. In other words

$$J_k^*(x_k) \equiv \psi(x_N) + \frac{1}{2}\sum_{k=k}^{N-1}(\bar{Q}(x_k^*) + u_k^* \bar{R} u_k^*). \tag{8}$$

Bellman equation [36] provides optimal solution to the problem of minimizing cost function (7) subject to dynamics (6)

$$J_N^*(x_N) = \psi(x_N), \tag{9}$$

$$J_k^*(x_k) = \frac{1}{2}\left(\bar{Q}(x_k) + {u_k^*}^T \bar{R} u_k^*\right) + J_{k+1}^*(x_{k+1}^*),\ k = 0,1, \dots, N-1, \tag{10}$$

$$u_k^* = -\bar{R}^{-1}\bar{g}(x_k, k)^T \left.\frac{\partial J_{k+1}^*(x_{k+1})}{\partial x_{k+1}}\right|_{x_{k+1}^*},\ k = 0,1, \dots, N-1. \tag{11}$$

where $x_{k+1}^* = \bar{f}(x_k, k) + \bar{g}(x_k, k)u_k^*$. Gradient $\partial J_{k+1}^*(x_{k+1}/\partial x_{k+1})$ is forms as a column vector.

Denoting the approximated optimal cost-to-go and the approximated optimal control with $J_k(x_k)$ and $u_k(x_k)$, respectively, an iterative learning scheme can be derived from the Bellman equation for learning these unknowns for the fixed-final-time problem as [31]

$$J_N(x_N) = \psi(x_N), \tag{12}$$

$$u_k^{i+1}(x_k) = -\bar{R}^{-1}\bar{g}(x_k, k)^T \frac{\partial J_{k+1}^*(x_{k+1})}{\partial x_{k+1}}\Big|_{x_{k+1}^i}, k = 0,1, \dots, N-1, \tag{13}$$

$$J_k(x_k) = \frac{1}{2}\left(\bar{Q}(x_k) + u_k(x_k)^T \bar{R} u_k(x_k)\right) + J_{k+1}(x_{k+1}), k = 0,1, \dots, N-1. \tag{14}$$

Superscript i denotes the index of iteration. Moreover, in (13) one has $x_{k+1}^i \equiv \bar{f}(x_k, k) + \bar{g}(x_k, k)u_k^i(x_k)$, and the converged value of u_k^i is denoted with u_k. Note that in a dual network AC scheme for *finite horizon* optimal control, 'iteration' takes place only in training the actor, as seen in (13). In other words, one starts with an initial guess on u_k^0, $k = 0,1, \dots, N-1$, and iterates using (13). Once the converged control value is obtained, optimal cost-to-go is calculated using (14), without any need for iteration.

By selecting linear in parameter networks, the expressions for the actor (control) and the critic (cost), can be written as

$$u_k(x) = V_k^T \sigma(x), \ k = 0,1, \dots, N-1, \tag{15}$$

$$J_k(x) = W_k^T \phi(x), \ k = 0,1, \dots, N, \tag{16}$$

where $V_k \in \mathbb{R}^{p \times m}$ and $W_k \in \mathbb{R}^q$ are the weights of the actor and the critic networks at time step k, respectively. The linearly independent smooth basis functions are given by $\sigma: \mathbb{R}^n \to \mathbb{R}^p$ and $\phi: \mathbb{R}^n \to \mathbb{R}^q$ for p and q being positive integers denoting the number of neurons. The objective is using Eqs. (12)-(14) in order to determine network weights V_k and W_k, $\forall k$. Considering Eqs. (12)-(14) unknowns $J_k(.)$ and $u_k(.)$ can be calculated in a backward-in-time fashion. In other words, using (12) one can calculate $J_N(.)$. Then, having $J_N(.)$ one can calculate $u_{N-1}(.)$ using the iterations given in (13). Having calculated $J_N(.)$ and $u_{N-1}(.)$, unknown $J_{N-1}(.)$ can be found using (14). Repeating this process from $k = N-1$ to $k = 0$, all the unknowns can be calculated. This idea is used

for training network weights V_k and W_k, $\forall k$, as detailed in Algorithm 1. Note that the superscript on V denote the iteration index and $\nabla\phi(x) \equiv \partial\phi(x)/\partial x$ is formed as a column vector in the algorithm.

Algorithm 1

Step 1: Find W_N such that $W_N^T\phi(x_N) \cong \psi(x_N)$ for different $x_N \in \Omega$ where Ω denotes a compact subset of $\mathbb{R}^n$ representing the domain of interest.

Step 2: For $k = N-1$ to $k = 0$ repeat

{

Step 3: Set $i = 0$ and select a guess on V_k^0.

Step 4: Randomly select n many different state vectors $x_k^{(j)} \in \Omega$, $j \in \{1,2,..,n\}$, for n being a large positive integer.

Step 5: Set $u_k^{(j)} = {V_k^i}^T\sigma\left(x_k^{(j)}\right), j \in \{1,2,..,n\}$.

Step 6: Set $x_{k+1}^{(j)} = \bar{f}\left(x_k^{(j)},k\right) + \bar{g}\left(x_k^{(j)},k\right)u_k^{(j)}, j \in \{1,2,..,n\}$.

Step 7: Find V_k^{i+1} such that

$${V_k^{i+1}}^T\sigma\left(x_k^{(j)}\right) \cong -\bar{R}^{-1}\bar{g}\left(x_k^{(j)},k\right)^T\nabla\phi\left(x_{k+1}^{(j)}\right)^T W_{k+1}, \forall j \in \{1,2,..,n\}.$$

Step 8: Set $i = i+1$ and repeat Step 7, until $\|V_k^{i+1} - V_k^i\|$ converges to a number less than a small preset tolerance.

Step 9: Set $V_k = V_k^i$.

Step 10: Find W_k such that

$$W_k^T\phi\left(x_k^{(j)}\right) \cong \frac{1}{2}\bar{Q}\left(x_k^{(j)}\right) + \frac{1}{2}\sigma\left(x_k^{(j)}\right)^T V_k\bar{R}V_k^T\sigma\left(x_k^{(j)}\right) +$$

$$W_{k+1}^T\phi\left(\bar{f}\left(x_k^{(j)},k\right) + \bar{g}\left(x_k^{(j)},k\right)V_k^T\sigma\left(x_k^{(j)}\right)\right), \forall j \in \{1,2,..,n\}.$$

}

In Steps 1, 7, and 10 of Algorithm 1, the method of Least Squares, explained in Appendix A, can be used for finding the unknown weights in terms of the given parameters.

Remark 2: If the switching time sequence is fixed and given, optimal control of the time-varying system is found by using the respective *active subsystem* at each time index k in

the process of propagation of x_k to x_{k+1}, for use in the weight update equations of W_k and V_k. This step is the main difference between this process and the non-switching adaptive critic based finite-horizon optimal controllers given in [30,31].

Remark 3: Note that the number of iterations may vary for different time steps. The iteration index, i, is reset to zero for the next time-step.

Remark 4: One can modify the algorithm in the sense that at each iteration of Step 7, different random states $x_k^{(j)}$ being selected. For this purpose, one needs to repeat Steps 4 to 7 instead of only repeating Step 7 at each iteration. Note that as long as the number of samples n is large enough, selecting new samples at each iteration of Step 7 is not necessary.

Remark 5: Capability of *uniform approximation* of neural networks [37,38] indicates that once the network is trained for a large enough number of samples from the domain of interest, denoted with n, the network is able to approximate the output for any new sample from the domain with a bounded approximation error. This error bound can be made arbitrarily small if the network is rich enough. For the linear in weight neural network selected in this study and the polynomial basis function utilized in the numerical examples, Weierstrass approximation theorem [39] proves a similar uniform approximation capability.

Remark 6: Considering the possible discontinuity of $\bar{g}(x_k, k)$ at the switching times (due to the switching between G_js,) and the presence of this term in the optimal control equation (11), it is natural to have discontinuity in the optimal control history at the switching times, i.e., at t_ks. On the other hand, the uniform approximation property of NN holds for approximating a 'continuous' function. However, since the NNs are selected with time-varying weights, and the switching times are *fixed* and given, the desired discontinuity can be formed using the discrete set of weights at different ks. To make it clearer, one should note that the control is a function of *two* variables, the time and the state. As long as function $u_k(x)$ for each given k is a continuous function versus x, it can be uniformly approximated using $V_k^T \sigma(x)$ by the given V_k.

B. Adaptive Critics for Switching Optimal Control

In this section, we modify the network structure and the training algorithm to find the optimal switching times along with optimal control. In order to synthesize the ACs

for switching systems, the critic network is trained to approximate the optimal cost-to-go for *different* switching time sequences. It should be made clear that both the optimal cost-to-go and control are functions of the selected switching time sequence, τ.

In this section, an approximation of $J_k^*(x_k, \tau)$ is learned in as a function of x_k and τ and is denoted with $J_k(x_k, \tau)$. Considering time $k = 0$ the approximate optimal cost-to-go $J_0(x_0, \tau)$ for a set of initial conditions x_0 and a switching time sequence τ will be learnt first. Since the NN mapping functions are analytical, once $J_0(x_0, \tau)$ is learned, finding the global optimal τ for a given x_0 reduces to finding the minima of $J_0(x_0, \tau)$ with respect to τ. For example, if the problem involves only one switch, then the only unknown of the optimal switching sequence $\tau = (t_0, t_1)$ is t_1, and finding it is as simple as calculating the roots of the derivative of $J_0(x_0, \tau)$ with respect to t_1 and comparing the value of $J_0(x_0, \tau)$ at those roots along with the value at the boundary points to find the global minimum. As mentioned earlier, even for linear systems with a quadratic cost function, function $J_0(x_0, \tau)$ is not known in a closed form, i.e., with respect to x_0 and τ [7,19].

If the switching times are not fixed, the NNs fail to uniformly approximate $u_k^*(x_k, \tau)$, and hence $J_k^*(x_k, \tau)$, due to possible discontinuity of $u_k^*(x_k, \tau)$ versus k at the *free* switching times. To remedy this problem a transformation is used. The idea is to transform the independent variable t to a new independent variable $\hat{t}$ such that the switching times are fixed in terms of $\hat{t}$ [7]. To motivate the basic idea, assume that the number of subsystems is two with one switching instant at t_1. Define a new independent variable $\hat{t} \in [0\ 2]$ as

$$t = \begin{cases} t_0 + (t_1 - t_0)\hat{t} & if\ 0 \le \hat{t} < 1 \\ t_1 + (t_f - t_1)(\hat{t} - 1) & if\ 1 \le \hat{t} \le 2 \end{cases} \tag{17}$$

one has $t = t_0$ if $\hat{t} = 0$, $t = t_1$ if $\hat{t} = 1$, and $t = t_f$ if $\hat{t} = 2$. Using the new independent variable, $\hat{t}$, the state equations given in (1) can be expressed as

$$x'(\hat{t}) = \begin{cases} (t_1 - t_0)\left(F_1(x(\hat{t})) + G_1(x(\hat{t}))u(\hat{t})\right) & if\ 0 \le \hat{t} < 1 \\ (t_f - t_1)\left(F_2(x(\hat{t})) + G_2(x(\hat{t}))u(\hat{t})\right) & if\ 1 \le \hat{t} \le 2 \end{cases} \tag{18}$$

where the prime notation, x', denotes the derivative of x with respect to $\hat{t}$. Cost function (3) then becomes

$$J = \psi\big(x(2)\big) + \frac{1}{2}(t_1 - t_0)\int_0^1 \Big(Q\big(x(\hat{t})\big) + u(\hat{t})^T R u(\hat{t})\Big)\, d\hat{t} + \frac{1}{2}(t_f - t_1)\int_1^2 \Big(Q\big(x(\hat{t})\big) + u(\hat{t})^T R u(\hat{t})\Big)\, d\hat{t} \tag{19}$$

As can be seen, the benefit of the transformed time is the fact that the switching always happens at the fixed transformed time of $\hat{t} = 1$. Note that, the *actual* switching time is still free and given by t_1. This feature provides the capability to have discontinuities in the history of the weights at $\hat{t} = 1$ to account for a possible discontinuity in control at $t = t_1$, as mentioned in Remark 6. For problems having more switches or more subsystems, e.g. number of switching equal to K, this remedy can be extended where $\hat{t} \in [0 \;\; K+1]$ and the switches happen at $\hat{t} = 1,2,\dots,K$.

After performing the transformation of t to $\hat{t}$, one needs to discretize the resulting transformed dynamics and the transformed cost function (respectively Eqs. (18) and (19) if $K = 1$,) to end up with the discrete-time problem suitable for the ADP scheme. Let the transformed time $\hat{t}$ be discretized to N segments. Note that the initial time and the final time for $\hat{t}$ are 0 and $K+1$, respectively, therefore $N = (K+1)/\Delta\hat{t}$, where $\Delta\hat{t}$ denotes the sampling time for discretizing $\hat{t}$. The discretized dynamics read

$$x_{k+1} = \bar{f}(x_k, \tau, k) + \bar{g}(x_k, \tau, k)u_k, \;\; k = 0, 1, 2, \dots, N, \tag{20}$$

where $\bar{f}(x,\tau,k) \equiv x + \Delta\hat{t}(t_{\hbar+1} - t_{\hbar})F_{j(k\Delta\hat{t}+t_0)}(x)$, $\bar{g}(x,\tau,k) \equiv \Delta\hat{t}(t_{\hbar+1} - t_{\hbar})G_{j(k\Delta\hat{t}+t_0)}(x)$ and $\hbar$ is such that $\hbar \le k\Delta\hat{t} < \hbar + 1$, $\hbar \in \{0,1,\dots,K\}$. Note that functions $\bar{f}(x,\tau,k)$ and $\bar{g}(x,\tau,k)$ depend on the selected τ through the presence of $t_{\hbar}$s in their definition. For example $\bar{f}(x,\tau,k)$ denotes the internal dynamics at state x, which is active at time k given the switching time sequence τ. The discretized cost function will be having time-varying penalizing terms as

$$J = \psi(x_N) + \frac{1}{2}\sum_{k=0}^{N-1}(\bar{Q}(x_k,\tau,k) + u_k^T \bar{R}(\tau,k)u_k) \tag{21}$$

where $\bar{Q}(x,\tau,k) \equiv \Delta\hat{t}(t_{\hbar+1} - t_{\hbar})Q(x)$, and $\bar{R}(\tau,k) \equiv \Delta\hat{t}(t_{\hbar+1} - t_{\hbar})R$ and $\hbar$ is such that $\hbar \le k\Delta\hat{t} < \hbar + 1$, $\hbar \in \{0,1,\dots,K\}$.

An interesting point in minimizing cost function (21) subject to dynamics (20) is the fact that it is just a conventional optimal control problem with time-varying dynamics and cost function terms. The problem however, has free parameters $t_1, t_2, \ldots, t_K$ which form the switching time sequence τ. In other words, the switching time sequence in terms of the transformed time is *given*, but, the switching time sequence in term of the original time is still free and appears as *scaling parameters* $t_1, t_2, \ldots, t_K$ in (20) and (21). The *mapping* between the scaling parameters and the optimal control/cost-to-go can be easily learned using function approximation capability of NN. In the rest of this subsection, the ADP scheme is used for learning the optimal control and the optimal cost-to-go as a function of these scaling parameters. For this purpose, the following modified network structures are proposed:

$$u_k(x,\tau) = V_k^T \sigma(x,\tau), \quad k = 0,1,\ldots,N-1, \tag{22}$$

$$J_k(x,\tau) = W_k^T \phi(x,\tau), \quad k = 0,1,\ldots,N. \tag{23}$$

The inputs to the NNs are the current state and the switching time sequence τ, since the optimal control and optimal cost-to-go are dependent on these two parameters. Let the switching time sequence τ be formed as a K-vector. The new smooth basis functions are $\sigma: \mathbb{R}^n \times \mathbb{R}^K \to \mathbb{R}^p$ and $\phi: \mathbb{R}^n \times \mathbb{R}^K \to \mathbb{R}^q$, where, the switching time sequence is formed as a K-vector whose elements are the switching times.

Using these structures for the actor and critic networks along with dynamics (20) and cost function (21) in the iterative learning scheme (12)-(14) which were derived earlier based on Bellman equation, the weight update laws can now be determined. In this process it should be noted that functions and parameters $\bar{f}(x_k,k)$, $\bar{g}(x_k,k)$, $\bar{Q}(x_k)$, and $\bar{R}$ in (12)-(14) need to be replaced with $\bar{f}(x_k,\tau,k)$, $\bar{g}(x_k,\tau,k)$, $\bar{Q}(x_k,\tau,k)$, and $\bar{R}(\tau,k)$, respectively. Algorithm 2 gives the details of the resulting training/learning process to find V_k and W_k, $\forall k$. In this algorithm, compact domain $\bar{\Omega}$ is defined as $\bar{\Omega} \equiv \{\tau = [t_1, t_2, \ldots, t_K]^T \in \mathbb{R}^K : t_0 \le t_1 \le \cdots \le t_K \le t_f\}$

Algorithm 2

Step 1: Find W_N such that $W_N^T \phi(x_N,\tau) \cong \psi(x_N)$ for different random $x_N \in \Omega$ and different random $\tau \in \bar{\Omega}$.

Step 2: For $k = N-1$ to $k = 0$ repeat

{

Step 3: Set $i = 0$ and select a guess on V_k^0.

Step 4: Randomly select n many different state vectors $x_k^{(j)} \in \Omega$, and n many different switching time sequence $\tau^{(j)} \in \overline{\Omega}$, $j \in \{1,2,..,n\}$, for n being a large positive integer.

Step 5: Set $u_k^{(j)} = {V_k^i}^T \sigma\left(x_k^{(j)}, \tau^{(j)}\right), j \in \{1,2,..,n\}$.

Step 6: Set $x_{k+1}^{(j)} = \bar{f}\left(x_k^{(j)}, \tau^{(j)}, k\right) + \bar{g}\left(x_k^{(j)}, \tau^{(j)}, k\right)u_k^{(j)}$.

Step 7: Find V_k^{i+1} such that

$$
\begin{aligned}
&{V_k^{i+1}}^T \sigma\left(x_k^{(j)}, \tau^{(j)}\right) \cong \\
&-\bar{R}\left(\tau^{(j)}, k\right)^{-1} \bar{g}\left(x_k^{(j)}, \tau^{(j)}, k\right)^T \nabla\phi\left(x_{k+1}^{(j)}, \tau^{(j)}\right)^T W_{k+1} .
\end{aligned} \tag{24}
$$

Step 8: Set $i = i + 1$ and repeat Step 7, until $\|V_k^{i+1} - V_k^i\|$ converges to a number less than a small preset tolerance.

Step 9: Set $V_k = V_k^i$.

Step 10: Find W_k such that

$$
\begin{aligned}
&W_k^T \phi\left(x_k^{(j)}, \tau^{(j)}\right) \cong \frac{1}{2}\bar{Q}\left(x_k^{(j)}, \tau^{(j)}, k\right) + \frac{1}{2}\sigma\left(x_k^{(j)}, \tau^{(j)}\right)^T V_k \bar{R}\left(\tau^{(j)}, k\right) V_k^T \sigma\left(x_k^{(j)}, \tau^{(j)}\right) + \\
&W_{k+1}^T \phi\left(\bar{f}\left(x_k^{(j)}, \tau^{(j)}, k\right) + \bar{g}\left(x_k^{(j)}, \tau^{(j)}, k\right) V_k^T \sigma\left(x_k^{(j)}, \tau^{(j)}\right), \tau^{(j)}\right), \forall j \in \{1,2,..,n\}.
\end{aligned} \tag{25}
$$

}

Remark 7: By comparing Algorithm 2 with Algorithm 1, it can be seen that the main modification is selecting random sets of parameters τ and training the networks to give optimal solution for every given τ.

The converged value, V_k, in Step 7 can be used in Step 10 and a least squares solution can be found for W_k once W_{k+1} and V_k are given, see Appendix A. The following theorem provides the sufficient condition for the convergence of iterative equation (24).

Theorem 1: The iterations given by Step 7 converge for any selected initial guess on V_k^0 for $k = 0,1,\ldots,N-1$, providing the sampling time selected for discretization of the continuous dynamics (1) is small enough.

The proof is given in Appendix B.

In Theorem 1 the role of the sampling time in discretization of a continuous system is emphasized. It is worthwhile to discuss this issue in detail. Substituting (22) and (23) in optimal control equation (11), leads to

$$V_k^T \sigma\left(x_k^{(j)}, \tau^{(j)}\right) \cong -\bar{R}\left(\tau^{(j)}, k\right)^{-1} \bar{g}\left(x_k^{(j)}, \tau^{(j)}, k\right)^T$$

$$\times \nabla\phi\left(\bar{f}\left(x_k^{(j)}, \tau^{(j)}, k\right) + \bar{g}\left(x_k^{(j)}, \tau^{(j)}, k\right) V_k^T \sigma\left(x_k^{(j)}, \tau^{(j)}\right), \tau^{(j)}\right)^T W_{k+1} \qquad (26)$$

which is the same as (24) except that V_k^{i+1} and V_k^i on both sides are replaced with V_k. Optimal weights V_k, $k \in \{0,1,\ldots,N-1\}$ at each time instant k can be calculated by solving the nonlinear equation given in (26), without using the iteration given in (24). However, there is no analytical solution to the set of nonlinear equations (26) in general. Therefore, one needs to resort to numerical methods for solving the set of equations. Theorem 1 proves that for any given smooth dynamics and smooth basis functions, if the sampling time is small enough, the iterations given in (24) converge to the solution to the nonlinear equation (26). However, if the sampling time is fixed, certain conditions on the dynamics or the cost function weight terms need to hold in order for the iterations to converge. These conditions can be easily derived from the proof of Theorem 1.

Once the network weights converge, they solve Bellman Eqs. (10) and (11) and satisfy the final condition (9). Therefore, the resulting cost-to-go and control are *optimal*. In practice, however, due to existence of NN approximation errors, the results will be an approximation of the optimal solution.

When $J_0(x_0, \tau)$ is learned using Algorithm 2, calculating the optimal switching time for any x_0 within the domain of interest reduces to a simple function minimization, i.e., minimizing $J_0(x_0, \tau)$ versus τ. Since the basis functions are smooth, they are of bounded variation [40] and hence, have a finite number of minima in the compact set $\overline{\Omega}$ by definition [40,41]. Therefore, the resulting $J_0(x_0, \tau) = W_0^T \phi(x_0, \tau)$ also will have a finite number of minima. Due to the smoothness of the function, the minima can be calculated using first derivative test, i.e., through finding the stationary points which are the real roots of $\partial J_0(x_0, \tau)/\partial\tau = 0$. Once all of the stationary points are calculated,

comparing the value of the function at the stationary point and at the boundary points, the global minimum can be determined.

Note that, for the result to be global optimum, two other conditions also need to hold, listed below:

1- Eqs. (12)-(14), and hence, Eqs. (24) and (25) provides globally optimal *input-target pairs* for network training.

2- The NN training provides globally optimal *weights* for approximating the mapping between the given input-target pairs.

The proof that condition 1 holds follows from the proof of Theorem 1, once the conditions given in Theorem 1 hold. In other words, once it is proved that (24) is a *contraction mapping*, the uniqueness of fixed point V_k to the iterative Eq. (24) follows [42]. In other words, once the conditions given in Theorem 1 hold, ADP provides *global optimal solution*. Details of this result are beyond the scope of this study and are given in [43]. Condition 2, also, holds as long as the method of least squares is used for calculating the NN weights. Because of the convexity of least squares problems, the concern of getting stuck in a local minimum does not exist.

As compared to [6]-[12], the advantages of the method presented here are twofold: 1) global optimal switching time sequence is obtained rather than local ones, 2) the method provides optimal switching times for *any initial conditions* as long as the resulting state trajectory lies within the domain on which the networks are trained, while the other studies are solutions for a selected initial condition. Note that as seen in this section, the approximated optimal control is given by Eq. (22). Once the network weights are trained and the optimal switching time sequence τ is calculated, Eq. (22) can be utilized for online calculation of the control in a feedback form.

IV. NUMERICAL ANALYSIS

A. Example 1

The first example is a switching nonlinear system with two subsystems and one switch. The subsystem dynamics are

$$\text{Subsystem 1: } \frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}x_2\\x_1^2 - x_2^2\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}u \tag{27}$$

$$\text{Subsystem 2:} \quad \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} u \tag{28}$$

where the elements of the state vector x are denoted by x_1 and x_2. A quadratic cost function in the following form is selected

$$J = x\left(t_f\right)^T S x(t_f) + \frac{1}{2}\int_0^{t_f}\left(x(t)^T Q x(t) + u(t)^T R u(t)\right) dt \tag{29}$$

where $t_f = 3$ sec, $S = diag(10\ \ 10)$, $Q = diag(1\ \ 1)$, $R = 1$, and once the independent variable t is transformed to $\hat{t}$, a sampling time of $\Delta\hat{t} = 0.002$ sec is used for discretization of the continuous dynamics.

Since there is only one switching, the switching time sequence simplifies to finding the switching time t_1. An important step in the design is the selection of the basis functions. Considering Remarks 5 and 6 the basis functions for the actor and critic are selected as polynomials made of different combinations of the two states and the switching time t_1. The selected basis functions for $\phi(.,.)$ are $t_1^a x_1^2$, $t_1^a x_2^2$, $t_1^a x_1 x_2$, $t_1^a x_1^3$, $t_1^a x_2^3$, $t_1^a x_1 x_2^2$, $t_1^a x_1^2 x_2$, $t_1^a x_1^4$, $t_1^a x_1^4$, $t_1^a x_1 x_2^3$, $t_1^a x_1^3 x_2$, $t_1^a x_1^2 x_2^2$ for $a = 0,1,3,\ldots,8$. The selections for $\sigma(.,.)$ are $t_1^a x_1$, $t_1^a x_2$,$t_1^a x_1^2$, $t_1^a x_2^2$, $t_1^a x_1 x_2$, $t_1^a x_1^3$, $t_1^a x_2^3$, $t_1^a x_1 x_2^2$, $t_1^a x_1^2 x_2$ for $a = 0,1,3,\ldots,8$. As with any linear in parameter networks, it is important to select suitable basis functions to allow the NNs to capture the mappings accurately.

The domain of interest for the states is given by $\Omega = \{[x_1\ \ x_2]^T : x_1 \in [-1.25\ \ 1.25],\ x_2 \in [-1.25\ \ 1.25]\}$. Five hundred random states from Ω are used in the least squares process (n=500). The iterative learning of V_k converged after less than five iterations. For initial conditions $x_0 = [-1\ \ 0.5]^T$, function $J_0(x_0, t_1)$ which gives the optimal cost is approximated by $W_0^T \phi(x_0, t_1)$ given in terms of t_1 as

$$J_0(x_0, t_1) = +0.883 - 2.120t_1 + 5.195t_1^2 - 6.873t_1^3 + 5.540t_1^4 - 2.785t_1^5 + 0.85t_1^6 - 0.1437t_1^7 + 0.010t_1^8. \tag{30}$$

Taking the derivative of $J_0(x_0, t_1)$ with respect to t_1 and setting it equal to zero gives three real roots of $t_1 = 0.55, 2.68$, and 2.85. Examining the roots and the boundary points shows that the global minimum of the cost happens at $t_1 = 0.55$ sec with a resulting cost-to-go of 0.53.

In order to evaluate the preciseness of the estimated function $J_0(x_0, t_1)$, the optimal cost-to-go for different preselected switching times is calculated using Algorithm 1 for which a separate pairs of networks are trained for 30 different switching times and the results are given in Fig. 1 denoted by asterisks and compared with the approximated cost-to-go, i.e., function $J_0(x_0, t_1)$. It is seen that the basis functions accurately describe the cost function and therefore, optimal switch times. Once optimal t_1 is found, the actor network can be used for the selected t_1 to give the closed loop optimal control for propagation of the states.

An important feature of the developed method is learning the cost function $J_0(x_0, t_1)$ for different initial conditions in Ω as a part of training the cost network. To test this capability, the trained network is used for calculation of $J_0(x_0, t_1)$ for the new initial condition of $x_0 = [0 \ \ -1]^T$. Optimal cost (denoted by asterisks) and the outputs of the cost network for different t_1s are shown in Fig. 2. The results are quite identical.

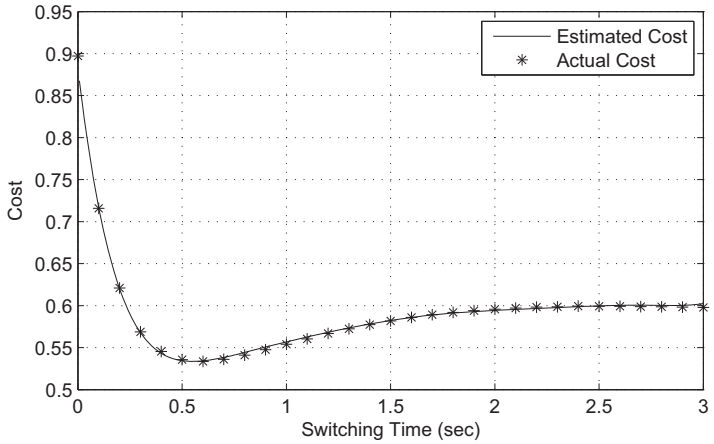

Fig 1: Actual and estimated cost-to-go versus switching times for initial condition of $x_0 = [-1 \ \ 0.5]^T$, Example 1.

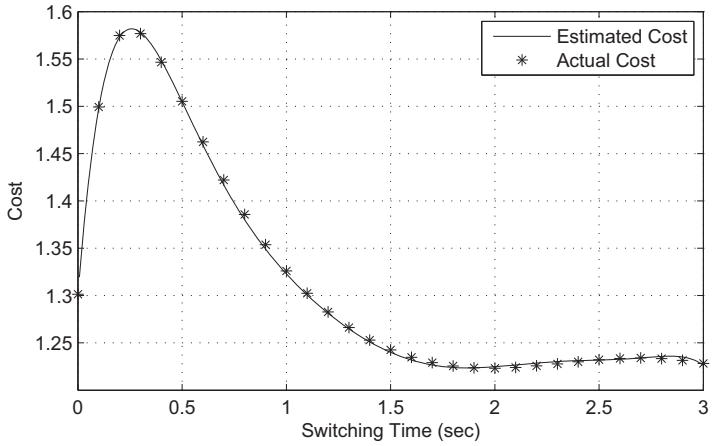

Fig 2: The actual and estimated cost-to-go versus switching times for initial condition of $x_0 = [0 \quad -1]^T$, Example 1.

