Universidade Estadual do Maranhão
Centro de Ciências Tecnológicas
Programa de Pós-Graduação em Engenharia de Computação e Sistemas
Mestrado Profissional em Engenharia da Computação e Sistemas

Guilherme Bonfim de Sousa

Convergência e Estabilidade Numérica de Algoritmos de Programação Dinâmica Heurística Dependente de Ação Baseados na Abordagem RLS para Controle DLQR Online

São Luís

2018

Guilherme Bonfim de Sousa

Convergência e Estabilidade Numérica de Algoritmos de Programação Dinâmica Heurística Dependente de Ação Baseados na Abordagem RLS para Controle DLQR Online

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia da Computação e Sistemas da Universidade Estadual do Maranhão como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia da Computação.

Orientadora: Profa. Patrícia Helena Moraes Rêgo, Dra.

São Luís

2018

Sousa, Guilherme Bonfim de.

Convergência e Estabilidade Numérica de Algoritmos de Programação Dinâmica Heurística Dependente de Ação Baseados na Abordagem RLS para Controle DLQR Online / Guilherme Bonfim de Sousa. – São Luís, 2017.

171f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia da Computação e Sistemas, Universidade Estadual do Maranhão, 2017.

Orientador: Profa. Dra. Patrícia Helena Moraes Rêgo.

1. Aprendizagem por reforço. 2. Programação dinâmica heurística dependente de ação. 3. Regulador linear quadrático discreto. 4. Decomposição QR. 5. Fatoração UDU^T. I. Título.

CDU 004.42

CONVERGÊNCIA E ESTABILIDADE NUMÉRICA DE APROXIMADORES PARAMÉTRICOS DE FUNÇÃO VALOR BASEADOS NA ABORDAGEM RLS PARA CONTROLE DLQR ONLINE VIA PROGRAMAÇÃO DINÂMICA HEURÍSTICA DEPENDENTE DE AÇÃO

Guilherme Bonfim de Sousa

APROVADO EM: 19 de fevereiro de 2018.

Profa. Patrícia Helena Moraes Rêgo, Dra.

(Orientadora)

Prof. Henrique Mariano Costa do Amaral, MsC.

(Membro da Banca Examinadora)

Prof. Areolino de Almeida Neto, Dr.

(Membro da Banca Examinadora)

Prof. Vilemar Gomes da Silva, Dr.

(Membro da Banca Examinadora)

“A vida só pode ser compreendida, olhando-se para trás; mas só pode ser vivida, olhando-se para frente.”

Soren Kierkegaard

À minha família pelo suporte emocional que ajudou a tornar esse sonho possível e à minha orientadora por sua dedicação e instruções que me motivaram a concluir esse projeto.

Agradecimentos

Em primeiro lugar a Deus, fonte de inspiração para os momentos diários dessa jornada.

A meus pais pelo amor incondicional e aporte moral, a meus tios pelo carinho e segundo lar que me proporcionaram nesses anos.

À minha orientadora, Profª. Dra. Patrícia Helena Moraes Rêgo, pela orientação precisa, dedicação e companheirismo durante a realização deste trabalho.

Aos grandes amigos do PECS que me acompanharam e me ajudaram nessa jornada, Marta Raquel, David Silva, Beatriz Nery, Sara Martins, Rildenir Silva e Cristovam Filho. A todos que contribuíram de forma direta ou indireta nesse árduo projeto.

À Universidade Estadual do Mareanhão - UEMA, ao Programa de Pós Graduação em Engenharia da Computação e Sistemas, e à Fundação de Amparo à Pesquisa e ao Desenvolvimento Científico e Tecnológico do Maranhão - FAPEMA pelos recursos materiais e financeiros destinados a essa pesquisa.

Resumo

Para contornar problemas de estabilidade numérica que geralmente ocorrem em paradigmas de Programação Dinâmica Aproximada baseados na abordagem dos mínimos quadrados recursivos (RLS - *Recursive Least-Squares*), propõe-se um método para promover melhorias nas aproximações da função valor de ação para algoritmos *online* do projeto de sistema de controle ótimo. Os algoritmos resultantes desta metodologia são incorporados nas arquiteturas ator-crítico baseadas em Programação Dinâmica Heurística Dependente de Ação (ADHDP – *Action Dependent Heuristic Dynamic Programming*), o que caracteriza este projeto como um quadro de aprendizagem que resolve problemas de decisão ótima *online* em tempo real sem conhecimento *a priori* do modelo da dinâmica do sistema. A solução proposta está fundamentada em transformações unitárias e métodos de decomposição $\mathcal{QR}$ e UDU^T, as quais são integradas na rede crítica para melhorar a eficiência do aprendizado RLS para realização *online* do projeto do regulador linear quadrático discreto. Do ponto de vista de estabilidade numérica e custo computacional, a estratégia de aprendizagem desenvolvida é projetada para fornecer algoritmos de ADHDP com melhores especificações para implementação em sistemas de controle ótimo do mundo real.

Palavras-chaves: Programação Dinâmica, Aprendizagem por Reforço, Programação Dinâmica Heurística Dependente de Ação, Controle Ótimo, Regulador Linear Quadrático Discreto, Mínimos Quadrados Recursivos, Decomposições $\mathcal{QR}$ e UDU^T.

Abstract

In order to overcome numerical stability problems that usually occur in approximate dynamic programming paradigms based on recursive least-squares (RLS) approach, a method to promote improvements in the action-value function approximations for online algorithms of the optimal control system design is proposed. The algorithms resulting from that methodology are embedded in actor-critic architectures based on Action Dependent Heuristic Dynamic Programming (ADHDP), which characterizes this design as a learning framework that solves optimal decision problems online in real-time without requiring a priori knowledge of the system dynamic model. The proposed solution is grounded on unitary transformations and $\mathcal{QR}$ and UDU^T decomposition methods, which are integrated in the critic network to improve the RLS learning efficiency for online realization of the discrete linear quadratic regulator design. From the numerical stability and computational cost point of view, the developed learning strategy is designed to provide ADHDP algorithms with better specifications for implementation in real-world optimal control systems.

Keyword: Dynamic Programming, Reinforcement Learning, Action-Dependent Heuristic Dynamic Programming, Optimal Control, Discrete Linear Quadratic Regulator, Recursive Least-Squares, $\mathcal{QR}$ and UDU^T Decompositions.

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas e Siglas

AC	*Adaptive Critic* (Crítico Adaptativo)
ACDs	*Adaptive Critic Designs* (Projetos Críticos Adaptativos)
ADDHP	*Action Dependent Dual Heuristic Programming* (Programação Heurística Dual Dependente de Ação)
ADHDP	*Action Dependent Heuristic Dynamic Programming* (Programação Dinâmica Heurística Dependente de Ação)
ADP	*Approximate/Adaptive Dynamic Programming* (Programação Dinâmica Aproximada/Adaptativa)
DARE	*Discrete Algebraic Riccati Equation* (Equação Algébrica de *Riccati* Discreta)
DHP	*Dual Heuristic Programming* (Programação Heurística Dual)
DLQR	*Discrete Linear Quadratic Regulator* (Regulador Linear Quadrático Discreto)
DP	*Dynamic Programming* (Programação Dinâmica)
HDP	*Heuristic Dynamic Programming* (Programação Dinâmica Heurística)
HJB	*Hamilton-Jacobi-Bellman*
LQR	*Linear Quadratic Regulador* (Regulador Linear Quadrático)
LS	*Least-squares* (Mínimos Quadrados)
MIMO	*Multiple-Input and Multiple-Output* (Múltiplas-Entradas e Múltiplas-Saídas)
PDM	Processo de Decisão Markoviano
RL	*Reinforcement Learning* (Aprendizagem por Reforço)
RLS	*Recursive Least-Squares* (Mínimos Quadrados Recursivos)
TD	*Temporal Difference* (Diferença Temporal)

Sumário

1

INTRODUÇÃO

A operação de sistemas ligados aos mais diferentes setores, tais como indústria, ciência, tecnologia, dentre outros, tem requerido cada vez mais estratégias de controle que satisfaçam os objetivos de desempenho prescritos de uma maneira robusta. Tais sistemas são de uma complexidade elevada, que exige controladores de malha fechada de alto desempenho para executar as tarefas programadas. Uma abordagem para atender aos requisitos desses projetos é encontrada na teoria de controle ótimo, que lida com a operação de um sistema dinâmico complexo com um custo mínimo (LEWIS; VRABIE; VAMVOUDAKIS, 2012; KIRK, 2004).

Métodos clássicos de controle ótimo têm alcançado grande sucesso para sistemas lineares via equações do tipo *Riccati*, mas este sucesso é restrito a algumas aplicações *online*. De maneira geral, os métodos são *offline* e requerem o conhecimento completo do modelo do sistema dinâmico. De fato, a solução de controle ótimo baseada em modelo é geralmente difícil ou impossível de ser obtida devido à não-linearidades e/ou restrições de constante de tempo presentes nos sistemas do mundo real (planta).

Nessa perspectiva, diversos conceitos matemáticos e métodos de otimização dinâmica foram agregados para desenvolver soluções mais abrangentes de controle ótimo, dentre os quais destacam-se os métodos de *Bellman* da Programação Dinâmica (*Dynamic Programming* - DP), com a subsequente equação de *Hamilton-Jacobi-Bellman* (HJB), (BELLMAN, 1958, 2003; BERTSEKAS, 1987; BERTSEKAS et al., 1995; HOWARD, 1960). A programação dinâmica fornece um formalismo matemático para tratar problemas estocásticos não-lineares sob condições que são difíceis de lidar usando-se a abordagem de cálculo variacional, (ATHANS; FALB, 2007; BRYSON, 1975; KIRK, 2004; LEWIS; VRABIE; VAMVOUDAKIS, 2012).

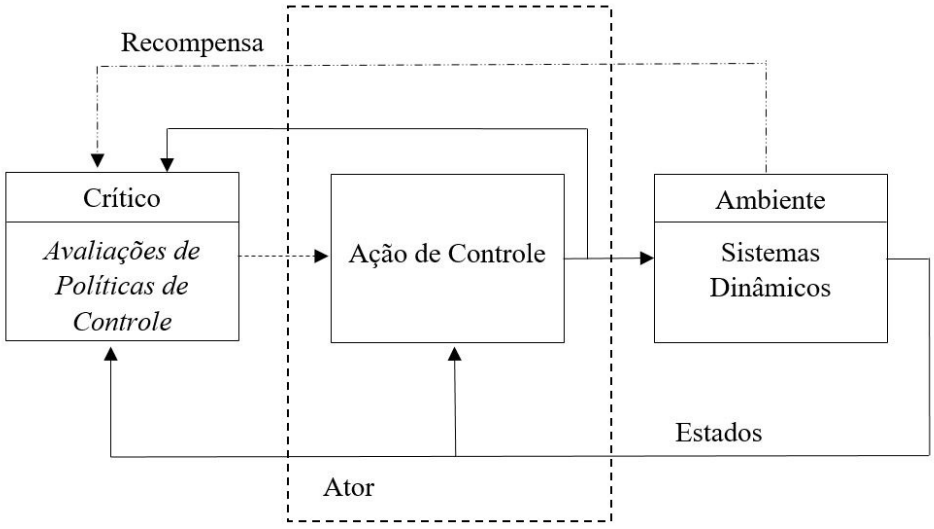
Contudo, a principal desvantagem em programação dinâmica clássica é que os recursos computacionais associados com a solução da equação HJB são de alto custo para implementação *online* (LENDARIS, 2009b), uma vez que a DP é um método "para trás no tempo" (*backward-in-time*). Além disso, em várias aplicações, a dinâmica da planta muda tanto durante a operação, de modo que os projetos de controladores sintetizados pelos métodos men-

cionados acima não apresentam um desempenho satisfatório. Uma alternativa para contornar essas limitações são os métodos de aproximação que empregam um tipo de crítico adaptativo da aprendizagem por reforço (RL - *Reinforcement Learning*)(RÊGO, 2014).

A abordagem crítica adaptativa (AC - *Adaptive Critic*), também denominada programação dinâmica aproximada/adaptativa (ADP - *Approximate/Adaptive Dynamic Programming*), possui a importante vantagem de resolver as equações HJB em uma maneira "para frente no tempo" (*forward-in-time*), diretamente viável para aplicações *online*. Nesta abordagem, o sistema de aprendizagem por reforço é caracterizado como um mecanismo de busca de política de controle ótima que constrói, incrementalmente, estimativas da função de custo de uma dada política de controle baseadas em dados observados do sistema ao longo de sua trajetória (estados, e recompensas instantâneas associadas que caracterizam o desempenho instantâneo de um dado controlador).

Um esquema de crítico adaptativo está ilustrado na Figura 1.1, em que o agente de aprendizagem (controlador/ator) interage com um ambiente (sistema dinâmico) inicialmente desconhecido e modifica sua política de controle atual de modo que a nova política produz um valor que é melhorado em relação aos valores anteriores. Desta forma, o agente RL é suposto aprender a política ótima a partir de suas experiências sem conhecer os parâmetros do sistema dinâmico.

Figura 1.1: Aprendizagem por reforço com estrutura ator-crítico.



As variações de métodos críticos adaptativos mais comuns na literatura são os métodos de diferenças temporais, (SUTTON, 1988), e aprendizagem Q, (WATKINS, 1989; WATKINS; DAYAN, 1992). Estes, por sua vez, têm sido classificados por Werbos (1990, 1992) em progra-

mação dinâmica heurística (*Heuristic Dynamic Programming* - HDP) e programação heurística dual (*Dual Heuristic Programming* - DHP). Existem versões modificadas chamadas críticas dependentes da ação de cada uma delas, nominalmente Programação Dinâmica Heurística Dependente de Ação (*Action Dependent Heuristic Dynamic Programming* - ADHDP) e Programação Heurística Dual Dependente de Ação (*Action Dependent Dual Heuristic Programming* - ADDHP).

As aproximações são classificadas de acordo com as abordagens estabelecidas para estimar a função valor $V(x)$, sendo x as variáveis de estado. Em termos funcionais, HDP é baseada em estimativas de $V(x)$, enquanto DHP usa estimativas do gradiente de $V(x)$ com respeito às variáveis de estado x. Estratégias dependentes de ação, por sua vez, estão associadas com estimativas da função $Q(x, u)$ e seus gradientes com respeito às variáveis de estado x e decisão u (lei de controle) para HDP e DHP, respectivamente. Durante as últimas décadas, várias metodologias de ADP baseadas em projetos críticos adaptativos têm sido propostas na literatura. Lewis e Liu (2012) referem-se a ADP como uma família de métodos ator-crítico para encontrar as melhores soluções em tempo real do problema de controle ótimo.

Os métodos RL são inspirados em mecanismos de aprendizagem natural (grupos sociais, sistemas naturais), em que homens/animais adaptam suas ações baseados na interação com o ambiente. Tais métodos foram usados por Ivan Pavlov na década de 1860 para treinar seus cães (PAVLOV; ANREP, 2003). Em RL o ator/agente interage com o ambiente e modifica suas ações (políticas de controle) baseadas em estímulos que recebe do ambiente decorrentes de suas ações. Os algoritmos de RL são desenvolvidos baseados na ideia de que as decisões de controle de sucesso devem ser lembradas, por meio da utilização de um sinal de reforço, para que essas decisões possam ser usadas outras vezes. Assim, o foco do processo de aprendizagem está direcionado a um índice de desempenho que quantifica o quão próximo do estado ótimo se encontra o sistema de controle de malha fechada, sem levar em consideração o modelo da dinâmica do sistema.

Um assunto relevante no quadro de Controle Ótimo e Programação Dinâmica Aproximada refere-se às arquiteturas de aproximação e algoritmos de treinamento de críticos adaptativos para aproximar funções de custo e políticas de controle ótimo. Por exemplo, estruturas baseadas em redes neurais foram propostas em (HANSELMANN; MUSICKI; PALANISWAMI, 2006) para aprender a solução de controle ótimo via equação HJB. Algoritmos críticos adaptativos e de treinamento de redes neurais foram apresentados tanto no quadro de tempo discreto

(SI, 2004), quanto no quadro de tempo contínuo (WATKINS; DAYAN, 1992).

Em metodologias de projeto de controle baseadas em ADP, dentre os algoritmos iterativos propostos para estimar os parâmetros da função valor, o aprendizado dos mínimos quadrados recursivos (RLS do inglês *Recursive Least-Squares*) é um dos mais bem-sucedidos. Tal resultado favorável é principalmente devido à sua robustez para lidar com variações no tempo dos parâmetros de regressão e a rápida convergência quando comparado com os métodos de gradiente estocástico. Um grande número de desenvolvimentos e aplicações dos métodos RLS pode ser encontrado na literatura, alguns dos quais lidam com problemas de treinamento de redes neurais (YEH; SU; RUDAS, 2011; YEH; SU, 2012). Estruturas baseadas em RLS têm sido propostas em (XU; HE; HU, 2002) para melhorar a eficiência dos métodos heurísticos críticos adaptativos convencionais. Em (CHENG; FENG; WANG, 2013; XU; HE; HU, 2002), os autores exploram métodos RLS para resolver problemas de aprendizagem por reforço ator-crítico.

