Trends in Computational and Applied Mathematics, **25** (2024), e01686
Sociedade Brasileira de Matemática Aplicada e Computacional
Online version ISSN 2676-0029 www.scielo.br/tcam
ARTIGO ORIGINAL
doi: 10.5540/tcam.2024.025.e01686

# Sintonia *QR* do Regulador Linear Quadrático LQR Discreto e Programação Dinâmica Aproximada baseada em Ação-Estado para Aplicações Online do Projeto de Sistemas de Controle Ótimo

M. CERQUEIRA[1], L. LOPES[2] e J. V. FONSECA[3]

Recebido em 17 de janeiro de 2022 / Aceito em 15 de janeiro de 2024

**RESUMO.** Devido ao crescente desenvolvimento tecnológico e às consequentes aplicações industriais, novos métodos para o projeto de controle e Aprendizado por Reforço tem sido desenvolvidas, não apenas para resolver novos problemas de controle, mas também para melhorar o desempenho de controladores já implementados em sistemas do mundo real. As abordagens de Aprendizagem por Reforço e Regulador Quadrático Linear Discreto são conectadas por métodos de Programação Dinâmica Adaptativa. Esses paradigmas são orientados para o projeto de controladores ótimos em sistemas multivariáveis. Para o caso do Regulador Quadrático Linear Discreto, AD-HDP, Aprendizado por Reforço, Política de Iteração e Valor de Iteração, um método e um algoritmo são desenvolvidos e implementados para projeto de controle online. Com base na seleção das matrizes de ponderação Q e R, também é apresentado um método para ajustar controladores reguladores lineares quadráticos discretos. Este método fornece diretrizes para a construção de heurísticas para a seleção de matrizes de ponderação, aspectos de convergência relacionados às variações das matrizes de ponderação são investigados. Para um sistema dinâmico multivariável de terceira ordem, o algoritmo proposto e a heurística de ajuste são avaliados pela capacidade de estabelecer a política de controle ótima.

**Palavras-chave:** programação dinâmica, controle ótimo, *Q-Function*, AD-HDP, sistemas multivariáveis, convergência, DLQR.

## 1 DESCRIÇÃO E FORMULAÇÃO DO PROBLEMA

As matrizes de ponderação do índice de desempenho são escolhidas pela característica de controle. A dificuldade de escolha se deve ao fato de não existir um método sistemático para tal seleção, sendo normalmente adotada a forma diagonal para essas matrizes e tendo sua escolha realizada por meio de várias simulações, tentativa e erro. Os valores selecionados para as matrizes

[1]Universidade Federal do Maranhão, PPGEE - Dep. Eng. de Computação – E-mail: cerqueira.marcio@ufma.br https://orcid.org/0000-0001-7353-0326
[2]Universidade Federal do Maranhão, PPGEE – E-mail: leandror4@gmail.com https://orcid.org/0009-0000-1154-2137
[3]Universidade Federal do Maranhão, PPGEE - Dep. Eng. Elétrica – E-mail: viana.fonseca@ufma.br http://orcid.org/0000-0003-4606-7510

de ponderação, devem satisfazer os critérios estabelecidos pelo projetista, que por consequência, influenciam na determinação do ganho do controlador.

### 1.1 Seleção das matrizes de ponderações *QR* do índice de desempenho

Algumas metodologias podem ser citadas. No trabalho [18], as técnicas disponíveis para seleção destas matrizes, são divididas em quatro metodologias: métodos heurísticos, controle ótimo modal, método do lugar das raízes ótimo assintótico e método da ponderação dinâmica. Em [38] um novo algoritmo de ADP é desenvolvido para resolver problemas de controle ótimo para sistemas não lineares de tempo discreto de horizonte infinito.

A primeira técnica projetada para seleção das matrizes de ponderação do custo funcional foram as heurísticas. O método dos mínimos quadrados é, ainda, usado como um primeiro teste exploratório por diversos projetistas como metodologia de projeto de controle ótimo [5].

Para tal, aplica-se o **Método de *Bryson***. A ideia básica é normalizar as contribuições que as saídas e o termo de controle devem ter dentro da função de índice de desempenho quadrático. Esta normalização é realizada utilizando o máximo valor de desvio das saídas e controles individuais [18].

### 1.2 O Método de Sintonia Heurística *QR*

Os autovalores são utilizados para verificar se a especificações de projeto estão sendo contempladas durante a operação do sistema. Estabelece-se uma relação entre as figuras de mérito do sistema dinâmico em função das matrizes de ponderação. A nova lei de controle em função das matrizes de ponderação são dadas por

$$u_k(QR) = -K_{QR}x_k, \tag{1.1}$$

sendo $K_{QR}$ o ganho do controlador que depende diretamente da seleção das matrizes de ponderação.

O método de Sintonia Heurística *QR* (MSH-*QR*) [10] se baseia nas relações entre as matrizes $Q$ e $R$ quando a recorrência de *Riccati* que é dada por

$$P = Q + A_d^T PA - A_d^T PB_d K_{ric}, \tag{1.2}$$

então, substituindo a Equação (1.2) na equação do ganho a seguir

$$K_{ric} = (R + B_d^T PB_d)^{-1} B_d^T PA_d, \tag{1.3}$$

tem-se que o ganho ótimo da Equação (1.3) é determinado por

$$K_{ric} = (R + B_d^T PB_d)^{-1} B_d^T (Q + A_d^T PA_d - A_d^T PB_d K_{ric}) A_d. \tag{1.4}$$

Com a Equação (1.3) no intuito de explicitar as relações para $Q$ e $R$ chega-se a

$$\begin{aligned} K_{ric}(Q,R) = &\left\{ \left[ (R + B_d^T PB_d)^{-1} B_d^T (Q + A_d^T PA_d) \right] \right. \\ &\left. - \left[ (R + B_d^T PB_d)^{-1} B_d^T (A_d^T PB_d K_{ric}) \right] \right\} A_d. \end{aligned} \tag{1.5}$$

A forma explícita final é dada por

$$K_{ric}(Q,R) \quad = \quad K_{f1}(QR) + K_{f2}(R), \tag{1.6}$$

sendo

$$\begin{aligned} K_{f1}(Q,R) &= (R + B_d^T P B_d)^{-1} B_d^T (Q + A_d^T P A_d) A_d. \\ K_{f2}(R) &= -(R + B_d^T P B_d)^{-1} B_d^T (A_d^T P B_d K_{ric}) A_d. \end{aligned}$$

Considerando as situações $R >> B_d^T P B_d$ e $Q >> A_d^T P A_d$ em que as formas quadráticas da entrada e do estado são bem menores que as ponderações $Q$ e $R$, tem-se que

$$K_{f1}(Q,R) \approx \left[R^{-1} B_d^T Q\right] A_d \tag{1.7}$$

$$K_{f2}(R) \approx -\left[R^{-1} B_d^T A_d^T P B_d R^{-1} B_d^T P A_d\right] A_d. \tag{1.8}$$

Observa-se que, se $R >> 0$ para o caso de matrizes diagonais, tem-se que $f_2(QR) \approx 0$. A demonstração mais detalhada do cálculo dos ganhos pode ser vista em [10].