B. Example 2

A more complex problem is presented next. Example 2 is a switching system with three linear subsystems and two switches. The subsystem dynamics are

$$\text{Subsystem 1: } \frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}0 & 0\\-1 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}u \tag{31}$$

$$\text{Subsystem 2: } \frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}0 & 1\\0 & -1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}0\\0.5\end{bmatrix}u \tag{32}$$

$$\text{Subsystem 3: } \frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}-.1 & 0\\0 & -.1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}0\\0.75\end{bmatrix}u \tag{33}$$

where the elements of the state vector x are denoted by x_1 and x_2. Subsystem 1 is *neither controllable nor stabilizable*, subsystem 2 is controllable, and subsystem 3 *is not controllable but it is stabilizable*. The same quadratic cost function as in (29) is selected where $t_f = 3$ sec, $S = diag(10 \;\; 10)$, $Q = diag(1 \;\; 1)$, $R = 1$, and once the independent variable t is transformed to $\hat{t}$, a sampling time of $\Delta\hat{t} = 0.002$ sec is used for discretization of the transformed continuous dynamics. Note that $0 \leq \hat{t} \leq 3$, because of two switches.

Switching sequence is composed of two switches here, with the mode sequence of (Subsystem 1, Subsystem 2, Subsystem 3), and so the switching time sequence simplifies to two unknowns t_1 and t_2. Basis functions for the actor and critic are selected as polynomials made up of different combinations of the two states and the switching times

t_1 and t_2. Assumed basis functions for $\phi(.,.)$ are $t_1^a x_1^2$, $t_1^a x_2^2$, $t_1^a x_1 x_2$ for $a = 0,1,2,\dots,8$ and also $t_1^a t_2^b x_1^2$, $t_1^a t_2^b x_2^2$, $t_1^a t_2^b x_1 x_2$ for $a = 1,2,3,4$ and $b = 1,2,3,4$ excluding $a = b = 4$. For $\sigma(.,.)$ the selections are $t_1^a x_1$, $t_1^a x_2$ for $a = 0,1,2,\dots,8$ and $t_1^a t_2^b x_1$, $t_1^a t_2^b x_2$ for $a = 1,2,3,4$ and $b = 1,2,3,4$ excluding $a = b = 4$. One could try other types of basis functions too.

Domain of interest for the states in training is the same in Example 1. One thousand random states from Ω are used in the least squares process, i.e., $n = 1000$. As can be observed from the weight histories presented in Fig. 3, weights evolve smoothly except at $\hat{t} = 1$ and $\hat{t} = 2$ where the switches happen. Note that in the weight history of the actor, there are jumps at the switching times; consequently, control values show jumps at the same instances. However, the weight history of the critic shows no jump, as expected, since it reflects an integrated value (see (3)).

The optimal cost $J_0(x_0, t_1, t_2)$ after the weight training for initial condition of $x_0 = [1\ \ 1]^T$ is given by

$$\begin{aligned} J_0(x_0, t_1, t_2) = {} & 4.47 - 13.1t_1 + 9.46t_2 + 50.1t_1^2 - 3.00t_2^2 - 85.9t_1^3 - 6.84t_2^3 + 78.3t_1^4 \\ & + 7.01t_2^4 - 47.4t_1^5 - 2.95t_2^5 + 16.1t_1^6 + .638t_2^6 - 2.95t_1^7 - .07t_2^7 \\ & + .225t_1^8 + .004t_2^8 - 27.0t_1t_2 + 33.3t_1^2t_2 + 15.3t_1t_2^2 - 24.4t_1^2t_2^2 \\ & - 6.68t_1^3t_2 - 2.07t_1t_2^3 + 3.34t_1^3t_2^2 + 5.56t_1^2t_2^3 + 6.57t_1^4t_2 - .149t_1t_2^4 \\ & - 3.04t_1^4t_2^2 - .243t_1^2t_2^4 + .477t_1^4t_2^3 - .184t_1^3t_2^4 \end{aligned}$$

In order to evaluate the accuracy of the learned optimal cost-to-go $J_0(x_0, t_1, t_2)$, optimal costs for different preselected switching times t_1 varying from 0 to 3 with the step size of 0.1 and t_2 varying from t_1 to 3 with the same step size, leading to 496 different switching time sequences, are calculated. The mean of the absolute value of the error between the optimal cost and the approximated cost using $J_0(x_0, t_1, t_2)$ turned out to be 1.1% and the standard deviation of the absolute value of the error was 0.9%. These results show that the proposed technique leads to fairly accurate optimal cost over a wide region.

Fig. 4 depicts $J_0(x_0, t_1, t_2)$ for different t_1 and t_2 where $t_2 \geq t_1$. As seen, $J_0(x_0, t_1, t_2)$ is not convex versus t_1 and t_2. Note that this technique results in a polynomial expression of switching time and therefore, the global optimum can be found using the first derivative test whereas existing methods assume a convex function and

consequently, may end up with a local minimum. In other words, one can calculate all the stationary points of the resulting $J_0(x_0, t_1, t_2)$ and find the one which is the global minimum. For this problem the globally optimal switching times turned out to be given by $t_1 = 1$ sec, and $t_2 = 2.93$ sec.

Fig. 5 shows the state trajectories and the control history from utilizing the actor network and the calculated optimal switching times to control the system. As seen in the control history, there are two jumps in the control at the switching instants. Optimal switching time t_1 being equal to 1 means that the controller prefers to work with the uncontrollable subsystem 1 for the first second and then switches to the controllable subsystem 2. It is interesting to note how optimal switching has exploited the nature of the dynamics of different subsystems. Initial positive values for x_2 in subsystem 2 leads to the growth of x_1, while in subsystem 1 it not only doesn't alter x_1 but positive initial values of x_1 help the controller decrease x_2 without much control effort, hence, the controller has utilized subsystem 1 until it controls x_2 to find some negative value and then switches to subsystem 2 to force x_1 converge to the origin along with x_2. Lack of controllability of subsystem 3 and its slow dynamics has caused the controller to prefer to work with the other two subsystems instead of subsystem 3 for the most of the simulation duration.

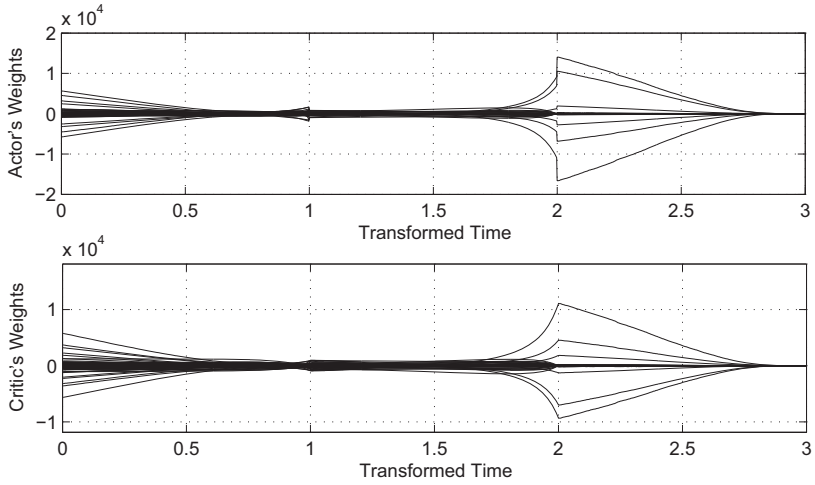

Fig. 3: Weights histories for the actor and critic versus transformed time, Example 2.

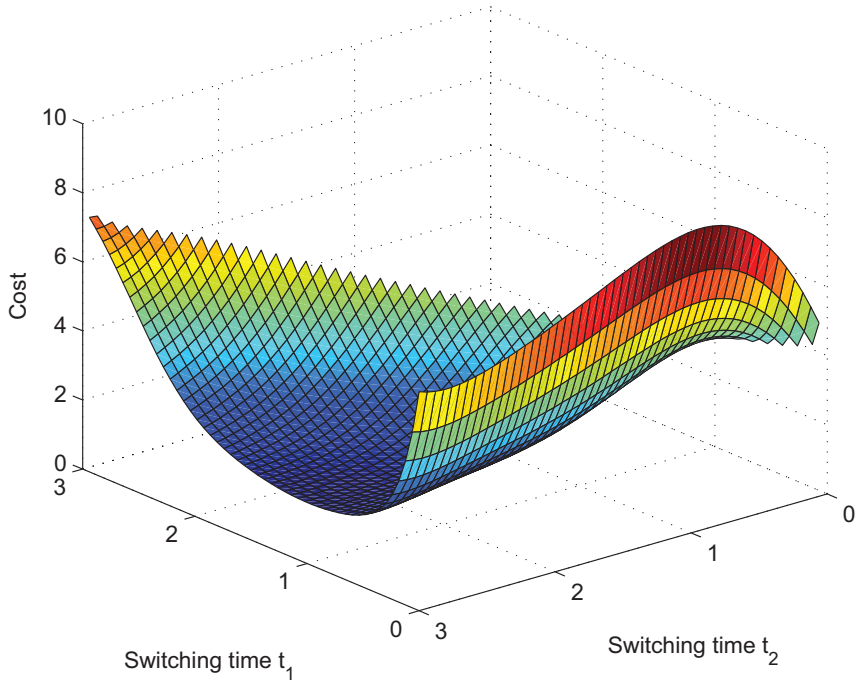

Fig 4: Estimated cost-to-go versus switching times t_1 and t_2 for the selected initial condition, Example 2.

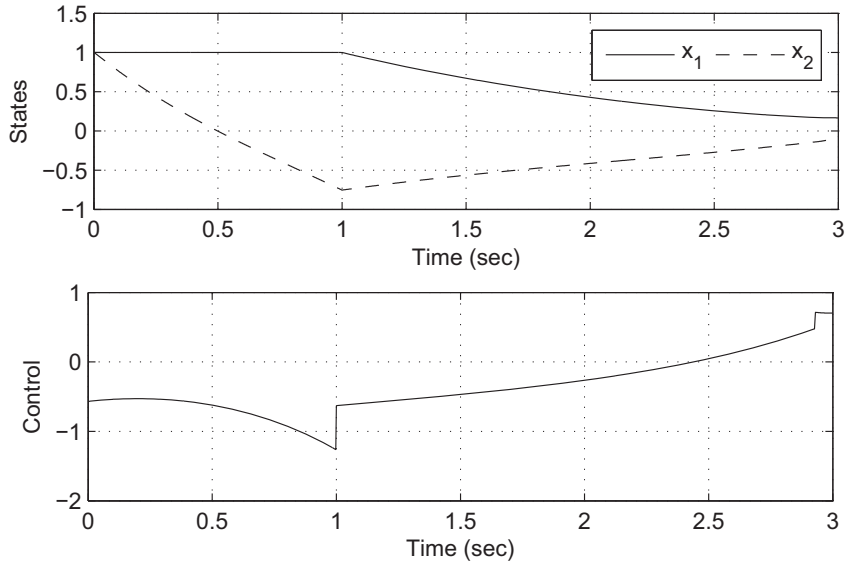

Fig 5: State trajectories and control history for $t_1 = 1$ sec and $t_2 = 2.93$ sec, Example 2.

C. Example 3

As the last example, a switching system selected for simulation analysis in [6,7,12,14,16] is selected to compare the results of the proposed method with the existing ones. The system is a linear problem with one switching and subsystems dynamics of

$$\text{Subsystem 1: } \frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}0.6 & 1.2\\-0.8 & 3.4\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix}u \tag{34}$$

$$\text{Subsystem 2: } \frac{d}{dt}\begin{bmatrix}x_1\\x_2\end{bmatrix} = \begin{bmatrix}4 & 3\\-1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}2\\-1\end{bmatrix}u \tag{35}$$

and the cost function

$$J = \frac{1}{2}(x_1(2) - 4)^2 + (x_2(2) - 2)^2 + \frac{1}{2}\int_0^2 ((x_2(t) - 2)^2 + u^2(t))\, dt \tag{36}$$

A sampling time of $\Delta\hat{t} = 0.001$ sec is used for discretization of the continuous dynamics to apply the method developed in this paper.

The mode sequence is (Subsystem 1, Subsystem 2), hence, the switching time sequence simplifies to one unknown t_1. Basis functions for the actor and critic are selected as polynomials made up of different combinations of the two states and the switching time t_1 up to the seventh order, and for the training, five hundred random states from the domain of interest are used in the least squares process (n=500). Once trained, using the initial condition $x(0) = [0\ \ 2]^T$, the optimal switching time is calculated as $t_1 = 0.1864$ sec. and the optimal cost-to-go turned out to be 9.7792.

Table 1 lists the results of different methods for the same problem and initial condition in order to be able to compare them with the method developed here. As seen in this table, the results from the proposed technique are quite accurate. Note that the method developed here has the advantage of solving the optimal switching problem for

Table 1: Comparison of results for Example 3 using different methods.

Method	Optimal switching time	Optimal cost
Ref. [6]	0.1897	9.7667
Ref. [7]	0.1897	9.7667
Ref. [12]	0.1866	9.7854
Ref. [14]	0.19	9.7686
Ref. [16]	0.1912	10.0035
Our method	0.1864	9.7792

any unspecified different initial condition in the domain of training, while [6,7,12,14,16] gives the optimal switching for a single initial condition only. Finally, Fig. 6 shows the resulting state trajectory and the control history, which are quite similar to the ones given in the cited papers with this example.

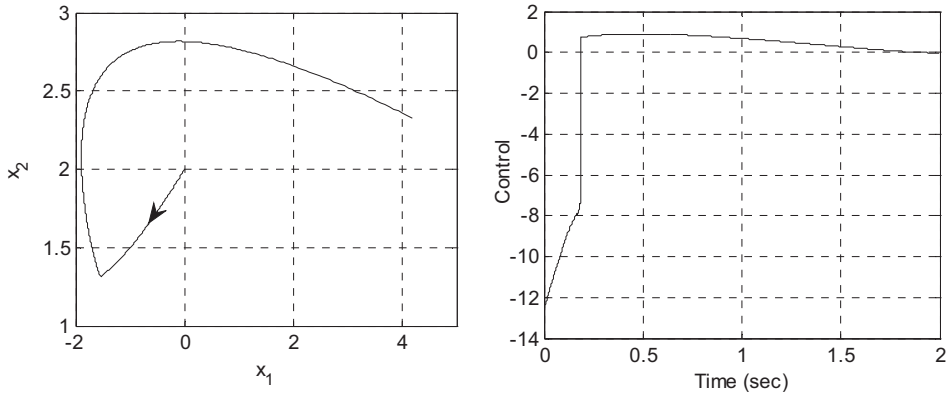

Fig 6: State trajectory and control history for Example 3.

V. CONCLUSIONS

An algorithm was developed for learning the optimal cost-to-go as a function of the current state and the switching time, for switching systems. Convergence of the given iterative weight update law was proved. Numerical analyses showed that the estimated function is accurate for the simulated switching systems with linear and nonlinear subsystems. These results indicate that the proposed method has a lot of potential. The fact that once the networks are trained, global optimal switching times for different initial conditions can easily be obtained makes this method very versatile as compared to the other existing methods in the field of control of switching systems.

APPENDIX A

In Algorithms 1 and 2, in different steps, different weight update rules for the weights of the actor and critic networks, i.e., V_k and W_k, are given. The least squares method can be used for rewriting these equations such that V_k and W_k are explicitly given based on the known parameters. In this appendix, the process for finding such an equation for V_k from Eq. (24) is explained and one can easily find the corresponding equation for W_k.

To perform least squares for the weight update of V_k, n random states and n random switching sequence denoted by $x^{(j)}$ and $\tau^{(j)}$, respectively, where $1 \leq j \leq n$, are selected. Denoting the right hand side of Eq. (24) resulting from each one pair of $x^{(j)}$ and $\tau^{(j)}$ with $Y\left(x^{(j)}, \tau^{(j)}\right)$, the objective is finding V_k such that it solves

$$\begin{cases} V_k^T \sigma\left(x^{(1)}, \tau^{(1)}\right) = Y\left(x^{(1)}, \tau^{(1)}\right) \\ V_k^T \sigma\left(x^{(2)}, \tau^{(2)}\right) = Y\left(x^{(2)}, \tau^{(2)}\right) \\ \vdots \\ V_k^T \sigma\left(x^{(n)}, \tau^{(n)}\right) = Y\left(x^{(n)}, \tau^{(n)}\right) \end{cases} \tag{37}$$

Define

$$\boldsymbol{\sigma} \equiv \left[\sigma\left(x^{(1)}, \tau^{(1)}\right) \;\; \sigma\left(x^{(2)}, \tau^{(2)}\right) \;\; \ldots \;\; \sigma\left(x^{(n)}, \tau^{(n)}\right)\right]$$

$$\boldsymbol{Y} \equiv \left[Y\left(x^{(1)}, \tau^{(1)}\right) \;\; Y\left(x^{(2)}, \tau^{(2)}\right) \;\; \ldots \;\; Y\left(x^{(n)}, \tau^{(n)}\right)\right]$$

Using the method of least squares, solution to the system of linear equations (37) is given by

$$V_k = (\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\boldsymbol{Y}^T \tag{38}$$

Note that for the inverse of matrix $(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)$, which is a $p \times p$ matrix, to exist, one needs the basis functions $\sigma(.,.)$ to be linearly independent and n to be greater than or equal to the number of the basis functions.

APPENDIX B

Proof of Theorem 1: The iteration performed on V_k^i, given in (24) and repeated here, is a successive approximation to find a fixed point of a function

$$\begin{aligned} {V_k^{i+1}}^T \sigma\left(x_k^{(j)}, \tau^{(j)}\right) & \\ \cong -\bar{R}\left(\tau^{(j)}, k\right)^{-1} \bar{g}\left(x_k^{(j)}, \tau^{(j)}, k\right)^T \nabla\phi\left(\bar{f}\left(x_k^{(j)}, \tau^{(j)}, k\right)\right. & \\ \left. + \bar{g}\left(x_k^{(j)}, \tau^{(j)}, k\right) {V_k^i}^T \sigma\left(x_k^{(j)}, \tau^{(j)}\right), \tau^{(j)}\right)^T W_{k+1} & \end{aligned}$$

i.e., there exists function $\mathcal{F}(.)$: $\mathbb{R}^{p\times m} \to \mathbb{R}^{p\times m}$ such that (24) is of form

$$V_k^{i+1} = \mathcal{F}(V_k^i). \tag{39}$$

The claim of the theorem is proved if it can be shown that (39) is a contraction mapping [42]. Since $\mathbb{R}^{p\times m}$ with 2-norm denoted by $\|.\|$ is a Banach space, iterations given by (39), regardless of initial V_k^0, converges to some $V_k^* = \mathcal{F}(V_k^*)$ if there is a $0 \le \rho < 1$ such that for every U_1 and U_2 in $\mathbb{R}^{p\times m}$, the following inequality holds [42]

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \le \rho\|U_1 - U_2\|. \tag{40}$$

Function $\mathcal{F}(.)$ can be formed by converting (24) to a least squares form performed in Appendix A. Rewriting Eq. (38), given in Appendix A, leads to

$$\mathcal{F}\left(V_k^i\right) \equiv (\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma} \times$$

$$\begin{bmatrix} \left(-\bar{R}\left(\tau^{(1)},k\right)^{-1}\bar{g}\left(x_k^{(1)},\tau^{(1)},k\right)^T\nabla\phi\left(\bar{f}\left(x_k^{(1)},\tau^{(1)},k\right)+\bar{g}\left(x_k^{(1)},\tau^{(1)},k\right)V_k^{i^T}\sigma\left(x_k^{(1)},\tau^{(1)}\right),\tau^{(1)}\right)^T W_{k+1}\right)^T \\ \left(-\bar{R}\left(\tau^{(2)},k\right)^{-1}\bar{g}\left(x_k^{(2)},\tau^{(2)},k\right)^T\nabla\phi\left(\bar{f}\left(x_k^{(2)},\tau^{(2)},k\right)+\bar{g}\left(x_k^{(2)},\tau^{(2)},k\right)V_k^{i^T}\sigma\left(x_k^{(2)},\tau^{(2)}\right),\tau^{(2)}\right)^T W_{k+1}\right)^T \\ \vdots \\ \left(-\bar{R}\left(\tau^{(n)},k\right)^{-1}\bar{g}\left(x_k^{(n)},\tau^{(n)},k\right)^T\nabla\phi\left(\bar{f}\left(x_k^{(n)},\tau^{(n)},k\right)+\bar{g}\left(x_k^{(n)},\tau^{(n)},k\right)V_k^{i^T}\sigma\left(x_k^{(n)},\tau^{(n)}\right),\tau^{(n)}\right)^T W_{k+1}\right)^T \end{bmatrix}$$

(41)

One has

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \le \sqrt{n}\|(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\| \times$$

$$\|\bar{R}\left(\tau^{(\ell)},k\right)^{-1}\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)^T\nabla\phi\left(\bar{f}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)+\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)U_1^T\sigma\left(x_k^{(\ell)},\tau^{(\ell)}\right),\tau^{(\ell)}\right)^T W_{k+1} -$$

$$\bar{R}\left(\tau^{(\ell)},k\right)^{-1}\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)^T\nabla\phi\left(\bar{f}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)+\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)U_2^T\sigma\left(x_k^{(\ell)},\tau^{(\ell)}\right),\tau^{(\ell)}\right)^T W_{k+1}\| \tag{42}$$

where $\ell \in \{1,2,\dots,n\}$ is such that

$$\ell = \underset{i\in\{1,2,\dots,n\}}{argmax}\ \|\bar{R}\left(\tau^{(i)},k\right)^{-1}\bar{g}\left(x_k^{(i)},\tau^{(i)},k\right)^T\nabla\phi\left(\bar{f}\left(x_k^{(i)},\tau^{(i)},k\right)+\bar{g}\left(x_k^{(i)},\tau^{(i)},k\right)U_1^T\sigma\left(x_k^{(i)},\tau^{(i)}\right),\tau^{(i)}\right)^T W_{k+1} -$$

$$\bar{R}\left(\tau^{(i)},k\right)^{-1}\bar{g}\left(x_k^{(i)},\tau^{(i)},k\right)^T\nabla\phi\left(\bar{f}\left(x_k^{(i)},\tau^{(i)},k\right)+\bar{g}\left(x_k^{(i)},\tau^{(i)},k\right)U_2^T\sigma\left(x_k^{(i)},\tau^{(i)}\right),\tau^{(i)}\right)^T W_{k+1}\|$$

In inequality (42), the following norm inequality is used

$$\left\|\begin{bmatrix}y_1\\y_2\\\vdots\\y_n\end{bmatrix}\right\|\leq\sqrt{n}\|y_\ell\| \tag{43}$$

where y_is are real-valued row-vectors and $\ell=\underset{i\in\{1,2,\dots,n\}}{argmax}\|y_i\|$.

Smoothness of $\phi(.,.)$ leads to the Lipschitz continuity of $\nabla\phi(.,.)$ on compact set Ω [44]. Therefore, there exists some positive real number ρ_ϕ, independent of τ, such that for every x_1 and x_2 in Ω and $\tau\in\bar{\Omega}$, one has $\|\nabla\phi(x_1,\tau)-\nabla\phi(x_2,\tau)\|\leq\rho_\phi\|x_1-x_2\|$. Using this feature of $\nabla\phi(.,.)$, inequality (42) can be written as

$$\|\mathcal{F}(U_1)-\mathcal{F}(U_2)\|\leq\rho_\phi\sqrt{n}\|(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\|\left\|\bar{R}\left(\tau^{(\ell)},k\right)^{-1}\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)^T\right\|$$
$$\times\|\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)\|\|\sigma\left(x_k^{(\ell)},\tau^{(\ell)}\right)\|\|W_{k+1}\|\|(U_1^T-U_2^T)\| \tag{44}$$

By defining

$$\rho\equiv\rho_\phi\sqrt{n}\|(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\|\left\|\bar{R}\left(\tau^{(\ell)},k\right)^{-1}\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)^T\right\|$$
$$\times\|\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)\|\|\sigma\left(x_k^{(\ell)},\tau^{(\ell)}\right)\|\|W_{k+1}\| \tag{45}$$

one can select the sampling time $\Delta\hat{t}$ in discretization of the continuous dynamics (1) small enough such that the condition $0\leq\rho<1$ is satisfied, since a smaller $\Delta\hat{t}$, directly results in a smaller $\|\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)\|$ while the other terms including $\|\bar{R}\left(\tau^{(\ell)},k\right)^{-1}\bar{g}\left(x_k^{(\ell)},\tau^{(\ell)},k\right)^T\|$ are not affected. Note that smoothness, and hence continuity, of $G_j(.)$s and $\sigma(.,.)$ in their domain result in being bounded in the compact sets Ω and $\bar{\Omega}$ [40], therefore, the $x_k^{(\ell)}$ and $\tau^{(\ell)}$ dependent terms in (45) are upper bounded.

The expression given for the contraction mapping coefficient ρ in (45) involves $\|W_{k+1}\|$ also. It should be noted that W_{k+1} is already learned from the previous step in

the algorithm, therefore, it is bounded. In other words, starting from $k = N - 1$, one uses the successive approximation given by **(24)** and once V_k^i converges, it is used in (25) to calculate the bounded W_k. This process is repeated till $k = 0$.

Note that if the selected sampling time $\Delta\hat{t}$ is not small enough, at some k, $0 \leq k \leq N - 1$, the respective ρ given in (45) does not satisfy condition $0 \leq \rho < 1$, therefore, V_k^i does not converge as $i \to \infty$. In that case, one may select a smaller sampling time and restart the algorithm, i.e., from $k = N - 1$ to calculate the weights corresponding to the smaller sampling time. Refining the sampling time leads to a change in W_{k+1} as well. However, it can be shown that as the sampling time becomes smaller, W_{k+1} remains bounded. This boundedness follows from looking at the definition of W_{k+1}, which is the weights for the network that approximates a discretized cost-to-go. In other words,

$$W_{k+1}^T \phi(x_{k+1}, \tau) \cong \psi(x_N) + \frac{1}{2}\sum_{\hbar=k+1}^{N-1} \bar{Q}(x_\hbar, \tau, \hbar) + u_\hbar^T \bar{R}(\tau, \hbar) u_\hbar. \tag{46}$$

As the sampling times go to zero, the value of the discretized cost-to-go converges to the cost-to-go given by

$$J(x(\bar{t}), \bar{t}) = \psi\left(x(t_f)\right) + \frac{1}{2}\int_{\bar{t}}^{t_f} \left(Q\big(x(t)\big) + u(t)^T R u(t)\right) dt, \tag{47}$$

where $\bar{t}$ is the time corresponding to the transformed time $(k+1)\Delta\hat{t}$. On the other hand, since the system does not have a finite-scape time (Assumption 1,) the finite-horizon cost-to-go will be finite, using any finite control. Note that the control history included in integration (47) correspond to the already converged time-steps, hence, they are bounded. Therefore, as $\Delta\hat{t} \to 0$, the value of $W_{k+1}^T \phi(x_{k+1}, \tau)$ will be finite. Since the basis functions $\phi(x_{k+1}, \tau)$ are linearly independent, a finite $W_{k+1}^T \phi(x_{k+1}, \tau)$ leads to a finite W_{k+1}^T, as seen in the least squares operation described in Appendix A. Therefore, term $\|W_{k+1}\|$ existing in the expression for ρ in (45) remains bounded as the sampling time is refined. This completes the proof of convergence of V_k^i to V_k for $0 \leq k < N - 1$ using any initial guess on V_k^0, for any small enough sampling time. ■

ACKNOWLEDGEMENT

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

REFERENCES

[1] Rinehart, M., Dahleh, M., Reed, D., Kolmanovsky, I., "Suboptimal control of switched systems with an application to the DISC engine," *IEEE Transactions on Control Systems Technology*, vol. 16 (2), pp.189-201, 2008.

[2] Benmansour, K., Benalia, A., Djemaï, M., and de Leon, J., "Hybrid control of a multicellular converter," *Nonlinear Analysis: Hybrid Systems*, vol. 1 (1), pp.16-29, 2007.

[3] Hernandez-Vargas, E.A., Middleton, R.H., Colaneri, P., Blanchini, F., "Dynamic optimization algorithms to mitigate HIV escape," Proc *IEEE Conference on Decision and Control*, pp.827-832, 2010.

[4] Gong, Z., Liu, C., Feng, E., Wang, L., Yu, Y., "Modelling and optimization for a switched system in microbial fed-batch culture," *Applied Mathematical Modelling*, vol. 35 (7), pp.3276-3284, 2011.

[5] Soler, M., Olivares, A., and Staffetti, E., "Framework for aircraft trajectory planning toward an efficient air traffic management," *Journal of Aircraft*, vol. 49 (1), pp.985-991, 2012.

[6] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions," *International Journal of Control*, vol. 75 (16), pp. 1406-1426, 2002.

[7] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Trans. on Automatic Control*, vol. 49 (1), pp.2- 16, 2004.

[8] Egerstedt, M., Wardi, Y., and Axelsson, H., "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. on Automatic Control*, vol. 51 (1), pp.110-115, 2006.

[9] Axelsson, H., Boccadoro, M., Egerstedt, M., Valigi, P., and Wardi, Y., "Optimal mode-switching for hybrid systems with varying initial states," *Nonlinear Analysis: Hybrid Systems*, vol. 2 (3), pp.765–772, 2008.

[10] Ding, X., Schild, A., Egerstedt M., and Lunze J., "Real-time optimal feedback control of switched autonomous systems," Proc. *IFAC Conference on Analysis and Design of Hybrid Systems*, pp.108-113, 2009.

[11] Kamgarpoura, M., and Tomlin, C., "On optimal control of non-autonomous switched systems with a fixed mode sequence," *Automatica*, vol. 48, pp.1177–1181, 2012.

