International Journal of Innovative
Computing, Information and Control
Volume **10**, Number **3**, June **2014**

 ISSN 1349-4198
pp. **1071-1094**

QR-TUNING AND APPROXIMATE-LS SOLUTIONS OF THE HJB EQUATION FOR ONLINE DLQR DESIGN VIA STATE AND ACTION-DEPENDENT HEURISTIC DYNAMIC PROGRAMMING

João Viana da Fonseca Neto[1] and Patrícia Helena Moraes Rêgo[2]

[1]Department of Electrical Engineering
Federal University of Maranhão
São Luís-MA 65085-580, Brazil
jviana@dee.ufma.br

[2]Department of Mathematics and Computing
State University of Maranhão
São Luís-MA 65055-310, Brazil
phmrego@yahoo.com.br

Received April 2013; revised November 2013

Abstract. *A novel approach for online design of optimal control systems based on QR-tuning, state and action-dependent heuristic dynamic programming, and approximate-LS solutions of the Hamilton-Jacobi-Bellman (HJB) equation is the main concern of this paper. The QR-tuning for optimal control systems takes into account heuristic variations in the weighting matrices Q and R of the discrete linear quadratic regulator (DLQR) performance index. These heuristics are guided by an approximate relation of the matrices Q and R. Specifically, the proposed approximate solutions are based on the least-squares approach, and applications are performed on the DLQR problem. The parameterizations of the Bellman equation, utility function and dynamic system assemble a framework for the solution of the DLQR problem. The approximate solutions of the HJB-Riccati equation are given by a vectorization of a quadratic form which is the value function of DLQR. Such formulation allows a least-squares (LS) solution of the discrete algebraic Riccati equation (DARE). The LS formulation of the problem is based on a transformation of the state space variables to assemble the regressor vectors, and observations are based on the utility function and DARE solutions. The computational experiments are provided to evaluate the efficiency of QR-tuning heuristics to map the stable Z-plane.*
Keywords: Heuristic dynamic programming, Discrete linear quadratic regulator, *QR*-tuning, Least-squares estimation

1. **Introduction.** This paper is concerned with the development of a QR-tuning method and approximate solutions of the Hamilton-Jacobi-Bellman (HJB) equation for online optimal control design in the context of reinforcement learning. We propose a methodology that employs approximate dynamic programming paradigms to assign eigenvalues in the Z-plane for multiple-input multiple-output (MIMO) systems. The QR-tuning method is a heuristic method that guides the search of weighting matrices Q and R of the utility function according to the duality principle of these two matrices in optimal decision rule.

Specifically, the adaptive actor-critic paradigm is presented in the context of approximate/adaptive dynamic programming (ADP), taking into account the state and action-dependent heuristic dynamic programming methods to develop online algorithms for discrete linear quadratic regulator (DLQR) design. These algorithms are used to evaluate the QR-tuning method on a framework for approximate solutions of the HJB equation that requires no knowledge of the plant model, i.e., the solution of the HJB equation, which appears in the form of the discrete algebraic Riccati equation (DARE) on the DLQR control

system, does not depend on the dynamic system model. In the action-dependent heuristic dynamic programming (AD-HDP) paradigm, the decision rule is fully independent of plant model. On the other hand, in the state heuristic dynamic programming (state HDP) paradigm, the decision rule computation is model-dependent, but the solution of the HJB equation is model-free.

The relevance of the HJB equation is observed by theoretical developments and applications reported in the scientific and technical media. Adaptive dynamic programming and reinforcement learning for control can be seen in [1] in the perspective of a promising research field for intelligent controllers that learn on-the-fly, and show optimal behavior. In [2], the authors present a supervised adaptive dynamic programming algorithm for adaptive cruise control system. An ADP based strategy for real-time energy control of hybrid electric vehicles is presented in [3]. The authors developed a fuel-optimal control that depends on the current system operation and it is not dependent on prior knowledge of future conditions. Applications of ADP and computational intelligence are presented in [4], consisting of a fuzzy logic controller with ADP optimization for traffic signals. The action network and critic network of the ADP approach are trained by the swarm optimization method in [5], and the authors [6] developed a research on a parallel learning adaptive dynamic programming based on genetic algorithms.

An ADP advantage is its ability to deal with the nonlinear characteristics of the dynamic systems. There are many references on this topic reported in recent papers. For example, an intelligent optimal control scheme for unknown nonlinear discrete-time systems is proposed in [7]. The authors [8] present an ADP approach as a learning control method for inverted pendulum control. The problem of finite-horizon optimal tracking control for a discrete-time nonlinear system is solved by an iterative ADP algorithm [9].

A convergence analysis is presented in [9] with respect to cost function and control law. The other author [10] presents a convergence analysis for DLQR control system design. The convergence of the value iteration based HDP algorithm is proven in [11] for the DLQR. The proposed convergence procedure for convergence analysis is based on the traces of matrices Q and R that guide the eigenvalue assignment. This method has been evaluated for DLQR design based on a model [12].

In general, the importance of these investigations is in the scope of theoretic developments and applications of the HJB equation in the form of algebraic Riccati equation (ARE). In reference [13], the authors present a positive stabilizing solution to algebraic Riccati equations with an indefinite quadratic term via a recursive method. An investigation on the existence and representation of stabilizing solutions to a class of generalized ARE that appears in analysis and synthesis problems of continuous-time descriptor systems is presented in [14]. The research on the ARE solution is important for H_∞, LQR and LQG controllers which are designed with Riccati equations, as can be seen in [15, 16]. Further readings on DARE solutions and convergence issues are given on [17, 18, 19].

In the control context, this paper focuses on a methodology to assign eigenstructures based on the traces of weighting matrices Q and R of the DLQR control system, where the weighting matrices play the role of design parameters. In a conservative manner, those matrices are used to penalize the cost of control policy and state variable deviations, but here they are taken as design parameters [12, 20]. Any proposed convergence improvement of the HJB-Riccati equation solution based on a model-free paradigm is challenging, because viable approximate solutions are not guaranteed for all deviations of DLQR weighting matrices. The proposed strategies to improve the approximate solution of the HJB-Riccati equation are based on QR-tuning and least-squares methods.

This article presents a general theory on approximate dynamic programming, the problem formulation and characterization, and applications on the DLQR control and computational experiments. These topics are organized in several sections and an appendix. Initially, the ADP formulations and the approximations based on state value function and Q_L learning are presented in Section 2. In Section 3, the basic concepts on Hamilton-Jacobi-Bellman equation are presented for the discrete linear quadratic regulator problem. The characterization and formulation of the approximate HJB solution via parametric estimation problem are presented in Section 4 for the DLQR design. The approximate solution of the discrete algebraic Riccati equation is formulated in the least-squares sense, where the observation consists of value function, and regressor vectors are built from the state space variables of the dynamic system.

The HDP algorithms of Section 5 are procedures that have their development based on the DLQR parameterizations presented in Section 3. The state and action-dependent heuristic dynamic programming algorithms are developed to evaluate the skills of the proposed method for eigenstructure assignment of multivariable systems. The QR-tuning and approximate solutions of the HJB-Riccati equation for the DLQR design via state and action-dependent heuristic dynamic programming results are presented in the computational experiments of Section 6. The proposed approximation algorithms are illustrated in a third-order dynamic system. The accuracy is compared with admissible Riccati solutions and the convergence rates are compared with the state (state HDP) and action-dependent (AD-HDP) approximations that generate regressor vectors by the state variables Kronecker product and vectorization of the HJB-Riccati solution, taking into account the ADP approaches. The HJB equation solution is performed by strategies that are based on the least-squares (LS) methods. The final remarks are presented in Section 7. Appendix A shows the dynamic system models and the initial conditions of the algorithms.

2. **Approximate Dynamic Programming.** The dynamic behavior of the system is represented by models in state space and the Bellman equation. The system model is given by

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

$$u_k = g(x_k), \tag{2}$$

where x_{k+1} denotes the system state at instant $k+1$ (sampling points), and according to a control policy g, u_k is the control action to be taken at current instant k.

The representation of trade-offs for purposes of evaluating the control objective is performed by the model of long term reward (often called the value function), which is given by

$$V_g(x_k) = \sum_{i=k}^{N} \gamma^{i-k} r(x_i, g(x_i)), \tag{3}$$

where r is a utility function that establishes the reward $r(.)$ of the transition from x_i to x_{i+1}, and $0 < \gamma \leq 1$ is a discount factor. In the form of the Bellman equation, it follows that the value function (3) is given by

$$V_g(x_k) = r\left(x_k, g(x_k)\right) + \gamma V_g\left(f(x_k, g(x_k))\right) \tag{4}$$

$$= r(x_k, g(x_k)) + \gamma V_g(x_{k+1}). \tag{5}$$

The optimal control policy is defined as the mapping g^* that promotes the largest possible set of rewards, which satisfies

$$V_{g^*}(x) \geq V_g(x), \quad \forall\{g, x\}, \tag{6}$$

where V_{g^*} is the optimal value function due to control policy g^*. According to Bellman's Optimality Principle, the optimal value function, denoted by V^*, must satisfy the discrete time Hamilton-Jacobi-Bellman equation or the Bellman optimality equation, i.e.,

$$V^*(x_k) = \max_{g(\cdot)} \left\{ r(x_k, g(x_k)) + \gamma V^*(f(x_k, g(x_k))) \right\}, \tag{7}$$

and the optimal control policy g^* is given by

$$g^*(x_k) = \arg\max_{g(\cdot)} \left\{ r(x_k, g(x_k)) + \gamma V^*(f(x_k, g(x_k))) \right\}. \tag{8}$$

The supervised learning can be introduced in Equation (5) using an iterative scheme which has V_g approximated by parameterized models. In their general form, these models have as starting point the parameterized model given by

$$\Theta(x, r, f, \theta) = r(x, g(x)) + \gamma V(f(x, g(x)), \theta), \tag{9}$$

where θ is the parameter vector of the approximation. This vector should minimize the mean square error between the estimated value $V(x, \theta)$ and the measured value $\Theta(\cdot)$, which is given as

$$\theta_{k+1} = \arg\min_{\theta} E\{|V(x, \theta) - \Theta(x, r, f, \theta_k)|^2\}, \tag{10}$$

where $E(\cdot)$ denotes the expected value.

