Discrete-Time Lyapunov based Kinematic Control of Robot Manipulator using Actor-Critic Framework

Ankur Kamboj
Dept. of Electrical Engg.
Indian Institute of Technology Kanpur
India
ORCID: 0000-0001-9010-138X

Ravi Prakash
Dept. of Electrical Engg.
Indian Institute of Technology Kanpur
India
ORCID: 0000-0002-9058-434X

Jayant Kumar Mohanta
Dept. of Electrical Engg.
Indian Institute of Technology Kanpur
India
ORCID: 0000-0002-4738-6399

Laxmidhar Behera
Dept. of Electrical Engg.
Indian Institute of Technology Kanpur
India
ORCID: 0000-0003-1879-5609

***Abstract*—Stability and optimality are the two foremost requirements for robotic systems that are deployed in critical operations and are to work for long hours or under limited energy resources. To address these, in this work we present a novel Lyapunov stability based discrete-time optimal kinematic control of a robot manipulator using actor-critic (AC) framework. The robot is actuated using optimal joint-space velocity control input to track a time-varying end-effector trajectory in its task space. In comparison to the existing near-optimal kinematic control solutions for robot manipulator under AC framework, proposed controller exhibits guaranteed analytical stability. We derive a novel critic weight update law based on Lyapunov stability, thus ensuring that the weights are updated along the negative gradient of Lyapunov function. This eventually ensures closed-loop system stability and convergence to the optimal control in discrete-time. Extensive simulations are performed on a 3D model of 6-DoF Universal Robot (UR) 10 in Gazebo, followed by implementation on real UR 10 robot manipulator to show the efficacy of the proposed scheme.**

***Index Terms*—Robot Manipulator, Discrete-Time Kinematic Control, Actor-Critic, Lyapunov Stability**

I. Introduction

As the robotics industry grows, the robots are expected to deal with a variety of complex tasks. The increasing complexity of tasks for robot manipulators employed in fields such as warehouse automation, production lines, medical applications, defense, underwater surveys, etc. [1] combined with their inherent non-linear dynamics and high Degrees of Freedom (DoF) makes their dynamic control [2] a rather challenging task. An analytical solution to the inverse kinematics of such high-order non-linear systems is not possible [3]. Therefore learning-based strategies such as neural and fuzzy neural network play an increasingly significant role in kinematic control of manipulators as compared to traditional model-based strategies [4], [5]. For most of the works in the literature, stability is a major concern in designing these learning-based controllers for robot manipulator systems. But for the aforementioned applications where a robot has to work round the clock, the controller needs to be optimal in terms of input actuation cost so as to save resources. Thus, learning-based controllers for robot manipulator systems should take into account minimization of a global cost function while learning the inverse kinematics. Optimization can be performed by several techniques as has been discussed in the literature.

Kinematic control using numerical techniques for robot manipulator are based on numerical formulations like the Newton-Raphson iteration [6], predictor-corrector integration [7], gradient based [8], and Euler and trapezoid method [9]. These solutions have stability issues and are inefficient, thus aren't very reliable; though an exact cause cannot be implied from failure of such algorithms (whether the solution failed or there are other undesirable numerical phenomena such as 'large step' divergence or 'curvature reversals' [10]). An interesting kinematic control scheme was suggested in [11] to avoid such problems by an extension of sampling-based RRT algorithm.

Instantaneous optimal solutions have been previously achieved by local optimization using convex optimization based kinematic control. These methods rely on existing efficient convex optimization packages and moreover, are versatile (they can, for example, trade-off different parameters such as actuator input or error). [12] presents an analysis on convex optimization framework being used to achieved kinematic control of robot manipulator using closed-loop inverse kinematics algorithm. In this, based on the pseudo-inverse of Jacobian and null-space, authors achieved the solution using gradient projection based methods. Kinematic control is formulated using nonlinear optimization framework base on Quadratic Programming in [13] and using Neural Network (NN) based optimization framework in [14].

However, optimization based on optimal control theory for non-linear robot systems requires solution of Discrete-Time HJB (DTHJB) equation, and is, as opposed to previous methods, a necessary and sufficient condition for global optimality in non-linear systems. But this is a notoriously hard non-linear partial difference equation to solve analytically [15]. While

employing dynamic programming to solve this, the infamous *'curse of dimensionality'*, and infeasibility of an online solution are the major limitations to this approach. This gave rise to Adaptive Dynamic Programming (ADP) [16]. ADP uses approximations of value function and develops approximation adjustment laws, to obtain the optimal value function as well as an optimal control to solve the DTHJB equation optimally. Model-based ADP solutions employ different variations of AC algorithms [17].
Now, one of the major challenges in ADP is guaranteeing closed-loop stability of the robot system [18]. Lyapunov stability based methods have been very popular within the control community owing to the strong stability guarantees it conclusively draws [19], [20]. The existing works in discrete-time literature such as [21], [22] have achieved kinematic control using Single Network Adaptive Critic, but without any analytical stability proof. Thus, the solution, though is optimal, but a lack of conclusive stability could have serious impacts if robot is employed in critical operations. Unlike previous works, we derive a novel critic weight update law based on Lyapunov stability. This is advantageous as weight update takes place along the negative gradient of the first difference of Lyapunov function, thus eliminating the dependence of robot system stability on initialized values of network weights. To the best of authors' knowledge, there exists no work in robotics literature which explicitly takes system stability into account when deriving update laws for discrete-time AC based optimal kinematic control of robot. The results have been further discussed to prove the efficacy of the proposed scheme. Rest of the paper is organized as follows. Formulation of robot forward kinematics and actor-critic framework is presented in the next section. Section III contains the derivation of novel critic weight update law. The simulation and real-time experimental results are discussed in Section IV, followed by conclusion in Section V.