No contexto de aprendizagem por reforço e programação dinâmica aproximada, aprendizado RLS tem despertado grande interesse em questões relacionadas à sua estabilidade numérica. Alguns trabalhos tais como (FERREIRA; NETO; RÊGO, 2016; RÊGO; NETO; FERREIRA, 2013; RÊGO, 2014) fornecem propostas com base em transformações unitárias para resolver problemas de convergência e estabilidade numérica relacionados ao mal condicionamento da matriz de covariância da abordagem RLS para aproximação de função valor via HDP para solução *online* do DLQR. Em geral, a importância destas investigações está no escopo de aplicações de aprendizagem por reforço para o projeto de controle ótimo *online* via solução em tempo real da equação HJB.

Do ponto de vista de análise de estabilidade numérica mostramos, nesta pesquisa, a necessidade para melhores implementações da metodologia de ADHDP baseada em estimadores RLS para o projeto de controle ótimo DLQR *online*. Tem-se observado que o problema da perda de estabilidade numérica ocorre principalmente durante o comportamento em regime permanente do sistema dinâmico controlado, enquanto que durante o comportamento transitório, tem-se informação o suficiente para gerar a base de vetores de regressores. *À priori*, parte-se da premissa que os elementos que compõem o vetor de funções de base tendem a tornar-se linearmente dependentes após alcançar o regime permanente. Este tipo de fenômeno inviabiliza o projeto *online* de controle ótimo já que o sistema perde a estabilidade numérica. Neste trabalho, propomos um método que lida com assuntos numéricos. O método proposto não somente melhora estabilidade numérica, mas também reduz, substancialmente, o esforço computacio-

nal ao aproximar a função de custo DLQR. Isso permite mais aplicações em tempo real de controladores ótimos baseados em ADHDP.

1.1 Estado da Arte

A automação desempenha na engenharia e na ciência um papel muito importante, uma vez que busca-se aprimorar os processos de maneira a conseguir a maior produção e menor custo, sendo desta forma uma abordagem indispensável a ambientes industriais de alta produtividade. Tal processo acarreta uma busca por estratégias de controle mais robustas, visto que os sistemas dinâmicos que modelam os projetos de controle adquirem um grau elevado de complexidade. Para tanto várias técnicas de controle têm sido propostas com o objetivo de alcançar índices de satisfação. Uma metodologia bem estudada é a do Controle Ótimo, que propõe em sua solução, minimizar ou maximizar um índice de desempenho associado ao comportamento da planta (ANDERSON; MOORE, 2007; DIERKS; JAGANNATHAN, 2011; QUIRION; GUNN; GU, 2004).

Uma desvantagem dos métodos clássicos de controle ótimo é que os projetos de controle são normalmente realizados via solução *offline* da equação de *Hamilton-Jacobi-Bellman*, visto ainda que métodos exigem o conhecimento do modelo dinâmico do sistema. Assim, para aplicações em que a dinâmica do sistema sofre muitas variações, o projeto de um controlador fixo, via métodos clássicos, não é suficientes (LENDARIS, 2009a).

Indo de encontro a abordagens para soluções cada vez mais abrangentes, uma série de questões relacionadas a estes processos foram tratadas e resolvidas, implicando o desenvolvimento de conceitos matemáticos e algoritmos. Dentre eles, a Programação Dinâmica ganhou espaço como um método que visa encontrar a solução ótima para problemas de tomada de decisão sequencial, com seu desenvolvimento ocorrendo no final dos anos 50 com os trabalhos de *Bellman* e *Pontrygin* (BELLMAN, 1958; BERTSEKAS et al., 1995), tendo a equação HJB conseguido sucesso na solução de uma ampla classe de problemas de otimização não-linear.

O problema para a DP está no alto custo computacional associado à realização das formulações, dificultando a solução de problemas de controle ótimo de uma forma "para frente no tempo", o que é necessário em aplicações em tempo real. Sendo assim, a DP caracteriza-se por ser um procedimento "para trás no tempo" com solução por recorrência (LENDARIS, 2009a).

Inspirada nos mecanismos de aprendizagem natural, em que organismos vivos interagem com seu ambiente e usam essas interações para melhorar suas ações, e assim sobreviverem,

surge a Aprendizagem por Reforço (RL). Essa metodologia caracteriza-se por um ator ou agente que interage com o ambiente e modifica suas ações, ou políticas de controle, baseados nos estímulos recebidos em resposta a essas ações (SUTTON; BARTO, 1998), o que garante ao sistema aprender qual é a melhor decisão tendo por base experiências passadas, fornecendo dessa maneira uma metodologia para resolver problemas de decisões sequenciais incertas e complexas a muitas aplicações.

Da junção desses métodos, chega-se à Programação Dinâmica Aproximada (MURRAY et al., 2002; STINGU; LEWIS, 2011), vista como uma classe de algoritmos que fornece uma solução *online* para problemas de controle ótimo usando representações aproximadas da função valor, função essa a ser minimizada tendo por base o princípio da otimalidade de *Bellman*. Objetivando-se estabelecer políticas de tomada de decisão que minimizem o custo total sobre um número de estágios, tais problemas são desafiadores por causa da compensação entre custos imediatos e custos futuros. Um tratamento introdutório sobre programação dinâmica e sua relação com a aprendizagem por reforço é apresentado em Haykin (2004).

Um campo de investigação similar foi desenvolvido por Werbos, que propôs uma família de métodos denominados projetos críticos adaptativos (*Adaptive Critic Designs* - ACDs) (WERBOS, 1992) sendo esse conceito baseado na combinação das ideias de RL e DP. Enquanto a programação dinâmica determina o controle por meio da função valor ótimo, o crítico adaptativo utiliza uma aproximação dessa função para realizar o projeto de controle, possibilitando sua aplicação *online* com solução da equação HJB de uma forma "para frente no tempo"(LENDARIS, 2009a).

Várias pesquisas têm fornecido uma visão unificada de controle ótimo e controle adaptativo baseadas em RL e ADP. O trabalho de Sutton e outros (SUTTON; BARTO; WILLIAMS, 1992) discute aprendizagem por reforço (aprendizagem-Q) como uma abordagem de controle ótimo adaptativo direto. Lee e outros (LEE; PARK; CHOI, 2010) apresentam provas de convergência e estabilidade de malha fechada para um esquema de controle ótimo adaptativo baseado em iteração de política para sistemas lineares de tempo contínuo incertos com sinais de excitação. São exemplos de aplicações de projeto de controle ótimo *online*: aeronaves de alto desempenho ou sistemas de mísseis, (BERTSEKAS et al., 2000), piloto automático, (FERRARI; STENGEL, 2004; LIN, 2005), e robótica, (KHAN et al., 2012).

Pesquisas usando os métodos de mínimos quadrados em conjunção com iteração política têm sido apresentadas para solução *online* de problemas de controle ótimo, (BUSONIU et

al., 2010; LI; LITTMAN; MANSLEY, 2009). A partir de um conjunto de abordagens, métodos RLS também têm sido usados para viabilizar a realização *online* em uma forma mais eficiente, (CHENG; FENG; WANG, 2013; XU; HE; HU, 2002). No entanto, observa-se nos algoritmos RLS convencionais uma degradação da exatidão numérica associada a problemas de mal condicionamento da matriz de covariância (POTTER, 1963). Diferentes formas de análise foram desenvolvidas para explicar a origem, bem como a propagação do fenômeno da instabilidade numérica (LIAVAS; REGALIA, 1999; LJUNG; LJUNG, 1985; VERHAEGEN, 1989).

Na realização de controle ótimo em tempo real tem-se a preocupação em relação a estabilidade e convergência numérica dos algoritmos de controle, sendo a estabilidade numérica uma característica que garante que o algoritmo apresentará o mesmo comportamento ao passar do tempo de execução (KITAHARA; SARIDIS, 1972).

Um procedimento alternativo de solução para problemas de convergência e estabilidade numérica inerente à implementação de algoritmos RLS envolve a redução da matriz de covariância às formas de fatoração $\mathcal{QR}$ e UDU^T. Um dos primeiros trabalhos que utilizou decomposição $\mathcal{QR}$ para resolver o problema RLS foi proposto por Gentleman (CAMPOS; STRANG, 2009; GENTLEMAN, 1973). Ele usou um arranjo triangular para evitar a inversão da matriz e propôs uma sequência direcionada de rotações de Givens para realizar o processo de "substituição para trás" necessário para resolver o sistema associado de equações. A partir dos trabalhos de Gentleman o algoritmo $\mathcal{QR}$-RLS convencional foi concebido por McWhirter (CAMPOS; STRANG, 2009; MCWHIRTER, 1983a, 1983b). Ele foi o primeiro a descrever uma matriz sistólica, usando uma sequência de rotações de Givens, executando uma triangularização ortogonal da matriz de dados de entrada e gerando o erro estimado sem ter que recorrer à "substituição para trás". O uso da fatoração UDU^T na estimação RLS garante também propriedades numéricas desejáveis (RÊGO; NETO; FERREIRA, 2013). Questões sobre estabilidade computacional e precisão dos métodos de fatoração UDU^T foram amplamente estudadas em (GOLUB; LOAN, 2012; WILKINSON, 1968).

Desta forma, a abordagem aqui proposta é vista como uma melhoria no processo de estimação RLS de políticas de decisão ótima DLQR para evitar problemas de convergência e estabilidade numérica via fatoração $\mathcal{QR}$ e UDU^T, fornecem medidas de desempenho que avaliam o grau de dependência linear dos vetores de regressores que devem formar uma base para a aproximação RLS da solução da equação HJB-*Riccati*. O número de condição e parâmetro de positividade da matriz de covariância são usados como estratégia para avaliar o comportamento

do processo de estimação RLS. A fatoração UDU^T é, então, inserida no processo de cálculo da matriz de covariância da abordagem RLS.

1.2 Motivação e Relevância

Os avanços crescentes dos setores tecnológicos e industriais exigem cada vez mais sistemas de controle capazes de serem implementados em tempo real, com rigorosas especificações de projetos e aptos a rejeitarem perturbações e incertezas, tais como: projeto de aeronaves de alto desempenho, redes elétricas e robôs de alta precisão. Assim, sistemas de controle MIMO (*Multiple-Input and Multiple-Output*) tornam-se preferíveis visto à disponibilidade de sensores e atuadores disponíveis. Além disso é importante destacar que os sistemas de controle não-lineares correspondem à grande maioria dos sistemas do mundo real. Somado a esses requisitos, busca-se um custo computacional associado a resolução do problema de controle com quantidade de cálculos e uso de memória pertinentemente baixos, visto que os processos devem ser executados em microprocessadores que são responsáveis pelo controle de elementos cada vez mais integrados ao nosso meio, como automóveis, máquinas automáticas e processos industriais. Combinar todos esses requisitos e informações para alcançar o melhor desempenho possível é o principal desafio em sistemas de controle *online*.

Em pesquisas recentes, os métodos de controle ótimo baseados em programação dinâmica aproximada e aprendizagem por reforço têm se destacado como uma ferramenta muito útil para melhorar o desempenho de sistemas do mundo real e contribuído para viabilizar as implementações em tempo real de controladores ótimos. Mas alguns requisitos de desempenho ainda necessitam ser melhorados na implementação de métodos de aprendizagem por reforço e programação dinâmica aproximada para o controle de sistemas não-lineares, tais como a estabilidade numérica e a convergência dos algoritmos, dentre outros. Este trabalho aborda alguns desses requisitos e fornece métodos baseados em RL e ADP via abordagem RLS para um controle ótimo em tempo real de sistemas não-lineares.

1.3 Objetivos

O objetivo geral deste trabalho é a melhoria no desempenho de estimadores RLS-ADHDP da função de custo DLQR via métodos de decomposição $\mathcal{QR}$ e UDU^T, tendo como referência a complexidade computacional e estabilidade numérica dos algoritmos propostos, para obter políticas de decisão ótimas baseadas na Equação HJB-*Riccati* discreta para realização *online*

de projeto de sistema de controle ótimo. Tal proposta é baseada no estudo do comportamento de convergência da aproximação de ADHDP em termos da condição de excitação persistente, consistência e polarização do estimador RLS quando associado com métodos de programação dinâmica heurística dependente de ação para solução *online* do problema do DLQR.

Nesse contexto, as melhorias são no sentido de se obter soluções numericamente mais estáveis da equação HJB-*Riccati* associada ao problema de controle ótimo DLQR, que tem por base uma estrutura paramétrica RLS da função valor e métodos de fatoração, tais como fatoração $\mathcal{QR}$ e UDU^T.

O objetivo principal é investigar problemas relacionados ao mal condicionamento da matriz de covariância/autocorrelação da abordagem RLS para aproximação *online* da solução da equação HJB-*Riccati* associada ao problema do regulador linear quadrático discreto formulado no âmbito da programação dinâmica aproximada e aprendizagem por reforço ator-crítico. Métodos de fatoração, tais como $\mathcal{QR}$ e UDU^T, são inseridos visando a melhoria no processo de estimação RLS da solução da Equação HJB-*Riccati*, de maneira a contornar problemas de convergência e estabilidade numérica.

1.4 Organização da Dissertação

Esta dissertação está estruturada da seguinte maneira.

No Capítulo 2 são apresentados alguns fundamentos teóricos relacionados à aprendizagem por reforço, a qual é caracterizada como um mecanismo que engloba processos de decisão markovianos, programação dinâmica, esquemas de iteração de política e diferenças temporais. Esses elementos básicos contribuem para a formulação do problema e sua associação com a equação de *Bellman* para desenvolver os esquemas de iteração de política aproximada e parametrização de PDM.

No Capítulo 3, apresenta-se o desenvolvimento de esquemas ADHDP para determinar a política de controle ótima para o problema DLQR que está relacionado com a solução da equação HJB-*Riccati*. O Capítulo 4 é onde se explicita as principais contribuições do trabalho, ou seja, a proposta de uma metodologia de projeto para sistemas de controle ótimo online baseada em ADP e o desenvolvimento matemático necessário para o projeto dos algoritmos resultantes dessa metodologia, nominalmente RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR.

No Capítulo 5, apresenta-se a Complexidade Computacional bem como alguns méto-

dos para avaliação da mesma, como: Custo Computacional, Estabilidade Numérica e Tempo de Convergência. Na sequência, testes computacionais são apresentados no Capítulo 6, verificando-se a melhoria dos algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR desenvolvidos ao usar decomposição $\mathcal{QR}$ e fatoração UDU^T. O desempenho computacional é avaliado a nível de estabilidade numérica desses algoritmos.

No Capítulo 7, apresentam-se as conclusões sobre as metodologias abordadas para aproximação da solução da equação HJB-*Riccati* e sugestões de trabalhos futuros. Por fim, os Apêndices são voltados para complementar a fundamentação das abordagens utilizadas.

2

FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução

A aprendizagem por reforço refere-se ao problema de um agente interagindo com um ambiente incerto, tendo por base os organismos vivos que interagem com o ambiente e usam estas interações para melhorar suas ações e sobreviverem. As modificações das ações baseadas nas interações, é o que caracteriza-se como Aprendizagem por Reforço (RL). O objetivo da aprendizagem é maximizar uma recompensa escalar a longo prazo, ao invés de simplesmente uma recompensa imediata. Isso ocorre pela percepção do estado do ambiente e tomada de decisões que afetarão este estado.

Nesse quadro, um sistema RL recebe uma resposta sobre o desempenho da sua ação, permitindo-lhe melhorar o desempenho das ações subsequentes. O sistema é então projetado para aprender por reforço atrasado, isto é, o sistema observa uma sequência temporal de estímulos, correspondente aos vetores de estado, advindos também do ambiente, que resultaram num sinal de reforço heurístico. É relevante destacar que certas ações tomadas em momentos anteriores, numa sequência de passos de tempo, e são os melhores determinantes do comportamento global do sistema (HAYKIN, 2008).

Em RL, diversos métodos têm sido desenvolvidos e aplicados. Os algoritmos são construídos com a ideia de que decisões sucessivas de controle são tomadas com a intenção de maximizar o reforço ao longo do tempo, em oposição a uma simples recompensa imediata, mantendo a estabilidade do sistema.

O presente capítulo tem por propósito apresentar, de forma introdutória, os fundamentos teóricos relacionados à aprendizagem por reforço considerando uma abordagem genérica. Os conceitos e métodos são apresentados no contexto da formulação de *Bellman* de programação dinâmica. Define-se formalmente o problema de aprendizagem por reforço em termos de um processo de decisão *Markoviano*.

Em RL, não há o conhecimento da saída correta; o agente apenas recebe alguma informação do ambiente por um reforço, uma penalidade (sinal RL negativo) ou recompensa (sinal RL positivo) baseada nos estados e ações de controle. A tarefa de aprendizagem do agente é encontrar uma política para seleção de ação que maximize sua recompensa a longo prazo. Esse processo busca a escolha de ações que levem o agente às partes mais lucrativas do espaço de estado, indo além das altas recompensas associadas ao estado atual.

Na abordagem de controle, a recompensa equivale a diminuição de custo de controle, enquanto penalidade implica em aumento de custo de controle. A interação do agente RL é caracterizada pelo sinal de estado, sinal de controle e o sinal de recompensa. O sinal de estado descreve o estado do ambiente. Por meio do sinal de controle, o agente RL influencia seu ambiente. O sinal de recompensa fornece realimentação do resultado positivo ou negativo da ação tomada (desempenho do agente).

Os principais elementos de um problema de aprendizagem por reforço são:

- Política (decisão), que é definida como o comportamento de um agente em um instante de tempo particular. Ela é uma parte muito importante dos esquemas de RL e pode ser representada por uma tabela de consulta ou uma função. A política pode ser estocástica ou determinística. Em alguns esquemas RL, um processo de busca complexo é empregado no cálculo da política (geralmente, é calculada por maximização da função valor). O processo RL levará a uma política ótima (comportamento) ao longo do tempo.

