2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

SNAC Based Near-Optimal Controller for Robotic Manipulator with Unknown Dynamics

Samrat Dutta* and Laxmidhar Behera

***Abstract*—A near optimal control technique for robotic manipulator with completely unknown dynamics is described in this work. Obtaining the optimal control law u^* depends on solving Hamilton Jacobi Bellman equation but getting an analytic solution is not possible for unknown models. It is shown that instead of solving HJB equation analytically, the optimal control law can be obtained through learning of a Single Network Adaptive Critic (SNAC). The generic nonlinear model of manipulator dynamics is represented as Takagi-Sugeno-Kang fuzzy combination of local linear models. A stabilizing fixed gain controller is designed for the TSK fuzzy system using an unconventional Lyapunov function that is used to represent the value function. Stable Lyapunov $\mathbf{P}^{(l)}$ matrices are selected using the Genetic Algorithm (GA) Toolbox in Matlab. This approach avoids the learning of initial cost that can be accumulated by an existing controller. The critic is trained to approximate the optimal cost J^* by renewing the policy in iterations. Validation of the proposed technique is done through simulation on a robotic manipulator model. Results show the effectiveness of the presented work.**

***Index Terms*—single network adaptive critic, system identification, fuzzy Lyapunov function, nonlinear systems, genetic algorithm.**

I. Introduction

Importance of robotic manipulators is increasing rapidly in modern life. Humans are replaced by the robots where accuracy, precision or repeatability are the foremost criterion to execute certain tasks. Real time pick and place jobs where sequence of sub tasks are needed to be accomplished within a specified time, robotic manipulators making their presence increasingly. Optimal control of robotic manipulators could be very helpful for such problems.

It is well known that dynamic programming technique provides the most comprehensive formulation to solve nonlinear optimal control problem in state feedback form [1] [2]. In dynamic programing, a multistage decision problem is solved and the goal is to minimize a cost function over time. The control action has to be decided at each time instant based on the present system states, in order to minimize the function in long run. To stay on the optimal trajectory, the control solution has to satisfy Hamilton-Jacobi-Bellman (HJB) equation. However, solving the associated HJB equation is easier for linear dynamic systems. But the HJB equation of a nonlinear dynamical system is a differential equation where the solution is not straight forward and requires knowledge of the system dynamics. Moreover, solving such HJB equation demands huge computation and data storage facility [3]. This problem is resolved by introducing *approximate dynamic programming* (ADP) where authors [4] solve for a near optimal solution instead of exact solution. This formulation is known as Adaptive Critic (AC) network. In general AC formulations deal with two Neural Networks which work as actor and critic and utilize system dynamic information during initial training. An adaptive reinforcement learning-based solution is found in [5] where two neural networks are used to solve infinite-horizon optimal control problem of constrained-input continuous-time nonlinear systems. While this approach deals with two networks, [6] uses single network to get the approximate solution for a class of systems. It is shown that if the partial knowledge about the system dynamics is available for a input affine system, single network adaptive critic can efficiently solve the optimal control problem. However, most of the approaches discussed above are for discrete time systems. A critic based optimal controller for continuous time system is proposed by [7] where advantage updating is introduced by the author to train the neural network. Single network adaptive critic is used by [8] to obtain the optimal solution in continuous time. However, the methodology presented here, leads to an underdetermined system and least square technique is used to get the optimal solution. [9] overcomes the problem where the critic network learns the value function using optimal policy. However, all the techniques cited here, use at least partial knowledge of the system. This motivates us to work for an algorithm that gives a near optimal solution for a optimal control problem even if the dynamic information is unavailable.

SNAC is used to solve an optimal control problem of a robotic manipulator where the input matrix of the system is known [10]. The methodology presented here, approximates initial value function using a TSK fuzzy network. The fuzzy network is then learnt to approximate the optimal cost by updating optimal policy. However, it is experienced that finding stable weight matrices is not very straight forward for highly nonlinear systems like robotic manipulators. There could be number of solutions of weight matrices for which the network training error is minimum but it is not necessary that each solution will lead to a stable controller unless we choose them wisely. This motivates us for the current work where Genetic

Samrat Dutta is doctoral student at Department of Electrical Engineering, Indian Institute of Technology Kanpur, PIN 208016, Uttar Pradesh, India (email: samratd@iitk.ac.in)

Laxmidhar Behera is professor, Department of Electrical Engineering, Indian Institute of Technology Kanpur, PIN 208016, Uttar Pradesh, India (email: lbehera@iitk.ac.in)

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

Algorithm is used to select the stable weight matrices.

Rest of the paper is arranged in the following manner: section II describes the formulation of optimal control problem. Proposed solution methodology is discussed in section III. Findings of the present work is discussed in section IV and the conclusion is given in section V.

II. Problem description

Robotic manipulators are inherently nonlinear and involve multi-body dynamics. Let us consider an *m-link* robotic manipulator model in state space form

$$\begin{aligned}\dot{\mathbf{x}} &= \begin{bmatrix}\mathbf{0} & \mathbf{1}\\ \mathbf{0} & -\mathbf{M}^{-1}(\boldsymbol{\theta})\mathbf{C}(\boldsymbol{\theta})\end{bmatrix}\mathbf{x} + \begin{bmatrix}\mathbf{0}\\ \mathbf{M}^{-1}(\boldsymbol{\theta})\end{bmatrix}\mathbf{u}\\ &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}\end{aligned} \tag{1}$$

here $\mathbf{M}(\theta) \in \Re^{m\times m}$ represents the mass matrix which is real and positive definite for a *m-link* manipulators, $\mathbf{C}(\theta, \dot{\theta}) \in \Re^{m\times m}$ contains centrifugal and Coriolis force related terms, $\mathbf{u} \in \Re^m$ represents the combination of applied joint torque and torque due to gravity, acting on the joints and $\theta \in \Re^m$ represents the joint angles of the arm. On the other hand, $\mathbf{A} \in \Re^{2m\times 2m}$, $\mathbf{B} \in \Re^{2m\times m}$ and $\mathbf{x} = [\boldsymbol{\theta} \quad \dot{\boldsymbol{\theta}}]^T$ defines the system states.