## 2 AD-HDP PARA PROJETO *ON-LINE* DO REGULADOR LINEAR QUADRÁTICO

### 2.1 Fundamento do Controle ótimo para tempo discreto

A maioria dos estudos em ADP tem sido realizada para sistemas que operam em tempo discreto. Dessa forma, considere o seguinte sistema discreto

$$x_{k+1} = f(x_k) + g(x_k) u_k. \tag{2.1}$$

Partindo-se da Equação (2.1), defini-se uma função de custo, para horizonte infinito, como

$$V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) = \sum_{i=0}^{\infty} \gamma^i r(x_{k+i}, u_{k+i}). \tag{2.2}$$

A DP como solução do controle ótimo, é um procedimento de retrocesso no tempo (*backwards-in-time*), sendo utilizado para planejamento *offline*, por isso tem-se que

$$V_h(x_k) = r(x_k, u_k) + \gamma V_h(x_{k+1}), \quad V_h(0) = 0. \tag{2.3}$$

Com $0 < \gamma \leq 1$ um fator de desconto e $u_k = h(x_k)$, sendo política de controle de realimentação. A Função $r(x_k, u_k)$ é conhecida como fator de utilidade (ou reforço em AR), sendo a medida de um passo da função de custo. Ela pode ser selecionada baseada em algumas considerações como energia mínima, risco mínimo, etc. Uma forma padrão utilizada na área de controle é a função quadrática $r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k$.

Assume-se que um sistema é estabilizável em um conjunto $\Omega \in R^n$, se existir uma política de controle $u_k = h(x_k)$ em que o sistema da Equação (2.1) em malha fechada seja assintoticamente

estável em $\Omega$. A política de controle $u_k = h(x_k)$ é dita admissível se garante estabilidade e produz um custo finito $V_h(x_k)$ [26].

O objetivo do controle ótimo é selecionar a política que minimiza

$$V^*(x_k) = \min_{h(.)}(\sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h(x_i))) \tag{2.4}$$

A política ótima de controle é dada por

$$u_k^* = h^*(x_k) = \arg\min_{h(.)}(\sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h(x_i))) \tag{2.5}$$

Percebe-se que no campo da inteligência computacional, a Equação (2.2) é interpretada como reforço e seu objetivo é maximizá-la. A maneira de como o valor de custo é encontrado é a principal diferença entre controle com realimentação e Aprendizagem por Reforço.

A Equação (2.3) de *Bellman*, leva a dois métodos iterativos (Política de Iteração - PI e Valor de Iteração - VI) para aprender a solução do controle ótimo sem ter que resolver a equação de *Hamilton-Jacobi-Bellman*.

## 2.2 ADP- Temporal Diferencial (TD) e Aproximação da Função Valor

O conceito chave para a implementação de controladores ótimos *online* em avanço no tempo é a diferença temporal do erro, que é definido em termos da equação de *Bellman*, seja

$$e_k = H(x_k, h(x_k), \Delta V_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k). \tag{2.6}$$

Percebe-se que o lado direito nada mais é que o Hamiltoniano. A função $e_k$ é conhecida como TD do erro. A intenção é levar a solução do Hamiltoniano de tal forma que $e_k = 0$ a cada intervalo $k$ para função valor $V_h(\cdot)$, logo temos que

$$0 = H(x_k, h(x_k), \Delta V_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k). \tag{2.7}$$

A Equação (2.7) é um elemento importante na resolução *online* da equação não linear de *Lyapunov* utilizando-se somente dados medidos ao longo da trajetória do sistema. Para sistemas não lineares, TD é de difícil solução.

Uma Aproximação da Função Valor (*Value Function Approximation* - VFA) para o caso do DLQR pode ser considerado. Sabe-se que $u_k = -Kx_k$ é quadrática nos estados, ou seja, vale para alguma matriz $P$. Tem-se para o DLQR o TD de erro da seguinte forma

$$e_k = x_k^T Q x_k + u_k^T R u_k + \gamma(x_{k+1}^T P x_{k+1}) - x_k^T P x_k. \tag{2.8}$$

Para simplificar-se a Equação (2.8), utiliza-se o produto de *Kronecker* para chegar a

$$V_K(x_k) = x_k^T P x_k = vec(x_k^T P x_k) = (x_k \otimes x_k)(vec(P))^T \equiv \bar{p}^T \bar{x}_k, \tag{2.9}$$

sendo $\otimes$ o produto de *Kronecker* [9] e $vec(P)$ o vetor formado pelos elementos da matriz $P$ em um vetor coluna.

O erro TD, passa ser visto da seguinte forma

$$\begin{aligned} e_k &= x_k^T Q x_k + u_k^T R u_k + \gamma(\bar{p}^T \bar{x}_{k+1}) - \bar{p}^T \bar{x}_k \\ e_k &= r(x_k, u_k) + \gamma(\bar{p}^T \bar{x}_{k+1}) - \bar{p}^T \bar{x}_k. \end{aligned} \tag{2.10}$$

Para o DLQR, um conjunto base completo para Função Valor $V_h(x_k)$ é fornecida pela função quadrática dos componentes de $x_k$.

### 2.3 ADP-AR *online* para controle ótimo

Uma aproximação da função valor $V_h(x)$ do DLQR pode ser dada por

$$V_h(x) = W_{V_h}^T \phi(x) + \varepsilon_L(x), \tag{2.11}$$

sendo $W_{V_h}^T$ função de ponderação da matriz de entrada, $\phi(x)$ é a matriz de regressores e $\varepsilon_L(x)$ é o TD do erro.

Tendo como vetor base $\phi(x) = \begin{bmatrix} \varphi_1(x) & \varphi_2(x) & \dots & \varphi_L(x) \end{bmatrix} : R^n \to R^L$ e $\varepsilon_L(x)$ converge para zero à medida que $L \to \infty$.

Assumindo-se agora, a aproximação da forma a seguir

$$V_h(x) = W_{V_h}^T \phi(x). \tag{2.12}$$

Substituindo-se a aproximação da Equação (2.12) na Equação (2.6) chega-se em

$$e_k = r(x_k, u_k) + \gamma W_{V_h}^T \phi(x_{k+1}) - W_{V_h}^T \phi(x_k). \tag{2.13}$$

Procedimentos iterativos para resolver a equação TD podem ser utilizados, incluindo a Política de Iteração e Valor de iteração [6], [8] e [39].

### 2.4 Algoritmo AD-HDP para o DLQR

O princípio da otimalidade de *Bellman* é formulado usando o conceito de Função-Q (*Q-function*) [8] e [39].

A equação de *Bellman* nos permite calcular o valor usando qualquer política de controle admissível. Entretanto, pode-se definir a Função-Q associada com a política $u = h(x)$ da seguinte forma

$$Q_h(x_k, u_k) = r(x_k, u_k) + \gamma V_h(x_{k+1}). \tag{2.14}$$

Percebe-se que a Função-Q, fica em função do estado $x_k$ e do controle $u_k$ no tempo $k$. A Função-Q ótima é dada por

$$Q^*(x_k, u_k) = r(x_k, u_k) + \gamma V^*(x_{k+1}). \tag{2.15}$$

Em termos de $Q^*$, pode-se escrever a equação de otimalidade de *Bellman* por

$$V^*(x_k) = \min_{u_k}(Q^*(x_k,u_k)) \tag{2.16}$$

e o controle ótimo da forma

$$h^*(x_k) = \arg\min_{u_k}(Q^*(x_k,u_k)). \tag{2.17}$$

O valor mínimo então pode ser obtido pela Equação (2.14), logo

$$\frac{\partial Q^*(x_k,u_k)}{\partial u_k} = 0. \tag{2.18}$$

Para determinar a equação de ponto fixo para a Função-Q, tem-se

$$Q_h(x_k,h(x_k)) = V_h(x_k), \tag{2.19}$$

então, a "equação de *Bellman*" para o $Q$ é dada por

$$Q_h(x_k,u_k) = r(x_k,u_k) + \gamma Q_h(x_{k+1},h(x_{k+1})) \tag{2.20}$$

e o valor $Q$ ótimo é

$$Q^*(x_k,u_k) = r(x_k,u_k) + \gamma Q^*(x_{k+1},h^*(x_{k+1})). \tag{2.21}$$

A Equação (2.20) é a equação de ponto fixo ou "Equação de *Bellman*" para a Função-Q. A partir dela pode-se aplicar qualquer técnica de AR para solução, incluindo PI e VI [6].