The parameterization $g(x) = g(x, \kappa)$ of control policy and greedy iteration allow determination of parameters corresponding to the optimal value function. The optimal parameter κ^* is computed according to the Bellman optimality equation and the value function estimate V, which is given as

$$\kappa^* = \arg\max_{\kappa} \left\{ r(x, g(x, \kappa)) + \gamma V(f(x, g(x, \kappa), \theta) \right\}. \tag{11}$$

2.1. **State HDP.** The estimation of the state value function $V(x)$ for a given policy only requires samplings from the instantaneous reward function r, while the models of the environment and the instantaneous reward are needed to determine the value function $V(x)$ corresponding to the optimal policy. The optimal solution $V_{g^*}(x)$ of Equation (6) that satisfies the Bellman Equation (5) is obtained by the optimal parameters of Equation (11). The optimal solution is obtained by solving the equation of the gradient for κ that is given by

$$\frac{\partial V}{\partial \kappa} = 0. \tag{12}$$

Clearly, the derivatives of the parameterized models f, r, V and g are required to determine the gradient $\frac{\partial V}{\partial \kappa}$. Thus, the optimal of Equation (11) should satisfy:

$$\frac{\partial r}{\partial g}\frac{\partial g}{\partial \kappa} + \gamma \frac{\partial V}{\partial f}\frac{\partial f}{\partial g}\frac{\partial g}{\partial \kappa} = 0. \tag{13}$$

2.2. **Action-dependent HDP.** The action-dependent heuristic dynamic programming is based on the Q-learning approach, which will be denoted by Q_L, and consists of a method for estimating the Q_L-function for any optimal or non-optimal policy [10, 21]. The AD-HDP formulation only requires samples from the instantaneous reward function r. The Q_L-function, or action value function, is defined by

$$Q_L(x,u) = r(x,u) + \gamma V_g(f(x,u)). \tag{14}$$

From Equations (4) and (14), it can be seen that $V_g(x) = Q_L(x, g(x))$. Thus, the optimal Q_L-function, denoted by Q_L^*, must agree with

$$Q_L^*(x,u) = r(x,u) + \gamma Q_L^*(f(x,u), g^*(f(x,u))). \tag{15}$$

The target of Equation (14) is given by the same structures that are presented in Equations (9)-(11). The greedy iteration is the policy used for determining the Q_L-function. The policy is parameterized as $g(x) = g(x,\kappa)$. The control law is obtained by the Bellman optimality equation for the Q_L-function, which is given by $u^* = \arg\max_u Q_L^*(x,u)$.

The squared errors between the model estimate and target are minimized, in view of improving through the greedy policy the optimal estimate of parameters. The new parameters yield new targets, so the process is computationally realizable.

3. **Bellman Equations for DLQR Control Systems.** The DLQR design, as a parameterization of Bellman equations, is the main topic of this section. The parameterizations of the state value function, the utility function, and the state and control policy mappings presented in Section 2 are shown herein for linear systems and discrete linear quadratic regulator (DLQR) problem. The Bellman equations are presented in the Lyapunov and Riccati forms which are linear and nonlinear for unknown parameters, respectively.

3.1. **DLQR parameterization.** For the DLQR control system, the parametric representation of Equation (1) for a linear time invariant system is given by

$$x_{k+1} = Ax_k + Bu_k, \tag{16}$$

where $A \in R^{n\times n}$, n is the order of the system, $x_k \in R^n$ is the state, $B \in R^{n\times n_e}$, n_e is the number of control inputs and $u_k \in R^{n_e}$ is the control input. The control action of Equation (2) is parameterized by the linear relation of the states which is given by

$$u_k = K(\cdot)x_k, \tag{17}$$

where $K(\cdot) \in R^{n_e\times n}$ is the feedback gain matrix.

The state feedback performed by the control law of Equation (17) provides the closed-loop system, which is given by

$$x_{k+1} = (A + BK)x_k. \tag{18}$$

The utility function associated with the system (16) and (17) has a quadratic form that is given by

$$r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k, \tag{19}$$

where $Q = Q^T \geq 0$ and $R = R^T > 0$ are state-weighting and control-weighting matrices, respectively.

By substituting the parameterization of the utility function of Equation (19) into Equation (3), and assuming the discount factor γ is equal to 1, one obtains the parameterized DLQR cost function. Specifically, the goal now is to determine the control law u that will

yield the smallest possible set of costs (negative rewards). Thus, the DLQR problem is formulated as the following optimization problem

$$\begin{aligned} \min_{u} \quad & \sum_{i=k}^{N} (x_i^T Q x_i + u_i^T R u_i) \\ \text{subject to} \quad & \\ & x_{k+1} = Ax_k + Bu_k. \end{aligned} \tag{20}$$

It can be shown that the optimal value associated with the solution of the DLQR problem, Equation (20), admits the following quadratic form, [22],

$$V(x_k) = x_k^T P x_k, \tag{21}$$

with $P \in R^{n \times n} > 0$ being symmetric.

Thus, the parameterized functions for the DLQR are classified into parameters A and B of the dynamic system (environment), parameter K of the control law (control policy), and parameters Q and R of the instantaneous cost (utility function). The cost solution is a state quadratic form parameterized by P which is used to represent the reward.

3.2. **Bellman-DLQR formulation.** The Bellman-DLQR equations provide the values of the parameter P as an approximate solution of the Riccati/Lyapunov equation.

From Equations (5), (19) and (21), one obtains the Bellman equation with no dynamic system model parameters (matrices A and B) for the discrete LQR, which is given by

$$x_k^T P x_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1}. \tag{22}$$

The optimal control law is obtained by minimization of Equation (22) with respect to u_k, which yields

$$R u_k + B^T P (A x_k + B u_k) = 0. \tag{23}$$

Equation (23) is solved for u_k and compared with Equation (17), so it follows that the optimal gain is given by

$$K = -(R + B^T P B)^{-1} B^T P A. \tag{24}$$

In terms of the feedback gain of Equation (17), and the dynamics of the closed loop system of Equation (18), Equation (22) is expressed by

$$\begin{aligned} x_k^T P x_k = x_k^T (Q + K^T R K + \\ + (A + BK)^T P (A + BK)) x_k. \end{aligned} \tag{25}$$

Since Equation (25) must be satisfied for all states x_k, one has a linear equation in P that is given by

$$P = Q + K^T R K + (A + BK)^T P (A + BK), \tag{26}$$

known as a Lyapunov equation.

The optimal control law u_k of Equation (23) is substituted into Equation (22). One thus obtains a parameterization of the HJB equation in form of the discrete time algebraic Riccati equation (DARE), which is given by

$$P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A. \tag{27}$$

This matrix equation is quadratic in the parameter P and is also referred to in this text as HJB-Riccati equation.

Equation (27) is dismembered into two equations: one for evaluation and the other for determining the optimal control law. The equations resulting from that dismembering are given by

$$P = Q + A^T PA + A^T PBK_{ric} \tag{28}$$

$$K_{ric} = -(R + B^T PB)^{-1} B^T PA. \tag{29}$$

Equations (28) and (29) suggest an iterative numerical process to solve the DLQR control policy, which is a policy iteration scheme consisting of the HJB equation solution and the control law. In this process, for each value of P given in Equation (28), one has an optimal policy K given by Equation (29).

4. **Problem Descriptions.** The problem is characterized in the context of least-squares estimation approach for the DLQR design, where the state value quadratic function is transformed in linear combination of states and DARE parameters. In this manner, the problem is characterized as the DARE solution via parametric methods. The problem formulation is presented in the first part, where the approximate solution of the HJB-Riccati equation is formulated in terms of the least-squares method.

The QR-tuning method is presented in the second part as a proposed methodology to tune optimal controllers based on relationships of reward matrices. The theoretical core of the proposed optimal control method is the discrete linear quadratic regulator design that provides the optimal control policy via approximate solutions of the Bellman equation. These solutions are guided by systematic variations in the values of cost matrices Q and R. The proposed methodology is oriented for on-line solutions of optimal controllers, i.e., to impose stability margins of DLQR design of the proposed method.

4.1. **HJB-Riccati equation solution via LS approximation.** The quadratic form of the optimal cost is represented by a transformation that uses the Kronecker product for obtaining a linear equation system in P (Riccati solution). By applying the Kronecker product and vectorization definitions [23], one has that the optimal cost value, quadratic form of Equation (21), is represented by

$$x^T Px = (x^T \otimes x^T) vec(P) \tag{30}$$

$$= (x_1 x^T \; x_2 x^T \; \ldots \; x_n x^T) vec(P) \tag{31}$$

$$= vec(xx^T)^T vec(P). \tag{32}$$

It is noticed that $x^T Px$ of Equation (32) can be represented as a dot product. Supposing that $x^T Px$ and $vec(xx^T)$ are known variables and $vec(P)$ is the unknown variable, the situation can be characterized as least-squares problem, because $vec(P)$ can be estimated based on the values of observations $x^T Px$ and regressor vectors $vec(xx^T)$.

The least-squares approach [24] is used to determine the parameters of the value function approximation. Equation (21) that minimizes the expected squared error is parameterized for

$$P_{k+1} = \arg\min_P E\{|x^T Px - \Theta(x, r, f, P_k)|^2\}. \tag{33}$$

The solution of Equation (33) by approximation and vectorization is represented by

$$\theta_{k+1} = \arg\min_\theta E\{|\bar{x}^T \theta - \Theta(x, r, f, \theta_k)|^2\}, \tag{34}$$

where $\bar{x} \in R^{n(n+1)/2}$ is defined according to the Kronecker product that is given by $\bar{x}^T = [x_1^2 \ldots x_1 x_n \;\; x_2^2 \ldots x_{n-1} x_n \; x_n^2]$.

The function $\theta = vec(P) \in R^{n(n+1)/2}$ of the square matrix P is a vector containing the n diagonal entries of P and the $n(n+1)/2 - n$ distinct sums $P_{ij} + P_{ji}$. Assuming an

ordering between the vector $\bar{x}$ and vectorization $vec(P)$ to represent the quadratic form $x^T P x = \bar{x}^T vec(P)$, the least-squares parametric estimate is given by

$$\theta_{k+1} = (E\{\bar{x}^T \bar{x}\})^{-1} E\{\bar{x}\Theta(x, r, f, P_k)\}. \tag{35}$$

The function $\Theta(x, r, f, P_k)$ of Equation (35) is the desired value which consists of the current cost and the cost function from the next state $f(x, u)$, as can be seen in Equation (9).

4.2. ***QR*-tuning kernel.** The QR-tuning method is based on the mapping K_{QR} from traces of weighting matrices Q and R into assignment of closed loop eigenvalues on the Z-plane, i.e., the QR-tuning mapping is given by

$$K_{QR} : \{\text{trac } Q, \text{trac } R\} \to \{\lambda_1, \lambda_2, \ldots, \lambda_n\}, \tag{36}$$

where trac Q and trac R are the traces of matrices Q and R, respectively, and $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the closed loop eigenvalues that are specified by the designer.