Thus, we formulate the optimal control problem as the following. Given a continuous time nonlinear system represented by (1), design a controller $\mathbf{u} = f(\mathbf{x})$ that stabilizes the system (1) and minimizes the infinite-horizon cost (2) without the knowledge of system dynamics.

$$\begin{aligned}J(\mathbf{x}(t_0), \mathbf{u}(t_0)) &= \int_{t_0}^{\infty} (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})\, \mathrm{d}t\\ &= \int_{t_0}^{\infty} \varphi(\mathbf{x}, \mathbf{u}),\end{aligned} \tag{2}$$

here, $\mathbf{Q} \in \Re^{2m\times 2m}$ is positive semi definite, $\mathbf{R} \in \Re^{m\times m}$ is positive definite matrices.

III. Solution methodology

Starting with a completely unknown system (1) we first learn system dynamics as a TSK fuzzy system in state space form of local linear models from input-output data.

$$\dot{\mathbf{x}} = \mathcal{A}(x)\mathbf{x} + \mathcal{B}(x)u, \tag{3}$$

where $\mathcal{A}$ and $\mathcal{B}$ are fuzzy representation of system and input matrix respectively. Learning of initial value function might be difficult when the system is highly nonlinear, since there exists many solution of weight matrices of Lyapunov function. Thus we are motivated to design an initial stable fixed gain controller which helps us to avoid initial value function learning and to start with stable weight matrices. We design a stable fuzzy controller $\mathbf{u}_0(\mathbf{x}) = \boldsymbol{f_0}(\mathbf{x})$ for system (3) and eventually achieve a near optimal controller for the nonlinear system using single network adaptive critic. The Hamiltonian of the above optimal control problem can be expressed as

$$H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}) = \varphi(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^T(\mathcal{A}\mathbf{x} + \mathcal{B}\mathbf{u}), \tag{4}$$

where $\boldsymbol{\lambda} = \frac{\partial J}{\partial \mathbf{x}}$ is the co-state vector of the system. The control law $\mathbf{u}$ is optimal when it satisfies the necessary condition,

$$\frac{\partial H(\mathbf{x}, \boldsymbol{\lambda}^*, \mathbf{u})}{\partial \mathbf{u}} = \frac{\partial \varphi}{\partial \mathbf{u}} + \boldsymbol{\lambda}^{*T}\frac{\partial}{\partial \mathbf{u}}(\mathcal{A}\mathbf{x} + \mathcal{B}\mathbf{u}) = \mathbf{0}. \tag{5}$$

The following expression for the control law $\mathbf{u}$ can be obtained by solving (5).

$$\mathbf{u}^* = -\frac{1}{2}\mathbf{R}^{-1}\mathcal{B}^T\boldsymbol{\lambda}^*, \tag{6}$$

where co-state vector $\boldsymbol{\lambda}^* = \frac{\partial J^*}{\partial \mathbf{x}}$ is along the optimal trajectory in the state space and J^* is the optimal cost. It's well known that the optimal cost satisfies the Hamilton-Jacobi-Bellman equation:

$$\frac{\partial J^*}{\partial t} + \min_u H(\mathbf{x}, \boldsymbol{\lambda}^*, \mathbf{u}) = 0. \tag{7}$$

Equation (7) completes the solution of optimal control problem of system (1). However, obtaining an analytical solution of the optimal control problem of such a system (1) is a difficult task since the solution depends upon J^*. We take a different approach where the optimal cost is learnt over time instead of searching for an analytical solution.

We propose the design of near optimal controller in three stages as the following:

- *Identification of system dynamics as TSK fuzzy model.*
- *Design of stabilizing controller.*
- *Policy Iteration (PI) based learning of optimal control law.*

A. Identification of system dynamics as TSK fuzzy model

We represent the system in n-dimensional state space model where the states are controlled by m-dimensional control input. Since we are interested in the dynamic model of the system, we consider a Takagi-Sugeno-Kang fuzzy logic based parametric system identification technique [11], where the system is represented with minimum number of rules. Thus, fuzzy rule base of L rules that defines the system dynamics (1), can be stated as the following

l^{th} *rule:*

IF $x_1(t)$ is Γ_1^l AND $\cdots$ AND $x_n(t)$ is Γ_n^l THEN

$$\dot{\mathbf{x}}^{(l)} = \mathbf{A}^{(l)}\mathbf{x} + \mathbf{B}^{(l)}\mathbf{u} \tag{8}$$

where

$$\mathbf{A}^l = \begin{bmatrix} a_{11}^l & a_{12}^l & \dots & a_{1n}^l\\ a_{21}^l & a_{22}^l & \dots & a_{2n}^l\\ \vdots & \vdots & \ddots & \vdots\\ a_{n1}^l & a_{n2}^l & \dots & a_{nn}^l\end{bmatrix}\mathbf{B}^l = \begin{bmatrix} b_{11}^l & b_{12}^l & \dots & b_{1m}^l\\ b_{21}^l & b_{22}^l & \dots & b_{2m}^l\\ \vdots & \vdots & \ddots & \vdots\\ b_{n1}^l & b_{n2}^l & \dots & b_{nm}^l\end{bmatrix}$$

Overall model is be given by

$$\dot{\mathbf{x}} = \frac{\sum_{l=1}^R w^{(l)}\mathbf{A}^l}{\sum_{l=1}^R w^{(l)}}\mathbf{x} + \frac{\sum_{l=1}^R w^{(l)}\mathbf{B}^l}{\sum_{l=1}^R w^{(l)}}\mathbf{u} \tag{9}$$

$$= \mathcal{A}\mathbf{x} + \mathcal{B}\mathbf{u} \tag{10}$$

where $\mathcal{A}$ and $\mathcal{B}$ are the fuzzy aggregated system matrices and Γ_j^l, $j = 1, 2, \cdots, n$, is the j^{th} fuzzy set of the l^{th} rule. The rule membership degree $w^{(l)}$, associated with the l^{th} rule is defined as

$$w^{(l)} = \prod_{j=1}^{n} \mu_j^l(x_j), \tag{11}$$

where $\mu_j^l(x_j)$ is the membership value of the fuzzy set Γ_j^l, $l = 1, 2, \cdots, L$. In this work, we consider Gaussian function as Γ_j^l.