[12] Zhao, R., and Li, S., "Switched system optimal control based on parameterizations of the control vectors and switching instant," Proc. *Chinese Control and Decision Conference*, pp. 3290-3294, 2011.

[13] Xu, J., and Chen, Q., "Optimal control of switched hybrid systems," Proc. *8th Asian Control Conference*, Kaohsiung, Taiwan, 2011.

[14] Luus, R., and Chen, Y, "Optimal switching control via direct search optimization," *Asian Journal of Control*, Vol. 6 (2), pp. 302-306, 2004.

[15] Rungger, M., and Stursberg, O., "A numerical method for hybrid optimal control based on dynamic programming," *Nonlinear Analysis: Hybrid Systems*, vol. 5 (2), pp.254–274, 2011.

[16] Sakly, M., Sakly, A., Majdoub, N., and Benrejeb, M., "Optimization of switching instants for optimal control of linear switched systems based on genetic algorithms," Proc. *IFAC Int. Conf. Intelligent Control Systems and Signal Processing*; Istanbul, 2009.

[17] Long, R., Fu, J., Zhang, L., "Optimal control of switched system based on neural network optimization," Proc. *Int. Conference on Intelligent Computing*, pp.799-806, 2008.

[18] Gokbayrak, K., and Cassandras, C.G., "A hierarchical decomposition method for optimal control of hybrid systems," Proc. *Decision and Control Conference*, vol.2, pp.1816-1821, 2000.

[19] Zhang, W., Hu, J., and Abate, A., "On the value functions of the discrete-time switched LQR problem," *IEEE Trans. on Automatic Control*, vol. 54 (11), pp.2669-2674, 2009.

[20] Heydari, A., and Balakrishnan, S. N., "Optimal multi-therapeutic HIV treatment using a global optimal switching scheme," *Applied Mathematics and Computation*, Vol. 219, pp. 7872-7881, 2013.

[21] Werbos, P.J., "Approximate dynamic programming for real-time control and neural modeling". In White D.A., & Sofge D.A (Eds.), *Handbook of Intelligent Control*, Multiscience Press, 1992.

[22] Balakrishnan, S.N., and Biega, V., "Adaptive-critic based neural networks for aircraft optimal control", *Journal of Guidance, Control & Dynamics*, Vol. 19, No. 4, 1996, pp. 893-898.

[23] Prokhorov, D.V., and Wunsch, D.C. II, "Adaptive critic designs," *IEEE Trans. Neural Networks*, Vol. 8, No. 5, 1997, pp. 997-1007.

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

[25] Dierks, T., Thumati, B. T., and Jagannathan, S., "Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence," *Neural Networks*, vol. 22, pp. 851-860, 2009.

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

[27] Venayagamoorthy, G. K., Harley, R. G., and Wunsch, D. C., "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13 (3), pp. 764-773, 2002.

[28] Padhi, R., Unnikrishnan, N., Wang, X., and Balakrishnan, S. N., "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.

[29] Ding J. and Balakrishnan, S. N., "An online nonlinear optimal controller synthesis for aircraft with model uncertainties," in Proc. *AIAA Guidance, Navigation, and Control Conference*, 2010.

[30] Han, D. and Balakrishnan, S.N., "State-constrained agile missile control with adaptive-critic-based neural networks," *IEEE Trans. Control Systems Technology*, Vol. 10, No. 4, 2002, pp. 481-489.

[31] Heydari, A., and Balakrishnan, S.N., "Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24 (1), 2013, pp. 145-157.

[32] Wang, F., Jin, N., Liu, D., and Wei, Q., "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw*., vol. 22 (1), pp. 24-36, 2011.

[33] Song, R., and Zhang, H., "The finite-horizon optimal control for a class of time-delay affine nonlinear system," *Neural Comput & Applic*, DOI 10.1007/s00521-011-0706-3, 2011.

[34] Wei, Q., and Liu, D., "An iterative ϵ-optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state," *Neural Networks*, vol. 32, pp. 236-244, 2012.

[35] Lane, S. H., and Stengel, R. F., "Flight control design using nonlinear inverse dynamics," Proc. *American Control Conference*, pp. 587-596, 1986.

[36] Kirk, D. E., *Optimal control theory: an introduction*, Dover Publications, 2004.

[37] Homik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," NeuralNetworks, vol. 2, pp. 359-366, 1989.

[38] Attali, J. G. and Pages, G., "Approximations of Functions by a Multilayer Perceptron: a New Approach," *Neural Networks*, vol. 10 (6), pp. 1069-1081, 1997.

[39] Stone, M., and Goldbart, P., *Mathematics for Physics - A Guided Tour for Graduate Students*, Cambridge, England, Cambridge University Press, p. 70, 2009.

[40] Rudin, W. *Principles of Mathematical Analysis*, McGraw-Hill, 1964, p. 118.

[41] Trench, W. F., *Introduction to Real Analysis*, available online at http://ramanujan.math.trinity. edu/wtrench/texts/trench_real_analysis.pdf, 2012, pp. 134,313.

[42] Khalil, H., *Nonlinear Systems*, 3rd ed., Prentice Hall, New Jersey, 2002.

[43] Heydari, A., and Balakrishnan, S.N., "Global Optimality of Approximate Dynamic Programming and its use un Non-Convex Function Minimization," to be submitted to *Automatica*.

[44] Marsden J. E., Ratiu T., and Abraham R., *Manifolds, Tensor Analysis, and Applications*, 3rd ed. Springer-Verlag Publishing Co., New York, 2001, p. 74.

6. OPTIMAL SWITCHING BETWEEN AUTONOMOUS SUBSYSTEMS

Ali Heydari and S. N. Balakrishnan

ABSTRACT

A novel scheme is presented for solving the problem of optimal switching with nonlinear autonomous subsystems. This scheme approximately determines the global optimal solution for different initial conditions in a feedback form. Restrictions, including the need to enforce the mode sequence and/or the number of switching, do not exist for the developed method. Performance is evaluated in several examples with different complexities and the numerical simulation shows great promises for the controller.

I. INTRODUCTION

Optimal scheduling of systems with a switching nature has attracted many researchers during the last decade [1-21]. A switching system is comprised of subsystems with different dynamics which at each time instant only one of them is active. Hence, controlling these systems includes determining both 'when' to switch and 'what mode' to switch to. Systems with such a nature appear in different fields, from trajectory planning to disease therapy [1-5].

The developments in the field of optimal switching can be divided to two main categories: nonlinear programming based developments and discretization based developments. The former utilizes the gradient of the cost with respect to the switching instants to calculate the local optimal switching times [6-13]. In these developments, the sequence of active subsystems, known as mode sequence, is typically selected a priori. The problem is determining the switching instants between the modes. Among the nonlinear programming based methods, some ideas are presented in [13,14] for admitting free mode sequence conditions. In [13] a two stage optimization algorithm was developed which in one stage the switching time is being updated and at another stage the mode sequence is modified. In [14] the process was improved such that a single stage algorithm which solely updates the mode sequence for the selected initial condition is utilized.

Discretization based developments include studies that discretize the switching problem to deal with a *finite* number of options. An optimization scheme was developed in [1] to find both the optimal mode sequence and the switching time for positive linear

systems. A direct search has been utilized in [15] to evaluate the cost function for different randomly selected switching time sequences. The discretization of both the state and input spaces was used to calculate the value function for optimal switching through dynamic programming in [16]. Genetic algorithm and neural networks were used in [17] and [18], respectively, to determine the optimal switching for a preselected initial condition within intelligent methods.

Each of these methods requires a large amount of computations to numerically find the optimal switching time for an a priori selected initial condition. Each time the initial condition is changed, a new set of computations must be performed to find the corresponding optimal switching instants. In [9] the validity of the results was extended for different initial conditions within a pre-selected set. This is done through determining the switching parameter as the local optimum in the sense that it minimizes the worst possible cost for all trajectories starting in the selected set of initial states. A neural network based method for optimal switching was recently proposed in [19] by the authors of this study for problems with *fixed* mode sequence. Once the network is trained based on the algorithm given in [19], the optimal switching scheme for every selected initial condition can be calculated through a static function minimization before the online implementation. Some other researchers have focused on stabilization of switching systems, for example [20,21]. Moreover, interested readers are referred to [22] for a theoretical analysis on the properties of the value function for discrete-time switching systems with linear dynamics and quadratic cost function terms.

The contribution of this work is developing a simple straightforward scheme to solve the optimal switching problem for systems with nonlinear autonomous subsystems. The only control to be determined, in switching problems with autonomous subsystems, is the active mode at each instant. A number of papers, including [1,4,5,8-10,13,14], focus on such problems. The scheme presented in this study is based on the Bellman principle of optimality [23], providing optimal solution in real-time. To this goal, the function representing the nonlinear mapping between the optimal cost-to-go as the output, and the current state and time as the inputs, is required. An algorithm is developed, motivated by studies in adaptive critics (AC) [24-26], to learn this function with a desired degree of accuracy. This function approximation is done through utilizing

a neural network (NN) as a global function approximator and training it using the algorithm. The developed method has the following four advantages differentiating it from methods available in the literature: 1) This method offers approximate global optimal switching instants versus local optimal ones resulting from nonlinear programming based methods. 2) This method does not require enforcing a mode sequence or a number of switching. 3) The solution is calculated in a feedback form. Hence, it will have the relative robustness of feedback controllers toward unmodeled disturbances, compared with open loop solutions. 4) This method offers an optimal solution for a vast domain of initial conditions. Thus, an optimal switching solution for different initial conditions can be readily calculated using the same trained NN.

This article is organized as follows. Problem formulation is presented in section II. The main idea of the proposed method is given in section III. The process of approximating the cost-to-go function is discussed in section IV. Details regarding the implementation of the method for online control are presented in section V. Simulation studies are given in section VI, followed by concluding remarks.

II. PROBLEM FORMULATION

A discrete-time switching system with autonomous subsystems can be represented by a set of M subsystems/modes:

$$x_{k+1} = f_i(x_k), k \in K, i \in I, \tag{1}$$

where $f_i: \mathbb{R}^n \to \mathbb{R}^n$ is a continuous function representing the dynamics of mode i, $K \equiv \{0, 1, 2, \dots, N-1\}$, $I \equiv \{1, 2, \dots, M\}$, and n denotes the dimension of the state vector x_k. Subscript k in x_k denotes the discrete time index, and the final time is denoted by N. Moreover, subscript i in f_i denotes the index of the active subsystem. At each instant k, only one subsystem can be active. A controller for the system is defined as a switching sequence that allows the system to operate, from the initial time $k = 0$ to the final time N. The optimal solution, however, is defined as a switching schedule using which, the performance index given below is optimized.

$$J = \psi(x_N) + \sum_{k=0}^{N-1} Q(x_k) \tag{2}$$

Convex functions $Q: \mathbb{R}^n \to \mathbb{R}$ and $\psi: \mathbb{R}^n \to \mathbb{R}$ correspond to the cost during the time period and at the end, respectively.

Denoting the index of the active subsystem at time k with i_k, the optimal solution may be denoted with $i_k^* \in I$, $\forall k \in K$. Unlike studies that freeze either the mode sequence or the number of switching [6-12], in this work, both the mode sequence and the number of switching are free. The mode sequence, the number of switching, and the switching instant are each subject to be determined such that the cost function is optimized.

III. MAIN IDEA

Denoting the cost-to-go at each time step k and state vector x_k by $J_k(x_k)$ leads to

$$J_k(x_k) = \psi(x_N) + \sum_{j=k}^{N-1} Q(x_j). \tag{3}$$

Note that, from the form of the cost function, it directly follows that

$$J_N(x_N) = \psi(x_N),\ J_k(x_k) = Q(x_k) + J_{k+1}(x_{k+1}),\ \forall k \in K. \tag{4}$$

Based on the Bellman principle of optimality [23], regardless of what decisions are made for the past, the optimal solution is a solution which optimizes the future. Therefore, regardless of values selected for i_j, $j \in \{0,1,\dots,k-1\}$, the optimal solution for the remained time steps, i.e., $j \in \{k, k+1, \dots, N-1\}$ is the solution which optimizes $J_k(x_k)$. From (4), because term $Q(x_k)$ does not depend on the selection of i_j, $j \in \{k, k+1, \dots, N-1\}$, optimizing $J_k(x_k)$ is equivalent of optimizing $J_{k+1}(x_{k+1})$. The main idea of the method in this study is to approximate the optimal cost-to-go $J_k^*(x_k)$ in a closed form (i.e., versus parameters k and x_k). Once this function is available, the optimal solution at each instant k and state vector x_k is given by

$$i_k^*(x_k) = argmin_{i \in I}\, J_{k+1}^*\big(f_i(x_k)\big), \tag{5}$$

hence, $i_k^*: \mathbb{R}^n \to I$.

For example, if the system has two subsystems, finding optimal solution at each instant k simplifies to evaluating the scalar-valued function $J_{k+1}^*\big(f_i(x_k)\big)$ for $i = 1$ and $i = 2$ and selecting the i for which $J_{k+1}^*\big(f_i(x_k)\big)$ is smaller. This calculation needs to be done online at each instant k for $k \in K$. Therefore, the optimal solution will be calculated in real-time and in a feedback form.

The following section provides an algorithm for learning the desired function $J_k^*(x_k)$. Before proceeding to the section, the following assumption is needed to guarantee

the finiteness of the optimal cost-to-go. It should be noted that even though the horizon of the problem subject to this study is finite, subsystems with finite escape time may lead to an infinite cost-to-go.

Assumption 1: The method developed in this study assumes that there exists a switching schedule for which the cost function remains finite.

IV. COST-TO-GO FUNCTION APPROXIMATION

In this section the process of learning the cost-to-go for a system is explained. In order to motivate the idea, initially the case of conventional systems, i.e., non-switching systems, is discussed and an algorithm is proposed for learning the cost-to-go function in a closed-form. Afterward, the algorithm is modified to learn the cost-to-go function for the switching system subject to this study.

A. Cost-to-go Approximation for a Conventional System

Let the dynamics of the system be

$$x_{k+1} = f(x_k), k \in K, \tag{6}$$

where $f: \mathbb{R}^n \to \mathbb{R}^n$ is the sole mode of the system. Note that, the system does not include a control or switching. However, the cost-to-go at each instant k and state vector x_k, i.e., $J_k(x_k)$ can be calculated using cost function (2). The objective is approximating function $J_k(x_k)$ versus k and x_k.

An algorithm is suggested for learning the cost-to-go function. The algorithm trains a NN as a global function approximator for the purpose. The concept is motivated by the notion of AC developments in implementation of Heuristic Dynamic Programming (HDP) [24,25] for infinite-horizon optimal control of conventional systems. In the HDP scheme, the so called critic network learns the optimal cost-to-go, and the so called actor learns the optimal control. In this study, the actor is skipped. The critic is utilized to learn the cost-to-go at each time step k and state vector x_k for the nonlinear system (6). Moreover, the training algorithm is modified in order to admit the finite-horizon cost function (2) according to [26]. Selecting a linear in the parameter NN as the function approximator, the expressions for the critic (cost-to-go approximator) can be written as

$$J_k(x_k) \cong W_k^T \phi(x_k), k \in K \cup \{N\}, \tag{7}$$

where $W_k \in \mathbb{R}^m$ is the unknown optimal weights of the network at time step k. The selected smooth basis functions are given by $\phi: \mathbb{R}^n \to \mathbb{R}^m$, with m being a positive integer denoting the number of neurons. The training process for determining weights W_k, $\forall k$, is detailed in Algorithm 1. In this algorithm, the recurrence equation given by (4) is used to learn the optimal cost-to-go in a backward fashion (i.e., from $k = N$ to $k = 0$).

Algorithm 1

Step 1: Randomly select n different state vectors $x_N^{[j]} \in \Omega, j \in \{1,2,..,n\}$, for n being a large positive integer.

Step 2: Train network weights W_N (see the Appendix) such that

$$W_N^T \phi\left(x_N^{[j]}\right) = \psi(x_N^{[j]}), \forall j \in \{1,2,..,n\}. \tag{8}$$

Step 3: Set $k = N - 1$.

Step 4: Randomly select n different state vectors $x_k^{[j]} \in \Omega, j \in \{1,2,..,n\}$.

Step 5: Train network weight W_k (see the Appendix) such that

$$W_k^T \phi\left(x_k^{[j]}\right) = Q\left(x_k^{[j]}\right) + W_{k+1}^T \phi\left(f\left(x_k^{[j]}\right)\right), \forall j \in \{1,2,..,n\}. \tag{9}$$

Step 6: Set $k = k - 1$. Go back to Step 4 until $k = 0$.

Once the training is done, the cost-to-go function is approximated by $W_k^T \phi(x_k)$ in a closed form, i.e., versus the given k and x_k.

B. Cost-to-go Approximation for a Switching Problem

Considering the cost-to-go approximation method discussed in the foregoing subsection, the same concept may be adapted for approximating the optimal cost-to-go of the switching system (1). Note that once the $i_k^*(x_k)$, $\forall k$, is found for a given initial condition x_0, system (1) simplifies to a conventional problem with a nonlinear time-varying system due to the frozen switching. Therefore, a NN can be used to learn its optimal cost-to-go; the switching nature of the problem does not refrain one from being able to use HDP to approximate the optimal cost-to-go.

Assuming the network structure (7), Algorithm 2 is proposed for learning the optimal cost-to-go function in a closed form.

Algorithm 2

Step 1: Randomly select n different state vectors $x_N^{[j]} \in \Omega, j \in \{1,2,..,n\}$, for n being a large positive integer.

Step 2: Train network weights W_N (see the Appendix) such that

$$W_N^T \phi\left(x_N^{[j]}\right) = \psi(x_N^{[j]}), \forall j \in \{1,2,..,n\}. \tag{10}$$

Step 3: Set $k = N - 1$.

Step 4: Randomly select n different state vectors $x_k^{[j]} \in \Omega, j \in \{1,2,..,n\}$.

Step 5: Calculate

$$i_k^*\left(x_k^{[j]}\right) = argmin_{i \in I}\, W_{k+1}^T \phi\left(f_i\left(x_k^{[j]}\right)\right), j \in \{1,2,..,n\}. \tag{11}$$

Step 6: Train network weight W_k (see the Appendix) such that

$$W_k^T \phi\left(x_k^{[j]}\right) = Q\left(x_k^{[j]}\right) + W_{k+1}^T \phi\left(f_{i_k^*(x_k^{[j]})}\left(x_k^{[j]}\right)\right), \forall j \in \{1,2,..,n\}. \tag{12}$$

Step 7: Set $k = k - 1$. Go back to Step 4 until $k = 0$.

Noting the backward nature of Algorithm 2, i.e., learning W_ks form $k = N$ to $k = 0$, at each instant k, the optimal W_{k+1} is already learned. Therefore, the optimal $i_k^*(x_k)$ can be found using (11). As discussed earlier, having i_j^*, $j \in \{k, k+1, \dots, N-1\}$, the problem simplifies to a conventional problem with time-varying dynamics. Therefore, an adapted version of Algorithm 1 can be used for learning W_j, $j \in \{k, k-1, \dots, 0\}$ as detailed in Algorithm 2.

Assuming the basis functions of the NN are selected rich enough to approximate the cost-to-go function with the desired accuracy, the method developed here provides optimal solution due to its basis on Dynamic Programming [23]. In other words, if $J_k^*(x_k)$ is approximated and available, the optimal mode will always be given through (5). Therefore, an analysis on the approximation capability of the NN is required. It is well known that NNs can provide uniform approximation within the domain of interest providing the function subject to approximation in a continuous function. Interested readers are referred to [27] and [28] for multi-layer NNs and linear in parameter NNs with polynomial basis functions, respectively. Considering Eqs. (8), (9), and (10), the

continuity of the functions subject to approximation, given in the right hand sides of these equations, follows from the convexity of $\psi(.)$ and $Q(.)$ as well as the continuity of $f_i(.)$s and the basis functions. For Eq. (12), however, due to the switching between the modes, i.e., the discontinuous nature of $i_k^*(.)$, the continuity of the right hand side is not obvious. Theorem 1 proves the required continuity.

Theorem 1: If the active mode at each instant k and state vector x is given by

$$i_k^*(x) = argmin_{i\in I}\, W_{k+1}^T \phi\big(f_i(x)\big), \tag{13}$$

then, scalar-valued function $W_{k+1}^T \phi\left(f_{i_k^*(x)}(x)\right)$ is a continuous function versus x at every $x \in \Omega$.

Proof: Let $\bar{x}$ be any point in Ω and set

$$\bar{\imath} = i_k^*(\bar{x}). \tag{14}$$

Select an open set $\alpha \subset \Omega$ such that $\bar{x}$ belongs to the boundary of α and limit

$$\hat{\imath} = \lim_{\substack{\|x-\bar{x}\|\to 0 \\ x\in\alpha}} i_k^*(x) \tag{15}$$

exists. If $\bar{\imath} = \hat{\imath}$, for every such α, then there exists some open set $\beta \subset \Omega$ containing $\bar{x}$ such that $i_k^*(x)$ is constant for all $x \in \beta$, because $i_k^*(x)$ only assumes integer values. In this case the continuity of $W_{k+1}^T \phi\left(f_{i_k^*(x)}(x)\right)$ at $\bar{x}$ follows from the fact that $W_{k+1}^T \phi\big(f_i(x)\big)$ is continuous at $\bar{x}$, for every fixed $i \in I$. Finally, the continuity of the function subject to investigation at every $\bar{x} \in \Omega$, leads to the continuity of the function in Ω.

Now assume $\bar{\imath} \neq \hat{\imath}$, for some α. From the continuity of $W_{k+1}^T \phi\big(f_{\hat{\imath}}(x)\big)$ at $\bar{x}$, for the given $\hat{\imath}$, one has

$$W_{k+1}^T \phi\big(f_{\hat{\imath}}(\bar{x})\big) = \lim_{\delta x\to 0} W_{k+1}^T \phi\big(f_{\hat{\imath}}(\bar{x}+\delta x)\big). \tag{16}$$

If it can be shown that for every selected α, one has

$$W_{k+1}^T \phi\big(f_{\bar{\imath}}(\bar{x})\big) = W_{k+1}^T \phi\big(f_{\hat{\imath}}(\bar{x})\big), \tag{17}$$

then the continuity of $W_{k+1}^T \phi\left(f_{i_k^*(x)}(x)\right)$ at $\bar{x}$ follows, because from (17) and (16) one has

$$W_{k+1}^T \phi\big(f_{\bar{\imath}}(\bar{x})\big) = \lim_{\delta x\to 0} W_{k+1}^T \phi\big(f_{\hat{\imath}}(\bar{x}+\delta x)\big), \tag{18}$$

and (18) leads to the continuity by definition [29]. The proof that (17) holds is done by contradiction. Assume that for some $\bar{x}$ and some α one has

$$W_{k+1}^T \phi\big(f_{\bar{\imath}}(\bar{x})\big) < W_{k+1}^T \phi\big(f_{\hat{\imath}}(\bar{x})\big), \tag{19}$$

then, due to the continuity of $W_{k+1}^T \phi\big(f_{\bar{\imath}}(\bar{x})\big)$ and $W_{k+1}^T \phi\big(f_{\hat{\imath}}(\bar{x})\big)$ at $\bar{x}$, there exists an open set γ containing $\bar{x}$, such that

$$W_{k+1}^T \phi\big(f_{\bar{\imath}}(x)\big) < W_{k+1}^T \phi\big(f_{\hat{\imath}}(x)\big), \forall x \in \gamma. \tag{20}$$

On the other hand, Eq. (15) implies that there exists a neighborhood of $\bar{x}$ at which $\hat{\imath} = i_k^*(x)$, hence, because $\bar{x} \in \gamma$, one has

$$W_{k+1}^T \phi\big(f_{\bar{\imath}}(x)\big) \geq W_{k+1}^T \phi\big(f_{\hat{\imath}}(x)\big), \exists x \in \gamma. \tag{21}$$

But, (21) contradicts (20). Hence, (19) is not possible. The impossibility of

$$W_{k+1}^T \phi\big(f_{\bar{\imath}}(\bar{x})\big) > W_{k+1}^T \phi\big(f_{\hat{\imath}}(\bar{x})\big) \tag{22}$$

directly follows from (14). Because if (22) holds then $\bar{\imath} \neq i_k^*(\bar{x})$, which is against (14). Therefore, equality (17) holds and hence, $W_{k+1}^T \phi\left(f_{i_k^*(x)}(x)\right)$ is continuous at every $\bar{x} \in \Omega$. This completes the proof. ■

The point which leads to the result given in Theorem 1 is the fact that $i_k^*(.)$ is defined by the 'argmin' function given in (13). Even though $i_k^*(x)$ could discontinuously change as x does, function $W_{k+1}^T \phi\left(f_{i_k^*(x)}(x)\right)$ will be continuous at the continuous and discontinuous points of $i_k^*(x)$. In order to better understand this point, one may consider the example of having two subsystems with scalar dynamics. Assume the cost-to-go of utilizing each subsystem, given by $W_{k+1}^T \phi\big(f_i(x)\big)$, $i = 1,2$, changes linearly versus x as given in Fig. 1. In this case, function $W_{k+1}^T \phi\left(f_{i_k^*(x)}(x)\right)$ will be given by the solid plots in the figure. As seen, the jump of $i_k^*(x)$ from one value to another, does not create any discontinuity in $W_{k+1}^T \phi\left(f_{i_k^*(x)}(x)\right)$.

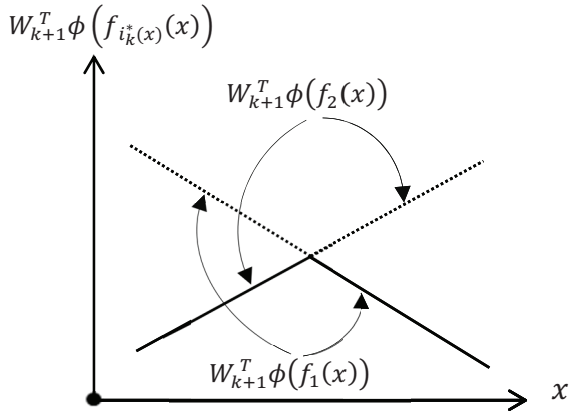

Fig. 1. Symbolic representation of the continuity of $W_{k+1}^T\phi\left(f_{i_k^*(x)}(x)\right)$ at the discontinuous points of $i_k^*(x)$. Solid plots represent function $W_{k+1}^T\phi\left(f_{i_k^*(x)}(x)\right)$ versus x.

V. IMPLEMENTATION AND CONTROL

The trained neurocontroller through Algorithm 2 can be used for online optimal control/scheduling of the system, once the NN weights are trained with Algorithm 2. The control/scheduling is done in real-time by feeding the current state x_k at each time step $k \in K$, to equation (13) to calculate $i_k^*(x_k)$ and applying it on the system. Note that because I has a finite number of elements, the minimization given in (13) simplifies to comparing the values of a scalar-valued function for different $i \in I$ to determine the optimal i. Hence, the online global minimization can be easily conducted. Unlike the nonlinear programming based methods [6-14] which give a local optimum, this method leads to an approximation of the global optimal solution[1]. Moreover, no restriction is enforced on the order of the active subsystems and the number of switching, which is another advantage of this method compared to many of the developments cited in section I. Finally, a great potential of this method is giving the optimal switching for different initial conditions $x_0 \in \Omega$ as long as the resulting state trajectory lies in the domain on

[1] Note that for the resulting i_k to be the *global* optimum, besides finding the global minimum of equation (13) the respective network weights also need to be the global optimal weight. Since NN training methods are susceptible to getting stuck in a local minimum, the method of least squares discussed in the Appendix is recommended for training to end up with the global optimal weights for the NN.

which the network is trained, i.e., $x_k \in \Omega, \forall k$. The reason is the cost-to-go approximation is valid when the state belongs to Ω, therefore, one can always use (13) for finding the optimal mode, as long as $x_k \in \Omega$. Note that, except [19], the cited methods in the literature calculate the optimal switching only for a pre-specified initial condition.

Looking at Eq. (13), ('the decision maker' for switching,) one may observe high frequency switching between the modes, in some problems. This behavior is observed in Example 3 included in this study. The following two alternative remedies are suggested to avoid high frequency switching:

1- The Minimum Dwell Time Remedy: Dictating a minimum dwell time after each switching can eliminate high frequency switching. Once the optimal subsystem is determined at $k = 0$ it will be applied. Afterward, throughout the horizon, one can dictate a minimum dwell time before switching to another mode. That is, once a switching occurs, one may skip the evaluation of Eq. (13) and instead stay with the current active subsystem until the minimum dwell time is passed.

2- The Threshold Remedy: This method allows selecting a positive real number as the threshold. When switching to another mode gives a reward (in the sense of less cost-to-go) more than the selected threshold, the switching is applied. Otherwise, the current active subsystem remains active. To be more specific, assume the active subsystem is i at the time instant k, and by evaluating Eq. (13) one realizes that switching to subsystem j leads to the cost-to-go less than the cost-to-go of staying with subsystem i. In such a case switching to subsystem j is allowed only if

$$W_{k+1}^T \phi\left(f_j(x_k)\right) < W_{k+1}^T \phi\left(f_i(x_k)\right) + \tau,$$

where the pre-selected threshold is denoted with τ.

Note that the same algorithm (Algorithm 2) may still be used in the offline training stage of the NN. The above alternative remedies, however, can be used in the online control. The alterations dictated by the remedies result in a 'sub-optimal' control of the system. The result will remain sub-optimal because the neurocontroller calculates the optimal solution in a feedback form and in real-time. More specifically, the perturbation due to the applied remedy can be considered as a disturbance for the controller. Providing suitable selection of the minimum dwell time or the threshold, the

feedback nature of the controller can deal with the resulting disturbance without too much performance degradation. This behavior is due to the inherent nature of feedback controllers in moderate disturbance rejection.

VI. NUMERICAL ANALYSIS

Two different examples are presented in this section to evaluate the proposed switching scheme. Before going through them, a preliminary example is discussed to evaluate Algorithm 1 for cost-to-go approximation of non-switching systems.