In terms of weighting matrices, the specified performance is imposed by a new control law, which is given by

$$u_k(QR) = K_{QR} x_k, \tag{37}$$

where K_{QR} is the controller gain that directly depends on the selection of the weighting matrices.

The heuristic QR-tuning method is based on relationships between the matrices Q and R. In order to obtain a QR relationship that provides a guide to heuristic search method, the Riccati recurrence of Equation (28) is replaced in Equation (29) of the gain. Consequently, the optimal gain is given by

$$K_{QR} = -(R + B_d^T P B_d)^{-1} B_d^T (Q + A_d^T P A_d + A_d^T P B_d K_{QR}) A_d, \tag{38}$$

where $K_{QR} = K(Q, R)$ is the gain matrix K_{ric} of Equation (29) as a function of the weighting matrices.

Considering the situations $R \gg B_d^T P B_d$ and $Q \gg A_d^T P A_d$ in which the quadratic forms of the input and of the state are quite smaller than the weighting matrices Q and R, it follows that

$$K_{ric}(Q, R) \approx K_{f1}(Q, R) + K_{f2}(R), \tag{39}$$

where

$$K_{f1}(Q, R) \approx -(R^{-1} B_d^T Q) A_d \tag{40}$$

$$K_{f2}(R) \approx (R^{-1} B_d^T A_d^T P B_d R^{-1} B_d^T P A_d) A_d. \tag{41}$$

Based on Equation (39), a heuristic is established to map the Z-plane. While the values of the elements of matrices R are kept fixed, the values of the elements of matrices Q are submitted to variations during the tuning process, with matrices Q and R being diagonal. The effectiveness of QR-tuning is evaluated by the traces of matrices Q.

Let $R = R_{fix}$ be a matrix with constant trace and $R_{f2} = B_d^T A_d^T P B_d R^{-1} B_d^T P A_d$. Consequently, Equation (41) of the Riccati gain is written in a form that allows to design a heuristic that is guided by

$$K_{ric}(Q, R_{fix}) \approx -(R_{fix}^{-1} B_d^T Q) A_d + (R_{fix}^{-1} R_{f2}) A_d. \tag{42}$$

As can be seen in Equation (42), if the restrictions $R \gg B_d^T P B_d$ and $Q \gg A_d^T P A_d$ are held, the changes of gains $K_{ric}(Q, R_{fix})$, for a fixed matrix R_{fix}, lead to eigenvalues changes in Z-plane for variations in the values of the elements of matrix Q. It is observed that if $R \gg 0$ for the case of diagonal matrices, one has that $K_{f2}(R) \approx 0$. In [12], the

authors present the first insights on QR-tuning in the context of offline design, in which the HJB equation solution is model-based.

Stability aspects in the implementation of an ADP in a feedback system were studied in [25]. The QR convergence analysis consists on the evaluation of the traces of matrices Q and R and their relationships with the eigenvalue assignment of MIMO systems in the Z-plane via optimal controllers. The results are presented in a table built following a heuristic which is established from Equations (40) and (41). The iterative process for systematic variations in the values of matrices Q and R follows a growing pattern of the trace of matrix Q, whereas the matrix R is an identity matrix during the entire solution process.

5. **HDP Algorithms.** Two HDP algorithms are developed to compute approximate solutions of the DARE. The first is based on the state value function approximation, and the second is based on the Q_L approximation of the value function. The development of approximate algorithms for HDP and AD-HDP, which are oriented for realization of discrete optimal control, is presented in this section. The foundations of optimal control required for the development of the above mentioned algorithms can be found in [22, 26]. The discrete control approaches, such as DLQR, are highlighted in references [27, 28].

5.1. **Actor-critic schemes for DLQR.** The actor and critic schemes for reinforcement learning are presented in terms of least-squares method and optimal policy for the DLQR control system design.

5.1.1. *Critic DLQR scheme.* The parameterizations of the environment of Equation (16), the control policy of Equation (17), and the instantaneous cost of Equation (19) establish the parameterized function approximation $\Theta(x, r, f, P)$ for DLQR which is given by

$$\begin{aligned}\Theta(x, r, f, P) = x^T Q x + u^T R u + \\ +(Ax + Bu)^T P (Ax + Bu).\end{aligned} \tag{43}$$

The theories of matrix vectorization and Kronecker product contribute for the Riccati equation solution that is approximated by a linear equation system for the coefficients of the matrix P. Consequently, the DARE solution approximation is a least-squares solution that is given by

$$\overline{x}^T \theta = \Theta(x, r, f, P^\theta), \tag{44}$$

where θ is the vector corresponding to the elements of the Riccati matrix. The regression vector and optimal cost are given by

$$\overline{x} = [x_{1k}^2 \dots x_{1k} x_{nk} \; x_{2k}^2 \dots x_{n-1k} x_{nk} \; x_{nk}^2]^T \tag{45}$$

$$\Theta(x, r, f, P^\theta) = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P^\theta x_{k+1}, \tag{46}$$

where x_{lk} is the l-th, $l = 1, 2, \dots, n$, component of the state vector x at time k.

5.1.2. *Actor DLQR scheme.* The optimal parameters of the approximation satisfy the condition of Equation (12), with $\kappa = vec(K)$, i.e., parameter κ is a vectorization of the gain matrix K. Assuming $\frac{\partial g}{\partial \kappa} \neq 0$, Equation (13) reduces to the

$$\frac{\partial r}{\partial g} + \frac{\partial V}{\partial f} \frac{\partial f}{\partial g} = 0. \tag{47}$$

By replacing the derivatives $\frac{\partial r}{\partial g} = 2u^T R$, $\frac{\partial V}{\partial f} = 2f^T P$ and $\frac{\partial f}{\partial g} = B$ in Equation (47), one has that the optimal policy u is given by

$$u = K(P^\theta) x = Kx, \tag{48}$$

where $K = -(R + B^T P^\theta B)^{-1} B^T P^\theta A$ is the optimal gain.

5.2. **State HDP algorithm.** The algorithm is developed according to Equations (43) and (44) for the main core. The steps for determining the optimal control policy are presented in Algorithm 1. This algorithm is made up of two segments that can be classified on initial conditions established in steps 2-10 and on iterative process that are instructions established from steps 12-34.

Algorithm 1 – HDP Algorithm – Least-Squares and Recurrence Step

PDynamic-DLQR-HDP(N)

```
1  ------------------------------------------------------------
2  ▷ - Setup
3  ▷ - Dynamic Systems
4  [A_d, B_d, t_samp, x_inic, x_revit] ← [      ]
5  ▷ - Weighting and Dynamic Systems
6  [Q, R, q_i, r_i] ← [      ]
7  ▷ - Iterative Process Parameters
8  [N, n_rec] ← [      ]
9  ▷ - P and K Initial Values
10 [P_0^θ, K_0^θ] ← [      ]
11 ------------------------------------------------------------
12 ▷ Iterative Process
13 for i ← 1 : N
14       do
15          ▷ Optimal Policy
16          u_i ← K^θ x_i
17          ▷ States
18          x_{i+1} ← A_d x_i + B_d u_i
19          ▷ Basis Set - Kronecker Product
20          x̄_i ← [x_1i^2; x_1i x_2i; x_1i x_3i; x_2i^2; x_2i x_3i; x_3i^2]
21          ▷ Target Vector Assembling
22          Θ(x, r, f, P^θ) ← x_i^T Q x_i + u_i^T R u_i + x_{i+1}^T P^θ x_{i+1}
23          ------------------------------------------------------
24          if i%n_rec = 0
25            then
26                 ▷ Least-Squares
27                 θ ← (x̄ x̄^T)^{-1} x̄ Θ(x, r, f, P^θ)
28                 ▷ Matrix P^θ recovery from vector θ
29                 P^θ ← [θ_1,   θ_2/2,   θ_3/2;
30                        θ_2/2,   θ_4,   θ_5/2
31                        θ_3/2,   θ_5/2,   θ_6
32                 ▷ Feedback Optimal Gain K
33                 K^θ ← -(R + B_d^T P^θ B_d)^{-1} B_d^T P^θ A_d
34                 x_{i+1} ← x_revit
35          ------------------------------------------------------
36 End - Iterative Process
```

The setup segment of Algorithm 1 consists of the following information:

$$I_{setup}^{HDP} = \left\{ \; I_{c1}, \quad I_{c2}, \quad I_{c3}, \quad I_{c4} \; \right\}, \tag{49}$$

where I_{c1}, I_{c2}, I_{c3} and I_{c4} are the sets of information that are related to the dynamic system, instantaneous reward weightings, iterative process parameters and initial conditions of the Riccati solution, and the optimal gain, respectively. In terms of their components, these sets for the HDP are established by the enumeration of the elements of setup segment which are given by

$$\begin{aligned} I_{c1} &= \left\{ \; A_d, \quad B_d, \quad x_{inic}, \quad t_{samp}, \quad x_{revit} \; \right\} \\ I_{c2} &= \left\{ \; Q, \quad R, \quad q_i, \quad r_i \; \right\} \\ I_{c3} &= \left\{ \; N, \quad n_{rec} \; \right\} \end{aligned}$$

$$I_{c4} = \left\{ \; P_0^{\theta}, \;\; K_0^{\theta} \; \right\},$$

where A_d and B_d are the discrete model matrices, t_{samp} is the sampling time, x_{inic} and x_{revit} are system initial conditions and revitalization of the states (state resetting), respectively, which constitute the set I_{c1}, corresponding to step 4 of Algorithm 1. The elements Q and R are the weighting matrices and the pair (q_i, r_i) represents variations in the values of the elements of respective matrices in set I_{c2} that correspond to step 6. For the set I_{c3}, the parameters N and n_{rec} are the number of iterations and delays for updating the parameters with respect to a given amount of interval (time window), respectively, for updating the vectorization parameter θ that corresponds to step 8. For the set I_{c4}, P_0^{θ} and K_0^{θ} are the initial conditions of the Riccati equation and the gain, respectively, that corresponds to step 10.

The second segment of the standard HDP algorithm implements the updates of the regression vector in step 20 as well as the value function in step 22 for each reading of the sampling interval. It also implements the updates of the parameter θ vector in step 27, the matrix P^{θ} recovery from vector θ in step 29 and the gain K^{θ} in step 33. This sequence of steps occurs for a time window, called recurrence step, given for values greater than or equal to $(n(n+1)/2)t_{samp}$ time units.