### 2.5 Caracterização da configuração Função-Q para o DLQR

A partir da formulação já proposta usando Função-Q, agora leva-se a uma aplicação para o caso do DLQR. Partindo-se da Equação (2.14) tem-se

$$Q_K(x_k,u_k) = x_k^T Q x_k + u_k^T R u_k + \gamma[x_{k+1}^T P x_{k+1}] = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \tag{2.22}$$

sendo $P$ a solução da equação de *Lyapunov* para um determinado ganho de controle $K$ e uma matriz $H$ associada a solução de *Lyapunov*. Desenvolvendo-se a Equação (2.22) chega-se em

$$Q_K(x_k,u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q+\gamma A_d^T P A_d & \gamma A_d^T P B_d \\ \gamma B_d^T P A_d & R+\gamma B_d^T P B_d \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \tag{2.23}$$

A Equação (2.23) é a Função-Q, quadrática em $(x_k,u_k)$, para o DLQR. A matriz $H$ então é dada por

$$H = \begin{bmatrix} Q+\gamma A_d^T P A_d & \gamma A_d^T P B_d \\ \gamma B_d^T P A_d & R+\gamma B_d^T P B_d \end{bmatrix}. \tag{2.24}$$

Com a Equação (2.21), para o caso do DLQR, pode-se obter

$$\begin{aligned}\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} &= r(x_k,u_k)+\gamma Q^*(x_{k+1},u_{k+1}) \\ &= x_k^T Q x_k + u_k^T R u_k + \gamma \begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix}^T H \begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix}. \end{aligned} \tag{2.25}$$

Atribuindo-se $h(x_k) = u_k = -Kx_k$ e desenvolvendo a Equação (2.25) então $H$ pode ser escrito da seguinte forma [17]

$$\begin{aligned}\begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} &= G+\gamma \begin{bmatrix} A_d & B_d \\ -KA_d & -KB_d \end{bmatrix}^T H \begin{bmatrix} A_d & B_d \\ -KA_d & -KB_d \end{bmatrix} \\ &= G+\gamma \begin{bmatrix} A_d^T \\ B_d^T \end{bmatrix} \begin{bmatrix} I & -K^T \end{bmatrix} H \begin{bmatrix} I \\ -K \end{bmatrix} \begin{bmatrix} A_d & B_d \end{bmatrix}. \end{aligned} \tag{2.26}$$

A Função-Q ótima é igual a Função Valor $V^*(x_k)$, quando a política $u_k$ é ótima, logo

$$\begin{aligned} V^*(x_k) &= \min_{u_k} Q^*(x_k,u_k) \\ &= \begin{bmatrix} x_k^T & u_k^T \end{bmatrix} H \begin{bmatrix} x_k^T & u_k^T \end{bmatrix}^T. \end{aligned} \tag{2.27}$$

Para se minimizar em função de $u_k$, aplica-se a Equação (2.18) e obtém-se

$$\begin{aligned} 0 &= H_{ux}x_k + H_{uu}u_k \\ u_k &= (H_{uu})^{-1}H_{ux}x_k. \end{aligned} \tag{2.28}$$

Como $u_k = -Kx_k$, chega-se em

$$K = (H_{uu})^{-1}H_{ux}. \tag{2.29}$$

A Equação (2.23) pode ser utilizada para se obter o ganho da Equação (2.29) em função da matriz $P$. Nota-se que o ganho $K$ do controlador fica dependente somente da matriz $H$, não necessitando do conhecimento das matrizes $A_d$ e $B_d$.

## 2.6 Formulação *online* do algoritmo ADP-ADHDP-DLQR

O algoritmo *Q-Learning* tem sido aplicado para resolver o problema do DLQR [23]. No *Q-Learning*, uma estrutura paramétrica é usada para aproximar a Função-Q da política de controle atual [2].

Seja a seguinte equação ajustada de Bellman para o DLQR

$$\begin{aligned} Q_{j+1}(x_k,u_k) &= x_k^T Q x_k + u_k^T R u_k + \min_{u_k} \gamma Q_j(x_{k+1},u_{k+1}) \\ &= x_k^T Q x_k + u_k^T R u_k + V_j(x_{k+1}) \\ &= x_k^T Q x_k + u_k^T R u_k + \gamma V_j(A_d x_k + B_d u_k). \end{aligned} \tag{2.30}$$

No *Q-Learning*, inicia-se com uma Função-Q inicial $Q_0(x,u) \geq 0$ não necessariamente ótima e depois encontra-se $Q_1(x,u)$ resolvendo-se a Equação (2.30) com $i = 0$ [2]. Consequentemente tem-se que

$$\min_{u_k} Q_{j+1}(x_k,u_k) = \min_{u_k} \begin{bmatrix} x_k^T & u_k^T \end{bmatrix} H_{j+1} \begin{bmatrix} x_k^T & u_k^T \end{bmatrix}^T. \tag{2.31}$$

De acordo com a Equação (2.29), a política de atualização de realimentação é dada por

$$K_j = (H_{uu}^j)^{-1} H_{ux}^j, \tag{2.32}$$

sendo

$$h_j(x_k) = -K_j x_k. \tag{2.33}$$

Este é o método de política de iteração gulosa (*greedy*) que é baseado na Função-Q [2], então

$$\begin{aligned} \min_{u_k} Q_{j+1}(x_k,u_k) &= V_{j+1}(x_k) \\ &= \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k + \gamma V_j(A_d x_k + B_d u_k) \right\}. \end{aligned} \tag{2.34}$$

Para obter-se a solução em avanço no tempo, deve-se substituir a Equação (2.33) na Equação (2.30), obtendo-se a seguinte relação de recorrência

$$\begin{aligned} Q_{j+1}(x_k,u_j(x_k)) &= x_k^T Q x_k + h_j^T(x_k) R h_j(x_k) \\ &+ \gamma \begin{bmatrix} x_{k+1}^T & h_j^T(x_{k+1}) \end{bmatrix} H_j \begin{bmatrix} x_{k+1}^T & h_j^T(x_{k+1}) \end{bmatrix}^T \end{aligned} \tag{2.35}$$

O objetivo é resolver $Q_{j+1}$ para $j = 0,1,2,....$ Quando $i \to \infty$, então tem-se $Q_{j+1}(x_k,u_j(x_k)) \to Q^*(x_k,u_k)$, implica que $H_j \to H$ e $K_j \to K$.