II. Problem Formulation

This section undertakes the discrete time formulation of forward kinematics giving us the state and the output equations of robot system, and then succinctly formulating the actor-critic based optimal kinematic control framework.

A. Discrete Time Kinematic Model

The forward kinematics of manipulator is a non-linear mapping from the joint-space to the Cartesian-space [23],
$x(t) = \mathbf{f}(q(t))$, where $x(t) \in \mathbb{R}^n$ is the position and orientation vector of end effector in task-space, $\mathbf{f}(.)$ is the non-linear mapping, and $q(t) \in \mathbb{R}^m$ is the vector of joint angles at time t. On the contrary, mapping of velocity from joint to task-space is an affine one,
$\dot{x}(t) = \mathbf{J}(q(t))\dot{q}(t)$, where $\mathbf{J}(q(t)) = \partial \mathbf{f}(q(t))/\partial q(t)$ is the *manipulator Jacobian* matrix $\in \mathbb{R}^{n\times m}$ associated with $\mathbf{f}(q(t))$. The angular speed of robot joints is taken as the control input for the tracking control in this paper. Thus we get our state equation as, $\dot{q}(t) = u(t)$.
Also, we re-write the robot manipulator forward kinematics as the output equation, $\dot{x}(t) = \mathbf{J}(q(t))u(t)$ ($\dot{x}(t)$ being the output vector). Using Euler approximation to discretize, state and output equations are as under,

$$\begin{aligned} q(k+1) &= q(k) + Tu(k) \\ x(k+1) &= \mathbf{f}(q(k)) + g(k)u(k) = x(k) + T\mathbf{J}(k)u(k) \end{aligned} \tag{1}$$

where, we write $\mathbf{J}(q(k))$ as $\mathbf{J}(k)$, and T= Sampling/ Discretization time step. Also, $x(k) \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^m$.
The desired output is,

$$x_d(k+1) = \mathbf{f}_d(q(k)) + g_d(k)u_d(k) = x_d(k) + G(k) \tag{2}$$

which are written in terms of $x_d(k) \in \mathbb{R}^n$, which denotes the desired end effector position and orientation in task space, $u_d \in \mathbb{R}^m$ denotes the desired joint angular velocity input, and $G(k) = g_d(k)u_d(k) \in \mathbb{R}^n$.

B. Optimal Control Formulation

The kinematic control problem is formulated as tracking problem for optimal control formulation. Using (1) and (2) we write the output error and its dynamics for our system as,

$$e(k) = x_d(k) - x(k) \tag{3}$$

$$e(k+1) = e(k) - g(k)u_e(k) = e(k) + G(k) - T\mathbf{J}(k)u(k) \tag{4}$$

where, $u_e(k) = u_d(k) - u(k)$. To solve for the optimal kinematic control of the robot manipulator, it is required to find the control policy $u_e(k)$ that minimizes the infinite horizon Discrete-Time HJB (DTHJB) cost function,

$$\mathfrak{J}(q(k), u_e(k)) = \sum_{i=k}^{\infty} \psi(q(i), u_e(i)) \tag{5}$$

where, ψ is the utility function, or one step cost ($\psi(0,0) = 0$, and $\psi(q(i), u_e(i)) \geq 0 \ \forall \ q(i), \ u_e(i)$). The utility function $\psi(q(i), u_e(i)) = 1/2\left(e(q(i))^T Q e(q(i)) + u_e(i)^T R u_e(i)\right)$, where, $Q > 0 \in \mathbb{R}^{n\times n}$ and $R > 0 \in \mathbb{R}^{m\times m}$ are constant symmetric weighting matrices. The positive definite DTHJB cost-function can also be written as,

$$\begin{aligned} &\mathfrak{J}(q(k), u_e(k)) = \mathfrak{J}(k) \\ &= \psi(q(k), u_e(k)) + \mathfrak{J}(q(k+1), u_e(k+1)) \\ &= \frac{1}{2}\left(e(k)^T Q e(k) + u_e(k)^T R u_e(k)\right) + \left(\sum_{i=k+1}^{\infty} \psi(e(i), u_e(i))\right) \end{aligned} \tag{6}$$

Here, we write $e(q(k))$ as $e(k)$ for ease of representation.
Co-State Equation: The co-state equation is defined as,

$$\frac{\partial \mathfrak{J}(k)}{\partial q(k)} = \lambda(k) \Rightarrow \lambda(k) = -\mathbf{J}^T(k) Q e(k) + I_m^T \lambda(k+1) \tag{7}$$

where, $\lambda(k) = \partial \mathfrak{J}(k)/\partial q(k)$ is the co-state vector in $\mathbb{R}^m$.
Optimal Control Equation: The optimal control policy is given by stationary condition, $\partial \mathfrak{J}(k)/\partial u_e(k) = 0$, so that

$$u_e^*(k) = arg\min_{u_e(k)}\left(\mathfrak{J}(k)\right) = R^{-1}T\lambda^*(k+1) \tag{8}$$

$$\text{or, } u^*(k) = u_d(k) - R^{-1}T\lambda^*(k+1) \tag{9}$$

The optimal tracking control input consists of a feedback term that is a function of co-state (7) and a feed-forward term, $u_d(k)$ ($u_d(k) = g_d(k)^{-1}\Delta x_d(k)$, where $\Delta x_d(k) = x_d(k+1) - x_d(k)$). Next, the AC framework will be discussed.