5.3. **AD-HDP algorithm.** The DLQR-AD-HDP heuristic algorithm has the same parameterizations for the dynamic system of Equation (16), the control policy of Equation (17), and the instantaneous cost of Equation (19). The Q_L-function of Equation (14) is parameterized by

$$Q_L(x, u) = z^T Q_{LV} z, \tag{50}$$

where $z^T = [x^T \;\; u^T]$, and Q_{LV} is the learning matrix associated with the Q_L-function that is given by

$$Q_{LV} = \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix}, \tag{51}$$

where $Q_{LV} \in R^{(n+n_e)\times(n+n_e)}$ and the matrices Q_{xx}, Q_{xu}, Q_{ux} and Q_{uu} represent the weightings of state x and the policy u.

5.3.1. *DLQR-AD-HDP development.* The minimization of the parameterized Q_L function of Equation (50) provides the means for determining the optimal policy u. The gradient equation $\partial Q_L/\partial u = 0$, when solved for u, provides the optimal policy. Consequently, the optimal control law is given by

$$u = K(Q_{LV})x, \tag{52}$$

where $K(Q_{LV}) = -Q_{uu}^{-1}Q_{ux}$.

The parameterization of $Q_L(Q_{LV})$ induces a parameterization of the policy $g(Q_{LV})$. According to the parameterization $u = g(x, \kappa)$ of control policy, the parameter vector κ is a vectorization of the gain matrix K, i.e., $\kappa = vec(K)$. In comparison with HDP, parameters besides the ones existing in the Q_L-function are not needed, i.e., AD-HDP is a model-free method in which all information is picked up from the states and actions.

5.3.2. *Q_L-function estimation.* The estimation of parameters of the Q_L-function uses the iterative scheme of Equations (9)-(11) for estimating the Q_L-function. After considerations on the results of vectorization, one has that the DARE solution approximation by the least-squares method has its final form given by

$$\bar{z}^T \bar{z} \theta_{k+1} = \bar{z}^T \Theta(z, r, g(f), K^{\theta}), \tag{53}$$

where $\Theta(z, r, g(f), K^\theta)$ is the value function model which is given by

$$\Theta(z, r, g(f), K^\theta) = x_k^T Q x_k + u_k^T R u_k + \begin{bmatrix} x_{k+1} \\ K^\theta x_{k+1} \end{bmatrix}^T Q_{LV} \begin{bmatrix} x_{k+1} \\ K^\theta x_{k+1} \end{bmatrix}. \tag{54}$$

Algorithm 2 – AD-HDP Algorithm – Least-Squares and Recurrence Step

PDynamic-DLQR-AD-HDP(N)

```
1  ------------------------------------------------------------
2  ▷ - Setup
3  ▷ - Dynamic Systems
4  [A_d, B_d, t_samp, x_inic, x_revit] ← [      ]
5  ▷ - Weighting and Dynamic Systems
6  [Q, R, q_i, r_i] ← [      ]
7  ▷ - Iterative Process Parameters
8  [N, n_rec] ← [      ]
9  ▷ - K and Matriz Q_LV Initial Values
10 [K_0^θ, Q_LV0^θ] ← [      ]
11 ------------------------------------------------------------
12 ▷ Iterative Process
13 for i ← 1 : N
14       do
15          ▷ Control Noise
16          e_i ← [   ]
17          ▷ Control Action
18          u_i ← K^θ x_i + e_i
19          ▷ States
20          x_{i+1} ← A_d x_i + B_d u_i
21          ▷ Total Cost Assembling
22          Θ_{Q_LV} ← x_i^T Q x_i + u_i R u_i +
23                [x_{i+1}; K^θ x_{i+1}]^T Q_L [x_{i+1}; K^θ x_{i+1}];
24          ▷ Basis Set - Kronecker Product
25          z̄_i ← [x_{1i}^2;
26                 x_{1i}x_{2i}; x_{1i}x_{2i}; x_{1i}x_{3i}
27                 x_{1i}u_{1i}; x_{1i}u_{2i}
28                 x_{2i}^2; x_{2i}x_{3i}; x_{2i}u_{1i}
29                 x_{2i}u_{2i}; x_{3i}^2; x_{3i}u_{1i}
30                 x_{3i}u_{2i}; u_{1i}^2; u_{1,i}u_{2,i}
31                 u_{2,i}^2];
32          ------------------------------------------
33          if i%n_rec = 0
34            then
35                 ▷ Matrix Q_LV
36                 θ ← (z̄ z̄^T)^{-1} z̄ Θ_{Q_L}
37                 Q_LV^θ ← [θ_1   θ_2/2   θ_3/2   θ_4/2   θ_5/2;
38                              θ_6,   θ_7/2   θ_8/2   θ_9/2
39                                     θ_10,   θ_11/2   θ_12/2
40                                             θ_13   θ_14/2
41                                                    θ_15]
42                 ▷ Separation in Partitions
43                 Q_ux ← Q_LV(n + 1 : n + ne, 1 : n)
44                 Q_uu ← Q_LV(n + 1 : n + ne, n + 1 : n + ne)
45                 ▷ Feedback Optimal Gain K
46                 K^θ ← −(Q_uu)^{-1} Q_ux
47                 x_{i+1} ← x_revit
48          ------------------------------------------
49 End - Iterative Process
```

5.3.3. *AD-HDP algorithm.* The core of Algorithm 2 for the AD-HDP is assembled according to Equation (52), representing the action, and Equation (54) that represents the observation as a function of states and action. Finally, in order to obtain the approximation of HJB-Riccati equation solution, the data from those two equations are processed by

Equation (53). Algorithm 2 has the same operational procedure as Algorithm 1 of state HDP, within the context of the setup segment and the iterative process. The difference between the two approaches is clear by the increase of the regressor vector dimension and by the straightforward determination of the optimal gain. It is not necessary to determine the Riccati solution.

Similar to HDP, a setup segment is given by

$$I_{setup}^{AD-HDP} = \left\{ \; I_{c5}, \quad I_{c6}, \quad I_{c7}, \quad I_{c8} \; \right\}, \tag{55}$$

where I_{c5}, I_{c6}, I_{c7} and I_{c8} are sets of information that are related to the dynamic system, instantaneous reward weightings, iterative process parameters and initial conditions of the optimal gain, respectively. In terms of their components, these sets for the AD-HDP are given by

$$\begin{aligned} I_{c5} &= \left\{ \; A_d, \quad B_d, \quad x_{inic}, \quad t_{samp}, \quad x_{revit} \; \right\} \\ I_{c6} &= \left\{ \; Q, \quad R, \quad q_i, \quad r_i \; \right\} \\ I_{c7} &= \left\{ \; N, \quad n_{rec} \; \right\} \\ I_{c8} &= \left\{ \; K_0^{\theta}, \quad Q_{LV0}^{\theta} \; \right\}, \end{aligned}$$

where A_d and B_d are the discrete model matrices, t_{samp} is the sampling time, x_{inic} and x_{revit} are system initial conditions and revitalization of the states, respectively, which constitute the set I_{c5}, corresponding to step 4 of Algorithm 2. The elements Q and R are the weighting matrices and the pair (q_i, r_i) of parameters represents variations in the values of the elements of respective matrices in the set I_{c6} that correspond to step 6. For the set I_{c7}, the parameters N and n_{rec} correspond to the number of iterations and delays for updating the parameters with respect to a given amount of interval (time window), respectively, for updating the vectorization parameter θ, corresponding to step 8. For the set I_{c8}, K_0^{θ} and Q_{LV0}^{θ} are the initial conditions of the Riccati equation and the gain, respectively, that corresponds to step 10.

The updates of the value function in step 22 and the regression vector in step 25 for each reading of the sampling interval are implemented in the second segment of the Algorithm 2 of the AD-HDP. The updates of the parameter vector θ are implemented in step 36 by the least-squares method. Following that, the matrix Q_{LV}^{θ} recovery is carried out from θ in step 37, according to the rule previously established in the formulation of the AD-HDP equations. The conclusion of one recurrence process cycle occurs in step 46 with the update of the gain matrix K^{θ} for delays that are greater than or equal to $((n + n_e)(n + n_e + 1)/2)t_{samp}$.

5.4. **Features of HDP and AD-HDP algorithms.** The features of HDP and AD-HDP algorithms are understood as common relations and differences between the two approaches, aiming at the ADP solution for determining the optimal control policy for the DLQR design.

The setup parameters of the HDP and AD-HDP algorithms contribute to increasing the convergence speed, or lead to solutions diverging from the true values, or cause other types of events that have an impact on the algorithm performance. During the solution process, the behavior of algorithms is evaluated for variations of the parameters established by the setup segments I_{setup}^{HDP} and I_{setup}^{AD-HDP}.

The non-null state condition [29] for design purposes, step 34 of the HDP algorithm and step 47 of the AD-HDP algorithm, contributes to overcoming problems of ill-conditioning of the regression matrix. The convergence strategies are established according to the process. In this case, the algorithm has a persistence restarting due to null state problems

that lead to the linearly dependent columns in the matrix of Equation (44) of the HDP and AD-HDP algorithms.

Heuristics are established according to the development of the algorithm for convergence. In this case, the algorithm has a persistence restarting to overcome null state problems that lead to the regression matrix with null rank.

5.5. **HDP convergence.** The convergence equation represents the variation of the discrete HJB equation solution. In its most primitive form, this equation is presented as a weighting of the difference between the k starting situation and the $k+1$ arriving situation (point, time, stage), i.e., to update the HJB solution from the starting solution P_k to next solution P_{k+1}. This situation is represented by

$$P_{k+1}^{\theta} = P_k^{\theta} + \alpha \left[P_k^{-\theta} - P_k^{\theta} \right], \tag{56}$$

where $\{P_k^{\theta}\}$ is the matrix sequence generated by the optimality ensured by the HJB equation, and $P_k^{-\theta}$ is the value obtained from the Lyapunov recursion which is given by

$$\begin{aligned} P_k^{-\theta} &= Q + K_k^T(\theta) R K_k(\theta) + \\ &\quad + (A + BK_k(\theta))^T P_k^{\theta} (A + BK_k(\theta)). \end{aligned} \tag{57}$$

In the convergence analysis, one evaluates the sequence $\{P_k\}_{k=0}^{\infty}$ in the recurrence of Equation (57) in the form of Lyapunov. The HJB gain is given by

$$K_k(P_k^{\theta}) = -(R + B^T P_k B)^{-1} B^T P_k A. \tag{58}$$

Equation (57) ensures the HJB optimality, due to its development resulting in the HJB gain of Equation (58). And also, Equation (57) is a primitive that originates the discrete Ricatti algebraic equation by its optimization (minimization) with respect to the decision $u_k = K_k x_k$ that ensures the optimal value, as can be confirmed in [12].