Em um sistema realimentado, na aproximação usando AD-HDP, a Função-Q é geralmente difícil de se obter. Entretanto, uma estrutura paramétrica pode ser utilizada para se aproximar $Q_j(x,u)$ [2]. Similarmente, uma estrutura paramétrica é utilizada para aproximar a ação de controle $\hat{u}_j(x,K)$ [3], então

$$\hat{h}_j(x) = -K_j x \tag{2.36}$$

$$\widehat{Q}(\bar{z},\bar{H}_j) = z^T H_j z = \bar{H}_j^T \bar{z}_k, \tag{2.37}$$

sendo $z = \begin{bmatrix} x^T & u^T \end{bmatrix}^T \; z \in R^{n+m=q}$, $\bar{z} = (z_1^2,...,z_1 z_q, z_2^2, z_2 z_3,...,z_{q-1} z_q, z_q^2)$ é o vetor base polinomial do produto de *Kronecker*, e $\bar{H} = vec(H)$ com $vec(\cdot)$ sendo o vetor função que atua sobre

a matriz $q \times q$ e dá como saída o vetor coluna $\frac{q(q+1)}{2} \times 1$. A saída de $vec(\cdot)$ é construída com o empilhamento das colunas da matriz quadrada em um único vetor coluna com os elementos fora da diagonal somados como $H_{ij} + H_{ji}$. As estruturas paramétricas (2.36) e (2.37) fornecem uma representação da Equação (2.35).

A aproximação da Equação (2.35), utilizando-se as estruturas paramétricas propostas é dada por

$$d(z_k(x_k), H_j) = x_k^T Q x_k + \hat{h}_j^T(x_k) R \hat{h}_j(x_k) + \gamma Q_j(x_{k+1}, \hat{h}_j(x_{k+1})), \tag{2.38}$$

que é a função de objetivo a ser alcançada a partir da estimação de $\widehat{Q}(z, \bar{H}_{j+1})$ por LS para se encontrar $\bar{H}_{j+1}$, logo

$$\bar{H}_{j+1}^T \bar{z}(x_k) = d(\bar{z}(x_k), \bar{H}_j). \tag{2.39}$$

O parâmetro $\bar{H}_{j+1}$ é encontrado pela minimização do erro entre as Equações (2.37) e (2.38) através do LS em um dado conjunto compacto $\Omega$, assim

$$\bar{H}_{j+1} = \arg\min_{\bar{H}_{j+1}} \left\{ \int_{\Omega} \left| \bar{H}_{j+1}^T \bar{z}(x_k) - d(\bar{z}(x_k), \bar{H}_j) \right|^2 dx_k \right\}. \tag{2.40}$$

Resolvendo-se o problema do LS, obtém-se

$$\bar{H}_{j+1} = \left( \int_{\Omega} \bar{z}(x_k) \bar{z}(x_k)^T dz \right)^{-1} \int_{\Omega} \bar{z}(x_k) d(\bar{z}(x_k), \bar{H}_j) dx, \tag{2.41}$$

entretanto sabe-se que $z(x_k)$ é dado por

$$\begin{aligned} z(x_k) &= \begin{bmatrix} x_k^T & (\hat{h}_j(x_k)) \end{bmatrix}^T & (2.42) \\ &= \begin{bmatrix} x_k^T & (-K_j x_k) \end{bmatrix}^T & (2.43) \\ &= \left( x_k^T \begin{bmatrix} I & -K_j^T \end{bmatrix}^T \right)^T, & (2.44) \end{aligned}$$

com da Equação (2.44) pode-se notar a dependência linear de $x_k$ em $\hat{h}_j$, portanto, tem-se que

$$\int_{\Omega} \bar{z}(x_k) \bar{z}(x_k)^T dx_k, \tag{2.45}$$

não possui inversa isso torna o problema sem solução por LS. Para contornar este problema adiciona-se um ruído branco a entrada, então,

$$\hat{h}_{ej}(x_k) = -K_j x_k + n_k, \tag{2.46}$$

sendo $n(0, \sigma)$ com variância $\sigma^2$, portanto $z(x_k)$ torna-se

$$z(x_k) = \begin{bmatrix} x_k \\ \hat{h}_{ej}(x_k) \end{bmatrix} = \begin{bmatrix} x_k \\ -K_j x_k + n_k \end{bmatrix} = \begin{bmatrix} x_k \\ -K_j x_k \end{bmatrix} + \begin{bmatrix} 0 \\ n_k \end{bmatrix}. \tag{2.47}$$

Modificando-se a Equação (2.38) tem-se

$$d(z_k(x_k),H_j) = x_k^T Q x_k + \hat{h}_{ej}(x_k)^T R \hat{h}_{ej}(x_k) + \gamma Q_j(x_{k+1},\hat{h}_j(x_{k+1})), \tag{2.48}$$

e a inversa da matriz dada por

$$\bar{H}_{j+1} = (ZZ^T)^{-1}ZY \tag{2.49}$$

é garantida pela condição de excitação. Então tem-se

$$\begin{aligned} d(z_k(x_k),H_j) &= x_k^T Q x_k + \hat{h}_{ej}(x_k)^T R \hat{h}_{ej}(x_k) \\ &+ \gamma \begin{bmatrix} x_{k+1}^T & (-K_j x_{k+1})^T \end{bmatrix} H_j \begin{bmatrix} x_{k+1}^T & (-K_j x_{k+1})^T \end{bmatrix}^T, \end{aligned} \tag{2.50}$$

sendo

$$x_{k+1} = A_d x_k + B_d \hat{h}_{ej}(x_k). \tag{2.51}$$

A solução por LS, Equação (2.49), pode ser resolvida em tempo real através de medições suficientes geradas por $d(z_k,H_j)$ na Equação (2.48). Necessita-se do conhecimento dos estados $x_k$, $x_{k+1}$, da função de reforço $r(z_k) = x_k^T Q x_k + \hat{h}_{ej}(x_k)^T R$, $\hat{h}_{ej}(x_k)$ e $Q_j$. Portando, o algoritmo *Q-Learning*, não necessita do modelo do sistema para atualização do Crítico ou da Ação de Controle, sendo assim, um modelo de ajuste livre (*Model-Free Tunning*) [2].

Para satisfazer a condição de excitação do problema de LS, precisa-se ter o número de medições $N \geq {}^{q(q+1)}/_{2}$ , sendo $q = n+m$ o número de estados e da política de controle, respectivamente. Na implementação *online* do LS, as matrizes $Y$ e $Z$ são obtidas em tempo real como

$$\begin{aligned} Z &= \begin{bmatrix} \bar{z}(x_{k-N-1}) & \bar{z}(x_{k-N-2}) & \ldots & \bar{z}(x_{k-1}) \end{bmatrix} \\ Y &= \begin{bmatrix} d(\bar{z}(x_{k-N-1}),\bar{H}_j) & d(\bar{z}(x_{k-N-2}),\bar{H}_j) & \ldots & d(\bar{z}(x_{k-1}),\bar{H}_j) \end{bmatrix}^T. \end{aligned} \tag{2.52}$$

No desenvolvimento do algoritmo AD-HDP, o parâmetro da ação de controle é atualizado de acordo com a Equação (2.32). A seguir é apresentado o Algoritmo 1 conforme foi formulado nas seções anteriores.