B. Design of stabilizing controller

The controller we design for stabilization of system (10), is conceptually a fixed gain state feedback controller [12]. The use of this controller is to get starting $\mathbf{P}^{(l)}$ matrices of Lyapunov function for the system. Since, our main objective is not designing a fuzzy controller, we choose to design a fixed gain controller. Thus, the stabilizing controller can be given by

$$\mathbf{u}_0 = \mathbf{K}\mathbf{x} \tag{12}$$

Control law (12) gives the closed loop system dynamics as following

$$\dot{\mathbf{x}} = \sum_{l=1}^{L} \sigma_l \left(\mathbf{A}^{(l)} + \mathbf{B}^{(l)}\mathbf{K}\right)\mathbf{x} \tag{13}$$

Here σ_l is the membership grade of l^{th} rule that follows $\sum_{l=1}^{L} \sigma_l = 1$ and is defined by

$$\sigma_l = \frac{w^{(l)}}{\left(\sum_{l=1}^{L} w^{(l)}\right)^2} \tag{14}$$

1) Conditions for stabilization: We select a fuzzy function (15) [13]

$$V(\mathbf{x}(\mathbf{t})) = 2\int_{\mathbf{0}}^{\mathbf{x_0}} \ominus(\mathbf{s}) \cdot \mathbf{ds} \tag{15}$$

Here, $\mathbf{s} \in \Re^n$ and $\ominus(\mathbf{s}) = [\Psi_1(\mathbf{s}), \Psi_2(\mathbf{s}), ..., \Psi_n(\mathbf{s})]^T$. However, $V(\mathbf{x}(\mathbf{t}))$ to be considered as a Lyapunov function, it should be independent of state trajectory, i.e.

$$\frac{\partial \Psi_i(\mathbf{x})}{\partial \mathbf{x_j}} = \frac{\partial \Psi_j(\mathbf{x})}{\partial \mathbf{x_i}} \tag{16}$$

for i, j=1,2,...,n ;

Let us choose a fuzzy function $\ominus(\mathbf{s})$ such that

l^{th} *rule:*

IF $x_1(t)$ is Γ_1^l AND $\cdots$ AND $x_n(t)$ is Γ_n^l THEN

$$\ominus^{(l)}(\mathbf{x}) = \mathbf{P}^{(l)}\mathbf{x} \tag{17}$$

for l=1,2,...,L. Here, $\mathbf{P}^{(l)} \in \Re^{n\times n}$ is symmetric positive definite matrix and fuzzy aggregation of (17) is given as

$$\ominus(\mathbf{x}) = \mathbb{P}(\mathbf{x})\mathbf{x} = \frac{\Sigma_{l=1}^{L} w^{(l)}\mathbf{P}^{(l)}}{\Sigma_{l=1}^{L} w^{(l)}}\mathbf{x} \tag{18}$$

Thus, the following theorem can be stated from [13].

Theorem 1: $V(\mathbf{x})$ is a Lyapunov function candidate if there exists $\mathbf{P}^{(l)} = \hat{\mathbf{P}} + \mathbf{D}^{(l)} > 0$ such that

$$\hat{\mathbf{P}} = \begin{bmatrix} 0 & p_{12} & \dots & p_{1n} \\ p_{12} & 0 & \dots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1n} & p_{2n} & \dots & 0 \end{bmatrix} \tag{19}$$

and

$$\mathbf{D}^{(l)} = \begin{bmatrix} d_{11}^{(l)} & 0 & \dots & 0 \\ 0 & d_{22}^{(l)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_{nn}^{(l)} \end{bmatrix} \tag{20}$$

The off diagonal terms of $\mathbf{P}$ matrices will remain same for all the rules. The diagonal elements construct different $\mathbf{P}$s for each rule. It can be shown that (15) is a generic form of conventional Lyapunov candidate function $V(\mathbf{x}) = \mathbf{x}^T\mathbb{P}\mathbf{x}$ where $\mathbb{P} = \hat{\mathbf{P}} + \hat{\mathbf{D}}$ and $\hat{\mathbf{D}}$ is the fuzzy aggregated form.

Thus, time derivative of Lyapunov function of the closed loop system (13) can be given by

$$\begin{aligned} \dot{V}(\mathbf{x}(t)) &= \mathbf{x}^T\mathbb{P}\dot{\mathbf{x}} + \dot{\mathbf{x}}^T\mathbb{P}\mathbf{x} \\ &= \mathbf{x}^T\left(\sum_{i=1}^{L}\sigma_i\mathbf{P}^{(i)}\left(\sum_{j=1}^{L}\sigma_j\left(\mathbf{A}^{(j)} + \mathbf{B}^{(j)}\mathbf{K}\right)\right)\right. \\ &\quad \left. + \left(\sum_{j=1}^{L}\sigma_j\left(\mathbf{A}^{(j)^T} + \mathbf{K}^T\mathbf{B}^{(j)^T}\right)\right)\sum_{i=1}^{L}\sigma_i\mathbf{P}^{(i)}\right)\mathbf{x} \end{aligned} \tag{21}$$

Therefore, the stability condition can be stated by the following theorem.