The optimal value P^* is the DARE positive definite solution which is given by

$$P^* = Q + A^T (P^* - P^* B (R + B^T P^* B)^{-1} B^T P^*) A. \tag{59}$$

Equation (56) obeys the following conditions $0 < \alpha \leq 1$, $R > 0$, $Q \geq 0$, and the pair (A, B) is controllable. Thus, design specifications that ensure the optimality to evaluate the behavior of the process are established until the target controller is achieved, i.e., $P_k \to P^*$ as $k \to \infty$ in the recurrence relation of Equation (56). When customized for Lyapunov relation of Equation (56), one has

$$\begin{aligned} P_{k+1} &= P_k + \alpha \left[Q + K_k^T R K_k + \right. \\ &\quad \left. + (A + BK_k)^T P_k (A + BK_k) - P_k \right]. \end{aligned} \tag{60}$$

6. **Computational Experiments.** The procedures to evaluate the performance of QR-tuning heuristics and approximate solutions of the HJB equation are established in a set of computational experiments that consists of three main parts: a) environment simulator and initial conditions, b) experiments to evaluate state and action-dependent HDP algorithms to eigenstructure assignment, c) experiments to evaluate the QR-tuning. The first part deals with the dynamic model that represents the actor actions on the environment by means of actuators as well as the real world system reactions to the actor that are captured via sensors. The dynamic system model and the setup of the computational experiments are presented in Appendix A.

The second part of the procedure consists of comparative analysis to evaluate the the approximate solution of the HJB equation via least-squares estimation method and adaptive dynamic programming paradigms, such as state and action-dependent HDPs. The performance analysis is based on the Schur method to evaluate the accuracy of the

approximate solutions (decisions), i.e., the approximate solution of the HJB equation for a fixed pair of matrices Q and R. The trace of the closed loop matrix is used to evaluate the eigenstructure assignment.

The third part of procedure is the evaluation of QR-tuning heuristics that are designed to sweep the Z-plane through variations in the weighting matrices values of utility function of DLQR design. In these experiments, the traces of matrices Q and R are the main vehicle to guide the heuristic search of these matrices.

The third-order model that represents an aircraft is used to evaluate the performance of the HDP and AD-HDP algorithms for optimal controller design. The time evolution of the solutions of matrix P_k^{θ} is evaluated for both approaches. The steady state solutions are compared with the elements generated by the Schur method.

6.1. **DLQR-Schur design.** The computational solutions of the discrete HJB equation, and the optimal gain by the Schur method [19] are compared with the values of P and K for state HDP and AD-HDP approaches. The Schur solution for the dynamic system that is presented in Appendix A is given by

$$P_{\infty}^{schur} = \begin{bmatrix} 5.3947 & 0.7427 & -0.0078 \\ 0.7427 & 1.6203 & -0.0067 \\ -0.0078 & -0.0067 & 1.1037 \end{bmatrix}. \tag{61}$$

Consequently, the DLQR optimal policy of Equation (58) is given by

$$K_{\infty}^{schur} = \begin{bmatrix} 0.0050 & 0.0039 & -0.1782 \\ -0.5632 & -0.6129 & 0.0066 \end{bmatrix}. \tag{62}$$

The values of the HJB equation solution and the optimal gain given by (61) and (62), respectively, are used to compare the accuracy level of the approximate solutions, taking into account that the Schur values are the true values [19].

6.2. **State HDP experiments.** Algorithm 1 for the HDP, Section 5.2, is evaluated in third-order system. The time evolution of the solutions of matrix $P_k^{\theta-HDP}$ is compared with the true value of $P_{\infty}^{\theta-HDP}$ that corresponds to the Schur solution P_{∞}^{schur}, as presented in Section 6.1.

The analysis of the state HDP-QR-tuning and approximate HJB solutions is based on the convergence relations of Subsection 5.5. For state HDP algorithms, Equation (56) is written as

$$P_{k+1}^{\theta-HDP} = P_k^{\theta-HDP} + \alpha_{HDP}\left[P_{k+1}^{\theta} - P_k^{\theta-HDP}\right], \tag{63}$$

where $P_k^{\theta-HDP}$ and $P_{k+1}^{\theta-HDP}$ are the estimate solutions of the HJB equation in steps k and $k+1$, respectively, and α_{HDP} is a learning factor, with $0 < \alpha_{HDP} \leq 1$. The value $P_k^{-\theta}$ in Equation (56) that was obtained from the Lyapunov recursion of Equation (57) for DLQR design is model-based. Herein, the estimate solution $P_k^{-\theta}$ is model-free, with such an estimative being obtained by the least-square method.

The evolution of the Riccati equation solution of Figure 1 is associated with the convergence equation of the approximate solution, Equation (56), in state HDP approach. The curves of Figure 1 show that the stability of $P_{\infty}^{\theta-HDP}$ lies in the interval $4 < t_{RcT} < 11$, where t_{RcT} is the number of recurrence time steps. The recurrence interval is defined by a full updating of the vectors $\bar{x}$, according to Equation (45) and the target vector $\Theta(x, r, f, P^{\theta})$, according to Equation (46), e.g., for step 27 of Algorithm 1, the computation of inverse of the matrix $(\overline{x}\ \overline{x}^T)$ is performed after a set of six sampling time steps, as represented in the graphs of Figure 1, during the iterative process until reaching the steady state solution.

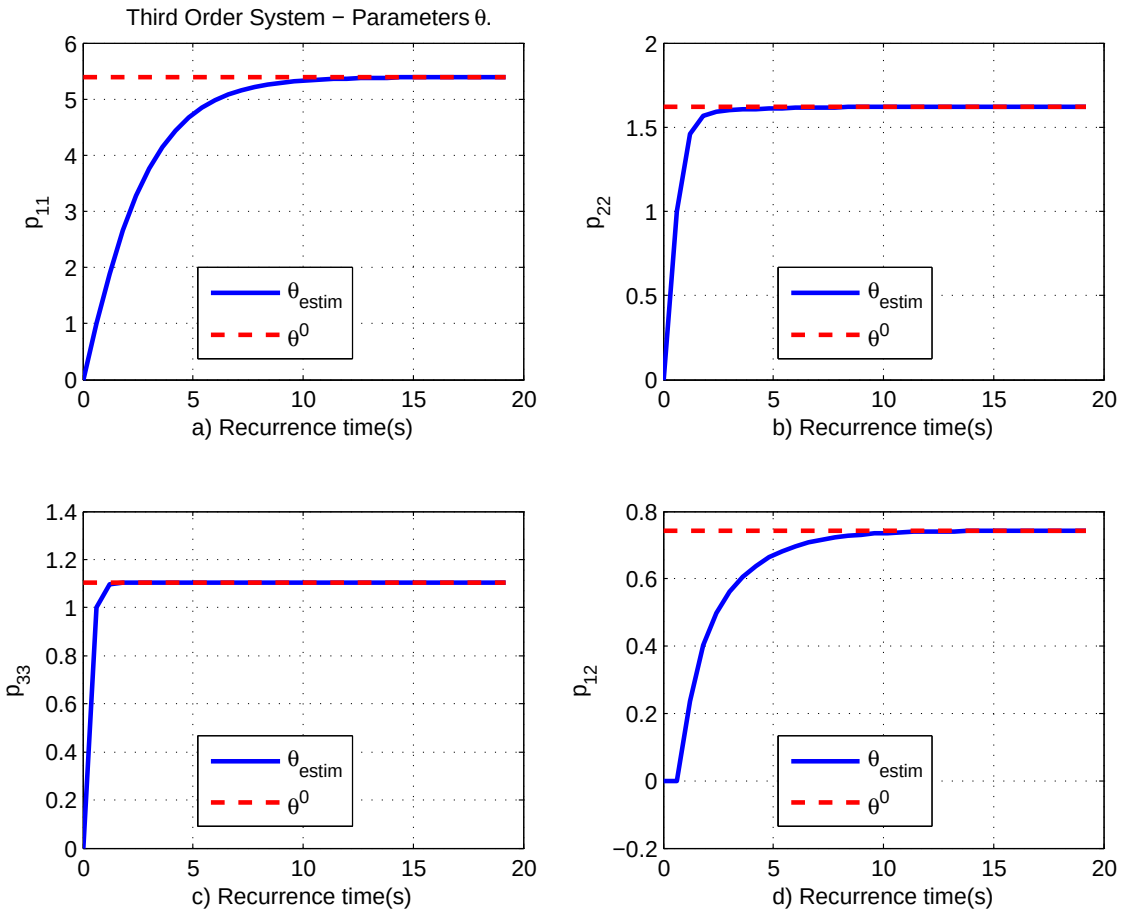

FIGURE 1. Parameters θ of matrix P – Third-order system

The graphs a), b) and c) of Figure 1 represent the diagonal elements of matrix P_k^{θ}. It also shows the evolution of the parameter θ_2 that corresponds to the element P_{12}^{θ} in graph d). The remaining elements of the matrix $P_{\infty}^{\theta-HDP}$ tend to the steady state value of the Schur solution, Equation (61), when $k \to \infty$. It is observed that the maximum time to reach the steady state values in all graphs is around sixty sampling time steps, corresponding to the ten recurrence time steps. The steady state solution associated with the parametric evolution of Figure 1 is given by

$$P_{\infty}^{\theta-HDP} = \begin{bmatrix} 5.3691 & 0.7399 & -0.0078 \\ 0.7399 & 1.6200 & -0.0067 \\ -0.0078 & -0.0067 & 1.1037 \end{bmatrix}. \tag{64}$$

The optimal policy is given by

$$K_{\infty}^{\theta-HDP} = \begin{bmatrix} 0.0050 & 0.0039 & -0.1782 \\ -0.5632 & -0.6128 & 0.0066 \end{bmatrix}. \tag{65}$$

The comparison of steady state value of the matrix $P_{\infty}^{\theta-HDP}$, given by Equation (64), with the Schur solution of Equation (61) and the convergence Equation (56) shows that the values of the elements of the approximate matrix have an accuracy of at least up to the first decimal place.