**Algoritmo 1** *ADP – AD-HDP – DLQR*

$\triangleright$ **Inicialização**
$\bar{H}_0 = vec(H_0) \geq 0$; $P_0 \geq 0$; $x_0 \leftarrow [\quad\quad]$; $j = 0$; $K_0 = 0$
Ponderações e Sistema Dinâmico
$[Q, R, A_d, B_d]$
Selecionar o Fator de Desconto:
$0 < \gamma \leq 1$
$\triangleright$ **Número de Medições**
$N \geq q(q+1)/2$ , sendo $q = n+m$
$\triangleright$ **Processo Iterativo**
**for** $j \rightarrow j+1$
    **do**
        $\triangleright$ **Implementação do Ruído Branco**
        $n_j \leftarrow [\quad\quad]$
        $\triangleright$ **Ação de Controle**
        $\hat{h}_{ej}(x_j) = -K_j x_j + n_j$
        $\triangleright$ **Sistema Dinâmico**
        $x_{j+1} = A_d x_j + B_d \hat{h}_{ej}(x_j)$
        $\triangleright$ **Resolução através do LS Batelada**:
        $Z = \left[ \begin{array}{cccc} \bar{z}(x_{k-N-1}) & \bar{z}(x_{k-N-2}) & \ldots & \bar{z}(x_{k-1}) \end{array} \right]$
        $Y = \left[ \begin{array}{cccc} d(\bar{z}(x_{k-N-1}), \bar{H}_j) & d(\bar{z}(x_{k-N-2}), \bar{H}_j) & \ldots & d(\bar{z}(x_{k-1}), \bar{H}_j) \end{array} \right]^T$
        **if** $j == N$
          **then**
              $\triangleright$ **Montagem da Matriz** $H$
              $\bar{H}_{j+1} = (ZZ^T)^{-1} ZY$
              $H_{j+1} = f(\bar{H}_{j+1})$
              $\triangleright$ **Ganho Ótimo de Realimentação** $K$
              $K_{j+1} = (H_{uu}^{j+1})^{-1} H_{ux}^{j+1}$
        **if** $\left\| \bar{H}_{j+1} - \bar{H}_j \right\|_F < \varepsilon$
          **then**
**Fim do Processo Iterativo**

### 2.7 Influência do ruído de controle e fator de desconto

Para analisar a influência do ruído na entrada de controle, considera-se a Equação (2.3) de *Bellman*, com a entrada de controle dada por $\hat{h}_e(x_k) = u_k + n_k$, sendo $n_k$ o ruído branco adicionado [24] e [27]. Seja o valor ótimo dado por $V_{h_e}(x_k) = V_k(x_k) = x_k^T P x_k$ para uma matriz P

Hermitiana ou real simétrica definida positiva para todos os estados $x_k$ e realizando todos os desdobramentos algébricos chega-se em

$$V_{h_e}(x_k) = \sum_{i=0}^{\infty} \gamma^i (x_{k+i}^T Q x_{k+i} + n_{k+i}^T R n_{k+i}). \tag{2.53}$$

Portanto, observa-se claramente que a escolha do fator de desconto $\gamma$, além de exercer esta forte influência na convergência do algoritmo, o ruído $n_k$ adicionado na ação de controle do algoritmo AD-HDP, pode ser minimizado pela escolha do fator de desconto $\gamma$.

## 3 AVALIAÇÃO DO ALGORITMO AD-HDP

A avaliação do desempenho dos algoritmos propostos no decorrer deste trabalho, são realizados por meio de procedimentos que metrificam tempo e exatidão para obtenção da política ótima de controle. Neste capítulo mostra-se o modelo do sistema dinâmico utilizado, as metodologia de convergência das matrizes de ponderação $Q$ e $R$ e as análises dos resultados obtidos por meio computacional utilizando-se como plataforma o MATLAB®.

O sistema de avião F-16, extraído do livro [25] é utilizado como sistema base para avaliação do projeto de controlador ótimo abordado. O modelo contínuo é dado por

$$\dot{x}(t) = \begin{bmatrix} -1.10188 & 0.90528 & -0.00212 \\ 4.0639 & -0.7013 & -0.16919 \\ 0 & 0 & -10 \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0 \\ 0 & 10 \\ 10 & 0 \end{bmatrix} u(t) \tag{3.1}$$

Os estados são $x = \begin{bmatrix} \alpha & q & \delta_e \end{bmatrix}^T$, sendo $\alpha$ o ângulo de ataque, $q$ a taxa de *pitch* e $\delta_e$ ângulo de deflexão elevador.

A discretização do modelo é obtida pelo método padrão do segurador de ordem zero (*zero order hold* - zoh) com intervalo de amostra de $0,1s$. O modelo discreto fica definido por

$$x_{k+1} = \begin{bmatrix} 0.9124 & 0.0829 & -0.0007 \\ 0.3724 & 0.9428 & -0.0103 \\ 0 & 0 & 0.3679 \end{bmatrix} x_k + \begin{bmatrix} -0.0003 & 0.0427 \\ -0.0061 & 0.9682 \\ 0.6321 & 0 \end{bmatrix} u_k, \tag{3.2}$$

sendo que o modelo da Equação (3.2) será utilizado para o projeto do controlador ótimo.

### 3.1 Convergência *QR* e Resultados de AD-HDP para o DLQR

Aspectos de estabilidade na implementação de uma ADP num sistema realimentado tem sido estudado durante os anos [4] [15] [35] [41] [34]. A análise de convergência *QR* consiste na avaliação dos autovalores das matrizes $Q$ e $R$ e suas relações com a alocação de autovalores de sistemas MIMO no plano $Z$ por meio dos controladores ótimos. Os resultados são apresentados na forma de tabela seguindo uma heurística que é estabelecida a partir das Equações (1.7) e (1.8) [10]. O processo iterativo para variações sistemáticas nas matrizes $Q$ e $R$ da função de custo

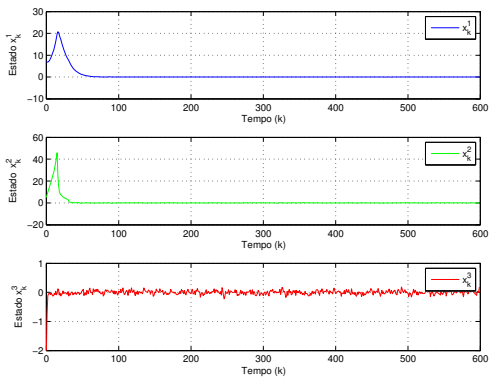
seguem um padrão de crescimento da matriz $Q$, enquanto que a matriz $R$ é uma matriz identidade durante todo o processo de solução.

Serão analisados os resultados obtidos por AD-HDP, para o algoritmo 01, AD-HDP para o DLQR, do sistema MIMO de terceira ordem. Análises sobre os aspectos de convergência do algoritmo em relação ao número de iterações e a relação dos valores das matrizes $Q$ e $R$ com alocação de autovalores no plano $Z$ serão feitas.

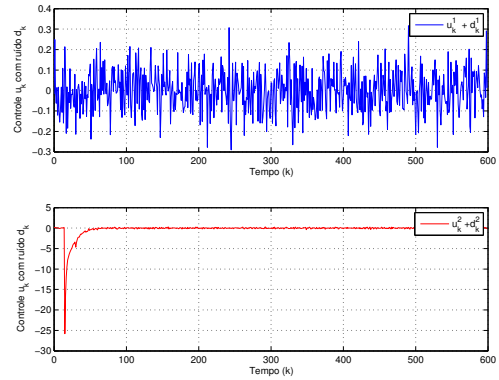
Na implementação *online* do algoritmo AD-HDP para sistemas MIMO o algoritmo (*Q-Learning*) é utilizado para controle *online* do avião F-16 em avanço no tempo.