- Função de Utilidade, definida sobre a base do objetivo do problema de aprendizagem por reforço, fornece a recompensa que avalia o desempenho imediato do agente por cada tomada de decisão. O agente sempre tenta otimizar o desempenho a longo prazo, medido pela sua recompensa acumulada ao longo da interação e a partir daí, produzir uma política ótima ao longo do tempo. Um fator de desconto pode ser usado para estabelecer a preferência por recompensa imediata ou recompensa orientada mais para o futuro.

- Uma função valor, geralmente representada por $V(x)$, que é uma predição da soma esperada em longo prazo de recompensas descontadas num tempo futuro quando o processo inicia-se em um estado x e segue uma política π. Tal função é a base sobre a qual o agente antecipa a tomada de uma ação que produzirá recompensas maiores. Existe uma abordagem, comumente usada para avaliar uma política objetivo quando o modelo do ambiente não está disponível, que consiste de uma função valor dependente de ação (ou função Q).

Esta função é uma predição de recompensas descontadas totalmente esperadas e associadas com pares estado-ação (x, u) iniciais, normalmente representada por $Q(x, u)$. Como explicado em (SUTTON; BARTO, 1998), uma função de utilidade avalia a recompensa imediata enquanto a função valor é a predição da recompensa futura total. Daí, a tomada de decisão é realizada sobre a base da função valor de estado ou ação.

Quanto à característica dual do sinal de reforço, sobre a necessidade de se obter o máximo em recompensa ou mínimo em custo, trata-se de um problema de otimização, cuja solução é necessária para se implementar o projeto do controlador. Para tal projeto, a otimalidade dos controladores é garantida por meio da equação de *Hamilton-Jacobi-Bellman*, tendo-se a possibilidade de estabelecer um caminho de trajetórias ótimas através da Programação Dinâmica.

2.2 Programação Dinâmica

A metodologia conhecida como programação dinâmica lida com situações em que as decisões são feitas por etapas. Para cada decisão, o resultado pode ser previsto até determinado ponto, antes de ser tomada a próxima decisão. Objetiva-se dessa maneira minimizar um determinado custo de uma função, definida por sua expressão matemática. O relevante em tais situações é que as decisões não podem ser tomadas isoladamente, uma vez que deve-se ponderar o desejo de um baixo custo no presente em relação a altos custos indesejáveis no futuro (HAYKIN, 2008). Em cada fase, classificam-se as decisões com base na soma do custo instantâneo e do custo futuro esperado, assumindo que a decisão ótima é aplicada nas etapas subsequentes. Este procedimento baseia-se no princípio da otimalidade de *Bellman*, para o qual, "uma estratégia ótima apresenta a propriedade segundo a qual, independente das decisões tomadas para se alcançar um estado particular num certo estágio, as decisões restantes a partir deste estado devem constituir uma estratégia ótima".

A programação dinâmica, portanto, tem como quesito elementar o interesse em buscar a resposta de como um sistema pode aprender a melhorar o seu desempenho a longo prazo, quando isto pode exigir o sacrifício do desempenho atual. Um problema relacionado com as técnicas de DP é que sua complexidade e o custo de executá-las cresce de maneira proporcional ao número de estados que, por sua vez, aumenta exponencialmente com a dimensionalidade do espaço de estado, acarretando no que conhece-se por "*maldição da dimensionalidade*".

No panorama de programação dinâmica/aprendizagem por reforço, há a possibilidade de superar tal limitação e assim solucionar problemas de uma variedade de campos, incluindo,

por exemplo, controle automático, inteligência artificial, pesquisa operacional e economia. O controle automático e inteligência artificial são, indiscutivelmente, os campos mais importantes de origem para a DP e RL. No controle automático, a DP pode ser utilizada para resolver problemas não-lineares e também problemas de controle ótimo estocásticos (BERTSEKAS, 1987), enquanto que a RL pode, alternativamente, ser vista como uma forma de controle ótimo adaptativo (VRABIE; LEWIS, 2008). Em Inteligência Artificial, a RL auxilia na construção de um agente artificial que aprende para sobreviver e otimizar seu comportamento em um ambiente desconhecido, sem exigir conhecimento prévio (SUTTON; BARTO, 1998). São nomenclaturas equivalentes usadas em programação dinâmica e aprendizagem por reforço respectivamente, "controlador" e "agente" assim como "processo" e "ambiente".

Nessas condições as abordagens de DP/RL para solução de problemas de controle, tem como objetivo encontrar uma política ótima que minimize o retorno esperado, que consiste na recompensa esperada que acumula-se ao longo da interação. Neste capítulo, considera-se principalmente retornos de horizonte infinito, que acumulam recompensas ao longo de trajetórias extensivamente longas. Esta escolha é feita porque retornos de horizonte infinito resultam em políticas ótimas estacionárias, o que significa que, para um determinado estado, a ação ótima escolhida será sempre a mesma, independentemente do momento em que o estado se encontra.

2.3 Processo de Decisão *Markoviano*

Problemas de programação dinâmica e aprendizagem por reforço podem ser formulados no quadro de Processos de Decisão *Markovianos* – PDM, onde as transições entre os estados são probabilísticas. Cada ação tem um custo/recompensa, que dependerá do estado em que o processo se encontra. Um PDM é definido como sendo uma quadrupla (X, U, P, R), onde: X corresponde ao conjunto de estados; U ao conjunto de ações; $P : X \times U \times X \rightarrow [0, 1]$ descreve, para cada estado $x \in X$ e ação $u \in U$, a probabilidade condicional $P^u_{x,x'} = \Pr\{x'|x, u\}$[1] de transição para o estado seguinte x' dado que o PDM está no estado x e toma-se a ação $u \in U$ e $R : X \times U \times X \rightarrow \mathbb{R}$ é a função de custo que fornece o custo imediato esperado incorrido da transição para o estado x' dado que o PDM inicia-se no estado x e toma-se a ação $u \in U$.

Em um Processo de Decisão *Markoviano*, as transições do estado x para o estado seguinte x' dependem apenas das ações permitidas no estado x, tal característica corresponde a

[1] Probabilidade condicional refere-se à probabilidade de um evento ocorrer com base em um evento anterior. Dessa forma, a probabilidade de x' acontecer, dado que (x, u) já aconteceu, é representada por $\Pr\{x'|x, u\}$.

propriedade de Markov, assim a função de transição de probabilidade $P^u_{x,x'}$ depende apenas do estado atual do sistema e não do histórico de como o PDM alcançou este estado.

Pode-se caracterizar um PDM como:

- um ambiente que evolui probabilisticamente de acordo com um conjunto finito e discreto de estados;
- para cada estado do ambiente, existe um conjunto finito de ações possíveis;
- a cada transição o agente recebe um retorno (custo) do ambiente em relação a ação tomada;
- estados são observados, ações são executadas e reforços são relacionados em tempo discreto.

Em um PDM, o problema básico reside na determinação de um mapeamento $\pi : X \times U \rightarrow [0, 1]$, que indica, para cada estado x e ação u, a probabilidade condicional $\pi(x, u) = Pr\{u|x\}$ para tomada de ação u dado que o PDM está no estado x, denominado de *política de controle*. Se o mapeamento $\pi : X \times U \rightarrow [0, 1]$ admitir apenas um controle, com probabilidade igual a um, em cada estado, então esse mapeamento corresponde a uma política determinística.

Considerando-se problemas de decisão sequencial, onde os sistemas dinâmicos evoluem casualmente através do tempo, impõem-se um índice de estágio discreto k tal que o PDM toma uma ação de controle e altera os estados por valores de estágio k inteiros não negativos, que correspondem geralmente a sequências de eventos. Os valores de estado e ações de controle são denotadas por $x_k,\ u_k$.

Para sistemas onde se deseja frequentemente a conservação de recursos como custos, tempo, dentre outros, a noção de otimalidade é utilizada na seleção de políticas de controle para os PDMs. Define-se o *custo de estágio* no tempo k por $r_k = r_k(x_k, u_k, x_{k+1})$. Assim, $R^u_{xx'} = E\{r_k|x_k = x, u_k = u, x_{k+1} = x'\}$, sendo $E\{\cdot\}$ o operador de valor esperado. O índice de desempenho é definido como um somatório de custos futuros sobre o intervalo de tempo $[k,\ k + T]$,

$$J_{k,T} = \sum_{i=0}^{T} \gamma^i r_{k+i} = \sum_{i=k}^{k+T} \gamma^{i-k} r_i, \tag{2.1}$$

onde γ é um escalar com $0 \leq \gamma < 1$ chamado de fator de desconto. Os ajustes em γ possibilitam reduzir o peso com que o sistema trata suas ações a longo prazo em detrimento das ações de curto prazo. De forma que, quanto mais próximo de 0 for o valor do fator de desconto, maior

será a importância que o agente dará para as recompensas imediatas. Por outro lado, a medida que γ se aproxima de 1 o agente passa a considerar recompensas futuras mais fortemente.

Suponha-se que o agente selecione uma política de controle $\pi_k(x_k, u_k)$ que será usada a cada estágio k do PDM. Considera-se que a política escolhida seja uma política estacionária fixa, da forma $\pi = \{\mu, \mu, \dots\}$. Então $\pi(x, u) = \Pr\{u|x\}$ e o PDM reduz-se a uma cadeia de *Markov* sobre o espaço de estado X, uma vez que a política estacionária especifica exatamente a mesma ação cada vez que um estado particular é visitado. As probabilidades de transição dessa cadeia de *Markov* são dadas por

$$p_{x,x'} \equiv P^{\pi}_{x,x'} = \sum_{u} Pr\{x'|x,u\}Pr\{u|x\} = \sum_{u} \pi(x,u)P^{u}_{x,x'}, \tag{2.2}$$

utilizando-se para isso a identidade de Chapman-Kolmogorov (PAPOULIS; PILLAI, 2002).

Sob a hipótese de ergodicidade da cadeia de *Markov*, que implica em todos os estados serem positivos, recorrentes e aperiódicos, é possível mostrar que o PDM tem uma política ótima estacionária e determinística. Para mais considerações consultar (BERTSEKAS, 1987).

O valor de uma política é definido como o valor esperado condicionalmente de custos futuros, dado que o processo inicia-se no estado x no instante k e segue-se a política $\pi(x, u)$. Em linguagem matemática, isto significa

$$V^{\pi}_{k}(x) = E_{\pi}\{J_{k,T}|x_k = x\} = E_{\pi}\left\{\sum_{i=k}^{k+T} \gamma^{i-k} r_i | x_k = x\right\}, \tag{2.3}$$

onde $V^{\pi}(x)$ é denominada função valor de estado e consiste do mapeamento do estado, tendo por solução um valor que é obtido a partir do reforço atual e dos reforços futuros.

O principal objetivo de um PDM consiste em determinar uma política $\pi(x, u)$ para minimizar a função valor

$$\pi^*(x,u) \quad = \quad \arg\min_{\pi} V^{\pi}_{k}(x) = \arg\min_{\pi} E_{\pi}\left\{\sum_{i=k}^{k+T} \gamma^{i-k} r_i | x_k = x\right\}, \tag{2.4}$$

sendo essa denominada de *política ótima*, e a função valor ótima correspondente é expressa por

$$V^{*}_{k}(x) = \min_{\pi} V^{\pi}_{k}(x) = \min_{\pi} E_{\pi}\left\{\sum_{i=k}^{k+T} \gamma^{i-k} r_i | x_k = x\right\}. \tag{2.5}$$

Usando a identidade de Chapman-Kolmogorov e a propriedade de *Markov*, a função valor da

política $\pi(x,u)$ pode ser escrita como

$$V_k^{\pi}(x) = E_{\pi}\{J_k|x_k = x\} = E_{\pi}\left\{\sum_{i=k}^{k+T}\gamma^{i-k}r_i|x_k = x\right\}, \tag{2.6}$$

$$V_k^{\pi}(x) = E_{\pi}\left\{r_k + \gamma\sum_{i=k+1}^{k+T}\gamma^{i-(k+1)}r_i|x_k = x\right\}, \tag{2.7}$$

$$V_k^{\pi}(x) = \sum_u \pi(x,u)\sum_{x'} P_{xx'}^u\left[R_{xx'}^u + \gamma E_{\pi}\left\{\sum_{i=k+1}^{k+T}\gamma^{i-(k+1)}r_i|x_{k+1} = x'\right\}\right]. \tag{2.8}$$

Portanto, a função valor para a política $\pi(x,u)$ corresponde a

$$V_k^{\pi}(x) = \sum_u \pi(x,u)\sum_{x'} P_{xx'}^u\left[R_{xx'}^u + \gamma V_{k+1}^{\pi}(x')\right], \tag{2.9}$$

sendo esta uma recursão para "trás no tempo" da função valor no instante k em termos do valor esperado para o instante $k+1$.

O custo ótimo pode ser expresso como

$$V_k^*(x) = \min_{\pi} V_k^{\pi}(x) = \min_{\pi}\sum_u \pi(x,u)\sum_{x'} P_{xx'}^u\left[R_{xx'}^u + \gamma V_{k+1}^{\pi}(x')\right]. \tag{2.10}$$

Pelo Princípio da Otimalidade de *Bellman*, segundo o qual "uma política ótima tem a propriedade de que não importa quais ações de controle tenham sido tomadas anteriormente, as decisões restantes constituem uma política ótima no que diz respeito ao estado resultante das ações anteriores". Com base neste princípio, a Eq.(2.10) pode ser escrita como

$$V_k^*(x) = \min_{\pi}\sum_u \pi(x,u)\sum_{x'} P_{xx'}^u\left[R_{xx'}^u + \gamma V_{k+1}^*(x')\right]. \tag{2.11}$$

Supondo que um controle arbitrário u seja aplicado no instante k e a política ótima π^* seja aplicada a partir do instante $k+1$ em diante, pelo princípio da otimalidade de *Bellman*, tem-se

$$\pi^*(x,u) = \arg\min_{\pi}\sum_u \pi(x,u)\sum_{x'} P_{xx'}^u[R_{xx'}^u + \gamma V_{k+1}^*(x')]. \tag{2.12}$$

Sob a hipótese de que a cadeia de *Markov* correspondente a cada política seja ergódica,

pode-se minimizar a esperança condicional sobre todas as ações u no estado x, obtendo-se:

$$V_k^*(x) = \min_u \sum_{x'} P_{xx'}^u \left[R_{xx'}^u + \gamma V_{k+1}^*(x') \right], \tag{2.13}$$

$$u_k^* = \arg\min_u \sum_{x'} P_{xx'}^u \left[R_{xx'}^u + \gamma V_{k+1}^*(x') \right]. \tag{2.14}$$

As recursões dadas pelas Eqs.(2.11) e (2.13) formam a base para a programação dinâmica, que fornecem métodos *offline* funcionando "para trás no tempo" afim de determinar políticas ótimas.

2.4 Equação de *Bellman*

A programação dinâmica consiste de um método *offline* que procura determinar o valor ótimo e a política ótima fundamentando-se no princípio da otimalidade de *Bellman*. A metodologia da aprendizagem por reforço, por outro lado, busca encontrar políticas ótimas baseadas em experiência, executando decisões sequenciais que melhoram as ações de controle com base nos resultados observados utilizando a política atual. Tal procedimento resulta na construção de métodos para encontrar valores ótimos e políticas ótimas que podem ser executados "para frente no tempo".

Na sua forma primária, o algoritmo de programação dinâmica trata de um problema de horizonte finito. Estamos interessados em estender o uso deste algoritmo para tratar do problema de horizonte infinito descrito pela função de custo. Para formular este método em termos matemáticos, considere o custo de horizonte infinito definido por

$$J_k = \sum_{i=0}^{\infty} \gamma^i r_{k+i} = \sum_{i=k}^{\infty} \gamma^{i-k} r_i. \tag{2.15}$$

A Função Valor de horizonte infinito associada à política $\pi(x, u)$ é

$$V^\pi(x) = E_\pi\{J_k | x_k = x\} = E_\pi \left\{ \sum_{i=k}^{\infty} \gamma^{i-k} r_i | x_k = x \right\}. \tag{2.16}$$

Usando a Eq.(2.8) com $T = \infty$, tem-se que a função valor para a política $\pi(x, u)$ satisfaz a

Equação de *Bellman* dada por

$$V^{\pi}(x) = \sum_{u} \pi(x,u) \sum_{x'} P^{u}_{xx'}[R^{u}_{xx'} + \gamma V^{\pi}(x')]. \tag{2.17}$$

A equação de *Bellman* é o ponto de partida para a construção de uma família de algoritmos de aprendizagem por reforço, que buscam encontrar políticas ótimas *online* em tempo real para a solução de problemas de controle. Um exemplo disso encontra-se na aproximação da função valor dada na Eq.(2.17) para uma estrutura paramétrica, podendo ser implementado *online* usando-se um algoritmo de identificação de sistema de controle adaptativo, como por exemplo os mínimos quadrados recursivos (RLS).

Para um PDM finito com N estados, a equação de *Bellman* (2.17) representa um sistema de N equações lineares simultâneas para a função valor $V^{\pi}(x)$ em cada estado x dada a política atual $\pi(x,u)$.