An important observation is about a comparison of the curves a), b), c) and d) with Equation (61). It is observed that around $t_{RcT} > 12$, $P_{k+1}^{\theta-HDP}$ is a good approximation of $P_k^{\theta-HDP}$, because $P_{k+1}^{\theta} - P_k^{\theta-HDP} = 0$ and P_{k+1}^{θ} is a model-free estimation in HDP

approach. Consequently, as can be observed in the Schur solution (61), convergence Equation (63) and computational experiments results of Figure 1, the iterative LS process reaches a neighborhood of the true value, when the approximation $P_{k+1}^{\theta-HDP}$ is the value $P_{\infty}^{\theta-HDP}$ of Equation (64), i.e., the value $P_{k+1}^{\theta-HDP}$ tends to P_{∞}^{schur}. For this operation point of dynamic system, the system reaches its stability taking into account a pair of weighting matrices Q and R with their respective values being given by identity matrices.

The gain matrix of Equation (65) is a straightforward application of $P_{\infty}^{\theta-HDP}$ to compute $K_{\infty}^{\theta-HDP}$ which is given by $-(R + B_d^T P^{\theta} B_d)^{-1} B_d^T P^{\theta} A_d$, and according with step 33 of Algorithm 1, this step implements Equation (24). As can be observed in optimal gain relations, if $P_k^{\theta-HDP}$ converges to P_{∞}^{schur}, the gain $K_{\infty}^{\theta-HDP}$ will converge to K_{∞}^{schur} in state HDP paradigm.

6.3. **AD-HDP experiments.** Algorithm 2 of Section 5.3.3 implements Equation (52) for determining the DLQR control gain without direct approximate solution of the HJB-Riccati equation. For state HDP algorithms, one has that the gain determination is performed in two steps: a) computation of approximate solutions of the Riccati/Lyapunov equation, and b) the gain matrices computation, as can be seen in steps 27-31 and 33 of Algorithm 1, respectively. However, in AD-HDP, only one step is needed in order to obtain the gain K_k^{AD-HDP}, as can be seen in step 46 of Algorithm 2.

For the AD-HDP algorithms, the convergence equation for analysis of the state HDP-QR-tuning and approximate solutions is similar to Equation (63), and it is given by

$$K_{k+1}^{\theta-AD} = K_k^{\theta-AD} + \alpha_{AD} \left[K_{k+1}^{\theta} - K_k^{\theta-AD} \right], \tag{66}$$

where $K_{k+1}^{\theta-AD}$ is the updated optimal gain of DLQR design, Equation (52), $K_k^{\theta-AD}$ is the current gain, and α_{AD} is a learning factor, with $0 < \alpha_{AD} \leq 1$. In step $k+1$, the gain value K_{k+1}^{θ} is obtained via least-squares estimation of the matrix $Q_L(x, u)$ that is given by Equation (50).

The evolution of the gain matrix elements is presented in Figure 2. The graph a) shows that the first line of the gain matrix K has a convergence less than five recurrence time steps, while elements of line 2 of the gain matrix have a convergence of about twelve recurrence time steps. In general, it is observed that the stability of P is reached after twelve recurrence time steps. The recurrence interval is defined by a full updating of the vectors $\bar{x}$. In this case, the AD-HDP is constructed with fifteen parameters and the target Q_L which is defined by the HDP solution structure.

The steady state optimal policy associated with the parametric evolution of Figure 2 is given by

$$K_{\infty}^{AD-HDP} = \begin{bmatrix} 0.0045 & 0.0038 & -0.1782 \\ -0.5166 & -0.6077 & 0.0066 \end{bmatrix}. \tag{67}$$

It is observed that the gain value K_{AD-HDP} in Equation (67) converges for the Schur gain value in Equation (62).

6.4. ***QR*-tuning experiments.** In these experiments, results and convergence analysis of the QR-tuning and least-squares approximate solutions of the HJB equation for online DLQR design approaches are presented. The results consist of estimate solutions P_k^{θ} and K_k^{θ} for state HDP and AD-HDP via least-squares estimation, respectively. The steady state HDP and AD-HDP approximate solutions are compared with the Schur method, which is model-based and it is very efficient for suboptimal DLQR design.

Traces of the closed loop matrices are used to evaluate the recurrence process behavior to map or assign eigenvalues in the stable Z-plane. The traces of the closed-loop matrices

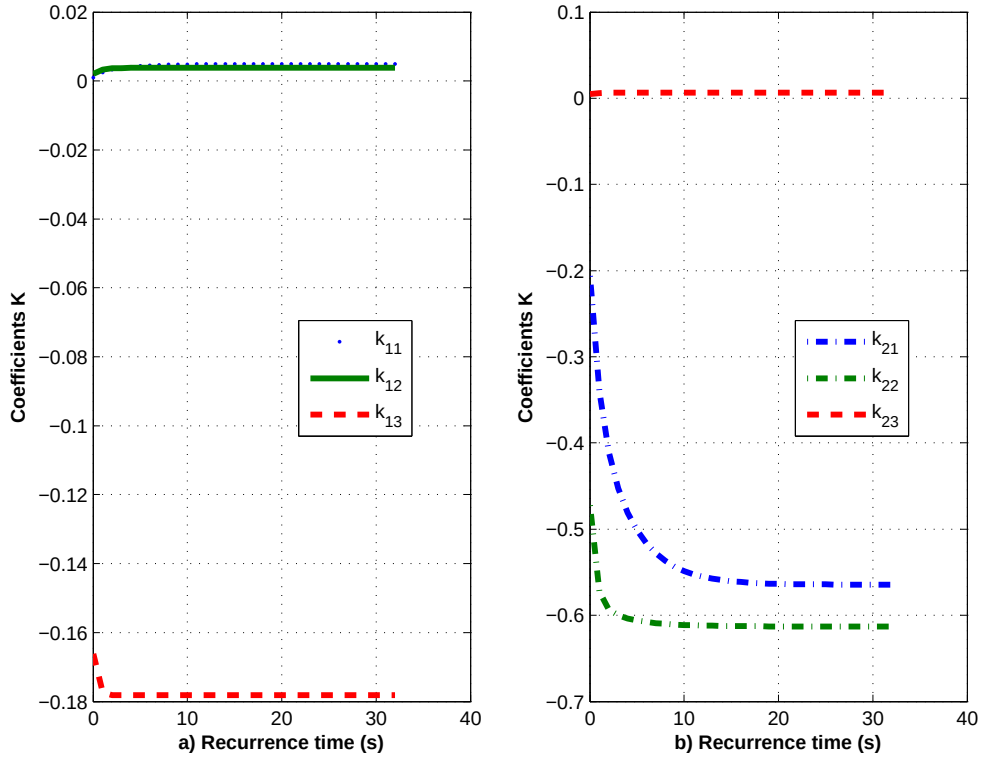

FIGURE 2. AD-HDP gains – Third-order system

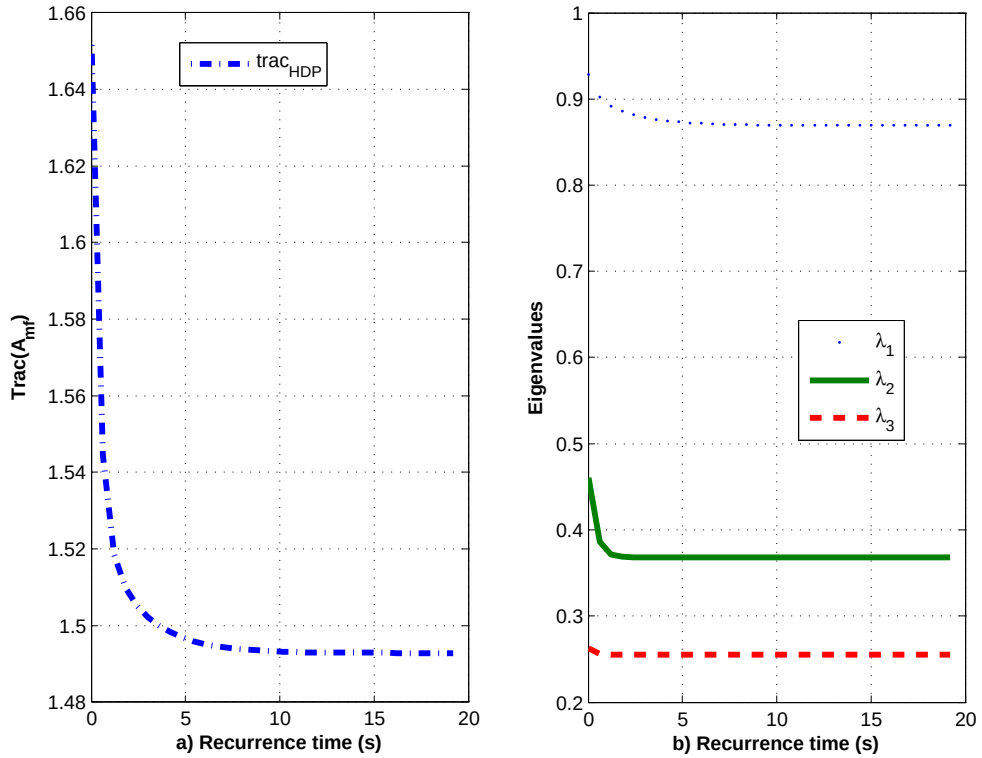

FIGURE 3. HDP traces – Third-order system

for the HDP recurrence process are shown in graph a), and the eigenvalues associated with those traces are shown in graph b) of Figure 3.