A. Example 1

As the first example, the performance of Algorithm 1 in approximating the cost-to-go of a non-switching system is investigated. The selected systems is the nonlinear scalar system

$$\dot{x} = f\big(x(t)\big) \equiv -x^3(t),$$

with the cost function

$$J = \int_0^5 10x^2 dt.$$

Note that the method developed in this study admits discrete-time dynamics and cost function. Hence, one needs to discretize the abovementioned system. For this purpose, Euler integration scheme with the sampling time of $0.05\ s$ is used. Hence, $N = 100$.

In order to conduct Algorithm 1, an important step in the design process is the proper selection of the basis functions. The well-known Weierstrass approximation theorem [28] proves that any continuous function on a closed and bounded interval can be uniformly approximated on that interval by polynomials to any degree of accuracy. In this example, the role of the richness of the basis functions is investigated. For this purpose, different sets of basis functions were selected and each one was used separately for learning the cost-to-go using Algorithm 1. The selected basis functions for each NN were polynomials x^{2j}, where $j \in \{1, \dots, p\}$. Positive integer p relates to the highest order of the included polynomials in the respective NN. It is selected as $p = 1,2,4,$ and 8 for the selected four NNs. The NNs were trained separately and used for approximating the cost-to-go for different initial conditions $x_0 \in [-2\ 2]$. The *actual* cost-to-go for each selected initial condition is calculated by propagating the states and calculating the summation representing the cost-to-go. The results are summarized in Fig. 2. As seen in

this figure, the basis functions represented by $p = 2$ had a poor performance in approximating the cost-to-go. As the order of the incorporated polynomials grows, the approximation becomes more accurate. For example, the basis functions generated using $p = 8$, which include eights polynomial functions or neurons, is shown to be able to approximate the cost-to-go with a suitable accuracy.

Considering these results, it should be noted that the richness of the basis functions plays an important role in the function approximation, and hence, in the performance of the method developed here for switching. If the basis functions are poor, the result will not be reliable. Therefore, the designer needs to utilize different sets of basis functions and compare the performance of the respectively trained networks to choose the best one.

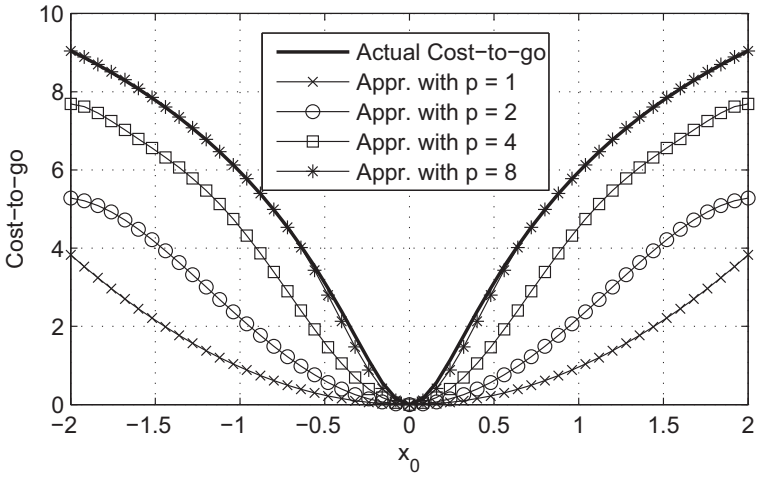

Fig. 2. Cost-to-go for different initial conditions, Example 1.

B. Example 2

Consider the continuous-time scalar problem with two modes,

$$\dot{x} = f_1(x(t)) \equiv -x(t),\ \dot{x} = f_2(x(t)) \equiv -x^3(t), \tag{23}$$

where $t \in [0, 5]$ denotes the time in seconds. The selected cost function is $J = 500x^2(5)$. For discretization, Euler integration, along with the sampling time of $0.05\ s$ was selected which leads to $N = 100$. Considering the subsystems' dynamics, both are stable. Comparing the derivative of the state, however, subsystem 1 has a faster convergence

rate for $x \in (-1,1)$. If $|x| > 1$, subsystem 2 will be the optimal choice. Therefore, the optimal solution is known as

$$i_k^* = \begin{cases} 1 & if\ x_k \in (-1,1) \\ 2 & if\ |x_k| > 1 \end{cases}. \tag{24}$$

The basis functions were selected as polynomials x^{2j}, where $j \in \{1,2,\dots,5\}$. As seen in Example 1 the accuracy of the approximation can be adjusted by the selection of the order of the polynomials. In this example, the selected basis functions were observed to provide the desired accuracy. For the training process, at each time step, 500 random states were selected from $\Omega = [-2\ 2]$ to train the network in a batch training scheme. The training was conducted by solving the least squares detailed in the Appendix. The resulting weight histories for the NN are plotted in Fig. 3. The time-dependency of the weights represents the time-dependency of the cost-to-go.

Having trained the network, initial condition $x_0 = 2$ is simulated using the developed method. The results are given in Fig. 4. Comparing both the resulting switching instant and the mode sequence with the optimal solution (24), reveals that the method has provided the optimal solution.

The controller is able to solve the optimal control problem for a vast variety of initial conditions. More specifically, the same trained network can control different initial conditions as long as the resulting state trajectory lies in Ω. From the dynamics of the subsystem it can be seen that selecting any $x_0 \in \Omega$ will produce $x_k \in \Omega$, $\forall k$. Therefore, the trained network can optimally control any initial condition $x_0 \in \Omega$. The initial conditions $x_0 = -1.5$ and $x_0 = 1$ are selected as the second and third simulations. The results are presented in Figs. 5 and 6, respectively. As for $x_0 = -1.5$, this method has provided the optimal solution by picking the right subsystem at the beginning of the simulation, switching at the right time, and switching to the right mode (see Fig. 5). Figure 6 illustrates the performance of the method through selecting Subsystem 1 and not switching at all, which is the optimal solution for initial condition $x_0 = 1$.

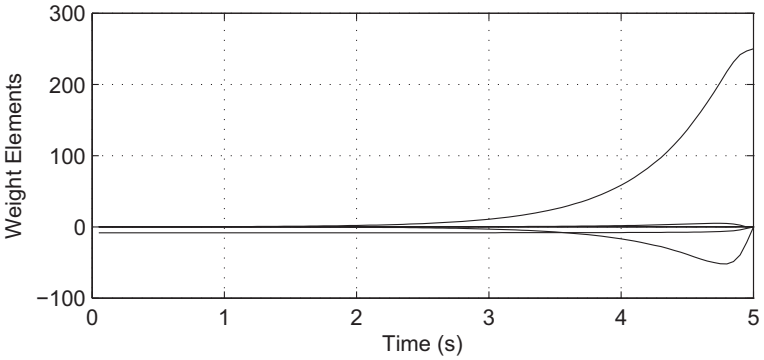

Fig. 3. Weight history of the NN, Example 2.

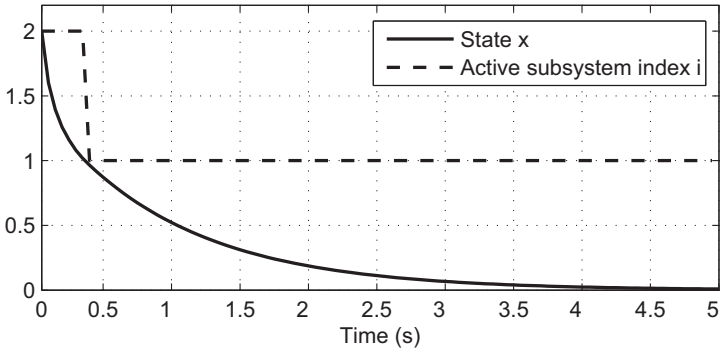

Fig. 4. Simulation results of Example 2 for $x_0 = 2$.

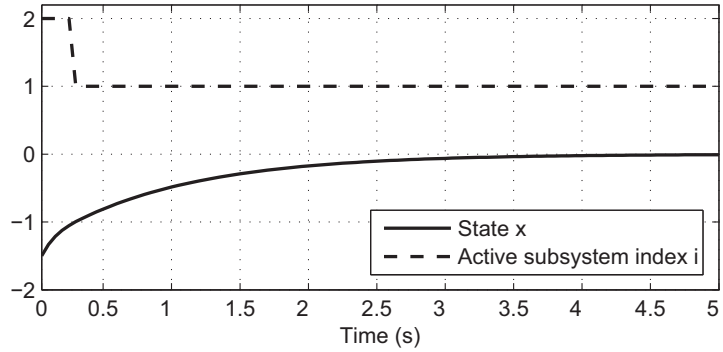

Fig. 5. Simulation results of Example 2 for $x_0 = -1.5$.

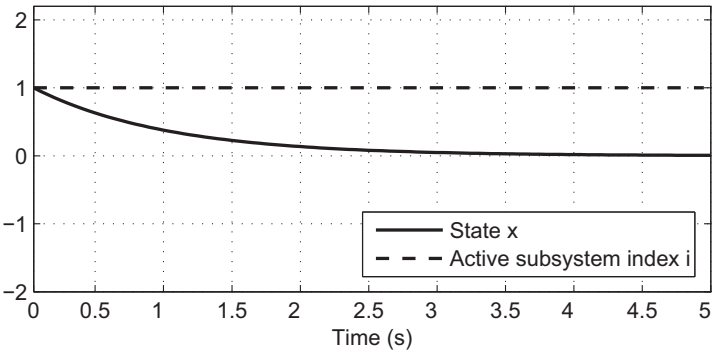

Fig. 6. Simulation results of Example 2 for $x_0 = 1$.

C. Example 3

The second order system with three modes, presented in [13,30], is simulated as the third example. The objective of this problem is controlling the fluid level in a two-tank setup. The fluid flow into the 'upper tank' can be adjusted through a valve which has three positions: fully open, half open, and fully closed. Each tank leaks fluid with a rate proportional to the square root of the height of the fluid in the respective tank. The upper tank leaks into the lower tank, and the lower tank leaks to the outside of the setup. Representing the fluid height in the upper tank with x_1 and the lower tank with x_2, the dynamics of the state vector $x = [x_1, x_2]^T$ are given by the following three modes

$$\dot{x} = f_1(x) \equiv \begin{bmatrix} -\sqrt{x_1} \\ \sqrt{x_1} - \sqrt{x_2} \end{bmatrix},\ \dot{x} = f_2(x) \equiv \begin{bmatrix} -\sqrt{x_1} + 0.5 \\ \sqrt{x_1} - \sqrt{x_2} \end{bmatrix},\ \dot{x} = f_3(x) \equiv \begin{bmatrix} -\sqrt{x_1} + 1 \\ \sqrt{x_1} - \sqrt{x_2} \end{bmatrix}. \quad (25)$$

The objective is forcing the fluid level in the lower tank (i.e., x_2) to track constant value 0.5. For this purpose, cost function $J = 10\int_0^5 (x_2 - 0.5)dt$ was used in [13] with a final time of $5\ s$. The control is the position of the valve and can assume one of the three discrete values 0, 0.5, and 1. Each of these values leads to one of the modes listed above. The basis functions for this example were selected as polynomials $x_1^j x_2^l$, where non-negative integers j and l are such that $j + l \leq 8$. This selection led to 45 neurons ($m = 45$). The problem was discretized using sampling time of $0.01\ s$. Domain $\Omega = \{x \in \mathbb{R}^2 : 0 \leq x_i \leq 1, i = 1,2\}$ was used for the training.

Once the network was trained, initial condition $IC1 \equiv [0.8, 0.2]^T$, simulated in [13], was used to determine the optimal solution. The results are given in Fig. 7. The

method did an excellent job controlling the fluid level of the lower tank by tracking the desired value. Comparing the result with the result reported in [13] for the proposed nonlinear programming based method, the tracking is done much more accurately using the method developed in this study. The cost-to-go for the method given in [13] is 0.25, while for our method it turned out to be 0.245. The lower cost-to-go represents the better approximation of the optimal solution. This perfect tracking was achievable, however, through high frequency switching between the three modes, as seen in Fig. 7. The minimum dwell time of $0.1\,s$ was selected and applied, according to the remedies suggested in Section V for avoiding high frequency switching. Both the resulting state trajectory and the switching are presented in Fig. 8. The alternative remedy for high frequency switching presented as the threshold remedy was simulated as well. The threshold was selected at 2×10^{-4}. The results are given in Fig. 9. Considering Figs. 8 and 9, the state continues to closely track the desired value with far less switching. Using either a longer minimum dwell time or a greater threshold, however, resulted in less precise tracking. These results were expected. The cost-to-go resulting from applying these remedies turned out to be around 0.246 for both remedies. As seen, it is still less than the cost-to-go resulting from the method developed in [13].

Finally, a new initial condition, namely $IC2 = [0,0]^T$, was simulated using the same trained network. Considering the dynamics of the three modes, it can be observed that as long as the initial condition belongs to Ω, regardless of what switching is applied, the states will always stay in Ω. Therefore, the trained network should produce an optimal solution for any initial condition in Ω. The simulation results for $IC2$ are given in Fig. 10. This figure demonstrates the capability of the method in producing an optimal solution for different initial conditions, as opposed to the other methods, including [13] whose solution depends on a single initial condition.

VII. CONCLUSIONS

A new scheme was developed for optimal switching between autonomous modes. This method is shown to provide optimal switching schedule without needing to enforce either a mode sequence or a number of switching. It is observed that the neurocontroller has attractive features that include providing optimal feedback solution for a vast variety of initial conditions. These capabilities were illustrated through numerical analyses. The

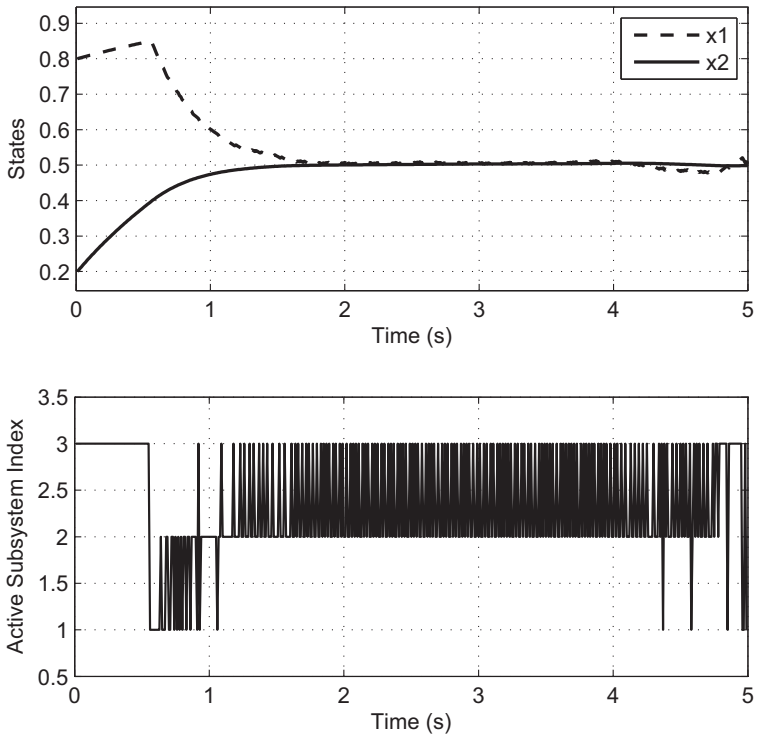

Fig. 7. Simulation results of Example 3 for $IC1$.

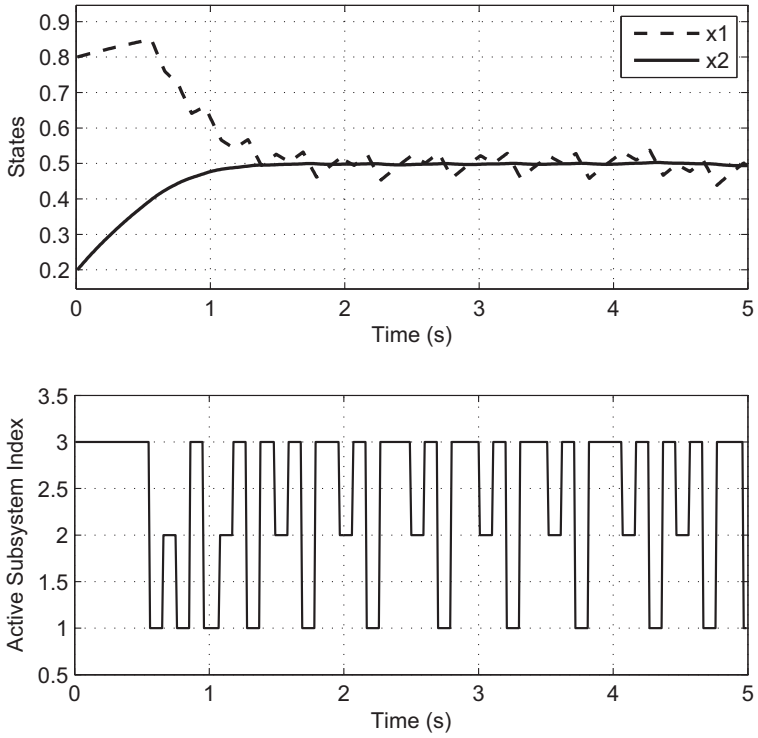

Fig. 8. Simulation results of Example 3 for $IC1$ with applied minimum dwell time.

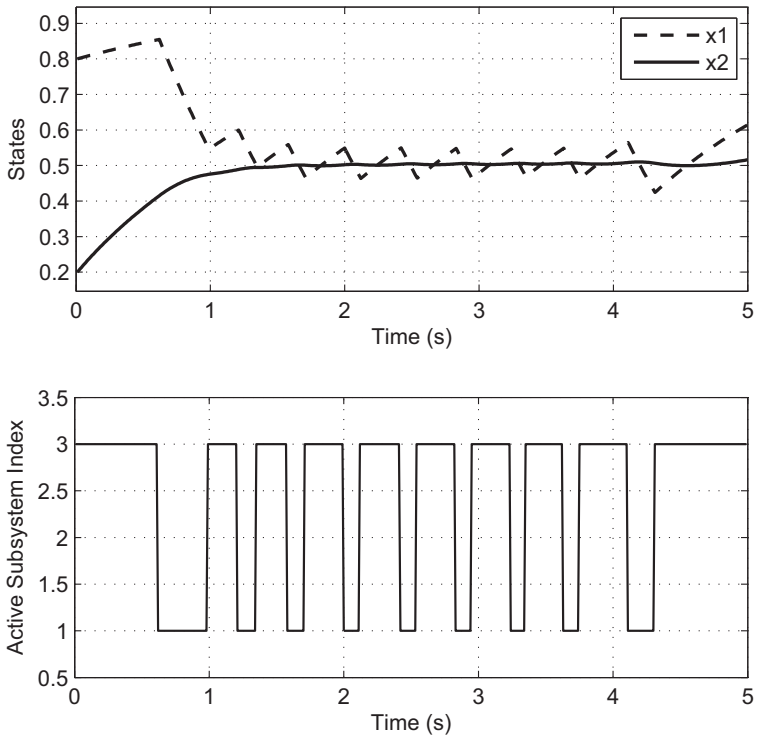

Fig. 9. Simulation results of Example 3 for $IC1$ with applied threshold.

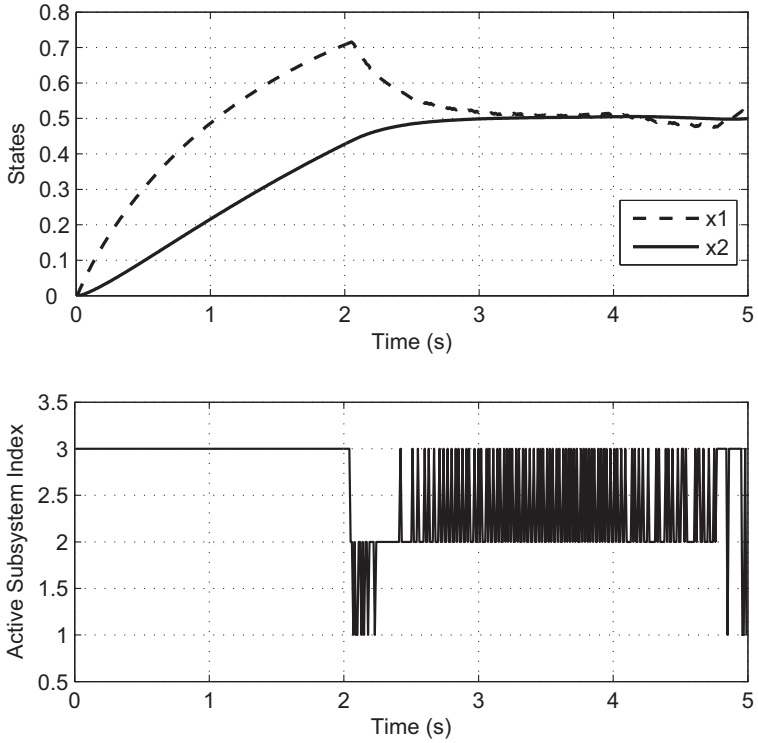

Fig. 10. Simulation results of Example 3 for $IC2$.

developed method can be utilized in different real world applications for real-time scheduling and control of switching systems.

ACKNOWLEDGEMENT

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

APPENDIX

In steps 2 and 5 of Algorithm 1 (or steps 2 and 6 of Algorithm 2) one may use the method of least squares for finding the unknown optimal W_k in a batch training scheme. For this purpose, n random states, denoted with $x^{[j]}$, $j \in \{1,2,..,n\}$, need to be selected to perform the least squares. Denoting the right hand side of equations (8), (9), (10), or (12) resulting from each $x^{[j]}$ with $Y(x^{[j]})$, the objective is finding W_k such that it solves

$$\begin{cases} W_k^T \phi(x^{[1]}) = Y(x^{[1]}) \\ \vdots \\ W_k^T \phi(x^{[n]}) = Y(x^{[n]}) \end{cases} \tag{26}$$

Define both $\boldsymbol{\phi} \equiv [\phi(x^{[1]}), \phi(x^{[2]}), \dots, \phi(x^{[n]})]$ and $\boldsymbol{Y} \equiv [Y(x^{[1]}), Y(x^{[2]}), \dots, Y(x^{[n]})]$. Using the method of least squares, the solution to the system of linear equations (26) is

$$W_k = (\boldsymbol{\phi}\boldsymbol{\phi}^T)^{-1}\boldsymbol{\phi}\boldsymbol{Y}^T. \tag{27}$$

Note that for the inverse of matrix $(\boldsymbol{\phi}\boldsymbol{\phi}^T)$, which is an $m \times m$ matrix, to exist, one needs the basis functions $\phi(.)$ to be linearly independent and $n \geq m$. The process of calculating W_k can be done in one shot. In this case, in order for the resulting W_k to be valid for the whole domain Ω, it is required to select a very large n and randomly select $x_k^{[j]}$s within Ω such that they represent the whole domain. Note that due to the convexity of least squares problems, this method leads to global optimal W_k and the issue of getting stuck in a local minimum does not exist.

REFERENCES

[1] Hernandez-Vargas, E., Colaneri, P., Middleton, R., and Blanchini, F., "Discrete-time control for switched positive systems with application to mitigating viral escape," *Int. J. Robust and Nonlinear Control*, vol. 21, pp. 1093–1111, 2011.

[2] Soler, M., Olivares, A., and Staffetti, E., "Framework for aircraft trajectory planning toward an efficient air traffic management," *Journal of Aircraft*, vol. 49 (1), pp.985-991, 2012.

[3] Rinehart, M., Dahleh, M., Reed, D., Kolmanovsky, I., "Suboptimal control of switched systems with an application to the DISC engine," *IEEE Transactions on Control Systems Technology*, vol. 16 (2), pp.189-201, 2008.

[4] Benmansour, K., Benalia, A., Djemaï, M., and de Leon, J., "Hybrid control of a multicellular converter," *Nonlinear Analysis: Hybrid Systems*, vol. 1 (1), pp.16-29, 2007.

[5] Gong, Z., Liu, C., Feng, E., Wang, L., Yu, Y., "Modelling and optimization for a switched system in microbial fed-batch culture," *Applied Mathematical Modelling*, vol. 35 (7), pp.3276-3284, 2011.

[6] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions," *International Journal of Control*, vol. 75 (16), pp. 1406-1426, 2002.

[7] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Trans. on Automatic Control*, vol. 49 (1), pp.2- 16, 2004.

[8] Egerstedt, M., Wardi, Y., and Axelsson, H., "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. on Automatic Control*, vol. 51 (1), pp.110-115, 2006.

[9] Axelsson, H., Boccadoro, M., Egerstedt, M., Valigi, P., and Wardi, Y., "Optimal mode-switching for hybrid systems with varying initial states," *Nonlinear Analysis: Hybrid Systems*, vol. 2 (3), pp.765–772, 2008.

[10] Ding, X., Schild, A., Egerstedt M., and Lunze J., "Real-time optimal feedback control of switched autonomous systems," Proc. *IFAC Conference on Analysis and Design of Hybrid Systems*, pp.108-113, 2009.

[11] Kamgarpoura, M., and Tomlin, C., "On optimal control of non-autonomous switched systems with a fixed mode sequence," *Automatica*, vol. 48, pp.1177–1181, 2012.

[12] Zhao, R., and Li, S., "Switched system optimal control based on parameterizations of the control vectors and switching instant," Proc. *Chinese Control and Decision Conference*, pp. 3290-3294, 2011.

[13] Axelsson, H., Egerstedt, M., Wardi, Y., and Vachtsevanos, G., "Algorithm for switching-time optimization in hybrid dynamical systems," Proc. *IEEE International Symposium on Intelligent Control*, Limassol, Cyprus, 2005.

[14] Wardi, Y., and Egerstedt, M., "Algorithm for optimal mode scheduling in switched systems," Proc. *American Control Conference*, Montréal, Canada, pp. 4546-4551, 2012.

[15] Luus, R., and Chen, Y, "Optimal switching control via direct search optimization," *Asian Journal of Control*, Vol. 6 (2), pp. 302-306, 2004.

[16] Rungger, M., and Stursberg, O., "A numerical method for hybrid optimal control based on dynamic programming," *Nonlinear Analysis: Hybrid Systems*, vol. 5 (2), pp.254–274, 2011.

[17] Sakly, M., Sakly, A., Majdoub, N., and Benrejeb, M., "Optimization of switching instants for optimal control of linear switched systems based on genetic algorithms," Proc. *IFAC Int. Conf. Intelligent Control Systems and Signal Processing*; Istanbul, 2009.

[18] Long, R., Fu, J., Zhang, L., "Optimal control of switched system based on neural network optimization," Proc. *Int. Conference on Intelligent Computing*, pp.799-806, 2008.

[19] Heydari, A., and Balakrishnan, S. N., "Optimal multi-therapeutic HIV treatment using a global optimal switching scheme," *Applied Mathematics and Computation*, vol. 219, pp. 7872-7881, 2013.

[20] Lien C.-H., Yu K.-W., Chang H.-C., Chung L.-Y., and Chen J.-D., "Switching signal design for exponential stability of discrete switched systems with interval time-varying delay," *Journal of the Franklin Institute*, vol. 349 (6), pp. 2182–2192, 2012.

[21] Zhai S., and Yang X.-S., "Exponential stability of time-delay feedback switched systems in the presence of asynchronous switching," *Journal of the Franklin Institute*, vol. 350 (1), pp. 34–49, 2013.

[22] Zhang, W., Hu, J., and Abate, A., "On the value functions of the discrete-time switched LQR problem," *IEEE Transactions on Automatic Control*, vol. 54 (11), pp. 2669-2674, 2009.

[23] Kirk, D. E., *Optimal Control Theory: An Introduction,* Dover Publications, New York, 2004, pp. 53-58.

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

[25] Ding, J., and Balakrishnan, S. N., "Approximate dynamic programming solutions with a single network adaptive critic for a class of nonlinear systems," *J Control Theory Appl, vol. 9* (3), pp. 370–380, 2011.

[26] Heydari, A., and Balakrishnan, S. N., "Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24 (1), pp. 145-157, 2013.

[27] Homik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," NeuralNetworks, vol. 2, pp. 359-366, 1989.

[28] Stone, M., and Goldbart, P., *Mathematics for Physics - A Guided Tour for Graduate Students*, Cambridge, England, Cambridge University Press, p. 70, 2009.

[29] Trench, W. F., *Introduction to Real Analysis*, available online at http://ramanujan.math.trinity.edu/ wtrench/texts/trench_real_analysis.pdf, 2012, p. 309.

[30] Malmborg J., and Eker, J., "Hybrid control of a double tank system", Proc. *IEEE Conference on Control Application*, Hartford, Connecticut, 1997.

7. OPTIMAL SWITCHING BETWEEN CONTROLLED SUBSYSTEMS WITH FREE MODE SEQUENCE

Ali Heydari and S.N. Balakrishnan

ABSTRACT

The problem of optimal switching and control of systems with nonlinear subsystems is investigated in this study where the mode sequence and the switching times between the modes are unspecified. An approximate dynamic programming based method is developed which provides an online solution for unspecified initial conditions and different final times. The convergence of the proposed algorithm is proved. Versatility of the proposed method and its excellent performance are illustrated through different numerical examples.

I. INTRODUCTION

Examples of switching systems can be found in dynamical systems in different fields, from aerospace to chemical engineering [1-5]. A switching system is characterized by a group of subsystems with different dynamics of which one is active at each time instant. Hence, in order to control such systems one needs a switching schedule along with a control input to be applied. There are a few papers in this area [6-16], however, still there are many open issues even for the case of linear subsystems with a quadratic cost functions [7,17].

Development in the field can be mainly classified into two categories. In the first category, the sequence of active subsystems, called mode sequence, is selected a priori [6-12], and the problem, i.e., finding the switching instants between the modes, is solved using nonlinear programming methods. In these papers, the gradient of the cost with respect to the switching instants/points is calculated. Afterward, the switching instants/points are adjusted to find the *local* optimum. Iterative solution to a nonlinear optimization problem is suggested in [10] and using the combination of this control approach with ideas from model predictive control, the authors developed the so-called crawling window optimal control scheme for the optimal switching problem. The second category is based on discretizing the problem in order to deal with a *finite* number of options. Authors of [13] utilized a direct search to evaluate the cost function for different randomly selected switching time sequences among the finite number of options to select

the best sequence. In [14], state and input spaces are discretized for calculation of the value function for optimal switching through dynamic programming. In [15] genetic algorithm is used to find the optimal switching times among the choices. A hybrid neural network (NN) is used for solving the optimal switching problem for a pre-specified initial condition in [16].