C. Optimal Solution using Actor-Critic Framework

In the actor-critic structure employed in this paper, the control policies are approximated by an actor NN $\hat{u}(\hat{W}_a(k), e(k))$ and the critic NN approximates the co-state $\lambda(k)$ given by $\hat{\lambda}(\hat{W}_c(k), e(k))$ such that,

$$\hat{\lambda}(k) = \hat{W}_c{}^o(k)^T \sigma\left(Z_c(k)\right) = \hat{W}_c(k)^T \sigma(k) \quad (10)$$

$$\hat{u}(k) = \hat{W}_a{}^o(k)^T \rho\left(Z_a(k)\right) = \hat{W}_a(k)^T \rho(k) \quad (11)$$

where, $Z_c(k) = \hat{W}_c^i(k)^T e(k)$ and $Z_a(k) = \hat{W}_a^i(k)^T e(k)$, $\hat{W}_c^i(k) \in \mathbb{R}^{n\times h}$ and $\hat{W}_c^o(k) \in \mathbb{R}^{h\times m}$ are the actual critic weights, and $\hat{W}_a^i(k) \in \mathbb{R}^{n\times h}$ and $\hat{W}_a^o(k) \in \mathbb{R}^{h\times m}$ are the actual/approximated actor weights of input and output layer (denoted by the superscripts i and o) respectively (h is the number of hidden layer neurons); $\sigma(.)$ and $\rho(.)$ are activation functions of the hidden layer. The NNs are constructed with non-linear activation functions and single hidden layer.
Writing control policy (9) in terms of ideal NN parameters as

$$u(k) = u_d(k) - R^{-1}T\hat{\lambda}(k+1) = W_a(k)^T\rho(k) + \varepsilon_a \quad (12)$$

Also, (7) can be written in terms of ideal NN parameters as,

$$\lambda(k) = -\mathbf{J}^T(k)Qe(k) + I_m^T\lambda(k+1) = W_c(k)^T\sigma(k) + \varepsilon_c \quad (13)$$

where, $W_c(k)$ and $W_a(k)$ are target/ideal weights, and ε_c and ε_a are bounded NN approximation errors of critic and actor NN respectively. Above equations give the desired optimal control input generated by actor NN, and the desired optimal co-state generated by the critic NN. These desired values are then fed back to the networks for weight updation through a novel Lyapunov based weight update algorithm as presented in the next section.
Before proposing the critic weight update algorithm, following assumptions need to be made:

Assumption 1: The Jacobian matrix is bounded as $\left\|\mathbf{J}(k)\right\| \leq \lambda_J \ \forall\ k > 0$, where λ_J is a known positive constant.

Assumption 2: The upper bounds for ideal weights are taken as $\|W_c\| \leq \phi_{cM}$, $\|W_a\| \leq \phi_{aM}$, whereas the NN approximation errors are upper bounded such that $\|\varepsilon_c\| \leq \varepsilon_{cM}$, and $\|\varepsilon_a\| \leq \varepsilon_{aM}$ hold, where ϕ_{cM}, ϕ_{aM}, ε_{cM}, ε_{aM} are known positive constants. Additionally, the activation function of NNs are bounded above as, $\left\|\sigma(.)\right\| \leq \sigma_M$, and $\left\|\rho(.)\right\| \leq \rho_M$ for known positive constants σ_M, and ρ_M [24].

III. Proposed Critic Weight Update Algorithm

We have our error dynamics as given by (4). In the AC architecture, the optimal control/control input $u(k)$ is computed at every iteration step by the actor NN, as given by (12). So the error dynamics can be written as,

$$e(k+1) = e(k) + G(k) - G_d(k) + D(k)\hat{\lambda}(k+1) \quad (14)$$

where, $D(k) = T^2\mathbf{J}(k)R^{-1}$ and $G_d(k) = T\mathbf{J}(k)u_d(k)$ is written for ease of representation.

From the critic NN equations, we define critic error as

$$e_c(k) = \hat{\lambda}(x(k)) - \lambda(x(k)) = \hat{\lambda}(k) - \lambda(k) \quad (15)$$

We now choose our Lyapunov candidate function as,

$$\mathcal{L}_c(k) = e(k)^T e(k) + 2\Gamma\mathfrak{J}(k) + tr(e_c(k)\Xi e_c^T(k))/2 \quad (16)$$

where, $e(k)$ is the system error as defined above; $\Xi(>0) \in \mathbb{R}$ represents the critic error gain, *i.e.*, the contribution of critic error (e_c) to the energy of the system (given by $\mathcal{L}_c$); $\Gamma > 0$ is a design parameter; $\mathfrak{J}(k)$ is the positive definite DTHJB cost-function. Clearly, $\mathcal{L}_c$ is positive-definite.
Now, the first difference of Lyapunov candidate function is,

$$\Delta\mathcal{L}_c(k) = \mathcal{L}_c(k+1) - \mathcal{L}_c(k) \quad (17)$$

Using (16), $\Delta\mathcal{L}_c(k)$ becomes

$$\Delta\mathcal{L}_c(k) = e(k+1)^T e(k+1) + tr\left(e_c(k+1)\Xi e_c^T(k+1)\right)/2 - e(k)^T e(k) - tr\left(e_c(k)\Xi e_c^T(k)\right)/2 + 2\Gamma\Delta\mathfrak{J}(k) \quad (18)$$