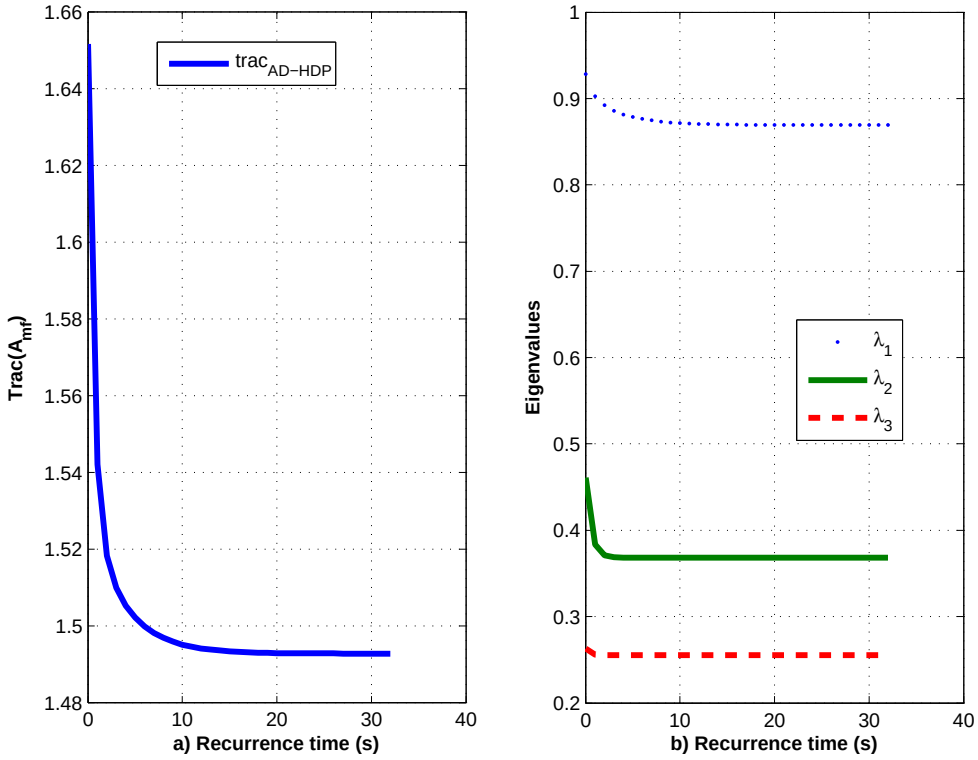

FIGURE 4. AD-HDP traces – Third-order system

The results for the action-dependent HDP are shown in Figure 4. The traces of the closed-loop matrices for the HDP recurrence process, generated by the HDP algorithm, converge faster than the traces generated by the AD-HDP algorithm.

The features of convergence speed between the two approaches, that can be interpreted according to the order of regression vectors, are related by the traces of the closed loop matrices. The regression vectors in the AD-HDP algorithm have a higher order than that of the regression vectors in the HDP algorithm. Thus, the number of samples to perform the updates should be 2.5 (sampling intervals) greater in the AD-HDP, i.e., the recurrence interval (step) is fifteen times the number of samples, while in the state HDP algorithm is six times the sampling interval. Consequently, the eigenvalues λ_{AD-HDP} achieve the region of Z-plane faster than eigenvalues λ_{HDP}, as can be seen in graph b) of Figures 4 and 3, respectively.

Associations between the traces of the weighting matrices Q and R are used to map the dynamic systems eigenvalues in stable Z-plane, according to the designer specifications. This task is performed in state space design as linear combinations of the state vector, whereas the decision law u_k is responsible for the transformation.

The heuristics are set according to the directives of [12], where the QR-tuning method is model-based. In the state and action-dependent HDP designs that are model-free, the dynamic of the states is captured by sensors. Specifically, the heuristic is based on an approximation of the relation (42) to guide the heuristic search by the duality principle. Therefore, for a given pair (Q, R_{fix}) of weighting matrices, the QR-tuning heuristic rule for the gain matrix is represented by

$$K_{ric}^{rule}(Q, R_{fix}) \approx -R_{fix}^{-1} B_d^T Q A_d, \tag{68}$$

where $K_{ric}^{rule}(Q, R_{fix})$ is a gross approximation of (42) to guide the QR-tuning search of online HDP-DLQR design. The arguments (Q, R_{fix}) of the decision rule $K_{ric}^{rule}(.)$ are the

matrices R with constant traces along the tuning process, while for matrices Q, the traces are not constant, and for both Q and R matrices, their respective diagonal elements are all equal, for each selected pair (Q, R) by the QR-tuning method.

The uniform variations in the values of diagonal matrix Q are evaluated for the number of iterations of the HDP convergence process as well as for the eigenvalue assignment, as shown in Table 1. The numbers in column q_i represent the values of the exponents of matrices $Q(q_i) = 10^{q_i} I_{33}$, where I_{33} is the identity matrix of third order. For this situation that is represented in the first column of Table 1, the exponents of $Q(q_i)$ are given by $q_i = \{4, 0, -2, -3\}$. Analogous reasoning follows for the matrices R, $R(r_i) = 10^{r_i} I_{22}$, where I_{22} is the identity matrix of second order, in this case $r_i = 0 \ \forall \ i$. Third column of this table represents the number of recurrence time steps that is given by N_{RcT}. The values of traces Q_{Trace} are shown in the fourth column. The closed-loop eigenvalues that are associated with the pairs (Q, R) are shown in the fifth column.

The results of the computational experiments for the HDP algorithm are presented in Table 1. Variations in the matrix Q values, and the eigenvalues associated in the Z-plane, are used to eigenstructure assignment.

For the AD-HDP algorithm, the variations in the matrix Q values, and the eigenvalues associated in the Z-plane, are shown in Table 2. The variations in the matrix Q values are given by $q_i = \{10, 2, 0, -2, -5, -10\}$. Observing the steady state values of approximations, e.g., the situation that is characterized by $q_i = -2$ and $r_i = 0$ in Tables 1 and 2 is compared with the Schur method of Table 3. This situation shows that the closed-loop eigenvalues for the AD-HDP algorithm converge faster than the eigenvalues generated by the HDP algorithm, because HDP and AD-HDP spent 180 and 90 sampling time steps, respectively, to reach the steady value. As can be seen, the steady state values are very close to the eigenvalues of Table 3, so, all eigenvalues of the associated situations are very good approximations.

The results for the action-dependent HDP are shown in Figure 4. The traces of the closed-loop matrices for the HDP recurrence process, generated by the HDP algorithm, converge faster than the traces generated by the AD-HDP algorithm.

The eigenvalues and traces of the closed loop system, which has their optimal gains determined by the Schur method, are presented in Table 3. These values are used for

TABLE 1. HDP algorithm and matrix Q variations

q_i	r_i	N_{RcT}	Q_{Trace}	Eigenvalues		
4	0	10	3×10^{4}	0.8681	0.1645	0.1268
0	0	10	3×10^{0}	0.8695	0.3683	0.2553
-2	0	30	3×10^{-2}	0.9001	0.7401	0.3662
-3	0	30	3×10^{-3}	0.9198	0.7532	0.3677

TABLE 2. AD-HDP algorithm and matrix Q variations

q_i	r_i	N_{RcT}	Q_{Trace}	Eigenvalues		
10	0	2	3×10^{10}	0.8723	0.0000	0.0000
2	0	2	3×10^{2}	0.8723	0.0097	0.0090
0	0	6	3×10^{0}	0.8693	0.3683	0.2553
-2	0	6	3×10^{-2}	0.9012	0.7390	0.3662
-5	0	10	3×10^{-5}	0.9049	0.7497	0.3679
-10	0	8	3×10^{-10}	0.9049	0.7497	0.3679

TABLE 3. Schur algorithm and matrix Q variations

q_i	r_i	Q_{Trace}	Eigenvalues		
10	0	3×10^{10}	0.8674	0.0000	0.0000
4	0	3×10^{4}	0.8674	0.0097	0.0090
2	0	3×10^{2}	0.8674	0.0097	0.0090
1	0	3×10^{1}	0.8693	0.3683	0.2553
-2	0	3×10^{-2}	0.9001	0.7401	0.3662
-3	0	3×10^{-3}	0.9052	0.7501	0.3677
-5	0	3×10^{-5}	0.9058	0.7512	0.3679
-10	0	3×10^{-10}	0.9058	0.7512	0.3679

comparison with the results presented in Tables 1 and 2. The variations in the matrix Q values are given by $q_i = \{10, 4, 2, 1, -2, -3, -5, -10\}$. With respect to matrix Q traces, it is verified that as the trace of matrix Q increases, the closed loop eigenvalues tend to be closer toward the origin of the Z-plane for both HDP and AD-HDP approaches.

It is observed that variations in the matrix Q values, while the matrix R values are kept constant, led to the mapping of real eigenvalues in the stable Z-plane, i.e., eigenvalues that are inside the unitary cycle with radius given by $\|z\| = 1$. These eigenvalues are limited to the real axis Z and the right half-plane. An investigation to map other regions of the stable Z-plane involves the development of biased heuristics for the selection of matrices Q and R, e.g., heuristics that take into account not only diagonal elements of weighting matrices, but also off-diagonal elements with combination of different elements of Q and R.

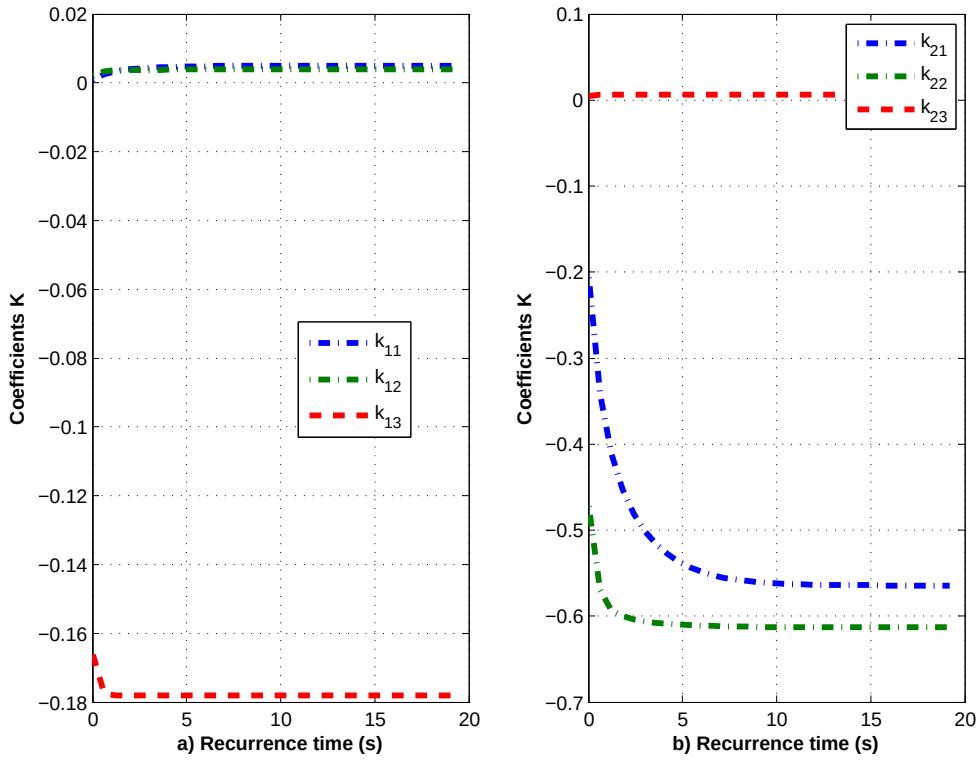

FIGURE 5. HDP gains – Third-order system

The performance of the HDP and AD-HDP methods shows that in terms of the accuracy of steady state estimation of gain values, these do not present significant deviations from the true value, i.e., when the LS estimation is compared with the Schur method. It is noteworthy that a difference between the HDP and AD-HDP approaches is that matrix P is not directly obtained for the AD-HDP approach, because the gain is calculated from the submatrices Q_{uu} and Q_{xu} of the parameterized function Q_L for DLQR.