Theorem 2: The system described in (3) is asymptotically stable with the controller (12) if there exists $\hat{\mathbf{P}}$, $\mathbf{D}^{(l)}$, and $\mathbf{K}$ such that

$$\mathbf{P}^{(l)} = \hat{\mathbf{P}} + \mathbf{D}^{(l)} > 0 \tag{22}$$

$$\mathbf{P}^{(l)}\mathbf{A}^{(l)} + \mathbf{P}^{(l)}\mathbf{B}^{(l)}\mathbf{K} + \mathbf{A}^{(l)^T}\mathbf{P}^{(l)} + \mathbf{K}^T\mathbf{B}^{(l)^T}\mathbf{P}^{(l)} < 0 \tag{23}$$

$$\begin{aligned} &\mathbf{P}^{(i)}\mathbf{A}^{(j)} + \mathbf{P}^{(i)}\mathbf{B}^{(j)}\mathbf{K} + \mathbf{A}^{(j)^T}\mathbf{P}^{(i)} + \mathbf{K}^T\mathbf{B}^{(j)^T}\mathbf{P}^{(i)} + \\ &\mathbf{P}^{(j)}\mathbf{A}^{(i)} + \mathbf{P}^{(j)}\mathbf{B}^{(i)}\mathbf{K} + \mathbf{A}^{(i)^T}\mathbf{P}^{(j)} + \mathbf{K}^T\mathbf{B}^{(i)^T}\mathbf{P}^{(j)} < 0 \end{aligned} \tag{24}$$

where, $l = 1, 2, ..., L$ $i = 1, 2, ..., L-1$ *and* $j = i+1, ..., L$

This is a constrained search problem that we solve using GA toolbox in Matlab. GA searches stable $\mathbf{P}^{(l)}$ and $\mathbf{K}$ that satisfy above constraints.

2) Design of fitness function: Fitness function assigns a fitness value to each genome and the reproductive ability of a genome is decided by the fitness value. Selection and preservation of good genomes depend on the fitness function design. Algorithm 1 shows the design of fitness function which is used in the Genetic Algorithm to select initial weights.

Algorithm 1 Fitness function for initial weight selection

$\mathbf{P}^{(l)} \leftarrow \hat{\mathbf{P}} + \mathbf{D}^{(l)},\ \mathcal{P}_0 \leftarrow 0$
for each $l \in L$ **do**
  $\Lambda_{min} \leftarrow$ *min eigenvalue of* $\mathbf{P}^{(l)}$
  if $\Lambda_{min} < 0$ **then**
    $\mathcal{P}_0 \leftarrow \mathcal{P}_0 - \Lambda_{min} \times \mathcal{N}_0$ [$\mathcal{N}_0$ is a large positive number]
  end if
end for
if $\mathcal{P}_0 > 0$ **then**
  return $\mathcal{P}_0$
else
  for each $l \in L$, $i \in L_1$ and $j \in L_2$ **do**
    $\mathbf{\Lambda}_{max} \leftarrow$ *max eigen value of* $\mathcal{M}$ [$\mathcal{M}$ represents the left hand side of equation (23) - (24)]
  end for
  $\mathcal{P}_1 \leftarrow max\ of\ \mathbf{\Lambda}_{max}$
  return $\mathcal{P}_1$
end if

C. Policy iteration (PI) based learning of optimal control law

This section presents the learning of near optimal controller. Single network adaptive critic methodology is utilized to approximate the infinite horizon cost. We use the same fuzzy sets for the SNAC, which is generated during identification of system dynamics. Critic network can be used to approximate the value function $J(\mathbf{x})$ or the co-state vector $\boldsymbol{\lambda}(\mathbf{x})$ on the optimal state trajectory. Here, we are motivated to approximate $J(\mathbf{x})$ as $V(\mathbf{x})$ in (17). Thus, $V(\mathbf{x})$ is given by [10]

$$V_c(\mathbf{x}(t), t) = \int_{t_c}^{\infty} \left(\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}_c^T(\mathbf{x})\mathbf{R}\mathbf{u}_c(\mathbf{x}) \right) dt. \tag{25}$$

Here, subscript c indicates the time stamp of time t_c at which the value function is estimated. The complete derivative of $V(\mathbf{x}(t), t)$ along the system state trajectory can be described as

$$\begin{aligned} \dot{V}(\mathbf{x}, t) &= \frac{\partial V(\mathbf{x}, t)}{\partial t} + \left(\frac{\partial V(\mathbf{x}, t)}{\partial \mathbf{x}} \right) \dot{\mathbf{x}}(t) \\ &= \frac{\partial V(\mathbf{x}, t)}{\partial t} + \left(\frac{\partial V(\mathbf{x}, t)}{\partial \mathbf{x}} \right) [\mathcal{A}\mathbf{x} + \mathcal{B}\mathbf{u}], \end{aligned} \tag{26}$$

To stay along the optimal system state trajectory, (26) has to satisfy the Hamilton-Jacobi-Bellman equation (7). This leads (26) to

$$\dot{V}(\mathbf{x}, t) = -\varphi(\mathbf{x}, \mathbf{u}). \tag{27}$$

This implies asymptotic stability while staying on optimal state trajectory.

Since, we are motivated to learn the optimal control law $\mathbf{u}^*$, we exploit a learning methodology [14] where the critic network is trained to approximate the optimal cost J^*. As the training progresses, the SNAC eventually learns to predict the optimal cost. We start with the stabilizing controller (12) that we design in previous section and collect cost-to-go in each T time to learn the optimal value function. The scheme is presented as follows

$$V_i(\mathbf{x}(t_0)) = \int_{t_0}^{t_0+T} (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}_i^T\mathbf{R}\mathbf{u}_i)\, \mathrm{d}t + V_i(\mathbf{x}(t_0 + T)) \tag{28a}$$

$$\mathbf{u}_{i+1}(\mathbf{x}) = -\frac{1}{2}\mathbf{R}^{-1}\mathcal{B}^T \frac{\partial V_i}{\partial \mathbf{x}} \tag{28b}$$

Here, subscript i denotes iteration index. At i^{th} iteration, critic network estimates $V_i(x(t))$ in (28a) and policy is renewed to $\mathbf{u}_{i+1}(\mathbf{x})$ at $(i+1)^{th}$ iteration. It is evident from (28a) that the cost is the result of optimal control policy (28b). Hence, as the learning progresses, it falls on the optimal trajectory and critic network eventually learns to approximate the optimal cost.