All the cited methods work only with *a specific initial condition*; each time the initial condition is changed, a new set of computations needs to be performed to find the new optimal switching instants. In order to extend the validity of the results for different initial conditions within a pre-selected set, in [9] a solution is found as the local optimum in the sense that it minimizes the worst possible cost for all trajectories starting in the selected initial states set. Also, the derivative of the switching parameters with respect to the initial conditions is sought through a sensitivity analysis.

Recently, the authors of this study proposed a NN based scheme in [18] for optimal switching of systems with *fixed mode sequence* and *autonomous* dynamics, i.e., where the subsystems do not admit control inputs. Two major contributions of the current paper lie in the fact that the mode sequence is considered 'free' to be selected and that the systems considered are non-autonomous. Investigation of controlled subsystem makes the problem more complicated due to the inter-coupling that exists between the effect of *switching* between the modes and *applying* different controls once a mode is active. Furthermore, solving a free mode sequence problem is much more complicated than a problem with a fixed number of changes.

In the past two decades, approximate dynamic programming (ADP) has been shown to have a lot of promise in solving conventional optimal control problems with NN as the function approximator [19-32]. ADP is usually carried out using a two network synthesis called adaptive critics (ACs) [20-22]. In the heuristic dynamic programming (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 [22,23]. In the dual heuristic programming (DHP) formulation, while the action network remains the same as the HDP, the critic network outputs the costates with the current states as inputs [20,24,25]. The Single Network Adaptive Critics (SNAC) architecture developed in [26] is shown to be able to eliminate

the need for the second network and perform DHP using only one network. Similarly, the J-SNAC eliminates the need for the action network in an HDP scheme [27]. Note that the developments in [19-27] are for *infinite-horizon* problems. The use of ADP for solving *finite-horizon* optimal control of conventional problems was considered in [28-32]. Authors of [28] developed a time-varying neurocontroller for solving a problem with state constraints. In [29] a single NN with a single set of weights was proposed which takes the time-to-go as an *input* along with the states and generates the fixed-final-time optimal control for discrete-time nonlinear systems. Finite-horizon problems with *unspecified* terminal times were considered in [30-32].

In this study, a method based on ADP is developed to solve optimal switching problems. The idea is as simple as learning the optimal cost-to-go and the optimal control for different active modes. It is shown that having these functions the optimal mode can be found in a feedback form, i.e., as a function of the instantaneous state of the system and the remaining time. An algorithm is developed which fits in the category of HDP for learning the desired functions along with a proof of convergence. This method has several advantages over existing developments in the field: a) It provides *global* optimal switching (subject to the assumed neural network structure) unlike the nonlinear programming based methods which could provide only local optimal solution. b) The order of active subsystems and the number of switching are free. c) The neurocontroller determines optimal solution for *unspecified initial conditions*, without needing to retrain the networks. d) Once trained, the neurocontroller gives solution to *any other final time as well, as long as the new final time is not greater than the final time for which the network is trained.* e) The switching is scheduled in a feedback form, hence, it has inherent robustness of feedback solutions in moderate disturbance rejection. f) The proposed method provides optimal *control* as well as optimal *switching* schedule for the control of the system.

The rest of this paper is organized as follows: The problem formulation is presented in section II and the proposed solution is described in section III. Approximations of the optimal cost-to-go and the optimal control with neural networks are explained in section IV. Numerical analyses are given in section V. Conclusions from this study are given in section VI.

II. PROBLEM FORMULATION

A switching system with nonlinear input-affine subsystems can be represented by a set of M subsystems or modes as

$$\dot{x}(t) = \bar{f}_{j(t)}\big(x(t)\big) + \bar{g}_{j(t)}\big(x(t)\big)u(t),\ j(t) \in \mathcal{J}, \tag{1}$$

where functions $\bar{f}_j: \mathbb{R}^n \to \mathbb{R}^n$, and $\bar{g}_j: \mathbb{R}^n \to \mathbb{R}^{n\times m}$, $\forall j \in \mathcal{J} \equiv \{1, 2, \dots, M\}$, represent the dynamics of the subsystems and are assumed to be smooth. Integers n and m denote the dimension of state vector x and control vector u, respectively. The continuous time is denoted with t and the initial and final times are denoted with t_0 and t_f, respectively. Controlling the switching systems requires a *control input*, $u: [t_0, t_f) \to \mathbb{R}^m$, and a *switching function*, $j: [t_0, t_f) \to \mathcal{J}$. The latter determines the active subsystem at time t and the former provides the input to the active subsystem. The optimal solution, however, is a solution that minimizes cost function

$$J = \psi\left(x\big(t_f\big)\right) + \int_{t_0}^{t_f}\left(\bar{Q}\big(x(t)\big) + u(t)^T\bar{R}u(t)\right)dt. \tag{2}$$

Convex positive semi-definite functions $\bar{Q}: \mathbb{R}^n \to \mathbb{R}$ and $\psi: \mathbb{R}^n \to \mathbb{R}$ penalize the states and $\bar{R} \in \mathbb{R}^{m\times m}$ is a positive definite matrix penalizing the control effort, in the selected cost function. The problem is to determine an input history $u(t)$ and a switching history $j(t)$ such that cost function (2) subject to dynamics (1) is minimized.

III. PROPOSED SOLUTION

Approximate dynamic programming [19-22] framework which is the backbone of the solution developed in this study is formulated with discrete-time dynamics. Therefore, the dynamics and the cost function are discretized using small sampling time Δt:

$$x_{k+1} = f_{j_k}(x_k) + g_{j_k}(x_k)u_k,\ k \in K,\ j_k \in \mathcal{J}, \tag{3}$$

$$J = \psi(x_N) + \textstyle\sum_{k=0}^{N-1}(Q(x_k) + u_k^T R u_k), \tag{4}$$

where $N = \big(t_f - t_0\big)/\Delta t$, $x_k = x(k\Delta t + t_0)$, $u_k = u(k\Delta t + t_0)$, and $j_k = j(k\Delta t + t_0)$. Subscript k denotes the discrete time index and $K \equiv \{0,1, \dots, N-1\}$. If Euler integration is used for discretization, one has $f_j(x) \equiv x + \Delta t\bar{f}_j(x)$, $g_j(x) \equiv \Delta t\bar{g}_j(x)$, $Q(x) \equiv \Delta t\bar{Q}(x)$, and $R \equiv \Delta t\bar{R}$.

Denoting the cost-to-go from each time step k and state vector x_k by $J_k(x_k)$ one has

$$J_k(x_k) = \psi(x_N) + \sum_{\hbar=k}^{N-1}\big(Q(x_\hbar) + u_\hbar^T R u_\hbar\big). \tag{5}$$

From the form of the cost function, it directly follows that

$$J_N(x_N) = \psi(x_N),$$

$$J_k(x_k) = Q(x_k) + u_k^T R u_k + J_{k+1}(x_{k+1}),\ \forall k \in K. \tag{6}$$

Based on Bellman principle of optimality [33], regardless of what decisions for $j_\hbar$, $\hbar \in \{0,1,\dots,k-1\}$ are made, the optimal solution for the remaining time steps, i.e., $\hbar \in \{k, k+1, \dots, N-1\}$ is the solution which optimizes $J_k(x_k)$. The method developed in this study is based on approximating the optimal cost-to-go, denoted with $J_k^*(x_k)$, and the optimal control 'given' the active subsystem j, denoted with $u_k^{j,*}(x_k)$, $\forall j \in \mathcal{J}$. Once these functions are learned, the optimal mode at current time, k, and current state, x_k, denoted with $j_k^*(x_k)$, is given by

$$j_k^*(x_k) = argmin_{j\in\mathcal{J}}\left(Q(x_k) + u_k^{j,*^T} R u_k^{j,*} + J_{k+1}^*\big(f_j(x_k) + g_j(x_k)u_k^{j,*}\big)\right)$$

$$= argmin_{j\in\mathcal{J}}\left(u_k^{j,*^T} R u_k^{j,*} + J_{k+1}^*\big(f_j(x_k) + g_j(x_k)u_k^{j,*}\big)\right). \tag{7}$$

The minimization given in (7) is among the finite number of elements of $\mathcal{J}$ and can be done online easily with relatively small number of computations. For example, if the system has two subsystems, finding optimal active subsystem at each instant k simplifies to evaluating scalar values $u_k^{j,*^T} R u_k^{j,*} + J_{k+1}^*\big(f_j(x_k) + g_j(x_k)u_k^{j,*}\big)$ for $j = 1$ and $j = 2$ and comparing the values to select the optimal j. This process needs to be done at each instant k, $\forall k \in K$. Once $j_k^*(x_k)$ is calculated, the respective control approximator can be used to output the control value. The next section gives an algorithm for learning desired functions $J_k^*(x_k)$ and $u_k^{j,*}(x_k)$, $\forall j \in \mathcal{J}$.

IV. APPROXIMATING OPTIMAL CONTROL AND OPTIMAL COST-TO-GO

In order to motivate the idea of using ADP for obtaining optimal control and optimal cost-to-go for *switching* problems, the use of ADP for *conventional* optimal control problems with fixed-final-time cost functions is discussed first.

A. Adaptive Critics for Conventional Optimal Control

Assume the conventional fixed-final-time optimal control problem

$$x_{k+1} = f(x_k) + g(x_k)u_k, \forall k \in K \tag{8}$$

where $f: \mathbb{R}^n \to \mathbb{R}^n$ and $g: \mathbb{R}^n \to \mathbb{R}^{n\times m}$ along with the cost function given in (4). The optimal solution to the problem of minimizing cost function (4) subject to dynamics (8) is given by Bellman equation [33]

$$J_N^*(x_N) = \psi(x_N), \tag{9}$$

$$J_k^*(x_k) = Q(x_k) + u_k^{*T} R u_k^* + J_{k+1}^*(x_{k+1}^*),\ \forall k \in K, \tag{10}$$

$$u_k^* = -\frac{1}{2}R^{-1}g(x_k)^T \frac{\partial J_{k+1}^*(x_{k+1})}{\partial x_{k+1}}\Big|_{x_{k+1}^*},\ \forall k \in K, \tag{11}$$

where $x_{k+1}^* = f(x_k) + g(x_k)u_k^*$ and gradient $\partial J_{k+1}^*(x_{k+1}/\partial x_{k+1})$ is forms as a column vector.

In the HDP scheme [21] with ADP, two NNs named actor and critic are trained for approximating the optimal control and the optimal cost-to-go, respectively, for infinite-horizon problems. Ref. [29] extends the idea to fixed-final-time problems, through approximating the optimal parameters versus the current state as well as the time-to-go (remaining time). An iterative learning scheme can be derived from Bellman equation for learning the optimal control and the optimal cost-to-go for the fixed-final-time problem by utilizing Eqs. (9) and (10) and replacing with [29]

$$u_k^{(i+1)}(x_k) = -\frac{1}{2}R^{-1}g(x_k)^T \frac{\partial J_{k+1}^*(x_{k+1})}{\partial x_{k+1}}\Big|_{x_{k+1}^{(i)}}, \forall k \in K, \tag{12}$$

Superscript '(i)' denotes the index of iteration. Moreover, in (12) one has $x_{k+1}^{(i)} \equiv f(x_k) + g(x_k)u_k^{(i)}(x_k)$, $x_{k+1}^* \equiv f(x_k) + g(x_k)u_k^*(x_k)$, and the converged value of $u_k^{(i)}$ is denoted with u_k^*. Note that the iterations take place only in determining the optimal

control, starting with an initial guess on u_k^0. Once the converged control value is obtained, the optimal cost-to-go is calculated using (10), without any need for iteration.

Denoting the approximated optimal cost-to-go and the approximated optimal control with $J_k(x_k)$ and $u_k(x_k)$, respectively, and selecting linear in parameter networks, the expressions for the actor (control) and the critic (cost), can be written as

$$u_k(x_k) = V_k^T \sigma(x_k),\ k \in K, \tag{13}$$

$$J_k(x_k) = W_k^T \phi(x_k),\ k \in K \cup \{N\}, \tag{14}$$

Functions $\sigma: \mathbb{R}^n \to \mathbb{R}^p$ and $\phi: \mathbb{R}^n \to \mathbb{R}^q$ represent the linearly independent smooth basis functions, for p and q being positive integers denoting the respective number of neurons. Matrices $V_k \in \mathbb{R}^{p \times m}$ and $W_k \in \mathbb{R}^q$ are the weights of the actor and the critic networks at time step k, respectively. Utilizing different weights for different time steps provides the network with the ability to learn the time-dependent behavior of the solution to fixed-final-time problems. Eqs. (9), (10), and (12) may be used to find network weights V_k and W_k, $\forall k$. Substituting (13) and (14) in equations (9), (12), and (10) leads respectively to

$$W_N^T \phi(x_N) = \psi(x_N), \tag{15}$$

$$V_k^{(i+1)^T} \sigma(x_k) = -\frac{1}{2} R^{-1} g(x_k)^T \nabla \phi \left(f(x_k) + g(x_k) V_k^{(i)^T} \sigma(x_k) \right)^T W_{k+1},\ \forall k \in K, \tag{16}$$

$$W_k^T \phi(x_k) = Q(x_k) + \sigma(x_k)^T V_k R V_k^T \sigma(x_k) + W_{k+1}^T \phi\Big(f(x_k) + g(x_k) V_k^T \sigma(x_k) \Big),$$

$$\forall k \in K. \tag{17}$$

where superscript '(i)' on $V_k^{(i)}$ denote the iteration index and the converged value is denoted with V_k. Moreover, $\nabla \phi(x) \equiv \partial \phi(x)/\partial x$ is formed as a column vector. Unknowns W_k and V_k can be calculated in a backward-in-time fashion, considering Eqs. (15)-(17). In other words, using (15) one can calculate W_N. Then, having W_N one can calculate V_{N-1} using the iterative relation given in (16). Having calculated W_N and V_{N-1}, unknown W_{N-1} can be found using (17). Repeating this process from $k = N-1$ to $k = 0$, all the unknowns weights can be calculated. This idea is detailed in Algorithm 1.

Algorithm 1

Step 1: Find W_N such that $W_N^T \phi(x_N) \cong \psi(x_N)$ for different $x_N \in \Omega$ where Ω denotes a compact subset of $\mathbb{R}^n$ representing the domain of interest.

Step 2: For $k = N-1$ to $k = 0$ repeat

{

Step 3: Set $i = 0$ and select a guess on $V_k^{(0)}$.

Step 4: Randomly select n different state vectors $x_k^{[j]} \in \Omega, j \in \{1,2,..,n\}$, for n being a large positive integer.

Step 5: Set $x_{k+1}^{[j]} = f\left(x_k^{[j]}\right) + g\left(x_k^{[j]}\right) u_k^{[j]}$, where $u_k^{[j]} = {V_k^{(i)}}^T \sigma\left(x_k^{[j]}\right)$, $\forall j \in \{1,2,..,n\}$.

Step 6: Find $V_k^{(i+1)}$ such that

$${V_k^{(i+1)}}^T \sigma\left(x_k^{[j]}\right) \cong -\frac{1}{2} R^{-1} g\left(x_k^{[j]}\right)^T \nabla\phi\left(x_{k+1}^{[j]}\right)^T W_{k+1}, \forall j \in \{1,2,..,n\}.$$

Step 7: Set $i = i+1$ and repeat Step 6, until $\|V_k^{(i+1)} - V_k^{(i)}\|$ converges with a preset tolerance.

Step 8: Set $V_k = V_k^{(i)}$.

Step 9: Find W_k such that

$$W_k^T \phi\left(x_k^{[j]}\right) \cong Q\left(x_k^{[j]}\right) + \sigma\left(x_k^{[j]}\right)^T V_k R V_k^T \sigma\left(x_k^{[j]}\right) + W_{k+1}^T \phi\left(f\left(x_k^{[j]}\right) + g\left(x_k^{[j]}\right) V_k^T \sigma\left(x_k^{[j]}\right)\right), \forall j \in \{1,2,..,n\}.$$

}

In Steps 1, 6, and 9 of Algorithm 1, the method of Least Squares, explained in Appendix A, can be used for finding the unknown weights in terms of the given parameters.

Remark 1: *Uniform approximation* capability of neural networks [34,35] in approximation of continuous functions indicates that once the network is trained for a large enough number of samples, denoted by n, distributed throughout the domain of interest, the network is able to approximate the output for any new sample of the domain with a bounded approximation error. This error bound can be made arbitrarily small if

NNs' activation functions are rich and the number of training samples, n, is large enough. For the linear in weight neural network selected in this study and the polynomial basis function utilized in the numerical examples, Weierstrass approximation theorem [36] proves a similar uniform approximation capability.

Once the networks are trained *offline*, the optimal control will be given in a feedback form by the actor and can be implemented for *online* control of the plant. The critic network approximates the optimal cost-to-go as a function of the given state and time. This feature of the critic will be used in the next subsection, along with the actor, to find solution to the optimal switching problem.

B. Adaptive Critics for Switching Optimal Control

The idea presented in the previous subsection for optimal control of conventional problems is extended to switching problems in this subsection. An algorithm is proposed for learning the optimal cost-to-go and the controls, where $(M+1)$ neural networks are utilized. In other words, one critic will be used to approximate $J_k^*(x_k)$, denoted with $J_k(x_k)$, and M actors will be used to approximate $u_k^{j,*}(x_k)$, denoted with $u_k^j(x_k)$,$\forall j \in \mathcal{J}$.

$$u_k^j(x_k) = {V_k^j}^T \sigma(x_k), j \in \mathcal{J}, k \in K, \tag{18}$$

$$J_k(x_k) = W_k^T \phi(x_k), k \in K \cup \{N\}, \tag{19}$$

Note that the superscript j on $V_k^j \in \mathbb{R}^{p \times m}$ relates the actor to the respective subsystem. In other words, the approximated optimal control 'given' the active subsystem j at time k is $u_k^j(x_k) = {V_k^j}^T \sigma(x_k)$.

As for the weight update laws for determining W_k and V_k^j, $\forall k$ and $\forall j$, the iterative learning scheme given in (9), (10), and (12) may be adapted as follows: The training starts with (9). Equations (12) and (10), however, are adapted as follows. The new fixed point iteration given by

$$u_k^{j,(i+1)} = -\frac{1}{2} R^{-1} g_j(x_k)^T \left. \frac{\partial J_{k+1}(x_{k+1})}{\partial x_{k+1}} \right|_{x_{k+1}^{j,(i)}}, \forall k \in K \text{ and } \forall j \in \mathcal{J}, \tag{20}$$

can be used to find the optimal control given every active subsystem j, denoted with $u_k^{j,*}$, $\forall j \in \mathcal{J}$, where $x_{k+1}^{j,(i)} \equiv f_j(x_k) + g_j(x_k)u_k^{j,(i)}$. Superscript '$(i)$' denotes the index of iteration and the converges value of $u_k^{j,(i)}$ is denoted with $u_k^{j,*}$, $\forall j \in \mathcal{J}$. Afterward, using

$$j_k^*(x_k) = argmin_{j\in\mathcal{J}}\left(u_k^{j,*^T} R u_k^{j,*} + J_{k+1}^*\left(f_j(x_k) + g_j(x_k)u_k^{j,*}\right)\right),\ \forall k \in K,$$

the optimal mode at the current time and state, denoted with $j_k^*(x_k)$, will be calculated and used in

$$J_k^*(x_k) = Q(x_k) + u_k^{j_k^*(x_k),*^T} R u_k^{j_k^*(x_k),*} + J_{k+1}^*\left(f_{j_k^*(x_k)}(x_k) + g_{j_k^*(x_k)}(x_k)u_k^{j_k^*(x_k),*}\right),$$

$$\forall k \in K \quad (21)$$

to find the optimal cost-to-go, $J_k^*(x_k)$. This process may continue in a backward form from $k = N-1$ to $k = 0$. By using the equations for the network structures (18) and (19) in equations (9), (20), and (21), the desired weight update law can be obtained. Algorithm 2 describes the detailed learning process.

Algorithm 2

Step 1: Find W_N such that $W_N^T\phi(x_N) \cong \psi(x_N)$ for different $x_N \in \Omega$ where Ω denotes a compact subset of $\mathbb{R}^n$ representing the domain of interest.

Step 2: For $k = N-1$ to $k = 0$ repeat Steps 3 through 11 below

{

Step 3: Randomly select n different state vectors $x_k^{[j]} \in \Omega$, $j \in \{1,2,..,n\}$, for n being a large positive integer.

Step 4: For $j = 1$ to $j = M$ repeat Steps 5 through 9 below.

{

Step 5: Set $i = 0$ and select a guess for $V_k^{j,(0)}$.

Step 6: Set $x_{k+1}^{j,[j]} = f_j\left(x_k^{[j]}\right) + g_j\left(x_k^{[j]}\right)u_k^{j,[j]}$, where $u_k^{j,[j]} = V_k^{j,(i)^T}\sigma\left(x_k^{[j]}\right), \forall j \in \{1,2,..,n\}$.

Step 7: Find $V_k^{j,(i+1)}$ such that

$$V_k^{j,(i+1)^T}\sigma\left(x_k^{[j]}\right) \cong -\frac{1}{2}R^{-1}g_j\left(x_k^{[j]}\right)^T \nabla\phi\left(x_{k+1}^{j,[j]}\right)^T W_{k+1}, \forall j \in \{1,2,..,n\}. \quad (22)$$

Step 8: Set $i = i + 1$ and repeat Step 7, until $\|V_k^{j,(i+1)} - V_k^{j,(i)}\|$ converges with a preset tolerance.

Step 9: Set $V_k^j = V_k^{j,(i)}$.

}

Step 10: Set $u_k^{j,[j]} = {V_k^j}^T \sigma\left(x_k^{[j]}\right)$, $\forall j \in \{1,2,..,n\}$, $\forall j \in \mathcal{J}$, and calculate

$$j_k^*(x_k^{[j]}) = argmin_{j\in\mathcal{J}}\left(u_k^{j,[j]^T} R u_k^{j,[j]} + W_{k+1}^T \phi\left(f_j\left(x_k^{[j]}\right) + g_j\left(x_k^{[j]}\right) u_k^{j,[j]}\right)\right).$$

Step 11: Find W_k such that

$$W_k^T \phi\left(x_k^{[j]}\right) \cong Q\left(x_k^{[j]}\right) + {u_k^{j_k^*(x_k^{[j]}),[j]}}^T R u_k^{j_k^*(x_k^{[j]}),[j]} +$$

$$W_{k+1}^T \phi\left(f_{j_k^*(x_k^{[j]})}\left(x_k^{[j]}\right) + g_{j_k^*(x_k^{[j]})}\left(x_k^{[j]}\right) u_k^{j_k^*(x_k^{[j]}),[j]}\right), \forall j \in \{1,2,..,n\}. \quad (23)$$

}

The iterative weight updates of Algorithm 2, i.e., Eq. (22) can be rewritten in terms of the NN weights as

$${V_k^{j,(i+1)}}^T \sigma\left(x_k^{[j]}\right) \cong -\frac{1}{2} R^{-1} g_j\left(x_k^{[j]}\right)^T \nabla\phi\left(f_j\left(x_k^{[j]}\right) + g_j\left(x_k^{[j]}\right) {V_k^{j,(i)}}^T \sigma\left(x_k^{[j]}\right)\right)^T W_{k+1},$$

$$\forall j \in \{1,2,..,n\}. \quad (24)$$

Eq. (24) relates $V_k^{j,(i+1)}$ to $V_k^{j,(i)}$, i.e., it is an iterative equation. Its converged value, V_k^j, will be used in Eq. (23), and a least squares solution can be found for W_k, see Appendix A. The following theorem provides the sufficient condition for the convergence of iterative equation (24).

Theorem 1: The iterations given by (24) converge with any selected initial guess on $V_k^{j,(0)}$, $\forall j \in \mathcal{J}$, and $\forall k \in K$, providing the sampling time selected for discretization of continuous dynamics (1) is small enough.

The proof is given in Appendix B.

In Theorem 1 the role of the sampling time in discretization of a continuous system is emphasized. It is worthwhile to discuss this issue in detail. Substituting (18) and (19) in optimal control equation (20), leads to

$$V_k^{j^T}\sigma\left(x_k^{[j]}\right) \cong -\frac{1}{2}R^{-1}g_j\left(x_k^{[j]}\right)^T \nabla\phi\left(f_j\left(x_k^{[j]}\right) + g_j\left(x_k^{[j]}\right)V_k^{j^T}\sigma\left(x_k^{[j]}\right)\right)^T W_{k+1},$$

$$\forall j \in \{1,2,..,n\}. \quad (25)$$

which is the same as (24) except that $V_k^{j,(i+1)}$ and $V_k^{j,(i)}$ on both sides are replaced with V_k^j. Optimal weights V_k^j, $\forall k \in K$ and $\forall j \in \mathcal{J}$, can be calculated by solving the nonlinear equation given in (25), without using the iteration given in (24). Typically, one needs to resort to numerical methods for solving the set of equations (25). Theorem 1 proves that for any given smooth dynamics and smooth basis functions, if the sampling time is small enough, the iterations given in (24) converge to the solution to the nonlinear equation (25). However, if the sampling time is fixed, then certain conditions on the dynamics or the cost function terms need to hold in order for the iterations to converge. These conditions can be easily derived from the proof of Theorem 1.

Assuming the basis functions of the NN are selected rich enough to approximate the cost-to-go and the optimal control functions with a desired accuracy, the method developed here provides optimal solution due to its basis on dynamic programming [33]. In other words, if $J_k^*(x_k)$ and $u_k^{j,*}(x_k)$s are accurately approximated, the optimal mode will always be given by (7). Therefore, an analysis on the approximation capability of the NN is required. As mentioned in Remark 1, NNs can provide a uniform approximation with any desired degree of accuracy providing the function subject to approximation in a *continuous* function. Considering Eqs. (15)-(17) and (22), the continuity of the functions subject to approximation, given in the right hand sides of these equations, follows from the convexity of $\psi(.)$ and $Q(.)$ as well as the continuity of $f_i(.)$s, $g_i(x_k)$s and the basis functions. For Eq. (23), however, due to the switching between the modes, i.e., the discontinuous nature of $j_k^*(.)$, the continuity of the right hand side is not obvious. Theorem 2 proves the required continuity.

Theorem 2: Let function $F^j: \mathbb{R}^n \to \mathbb{R}$, $\forall j \in \mathcal{J}$, be defined as

$$F^j(x) \equiv \sigma(x)^T V_k^j R V_k^{j^T} \sigma(x) + W_{k+1}^T \phi\left(f_j(x) + g_j(x) V_k^{j^T} \sigma(x)\right).$$

If the active mode at each instant k and state vector x is given by

$$j_k^*(x) = argmin_{j \in \mathcal{J}} F^j(x), \tag{26}$$

then, function $F^{j_k^*(x)}(x)$ is a continuous function versus x at every $x \in \Omega$.
The proof is given in Appendix B.

C. Implementation and Control

Once the NNs' weights are trained using Algorithm 2, one may use them for online optimal control/scheduling of the system. This is done in real-time through feeding the current state x_k, at each time step $k \in K$ to equation (7), repeated below in terms of the NNs, to calculated the optimal active mode, $j_k^*(x_k)$.

$$j_k^*(x_k) = argmin_{j \in \mathcal{J}} \left(\sigma(x_k)^T V_k^j R V_k^{j^T} \sigma(x_k) + W_{k+1}^T \phi\left(f_j(x_k) + g_j(x_k) V_k^{j^T} \sigma(x_k)\right)\right).$$

Having calculated $j_k^*(x_k)$, the optimal control u_k^* is given by $u_k^* = u_k^{j_k^*(x_k),*} \cong V_k^{j_k^*(x_k)^T} \sigma(x_k)$. Hence, the optimal solution can be found *online* in a *feedback form*. Note that $\mathcal{J}$ has a finite number of elements and the minimization given in (7) is as simple as comparing the scalar values of the argument subject to minimization for different $j \in \mathcal{J}$ and selecting the optimal one.

As mentioned in the introduction, one of the features of this method is providing approximate global optimal solution. A requirement for this characteristic is the learned cost-to-go and controls being approximations of the *global* optimal cost-to-go and controls. Using Algorithm 2, the global optimality of the trained networks follows from the proof of Theorem 1. In other words, once it is proved that (24) is a contraction mapping, the uniqueness of fixed point V_k^j to the iterative Eq. (24) follows [37]. Details of this result are beyond the scope of this study and are presented in [38]. As for the weights of the critic, W_k, using least squares leads to the global optimal weights, due to the convexity of least squares problems [39].

Looking at Eq. (7), which is 'the decision maker' for switching, one may observe high frequency switching between the modes in some problems. In fact, this behavior is observed in Example 2 in this study. The following two remedies are suggested to avoid high frequency switching:

1- The Minimum Dwell Time Remedy: Dictating a minimum dwell time after each switching can eliminate high frequency switching. After the first mode selection, one can dictate a minimum dwell time before switching to another mode at every change. That is, once a switching occurs, one may skip evaluating Eq. (7) and instead stay with the current active subsystem until the minimum dwell time is passed.

2- The Threshold Remedy: Selecting a positive real number as the threshold, the switching is allowed once the cost difference between activating the new mode and staying with the current mode is more than the threshold. To be more specific, assume the active subsystem is i right before time instant k, and by evaluating Eq. (7) one realizes that switching to subsystem j leads to the cost-to-go less than the cost-to-go of staying with subsystem i. In such a case switching to subsystem j is allowed only if

$$u_k^{j,*^T} R u_k^{j,*} + J_{k+1}^*\left(f_j(x_k) + g_j(x_k)u_k^{j,*}\right) < u_k^{i,*^T} R u_k^{i,*} + J_{k+1}^*\left(f_i(x_k) + g_i(x_k)u_k^{i,*}\right) + \tau,$$

where the pre-selected threshold is denoted with τ.

The same algorithm (Algorithm 2) may still be used, in the offline training stage of the NNs. The abovementioned alternative remedies, however, can be used in the online control. The alterations created by the remedies result in a 'sub-optimal' control of the system. The result will remain sub-optimal because the neurocontroller calculates the optimal solution in a feedback form. More specifically, the perturbation due to the applied remedy can be considered as a disturbance for the controller. Providing suitable selection of the minimum dwell time or the threshold, the feedback nature of the controller can deal with the resulting disturbance without too much performance degradation. This behavior is due to the inherent nature of feedback controllers in moderate disturbance rejection.