O Valor Ótimo satisfaz

$$V^{*}(x) \;=\; \min_{\pi} V^{\pi}(x) = \min_{\pi} \sum_{u} \pi(x,u) \sum_{x'} P^{u}_{xx'}[R^{u}_{xx'} + \gamma V^{\pi}(x')]. \tag{2.18}$$

O princípio da otimalidade resulta na Equação da Otimalidade de *Bellman*,

$$V^{*}(x) \;=\; \min_{\pi} V^{\pi}(x) = \min_{\pi} \sum_{u} \pi(x,u) \sum_{x'} P^{u}_{xx'}[R^{u}_{xx'} + \gamma V^{*}(x')]. \tag{2.19}$$

Sob a hipótese de ergodicidade sobre as cadeias de *Markov* correspondentes a cada política, a Eq.(2.19) pode ser escrita como

$$V^{*}(x) = \min_{u} \sum_{x'} P^{u}_{xx'}[R^{u}_{xx'} + \gamma V^{*}(x')]. \tag{2.20}$$

Esta equação é conhecida como Equação HJB, e o controle ótimo é expresso por

$$u^{*} = \arg\min_{u} \sum_{x'} P^{u}_{xx'}[R^{u}_{xx'} + \gamma V^{*}(x')]. \tag{2.21}$$

2.5 Avaliação de Política e Melhoria de Política

Seja uma política atual $\pi(x,u)$, a função valor correspondente é determinada via solução da Eq.(2.17), processo denominado de *avaliação de política*. Entretanto, é perfeitamente

possível se determinar uma outra política que seja melhor, ou no mínimo igual a política atual; tal procedimento é descrito como *melhoria de politica*. Define-se, então, uma nova política por

$$\pi'(x,u) = \arg\min_{\pi} \sum_{x'} P^{u}_{xx'}[R^{u}_{xx'} + \gamma V^{\pi}(x')]. \tag{2.22}$$

Este procedimento é descrito por Howard (1960) que prova que a nova política $\pi'(x,u)$ é melhorada com respeito à política anterior $\pi(x,u)$ no sentido que $V^{\pi'}(x) \leq V^{\pi}(x)$.

Os algoritmos que intercalam repetidamente os dois processos são:

Avaliação de Política

$$V^{\pi}(x) = \sum_{u} \pi(x,u) \sum_{x'} P^{u}_{xx'}[R^{u}_{xx'} + \gamma V^{\pi}(x')], \text{ para todo } x \in \mathcal{S} \subseteq X. \tag{2.23}$$

Melhoria de Política

$$\pi'(x,u) = \arg\min_{\pi} \sum_{x'} P^{u}_{xx'}[R^{u}_{xx'} + \gamma V^{\pi}(x')], \quad \text{para todo } x \in \mathcal{S} \subseteq X. \tag{2.24}$$

sendo $\mathcal{S}$ um subespaço do espaço de estados. A realização desses dois procedimentos, respectivamente, é composta de passo, e a cada um deles uma política é obtida, sendo essa um pouco melhor ou igual a política utilizada anteriormente. Tais processos podem ser implementados em tempo real via observação dos dados medidos ao longo das trajetórias do sistema.

O sistema necessita portanto de uma grande capacidade computacional para que os custos das transições estejam disponíveis para o agente, que é o subsistema responsável por implementar as políticas ótimas.

2.5.1 Iteração de Política

A iteração de política consiste de um processo iterativo de busca de política ótima π^*, em que a cada iteração do processo, realiza-se uma etapa de obtenção da função valor para a política atual, denominada *avaliação de política*, sendo que tal etapa funciona em $loop$ interno para cada iteração de política. Em seguida, uma nova política é obtida de modo melhorado, ou pelo menos igual à política atual, sendo essa etapa denominada de *melhoria de política*. Na busca da política ótima $\pi^*(x,u)$, realiza-se uma etapa de avaliação da política atual $\pi(x,u)$ e

obtêm-se a função valor correspondente a tal política

$$V_j(x) = \sum_u \pi_j(x,u) \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma V_j(x')], \text{ para todo } x \in X, \tag{2.25}$$

a cada passo j o algoritmo determina a solução da equação de *Bellman* (2.25) para o cálculo da função valor $V_j(x)$ usando a política atual $\pi_j(x,u)$, com funcionamento em *loop* interno. Em seguida, realiza-se a etapa de melhoria de política, onde será estabelecida uma nova política de controle π_{j+1}, expressa

$$\pi_{j+1}(x,u) = \arg\min_{\pi} \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma V_j(x')], \text{ para todo } x \in X. \tag{2.26}$$

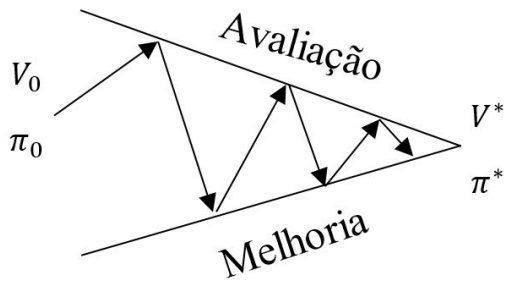
Sendo a equação de *Bellman* (2.25) um sistema de equações lineares simultâneas, ao invés de resolver diretamente tal equação, pode-se solucionar o problema por um procedimento conhecido como "avaliação de política iterativa". Define-se um mapeamento de avaliação de política iterativa por

$$V^{i+1}_j(x) = \sum_u \pi_j(x,u) \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma V^i_j(x')], \; i = 1, 2, ..., \tag{2.27}$$

onde o índice j em (2.27) refere-se ao número do passo do algoritmo da iteração de política, sendo i apenas um índice iterativo dentro do algoritmo.

Se as etapas de avaliação e melhoria de políticas são realizadas corretamente, o processo convergirá para π^* após vários ciclos de operação, sob uma política inicial admissível; essa ideia é representada graficamente na Fig.2.1. Em (BRADTKE; YDSTIE; BARTO, 1994) mostra-se que o algoritmo converge quando as Eqs.(2.26) e (2.27) forem satisfeitas.

Figura 2.1: Convergência na iteração de política.



2.5.2 Iteração Gulosa

Na estratégia IP padrão a política atual π é fixada possivelmente por um longo período de tempo até que a convergência do crítico seja alcançada, a fim de obter a melhoria de política. Existe uma variação do método IP em que as melhorias de políticas são realizadas usando-se um procedimento conhecido como *bootstrapping*, (LANDELIUS, 1997), em que a política atual é considerada ser sempre a política ótima. A busca é realizada em um único $loop$ por cada iteração de modo que a estimativa parametrizada da função valor, $\hat{V}^*$, é atualizada de acordo com a equação da otimalidade de *Bellman*, Eq.(2.19), que é dada por

$$\hat{V}^*(x) = \sum_u \pi^*(x,u) \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma \hat{V}^*(x')], \text{ para todo } x \in X, \tag{2.28}$$

$$\hat{\pi}^*(x,u) = \arg\min_{\pi} \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma \hat{V}^*(x')], \text{ para todo } x \in X. \tag{2.29}$$

Daí porque o método é também chamado *certainty equivalence control*, uma vez que a estimativa atual da função valor ótima é tratada como se ela fosse a estimativa ótima na obtenção da nova estimativa da política ótima. Neste caso as Eqs.(2.28) e (2.29) são avaliadas na ordem inversa pois o foco é sobre a política parametrizada $\hat{\pi}^*(x,\theta)$, sendo θ o vetor de parâmetros da representação parametrizada da função valor. O método é, em muitos aspectos, similar à iteração de política otimística estabelecida por (BERTSEKAS; TSITSIKLIS, 1995) que atualiza a política atual após cada transição de estado x_k para x_{k+1}.

2.5.3 Iteração de Valor

Um outro método de busca de políticas e funções valor ótimas em aprendizagem por reforço é a iteração de valor. Este usa a equação de otimalidade de *Bellman* para calcular iterativamente uma função valor ótima, que é usada para obter uma política ótima.

Os passos que compõem o algoritmo de iteração de valor iniciam-se com a seleção de uma política $\pi_0(x,u)$, com $j = 0$, iterando-se j até sua convergência. A atualização de valor é realizada de acordo com

$$V_{j+1}(x) = \sum_u \pi_j(x,u) \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma V_j(x')], \text{ para todo } x \in \mathcal{S}_j \subseteq X, \tag{2.30}$$

seguida da melhoria de política

$$\pi_{j+1}(x,u) = \arg\min_{\pi} \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma V_{j+1}(x')], \text{ para todo } x \in \mathcal{S}_j \subseteq X. \tag{2.31}$$

Esse método combina ambas as etapas, atualização de valor e melhoria de política, na seguinte equação

$$V_{j+1}(x) = \min_{\pi} \sum_{u} \pi(x,u) \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma V_j(x')], \text{ para todo } x \in \mathcal{S}_j \subseteq X, \tag{2.32}$$

na qual ocorrerá a atualização das estimativas da função valor, para todos os estados de x. De maneira equivalente, sob a suposição de ergodicidade, para uma política determinística tem-se

$$V_{j+1}(x) = \min_{u} \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma V_j(x')], \text{ para todo } x \in \mathcal{S}_j \subseteq X. \tag{2.33}$$

A iteração de valor tem sua convergência garantida para uma função que satisfaça a equação da otimalidade de *Bellman* (2.20) com base em uma representação tabular da função valor (BERTSEKAS, 1999).

2.6 Função Q

A função Q, ou função valor de ação, para uma política fixada $\pi(x,u)$ é definida por $Q^\pi : X \times U \to \mathbb{R}$

$$\begin{aligned} Q^\pi_k(x,u) &= E_\pi\{r_k + \gamma V^\pi_{k+1}(x') | x_k = x, u_k = u\}, \\ &= \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma V^\pi_{k+1}(x')]. \end{aligned} \tag{2.34}$$

Uma vez que $V^\pi_k(x) = Q^\pi_k(x, \pi(x,u))$, a Eq.(2.34) pode ser expressa como uma recurção "para trás no tempo" na função Q

$$Q^\pi_k(x,u) = \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma Q^\pi_{k+1}(x', \pi(x',u')). \tag{2.35}$$

A função Q ótima é definida por

$$\begin{aligned} Q_k^*(x,u) &= \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V_{k+1}^*(x')], \\ &= E_\pi\{r_k + \gamma V_{k+1}^*(x') | x_k = x, u_k = u\}. \end{aligned} \tag{2.36}$$

Em termos da função Q, a equação da otimalidade de *Bellman* é dada por

$$V_k^*(x) = \min_u Q_k^*(x,u), \tag{2.37}$$

$$u_k^*(x) = \arg\min_u Q_k^*(x,u). \tag{2.38}$$

A função Q é uma função tanto do estado atual x quanto da ação de controle u. Em contraste, a função valor de estado é uma função somente do estado. Para o PDM finito, a função Q pode ser armazenada como uma tabela de consulta para cada par de estado/ação. Percebe-se que a minimização direta nas Eqs.(2.11)-(2.12) requer conhecimento das probabilidades de transição de estado, que correspondem à dinâmica do sistema. O que não é necessário na minimização das Eqs.(2.37)-(2.38), para ambas é essencial apenas o conhecimento da função Q e não o do modelo da dinâmica do sistema.

Indo além, por conter informações sobre as ações de controle em cada estado, um melhor controle em cada estado pode ser selecionado usando a Eq.(2.38) conhecendo-se apenas a função Q.

A função Q de horizonte infinito para uma tal política fixa prescrita é dada por

$$Q^\pi(x,u) = \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma V^\pi(x')], \tag{2.39}$$

assim, $V^\pi(x) = Q^\pi(x, \pi(x,u))$ e, de acordo com Eq.(2.39), a função Q satisfaz a equação de *Bellman*:

$$Q^\pi(x,u) = \sum_{x'} P_{xx'}^u [R_{xx'}^u + \gamma Q^\pi(x', \pi(x',u'))]. \tag{2.40}$$

A equação da otimalidade de *Bellman* para a função Q é dada por,

$$Q^*(x,u) = \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma Q^*(x', \pi^*(x', u'))], \tag{2.41}$$

$$Q^*(x,u) = \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma \min_{u'} Q^*(x', u')]. \tag{2.42}$$

Dentre as vantagens inerentes a função Q, pelo fato de a função Q depender também da ação, ela já inclui informação sobre a qualidade das transições. Em contraste, a função valor de estado V somente descreve a qualidade dos estados. A função Q também pode ser estimada *online* "em tempo real" a partir dos dados observados ao longo da trajetória do sistema sem o conhecimento do modelo da dinâmica do sistema. Isso pode ser visto nos algoritmos de iteração de política e iteração de valor implementados em termos da função Q.

Iteração de Política usando a Função Q

Avaliação de Política

$$Q_j(x,u) = \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma Q_j(x', \pi_j(x', u'))], \quad \text{para todo } x \in X. \tag{2.43}$$

Melhoria de Política

$$\pi_{j+1}(x,u) = \arg\min_u Q_j(x,u), \text{ para todo } x \in X. \tag{2.44}$$

Iteração de Valor usando a Função Q

Atualização de Valor

$$Q_{j+1}(x,u) = \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma Q_j(x', \pi_j(x', u'))], \quad \text{para todo } x \in \mathcal{S}_j \subseteq X. \tag{2.45}$$

Melhoria de Política

$$\pi_{j+1}(x,u) = \arg\min_u Q_{j+1}(x,u), \quad \text{para todo } x \in \mathcal{S}_j \subseteq X. \tag{2.46}$$

Combinando ambas as etapas da iteração de valor, produz-se a forma

$$Q_{j+1}(x,u) = \sum_{x'} P^u_{xx'}[R^u_{xx'} + \gamma \min_{u'} Q_j(x',u')],$$
$$\text{para todo } x \in \mathcal{S}_j \subseteq X. \tag{2.47}$$

2.7 Diferenças Temporais

Métodos de diferenças temporais (do inglês *Temporal Difference* - TD) para solucionar a equação de *Bellman* conduzem a uma família de controladores ótimos adaptativos. Tais controladores são capazes de aprender *online* a solução de problemas de controle ótimo quando o modelo explícito do sistema não está disponível. Os métodos TD tem sua origem atrelada a problemas de atribuição de crédito e ao estudo de algoritmos de aprendizagem por reforço (ANDERSON, 1987; BARTO; SUTTON; ANDERSON, 1983; SUTTON, 1984, 1988).

2.7.1 Aprendizagem por Diferenças Temporais ao Longo das Trajetórias dos Estados

Métodos de aprendizagem por reforço de diferenças temporais são baseados na equação de *Bellman* para resolver as Eqs.(2.25) e (2.30) sem usar o conhecimento completo da dinâmica do sistema. Esses métodos podem ser considerados como técnicas de Aproximação Estocástica (BERTSEKAS, 1987), onde a equação de *Bellman* (2.17) ou suas variantes, são substituídas por avaliações ao longo de um caminho amostrado de um PDM.

A Eq.(2.9) é usada para escrever a equação de Belman (2.17) para o valor de horizonte infinito (2.16). De acordo com as Eqs.(2.7)-(2.9), uma forma alternativa para a equação de *Bellman* é

$$V^\pi(x_k) = E_\pi\{r_k|x_k\} + \gamma E_\pi\{V^\pi(x_{k+1})|x_k\}. \tag{2.48}$$

Então a Eq.(2.48) é substituída pela Equação de *Bellman* Determinística

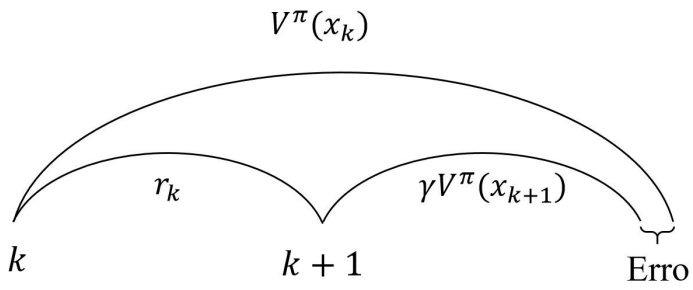
$$V^\pi(x_k) = r_k + \gamma V^\pi(x_{k+1}). \tag{2.49}$$

Assim, tem-se o erro por diferença temporal definido como

$$e_k = -V^\pi(x_k) + r_k + \gamma V^\pi(x_{k+1}), \tag{2.50}$$

onde a estimativa de valor é atualizada para tornar o erro de diferença temporal pequeno.

Figura 2.2: Erro de predição.



Como ilustra a Figura 2.2, o erro de diferenças temporais (DT) pode ser considerado como um erro de predição entre o desempenho predito e o desempenho observado em resposta a uma ação aplicada ao sistema (LEWIS; VRABIE, 2009a).

2.8 Controle Ótimo Adaptativo para Sistemas de Tempo Discreto

Algoritmos para controle ótimo adaptativo determinam a solução da equação *Hamilton-Jacobi* de maneira *online* no tempo sem o conhecimento da dinâmica do sistema. No caso do regulador linear quadrático (do inglês *Linear Quadratic Regulator* - LQR), isso corresponde à solução da equação de *Riccati* pelo algoritmo, sem o conhecimento da matriz A do sistema.

Para uma classe de sistemas de tempo discreto descritos por dinâmicas determinísticas não-lineares na forma de equações a diferença de espaço de estado, considera-se

$$x_{k+1} = f(x_k) + g(x_k)u_k, \tag{2.51}$$

com $x_k \in R^n$, $u_k \in R^m$.