The weight update laws for output critic weight vector ($\hat{W}_c^o(k)$) and input critic weight vector ($\hat{W}_c^i(k)$) are proposed as,

$$vec(\hat{W}_c^o(k+1)) = vec(\hat{W}_c^o(k)) - \Im_o^{-1}\left(\nabla_{\hat{W}_c^o}\hat{\lambda}(k)\right)^T \Xi^{-1} e(k+1)(G(k) - G_d(k))^T \frac{e_c(k)}{\left\|e_c(k)\right\|^2} \quad (19)$$

$$vec(\hat{W}_c^i(k+1)) = vec(\hat{W}_c^i(k)) - \Im_i^{-1}\left(\nabla_{\hat{W}_c^i}\hat{\lambda}(k)\right)^T \Xi^{-1} e(k+1)(G(k) - G_d(k))^T \frac{e_c(k)}{\left\|e_c(k)\right\|^2} \quad (20)$$

where, $\Im_o^{-1}$ and $\Im_i^{-1}$ are as defined in *Appendix*. Applying these update laws, $\Delta\mathcal{L}_c(k)$ becomes (refer (26) in *Appendix*),

$$\Delta\mathcal{L}_c(k) \leq -2\left\|e(k)\right\|^2\left[\Gamma\|Q\| - 2 - 2\|Q\|^2\left\|D(k)\right\|^2\left\|\mathbf{J}(k)\right\|^2\right] + 4\left\|D(k)\right\|^2\left[\phi_{cM}^2\sigma_M^2 + \varepsilon_{cM}^2\right] - 2\Gamma\|R\|\left\|u_e(k)\right\|^2 \quad (21)$$

The first term of (21) is less than zero if the design parameter is selected as, $\Gamma > 2(1 + \|Q\|^2 K_J^2\lambda_J^2)/\|Q\|$.
We finally observe that (21) is less than zero if design parameter is selected as above and the following inequalities hold,

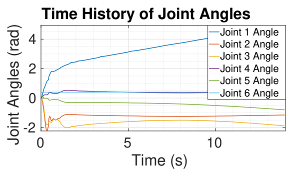
$$\left\|e(k)\right\| \geq \sqrt{\frac{2K_J^2\left[\phi_{cM}^2\sigma_M^2 + \varepsilon_{cM}^2\right]}{\Gamma\|Q\| - 2\left(1 + \|Q\|^2 K_J^2\lambda_J^2\right)}}$$

$$\text{or, } \left\|u_e(k)\right\| \geq \sqrt{\frac{2K_J^2\left[\phi_{cM}^2\sigma_M^2 + \varepsilon_{cM}^2\right]}{\Gamma\|R\|}} \quad (22)$$

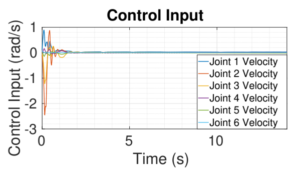
where, $K_J = T^2\lambda_J\left\|R^{-1}\right\|$. Thus the standard Lyapunov extensions conclude $\Delta\mathcal{L}_c(k)$ is less than zero outside of a compact set, implying the output error to be UUB. Further, the computed optimal control input(11) also converges to the neighbourhood of the optimal feedback control (9) with a finite bound given by (22).

IV. Results and Discussion

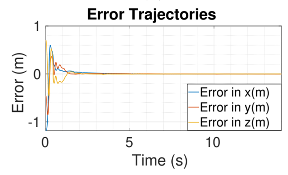
To test the effectiveness of the proposed kinematic control, it is validated by simulations followed by experimental validation on UR10 (6 DoF) robot manipulator.



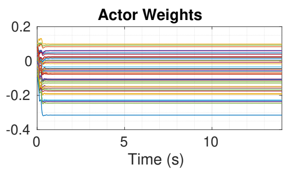
(a)



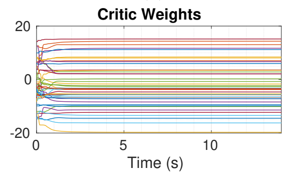
(b)



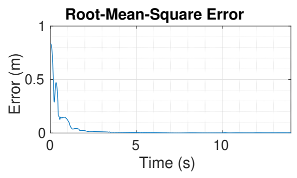
(c)



(d)



(e)



(f)

Fig. 1: Gazebo Simulation Results for Optimal Time Varying Trajectory Tracking

A. Simulation Studies

Firstly, simulations are performed on real 3D model of UR10 in Gazebo under realistic scenarios (such as gravity, joint angle constraints, etc.) wherein the forward kinematics of UR10 is derived using DH parameters.

System Parameters: The discretization time step is taken to be very fine at $2ms$. The reference time varying trajectory initializes at $[0,-0.3,0.75]m$ and is taken to be ellipse centered at $[0,0,1.0075]m$, radius $0.3m$ moving at an angular speed of $0.2\ rad/s$.

Controller Parameters: h (for actor and critic NN)= 6; activation function: sigmoid; actor learning rate: 0.01; $\Xi = 10$; $R = 0.1I_6$; $Q = 10I_3$. The actor and critic NN weights are initialised randomly in a range.
To show the effectiveness of the proposed controller, the initial end-effector position of the manipulator is taken to be as far as possible in its task space, at $[1.18, 0.16, 0.05]m$. This is followed by real-time implementation on a UR10.