V. NUMERICAL APPLICATIONS AND ANALYSIS

A. Example 1

First example is a scalar switching system with two modes, given below, is selected

$$\dot{x} = \begin{cases} f_1(x) + g_1(x)u \equiv -x + u \\ f_2(x) + g_2(x)u \equiv -x^3 + u \end{cases}$$

The selected cost function is $J = 100\big(x(t_f) + 2\big)^2 + \int_0^{t_f} u(t)^2 dt$ where $t_f = 1\ s$. Selecting the domain of interest of $\Omega = [-3\ 3]$ the optimal switching function for the given system can be analytically calculated as

$$j^*(t) = \begin{cases} 1 & if \quad 0 \le x(t) \le 1 \ \ or \ \ -2 \le x(t) \le -1 \\ 2 & if \quad 1 \le x(t) \ \ or \ \ -1 \le x(t) \le 0 \ \ or \ x(t) < -2 \end{cases} . \tag{27}$$

For example, if $x(t) > 1$, utilizing subsystem 2 leads to a faster convergence toward the origin with less control effort due to $|f_2(x)| > |f_1(x)|$, therefore, the optimal mode in this case is subsystem 2. Note that if $x(t) \in [-2, 0]$, the optimal mode is the mode which has smaller $|f_j(x)|$ in order to require less control effort to derive the state away from it point of attraction, i.e., the origin, toward the desired terminal point of -2. The existence of analytical optimal switching function (27) for this system makes it a suitable example for investigating the performance of the developed method.

Polynomial functions x^i, for $i \in \{0,1,2, \dots ,6\}$ and $i \in \{0,1,2, \dots ,5\}$ are selected for the basis functions $\phi(.)$ and $\sigma(.)$, respectively. Note that, as explained in Remark 1, the resulting NNs will have the desired uniform approximation capability. The horizon is discretized to $N = 200$ time-steps, i.e., $\Delta t = 0.005\ s$. The training steps detailed in Algorithm 2 are carried out using $n = 50$ and the iterations were observed to converge in less than 5 iterations. The history of the weights of the trained NNs is shown in Fig. 1. It depicts the time-dependent behavior of the elements of the weights throughout the horizon, i.e., from $t = 0$ to $t = 1\ s$. The NNs are utilized for controlling initial condition $x(0) = 2$, once the networks are trained. The results, including the histories of the state, the active mode, and the optimal control, are shown in Fig. 2. The state history shows that the controller has successfully driven the initial state to close to the desired terminal state

in the given time. The history of active modes shows that switching has happened exactly at the optimal times, considering the analytical optimal switching function given in (27).

An important feature of the developed method is providing optimal solution for different final times, without needing to retrain the networks. Let the new final time be $t_f = 0.5\ s$, i.e., the state should be brought to close to -2 in half of the previously selected final time. Note that once the optimal weights are available for $k \in [0,1,\dots,N]$, the optimal weights for the horizon of N_1 steps, where $N_1 < N$, are the last N_1 set of weights, i.e., they are given by W_k, and V_k^js where $k \in [N-N_1, N-N_1+1,\dots,N]$, due to Bellman principle of optimality [33]. Fig 3 shows the result of controlling initial condition $x(0) = 2$ with $t_f = 0.5\ s$. Interestingly the neurocontroller has successfully controlled the state to get to close to the desired terminal state in the shorter final time. To do this, a different control history and a different switching schedule are selected. The new active mode history, however, is still in accordance with the analytical $j^*(t)$ given in Eq. (27).

To further investigate the performance of the method, another initial condition, i.e., $x(0) = 0.5$ is selected and controlled using the same trained networks. The results are depicted in Fig. 4. Considering the resulting state history and the switching schedule shows that the controller is able to solve the problem of optimal switching for different initial conditions as well.

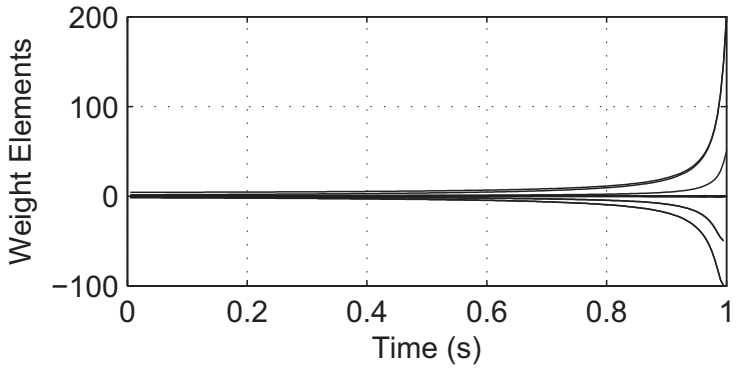

Fig 1: History of NN weights, Example 1.

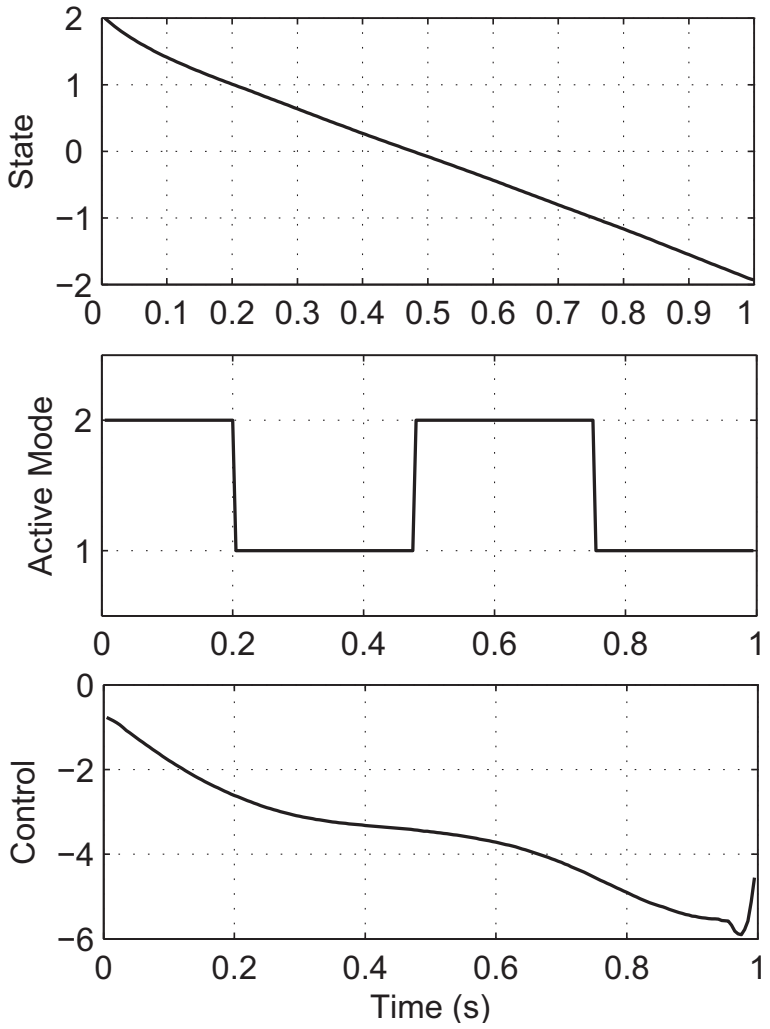

Fig 2: Simulation results for $x(0) = 2$ and $t_f = 1$, Example 1.

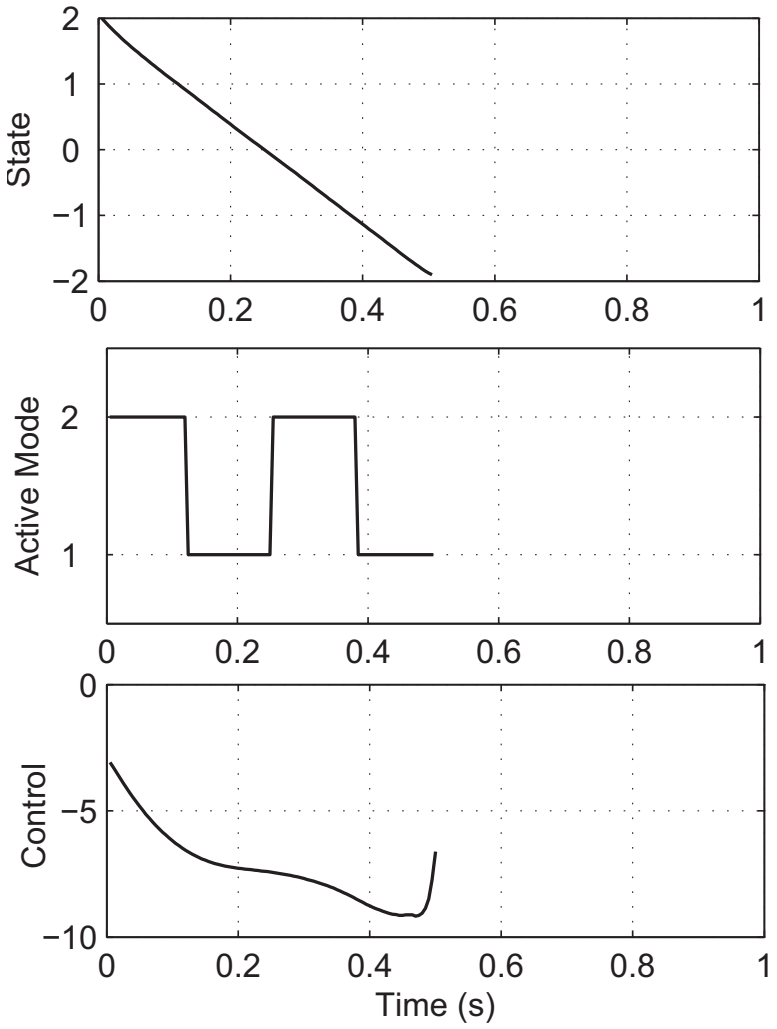

Fig 3: Simulation results for $x(0) = 2$ and $t_f = 0.5$, Example 1.

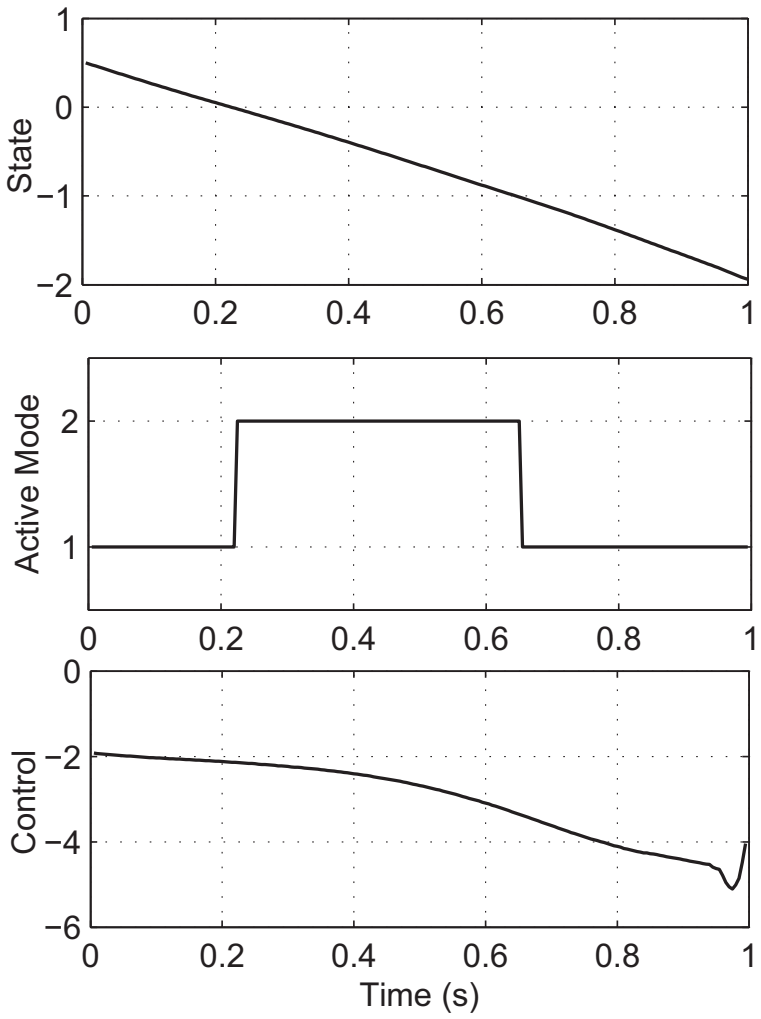

Fig 4: Simulation results for $x(0) = 0.5$ and $t_f = 1$, Example 1.

B. Example 2

The second example is a fourth order linear system which models the planar motion of a point mass in the absence of friction. The objective is moving the mass to the origin. The force, however, is limited to be applied either in the X or in the Y direction, where X and Y denote the perpendicular axis in the plane. Input $u(t)$ denotes the applied force and while its magnitude is subject to be calculated, its direction is limited to be parallel to the X or the Y axes. This problem is modeled as a switching problem where there are two modes; mode 1 in which the force steers the mass in the X direction and mode 2 in which the force steers it in the Y direction. The state vector is formed as $x = [x_1, x_2, x_3, x_4]^T$, where x_1 and x_2, respectively, are the X and Y positions and x_3 and

x_4 are the rates of change of x_1 and x_2, respectively. The model for the system is given by

$$\dot{x} = \begin{cases} f(x) + g_1(x)u \equiv [x_3, x_4, 0,0]^T + [0\ 0\ 1\ 0]^T u \\ f(x) + g_2(x)u \equiv [x_3, x_4, 0,0]^T + [0\ 0\ 0\ 1]^T u \end{cases}$$

Let the cost function be $J = 150x(t_f)^T x(t_f) + \int_0^{t_f} u(t)^2 dt$ where $t_f = 2\ s$. The domain of interest is selected as $\Omega = \{x \in \mathbb{R}^4 : |x_i| < 2.5, \forall i\}$.

Let the vector whose elements are all the *non-repeating* polynomials made up through multiplying the elements of vector $\bar{x}$ by those of vector $\bar{y}$ be denoted with $\bar{x} \otimes \bar{y}$. In this example the following basis functions are used:

$$\sigma(x) = [1, x, (x \otimes x)^T]^T,$$

$$\phi(x) = \left[1, x, (x \otimes x)^T, \left(x \otimes (x \otimes x)\right)^T\right]^T.$$

Using sampling time of $\Delta t = 0.005$, the horizon is discretized to 400 steps. The training is carried out using Algorithm 2 with $n = 200$ and the iterations were observed to converge in 4 iterations. Selecting initial condition of $x(0) = [0,2,2,0]^T$, the trained network is used for switching and control of the system and the results are given in Fig. 5. It can be seen that the controller has been able to move the mass to toward the origin through applying a history of force and performing a suitable switching between the applied directions.

The same problem is solved with the shorter final time of $t_f = 1.5$ and the results are given in Fig. 6. As expected, the trained network has been able to solve the problem with a new final time, as well. Fig. 7 shows the result of solving the problem with a new initial condition of $x(0) = [2,2,0,2]^T$ and the final time of $t_f = 2$. Interestingly, the controller has done a nice job in controlling the new initial condition, too.

Considering the switching schedule given in Fig. 5, it is observed that the controller exhibits high frequency switching between the modes in order to steer in the mass in the desired direction. If this behavior is unacceptable, one may apply one of the suggested remedies in subsection IV.C. As an example, the threshold remedy is simulated here to investigate its effect on the performance of the controller. The threshold was set as $\tau = 0.02$. Results of the first simulated problem in Example 2 are given in Fig. 8.

Comparing the state histories of Fig. 8 with those given in Fig. 5, it can be seen that the controller was able to bring the mass to the origin, while the number of switching is much less than in Fig. 5. Note that due to the coupling between the selected mode and the applied input control history given in Fig. 8, is different than the one given in Fig.5

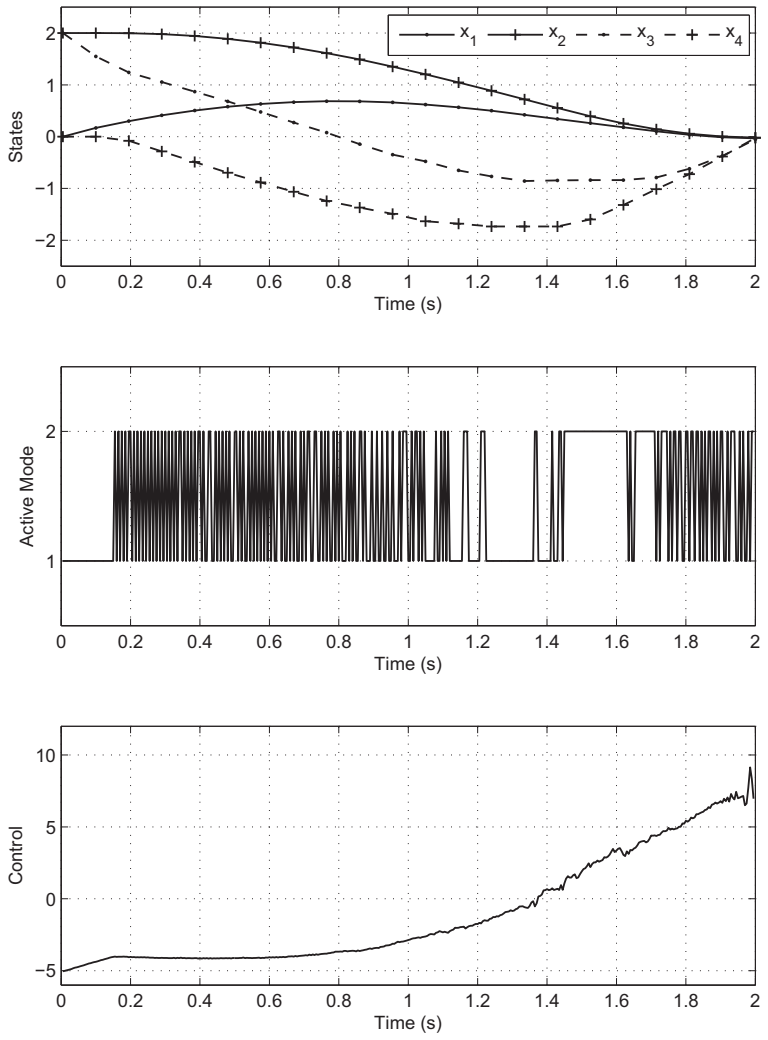

Fig 5: Simulation results for $x(0) = [0,2,2,0]^T$ and $t_f = 2$, Example 2.

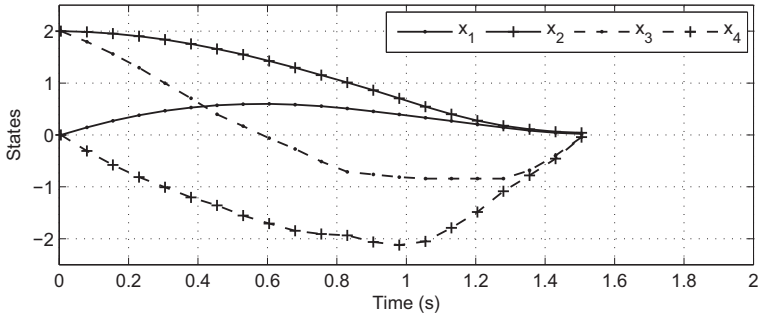

Fig 6: State histories for $x(0) = [0,2,2,0]^T$ and $t_f = 1.5$, Example 2.

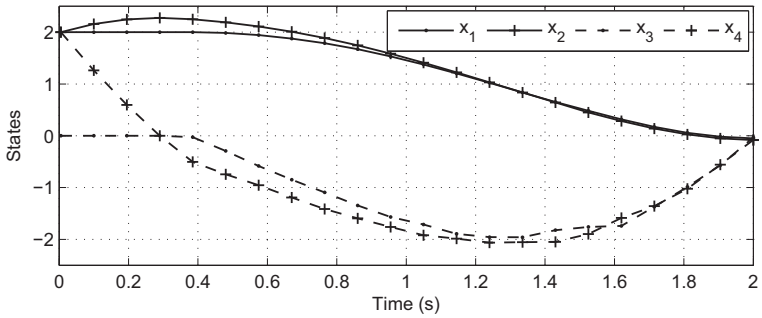

Fig 7: State histories for $x(0) = [2,2,0,2]^T$ and $t_f = 2$, Example 2.

VI. CONCLUSIONS

A method in the framework of approximate dynamic programming was developed for determining the optimal control and the optimal switching schedule for switching systems with controlled nonlinear subsystems and unspecified mode sequence. The performance of the method in solving problems with different initial conditions and different final times was investigated both analytically and numerically. These results and analyses lead one to conclude that the proposed method is versatile and is suitable for solving different real-world systems including aerospace, mechanical, and chemical problems.

ACKNOWLEDGEMENT

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

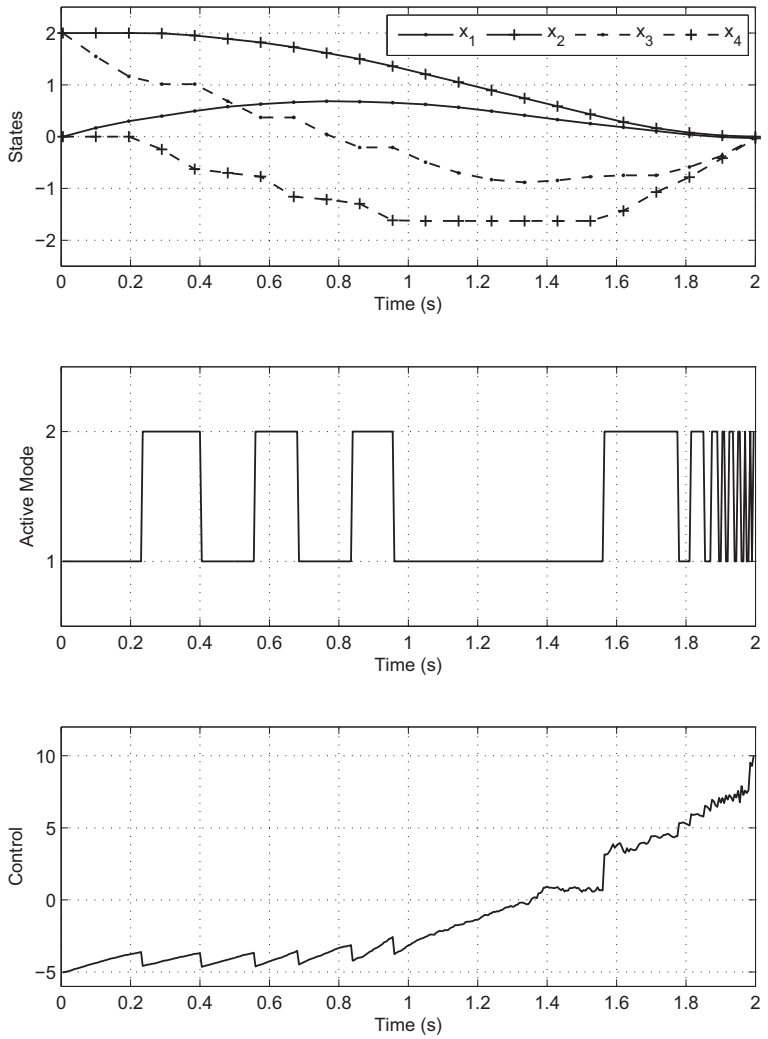

Fig 8: State histories for $x(0) = [0,2,2,0]^T$, $t_f = 2$, and threshold $\tau = 0.02$, Example 2.

APPENDIX A

In Algorithms 1 and 2, in different steps, different weight update rules for the weights of the actor and critic networks, i.e., V_k and W_k, are given. The least squares method can be used for rewriting these equations such that V_k and W_k are explicitly given based on the known parameters. In this appendix, the process for finding such an equation for V_k from Eq. (24) is explained and one can easily find the corresponding equation for W_k.

To perform least squares for the weight update of V_k^j, n random states denoted with $x^{[j]}$, where $1 \leq j \leq n$, are selected. Denoting the right hand side of Eq. (24) resulting from each $x^{[j]}$ with $Y(x^{[j]})$, the objective is finding V_k^j such that it solves

$$\begin{cases} {V_k^j}^T \sigma(x^{[1]}) = Y(x^{[1]}) \\ {V_k^j}^T \sigma(x^{[2]}) = Y(x^{[2]}) \\ \vdots \\ {V_k^j}^T \sigma(x^{[n]}) = Y(x^{[n]}) \end{cases} \tag{28}$$

Define

$$\boldsymbol{\sigma} \equiv \left[\sigma(x^{[1]}) \;\; \sigma(x^{[2]}) \;\; \ldots \;\; \sigma(x^{[n]})\right]$$

$$\boldsymbol{Y} \equiv \left[Y(x^{[1]}) \;\; Y(x^{[2]}) \;\; \ldots \;\; Y(x^{[n]})\right]$$

Using the method of least squares, solution to the system of linear equations (28) is given by

$$V_k^j = (\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\boldsymbol{Y}^T \tag{29}$$

Note that for the inverse of matrix $(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)$, which is a $p \times p$ matrix, to exist, one needs the basis functions $\sigma(.)$ to be linearly independent and n to be greater than or equal to the number of the basis functions.

APPENDIX B

This appendix includes the proofs of Theorems 1 and 2.

Proof of Theorem 1: The iteration performed on $V_k^{j,(i)}$, given in (24) and repeated here, is a successive approximation to find a fixed point of a function

$${V_k^{j,(i+1)}}^T \sigma\left(x_k^{[j]}\right) \cong -\frac{1}{2} R^{-1} g_j\left(x_k^{[j]}\right)^T \nabla\phi\left(f_j\left(x_k^{[j]}\right) + g_j\left(x_k^{[j]}\right) {V_k^{j,(i)}}^T \sigma\left(x_k^{[j]}\right)\right)^T W_{k+1},$$

i.e., there exists function $\mathcal{F}$: $\mathbb{R}^{p\times m} \to \mathbb{R}^{p\times m}$ such that (24) is of form

$$V_k^{j,(i+1)} = \mathcal{F}(V_k^{j,(i)}). \tag{30}$$

The claim of the theorem is proved if it can be shown that (30) is a contraction mapping [37]. Since $\mathbb{R}^{p\times m}$ with 2-norm denoted by $\|.\|$ is a Banach space, iterations given by

(30), regardless of initial $V_k^{j,(0)}$, converges to some $V_k^j = \mathcal{F}(V_k^j)$ if there exists a $0 \leq \rho < 1$ such that for every U_1 and U_2 in $\mathbb{R}^{p\times m}$, the following inequality holds [37]

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \leq \rho\|U_1 - U_2\|. \tag{31}$$

Function $\mathcal{F}(.)$ can be formed by converting (24) to a least squares form performed in Appendix A. Rewriting Eq. (29), given in Appendix A, leads to

$$\mathcal{F}(V_k^{j,(i)}) \equiv$$
$$(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\begin{bmatrix} \left(-\frac{1}{2}R^{-1}g_j\left(x_k^{[1]}\right)^T \nabla\phi\left(f_j\left(x_k^{[1]}\right) + g_j\left(x_k^{[1]}\right)V_k^{j,(i)^T}\sigma\left(x_k^{[1]}\right)\right)^T W_{k+1}\right)^T \\ \left(-\frac{1}{2}R^{-1}g_j\left(x_k^{[2]}\right)^T \nabla\phi\left(f_j\left(x_k^{[2]}\right) + g_j\left(x_k^{[2]}\right)V_k^{j,(i)^T}\sigma\left(x_k^{[2]}\right)\right)^T W_{k+1}\right)^T \\ \vdots \\ \left(-\frac{1}{2}R^{-1}g_j\left(x_k^{[n]}\right)^T \nabla\phi\left(f_j\left(x_k^{[n]}\right) + g_j\left(x_k^{[n]}\right)V_k^{j,(i)^T}\sigma\left(x_k^{[n]}\right)\right)^T W_{k+1}\right)^T \end{bmatrix} \tag{32}$$

One has

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \leq$$
$$\sqrt{n}\|(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\|\|\frac{1}{2}R^{-1}g_j\left(x_k^{[\ell]}\right)^T \nabla\phi\left(f_j\left(x_k^{[\ell]}\right) + g_j\left(x_k^{[\ell]}\right)U_1^T\sigma\left(x_k^{[\ell]}\right)\right)^T W_{k+1} -$$
$$\frac{1}{2}R^{-1}g_j\left(x_k^{[\ell]}\right)^T \nabla\phi\left(f_j\left(x_k^{[\ell]}\right) + g_j\left(x_k^{[\ell]}\right)U_2^T\sigma\left(x_k^{[\ell]}\right)\right)^T W_{k+1}\| \tag{33}$$

where integer ℓ, $0 \leq \ell \leq n$, is given by

$$\ell = \underset{1\leq i\leq n}{argmax}\,\|\frac{1}{2}R^{-1}g_j\left(x_k^{[i]}\right)^T \nabla\phi\left(f_j\left(x_k^{[i]}\right) + g_j\left(x_k^{[i]}\right)U_1^T\sigma\left(x_k^{[i]}\right)\right)^T W_{k+1} -$$
$$\frac{1}{2}R^{-1}g_j\left(x_k^{[i]}\right)^T \nabla\phi\left(f_j\left(x_k^{[i]}\right) + g_j\left(x_k^{[i]}\right)U_2^T\sigma\left(x_k^{[i]}\right)\right)^T W_{k+1}\|.$$

In inequality (33), the following norm inequality is used

$$\left\|\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}\right\| \leq \sqrt{n}\|y_\ell\| \tag{34}$$

where y_is are real-valued row-vectors and $\ell = \underset{1\leq i\leq n}{argmax}\|y_i\|$.