No projeto de AD-HDP, os estados do avião são inicializados com $x_0 = [\ 7 \quad 5 \quad -2\ ]$. Qualquer valor pode ser selecionado. As matrizes de ponderação, $Q$ e $R$ são inicializadas como identidade com suas respectivas dimensões. É selecionado o fator de desconto, $\gamma = 1$. Os parâmetros do crítico e do ator são inicializados com zero. Após esta etapa de inicialização, a dinâmica do avião é executada em avanço no tempo e o ajuste das estruturas parâmetros é realizada por meio da observação dos estados *online*.

Na Figura 1, tem-se os estados e as entradas de controle em relação ao tempo.




(a) $x_k$




(b) $u_k$

Figura 1: Trajetória dos Estados e Ação de Controle por HDP.

Para manter a condição de excitação, utilizou-se a injeção de um ruído no controle, que pode ser visto na Figura 2. Assim, tem-se a condição de excitação persistente necessária para convergência do LS se evitando desvio de parâmetros. O ruído de controle está associado a convergência de $P$ como demonstrado anteriormente.

A matriz $H_j$ da Equação (2.25), é encontrada de forma *online* através do algoritmo proposto e é dada por

$$H_{AD-HDP} = \begin{bmatrix} 6.2200 & 1.6388 & -0.0202 & -0.0153 & 1.4620 \\ 1.6388 & 2.5934 & -0.0197 & -0.0144 & 1.5876 \\ -0.0202 & -0.0197 & 1.1496 & 0.2568 & -0.0197 \\ -0.0153 & -0.0144 & 0.2568 & 1.4411 & -0.0143 \\ 1.4620 & 1.5876 & -0.0197 & -0.0143 & 2.5901 \end{bmatrix}. \tag{3.3}$$

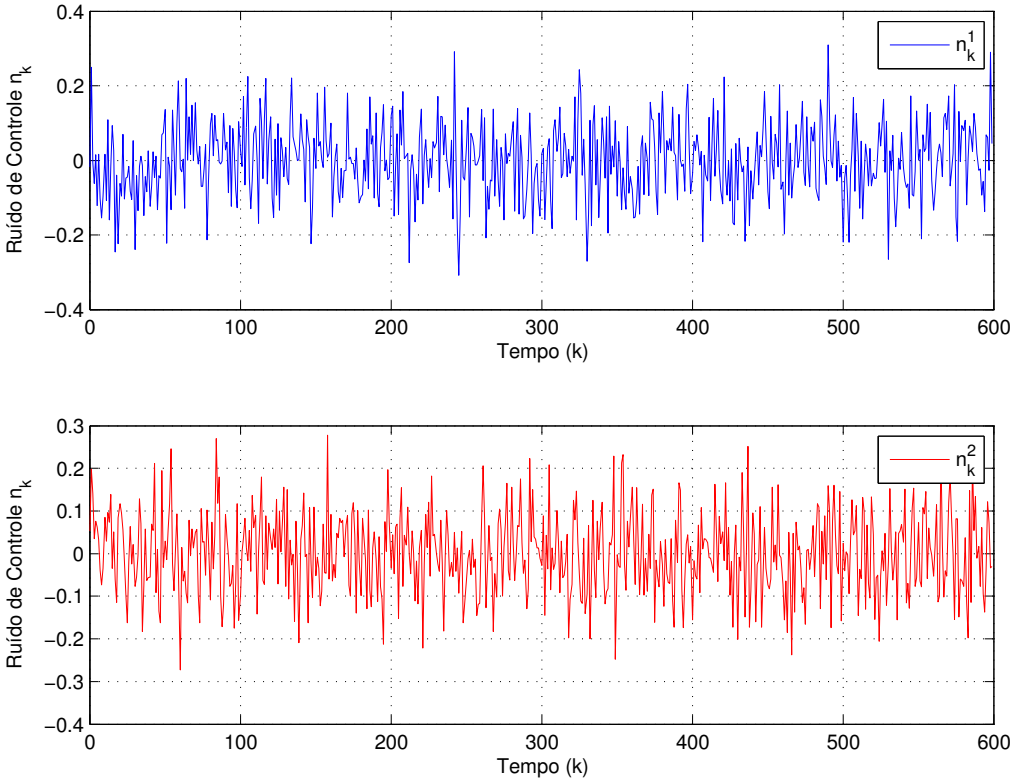



Figura 2: Ruído de controle $n_k$.

Com os valores de $H_j$ pode-se encontrar os parâmetros de $P_j$. Neste caso, por se trabalhar também com a ação $u$, 15 leituras foram necessárias para ajustar o crítico a cada atualização para resolver cada $H_j$. O crítico leva 599 intervalos de tempo para convergir para $H$ e consequentemente para $P$. Nas Figuras 3 e 4 são mostradas a convergência da ação de controle e a localização dos autovalores.

A fim de verificar a independência da ação de controle em relação a matriz $A_d$, na iteração 300, modificou-se os elementos $A_d(1,1) = -0.5$ e $A_d(3,2) = -1$. Nas Figuras 5 e 6 são mostradas a convergência da ação de controle e a localização dos autovalores.

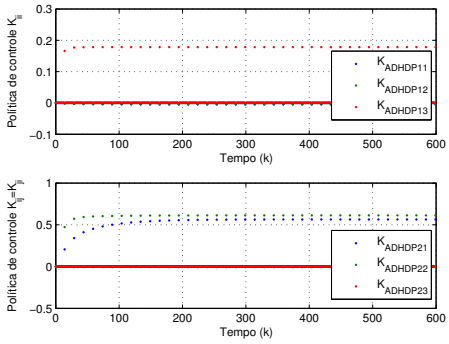



Figura 3: Convergência da política ótima $K$ por AD-HDP.

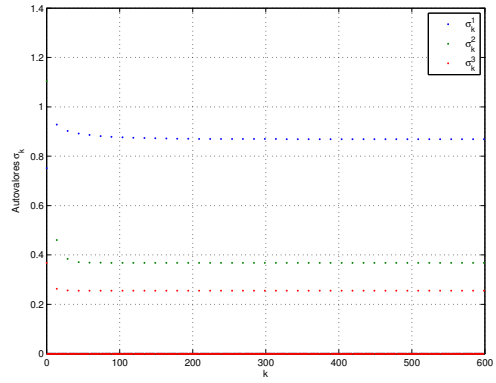



Figura 4: Autovalores $\sigma$ a cada iteração por AD-HDP.

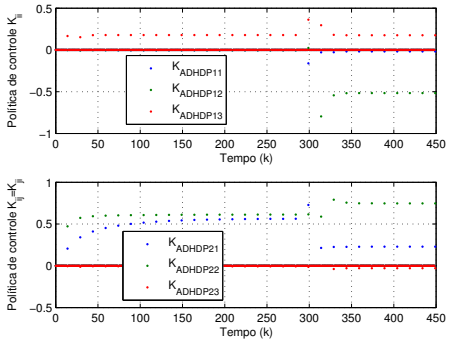



Figura 5: Convergência da política ótima $K$ por AD-HDP para modificação em $A_d$ na 300ª iteração.