B. Real-Time Implementation

Experimental Setup: The experimental setup with a 6 DoF UR10 robot manipulator alongwith its external computer/ host PC and internal computer/controller box shown in Fig 2. It can follow both traditional position commands, as well as joint velocity commands to move the end effector around a specified trajectory. Either a teach pendant or a (C++/Python) TCP communication can used to send and receive commands to the low-level controller. This low-level controller runs on robot's internal computer, controls the arm, interprets and receive the commands, and broadcasts robot joint states. The host PC streams joint commands are streamed by the host PC via URScript at 125Hz over Ethernet to the UR10 real-time interface. To implement our proposed velocity based kinematic control scheme, we configured UrDriver when starting the controller (which is an open source driver class based on python on the host PC running ROS) with parameters such as IP address of the robot, etc.

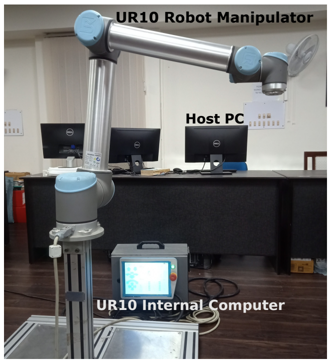
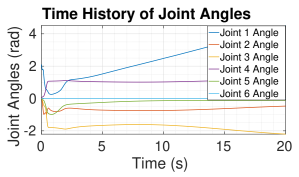
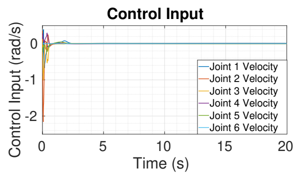


Fig. 2: Hardware Setup

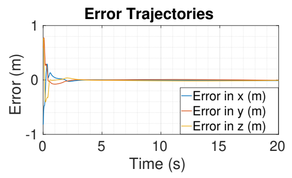
System and Controller Parameters: For experiments, various parameters chosen were: h (for actor and critic NN)= 15; $R = I_6$; $Q = 5I_3$. All the other parameters remaining same, the initial end effector position is chosen to be another farthest point in task space, $[0.8, -0.9, 0.01]m$.



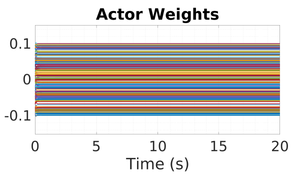
(a)



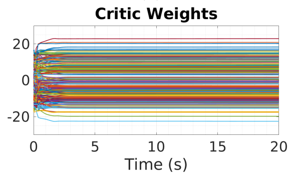
(b)



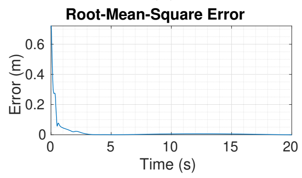
(c)



(d)



(e)



(f)

Fig. 3: Experimental Results for Optimal Time Varying Trajectory Tracking

C. Observations

It is apparent from the result fig. 1(c), 1(f), 3(c), and 3(f) that the error and thus the RMS error converges to zero after a short transient period (less than 5 s). Fig. 1(a) and 3(a) shows the time history of joint angles which initialize from $[0,0,0,0,0,0]rad$ for simulation, and $[2\pi/3,0,0,0,0,0]rad$ for experiments (farthest possible positions). Fig. 1(d), 1(e), 3(d), and 3(e) exhibit actor and critic weights' convergence under the proposed update laws. Also, fig. 1(b) and 3(b) show that despite large initial system error, joint velocities don't exceed system joint velocity limits. This shows the efficacy of our controller in optimally tracking a time-varying trajectory in Gazebo simulation as well as in real-time on UR10.

V. Conclusion

This paper was concerned with the discrete-time kinematic control of robot manipulators using AC framework. The forward kinematics of an m-DoF robot manipulator were presented in a control-affine form, and then AC based optimal control formulation was done. The proof of stability was shown using Lyapunov stability analysis and critic weight update laws were derived based on it. Thus desired optimal cost as well as system stability was proved analytically. The proposed scheme was then verified by simulations on 3D model of UR10 robot manipulator in Gazebo, followed by experimental validation on a real 6-DoF UR10 robot arm.

Appendix: Derivation of Lyapunov Based Critic Weight Update Law

Using (14) in (18) and writing $\hat{\lambda}(k+1)$ from (13), we get

$$\Delta\mathcal{L}_c(k) = \Big[e(k)+G(k)-G_d(k)+D(k)(W_c(k)^T\sigma(k)+\varepsilon_c+ \mathbf{J}^T(k)Qe(k))\Big]^T \Big[e(k)+G(k)-G_d(k)+D(k)(W_c(k)^T\sigma(k)+ \varepsilon_c+\mathbf{J}^T(k)Qe(k))\Big]+2\Gamma\Delta\mathcal{J}(k)+\Big(tr(e_c(k+1)\Xi e_c^T(k+1) -tr(e_c(k)\Xi e_c^T(k))\Big)/2 \tag{23}$$