1) Update of critic weight matrices: Cost-to-go can be represented as summation of cost accumulated in time t_0 to T and cost-to-go at time T, while staying on optimal trajectory. Thus the incremental cost can be given by the following

$$\Delta V(\mathbf{x}(t_0)) = \int_{t_0}^{t_0+T} (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^{*T}\mathbf{R}\mathbf{u}^*)\, \mathrm{d}t \tag{29}$$

When the network is trained properly, the difference between predicted costs at t_0^{th} and $(t_0 + T)^{th}$ instants should be same as $\Delta V(\mathbf{x}(t_0)$ in (29). Therefore, the Genetic Algorithm finds those weights of critic network for which $\|\Delta V(\mathbf{x}(k)) - (V(\mathbb{P}(k) - V(\mathbb{P}(k + T))\|$ is minimized. This must be noted that the stabilization criterion should be satisfied while minimizing the error norm.

Algorithm 2 Fitness function for PI based training

$\mathbf{P}^{(l)} \leftarrow \hat{\mathbf{P}} + \mathbf{D}^{(l)},\ \mathcal{P}_0 \leftarrow 0,\ \mathcal{P}_1 \leftarrow 0$
for each $l \in L$ **do**
  $\Lambda_{min} \leftarrow$ *min eigenvalue of* $\mathbf{P}^{(l)}$
  if $\Lambda_{min} < 0$ **then**
    $\mathcal{P}_0 \leftarrow \mathcal{P}_0 - \Lambda_{min} \times \mathcal{N}_0$ [$\mathcal{N}_0$ is a large positive number]
  end if
end for
if $\mathcal{P}_0 > 0$ **then**
  return $\mathcal{P}_0$
else
  for each $l \in L$, $i \in L_1$ and $j \in L_2$ **do**
    $\Lambda_{max} \leftarrow$ *max eigen value of* $\mathcal{M}$ [$\mathcal{M}$ represents the left hand side of equation (23) - (24)]
    if $\Lambda_{max} > 0$ **then**
      $\mathcal{P}_1 \leftarrow \mathcal{P}_1 + \Lambda_{max} \times \mathcal{N}_1$[$\mathcal{N}_1 < \mathcal{N}_0$, is a large positive number]
    end if
  end for
  if $\mathcal{P}_1 > 0$ **then**
    return $\mathcal{P}_1$
  end if
  $\mathcal{P}_2 \leftarrow \|\Delta V(\mathbf{x}(k)) - (V(\mathbb{P}(k) - V(\mathbb{P}(k + T))\|$
  return $\mathcal{P}_2$
end if

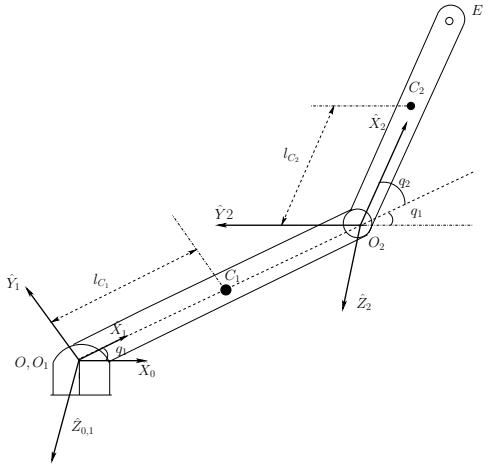

Fig. 1: A manipulator with 2-DOF

2) Fitness function for PI based training: A fitness value is assigned to each genome by the fitness function, based on the output of the function. The fitness function that is used in this phase of learning, by the GA is in fact some augmentation of the previous fitness function. Algorithm 2 shows the design of the fitness function.

IV. Simulation

Let us consider a manipulator of two Degree of Freedom (DOF) given in Fig. 1 [15].

Where the mass of link 1 & 2 are given by m_1 and m_2 respectively and the inertia matrices at the center of mass of both the links are given by

$$I_{C_1} = \begin{bmatrix} I_{xx1} & 0 & 0 \\ 0 & I_{yy1} & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix}, \text{ and } I_{C_2} = \begin{bmatrix} I_{xx2} & 0 & 0 \\ 0 & I_{yy2} & 0 \\ 0 & 0 & I_{zz2} \end{bmatrix}$$

For the given manipulator configuration (1) can be represented as

$$\begin{bmatrix} \mathbf{m_{11}} & \mathbf{m_{12}} \\ \mathbf{m_{21}} & \mathbf{m_{22}} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_\mathbf{1} \\ \ddot{\mathbf{q}}_\mathbf{2} \end{bmatrix} + \begin{bmatrix} \mathbf{c_{11}} & \mathbf{c_{12}} \\ \mathbf{c_{21}} & \mathbf{c_{22}} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_\mathbf{1} \\ \dot{\mathbf{q}}_\mathbf{2} \end{bmatrix} = \begin{bmatrix} \mathbf{u_1} \\ \mathbf{u_2} \end{bmatrix} \tag{30}$$

where m_{ij} and $c_{ij}, i = 1, 2; j = 1, 2$ are the elements of $\mathbf{M}$ and $\mathbf{C}$ respectively. And $\mathbf{q} = [q_1 \quad q_2]^T$ and $\mathbf{u} = [u_1 \quad u_2]^T$