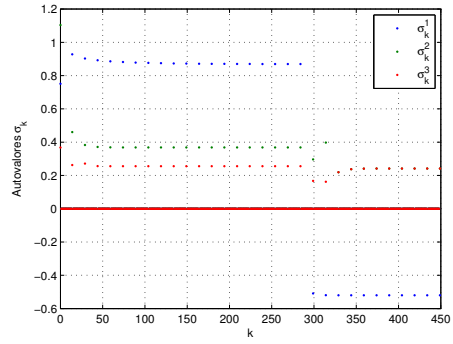



Figura 6: Autovalores $\sigma$ a cada iteração por AD-HDP para modificação em $A_d$ na 300ª iteração.

Para excitar mais o sistema dinâmico, simulou-se uma mudança brusca nos estados na iteração 209. Então, no instante posterior, os estados passaram a ser $x_{210} = \begin{bmatrix} 4 & 2 & 5 \end{bmatrix}$. As Figuras 7 e 8 retratam o comportamento dos estados e da ação de controle.

Pode-se observar as variações no instante 210 tanto para os estados $x_k$ quanto para a ação de controle $u_k$. Em $u_{210}$ tem-se uma pequena oscilação, porém não há modificação no ganho $K$ do controlador e consequentemente não há variação nos autovalores neste instante.

Análise da convergência $QR$ por AD-HDP, é exposta na Tabela 1. Para esta situação seguiu-se a heurística da escolha de $q_i = \{5, 2, 1, 0, 1, -2, -5\}$.

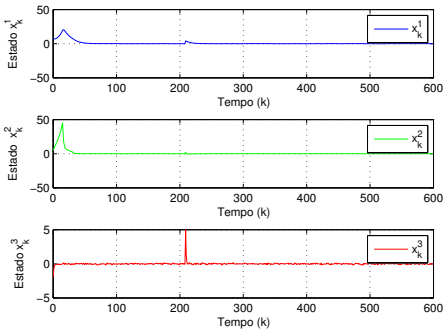



Figura 7: Trajetória dos Estados $x_k$ por AD-HDP com mudança no estado $x_{210}$.

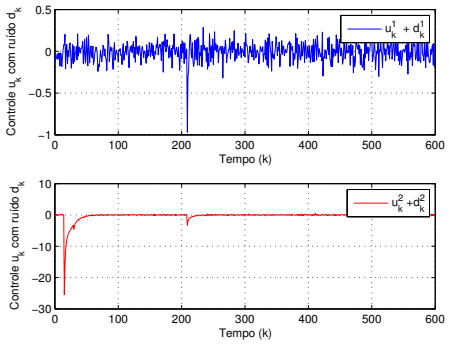



Figura 8: Ação de Controle $u_k$ por AD-HDP com mudança no estado $x_{210}$.

Tabela 1: DLQR-ADP-ADHDP e Variações da Matriz $Q$.

| $q_i$ | $r_i$ | $N$ | Autovalores $\sigma$ |
|---|---|---|---|
| 5 | 0 | 584 | 0.8674, 0.0000 e 0.0000 |
| 2 | 0 | 824 | 0.8674, 0.0097 e 0.0090 |
| 1 | 0 | 779 | 0.8676, 0.0826 e 0.0721 |
| 0 | 0 | 599 | 0.8693, 0.3683 e 0.2553 |
| -1 | 0 | 644 | 0.8808, 0.6584 e 0.3517 |
| -2 | 0 | 659 | 0.9001, 0.7401 e 0.3662 |
| -5 | 0 | 674 | 0.9058, 0.7512 e 0.3679 |

À medida que varia-se $q_i$ seguindo-se a heurística proposta, tem-se uma variação no número de iterações e na localização dos autovalores no plano $Z$. Para $q_i < 0$ tem-se um aumento no número de iterações e os autovalores tendem a se afastar da origem. Quando tem-se $0 \leq q_i \leq 2$ tem-se um aumento no número de iterações, porém dois dos autovalores tendem a se aproximar da origem. Para $q_i = 5$, o número de iterações diminui e os dois autovalores chegam a origem.

## 4 CONSIDERAÇÕES FINAIS

Neste trabalho, foi apresentado o projeto de controlador ótimo discreto do tipo DLQR utilizando um método de ADP, neste caso a AD-HDP. A técnica de ADP viabiliza a sintonia da política de controle e a resolução do problema da DP *online* em avanço no tempo, caracterizando-se assim um controle adaptativo ótimo.

A *priori* desenvolveu-se o projeto do DLQR por DP. Observou-se que o algoritmo chega a solução $P$ de *Riccati*.

Em seguida, foi realizada a implementação *online* do algoritmo AD-HDP apresentado neste trabalho em um sistema MIMO de um modelo de um $F-16$. Observou-se um número de

iterações elevado para a convergência, isto se deve ao fato da quantidade de pontos coletados para construção das matrizes dos regressores com três estados e duas entradas de controle. Este é um problema do método, pois quanto maior o grau do sistema, maior será a matriz de regressores e maior será o gasto computacional.

Foi observado ainda que no algoritmo AD-HDP *online* foi adotado a heurística da variação das matrizes *QR*, sendo assim, obteve-se convergência para diversos valores de $q_i$, porém nenhum padrão foi verificado em relação a variação de $q_i$ e o número de iterações.

O algoritmo AD-HDP desenvolvido, mostrou-se, dentro de sua limitação, uma solução viável e aplicável na prática. Foi mostrado ainda, que tendo somente uma pequena informação sobre os estados do sistema através de sensores e extraídos apenas em momentos específicos, pode-se obter a solução de *Riccati online*. Consequentemente tem-se um controlador ótimo, discreto e adaptativo.

**ABSTRACT.** Due to increasing technological development and the consequent industrial applications, new methods for control design and Reinforcement Learning have been developed, not only to solve new control problems, but also to improve the performance of controllers already implemented in real-world systems. Reinforcement Learning and Discrete Linear Quadratic Regulator approaches are connected by Adaptive Dynamic Programming methods. These paradigms are oriented towards the design of optimal controllers in multivariable systems. For the case of the Discrete Linear Quadratic Regulator, AD-HDP, Reinforcement Learning, Iteration Policy and Iteration Value, a method and an algorithm are developed and implemented for online control design. Based on the selection of the Q and R weighting matrices , a method to tune Discrete Linear Quadratic Regulator controllers is also presented, this method provides guidelines for constructing heuristics for the selection of weighting matrices, aspects of convergence related to the weighting matrices variations are investigated. For a third-order multivariable dynamic system, the proposed algorithm and tuning heuristics are evaluated by the ability to establish the optimal control policy.

**Keywords:** dynamic programming, optimal control, Q-Function, AD-HDP, multivariable systems, convergence, DLQR.

## REFERÊNCIAS

[1] M. Abouheaf & W. Gueaieb. Multi-agent synchronization using online model-free action dependent dual heuristic dynamic programming approach. In “2019 International Conference on Robotics and Automation (ICRA)”. IEEE (2019), p. 2195–2201.

[2] A. Al-Tamimi, F.L. Lewis & M. Abu-Khalaf. Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica*, **43**(3) (2007), 473–481.

[3] A. Al-Tamimi, D. Vrabie, M. Abu-Khalaf & F.L. Lewis. Model-free approximate dynamic programming schemes for linear systems. In “International Joint Conference On Neural Networks”. IEEE (2007), p. 371–378.

[4] S. Balakrishnan, J. Ding & F.L. Lewis. Issues on stability of ADP feedback controllers for dynamical systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **38**(4) (2008), 913–917.

[5] R. Bellman. Dynamic programming and stochastic control processes. *Information and control*, **1**(3) (1958), 228–239.