In discrete time, $e_c(k+1)=e_c(k)+\Delta e_c(k)$, where, Δe_c is incremental change in the error fed to critic NN (depending on change in dependent parameters). Assuming fine discretization, Δe_c is very small. So, we get

$$\Delta\mathcal{L}_c(k) = \Big[e(k)(1+D(k)\mathbf{J}^T(k)Q)+G(k)-G_d(k)+D(k) (W_c(k)^T\sigma(k)+\varepsilon_c)\Big]^T \Big[e(k)(1+D(k)\mathbf{J}^T(k)Q)+D(k)(W_c(k)^T \sigma(k)+\varepsilon_c)+G(k)-G_d(k)\Big]+tr(\Delta e_c(k)\Xi e_c^T(k))+2\Gamma\Delta\mathcal{J}(k)$$

Using the properties of trace, we have,

$$\Delta\mathcal{L}_c(k) = \Big[e(k)(1+D(k)\mathbf{J}^T(k)Q)+D(k)(W_c(k)^T\sigma(k)+\varepsilon_c)\Big]^T \Big[e(k)(1+D(k)\mathbf{J}^T(k)Q)+D(k)(W_c(k)^T\sigma(k)+\varepsilon_c)\Big]+tr\Big(\Delta e_c(k) \Xi e_c^T(k)+2e(k+1)(G(k)-G_d(k))^T\Big)-2\Gamma(\psi(k)) \tag{24}$$

We set the term in the trace equal to zero. Using Triangle Inequality and Completing the Squares (*cts*), we get,

$$\Delta\mathcal{L}_c(k) \le 2\|e(k)\|^2\Big(\big\|1+D(k)\mathbf{J}^T(k)Q\big\|^2\Big)+2\|D(k)\|^2 \Big(\big\|W_c(k)^T\sigma(k)+\varepsilon_c\big\|^2\Big)-2\Gamma(\|Q\|\|e(k)\|^2+\|R\|\|u_e(k)\|^2) \tag{25}$$

Applying *cts*, Cauchy-Schwarz Inequality, and *Assumption 2* gives us,

$$\Delta\mathcal{L}_c(k) \le -2\|e(k)\|^2\Big[\Gamma\|Q\|-2-2\|Q\|^2\|D(k)\|^2\|\mathbf{J}(k)\|^2\Big] +4\|D(k)\|^2\Big[\phi_{cM}^2\sigma_M^2+\varepsilon_{cM}^2\Big]-2\Gamma\|R\|\|u_e(k)\|^2 \tag{26}$$

and upon setting the term in trace equal to zero we get,

$$\Delta e_c(k)\Xi e_c^T(k) = -2e(k+1)(G(k)-G_d(k))^T \tag{27}$$

In (27), we post multiplication by $e_c(k)$ to get,

$$\Delta e_c(k) = -2\Xi^{-1}e(k+1)(G(k)-G_d(k))^T e_c(k)/\|e_c(k)\|^2 \tag{28}$$

Now, to introduce the critic NN parameters, we represent the small change in $e_c(k)$, *i.e.*, $\Delta e_c(k)$ as (assuming small T),

$$\Delta e_c(k) = \frac{\partial e_c(k)}{\partial \hat{W}_c(k)}\Delta\hat{W}_c(k) \tag{29}$$

where, $\Delta\hat{W}_c(k)\to 0$ as $\lim_{k\to\infty}$

A. For Output Weight Vector $\left(\hat{W}_c^o(k)\right)$

We now aim to find out the partial derivative of $e_c(k)$ *wrt* the output weight vector in order to calculate $\Delta e_c(k)$. We introduce a lemma for obtaining the results,

Lemma 1: Let a vector $f=A^TB$, where, $f\in\mathbb{R}^{m\times 1}$, and matrices $A\in\mathbb{R}^{h\times m}$ and $B\in\mathbb{R}^{h\times 1}$ respectively, then

$$f = A^TB = \left(I_m\otimes B^T\right) vec\ (A) \tag{30}$$

Proof: We have $f=A^TB=I_mA^TB$.
Now, invoking the property of vectorization and it's compatibility with Kronecker products [26], we get

$$vec(A^TB) = vec(I_mA^TB) = \left(B^T\otimes I_m\right) vec(A^T) \tag{31}$$

Now, according to [27], $vec(A^T)=\mathbf{K}^{(h,m)}vec(A)$ where, $\mathbf{K}^{(h,m)}$ is the commutation matrix for A.
And, $\mathbf{K}^{(1,1)}\left(B^T\otimes I_m\right)\mathbf{K}^{(h,m)}=\left(I_m\otimes B^T\right)$
Substituting in (31), we prove our lemma. *q.e.d.*

Since $\partial\hat{\lambda}(k)/\partial\hat{W}_c^o(k)$ is a 3rd order tensor, we use vectorization, and from *Lemma 1*, this partial derivative is,

$$\frac{\partial\hat{\lambda}(k)}{\partial vec(\hat{W}_c^o(k))} = \nabla_{\hat{W}_c^o}\hat{\lambda}(k) = \Big[I_m\otimes\sigma^T\left(\hat{W}_c^i(k)^Te(k)\right)\Big] \tag{32}$$

Equation (29) thus becomes,

$$\Delta e_c(k) = \frac{\partial e_c(k)}{\partial vec(\hat{W}_c^o(k))}\Delta vec(\hat{W}_c^o(k)) \tag{33}$$

Now, for output weight vector $\hat{W}_c^o(k)$, we have

$$\frac{\partial e_c(k)}{\partial vec(\hat{W}_c^o(k))} = \frac{\partial e_c(k)}{\partial\hat{\lambda}(k)}\times\frac{\partial\hat{\lambda}(k)}{\partial vec(\hat{W}_c^o(k))} \tag{34}$$