Uma política de controle é definida como $h(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$, isto é,

$$u_k = h(x_k). \tag{2.52}$$

Definindo-se uma função de custo determinística, produz-se a função valor

$$V^h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) = \sum_{i=k}^{\infty} \gamma^{i-k} (Q(x_i) + u_i^T R u_i), \tag{2.53}$$

com $0 < \gamma \leq 1$, $Q(x_k) > 0,\ R > 0$ e $u_k = h(x_k)$.

O custo de transição de estágio (custo imediato) é

$$r_k(x_k, u_k) = Q(x_k) + u_k^T R u_k. \tag{2.54}$$

Toma-se o custo de estágio quadrático em u_k para simplificar o desenvolvimento. Assumindo que o sistema é estável em algum conjunto $\Omega \in R^n$, então existe uma política de controle $u_k = h(x_k)$ tal que o sistema de malha fechada $x_{k+1} = f(x_k) + g(x_k)h(x_k)$ é assintoticamente estável em Ω. Essa política u_k é considerada ser *admissível* se estabilizar o sistema de malha-fechada e produzir um custo finito para $V^h(x_k)$.

Para um valor determinístico em (2.53), o valor ótimo é dado pela equação da otimalidade de *Bellman*,

$$V^*(x_k) = \min_{h(\cdot)}(r(x_k, h(x_k)) + \gamma V^*(x_{k+1})), \tag{2.55}$$

e a política ótima é dada por

$$h^*(x_k) = \arg\min_{h(\cdot)}(r(x_k, h(x_k)) + \gamma V^*(x_{k+1})). \tag{2.56}$$

Assim, a equação de *Bellman* determinística (2.49) é

$$\begin{aligned} V^h(x_k) &= r(x_k, h(x_k)) + \gamma V^h(x_{k+1}), \\ V^h(x_k) &= Q(x_k) + u_k^T R u_k + \gamma V^h(x_{k+1}), \text{ onde } V^h(0) = 0, \end{aligned} \tag{2.57}$$

esta é uma maneira de se determinar a função valor utilizando-se a política atual $u_k = h(x_k)$.

A função Hamiltoniana de tempo discreto pode ser definida como

$$H(x_k, h(x_k), \Delta V_k) = r(x_k, h(x_k)) + \gamma V^h(x_{k+1}) - V^h(x_k), \tag{2.58}$$

sendo $\Delta V_k = \gamma V_h(x_{k+1}) - V_h(x_k)$, um operador a diferenças para frente. Assim, o hamiltoniano é o erro de diferença temporal definido pela Eq.(2.50). A equação de *Bellman* requer que o Hamiltoniano seja igual a zero para o valor associado a uma política prescrita.

2.8.1 Iteração de Política e Iteração de Valor para Sistemas Dinâmicos de Tempo Disctreto

As duas formas de aprendizagem por reforço a seguir são baseadas em iteração de política e iteração de valor para aprendizagem por diferenças temporais.

Iteração de Política usando Aprendizagem por Diferenças Temporais

Início

Seleciona-se uma política de controle admissível $h_0(x_k)$. Inicia-se com $j = 0$, itera-se até convergência.

Avaliação de política

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1}). \tag{2.59}$$

Melhoria de Política

$$h_{j+1}(x_k) = \arg\min_{h(\cdot)}(r(x_k, h(x_k)) + \gamma V_{j+1}(x_{k+1})), \tag{2.60}$$

ou

$$h_{j+1}(x_k) = -\frac{\gamma}{2} R^{-1} g^T(x_k) \nabla V_{j+1}(x_{k+1})), \tag{2.61}$$

onde $\nabla V(x) = \partial V(x)/\partial x$ é o gradiente da função valor.

Iteração de Valor Usando Aprendizagem por Diferenças Temporais

Na iteração de valor qualquer política de controle inicial $h_0(x_k)$ pode ser adotada. Assim, o passo de Atualização de Valor é dado por:

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_j(x_{k+1}). \tag{2.62}$$

Para o LQR de tempo discreto a Iteração de Valor é uma recursão de Lyapunov que converge para a solução da equação algébrica de *Riccati* de tempo discreto (LANCASTER; RODMAN, 1995).

2.8.2 Aproximação de Função Valor

A ideia de Aproximação de Função Valor usada por (WERBOS, 1992), é chamada de Programação Dinâmica Aproximada/Adaptativa (do inglês *Approximate/Adaptive Dynamic*

Programming - ADP).

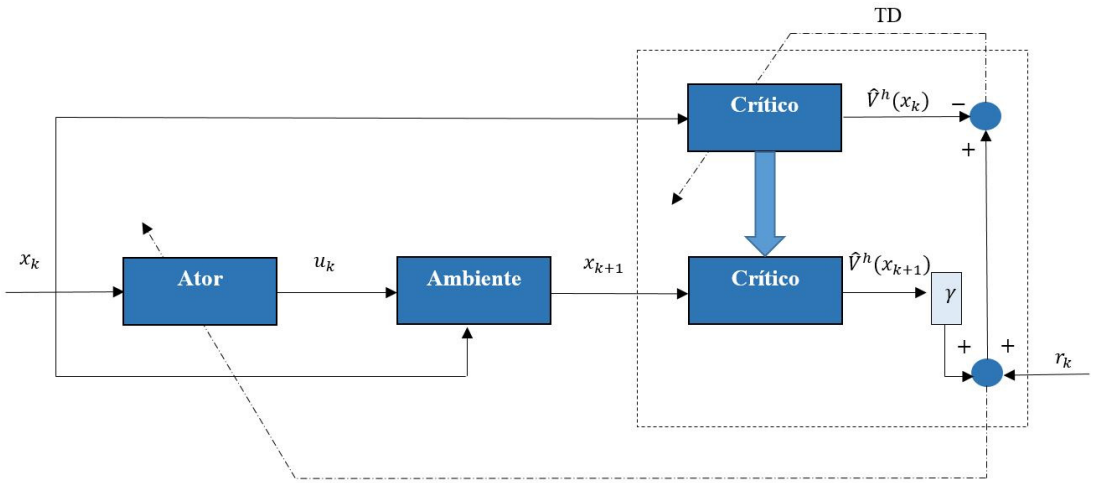
Para sistemas não-lineares descritos pela Eq.(2.51), a função valor contém não-linearidades de alta ordem. Assumindo que a equação de *Bellman* (2.57) tenha solução localmente suave. Então de acordo com o Teorema de Weierstrass para aproximação de alta ordem, existe um conjunto de base denso $\phi_i(x)$, de modo que

$$\begin{aligned} \hat{V}(x) &= \sum_{i=1}^{\infty} \phi_i(x)\theta_i \\ &= \sum_{i=1}^{L} \phi_i(x)\theta_i + \sum_{i=L+1}^{\infty} \phi_i(x)\theta_i \equiv \boldsymbol{\phi}^T(x)\boldsymbol{\theta} + \varepsilon_L(x), \end{aligned} \tag{2.63}$$

sendo o vetor de base $\boldsymbol{\phi}(x) = [\phi_1(x)\ \phi_2(x) \cdots\ \phi_L(x)]^T : R^n \rightarrow R^L$, e $\varepsilon_L(x)$ converge uniformemente para zero quando o número de termos retidos $L \rightarrow \infty$, e θ_i são pesos a se estimar.

A HDP é a estrutura de ADP mais básica e é ilustrada na Figura 2.3. Nesta estrutura o Crítico estima $V^h(x_k)$ baseado diretamente na informação do estado x_k da planta. O Crítico não requer o modelo da planta para cálculo. O treinamento do Ator (controlador), por outro lado, necessita encontrar as derivadas $\partial\widehat{V}^h(x_k)/\partial u_k$ em cada instante k. Assim, o algoritmo HDP utiliza o modelo da planta somente para a atualização do controlador (WANG; ZHANG; LIU, 2009).

Figura 2.3: Estrutura de HDP.



2.8.3 Aprendizagem Q para Controle Ótimo Adaptativo

O método de aprendizagem por reforço via aprendizagem Q resulta em um algoritmo de controle adaptativo que converge *online*. Esse resolve a equação de *Bellman* (2.57) e a equação

HJB (2.55) de maneira *online* em tempo real usando dados medidos ao longo da trajetória do sistema, sem o conhecimento das dinâmicas $f(x_k),\ g(x_k)$ (LEWIS; VRABIE; VAMVOUDAKIS, 2012).

A aprendizagem Q é chamada de ADHDP (*Action Dependent Heuristic Dynamic Programming*) por Werbos, uma vez que a função Q depende tanto do estado quanto da ação de controle, sendo assim um método de aprendizagem por reforço que funciona com PDMs desconhecidos, onde a dinâmica dos sistemas não é completamente conhecida. Para isso, aprende a função Q (2.39) usando métodos de *diferença temporal* através da execução de uma ação de controle u_k e medição à cada estágio de tempo

$$Q^{\pi}(x_k, u_k) = r(x_k, u_k) + \gamma Q^{\pi}(x_{k+1}, h(x_{k+1})), \tag{2.64}$$

que define um erro a diferenças temporais

$$e_k = -Q^{\pi}(x_k, u_k) + r(x_k, u_k) + \gamma Q^{\pi}(x_{k+1}, h(x_{k+1})). \tag{2.65}$$

O algoritmo de Iteração de Valor para a Função Q é dado por (2.47). Baseado neste, a função Q é atualizada usando o algoritmo

$$\begin{aligned} Q_k(x_k, u_k) = Q_{k-1}(x_k, u_k) + \alpha_k[r(x_k, u_k) \\ +\gamma \min_u Q_{k-1}(x_{k+1}, u) - Q_{k-1}(x_k, u_k)]. \end{aligned} \tag{2.66}$$

Para o LQR de tempo discreto, a função Q é quadrática em termos de $z \equiv [x_k^T\ u_k^T]^T$. Assume-se que, para sistemas não lineares, a Função Q é parametrizada como

$$\hat{Q}(x, u) = \boldsymbol{\phi}^T(z)\boldsymbol{\theta}, \tag{2.67}$$

para alguns parâmetros desconhecidos do vetor $\boldsymbol{\theta}$ e conjunto do vetor de base $\boldsymbol{\phi}(z)$. Substituindo a aproximação da função Q dentro do erro de diferenças temporais, Eq.(2.65), produz-se a

$$e_k = -\boldsymbol{\phi}^T(z_k)\boldsymbol{\theta} + r(x_k, u_k) + \gamma \boldsymbol{\phi}^T(z_{k+1})\boldsymbol{\theta}. \tag{2.68}$$

Considerando-se o algoritmo iteração de política, a equação que representa o passo de

avaliação da função Q é

$$(\boldsymbol{\phi}^T(z_k) - \gamma\boldsymbol{\phi}^T(z_{k+1}))\boldsymbol{\theta}_{j+1} = r(x_k, h_j(x_k)), \tag{2.69}$$

e o passo de melhoria de política

$$h_{j+1}(x_k) = \arg\min_u(\boldsymbol{\phi}^T(x_k, u)\boldsymbol{\theta}_{j+1}), \text{ para todo } x \in X. \tag{2.70}$$

A Aprendizagem Q usando iteração de valor é expressa por

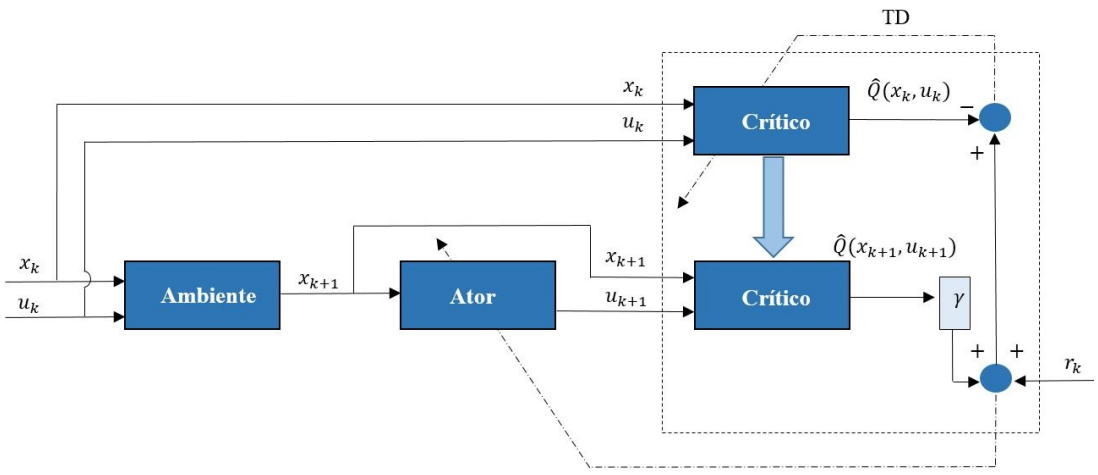
$$\boldsymbol{\phi}^T(z_k)\boldsymbol{\theta}_{j+1} = r(x_k, h_j(x_k)) + \gamma\boldsymbol{\phi}^T(z_{k+1})\boldsymbol{\theta}_j, \tag{2.71}$$

juntamente com a Eq.(2.70). Uma vez que estas equações não requerem o conhecimento das dinâmicas $f(\cdot),\ g(\cdot)$.

Para a implementação *online*, o RLS pode ser usado para resolver a Eq.(2.69) para o vetor de parâmetro $\boldsymbol{\theta}_{j+1}$, dado o vetor de regressão $(\boldsymbol{\phi}^T(z_k) - \gamma\boldsymbol{\phi}^T(z_{k+1}))$. Os dados observados a cada instante são $(z_k,\ z_{k+1},\ r(x_k,\ u_k))$ com $z_k \equiv [x_k^T\ u_k^T]^T$. O vetor $z_{k+1} \equiv [x_{k+1}^T\ u_{k+1}^T]^T$ é calculado usando $u_{k+1} = h_j(x_{k+1})$, com $h_j(\cdot)$ correspondente a política atual.

A Estrutura de ADHDP é mostrada na Figura 2.4. Observa-se que tanto o estado x_k quanto o controle u_k são entradas para o crítico. O controle é obtido através da equação do gradiente, $\frac{\partial\hat{Q}(x_k,u_k)}{\partial u_k} = 0$. Dessa forma, o algoritmo ADHDP não necessita do conhecimento do modelo da planta para treinamento do ator (AL-TAMIMI et al., 2007; LEWIS; VRABIE, 2009b).

Figura 2.4: Estrutura de ADHDP.



2.9 Considerações Finais

Este capítulo nos fornece os conceitos e métodos de aprendizagem por reforço formulados no contexto de processos de decisão markovianos e programação dinâmica, onde os problemas fundamentais à aprendizagem por reforço são descritos como o problema de avaliação de política e a síntese de política de decisão ótima. Fez-se uma revisão sobre PDMs, definindo-se o problema de decisão sequencial ótima. A programação dinâmica é abordada e descrita via formas da equação de *Bellman*, Eq.(2.20) da otimalidade de *Bellman* e Eq.(2.25) para iteração de política. A partir dessas tem-se os métodos convencionais de iteração de política e iteração de valor dependentes de modelo, que são implementados "para trás no tempo". Em seguida, apresenta-se a função Q e logo após descrevem-se métodos de diferenças temporais, que servem de base para a construção de algoritmos *online* baseados nas formas das equações de *Bellman* para avaliar políticas de decisão ou resolver problemas de decisão ótima em uma maneira "para frente no tempo" baseados em dados do sistema observados ao longo de sua trajetória. Apresenta-se também a abordagem de controle ótimo adaptativo, via aproximação de função valor e aprendizagem Q, sendo esta última capaz de produzir métodos em ADP que convergem de forma $online$ para a solução de problemas de controle ótimo para sistemas cujas dinâmicas são desconhecidas.

3

PROJETO ONLINE DE CONTROLE ÓTIMO DLQR

3.1 Introdução

A otimalidade de controladores está atrelada à solução da equação de *Riccati*. Por meio de uma discretização do sistema dinâmico, o DLQR pode ser caracterizado como um problema de possíveis estágios e estados, sendo possível assim, estabelecer um caminho das trajetórias através da Programação Dinâmica.

Neste capítulo, será visto como formular estes procedimentos em um método de aprendizagem por reforço usando-se dados do sistema ao longo de sua trajetória, utilizando a Programação Dinâmica Heurística Dependente de Ação, onde o crítico não necessita do modelo da planta para o cálculo da função valor.

O controle ótimo linear quadrático (LQ) é um tipo especial de controle ótimo. Suas principais características são otimização em termos de critério de desempenho quadrático e incorporação da teoria de reconstrução de estado ótima de *Kalman-Bucy*.

Uma classe ampla de problemas de controle ótimo que envolve uma planta descrita por uma equação de estado linear e variante no tempo dada por

$$x_{k+1} = A_k x_k + B_k u_k, \tag{3.1}$$

com estado inicial x_i dado, sendo x_k o vetor de estado de dimensão n e u_k o vetor de entrada de controle de dimensão m. A lei $u(\cdot)$ para controlar o sistema de Eq.(3.1) é estabelecida para tentar fazê-lo tão próximo quanto possível de uma trajetória desejada de estado, a qual é dada por uma função prescrita $\tilde{x}(\cdot)$ do tempo, em um intervalo de tempo especificado $[i, N]$ levando em consideração a energia mínima de controle utilizada. Assim, o problema LQ caracteriza-se por uma estrutura de otimização com o objetivo de determinar uma lei de controle $u_k,\ k \in$

$[i,\ N]$ que minimiza o índice de desempenho dado por

$$\begin{aligned} V(x_i, u_{(\cdot)}, i) &= (x_N - \tilde{x}_N)^T S(x_N - \tilde{x}_N) + \\ &+ \sum_{k=i}^{N-1} [(x_k - \tilde{x}_k)^T Q_k (x_k - \tilde{x}_k) + u_k^T R_k u_k], \end{aligned} \tag{3.2}$$

e que tenha como restrição dinâmica o sistema de Equação (3.1). As matrizes $Q = Q^T \geq 0$ e $R = R^T > 0$, com dimensões apropriadas, representam ponderações do erro entre o estado $x(\cdot)$ e a trajetória desejada $\tilde{x}(\cdot)$, e do controle, respectivamente. A matriz $S = S^T \geq 0$ pondera o desvio de estado final no instante N. O problema definido pelas Equações (3.1) e (3.2) é conhecido como o problema de rastreamento de tempo discreto.