[6] R. Bellman. "Dynamic programming". Dover Publications Inc (2003).

[7] D. Bertsekas. "Dynamic programming and optimal control: Volume I", volume 2. Athena scientific (1995).

[8] S.J. Bradtke, B.E. Ydstie & A.G. Barto. Adaptive linear quadratic control using policy iteration. In "Proceedings of 1994 American Control Conference-ACC'94", volume 3. IEEE (1994), p. 3475–3479.

[9] J. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on circuits and systems*, **25**(9) (1978), 772–781.

[10] J.V. da Fonseca Neto & L.R. Lopes. On the convergence of DLQR control system design and recurrences of riccati and lyapunov in dynamic programming strategies. In "UkSim 13th International Conference on Computer Modelling and Simulation". IEEE (2011), p. 26–31.

[11] F.N. Da Silva & J.V.D.F. Neto. Data driven state reconstruction of dynamical system based on approximate dynamic programming and reinforcement learning. *IEEE Access*, **9** (2021), 73299–73306.

[12] S. Dian, H. Fang, T. Zhao, Q. Wu, Y. Hu, R. Guo & S. Li. Modeling and trajectory tracking control for magnetic wheeled mobile robots based on improved dual-heuristic dynamic programming. *IEEE Transactions on Industrial Informatics*, **17**(2) (2020), 1470–1482.

[13] J. Doyle & G. Stein. Multivariable feedback design: Concepts for a classical/modern synthesis. *IEEE transactions on Automatic Control*, **26**(1) (1981), 4–16.

[14] G.F. Franklin, J.D. Powell, A. Emami-Naeini & J.D. Powell. "Feedback control of dynamic systems", volume 4. Prentice Hall, Upper Saddle River (2002).

[15] H. Gjorshevski, K. Trivodaliev, I.N. Kosovic, S. Kalajdziski & B.R. Stojkoska. Dynamic programming approach for drone routes planning. In "2018 26th Telecommunications Forum (TELFOR)". IEEE (2018), p. 1–4.

[16] M.M. Gupta & N.K. Sinha. "Intelligent control system". IEEE Press, New York (1996).

[17] Y. Jiang & Z.P. Jiang. Approximate dynamic programming for output feedback control. In "Proceedings of the 29th Chinese control conference". IEEE (2010), p. 5815–5820.

[18] M. Johnson & M. Grimble. Recent trends in linear optimal quadratic multivariable control system design. In "IEE proceedings D (control theory and applications)", volume 134. IET (1987), p. 53–71.

[19] D.E. Kirk. "Optimal control theory: an introduction". Courier Corporation (2004).

[20] F. Köpf, S. Ebbert, M. Flad & S. Hohmann. Adaptive dynamic programming for cooperative control with incomplete information. In "IEEE International Conference on Systems, Man, and Cybernetics (SMC)". IEEE (2018), p. 2632–2638.

[21] N. Kumar, A. Varshney, K. Shrama & B. Prasad. Dual heuristic programming based controller for deregulated AGC scheme. In "2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC)". IEEE (2019), p. 207–212.

[22] B.C. Kuo. "Digital Control System". Halt, Rinehart and Winston Inc (1980).

[23] T. Landelius & H. Knutsson. Greedy adaptive critics for LQR problems: Convergence proofs. Report LiTH-ISY-R-1896. *Computer Vision Laboratory. SE-581*, **83** (1996).

[24] G.G. Lendaris. A retrospective on adaptive dynamic programming for control. In "2009 International Joint Conference on Neural Networks". IEEE (2009), p. 1750–1757.

[25] F.L. Lewis & B.L. Stevens. Aircraft control and simulation.

[26] F.L. Lewis & D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE circuits and systems magazine*, **9**(3) (2009), 32–50.

[27] F.L. Lewis, D. Vrabie & V.L. Syrmos. "Optimal control". John Wiley & Sons (2012).

[28] D. Liu, S. Xue, B. Zhao, B. Luo & Q. Wei. Adaptive dynamic programming for control: A survey and recent advances. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **51**(1) (2020), 142–160.

[29] G.G. Meyer, K. Främling & J. Holmström. Intelligent products: A survey. *Computers in industry*, **60**(3) (2009), 137–148.

[30] C. Mu, D. Wang & H. He. Data-driven finite-horizon approximate optimal control for discrete-time nonlinear systems using iterative HDP approach. *IEEE transactions on cybernetics*, **48**(10) (2017), 2948–2961.

[31] J.J. Murray, C.J. Cox, G.G. Lendaris & R. Saeks. Adaptive dynamic programming. *IEEE transactions on systems, man, and cybernetics, Part C (Applications and Reviews)*, **32**(2) (2002), 140–153.

[32] W.B. Powell. "Approximate Dynamic Programming: Solving the curses of dimensionality", volume 703. John Wiley & Sons (2007).

[33] J. Si, A.G. Barto, W.B. Powell & D. Wunsch. "Handbook of learning and approximate dynamic programming", volume 2. John Wiley & Sons (2004).

[34] B. Sun & E.J. van Kampen. Launch vehicle discrete-time optimal tracking control using global dual heuristic programming. In "IEEE Conference on Control Technology and Applications (CCTA)". IEEE (2020), p. 162–167.

[35] B. Sun & E.J. van Kampen. Reinforcement-learning-based adaptive optimal flight control with output feedback and input constraints. *Journal of Guidance, Control, and Dynamics*, **44**(9) (2021), 1685–1691.

[36] T. Sun & X.M. Sun. An adaptive dynamic programming scheme for nonlinear optimal control with unknown dynamics and its application to turbofan engines. *IEEE transactions on industrial informatics*, **17**(1) (2020), 367–376.

[37] D. Vrabie, O. Pastravanu, M. Abu-Khalaf & F.L. Lewis. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica*, **45**(2) (2009), 477–484.

[38] Q. Wei, R. Song, Z. Liao, B. Li & F.L. Lewis. Discrete-time impulsive adaptive dynamic programming. *IEEE Transactions on Cybernetics*, **50**(10) (2019), 4293–4306.

[39] P.J. Werbos. "Neural Networks for Control, chapter A menu of designs for reinforcement learning over time". MIT Press, Cambridge (1990).

[40] P.J. Werbos. Foreword-ADP: The key direction for future research in intelligent control and understanding brain intelligence. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **38**(4) (2008), 898–900.

[41] S. Ye, Y.J. Zhou, G.P. Jiang & Q. Lin. Optimization control of UAVs based on self-learning adaptive dynamic programming. In "2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC)". IEEE (2020), p. 738–743.

[42] Q. Zhao, J. Si & J. Sun. Online reinforcement learning control by direct heuristic dynamic programming: From time-driven to event-driven. *IEEE transactions on neural networks and learning systems*, **33**(8) (2021), 4139–4144.

[43] N. Zheng & P. Mazumder. A low-power circuit for adaptive dynamic programming. In "31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)". IEEE (2018), p. 192–197.

**How to cite**

M. Cerqueira, L. Lopes & J.V. Fonseca. Sintonia *QR* do Regulador Linear Quadrático LQR Discreto e Programação Dinâmica Aproximada baseada em Ação-Estado para Aplicações Online do Projeto de Sistemas de Controle Ótimo. *Trends in Computational and Applied Mathematics*, **25**(2024), e01686. doi: 10.5540/tcam.2024.025.e01686.

(cc) BY