And from (15), above equation becomes,

$$\frac{\partial e_c(k)}{\partial vec(\hat{W}_c^o(k))} = I_n\times\frac{\partial\hat{\lambda}(k)}{\partial vec(\hat{W}_c^o(k))} = \frac{\partial\hat{\lambda}(k)}{\partial vec(\hat{W}_c^o(k))} \tag{35}$$

Therefore, (28) now is,

$$\frac{\partial e_c(k)}{\partial vec(\hat{W}_c^o(k))}\Delta vec(\hat{W}_c^o(k)) = \nabla_{\hat{W}_c^o}\hat{\lambda}(k)\Delta vec(\hat{W}_c^o(k))$$
$$= -2\Xi^{-1}e(k+1)(G(k)-G_d(k))^T\frac{e_c(k)}{\|e_c(k)\|^2} \quad (36)$$

Now, pre-multiplying by $\nabla_{\hat{W}_c^o}\hat{\lambda}^T(k)$, we have

$$\Delta vec(\hat{W}_c^o(k)) = -\Im_o^{-1}\left(\nabla_{\hat{W}_c^o}\hat{\lambda}(k)\right)^T\Xi^{-1}$$
$$e(k+1)(G(k)-G_d(k))^T\frac{e_c(k)}{\|e_c(k)\|^2} \quad (37)$$

where, $\Im_o = 2\left(\nabla_{\hat{W}_c^o}\hat{\lambda}(k)\right)^T \nabla_{\hat{W}_c^o}\hat{\lambda}(k)$.
Therefore, weight update for output critic weight vector takes place as given by,

$$vec(\hat{W}_c^o(k+1)) = vec(\hat{W}_c^o(k)) - \Im_o^{-1}\left(\nabla_{\hat{W}_c^o}\hat{\lambda}(k)\right)^T\Xi^{-1}$$
$$e(k+1)(G(k)-G_d(k))^T\frac{e_c(k)}{\|e_c(k)\|^2} \quad (38)$$

B. For Input Weight Vector $\left(\hat{W}_c^i(k)\right)$

The partial derivative of $e_c(k)$ *wrt* the input weight vector is computed in order to calculate $\Delta e_c(k)$. We have

$$\frac{\partial\hat{\lambda}(k)}{\partial vec(\hat{W}_c^i(k))} = \nabla_{\hat{W}_c^i}\hat{\lambda}(k) = \frac{\partial}{\partial vec(\hat{W}_c^i(k))}(\hat{W}_c^o(k)^T\sigma(Z_c(k)))$$
$$\nabla_{\hat{W}_c^i}\hat{\lambda}(k) = \hat{W}_c^o(k)^T diag\left(\sigma(Z_c(k)).^*\left(11^h-\sigma(Z_c(k))\right)\right)$$
$$\frac{\partial Z_c(k)}{\partial vec(\hat{W}_c^i(k))} \quad (39)$$

where, $().^*()$ is element wise multiplication. Now, using *Lemma 1*, and the property of vectorization, we get

$$\nabla_{\hat{W}_c^i}\hat{\lambda}(k) = \hat{W}_c^o(k)^T\sigma(Z_c(k))\left(11^h-\sigma(Z_c(k))\right)\left[I_h\otimes e(k)^T\right]$$

Now, for input weight vector $\hat{W}_c^i(k)$, (29) becomes,

$$\Delta e_c(k) = \frac{\partial e_c(k)}{\partial vec(\hat{W}_c^i(k))}\Delta vec(\hat{W}_c^i(k)) \quad (40)$$

$$\text{and thus,} \quad \frac{\partial e_c(k)}{\partial vec(\hat{W}_c^i(k))} = \frac{\partial e_c(k)}{\partial\hat{\lambda}(k)}\times\frac{\partial\hat{\lambda}(k)}{\partial vec(\hat{W}_c^i(k))} \quad (41)$$

Plugging (15) into above, (28) now is,

$$\frac{\partial e_c(k)}{\partial vec(\hat{W}_c^i(k))}\Delta vec(\hat{W}_c^i(k)) = \nabla_{\hat{W}_c^i}\hat{\lambda}(k)\Delta vec(\hat{W}_c^i(k))$$
$$= -2\Xi^{-1}e(k+1)(G(k)-G_d(k))^T\frac{e_c(k)}{\|e_c(k)\|^2} \quad (42)$$

Proceeding the same as in Appendix: A., we obtain the weight update for input critic weight vector as,

$$vec(\hat{W}_c^i(k+1)) = vec(\hat{W}_c^i(k)) - \Im_i^{-1}\left(\nabla_{\hat{W}_c^i}\hat{\lambda}(k)\right)^T\Xi^{-1}$$
$$e(k+1)(G(k)-G_d(k))^T\frac{e_c(k)}{\|e_c(k)\|^2} \quad (43)$$

where, $\Im_i = 2\left(\nabla_{\hat{W}_c^i}\hat{\lambda}(k)\right)^T \nabla_{\hat{W}_c^i}\hat{\lambda}(k)$.

References

[1] Y. Zhang, and L. Jin, Robot Manipulator Redundancy Resolution, Wiley, UK, 2017.

[2] A. Kamboj, N.K. Dhar, and N.K. Verma, "Event-triggered control for trajectory tracking by robotic manipulator," In Comp. Intelligence: Theories, App. and Future Dir., vol I, pp. 161-170, Springer, 2019.