Um dos resultados mais marcantes em teoria de controle ótimo linear quadrático é que se a otimização é sobre um horizonte de tempo infinito, a lei de controle ótima resultante fornece boas propriedades, incluindo estabilidade de malha fechada. Esses resultados estão intrinsecamente relacionados às propriedades de estabilidade e detectabilidade do sistema. Para mais discussão sobre este tópico, veja (ANDERSON; MOORE, 2007; BRYSON, 1975; KIRK, 2004).

3.2 As equações de *Bellman* para o DLQR

O problema do regulador linear quadrático discreto (DLQR) constitui um caso particular do problema LQ discreto e é formulado como uma estrutura de otimização dinâmica com objetivo de determinar uma lei de controle u_k que minimiza um índice de desempenho quadrático. Assim, para o problema do DLQR onde o PDM é determinístico e atende a equação de transição de estados dada por

$$x_{k+1} = Ax_k + Bu_k, \tag{3.3}$$

com índice k de tempo discreto. O índice de desempenho de horizonte infinito associado tendo custo de estágio determinístico é

$$J_k = \frac{1}{2} \sum_{i=k}^{\infty} r_i = \frac{1}{2} \sum_{i=k}^{\infty} (x_i^T Q x_i + u_i^T R u_i), \tag{3.4}$$

onde o espaço de estado $X = \mathbb{R}^n$ e o espaço de ação $U = \mathbb{R}^m$ são infinitos e contínuos. Esse caso particular do problema do DLQR ocorre quando as matrizes A e B do sistema dinâmico e as matrizes de ponderações Q e R são invariantes no tempo e o intervalo de otimização é infinito.

Estudam-se as equações de *Bellman* associadas ao problema do regulador linear quadrático no contexto de programação dinâmica. O índice de desempenho J_k depende do estado atual x_k e de todas a entradas de controle $u_k, u_{k+1}, \ldots$. Seleciona-se uma política $u_k = \mu(x_k)$ fixa de estabilização, e a função valor associada será

$$V(x_k) = \frac{1}{2}\sum_{i=k}^{\infty} r_i = \frac{1}{2}\sum_{i=k}^{\infty}(x_i^T Q x_i + u_i^T R u_i). \tag{3.5}$$

Uma vez que a política é fixada, a função valor dependerá apenas do estado inicial x_k. E a soma infinita (3.5) equivale a uma equação à diferenças, dada por

$$\begin{aligned} V(x_i) &= \frac{1}{2}(x_k^T Q x_k + u_k^T R u_k) + \frac{1}{2}\sum_{i=k+1}^{\infty}(x_i^T Q x_i + u_i^T R u_i) \\ &= \frac{1}{2}(x_k^T Q x_k + u_k^T R u_k) + V(x_{k+1}). \end{aligned} \tag{3.6}$$

A solução $V(x_k)$ dada por (3.5), quando a condição $V(0) = 0$ é satisfeita, corresponde à solução definida positiva para esta equação. A Eq.(3.6) é exatamente a equação de *Bellman* (LEWIS; LIU, 2012) para o LQR de tempo discreto. Esta formulação implica que estratégias de controle ótimo devem ser determinadas recursivamente no tempo a partir do estágio final (procedimento "para trás no tempo").

Para o DLQR, o valor ótimo da função de custo assume a seguinte forma quadrática

$$V(x_k) = \frac{1}{2}x_k^T P x_k, \tag{3.7}$$

para alguma matriz de Kernel P, temos a equação de *Bellman* na seguinte forma

$$2V(x_k) = x_k^T P x_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1}, \tag{3.8}$$

usando a equação de estado, reescrevemos

$$2V(x_k) = x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P(Ax_k + Bu_k). \tag{3.9}$$

Assumindo uma política estacionária $u_k = \mu(x_k) = -Kx_k$, para um ganho K estabilizante, segue

$$2V(x_k) = x_k^T P x_k = x_k^T Q x_k + x_k^T K^T R K x_k + x_k^T (A - BK)^T P (A - BK) x_k. \tag{3.10}$$

Uma vez que este ganho K for válido para todas as trajetórias de estado, temos

$$(A - BK)^T P (A - BK) - P + Q + K^T R K = 0, \tag{3.11}$$

sendo esta à equação de Lyapunov. Ou seja, a equação de *Bellman* (2.17) para o DLQR equivale a equação de Lyapunov. Uma vez que o índice de desempenho não é descontado, isto é, $\gamma = 1$, deve-se selecionar um ganho de estabilização K, ou seja, uma política estabilizadora.

Uma observação refere-se ao fato das Eqs.(3.6), (3.8), (3.10) e (3.11) serem todas formulações equivalentes a equação de *Bellman*, onde as formas (3.6) e (3.8) não envolvem as matrizes dinâmicas do sistema (A, B). Enquanto que a solução da equação de Lyapunov (3.11) está atrelada ao total conhecimento das dinâmicas do sistema.

Assim, empregando-se a forma (3.6) ou (3.8) para a equação de Bellman, algoritmos de aprendizado por reforço para a aprendizagem de soluções ótimas *online* podem ser planejados. Ou seja, a aprendizagem por reforço permite que a equação de Lyapunov seja resolvida *online* sem o conhecimento das matrizes do sistema (A, B).

A função Hamiltoniana para o DLQR é dada por

$$H(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P (Ax_k + Bu_k) - x_k^T P x_k, \tag{3.12}$$

sendo este o erro de diferença temporal em um PDM.

Pela condição de estacionariedade $\partial H(x_k, u_k)/\partial u_k = 0$, a condição necessária para a otimalidade é satisfeita. Resolvendo a equação acima, obtém-se o controle ótimo

$$u_k = -Kx_k = -(B^T P B + R)^{-1} B^T P A x_k. \tag{3.13}$$

Pela inserção de (3.13) em (3.12), produz-se a equação algébrica de *Riccati* de tempo discreto (*Discrete Algebraic Riccati Equation* - DARE), ou a equação da otimalidade de *Bell-*

man (2.20) para o LQR de tempo discreto

$$A^T PA - P + Q - A^T PB(B^T PB + R)^{-1} B^T PA = 0, \tag{3.14}$$

também denominada de equação HJB-*Riccati*.

3.3 Iteração de Política e Iteração de Valor para o DLQR

Os métodos de iteração de política e iteração de valor já descritos neste trabalho, são apresentados agora em relação ao regulador linear quadrático de tempo discreto.

A relevância desses dois processos para sistemas de controle realimentados reside na possibilidade de implementação de sistemas dinâmicos *online* em tempo real por meio dos dados observados e medidos ao longo da trajetória do sistema.

A equação de *Bellman* (2.17) para o DLQR é equivalente às formulações (3.6), (3.8), (3.10) e (3.11) descritas anteriormente, que são usadas para a implementação da iteração de política e iteração de valor.

3.3.1 Iteração de Política

Com um passo de índice j, a iteração de política utiliza o passo de avaliação descrito pela Eq.(2.25) em (3.6) para produzir

$$V^{j+1}(x_k) = \frac{1}{2}(x_k^T Q x_k + u_k^T R u_k) + V^{j+1}(x_{k+1}), \tag{3.15}$$

onde os sobrescritos representam os passos de iteração do algoritmo e os subscritos o horizonte de tempo k.

O método de iteração de política aplicado à Eq.(3.8) produz

$$x_k^T P^{j+1} x_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1}, \tag{3.16}$$

e a iteração de política dada com auxílio da Eq.(3.11) produz a equação de Lyapunov

$$(A - BK^j)^T P^{j+1}(A - BK^j) - P^{j+1} + Q + (K^j)^T R K^j = 0. \tag{3.17}$$

O passo de melhoria de politica é então descrito por

$$\mu^{j+1}(x_k) \quad = \quad K^{j+1}x_k = \arg\min(x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P^{j+1} x_{k+1}), \tag{3.18}$$

podendo ser reescrito como

$$K^{j+1} = -(B^T P^{j+1} B + R)^{-1} B^T P^{j+1} A. \tag{3.19}$$

Para as Eqs.(3.17) e (3.19) o algoritmo baseia-se na repetição das soluções da equação de Lyapunov a cada etapa, sendo dito algoritmo de Hewer. Este, por sua vez, converge para a solução da equação de *Riccati* (3.14). Esta técnica requer total conhecimento das dinâmicas do sistema, de modo que sua implementação é realizada *offline*.

3.3.2 Iteração de Valor

Quando se aplica a iteração de valor, Eq.(2.30) por meio da Eq.(3.8) da equação de *Bellman*, produz-se

$$x_k^T P^{j+1} x_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P_{k+1}^j, \tag{3.20}$$

e o formato (3.11), produz a recursão de Lyapunov

$$P^{j+1} = (A - BK^i)^T P^j (A - BK^i) + Q + (K^j)^T R K^j. \tag{3.21}$$

Para ambas as formulações, o passo de melhoria de política é descrito por (3.18) e (3.19).

Este algoritmo converge (LANCASTER; RODMAN, 1995) para a solução da equação de *Riccati* (3.14), sendo necessária à sua implementação o total conhecimento das dinâmicas do sistema (A, B). Portanto, tal recursão de Lyapunov é um algoritmo *offline*.

3.3.3 Solução *online* da Equação de *Riccati*

Nas formulações descritas para o algoritmo de Hewer e a recurção de Lyapunov, percebe-se a necessidade de modelo do sistema, caracterizando ambos como métodos de solução *offline*. Observa-se no entanto que nos passos de ambos os algoritmos a iteração de política nos formatos (3.16) e (3.18) e a iteração de valor nos formatos (3.20) e (3.18) para a implementação

ocorre sem dependência das dinâmicas do sistema (A, B). Para tais formulações, o valor e controle ótimos são determinados *online* em tempo real através dos dados obtidos ao longo da trajetória do sistema.

3.3.4 Avaliação de Política Iterativa

Dada uma política K fixa, o procedimento de avaliação de política iterativa torna-se

$$P^{j+1} = (A - BK)^T P^j (A - BK) + Q + K^T R K, \tag{3.22}$$

e esta recursão converge para a solução da equação de Lyapunov $P^{j+1} = (A - BK)^T P^j (A - BK) + Q + (K)^T RK$ se $(A - BK)$ for estável, para qualquer escolha inicial do valor de P^0.

3.4 Função Q para o DLQR

Em termos da formulação da Equação de *Bellman* para o DLQR, seguindo uma política $u_k = \mu(x_k)$, a função Q é

$$Q(x_k, u_k) = \frac{1}{2}(x_k^T Q x_k + u_k^T R u_k) + V(x_{k+1}), \tag{3.23}$$

onde o controle u_k é arbitrário e a política $u_k = \mu(x_k)$ é seguida por $k + 1$ e instantes subsequentes. Reescrevendo

$$2Q(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P (Ax_k + Bu_k), \tag{3.24}$$

com P sendo a solução de $Riccati$ produz-se a Função Q para o DLQR:

$$Q(x_k, u_k) = \frac{1}{2} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} A^T P A + Q & A^T P B \\ B^T P A & B^T P B + R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \tag{3.25}$$

Define-se

$$Q(x_k, u_k) \equiv \frac{1}{2} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T S \begin{bmatrix} x_k \\ u_k \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} S_{xx} & S_{xu} \\ S_{ux} & S_{uu} \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}, \tag{3.26}$$

sendo S uma matriz simétrica. Da Eq.(3.26), tem-se

$$Q(x_k, u_k) \equiv \frac{1}{2}\left[x_k^2 S_{xx} + u_k S_{ux} x_k + u_k x_k S_{xu} + u_k^2 S_{uu}\right]. \tag{3.27}$$

Aplicando $\partial Q(x_k, u_k)/\partial u_k = 0$ à Eq.(3.27) obtém-se

$$\frac{1}{2}[2S_{ux}x_k + 2u_k S_{uu}] = 0, \tag{3.28}$$

assim

$$u_k = -S_{uu}^{-1} S_{ux} x_k, \tag{3.29}$$

e para (3.25) produz-se

$$u_k = -(B^T P B + R)^{-1} B^T P A x_k. \tag{3.30}$$

A Eq.(3.30) é dependente de modelo, mas a Eq.(3.29) necessita apenas da matriz de Kernel S obtida da função Q. Tal formulação fornece meios para a construção de uma família de algoritmos que buscam a solução *online* da equação de *Riccati* sem o conhecimento das dinâmicas do sistema (A, B) (LEWIS; VRABIE; VAMVOUDAKIS, 2012).

3.5 Controlador Adaptativo para Solução *online* do DLQR

Com base na aprendizagem *Q* apresenta-se um algoritmo de controle adaptativo que converge *online* para a solução do problema DLQR. Através da solução da equação algébrica de *Riccati* em tempo real, sem o conhecimento das dinâmicas do sistema, utilizam-se dados medidos ao longo das trajetórias do sistema para realizar este procedimento.

A aprendizagem *Q* é implementada realizando-se repetidamente as iterações de avaliação e melhoria de política para a função Q. Uma vez que para o LQR de tempo discreto a função Q é quadrática nos estados e entradas de modo que $Q(x_k, u_k) = Q(z_k) \equiv \frac{1}{2} z_k^T S z_k$ onde $z_k = [x_k^T \;\; u_k^T]^T$. A matriz S pode ser estimada *online* sem dependência de modelo usando técnicas de identificação de modelo. Escreve-se a função Q na forma paramétrica

$$Q(x, u) = Q(z) = \boldsymbol{\phi}^T(z)\boldsymbol{\theta}, \tag{3.31}$$

onde $\boldsymbol{\theta}$ representa o vetor dos elementos de S e o vetor de base $\boldsymbol{\phi}(z)$ consiste dos termos quadráticos nas componentes de z, contendo os componentes de estado e ação. Sabe-se que a função valor de ação é quadrática nos termos de z, então $Q(x,u) = \frac{1}{2} z_k^T S z_k$ para alguma matriz de kernel S. Aqui as entradas redundantes são removidas, de modo que $\boldsymbol{\theta}$ é composto de $(n+n_e)(n+n_e+1)/2$ na parte superior de S, com $x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^m$.

Para o DLQR a formulação da equação de *Bellman* (2.69) via aprendizagem Q é dada por

$$(\boldsymbol{\phi}^T(z_k) - \boldsymbol{\phi}^T(z_{k+1}))\boldsymbol{\theta}_{j+1} = \frac{1}{2}(x_k^T Q x_k + u_k^T R u_k). \tag{3.32}$$

A matriz Q aqui corresponde à matriz de ponderação do índice de desempenho, e não a função Q. Esta equação deve ser resolvida em cada etapa j do processo de aprendizagem Q. Percebe-se que (3.32) é uma equação com $(n+n_e)(n+n_e+1)/2$ incógnitas, correspondendo às entradas de $\boldsymbol{\theta}$, sendo resolvida *online* usando métodos de controle adaptativo tais como mínimos quadrados recursivo (RLS).

A implementação do algoritmo está apresentada a seguir:

ALGORITMO 1 - CONTROLADOR APROXIMADO - DLQR

▷ **Bloco 0 - Inicialização**
▷ Matrizes de Ponderação - $[Q, R]$
▷ Selecione - Política Inicial - K_0.
▷ Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
▷ Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.
▷ Selecione - Número de Iterações - N.

$k \leftarrow 0, j \leftarrow 0$
▷ **Processo Iterativo**
Para cada iteração j de política
do
for $i \leftarrow 1 : N$
do
▷ **Bloco 1 - Simulação do Ambiente**
Estimar $(x_k, u_k, x_{k+1}, u_{k+1})$

▷ **Bloco 2 - Avaliação de Política Aproximada**
▷ Montagem do Alvo
$r_k \leftarrow x_k^T Q x_k + u_k^T R u_k$
▷ Vetor de Funções de Base - Produto de Kronecker
$\phi_k \leftarrow [x_k^T, u_k^T]^T \otimes [x_k^T, u_k^T]^T - \gamma [x_{k+1}^T, u_{k+1}^T]^T \otimes [x_{k+1}^T, u_{k+1}^T]^T$
▷ Mínimos Quadrados Recursivos - RLS
Atualize $\theta_j(i)$ por meio da equação 3.32
$k \leftarrow k + 1$
▷ Recuperação da matriz S a partir do vetor θ_{j+1}
$S^{\theta_{j+1}}$ é dado via Eq.(3.26).

▷ **Bloco 3 - Atualização da Política**
$K_{j+1} \leftarrow (S_{uu}^{\theta_{j+1}})^{-1} (S_{ux}^{\theta_{j+1}})$

▷ Reinicialização - Parâmetros do RLS
$\theta_{j+1}(0) = \theta_j$
until K_{j+1} seja satisfatória
then Fim do Processo Iterativo

End - Fim do Processo Iterativo

Este procedimento encerra-se quando não há mais atualizações para a função Q ou para a política de controle em cada etapa. Assim, o algoritmo resolve a equação algébrica de *Riccati online* usando dados $(x_k, u_k, x_{k+1}, u_{k+1})$ medidos em tempo real a cada passo de tempo k.