Smoothness of $\phi(.)$ leads to the Lipschitz continuity of $\nabla\phi(.)$ on compact set Ω [40]. Therefore, there exists some positive real number ρ_ϕ such that for every x_1 and x_2 in Ω, one has $\|\nabla\phi(x_1) - \nabla\phi(x_2)\| \leq \rho_\phi\|x_1 - x_2\|$. Using this feature of $\nabla\phi(.)$, inequality (33) can be written as

$$\|\mathcal{F}(U_1) - \mathcal{F}(U_2)\| \leq$$
$$\rho_\phi\sqrt{n}\|(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\|\|\frac{1}{2}R^{-1}g_j\left(x_k^{[\ell]}\right)^T\|\|g_j\left(x_k^{[\ell]}\right)\|\|\sigma\left(x_k^{[\ell]}\right)\|\|W_{k+1}\|\|(U_1^T - U_2^T)\| \quad (35)$$

By defining

$$\rho \equiv \rho_\phi\sqrt{n}\|(\boldsymbol{\sigma}\boldsymbol{\sigma}^T)^{-1}\boldsymbol{\sigma}\|\|\frac{1}{2}R^{-1}g_j\left(x_k^{[\ell]}\right)^T\|\|g_j\left(x_k^{[\ell]}\right)\|\|\sigma\left(x_k^{[\ell]}\right)\|\|W_{k+1}\| \quad (36)$$

one can select the sampling time Δt in discretization of the continuous dynamics (1) small enough such that the condition $0 \leq \rho < 1$ is satisfied, since a smaller Δt, directly results in a smaller $\|g_j\left(x_k^{[\ell]}\right)\|$ while the other terms including $\|\frac{1}{2}R^{-1}g_j\left(x_k^{[\ell]}\right)^T\|$ are not affected. Note that smoothness, and hence continuity, of $g_j(.)$s and $\sigma(.)$ in their domain results in being bounded in the compact set Ω [41], therefore, the $x_k^{[\ell]}$ dependent terms in (36) are upper bounded.

The expression given for the contraction mapping coefficient ρ in (36) involves $\|W_{k+1}\|$ also. It should be noted that W_{k+1} is already learned from the previous step in the algorithm, therefore, it is bounded. In other words, starting from $k = N - 1$, one uses the successive approximation given by (24) and once $V_k^{j,(i)}$ converges, it is used in (23) to calculate the bounded W_k. This process is repeated till $k = 0$.

Note that if the selected sampling time Δt is not small enough, at some k, $0 \leq k \leq N - 1$, the respective ρ given in (36) does not satisfy condition $0 \leq \rho < 1$, therefore, $V_k^{j,(i)}$ does not converge as $i \to \infty$. In that case, one may select a smaller sampling time and restart the algorithm, i.e., from $k = N - 1$ to calculate the weights corresponding to the smaller sampling time. Refining the sampling time leads to a change in W_{k+1} as well. However, it can be shown that as the sampling time becomes smaller, W_{k+1} remains bounded. This boundedness follows from looking at the definition of W_{k+1}, which is the weights for the network that approximates a discretized cost-to-go. In other words,

$$W_{k+1}^T \phi(x) \cong \psi(x_N) + \sum_{\hbar=k+1}^{N-1} \left(Q(x_\hbar) + u_\hbar^T R u_\hbar \right). \tag{37}$$

As the sampling times go to zero, the value of the discretized cost-to-go converges to the cost-to-go given by $\psi\left(x\left(t_f\right)\right) + \frac{1}{2}\int_{(k+1)\Delta t}^{t_f} \left(Q\left(x(t)\right) + u(t)^T R u(t) \right) dt$. On the other hand, since the system does not have a finite-scape time (which follows from smoothness on the compact domain of interest,) the finite-horizon cost-to-go will be finite, using any finite control. Note that the control history included in the integration given in (37) correspond to the already converged time-steps, hence, they are bounded. Therefore, as $\Delta t \to 0$, the value of $W_{k+1}^T \phi(x)$ will be finite. Since the basis functions $\phi(x)$ are linearly independent, a finite $W_{k+1}^T \phi(x)$ leads to a finite W_{k+1}^T, as seen in the least squares operation described in Appendix A. Therefore, term $\|W_{k+1}\|$ existing in the expression for ρ in (36) remains bounded as the sampling time is refined. This completes the proof of convergence of $V_k^{j,(i)}$ to V_k^j for $0 \le k < N-1$ using any initial guess on $V_k^{j,(0)}$, for any small enough sampling time. ■

Proof of Theorem 2: Let $\bar{x}$ be any point in 1Ω and set

$$\bar{J} = j_k^*(\bar{x}). \tag{38}$$

Select an open set $\alpha \subset \Omega$ such that $\bar{x}$ belongs to the boundary of α and limit

$$\hat{J} = \lim_{\substack{\|x-\bar{x}\|\to 0 \\ x\in\alpha}} j_k^*(x) \tag{39}$$

exists. If $\bar{J} = \hat{J}$, for every such α, then there exists some open set $\beta \subset \Omega$ containing $\bar{x}$ such that $j_k^*(x)$ is constant for all $x \in \beta$, because $j_k^*(x)$ only assumes integer values. In this case the continuity of $F^{j_k^*(x)}(x)$ at $\bar{x}$ follows from the fact that $F^j(x)$ is continuous at $\bar{x}$, for every fixed $j \in \mathcal{J}$. Finally, the continuity of the function subject to investigation at every $\bar{x} \in \Omega$, leads to the continuity of the function in Ω.

Now assume $\bar{J} \neq \hat{J}$, for some α. From the continuity of $F^{\hat{J}}(x)$ at $\bar{x}$, for the given $\hat{J}$, one has

$$F^{\hat{J}}(\bar{x}) = \lim_{\delta x\to 0} F^{\hat{J}}(\bar{x} + \delta x). \tag{40}$$

If it can be shown that for every selected α, one has

$$F^{\bar{J}}(\bar{x}) = F^{\hat{J}}(\bar{x}), \tag{41}$$

then the continuity of $F^{j_k^*(x)}(x)$ at $\bar{x}$ follows, because from (41) and (40) one has

$$F^{\bar{J}}(\bar{x}) = \lim_{\delta x \to 0} F^{\hat{J}}(\bar{x} + \delta x), \tag{42}$$

and (42) leads to the continuity by definition [41]. The proof that (41) holds is done by contradiction. Assume that

$$F^{\bar{J}}(\bar{x}) < F^{\hat{J}}(\bar{x}), \tag{43}$$

then, due to the continuity of $F^{\bar{J}}(x)$ and $F^{\hat{J}}(x)$ at $\bar{x}$, there exists an open set γ containing $\bar{x}$, such that

$$F^{\bar{J}}(x) < F^{\hat{J}}(x),\ \forall x \in \gamma. \tag{44}$$

On the other hand, Eq. (39) implies that there exists a neighborhood of $\bar{x}$ at which $\hat{J} = j_k^*(x)$, hence, because $\bar{x} \in \gamma$, one has

$$F^{\bar{J}}(x) \geq F^{\hat{J}}(x), \exists x \in \gamma. \tag{45}$$

But, (45) contradicts (44). Hence, (43) is not possible. The impossibility of

$$F^{\bar{J}}(\bar{x}) > F^{\hat{J}}(\bar{x}) \tag{46}$$

directly follows from (38). Because if (46) holds then $\bar{J} \neq j_k^*(\bar{x})$, which is against (38). Therefore, equality (41) holds and hence, $F^{j_k^*(x)}(x)$ is continuous at every $\bar{x} \in \Omega$. This completes the proof. ■

REFERENCES

[1] Soler, M., Olivares, A., and Staffetti, E., "Framework for aircraft trajectory planning toward an efficient air traffic management," *Journal of Aircraft*, vol. 49 (1), pp.985-991, 2012.

[2] Benmansour, K., Benalia, A., Djemaï, M., and de Leon, J., "Hybrid control of a multicellular converter," *Nonlinear Analysis: Hybrid Systems*, vol. 1 (1), pp.16-29, 2007.

[3] Hernandez-Vargas, E. A., Middleton, R. H., Colaneri, P., and Blanchini, F., "Dynamic optimization algorithms to mitigate HIV escape," in Proc *IEEE Conference on Decision and Control*, 2010, pp. 827-832.

[4] Gong, Z., Liu, C., Feng, E., Wang, L., Yu, Y., "Modelling and optimization for a switched system in microbial fed-batch culture," *Applied Mathematical Modelling*, vol. 35 (7), pp.3276-3284, 2011.

[5] Rinehart, M., Dahleh, M., Reed, D., Kolmanovsky, I., "Suboptimal control of switched systems with an application to the DISC engine," *IEEE Transactions on Control Systems Technology*, vol. 16 (2), pp.189-201, 2008.

[6] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions," *International Journal of Control*, vol. 75 (16), pp. 1406-1426, 2002.

[7] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Trans. on Automatic Control*, vol. 49 (1), pp.2- 16, 2004.

[8] Egerstedt, M., Wardi, Y., and Axelsson, H., "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. on Automatic Control*, vol. 51 (1), pp.110-115, 2006.

[9] Axelsson, H., Boccadoro, M., Egerstedt, M., Valigi, P., and Wardia, Y., "Optimal mode-switching for hybrid systems with varying initial states," *Nonlinear Analysis: Hybrid Systems*, vol. 2 (3), pp.765–772, 2008.

[10] Ding, X., Schild, A., Egerstedt M., and Lunze J., "Real-time optimal feedback control of switched autonomous systems," Proc. *IFAC Conference on Analysis and Design of Hybrid Systems*, pp.108-113, 2009.

[11] Kamgarpoura, M., and Tomlin, C., "On optimal control of non-autonomous switched systems with a fixed mode sequence," *Automatica*, vol. 48, pp.1177–1181, 2012.

[12] Zhao, R., and Li, S., "Switched system optimal control based on parameterizations of the control vectors and switching instant," Proc. *Chinese Control and Decision Conference*, pp. 3290-3294, 2011.

[13] Luus, R., and Chen, Y, "Optimal switching control via direct search optimization," *Asian Journal of Control*, Vol. 6 (2), pp. 302-306, 2004.

[14] Rungger, M., and Stursberg, O., "A numerical method for hybrid optimal control based on dynamic programming," *Nonlinear Analysis: Hybrid Systems*, vol. 5 (2), pp.254–274, 2011.

[15] Sakly, M., Sakly, A., Majdoub, N., and Benrejeb, M., "Optimization of switching instants for optimal control of linear switched systems based on genetic algorithms," Proc. *IFAC Int. Conf. Intelligent Control Systems and Signal Processing*; Istanbul, 2009.

[16] Long, R., Fu, J., Zhang, L., "Optimal control of switched system based on neural network optimization," Proc. *Int. Conference on Intelligent Computing*, pp.799-806, 2008.

[17] Zhang, W., Hu, J., and Abate, A., "On the value functions of the discrete-time switched LQR problem," *IEEE Trans. on Automatic Control*, vol. 54 (11), pp.2669-2674, 2009.

[18] Heydari, A., and Balakrishnan, S. N., "Optimal multi-therapeutic HIV treatment using a global optimal switching scheme," *Applied Mathematics and Computation*, Vol. 219, pp. 7872-7881, 2013.

[19] Werbos, P.J., "Approximate dynamic programming for real-time control and neural modeling". In White D.A., & Sofge D.A (Eds.), *Handbook of Intelligent Control*, Multiscience Press, 1992.

[20] Balakrishnan, S.N., and Biega, V., "Adaptive-critic based neural networks for aircraft optimal control", *Journal of Guidance, Control & Dynamics*, Vol. 19, No. 4, 1996, pp. 893-898.

[21] Prokhorov, D.V., and Wunsch, D.C. II, "Adaptive critic designs," *IEEE Trans. Neural Networks*, Vol. 8, No. 5, 1997, pp. 997-1007.

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

[23] Dierks, T., Thumati, B. T., and Jagannathan, S., "Optimal control of unknown affine nonlinear discrete-time systems using offline-trained neural networks with proof of convergence," *Neural Networks*, vol. 22, pp. 851-860, 2009.

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

[25] Venayagamoorthy, G. K., Harley, R. G., and Wunsch, D. C., "Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator," *IEEE Trans. Neural Netw.*, vol. 13 (3), pp. 764-773, 2002.

[26] Padhi, R., Unnikrishnan, N., Wang, X., and Balakrishnan, S. N., "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.

[27] Ding, J. and Balakrishnan, S. N., "An online nonlinear optimal controller synthesis for aircraft with model uncertainties," in Proc. *AIAA Guidance, Navigation, and Control Conference*, 2010.

[28] Han, D. and Balakrishnan, S.N., "State-constrained agile missile control with adaptive-critic-based neural networks," *IEEE Trans. Control Systems Technology*, Vol. 10, No. 4, 2002, pp. 481-489.

[29] Heydari, A., and Balakrishnan, S.N., "Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24 (1), 2013, pp. 145-157.

[30] Wang, F., Jin, N., Liu, D., and Wei, Q., "Adaptive dynamic programming for finite-horizon optimal control of discrete-time nonlinear systems with ε-error bound," *IEEE Trans. Neural Netw*., vol. 22 (1), pp. 24-36, 2011.

[31] Song, R., and Zhang, H., "The finite-horizon optimal control for a class of time-delay affine nonlinear system," *Neural Comput & Applic*, DOI 10.1007/s00521-011-0706-3, 2011.

[32] Wei, Q., and Liu, D., "An iterative ϵ-optimal control scheme for a class of discrete-time nonlinear systems with unfixed initial state," *Neural Networks*, vol. 32, pp. 236-244, 2012.

[33] Kirk, D. E., *Optimal control theory: an introduction*, Dover Publications, 2004, pp. 53-58.

[34] Homik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," NeuralNetworks, vol. 2, pp. 359-366, 1989.

[35] Attali, J. G. and Pages, G., "Approximations of Functions by a Multilayer Perceptron: a New Approach," *Neural Networks*, vol. 10 (6), pp. 1069-1081, 1997.

[36] Stone, M. and Goldbart, P., *Mathematics for Physics - A Guided Tour for Graduate Students*, Cambridge, England, Cambridge University Press, p. 70, 2009

[37] Khalil, H., *Nonlinear Systems*, 3rd ed., Prentice Hall, New Jersey, 2002.

[38] Heydari, A., and Balakrishnan, S.N., "Global Optimality of Approximate Dynamic Programming and its use un Non-Convex Function Minimization," to be submitted to *Automatica*.

[39] Boyd, S., and Vandenberghe, L.*, Convex Optimization*, Cambridge University Press, 2009, pp. 4-7,71,459.

[40] Marsden J. E., Ratiu T., and Abraham R., *Manifolds, Tensor Analysis, and Applications*, 3rd ed. Springer-Verlag Publishing Co., New York, 2001, p. 73.

[41] Trench, W. F., *Introduction to Real Analysis*, available online at http://ramanujan.math.trinity. edu/wtrench/texts/trench_real_analysis.pdf, 2012, pp. 309,313.

8. OPTIMAL SWITCHING OF NONLINEAR SYSTEMS WITH MODELING UNCERTAINTY

Ali Heydari and S. N. Balakrishnan

ABSTRACT

The problem of infinite-horizon optimal switching of nonlinear systems with modeling uncertainty is investigated where the mode sequence as well as the number of switching is free. An approximate dynamic programming based scheme is developed for solving the problem. The proposed method utilizes a nominal model of the switching system for offline training and once applied on the system, learns the unmodeled dynamics in real-time. A numerical example demonstrates the capability of the developed solution in a second order switching system with three subsystems.

I. INTRODUCTION

Many real-world systems are classified as switching systems in which different modes of operation are available and the controller needs to decide between the modes to activate one at each instant. As a short survey on the literature, formulating the problem as a nonlinear programming problem with *preselected* initial conditions, mode sequence, and number of switching was investigated in [1-7]. The discretization of both the state and input space was used for optimal switching through dynamic programming in [8]. Genetic algorithm and neural networks were used in [9] and [10], respectively, to determine the optimal switching for a preselected initial condition within intelligent methods. Two approximate dynamic programming (ADP) based schemes were proposed in [11] and [12] by the authors of this study, for finite-horizon optimal switching of systems with *fixed* and *free* mode sequences, respectively. These developments provide solution for a *vast domain of initial conditions*.

To the best of authors' knowledge, the available methods in the literature require a *perfect model* of the system ahead of the implementation time, for calculation of the solution. In practice, however, modeling uncertainties are ubiquitous. This fact gives rise to the need for developing a scheme for *online* calculation of the optimal switching schedule based on the *actual* dynamics of the subsystems. This problem is investigated here.

The motivation for the solution proposed in this study comes from the studies in ADP [13,14] in which the optimal cost-to-go is learned as a function of the states for conventional optimal control problems. The idea proposed in [12] for *finite-horizon* optimal switching is extended to *infinite-horizon* problems initially. It is shown that once the optimal cost-to-go function is approximated, the optimal switching solution can be determined in real-time through a simple equation that requires the model of the subsystems as well as the cost-to-go function. Afterwards, an online training phase is proposed for capturing the effect of unmodeled dynamics on the cost-to-go approximator and also for identifying the unmodeled dynamics, motivated by the work in [15] for conventional optimal control problems. In other words, a neural network (NN) is trained offline based on imprecise models of the subsystems and then it is utilized in the online operation of the system in which, the actual dynamics of the subsystems are captured and the network is re-trained based on the system's output to generate the optimal cost-to-go and hence, the optimal switching schedule. Besides solving the problems with modeling uncertainty, an important feature of the method is providing solution for different initial conditions. Moreover, the mode sequence and the number of switching are subject to be determined optimally.

This article is organized as follows. Problem formulation is presented in section II. The solution for the case of no modeling uncertainty is given in section III. The idea for handling modeling uncertainties is discussed in section IV. Simulation study is given in section V, followed by concluding remarks in section VI.

II. PROBLEM FORMULATION

A discrete-time switching system with autonomous subsystems can be represented by a set of M subsystems/modes:

$$x_{k+1} = f_i(x_k), k \in K, i \in I, \tag{1}$$

where $f_i: \mathbb{R}^n \to \mathbb{R}^n$, K denotes the set of non-negative integers, $I \equiv \{1, 2, \dots, M\}$, and n denotes the dimension of the state vector x_k. Subscript k in x_k denotes the discrete time index. Moreover, subscript i in f_i denotes the index of the active subsystem. At each instant k, only one subsystem can be active. A controller for the system is defined as a switching sequence that allows the system to operate. The optimal solution, however, is

defined as a switching schedule using which, the infinite-horizon cost function given below is optimized.

$$J = \sum_{k=0}^{\infty} Q(x_k) \tag{2}$$

Convex positive semi-definite function $Q: \mathbb{R}^n \to \mathbb{R}$ penalizes the deviation of the states from the desired values. Denoting the index of active subsystem at time k with i_k, the optimal solution may be denoted with $i_k^* \in I, \forall k \in K$. The mode sequence, the number of switching, and the switching instant are each subject to be determined such that the cost function is optimized.

III. SOLUTION PROCESS WITHOUT MODELING UNCERTAINTY

Denoting the cost-to-go at current state x_k by $J(x_k)$ leads to

$$J(x_k) = \sum_{j=k}^{\infty} Q(x_j). \tag{3}$$

Note that, from the form of the cost function, it directly follows that

$$J(x_k) = Q(x_k) + J(x_{k+1}), \forall k \in K. \tag{4}$$

Based on the Bellman principle of optimality [16], regardless of what decisions are made for the past, the optimal solution is a solution which optimizes the future. Therefore, regardless of values selected for i_j, $j \in \{0,1,\dots,k-1\}$, the optimal solution for the remained time steps, i.e., $j \in \{k, k+1, \dots\}$ is the solution which optimizes $J(x_k)$. From (4), optimizing $J(x_k)$ is equivalent of optimizing $J(x_{k+1})$, because term $Q(x_k)$ does not depend on the selection of i_j, $j \in \{k, k+1, \dots\}$. The idea is approximating the optimal cost-to-go $J^*(x_k)$ versus x_k. Once this function is available, the *online* optimal solution can be calculated at each instant k and state vector x_k using

$$i^*(x_k) = argmin_{i \in I} J^*(f_i(x_k)). \tag{5}$$

For example, if the system has two subsystems, finding optimal solution at each instant k simplifies to evaluating the scalar-valued function $J^*(f_i(x_k))$ for $i = 1$ and $i = 2$ and selecting the i for which $J^*(f_i(x_k))$ is smaller. This calculation needs to be done online at each instant k for $k \in K$. Therefore, the optimal solution will be calculated in real-time and in a feedback form. The following subsections provide the process for

learning $J^*(x_k)$. For the process, the following assumption is needed to guarantee the finiteness of the optimal cost-to-go.

Assumption 1: The method developed in this study assumes that there exists a switching schedule for which the cost function remains finite.

A. Cost-to-go Function Approximation

In this subsection the process of learning the cost-to-go for a switching system is explained. In order to motivate the idea, initially the case of conventional systems, i.e., non-switching systems, is discussed and an algorithm is proposed for learning the cost-to-go function. Afterward, the algorithm is modified to learn the cost-to-go function for the switching system subject to this study.

A.1. Cost-to-go approximation for a conventional system

Let the dynamics of the system be

$$x_{k+1} = f(x_k), k \in K, \tag{6}$$

where $f: \mathbb{R}^n \to \mathbb{R}^n$ is the sole mode of the system. Note that, the system does not include a control or switching. However, the cost-to-go at current state x_k, i.e., $J(x_k)$ can be calculated using cost function (2). The objective is approximating function $J(x_k)$ as a function of x_k. An algorithm is suggested for learning the cost-to-go function in this subsection. The algorithm trains an NN as a global function approximator for the purpose. The concept is motivated by the notion of Heuristic Dynamic Programming (HDP) [13,14] for infinite-horizon optimal control of conventional systems. In the HDP scheme, the so called critic network learns the optimal cost-to-go, and the so called actor learns the optimal control. In this study, the actor is skipped. The critic is utilized to learn the cost-to-go versus x_k for nonlinear system (6). Selecting a linear in the parameter NN as the function approximator, the expressions for the critic (cost-to-go approximator) can be written as

$$J(x_k) \cong W^T \phi(x_k), k \in K, \tag{7}$$

where $W \in \mathbb{R}^m$ is the unknown optimal weight matrix of the network. The selected basis functions are given by $\phi: \mathbb{R}^n \to \mathbb{R}^m$, with m being a positive integer denoting the number of neurons. The training process for learning the optimal weight matrix W is detailed in Algorithm 1.

Algorithm 1

Step 1: Select an initial weight matrix for the NN.

Step 2: Randomly select state vector $x \in \Omega$, where Ω represents the domain of interest.

Step 3: Calculate the training target J^t as

$$J^t = Q(x) + W^T \phi\big(f(x)\big). \tag{8}$$

Step 4: Train the weights based on the input-target pair (x, J^t).

Step 5: Repeat Steps 2 to 4 until W converges for different random xs.

A.2. Cost-to-go approximation for a switching problem

Considering the cost-to-go approximation method discussed in the foregoing subsection, the same concept may be adapted for approximating the optimal cost-to-go of switching system (1). Note that once the i_k^*, $\forall k$, is found for a given initial condition x_0, system (1) simplifies to a conventional system with a nonlinear time-varying dynamics due to the frozen switching. Therefore, a NN can be used to learn its cost-to-go. Assuming the network structure (7), Algorithm 2 is proposed for learning the optimal cost-to-go function in a closed form.

Algorithm 2

Step 1: Select an initial weight matrix for the NN.

Step 2: Randomly select state vector $x \in \Omega$.

Step 3: Calculate

$$i^*(x) = argmin_{i \in I}\, W^T \phi\big(f_i(x)\big). \tag{9}$$

Step 4: Calculate the training target J^t

$$J^t = Q(x) + W^T \phi\left(f_{i^*(x)}(x)\right). \tag{10}$$

Step 5: Train the weights based on the input-target pair (x, J^t).

Step 6: Repeat Steps 2 to 5 until W converges for different random xs.

Assuming the basis functions of the NN are selected rich enough to approximate the cost-to-go function with the desired accuracy, the method developed here provides optimal solution due to its basis on Dynamic Programming [16]. In other words, if $J^*(x_k)$

is approximated and available, the optimal mode will always be given through (5). Therefore, an analysis on the approximation capability of the NN is required. It is well known that NNs can provide uniform approximation within the domain of interest providing the function subject to approximation in a continuous function. Interested readers are referred to [17] and [18] for multi-layer NNs and linear in parameter NNs with polynomial basis functions, respectively. Considering Eq. (10), due to the switching between the modes, i.e., the discontinuous nature of $i^*(.)$, the continuity of the right hand side is not obvious. Theorem 1 proves the required continuity.

Theorem 1: If the active mode for the given state vector x is given by (9), then scalar-valued function $W^T\phi\left(f_{i^*(x)}(x)\right)$ is a continuous function versus x at every $x \in \Omega$.

Proof: Let $\bar{x}$ be any point in Ω and set

$$\bar{\imath} = i^*(\bar{x}). \tag{11}$$

Select an open set $\alpha \subset \Omega$ such that $\bar{x}$ belongs to the boundary of α and limit

$$\hat{\imath} = \lim_{\substack{\|x-\bar{x}\|\to 0 \\ x\in\alpha}} i^*(x) \tag{12}$$

exists. If $\bar{\imath} = \hat{\imath}$, for every such α, then there exists some open set $\beta \subset \Omega$ containing $\bar{x}$ such that $i^*(x)$ is constant for all $x \in \beta$, because $i^*(x)$ only assumes integer values. In this case the continuity of $W^T\phi\left(f_{i^*(x)}(x)\right)$ at $\bar{x}$ follows from the fact that $W^T\phi\big(f_i(x)\big)$ is continuous at $\bar{x}$, for every fixed $i \in I$. Finally, the continuity of the function subject to investigation at every $\bar{x} \in \Omega$, leads to the continuity of the function in Ω.

Now assume $\bar{\imath} \neq \hat{\imath}$, for some α. From the continuity of $W^T\phi\big(f_{\hat{\imath}}(x)\big)$ at $\bar{x}$, for the given $\hat{\imath}$, one has

$$W^T\phi\big(f_{\hat{\imath}}(\bar{x})\big) = \lim_{\delta x\to 0} W^T\phi\big(f_{\hat{\imath}}(\bar{x}+\delta x)\big). \tag{13}$$

If it can be shown that for every selected α, one has

$$W^T\phi\big(f_{\bar{\imath}}(\bar{x})\big) = W^T\phi\big(f_{\hat{\imath}}(\bar{x})\big), \tag{14}$$

then the continuity of $W^T\phi\left(f_{i^*(x)}(x)\right)$ at $\bar{x}$ follows, because from (14) and (13) one has

$$W^T\phi\big(f_{\bar{\imath}}(\bar{x})\big) = \lim_{\delta x\to 0} W^T\phi\big(f_{\hat{\imath}}(\bar{x}+\delta x)\big), \tag{15}$$

and (15) leads to the continuity by definition [21]. The proof that (14) holds is done by contradiction. Assume that for some $\bar{x}$ and some α one has

$$W^T\phi\left(f_{\bar{\imath}}(\bar{x})\right) < W^T\phi\left(f_{\hat{\imath}}(\bar{x})\right), \tag{16}$$

then, due to the continuity of $W^T\phi\left(f_{\bar{\imath}}(\bar{x})\right)$ and $W^T\phi\left(f_{\hat{\imath}}(\bar{x})\right)$ at $\bar{x}$, there exists an open set γ containing $\bar{x}$, such that

$$W^T\phi\left(f_{\bar{\imath}}(x)\right) < W^T\phi\left(f_{\hat{\imath}}(x)\right), \forall x \in \gamma. \tag{17}$$

On the other hand, Eq. (12) implies that there exists a neighborhood of $\bar{x}$ at which $\hat{\imath} = i^*(x)$, hence, because $\bar{x} \in \gamma$, one has

$$W^T\phi\left(f_{\bar{\imath}}(x)\right) \geq W^T\phi\left(f_{\hat{\imath}}(x)\right), \exists x \in \gamma. \tag{18}$$

But, (18) contradicts (17). Hence, (16) is not possible. The impossibility of

$$W^T\phi\left(f_{\bar{\imath}}(\bar{x})\right) > W^T\phi\left(f_{\hat{\imath}}(\bar{x})\right) \tag{19}$$

directly follows from (11). Because if (19) holds then $\bar{\imath} \neq i^*(\bar{x})$, which is against (11). Therefore, equality (14) holds and hence, $W^T\phi\left(f_{i^*(x)}(x)\right)$ is continuous at every $\bar{x} \in \Omega$. This completes the proof. ■

The point which leads to the result given in Theorem 1 is the fact that $i^*(.)$ is defined by the 'argmin' function given in (9). Even though $i^*(x)$ could discontinuously change as x does, function $W^T\phi\left(f_{i^*(x)}(x)\right)$ will be continuous at the continuous and discontinuous points of $i^*(x)$. In order to better understand this point, one may consider the example of having two subsystems with scalar dynamics. Assume the cost-to-go of utilizing each subsystem, given by $W^T\phi\left(f_i(x)\right)$, $i = 1,2$, changes linearly versus x as given in Fig. 1. In this case, function $W^T\phi\left(f_{i^*(x)}(x)\right)$ will be given by the solid plots in the figure. As seen, the jump of $i^*(x)$ from one value to another, does not create any discontinuity in $W^T\phi\left(f_{i^*(x)}(x)\right)$.

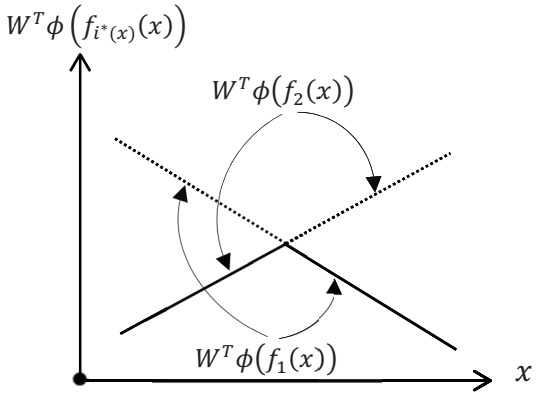

Fig. 1. Symbolic representation of the continuity of $W^T\phi\left(f_{i^*(x)}(x)\right)$ at the discontinuous points of $i^*(x)$. Solid plots represent function $W^T\phi\left(f_{i^*(x)}(x)\right)$ versus x.