[3] P. Agand, and M. A. Shoorehdeli, "Adaptive model learning of neural networks with UUB stability for robot dynamic estimation," 2019 Int. Joint Conf. on Neural Netw. (IJCNN), Budapest, Hungary, pp. 1-6, 2019.

[4] S. Li, Y. Zhang, and L. Jin, "Kinematic control of redundant manipulators using neural networks," IEEE Trans. Neural Netw. and Learning System, vol. 28, no. 10, pp. 2243-2254, 2017.

[5] J. F. Dalmedico, M. Mendonça, L. B. de Souza, R. V. P. Duarte Barros, and I. R. Chrun, "Artificial neural networks applied in the solution of the inverse kinematics problem of a 3D manipulator arm," 2018 Int. Joint Conf. on Neural Netw. (IJCNN), Rio de Janeiro, pp. 1-6, 2018.

[6] J. Angeles, "On the numerical solution of the inverse kinematic problem," The Int. Journ. of Robo. Research, vol. 4, no. 2, pp. 21-37, 1985.

[7] A. Goldenberg, J. Apkarian, and H. Smith, "A new approach to kinematic control of robot manipulators," Journ. of Dynamic System, measurement, and Control, vol. 109, no. 2, pp. 97-103, 1987.

[8] L.C. Wang, and C.C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," IEEE Trans. on Robo. and Autom., vol. 7, no. 4, pp. 489-499, 1991.

[9] E. Sariyildiz, and H. Temeltas, "Performance analysis of numerical integration methods in the trajectory tracking application of redundant robot manipulators," Int. Journ. of Adv. Robo. System, vol. 8, no. 5, pp. 63, 2011.

[10] D. Whitney, "Optimum step size control for newton-raphson solution of nonlinear vector equations," IEEE Trans. on Autom. Control, vol. 14, no. 5, pp. 572-574, 1969.

[11] M.V. Weghe, D. Ferguson, and S.S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in 7th IEEE-RAS Int. Conf. on Humanoid Robots, pp. 477-482, 2007.

[12] A. Colome, and C. Torras, "Closed-loop inverse kinematics for redundant robots: Comparative assessment and two enhancements," IEEE/ASME Trans. on Mech., vol. 20, no. 2, pp. 944-955, 2015.

[13] P. Beeson, and B. Ames, "Trac-ik, "An open-source library for improved solving of generic inverse kinematics," in Humanoid Robots (Humanoids)," 15th IEEE-RAS Int. Conf. IEEE, pp. 928-935, 2015.

[14] S. Li, Z. Shao, and Y. Guan, "A dynamic neural network approach for efficient control of manipulators," IEEE Trans. on System, Man, and Cybern.: System, vol. 49, no. 5, pp. 932-941, 2019.

[15] A. E. Bryson, Applied Optimal Control: Optimization, Estimation and Control, Routledge, 2018.

[16] P.J. Werbos, "Approximate dynamic programming for real time control and neural modeling," In Handbook of Intelligent Control, White and Sofge, Eds. Van Nostrand Reinhold, pp. 493-525, 1994.

[17] S. Ferrari, and R.F. Stengel, "Model-based adaptive critic designs," in J. Si, A. G. Barto, W. B. Powell and D. Wunsch, eds, 'Handbook of Learning and Approximate Dynamic Programming', IEEE Press, New York, pp. 64-94, 2004.

[18] A. Menon, R. Prakash, and L. Behera, "Adaptive critic based optimal kinematic control for a robot manipulator," in 2019 Int. Conf. on Robotics and Autom. (ICRA), pp. 1316-1322, 2019.

[19] J. Umlauft, L. Pöhler, annd S. Hirche, "An uncertainty-based control lyapunov approach for control-affine systems modeled by gaussian process," IEEE Control System Lett., vol. 2, no. 4, pp. 483-488, 2018.

[20] T. Göhrt, P. Osinenko, S. Streif ,"Adaptive dynamic programming using lyapunov function constraints," IEEE Control System Lett., vol. 3, no. 4, pp. 901-906, 2019.

[21] P. K. Patchaikani, L. Behera, and G. Prasad, "A single network adaptive critic-based redundancy resolution scheme for robot manipulators," IEEE Trans. on Industrial Elec., vol. 59, no. 8, pp. 3241-3253, 2012.

[22] J. Jose, R. Prakash, and L. Behera, "Reward based single network adaptive critic for optimal kinematic control of a redundant robot manipulator," In 2018 IEEE Symp. Series on Comp. Intelligence (SSCI), pp. 1555-1562, 2018.

[23] F.L. Lewis, D.M. Dawson, and C.T. Abdallah, Manipulator Control Theory and Practice, 2004.

[24] T. Dierks, and S. Jagannthan, ”Optimal control of affine nonlinear discrete-time systems,” in IEEE 17th Mediterranean Conf. on Control and Autom., pp. 1390-1395, 2009.

[25] V. Sundarapandian, ”An invariance principle for discrete-time nonlinear systems,” App. Mathe. Lett. 16, no. 1, pp. 85-91, 2003.

[26] H.D. Macedo, and J.N. Oliveira, ”Typing linear algebra: A biproduct-oriented approach,” Science of Comp. Programming, 78(11), pp. 2160-2191, 2013.

[27] J.R. Magnus, and H. Neudecker, ”Matrix differential calculus with applications in statistics and econometrics,” Wiley Series in Prob. and Stat., John Wiley & Sons, 1988.