$m_{11} = m_1 l_{C_1}^2 + I_{zz1} + m_2 l_1^2 + m_2 l_{C_2}^2 + I_{zz2} + 2m_2 l_1 l_{C_2}(cosq_2)$
$m_{12} = m_2 l_{C_2}^2 + I_{zz2} + m_2 l_1 l_{C_2}(cosq_2)$
$m_{21} = m_{12}$
$m_{22} = m_2 l_{C_2}^2 + I_{zz2}$
$c_{11} = -2m_2 l_1 l_{C_2} \dot{q}_2 (sinq_2)$
$c_{12} = -m_2 l_1 l_{C_2} \dot{q}_2 (sinq_2)$
$c_{21} = m_2 l_1 l_{C_2} \dot{q}_1 (sinq_2)$
$c_{22} = 0$

The specifications of the manipulator are as the following: $m_1 = 5.768$ kg, $m_2 = 1.802$ kg, $l_1 = 0.6$ m, $l_2 = 0.37$ m $l_{c1} = o.26$ m, $l_{c2} = 0.216$ and $I_{xx1} = I_{xx2} = 0$, $I_{yy1} = I_{yy2} = 0$, $I_{zz1} = 0.135$, $I_{zz2} = 0.0593$.

The dynamic model of the manipulator is presented here to generate input-output data set for system identification. The system model information is not used during the controller design. Training data can be generated using the existing controller. Though random input-output data points can also be used for system identification. It should be noted that the data set should cover all operating regions of the manipulator workspace. The design methodology is described in three steps.

Step 1. The nonlinear manipulator model is identified as a TSK fuzzy model (3) in state space form where each joint position and velocity are the system states. 30000 data points with sampling rate of 5ms are used to learn the system dynamics. We represent the fuzzy model by seven rules. The Gaussian functions are chosen as fuzzy set in the rule base. GA finds proper mean and variance of the fuzzy sets. The procedure is given in [11]. Fig 4 shows the estimated output from TSK fuzzy model for testing data.

Step 2. The proposed methodology avoids learning of the value function with initial controller. The identified TSK fuzzy model is used to select the initial stable weights of the critic network. The procedure involves finding stable Lyapunov $\mathbf{P}^{(l)}$, $l = 1, 2, .., L$ matrices. Algorithm 1 shows the fitness function design which is used by GA to find initial weights. GA searches for suitable $\mathbf{P}^{(l)}$ and $\mathbf{K}$ for which equation (22) to (24) satisfy. Algorithm 1 is coded as the fitness function of the search problem. For this example, we take $\mathcal{N}_0 = 10^{25}$. The controller in (6) is a stabilizing controller where $\boldsymbol{\lambda}^*$ is associated with stable $\mathbf{P}^{(l)}$ matrices and $R = \mathbb{I} \in \Re^2$. Fig. 2 shows the state evolutions when stabilizing controller is used. Results are presented for both the cases of known and unknown input matrix $\mathbf{B}$.

In the figure, plot 'a' is associated with known input matrix $\mathbf{B}$, where as plot 'b' is associated with unknown input matrix. It is evident that almost similar controller performance can be obtained without knowing the system dynamics.

Step 3. Finally, the critic weights are updated by improving the policy to approximate optimal cost-to-go from a given state. In this example we select $T = 0.5$ *sec*, $\mathbf{R} = \mathbb{I} \in \Re^2$ and $\mathbf{Q} = 10 * \mathbb{I} \in \Re^4$ where $\mathbb{I}$ represents identity matrix. The manipulator is operated to collect data points for optimal training of the critic network. The system is left to evolve from some initial joint positions to desired positions under the influence of controller (28b). All the states, control inputs and associated cost are stored with corresponding time stamp at each T time. The critic network is updated when 1000 such data points are collected. GA finds another set of $\mathbf{P}^{(l)}$ matrices that minimize the error norm. Fitness function for this part of training is given in Algorithm 2 where $\mathcal{N}_1 = 10^{15}$. The policy is renewed by the solution provided by GA and the process continues. The results are shown after five such updates.

Fig. 3 shows time evolution of system states and controller output after the critic is iteratively trained with updated policy. Since states are penalized more during the training with optimal control law, the driving input is relaxed to increase in magnitude. All the simulations are done for seven seconds and with same initial points, so that a good comparison can be made and the conclusion can be obtained easily. Fig. 5 depicts