B. Implementation and Control

If perfect model for the dynamics of the subsystems are available and used in Algorithm 2, the trained neurocontroller can be used for online optimal control/scheduling of the system. The control/scheduling is done in real-time by feeding the current state x_k at each time step $k \in K$, to equation (9) to calculate i_k^* and applying it on the system. Note that because I has a finite number of elements, the minimization given in (9) simplifies to comparing the values of a scalar-valued function for different $i \in I$ to determine the optimal i.

In this method, no restriction is enforced on the order of the active subsystems and the number of switching. Moreover, a great potential of this method is giving the optimal switching for different initial conditions $x_0 \in \Omega$ as long as the resulting state trajectory lies in the domain on which the network is trained, i.e., $x_k \in \Omega$, $\forall k$. The reason is the cost-to-go approximation is valid when the state belongs to Ω, therefore, one can always use (9) for finding the optimal mode, as long as $x_k \in \Omega$. Note that, except [11,12], the cited methods in the literature calculate the optimal switching only for a pre-specified initial condition.

IV. SOLUTION PROCESS WITH MODELING UNCERTAINTY

The models for the dynamics of the subsystems are required in both the 'offline training' and the 'online implementation' stages. The dependency of the trained NN on the models can be seen through the existence of $f_i(.)$s in Steps 3 and 4 in Algorithm 2.

Moreover, for the online implementation phase, the models are required for calculating $i^*(x)$ as seen in Eq. (9) and discussed in Section III.B. Hence, if the models are not exact, the NN will not provide a precise approximation of the cost-to-go and also the scheduler given in Eq. (9) will not be able to find the optimal mode.

Let the *actual* dynamics of the subsystems be given by

$$x_{k+1} = f_i(x_k) + d_i(x_k), k \in K, i \in I, \tag{20}$$

where $d_i: \mathbb{R}^n \to \mathbb{R}^n$ is a smooth function representing the unmodeled uncertainty existing in subsystem i. Based on the uniform approximation capability of NNs [17,18], there exists weight matrix $V_i \in \mathbb{R}^{n \times l}$, $\forall i \in I$, and basis functions $\sigma: \mathbb{R}^n \to \mathbb{R}^l$ where l is a positive integer, such that the uncertainties of the subsystems can be approximated using neural network $V_i^T \sigma(x)$, i.e.,

$$d_i(x) \cong V_i^T \sigma(x), \forall i \in I. \tag{21}$$

The networks weights, V_is, can be learned online based on the state measurement, i.e., such that network $V_i^T \sigma(x_k)$ approximates input-output mapping $\left(x_k, x_{k+1} - f_i(x_k)\right)$, $\forall k$. The idea is using the nominal model of the subsystems, given in Eq. (1), for offline training of the cost-to-go approximation network. The offline training is based on Algorithm 2. Afterwards, the controller is implemented on the system and identifiers $V_i^T \sigma(x)$s, whose weights are initially set to zero, are trained online to approximate the uncertainties in the subsystems. As V_is are being learned, the weights of the cost-to-go approximator also need to be updated based on the online information of the system output. In this manner, the offline trained W will be re-optimized to approximate the optimal cost-to-go of the actual dynamics. The process in summarized in Algorithm 3

Algorithm 3 (Exploiting Actions)

Step 1: Measure current state x_k.

Step 2: Calculate $i(x_k) = argmin_{i \in I} W^T \phi \left(f_i(x_k) + V_i^T \sigma(x_k)\right)$.

Step 3: Apply $i(x_k)$ on the system and wait for one time step.

Step 4: Measure system state x_{k+1}.

Step 5: Update $V_{i(x_k)}$ based on input-target pair $\left(x_k, x_{k+1} - f_{i(x_k)}(x_k)\right)$.

Step 6: Calculate training target J^t using $J^t = Q(x_k) + W^T\phi(x_{k+1})$.

Step 7: Update W using input-target pair (x_k, J^t).

Step 8: Set $k = k + 1$ and go back to Step 2 until W and V_is converge for different is.

As seen in Algorithm 3, the online learning is composed of both training V_is and re-training W. The actions, i.e., the $i(x_k)$ selections, are done in an *exploiting* fashion [19], as seen in Step 2 of Algorithm 3. The reason for calling the actions as *exploiting* is the fact that the available knowledge of cost-to-go and the dynamics of the subsystems are exploited in Step 2 of the algorithm to take the actions that minimizes the cost-to-go. However, since the utilized V_is are not precise yet, even if the weights of the cost-to-go approximator, i.e. W, is optimal, the selected $i(x_k)$s will not be optimal. Note that if a particular subsystem is not selected in Step 2 of Algorithm 3 to be active, it will never get the chance to be identified, based on Algorithm 3. As an example, assume there is a subsystem whose nominal model is such that the x_{k+1} resulting from activating that subsystem does not lead to the minimum cost-to-go for any k, compared to the case of using other subsystems. However, the actual mode of this subsystem could be completely different. Algorithm 3, however, will never give the chance to such a subsystem to be active at any k, and hence, its actual dynamics will never be identified, regardless of how desired or undesired it is. This behavior leads to the need for *exploring* actions [19] as well as exploiting actions in many reinforcement learning schemes.

Exploring actions are those which are taken to explore the options, not necessarily to minimize the cost-to-go. Such actions could be as simple as randomly selecting an $i(x_k)$ to give the chance to every subsystem to be active at some times and to be identified. Algorithm 4 provides the training scheme based on exploring actions.

Algorithm 4 (Exploring Actions)

Step 1: Measure current state x_k.

Step 2: Randomly select subsystem $i(x_k)$ to be active at time k.

Step 3: Apply $i(x_k)$ on the system and wait for one time step.

Step 4: Measure system state x_{k+1}.

Step 5: Update $V_{i(x_k)}$ using input-target pair $\left(x_k, x_{k+1} - f_{i(x_k)}(x_k)\right)$.

Step 6: Set $k = k + 1$ and go back to Step 2 until V_is converge for different is.

As seen in Algorithm 4, the optimal cost-to-go approximator's weight is not updated after exploring actions. The reason is the fact that the resulting x_{k+1} in here is not selected to be minimizing anything, hence, its resulting cost-to-go, that is $J^t = Q(x_k) + W^T\phi(x_{k+1})$, is not suitable to be used as a training target for updating W. For implementation, the designer should set a balance between exploiting and exploring actions, i.e., between utilizing Algorithms 3 and 4. As the models are being identified, the balance could be updated in favor of utilizing more exploiting actions. Ideally, once the models are completely identified, the actions could be limited to exploiting actions.

V. NUMERICAL ANALYSIS

The second order system with three modes selected in [20,22] is simulated as the numerical example. The objective of this problem is controlling the fluid level in a two-tank setup. The fluid flow into the 'upper tank' can be adjusted through a valve which has three positions: fully open, half open, and fully closed. Each tank leaks fluid with a rate proportional to the square root of the height of the fluid in the respective tank. The upper tank leaks into the lower tank, and the lower tank leaks to the outside of the setup. Representing the fluid height in the upper tank with x_1 and the height in the lower tank with x_2, the *nominal* dynamics of the state vector $x = [x_1, x_2]^T$ are given by the following three modes

$$\dot{x} = f_1(x) \equiv \begin{bmatrix} -\sqrt{x_1} \\ \sqrt{x_1} - \sqrt{x_2} \end{bmatrix},$$

$$\dot{x} = f_2(x) \equiv \begin{bmatrix} -\sqrt{x_1} + 0.5 \\ \sqrt{x_1} - \sqrt{x_2} \end{bmatrix},$$

$$\dot{x} = f_3(x) \equiv \begin{bmatrix} -\sqrt{x_1} + 1 \\ \sqrt{x_1} - \sqrt{x_2} \end{bmatrix}.$$

The objective is forcing the fluid level in the lower tank (i.e., x_2) to track the constant value 0.5. For this purpose, cost function $J = 10\int_0^\infty (x_2 - 0.5)dt$ is used. The control is the position of the valve and can assume one of the three discrete values 0, 0.5, and 1. Each of these values leads to one of the modes listed above. The basis functions for this example were selected as polynomials $x_1^j x_2^l$, where non-negative integers j and l are such that $j + l \leq 7$. This selection led to 36 neurons ($m = 36$). The problem was

discretized using sampling time of $0.05\,s$. The domain $\Omega = \{x \in \mathbb{R}^2 : 0 \leq x_i \leq 1, i = 1,2\}$ was used for the training. The method of least squares [11] was conducted over 1000 random points and the evolution of the weights during the iterative training is plotted in Fig. 2. As seen in this figure, the iterations converged in less than 80 iterations.

Once the network was trained, initial condition $\text{IC1} \equiv [0.8, 0.2]^T$, simulated in [20], was used to determine the optimal solution. Assuming the actual dynamics of the system being identical to the nominal model, the initial condition is simulated and the resulting state trajectories and mode sequence are given in Fig. 3 and 4, respectively. Fig. 3 shows that the network trained under Algorithm 2 did an excellent job controlling the fluid level of the lower tank by tracking the desired value. An interesting feature of the method is approximating optimal solutions for different initial conditions, without needing to retrain the network, as long as the resulting state trajectories lie within the domain of interest, Ω. To evaluate the controller in this regard, a new initial condition, namely $\text{IC2} = [0,0.7]^{\text{T}}$, was simulated using the same trained network. Considering the dynamics of the three modes, it can be observed that as long as the initial condition belongs to Ω, regardless of what switching is applied, the states will always stay in Ω. Therefore, the trained network should produce an optimal solution for any initial condition in Ω. The simulation results for IC2 are given in Fig. 5. This figure demonstrates the capability of the method to produce an approximate optimal solution for different initial conditions.

Note that so far the capability of the method using perfect model for the training is demonstrated only, i.e., assuming the actual dynamics and the nominal model are identical. Now, assume that the *actual* dynamics of the modes are different as given below

$$\dot{x} = f_1(x) + d_1(x) \equiv \begin{bmatrix} -1.2\sqrt{x_1} \\ 1.2\sqrt{x_1} - 0.8\sqrt{x_2} \end{bmatrix},$$

$$\dot{x} = f_2(x) + d_2(x) \equiv \begin{bmatrix} -1.2\sqrt{x_1} + 0.5 \\ 1.2\sqrt{x_1} - 0.8\sqrt{x_2} \end{bmatrix},$$

$$\dot{x} = f_3(x) + d_3(x) \equiv \begin{bmatrix} -1.2\sqrt{x_1} + 1 \\ 1.2\sqrt{x_1} - 0.8\sqrt{x_2} \end{bmatrix}.$$

Utilizing the network trained based on the nominal model for controlling the actual model, without utilizing any online retraining as discussed in Algorithm 3 and 4, the resulting state trajectories are given in Fig. 6. As expected, the controller has not been able to control the actual system. This can be seen through the steady state error in x_2 which was supposed to track 0.5. The performance degradation shows the need for online training based on Algorithm 3 and 4. For online training, the selection between Algorithm 3 (exploiting actions) and Algorithm 4 (exploring actions) is carried out randomly at each time step for the first 15 seconds and afterwards, only Algorithm 3 (exploiting actions) is implemented. The basis functions $\sigma(.)$ used for online model identification is selected as polynomials $x_1^j x_2^l$, where non-negative integers j and l are such that $j + l \leq 3$.

Figs. 7 and 8 show the resulting state trajectories and the mode sequence with online training. The random selection of the active mode in the exploring actions in the first 15 seconds can be seen in Fig. 8. Considering Fig. 7, it is seen that once the actions are switched to purely exploiting actions at time $\mathrm{t} = 15$ s. the controller has nicely forced x_2 to converge to the desired value. Figs. 9 and 10 depict the history of some of the weight elements of the cost-to-go approximator and the identifiers. The figures show the convergence of the weights during the exploring-exploiting phase, i.e., in the first 15 seconds. Note that the weights kept updating under this phase also, but, the rate of change of the weights is much smaller compared to the rates in the first 15 seconds.

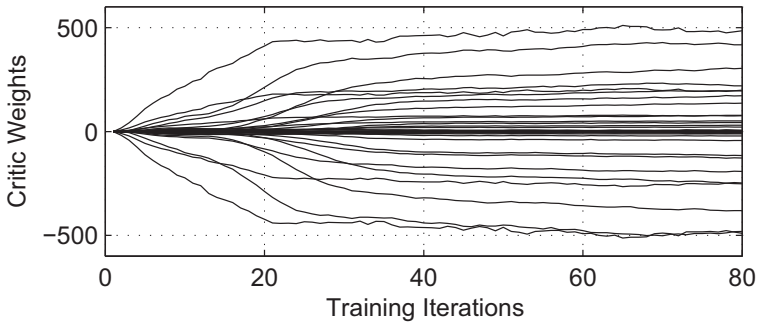

Fig. 2. Offline training weights.

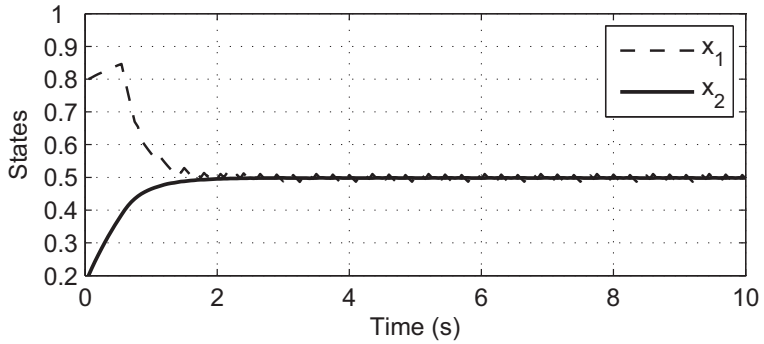

Fig. 3. State histories resulting from simulation with perfect models and IC1.

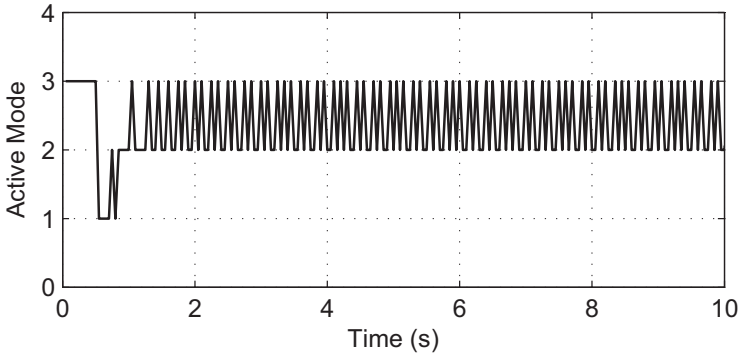

Fig. 4. Mode sequence resulting from simulation with perfect models and IC1.

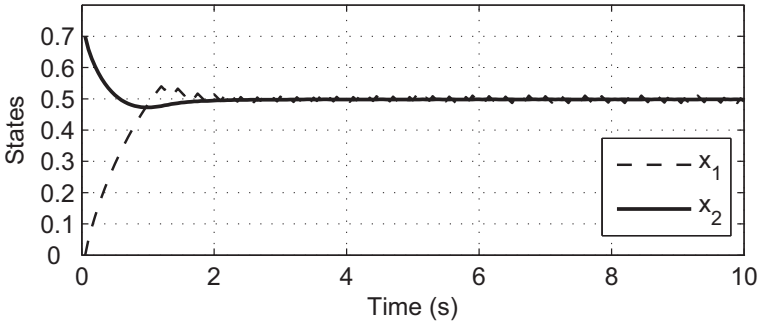

Fig. 5. State histories resulting from simulation with perfect models and IC2.

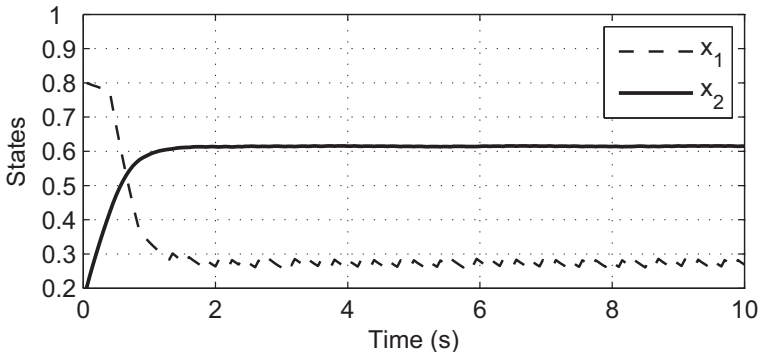

Fig. 6. State histories resulting from simulation with uncertainty and without online training.

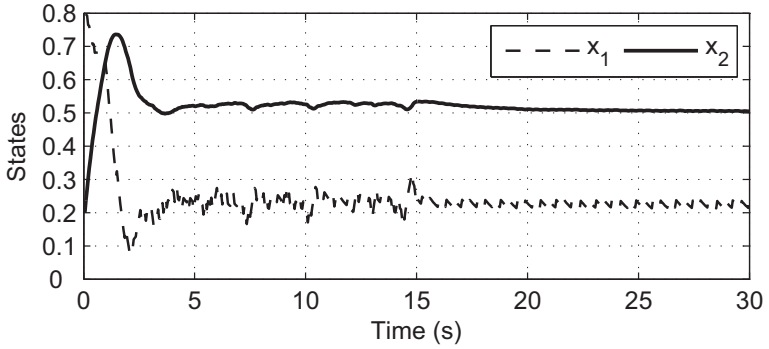

Fig. 7. State histories resulting from simulation with uncertainty and with online training.

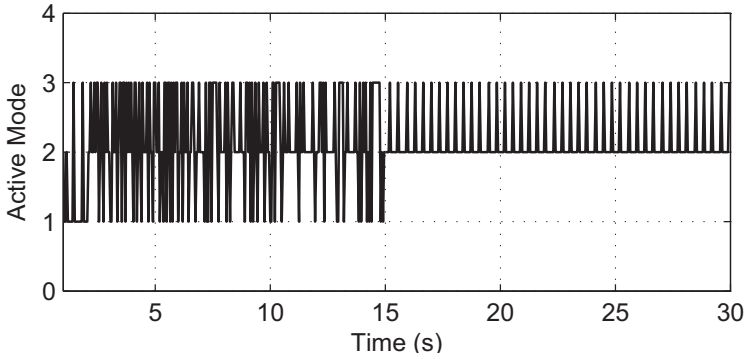

Fig. 8. Mode sequence resulting from simulation with uncertainty and with online training.

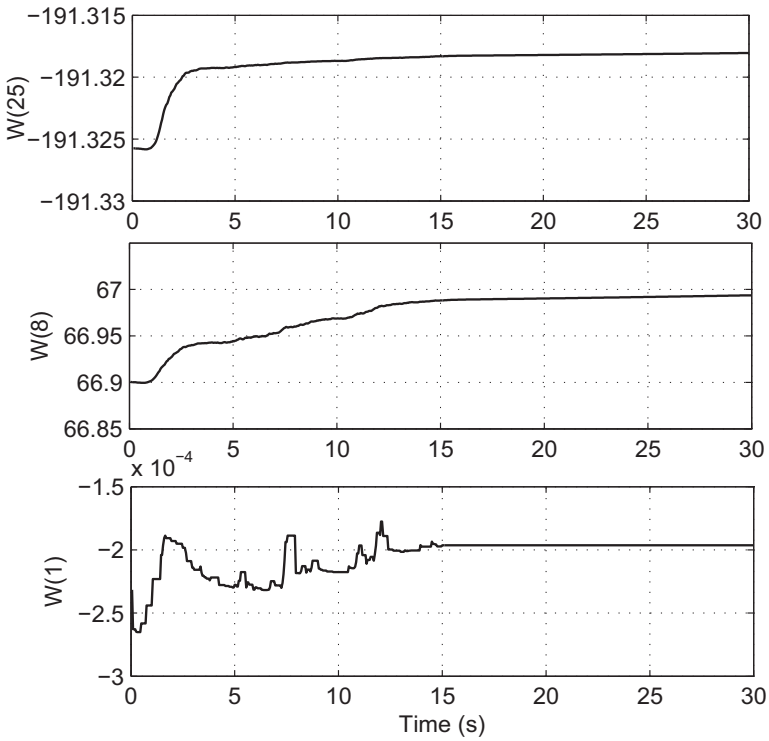

Fig. 9. Evolution of weights of the optimal cost-to-go approximator during online training.

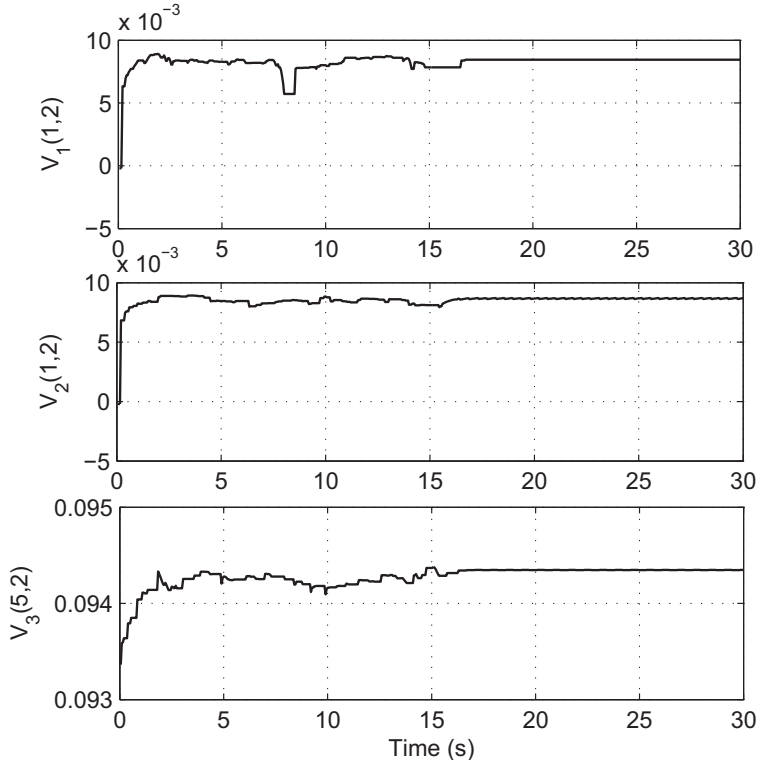

Fig. 10. Evolution of some of the weights of identifiers during online training.

VI. CONCLUSIONS

A method was proposed for optimal switching between subsystems with modeling uncertainty based on offline training of a cost-to-go approximator and online re-optimization of the weights. As many identifiers as the number of subsystems are required to be trained online, in order to capture the dynamic of the subsystems. The balance between exploring actions which only leads to model identification and exploiting actions which leads to both model identification and cost-to-go re-optimization was discussed. Finally, the performance of the method was investigated through a numerical example.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions," *International Journal of Control*, vol. 75 (16), pp. 1406-1426, 2002.

[2] Xu, X., and Antsaklis, P.J., "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Trans. on Automatic Control*, vol. 49 (1), pp.2- 16, 2004.

[3] Egerstedt, M., Wardi, Y., and Axelsson, H., "Transition-time optimization for switched-mode dynamical systems," *IEEE Trans. on Automatic Control*, vol. 51 (1), pp.110-115, 2006.

[4] Axelsson, H., Boccadoro, M., Egerstedt, M., Valigi, P., and Wardi, Y., "Optimal mode-switching for hybrid systems with varying initial states," *Nonlinear Analysis: Hybrid Systems*, vol. 2 (3), pp.765–772, 2008.

[5] Ding, X., Schild, A., Egerstedt M., and Lunze J., "Real-time optimal feedback control of switched autonomous systems," Proc. *IFAC Conference on Analysis and Design of Hybrid Systems*, pp.108-113, 2009.

[6] Kamgarpoura, M., and Tomlin, C., "On optimal control of non-autonomous switched systems with a fixed mode sequence," *Automatica*, vol. 48, pp.1177–1181, 2012.

[7] Zhao, R., and Li, S., "Switched system optimal control based on parameterizations of the control vectors and switching instant," Proc. *Chinese Control and Decision Conference*, pp. 3290-3294, 2011.

[8] Rungger, M., and Stursberg, O., "A numerical method for hybrid optimal control based on dynamic programming," *Nonlinear Analysis: Hybrid Systems*, vol. 5 (2), pp.254–274, 2011.

[9] Sakly, M., Sakly, A., Majdoub, N., and Benrejeb, M., "Optimization of switching instants for optimal control of linear switched systems based on genetic algorithms," Proc. *IFAC Int. Conf. Intelligent Control Systems and Signal Processing*; Istanbul, 2009.

[10] Long, R., Fu, J., Zhang, L., "Optimal control of switched system based on neural network optimization," Proc. *Int. Conference on Intelligent Computing*, pp.799-806, 2008.

[11] Heydari, A., and Balakrishnan, S. N., "Optimal multi-therapeutic HIV treatment using a global optimal switching scheme," *Applied Mathematics and Computation*, Vol. 219, pp. 7872-7881, 2013.

[12] Heydari, A., and Balakrishnan, S. N., "Optimal switching between autonomous subsystems," submitted to *Journal of the Franklin Institute*, 2013.

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

[14] Ding, J., and Balakrishnan, S. N., "Approximate dynamic programming solutions with a single network adaptive critic for a class of nonlinear systems," *J Control Theory Appl, vol. 9* (3), pp. 370–380, 2011.

[15] Unnikrishnan N., and Balakrishnan S. N., "Dynamic reoptimization of a missile autopilot controller in presence of unmodeled dynamics," *Proc. AIAA Guidance, Navigation, and Control Conference*, pp.1-15, 2005.

[16] Kirk, D. E., *Optimal Control Theory: An Introduction,* Dover Publications, New York, 2004, pp. 54.

[17] Homik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

[18] Stone, M. and Goldbart, P., *Mathematics for Physics - A Guided Tour for Graduate Students*, Cambridge, England, Cambridge University Press, p. 70, 2009.

[19] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, London, pp. 26-27, 2012, available online at http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html.

[20] Axelsson, H., Egerstedt, M., Wardi, Y., and Vachtsevanos, G., "Algorithm for switching-time optimization in hybrid dynamical systems," Proc. *IEEE International Symposium on Intelligent Control*, Limassol, Cyprus, 2005.

[21] Trench, W. F., *Introduction to Real Analysis*, available online at http://ramanujan.math.trinity.edu/ wtrench/texts/trench_real_analysis.pdf, 2012, p. 309.

[22] Malmborg J., and Eker, J., "Hybrid control of a double tank system", Proc. *IEEE Conference on Control Application*, Hartford, Connecticut, 1997.

SECTION

2. CONCLUSIONS

Several classes of problems with different challenges were investigated in this dissertation. It was seen that solutions to terminal control problems are time dependent, hence, in order to approximate the optimal solution in a feedback form, one needs to feed both the state vector and the time-to-go into the network, as done in Paper 1. Once the network is trained, it provides solution for different initial conditions and different final times as long as the new final time is not larger than the final time for which the network was trained. The convergence of the DHP-based iterative learning algorithm was proved and it was shown that the training error converges using the proposed weight update law.

The time-dependency of the solution in the rest of the fixed-final-time developments in this dissertation was accommodated using neural networks with time-varying weights. This structure leads to a straightforward algorithm which trains the weights in a *backward-in-time* fashion. It was seen that using this scheme, the critic training is as simple as learning a mapping, while, the actor requires an iterative learning scheme to be trained. The convergence of the iterative algorithm for training the actor was proved using contraction mapping theorem. In Paper 2, an idea was developed for incorporating the hard terminal constraint which was motivated by the solution to the respective linear problem. The network inputs were changed to the state vector as well as a Lagrange multiplier resulting from adjoining the terminal constraint to the cost function. It was seen that the Largrange multiplier can be calculated after the training phase, to be fed to the network for the optimal control calculation. The neurocontroller developed in Paper 2 was seen to be able to provide solution for different initial conditions, different final times, and different terminal points/surfaces.

The performance of approximate dynamic programming in finding the global optimal solution to the fixed-final-time control problem was investigated in Paper 3. A sufficient condition for global optimality of the result, regardless of the convexity or non-convexity of the functions representing the dynamics of the system or the state penalizing terms in the cost function, was derived. Moreover, an idea was presented in converting a static function optimization to an optimal control problem and using ADP for

approximating the global minimum of the selected convex or non-convex function. Numerical results showed that the proposed method results in a trajectory which directly goes to the proximity of the global minimum, regardless of the shape of the local level curves. This is a promising feature which differentiates the method from many nonlinear programming based optimization methods.

The idea of training the critic to approximate the cost-to-go as a function of the switching time, to be used in an offline optimization phase for the calculation of the initial conditions was investigated in Paper 4 and 5, for switching systems with autonomous and controlled subsystems, respectively. It was seen that the method leads to global optimal switching times for different initial conditions.

The approximation of the cost-to-go utilized in Paper 4 and 5 was utilized in Paper 6 and 7 with a new perspective. In Paper 6 and 7 the optimal cost-to-go at any given current state vector was learned, without feeding the switching time to the networks, and it was shown that having this function, the optimal mode for switching can be calculated in real-time in a feedback form. This leads to a much more robust solution to the switching problems and provides the ability to leave the mode sequence and the number of switching free, unlike in papers 4 and 5 where they were enforced.

Finally, an online training algorithm in a reinforcement learning scheme was proposed for learning the optimal switching solution for systems with modeling uncertainties. The method is based on using the available knowledge on the system for offline training and then using online state measurements for re-optimizaing the network through the so called *exploiting* and *exploring* actions. It was seen that having a mix of these two types of actions, the proposed method can adapt itself based on the actual dynamics of the system and provide the (approximate) optimal solution.

VITA

Ali Heydari received his BS and MS degrees in Aerospace Engineering from Sharif University of Technology, Tehran, in 2005 and 2008, respectively. He has been the recipient of the outstanding MS thesis award from Iranian Aerospace Society, the best student paper runner-up award from AIAA Guidance, Navigation and Control Conference, and the outstanding graduate teaching award from Academy of Mechanical and Aerospace Engineers, Missouri S&T. He is a member of IEEE, AIAA, ASME, SIAM, and Tau Beta Pi. His research interests include optimal control, nonlinear control, approximate dynamic programming, control of multi-agent/large-scale systems, and control of hybrid/switching systems. He received the degree of Doctor of Philosophy in August 2013.