3.6 Considerações Finais

Neste capítulo, realizou-se aplicações da teoria de aprendizagem por reforço sobre o problema do DLQR, com ênfase na solução *online* da equação de *Hamilton-Jacobi-Bellman*. Segue-se com as abordagens de iteração política e iteração de valor para o LQR de tempo discreto. Por fim, é apresentado o método de programação dinâmica heurística dependente de ação (aprendizagem Q), que está inserido na abordagem crítica adaptativa, para a determinação de soluções aproximadas da equação de *Hamilton-Jacobi-Bellman* e políticas de decisão ótimas para viabilizar o desenvolvimento de um projeto *online* de sistema de controle ótimo.

4

ALGORITMOS ONLINE PARA PROJETO DE CONTROLE ÓTIMO ADHDP-DLQR BASEADOS EM APRENDIZAGEM RLS

4.1 Introdução

O problema de aproximar uma função valor em programação dinâmica geralmente envolve métodos incrementais para resolver o problema de aprendizagem *online* dos parâmetros da rede crítica. Entre os algoritmos iterativos propostos para estimar tais parâmetros, destaca-se o aprendizado dos mínimos quadrados recursivos (RLS). A eficiência dos métodos RLS na aprendizagem incremental ator-crítica deve-se principalmente à sua robustez para lidar com variações de tempo nos parâmetros de regressão e a rápida velocidade de convergência quando comparados com métodos de gradiente estocástico (FERREIRA; RÊGO; NETO, 2017).

A aprendizagem RLS é enfatizada sob o ponto de vista da pesquisa de ADP e desenvolvimento de sistemas de controle. Estruturas baseadas em RLS foram propostas em (XU; HE; HU, 2002) para melhorar a eficiência dos métodos heurísticos crítico-adaptativos convencionais. Em (CHU et al., 2008) e (CHENG; FENG; WANG, 2013), os autores exploram métodos RLS para resolver problemas de aprendizagem por reforço ator-crítico. Pietquin e outros (PIETQUIN; GEIST; CHANDRAMOHAN, 2011) apresentaram um avanço recente em métodos de diferença temporal (TD) como a diferença temporal de Kalman (KTD). Neste esquema, um filtro de Kalman é incorporado na estimação do processo de aproximação da função valor usando uma representação de espaço de estado. Um desenvolvimento prévio sobre paradigmas de KTD é apresentado no trabalho por Geist e outros (GEIST; PIETQUIN; FRICOUT, 2009) para processos de decisão markovianos determinísticos.

Em termos de aprendizagem RLS para resolver a equação algébrica de *Riccati* discreta (DARE), também referida como equação HJB-*Riccati*, em problemas de controle ótimo que são resolvidos pela abordagem Programação Dinâmica Heurística (HDP), os autores (RÊGO; NETO; FERREIRA, 2013) desenvolveram métodos e algoritmos baseados no treinamento RLS

para o projeto *online* do regulador linear quadrático discreto (DLQR). Em (FERREIRA; NETO; RÊGO, 2016), os autores fornecem propostas baseadas em transformações unitárias para resolver problemas de convergência e estabilidade numérica relacionados ao mal condicionamento da matriz de covariância da abordagem RLS para aproximar a função de custo DLQR via HDP. No entanto, o método HDP é baseado no modelo do sistema dinâmico para calcular uma política de controle melhorada (FONSECA; RÊGO, 2014).

Em oposição à abordagem HDP, o método de aprendizagem Q, proposto por (WATKINS, 1989; WATKINS; DAYAN, 1992), é um método ADP baseado em dados, que (WERBOS, 1992) denominou de Programação Dinâmica Heurística Dependente da Ação (ADHDP). Para os algoritmos de aprendizagem Q, a função Q, também chamada função valor de ação, é usada em vez da função valor de estado V dos algoritmos iterativos de HDP. A função valor $V(x)$ é uma função apenas do estado x do sistema dinâmico. Em contrapartida, a função $Q(x, u)$ depende tanto do estado x quanto da ação de controle u, o que significa que ela já inclui informações sobre o sistema e a função de custo (BUSONIU et al., 2010). Como tal, os algoritmos baseados em aprendizagem Q são preferíveis para obter o controle ótimo de sistemas para os quais o modelo explícito não está disponível (BUSONIU et al., 2010). O controle ótimo pode ser estimado *online* em tempo real diretamente dos dados observados ao longo das trajetórias do sistema, sem conhecer o modelo de dinâmica do sistema.

Resultados sólidos de estabilidade e convergência foram obtidos para vários conceitos críticos adaptativos para o problema LQR (LANDELIUS, 1997). Em (WATKINS; DAYAN, 1992), uma prova de convergência do algoritmo de aprendizagem Q foi proposta para ambientes estocásticos. Outra contribuição importante inclui o trabalho de Bradtke, (BRADTKE; YDSTIE; BARTO, 1994), onde aprendizagem Q mostrou convergir ao usar aproximadores lineares da função valor Q. Especificamente, essa abordagem baseou-se em uma importante condição de excitação persistente dos métodos RLS. Embora a aprendizagem Q tenha convergência garantida para a função Q ótima, a estabilidade numérica do sistema sob a lei de controle iterativo não é assegurada.

Na presente pesquisa é apresentada a concepção de algoritmos ADHDP para a solução *online* de problemas de controle ótimo. O desenvolvimento dos métodos e algoritmos propostos é focado principalmente nas decomposições $\mathcal{QR}$ e UDU^T que são inseridas no contexto da solução dos problemas de instabilidade numérica para calcular a função valor de ação Q. Aqui, o fenômeno de instabilidade numérica refere-se a uma classe de problemas em que as

variáveis atualizadas na implementação computacional do método RLS-ADHDP perdem suas propriedades teóricas. Exemplos de tais propriedades são a simetria e a positividade da matriz de covariância da abordagem RLS. Devido aos problemas de mal condicionamento desta matriz, a solução para uma política de decisão ótima para um determinado ponto de operação pode não convergir. Deve-se notar que, para que os sistemas de controle ótimo tenham uma viabilidade mínima para operarem em tempo real, parâmetros de desempenho tais como convergência paramétrica, estabilidade numérica e complexidade numérica devem ser levados em consideração no projeto. Este tópico e os problemas de estabilidade numérica e convergência discutidos em (RÊGO; NETO; FERREIRA, 2013; FERREIRA; RÊGO; NETO, 2017) motivaram o desenvolvimento desta pesquisa tendo em vista o projeto de algoritmos de ADP para controle ótimo *online* com melhores especificações para operarem em tempo real.

4.2 Algoritmo RLS$_\mu$-ADHDP-DLQR

Inicialmente, considera-se que o problema de aproximação da função valor Q pode ser formulado por uma estrutura paramétrica da forma

$$Q(x,u) = \boldsymbol{\varphi}^T(z)\boldsymbol{\theta}, \tag{4.1}$$

para alguns parâmetros desconhecidos do vetor de pesos a serem estimados $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \ldots\ \theta_{n_\theta}]^T$ e conjunto do vetor de funções de base ou vetor de regressão $\boldsymbol{\varphi}(z) = [\phi_1(z)\ \phi_2(z)\ \ldots\ \phi_{n_\theta}(z)]^T$ com $z_k = [x_k^T\ \ u_k^T]^T$. Esta aproximação leva em consideração o método RLS para o problema de estimação do parâmetro $\boldsymbol{\theta}$. Tal abordagem busca realizar a aprendizagem *online* para projetos de controle ótimo, via estimativas da função de custo de uma dada política, construída a partir dos dados $(z_k, z_{k+1}, r(x_k, u_k))$, que são observados ao longo da trajetória do sistema.

O uso de critérios de avaliação de política com base na otimização, faz da estrutura de aprendizagem Q um esquema crítico adaptativo onde o passo de iteração de política, rede crítica, determina a solução de mínimos quadrados para $\boldsymbol{\theta}_{k+1}$

$$(\boldsymbol{\varphi}^T(z_k) - \gamma\boldsymbol{\varphi}^T(z_{k+1}))\boldsymbol{\theta}_k = x_k^T Q x_k + u_k^T R u_k, \tag{4.2}$$

e o passo de melhoria de política, rede de ação, determina uma política melhorada dada por

$$h_{k+1}(x_k) = \arg\min_{h(\cdot)}(\boldsymbol{\phi}^T(z_k)\boldsymbol{\theta}_{k+1}), \tag{4.3}$$

onde cada par de ponderações Q e R define um controlador diferente, portanto, explorando o espaço de possíveis ponderações, aproximamos a solução de programação dinâmica para o controlador ótimo. ADHDP é um método que aprimora o controlador de uma iteração para a próxima, desde o instante de tempo k até $k+1$ (SOKOLOV et al., 2015).

A vetorização de matriz e a teoria do produto de Kronecker (BREWER, 1978; RÊGO; NETO; FERREIRA, 2013) contribuem para uma solução aproximada da equação HJB-*Riccati* obtida através de um esquema iterativo dado por

$$\boldsymbol{\phi}^T(z_k)\boldsymbol{\theta}_k = d(\cdot), \tag{4.4}$$

sendo o vetor de parâmetros $\boldsymbol{\theta}$ correspondente à vetorização da matriz S dada na Eq.(3.26), $\boldsymbol{\theta} = \text{vec}(S)$, e o vetor de regressores $\boldsymbol{\phi}^T(z_k)$, assim como o alvo $d(\cdot)$ determinados por

$$\begin{aligned}\boldsymbol{\phi}_k \;=\; & [x_{1,k}^2; x_{1,k}x_{2,k}; \ldots; x_{1,k}u_{1,k}; \ldots; x_{1,k}u_{n_e,k}; \ldots; x_{n,k}^2; x_{n,k}u_{1,k}; \ldots; x_{n,k}u_{n_e,k}; \ldots \\ & u_{1,k}^2; \ldots; u_{n_e,k}^2]^T - \gamma[x_{1,k+1}^2; x_{1,k}x_{2,k+1}; \ldots; x_{1,k+1}u_{1,k+1}; \ldots; x_{1,k+1}u_{n_e,k+1}; \ldots \\ & x_{n,k+1}^2; x_{n,k+1}u_{1,k+1}; \ldots; x_{n,k+1}u_{n_e,k+1}; u_{1,k+1}^2; \ldots; u_{n_e,k+1}^2]^T \end{aligned} \tag{4.5}$$

e

$$d(\cdot) = x_k^T Q x_k + u_k^T R u_k. \tag{4.6}$$

O problema consiste em se determinar uma estimativa do vetor de parâmetros $\boldsymbol{\theta}$ a partir de um dado conjunto de pares de observações e regressores $\{(d(\cdot), \boldsymbol{\phi}_k),\ k = 1, 2,\ \ldots,\ N\}$, levando-se em consideração projetos *online* para aplicações em tempo real. A estimativa de mínimos quadrados de $\boldsymbol{\theta}$ é definida como o vetor que minimiza a seguinte função de perda

$$J(\theta, N) = \sum_{k=1}^{N} \mu^{N-k} \left[d(\cdot) - \boldsymbol{\phi}_k^T\boldsymbol{\theta}\right]^2, \tag{4.7}$$

sendo μ o fator de esquecimento, variando no intervalo $0 < \mu \leq 1$, sendo um parâmetro que pondera dinamicamente a influência de dados amostrados na função de perda, e N o número de dados da amostra.

Para satisfazer a condição de excitação do problema de mínimos quadrados necessita-se que a quantidade N de dados amostrados seja no mínimo igual a n_θ. Supondo-se que a matriz

Hessiana de $J(\theta, N)$ é definida positiva, a equação do gradiente $\nabla J(\theta, N) = 0$ fornece uma única solução, que é dada por

$$\boldsymbol{\theta}_N = \boldsymbol{\Phi}_N^{-1} \boldsymbol{\eta}_N, \tag{4.8}$$

onde

$$\boldsymbol{\Phi}_N = \sum_{k=1}^{N} \mu^{N-k} \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T \tag{4.9}$$

é a matriz de correlação, $n_\theta \times n_\theta$, dos dados de entrada do vetor $\boldsymbol{\phi}$, e

$$\boldsymbol{\eta}_N = \sum_{k=1}^{N} \mu^{N-k} \boldsymbol{\phi}_k d(\cdot) \tag{4.10}$$

é o vetor de correlação cruzada entre as entradas do estimador LS e a resposta desejada.

Por pequenas manipulações algébricas, as Eqs.(4.8)-(4.10) são desenvolvidas em uma forma recursiva dadas por

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{\Phi}_k^{-1} \boldsymbol{\phi}_k (d_k - \boldsymbol{\phi}_k^T \boldsymbol{\theta}_{k-1}), \tag{4.11}$$

sendo $\boldsymbol{\Phi}_k$ uma forma recursiva dada por

$$\boldsymbol{\Phi}_k = \mu \boldsymbol{\Phi}_{k-1} + \boldsymbol{\phi}_k \boldsymbol{\phi}_k^T. \tag{4.12}$$

Aplicando-se o lema da inversão de matriz na Eq.(4.12), a estimativa RLS na forma (4.11)-(4.12) pode ser reescrita por

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{L}_k (d_k - \boldsymbol{\phi}_k^T \boldsymbol{\theta}_{k-1}), \tag{4.13}$$

sendo

$$\boldsymbol{L}_k = \boldsymbol{\Gamma}_k \boldsymbol{\phi}_k = \frac{\boldsymbol{\Gamma}_{k-1} \boldsymbol{\phi}_k}{\mu + \boldsymbol{\phi}_k^T \boldsymbol{\Gamma}_{k-1} \boldsymbol{\phi}_k}, \tag{4.14}$$

introduz-se $\boldsymbol{\Gamma}_k = \boldsymbol{\Phi}_k^{-1}$ que é dado por

$$\boldsymbol{\Gamma}_k = \mu^{-1}\left(\boldsymbol{\Gamma}_{k-1} - \boldsymbol{L}_k \boldsymbol{\phi}_k^T \boldsymbol{\Gamma}_{k-1}\right), \tag{4.15}$$

a matriz $\boldsymbol{\Gamma}_k$, $n_\theta \times n_\theta$, é denominada matriz de correlação inversa/matriz de covariância. O vetor $\boldsymbol{L}_k$, $n_\theta \times 1$, é chamado vetor de ganho.

Um resumo do Algoritmo RLS$_\mu$-ADHDP-DLQR é apresentado a seguir.

ALGORITMO 2 - RLS$_\mu$-ADHDP-DLQR

▷ **Bloco 0 - Inicialização**
▷ Matrizes de Ponderação e do Sistema Dinâmico - Q, R, A_d, B_d
▷ Selecione - Política Inicial - K_0.
▷ Selecione - Revitalização dos Estados - x_{revit}, n_{revit}
▷ Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
▷ Selecione - Estado Inicial - x_0.
▷ Parâmetros do RLS: θ_0, Γ_0.
▷ Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.
▷ Selecione - Número de Iterações - N.

▷ Processo Iterativo
for $k \leftarrow 0 : N$
do
▷ **Bloco 1 - Simulação do Ambiente**
▷ Ruído de Controle (componente de "exploração" do sinal de controle)
$\epsilon_k \leftarrow [\ \ ]$
▷ Ação de Controle
$u_k \leftarrow -K_k x_k + \epsilon_k$
▷ Estado Seguinte
$x_{k+1} \leftarrow A_d x_k + B_d u_k$
▷ Ação de Controle Seguinte
$u_{k+1} \leftarrow -K_k x_{k+1}$

▷ **Bloco 2 - Atualização da Política e da Função Valor**
▷ Montagem do Alvo
$d(r) \leftarrow x_k^T Q x_k + u_k^T R u_k$
▷ Vetor de Funções de Base - Produto de Kronecker
$\phi_k \leftarrow [x_{1,k}^2; x_{1,k}x_{2,k}; \ldots; x_{1,k}u_{1,k}; \ldots; x_{1,k}u_{n_e,k}; \ldots; x_{n,k}^2;$
$x_{n,k}u_{1,k}; \ldots; x_{n,k}u_{n_e,k}; u_{1,k}^2; \ldots; u_{n_e,k}^2;] - \gamma[x_{1,k+1}^2;$
$x_{1,k}x_{2,k+1}; \ldots; x_{1,k+1}u_{1,k+1}; \ldots; x_{1,k+1}u_{n_e,k+1}; \ldots;$
$x_{n,k+1}^2; x_{n,k+1}u_{1,k+1}; \ldots; x_{n,k+1}u_{n_e,k+1}; u_{1,k+1}^2; \ldots; u_{n_e,k+1}^2;]$
▷ Mínimos Quadrados Recursivos - RLS
$L_{k+1} = \frac{\Gamma_k \phi_k}{\mu + \phi_k^T \Gamma_k \phi_k}$
$\theta_{k+1} = \theta_k + L_{k+1}(d(\cdot) - \phi_k^T \theta_k)$
$\Gamma_{k+1} = \mu^{-1}(\Gamma_k - L_{k+1}\phi_k^T \Gamma_k)$
▷ Recuperação da matriz S a partir do vetor θ
$n_\theta = (n + n_e)(n + n_e + 1)/2$

$$S^{\theta_{k+1}} \leftarrow \begin{bmatrix} \theta_1 & \theta_2/2 & \cdots & \theta_{(n+n_e)}/2 \\ \theta_2/2 & \theta_{(n+n_e)+1} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \theta_{n_{\theta-1}}/2 \\ \theta_{(n+n_e)}/2 & \theta_{2(n+n_e)-1}/2 & \cdots & \theta_{n_\theta} \end{bmatrix}$$

▷ Atualização da Política (Ganho Ótimo de Realimentação K) K
$K_{k+1} \leftarrow (S_{uu}^{\theta_{k+1}})^{-1}(S_{ux}^{\theta_{k+1}})$

---

if $k\%n_{revit} = 0$
then
$x_{k+1} \leftarrow x_{revit}$

End - Fim do Processo Iterativo

O algoritmo RLS$_\mu$-ADHDP-DLQR (algoritmo 2) possui em seu bloco principal as etapas de avaliações de políticas e melhorias de políticas. Dessa forma, observa-se que neste algoritmo a política é atualizada em um único *loop*, em direção à política ótima em cada passo de tempo k. O bloco 0 no algoritmo 2, passos 2-9, realiza a inicialização de parâmetros fixos do problema de otimização que são as ponderações Q e R, as matrizes do sistema dinâmico A_d e B_d, e o fator de desconto γ. Para o problema RLS, as condições iniciais são o fator de esquecimento, o parâmetro $\boldsymbol{\theta}$ e a matriz $\boldsymbol{\Gamma}$. Os passos 14-24 definem a lei de controle u_k (em termos de ação), sua reação do ambiente x_{k+1} (em termos de transição de estado) e o custo $d(\cdot)$ (em termos de custo de transição de estado). Para esta situação, fazendo o papel de um simulador do sistema dinâmico (atuador, planta, sensor) tem-se a interatividade ação-ambiente-observação para avaliações e melhorias de políticas de controle. Os passos 26-33 do algoritmo realizam a vetorização dos estados e ações de controle e a aproximação da matriz S via estimação RLS. Os passos 35-36 realizam a associação da aproximação $S^{\theta_{k+1}}$ que é usada no passo 38 para realização de melhorias de políticas K_{k+1}.