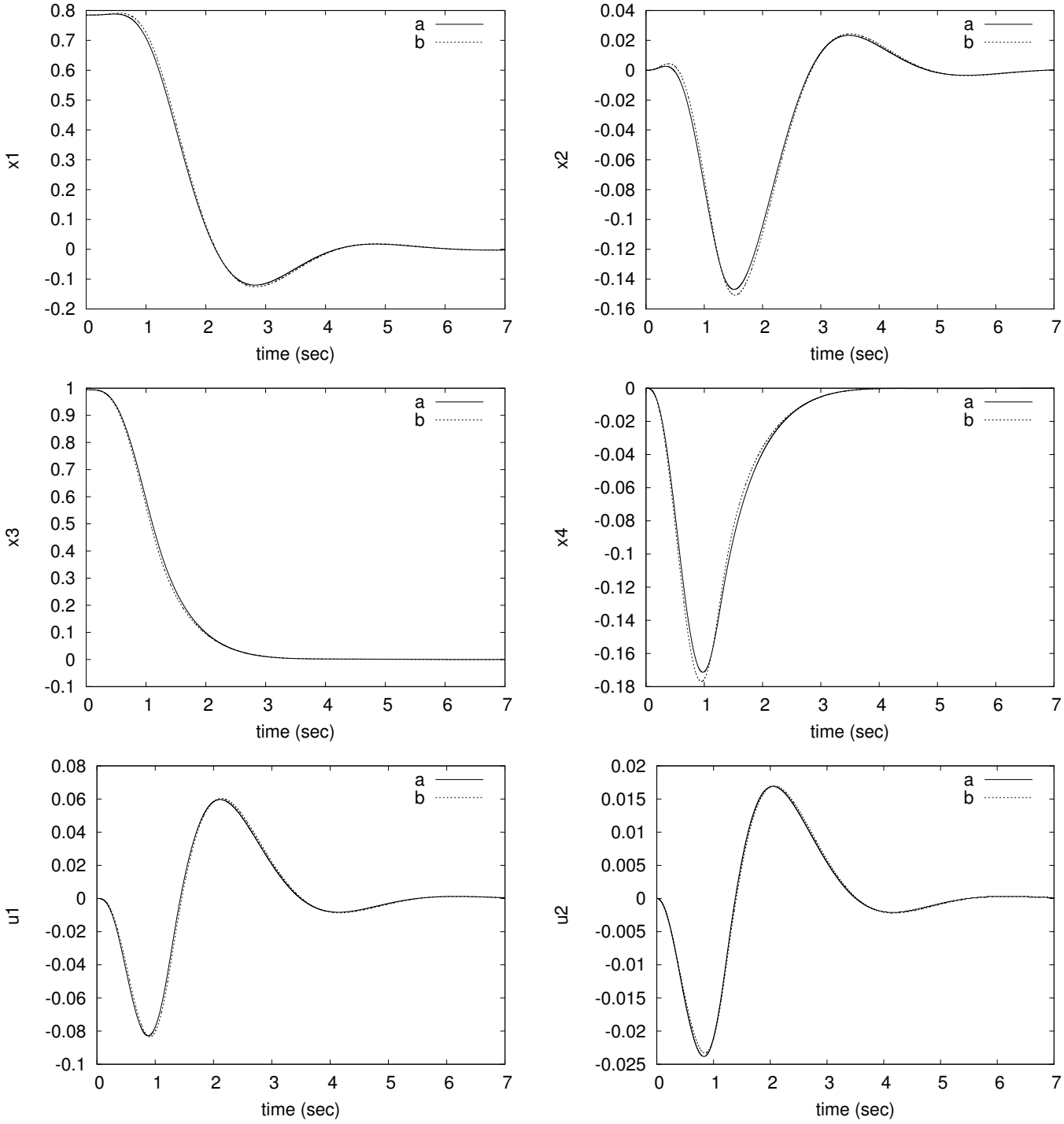

Fig. 2: Validation of control law before policy update

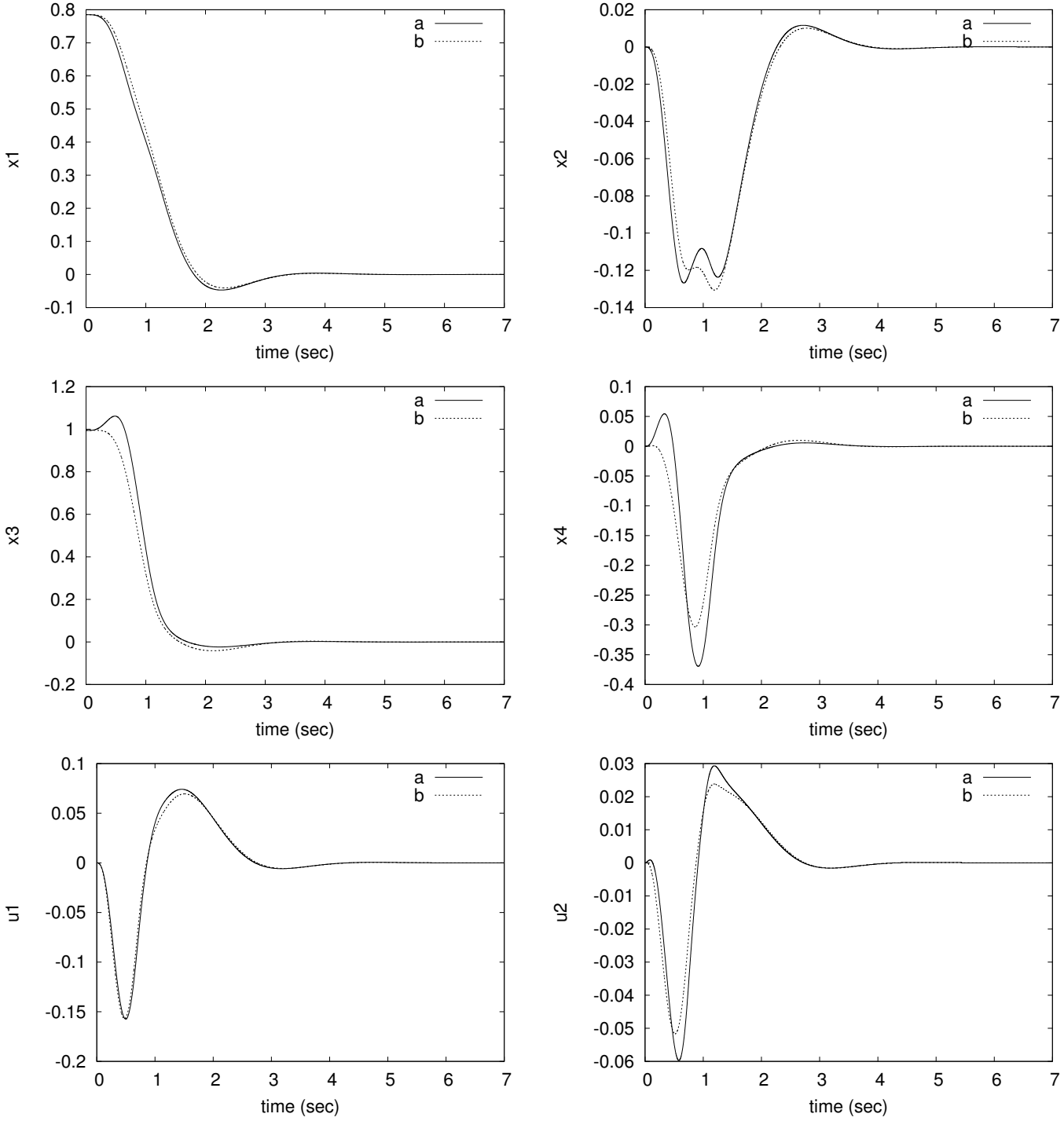

Fig. 3: Validation of control law after policy update

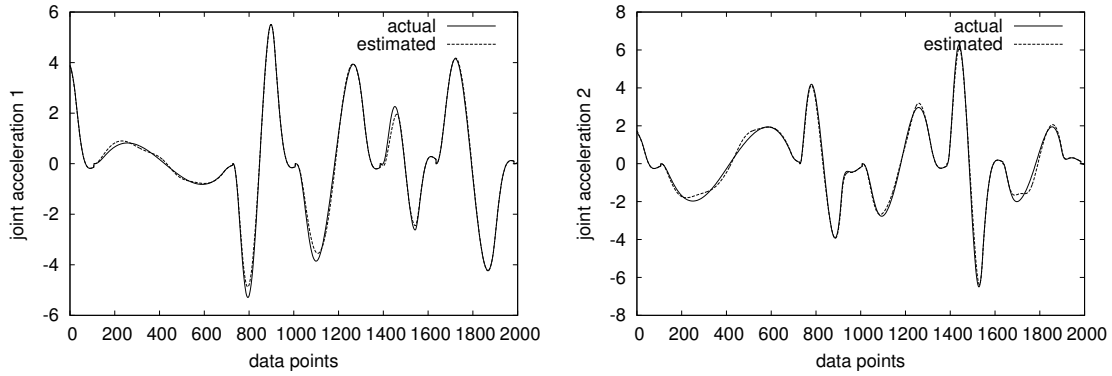

Fig. 4: Validation with test data

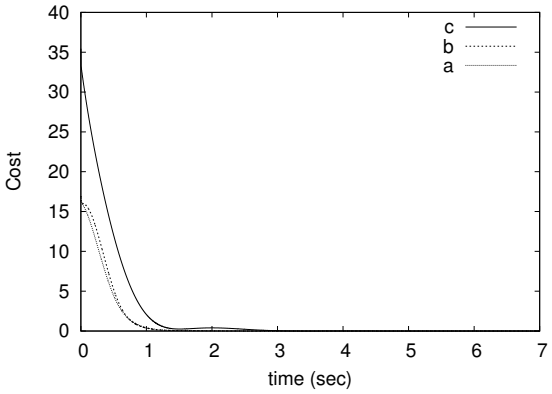

Fig. 5: Comparison of cost

convergence of cost on optimal trajectory. In the figure, plot 'c' denotes the cost that is generated by the initial control law before the critic is trained to estimate optimal cost. Plot 'a' indicates the cost when the original input matrix $\mathbf{B}$ in (1) is used to generate control input in (28b) and plot 'b' shows the cost when the control input is generated using fuzzy estimated input matrix $\mathcal{B}$ from (3). It's evident from the results that optimal performance in both the cases are comparable.

V. Conclusion

The present work describes the methodology of obtaining a near optimal control law for robotic manipulators where the dynamic information of the system is completely unknown. Learning of initial value function is avoided with the introduction of fixed gain controller and use of an unconventional Lyapunov function gives simple stability criteria on weight matrices. The stability criteria are incorporated in GA fitness function to ensure stability. Simulation results show that such updates with renewed policy proceeds towards optimality and the TSK fuzzy model that is learnt from input-output data, can be utilized for near optimal control problems.

References

[1] F. Lewis, "Applied optimal control and estimation." Prentice-Hall, 1992.
[2] D. S. Naidu, *Optimal Control Systems*. CRC Press, 2003.
[3] J. Ding, S. Balakrishnan, and F. Lewis, "A cost function based single network adaptive critic architecture for optimal control synthesis for a class of nonlinear systems," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, July 2010, pp. 1–8.