In terms of transient of the iterative process evolution, the behavior of the HDP gains is presented in Figure 5. This behavior is compared with that of the AD-HDP gains of Figure 2. It is observed that the AD-HDP policy estimation is faster in reaching its optimal values than the HDP policy estimation. In terms of steady state values of decision policies, comparing the steady policy matrices $K_{\infty}^{\theta-HDP}$ of Equation (65), and $K_{\infty}^{\theta-AD}$ of Equation (67), it is observed that steady state values reached for HDP and AD-HDP are the true value of the decision policy matrix, which is given by gain matrix K_{∞}^{schur} of Equation (62) of the Schur method. Consequently, the state and action-dependent approaches presented competitive results when compared with the model-based approach, and besides the online policy is optimal for every instant k of the iterative process.

7. **Conclusion.** In the context of optimal control and reinforcement learning, a proposal of a methodology to establish optimal decision policies that is based on the fusion of QR-tuning method and approximate solution of the Hamilton-Jacobi-Bellman equation was presented. The QR-tuning method performed the selection of the weighting matrices via QR duality principle. These matrices were designed as search guidance for decision policy. The selected pair of matrices, coupled to the HJB equation and its approximate solution via least-squares method, guaranteed optimality of decision rule and furnished the means to evaluate the impact of optimal actions. In general manner, the proposed methodology was shown suitable, based on Bellman optimality principle, to establish policies that must satisfy complex or conflictive design specifications.

Specifically in the DLQR application, the proposed methodology presented skills to impose the dynamic system performance with guaranteed optimality, without considering the full system dynamics model. The state and action-dependent dynamic programming algorithms were developed to compute approximate solutions of the HJB equation and decision policies for online design of DLQR control systems. The computational experiments have shown that the state HDP and AD-HDP approximations presented abilities to carry out eigenvalues mapping in the stable Z-plane for multivariable dynamic systems. Consequently, the two approaches that are based on QR-tuning methods and approximated solutions of the HJB equation are feasible alternatives for online implementation of optimal control systems.

REFERENCES

[1] J. Fu, H. He and X. Zhou, Adaptive learning and control for MIMO system based on adaptive dynamic programming, *IEEE Transactions on Neural Networks*, vol.22, no.7, pp.1133-1148, 2011.

[2] D. Zhao and Z. Hu, Supervised adaptive dynamic programming based adaptive cruise control, *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp.318-323, 2011.

[3] W. Li, G. Xu, Z. Wang and Y. Xu, Dynamic energy management for hybrid electric vehicle based on adaptive dynamic programming, *IEEE International Conference on Industrial Technology*, pp.1-6, 2008.

[4] Y. Tian, D. Zhao and J. Yi, A fuzzy logic controller with adaptive dynamic programming optimization for traffic signals, *The 5th International Conference on Fuzzy Systems and Knowledge Discovery*, vol.3, pp.191-195, 2008.

[5] D. Zhao, J. Yi and D. Liu, Particle swarn optimized adaptive dynamic programming, *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp.32-37, 2007.

[6] Z. Wang, Y. Dai and Y. Yao, Research of a parallel learning adaptive dynamic programming based on genetic algorithms, *The 2nd International Conference on Communication Systems, Networks and Applications*, vol.1, pp.350-353, 2010.

[7] D. Liu, D. Wang and D. Zhao, Adaptive dynamic programming for optimal control of unknown nonlinear discrete-time systems, *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp.242-249, 2011.

[8] Z.-Y. Wang, Y.-P. Dai, Y.-W. Li and Y. Yao, A kind of utility function in adaptive dynamic programming for inverted pendulum control, *International Conference on Machine Learning and Cybernetics*, vol.3, pp.1538-1543, 2010.

[9] W. Ding, D. Liu and Q. Wei, Adaptive dynamic programming for finite-horizon optimal tracking control of a class of nonlinear systems, *The 30th Chinese Control Conference*, pp.2450-2455, 2011.

[10] T. Landelius, *Reinforcement Learning and Distributed Local Model Synthesis*, Ph.D. Thesis, Linköping University, Sweden, 1997.

[11] A. Al-Tamimi, F. L. Lewis and M. Abu-Khalaf, Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.38, no.4, pp.943-949, 2008.

[12] J. V. da Fonseca Neto and L. R. Lopes, On the convergence of DLQR control system design and recurrences of Riccati and Lyapunov in dynamic programming strategies, *UkSim the 13th International Conference on Computer Modelling and Simulation UKSim*, pp.26-31, 2011.

[13] A. Lanzon, Y. Feng, B. D. O. Anderson and M. Rotkowitz, Computing the positive stabilizing solution to algebraic Riccati equations with an indefinite quadratic term via a recursive method, *IEEE Transactions on Automatic Control*, vol.53, no.10, pp.2280-2291, 2008.

[14] X. Zhang, G.-P. Zhong and C. Tan, Existence and representation of stabilizing solutions to generalized algebraic Riccati equations, *Proc. of the 48th IEEE Conference on Decision and Control, Held Jointly with the 28th Chinese Control Conference*, pp.5923-5928, 2009.

[15] Z. Aghaie and R. Amirifar, H_2 and H_∞ controllers design for an active suspension system via Riccati equations and LMIs, *The 2nd International Conference on Innovative Computing, Information and Control*, Kumamoto, Japan, p.341, 2007.

[16] S. Yoon, S. J. Lee, B. Lee, C. J. Kim, Y. J. Lee and S. Sung, Design and flight test of a small trirotor unmanned vehicle with a LQR based onboard attitude control system, *International Journal of Innovative Computing, Information and Control*, vol.9, no.6, pp.2347-2360, 2013.

[17] X. Xin and T. Mita, On the strong solutions of generalized algebraic Riccati equations, *Proc. of the 37th SICE Annual Conference*, pp.791-796, 1998.

[18] P. H. Petkov, N. D. Christov and M. M. Konstantinov, Solution of high order matrix Riccati equations with condition and accuracy estimates, *Proc. of the 1996 IEEE International Symposium on Computer-Aided Control System Design*, pp.93-98, 1996.

[19] A. J. Laub, A schur method for solving algebraic Riccati equations, *IEEE Transactions on Automatic Control*, vol.24, no.6, pp.913-921, 1979.

[20] J. V. da Fonseca Neto, I. S. Abreu and F. N. da Silva, Genetic synthesis for state-space controllers based on linear quadratic regulator design for eigenstructure assignment, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.40, no.2, pp.266-285, 2010.

[21] F. L. Lewis and D. Vrabie, Adaptive dynamic programming for feedback control, *The 7th Asian Control Conference*, pp.1402-1409, 2009.

[22] M. Athans and P. L. Falb, *Optimal Control – An Introduction to the Theory and Its Apllications*, McGraw-Hill Book Company, USA, 1966.

[23] J. Brewer, Kronecker products and matrix calculus in system theory, *IEEE Transactions on Circuits and Systems*, vol.25, no.9, pp.772-781, 1978.

[24] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd Edition, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.

[25] S. N. Balakrishnan, J. Ding and F. L. Lewis, Issues on stability of ADP feedback controllers for dynamical systems, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.38, no.4, pp.913-917, 2008.

[26] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control*, Taylor and Francis, UK, 1975.

[27] B. C. Kuo, *Digital Control Systems*, Oxford University Press, 1992.

[28] R. G. Jacquuot, *Modern Digital Control Systems*, Marcel Dekker Inc., 1994.

[29] F. L. Lewis and D. Vrabie, Reinforcement learning and adaptive dynamic programming for feedback control, *Circuits and Systems Magazine, IEEE*, vol.9, no.3, pp.32-50, 2009.

Appendix A. **Setups of Algorithms.** Relations (49) and (55) establish the necessary information to start-up of Algorithms 1 and 2, respectively. The requested information are the dynamic system model, design parameters, initial conditions and parameters of the iterative process.

A.1. **Dynamic system matrices.** The continuous system matrices are given by

$$A_c = \begin{bmatrix} -1.102 & 0.905 & -0.002 \\ 4.064 & -0.770 & -0.169 \\ 0.000 & 0.000 & -10.000 \end{bmatrix} \tag{69}$$

and

$$B_c = \begin{bmatrix} 0.000 & 0.000 \\ 0.000 & 10.000 \\ 10.000 & 0.000 \end{bmatrix}. \tag{70}$$

For the sampling interval $\Delta t_{samp} = 0.1$, the sampled system matrices are obtained by the zero-order-hold method. These matrices are given by

$$A_d = \begin{bmatrix} 0.912 & 0.083 & -0.001 \\ 0.372 & 0.943 & -0.010 \\ 0.000 & 0.000 & 0.368 \end{bmatrix} \tag{71}$$

and

$$B_d = \begin{bmatrix} -0.000 & 0.043 \\ -0.006 & 0.968 \\ 0.632 & 0.000 \end{bmatrix}. \tag{72}$$

The initial condition for the third-order system is given by

$$x = \begin{bmatrix} 4.000 & 2.000 & 5.000 \end{bmatrix}^T. \tag{73}$$

A.2. **Iterative process.** The setup of the iterative process for HDP is presented. Considering the peculiarities of each algorithm, the procedure for AD-HDP is similar to the HDP. General information on the setup are a) dynamic system of order 3, b) number of iterations is 200, and c) the recurrence factors are 6 for HDP and 15 for AD-HDP.

The initial conditions for the least-squares and the Riccati/Lyapunov equation are presented. According to the vectorizing process of Equation (31), the parameter vector of the HDP is given by

$$\theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 \end{bmatrix}^T. \tag{74}$$

The order of this vector is established from the size of the matrix P. For third-order system, one has that $\theta \in R^6$ and for the presented results, one has that $\theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta_5 = \theta_6 = 0$.

The initial matrix of Riccati/Lyapunov equation is associated with the parameters of the approximation for the third-order system, according to the formation law of symmetric matrix vectorization of Equation (31). The matrix is given by

$$P_0^\theta = \begin{bmatrix} \theta_1 & \theta_2/2 & \theta_3/2 \\ & \theta_4 & \theta_5/2 \\ & & \theta_6 \end{bmatrix}. \tag{75}$$

A.2.1. *Weighting matrices.* The weighting matrices Q and R structures are arrays with non-null diagonal elements, and off-diagonal elements are all equal to zero.