[4] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent control*, D. A. White and D. A. Sofge, Eds. Multiscience Press, 1992.
[5] X. Yang, D. Liu, and D. Wang, "Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints," *International Journal of Control*, vol. 87, no. 3, pp. 553–566, 2014.
[6] X. W. Radhakant Padhi, Nishant Unnikrishnan and S. Balakrishnan, "A single network adaptic critic (snac) architecture for optimal control," *Neural Networks*, vol. 19, pp. 1648–1660, 2006.
[7] L. C. B. III, "Reinforcement learning in continuous time: advantage updating," *IEEE World Congress on Computational Intelligence.*, pp. 2448–2453 vol.4, 1994.
[8] R. P. Swagat Kumar and L. Behera, "Direct adaptive control using single network adaptive critic," *IEEE International Conference on Systems of Systems Engineering*, 2007.
[9] N. H. S. Prem Kumar Patchaikani, Laxmidhar Behera and G. Prasad, "A t-s fuzzy based adaptive critic continuoust-time input affine nonlinear systems," *IEEE International Conference on System, Man and Cybernetics*, 2009.
[10] S. Dutta and L. Behera, "Policy iteration based near-optimal control scheme for robotic manipulator with model uncertainties," *IEEE Multi-Conference on Systems and Control*, 2013.
[11] A. Patnaik, S. Dutta, and L. Behera, "Data driven system identification using evolutionary algorithms," *Lecture Notes in Computer Science*, vol. 7665, pp. 568–576, 2012.
[12] K. Tanaka and H. O. Wang, *Fuzzy Control Syatems Design and Analysis*. New York: Wiley, 2001.
[13] B. J. Rhee and S. Won, "A new fuzzy lyapunov function approach for a takagi-sugeno fuzzy control system," *Fuzzy Sets and Systems*, vol. 157, pp. 1211–1228, 2006.
[14] P. K. Patchaikani and L. Behera, "Online policy iteration scheme for continuous time nonlinear systems with partially known dynamics," *International conference on Advances in Control and Optimization of Dynamical Systems*, 2012.
[15] J. J. Craig, *Introduction to robotics: Mechanics and Control*. Pearson, 2009.