Embora algoritmos do tipo RLS tenham uma alta velocidade de convergência. No entanto, estes algoritmos são propensos a sérias dificuldades numéricas que estão bem documentadas na literatura (KAMINSKI; BRYSON; SCHMIDT, 1971). Por exemplo, de acordo com a Eq.(4.15), a matriz de covariância $\boldsymbol{\Gamma}_{(\cdot)}$ é definida como a diferença entre matrizes definidas positivas. Portanto, a menos que a exatidão numérica empregada em cada iteração do algoritmo seja suficientemente alta, a matriz de covariância resultante desse cálculo pode não ser definida positiva.

Dessa forma a abordagem proposta neste trabalho é vista como uma melhoria no processo de estimação RLS de políticas de decisão ótima DLQR para evitar problemas de convergência e estabilidade numérica via decomposições $\mathcal{QR}$ e UDU^T. Tal metodologia consiste em melhorar ou reduzir a dependência linear, isto é, tornar os regressores linearmente independentes ou evitar que se tornem linearmente dependentes sob iteração de política no contexto de aprendizagem por reforço.

4.3 Algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

O uso de rotações de Givens juntamente com a decomposição $\mathcal{QR}$ para resolver problemas LS foi abordado por Gentleman (GENTLEMAN, 1973), que foi um dos pioneiros no uso da decomposição $\mathcal{QR}$ para resolver problemas (RLS) (KUNG; GENTLEMAN, 1982). McWhirter

(1983a, 1983b) e mais tarde Ward, Hargrave e McWhirter (1986) propuseram implementações de filtragem adaptativa do algoritmo RLS para resolver a equação de minimização do erro quadrático. Eles aproveitaram as rotações de Givens para a triangularização da matriz de dados e sugeriram uma implementação sistólica, que é interessante por se tratar de um método de paralelismo das rotações de Givens.

Nesta seção se mostra a adequação do Algoritmo RLS$_\mu$-ADHDP-DLQR para o algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR, ou seja, o algoritmo RLS$_\mu$-ADHDP-DLQR baseado em decomposição $\mathcal{QR}$, que é orientado para a avaliação de soluções aproximadas da equação HJB-*Riccati* que preservem suas características de estabilidade numérica.

Assumindo-se que para as equações normais do estimador LS, o valor ótimo do vetor de parâmetro $\boldsymbol{\theta}_k$ é tal que

$$\boldsymbol{\Phi}_k \boldsymbol{\theta}_k = \boldsymbol{\eta}_k. \tag{4.16}$$

E seja $\boldsymbol{\Phi}_k$ expresso em sua forma fatorada

$$\boldsymbol{\Phi}_k = \boldsymbol{\Phi}_k^{1/2} \boldsymbol{\Phi}_k^{H/2}, \tag{4.17}$$

sendo $\boldsymbol{\Phi}_k^{1/2}$ uma matriz triangular inferior definida como a raiz quadrada de $\boldsymbol{\Phi}_k$, e $\boldsymbol{\Phi}_k^{H/2}$ a transposta hermitiana de $\boldsymbol{\Phi}_k^{1/2}$.

A partir da Eq.(4.12), a equação recursiva de atualização de valor da matriz de correlação $\boldsymbol{\Phi}_k$ é dada por

$$\boldsymbol{\Phi}_k = \mu \boldsymbol{\Phi}_{k-1} + \boldsymbol{\phi}_k \boldsymbol{\phi}_k^H. \tag{4.18}$$

Então, pré-multiplicando ambos os lados da Eq.(4.17) pela matriz $\boldsymbol{\Phi}_k^{-1/2}$, produz-se uma nova variável dada por

$$\boldsymbol{\omega}_k = \boldsymbol{\Phi}_k^{H/2} \boldsymbol{\theta}_k = \boldsymbol{\Phi}_k^{-1/2} \boldsymbol{\eta}_k, \tag{4.19}$$

Deste modo, chega-se a uma nova forma para representar o vetor de correlação cruzada entre a entrada $\boldsymbol{\phi}_k$ e a saída desejada d_k que é dada por

$$\boldsymbol{\eta}_k = \mu \boldsymbol{\eta}_{k-1} + \boldsymbol{\phi}_k d_k^*, \tag{4.20}$$

ou, de forma equivalente,

$$\boldsymbol{\Phi}_k\boldsymbol{\theta}_k = \mu\boldsymbol{\Phi}_{k-1}\boldsymbol{\theta}_{k-1} + \boldsymbol{\phi}_k d_k^*, \tag{4.21}$$

em que o asterisco denota o complexo conjugado.

Agora, as Eq.(4.18) e (4.21) são descritas em suas formas transpostas hermitianas, expressando cada um dos termos que aparecem no segundo membro de cada equação em suas formas transpostas como segue

$$\mu\boldsymbol{\Phi}_{k-1}^H = [\mu^{1/2}\boldsymbol{\Phi}_{k-1}^{1/2}].[\mu^{1/2}\boldsymbol{\Phi}_{k-1}^{H/2}], \tag{4.22}$$

$$\begin{aligned}\mu\boldsymbol{\theta}_{k-1}^H\boldsymbol{\Phi}_{k-1}^H &= [\mu^{1/2}\boldsymbol{\theta}_{k-1}^H\boldsymbol{\Phi}_{k-1}^{1/2}].[\mu^{1/2}\boldsymbol{\Phi}_{k-1}^{H/2}] \\ &= [\mu^{1/2}\boldsymbol{\omega}_{k-1}^H][\mu^{1/2}\boldsymbol{\Phi}_{k-1}^{H/2}].\end{aligned} \tag{4.23}$$

Pode-se identificar dois outros termos, $\boldsymbol{\phi}_k\boldsymbol{\phi}_k^H$ e $d_k\boldsymbol{\phi}_k^H$, os quais juntamente com os elementos das Eqs.(4.22) e (4.23) são dispostos de modo a construir a seguinte matriz de dados (pré-matriz).

$$\begin{bmatrix} \mu^{1/2}\boldsymbol{\Phi}_{k-1}^{1/2} & \boldsymbol{\phi}_k \\ \mu^{1/2}\boldsymbol{\omega}_{k-1}^H & d_k \\ 0^T & 1 \end{bmatrix}.$$

Suponha, então, que escolhe-se uma rotação unitária Θ_k transformando-se essa pré-matriz de modo a produzir um bloco zero na segunda entrada da linha de bloco superior da matriz resultante (pós-matriz), como mostrado pela forma

$$A\Theta = B, \tag{4.24}$$

sendo

$$A = \begin{bmatrix} \mu^{1/2}\boldsymbol{\Phi}_{k-1}^{1/2} & \boldsymbol{\phi}_k \\ \mu^{1/2}\boldsymbol{\omega}_{k-1}^H & d_k \\ 0^T & 1 \end{bmatrix} \quad \text{e} \quad B = \begin{bmatrix} B_{11\,k}^H & 0 \\ b_{21\,k}^H & b_{22\,k}^* \\ b_{31\,k}^H & b_{32\,k}^* \end{bmatrix}. \tag{4.25}$$

Para avaliar os elementos dos blocos desconhecidos $B_{11\,k}^H$, $b_{21\,k}^H$, $b_{22\,k}^*$, $b_{31\,k}^H$ e $b_{32\,k}^*$ da pós-matriz, B, procede-se elevando ambos os membros da Eq.(4.24) ao quadrado. Reconhece-se que Θ_k é uma matriz unitária, e, portanto, $\Theta_k\Theta_k^H$ é igual a matriz identidade para todo k.

Então, a Eq.(4.24) pode ser escrita como:

$$AA^H = BB^H, \tag{4.26}$$

ou, de forma expandida obtém-se

$$\begin{bmatrix} \mu^{1/2}\boldsymbol{\Phi}_{k-1}^{1/2} & \boldsymbol{\phi}_k \\ \mu^{1/2}\boldsymbol{\omega}_{k-1}^H & d_k \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} \mu^{1/2}\boldsymbol{\Phi}_{k-1}^{H/2} & \mu^{1/2}\boldsymbol{\omega}_{k-1} & 0 \\ \boldsymbol{\phi}_k^H & d_k^* & 1 \end{bmatrix} = \begin{bmatrix} B_{11\,k}^H & 0 \\ b_{21\,k}^H & b_{22\,k}^* \\ b_{31\,k}^H & b_{32\,k}^* \end{bmatrix} \begin{bmatrix} B_{11\,k} & b_{21\,k} & b_{31\,k} \\ 0^T & b_{22\,k} & b_{32\,k} \end{bmatrix}.$$

Assim, comparando-se os respectivos termos em ambos os lados da Eq.(4.27), os elementos dos blocos desconhecidos da pós-matriz B são determinados. Assim, a Eq.(4.24) com os valores de $B_{11\,k}^H$, $b_{21\,k}^H$, $b_{22\,k}^*$, $b_{31\,k}^H$ e $b_{32\,k}^*$ é expressa na forma

$$\begin{bmatrix} \mu^{1/2}\boldsymbol{\Phi}_{k-1}^{1/2} & \boldsymbol{\phi}_k \\ \mu^{1/2}\boldsymbol{\omega}_{k-1}^H & d_k \\ 0^T & 1 \end{bmatrix} \Theta_k = \begin{bmatrix} \boldsymbol{\Phi}_k^{1/2} & 0 \\ \boldsymbol{\omega}_k^H & \xi_k\rho_k^{1/2} \\ \boldsymbol{\phi}_k^H\boldsymbol{\Phi}_k^{-H/2} & \rho_k^{1/2} \end{bmatrix}, \tag{4.27}$$

sendo ξ_k o erro de estimação *a priori* que é dado por $\xi_k = d_k - \boldsymbol{\theta}_{k-1}^H\boldsymbol{\phi}_k$, e ρ_k um parâmetro real o qual é definido como

$$\rho_k = 1 - \boldsymbol{L}_k^H\boldsymbol{\phi}_k, \tag{4.28}$$

sendo ρ_k referido como o parâmetro de positividade.

A matriz Θ_k é essencialmente qualquer rotação unitária que opera sobre os elementos do vetor de dados de entrada $\boldsymbol{\phi}_k$ na pré-matriz, eliminando-os um a um, de modo a produzir um bloco de entrada zero na linha do bloco superior da pós-matriz. Como resultado, a estrutura triangular inferior da raiz quadrada da matriz de correlação, ou seja, $\boldsymbol{\Phi}^{1/2}$, é preservada na sua forma exata antes e depois da transformação.

Observa-se que o cálculo do vetor de parâmetro $\boldsymbol{\theta}$ do RLS é realizado trabalhando diretamente com a matriz de dados de entrada através da decomposição $\mathcal{QR}$, ao invés de trabalhar com a matriz de correlação inversa dos dados de entrada como ocorre em algoritmos RLS.

Após computar os valores dos blocos atualizados $\boldsymbol{\Phi}_k^{1/2}$ e ρ_k^H, o vetor de parâmetro $\boldsymbol{\theta}_k$ do

LS, pode então ser calculado usando-se a Eq.(4.19), como segue

$$\boldsymbol{\theta}_k^H = \boldsymbol{\omega}_k^H \boldsymbol{\Phi}_k^{-1/2}. \tag{4.29}$$

As Eqs.(4.27) e (4.29) são agrupadas nos procedimentos do algoritmo de ADHDP (algoritmo 2), sob esquemas de iteração de política, em que a etapa de avaliação de política aproximada é realizada por meio de transformações unitárias (rotações de Givens, para mais detalhes consultar Apêndice C) que fornecem a matriz Θ_k, correspondendo a maior diferença entre as abordagens dos algoritmos RLS$_\mu$-ADHDP-DLQR e RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. Observa-se que o cálculo da solução da Eq.(4.29) é realizado usando o método de "substituição para trás" que explora a estrutura triangular inferior de $\boldsymbol{\Phi}_k^{1/2}$.

ALGORITMO 3 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR - Decomposição $\mathcal{QR}$ por Rotações de Givens

---

▷ **Bloco 0 - Inicialização**

▷ Matrizes de Ponderação e do Sistema Dinâmico - Q, R, A_d, B_d

▷ Selecione -Política Inicial - K_0.

▷ Selecione - Fator de Desconto - $0 < \gamma \leq 1$.

▷ Selecione - Estado Inicial - x_0.

▷ Selecione - Revitalização dos Estados - x_{revit}, n_{revit}

▷ Parâmetros do RLS$_\mu$-$\mathcal{QR}$: $\Phi_0^{1/2}, \omega_0$.

▷ Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.

▷ Selecione - Número de Iterações - N.

▷ Processo Iterativo

for $k \leftarrow 0 : N$

do

▷ **Bloco 1 - Simulação do Ambiente**

▷ Ruído de Controle (componente de "exploração"do sinal de controle)

$\epsilon_k \leftarrow [\quad]$

▷ Ação de Controle

$u_k \leftarrow -K_k x_k + \epsilon_k$

▷ Estado Seguinte

$x_{k+1} \leftarrow A_d x_k + B_d u_k$

▷ Ação de Controle Seguinte

$u_{k+1} \leftarrow -K_k x_{k+1}$

▷ **Bloco 2 - Atualização da Função Valor e da Política**

▷ Montagem do Alvo

$d(r) \leftarrow x_k^T Q x_k + u_k^T R u_k$

▷ Vetor de Funções de Base - Produto de Kronecker

$$\phi_k \leftarrow [x_{1,k}^2; x_{1,k}x_{2,k}; \ldots; x_{1,k}u_{1,k}; \ldots; x_{1,k}u_{n_e,k}; \ldots; x_{n,k}^2; \\ x_{n,k}u_{1,k}; \ldots; x_{n,k}u_{n_e,k}; u_{1,k}^2; \ldots; u_{n_e,k}^2;] - \gamma[x_{1,k+1}^2; \\ x_{1,k}x_{2,k+1}; \ldots; x_{1,k+1}u_{1,k+1}; \ldots; x_{1,k+1}u_{n_e,k+1}; \ldots; \\ x_{n,k+1}^2; x_{n,k+1}u_{1,k+1}; \ldots; x_{n,k+1}u_{n_e,k+1}; u_{1,k+1}^2; \ldots; u_{n_e,k+1}^2;]$$

▷ RLS baseado em decomposição $\mathcal{QR}$ através de "Rotações Givens"

$$A_{\mathcal{QR}} \leftarrow \begin{bmatrix} \mu^{1/2}\Phi_k^{1/2} & \phi_k \\ \mu^{1/2}\omega_k^H & d(\cdot) \\ 0^T & 1 \end{bmatrix}$$

$$C_{\mathcal{QR}} \leftarrow \frac{a_1}{\sqrt{a_1^2+a_2^2}}$$

$$S_{\mathcal{QR}} \leftarrow -\frac{a_2}{\sqrt{a_1^2+a_2^2}}$$

$$J_{\mathcal{QR}} \leftarrow \begin{bmatrix} C_{\mathcal{QR}} & S_{\mathcal{QR}} \\ -S_{\mathcal{QR}} & C_{\mathcal{QR}} \end{bmatrix}$$

$$\mathcal{R} \leftarrow J_{\mathcal{QR}} * A_{\mathcal{QR}}$$

$$\mathcal{R} \rightarrow \begin{bmatrix} \Phi_{k+1}^{1/2} & 0 \\ \omega_{k+1}^H & \xi_k \rho_{k+1}^{1/2} \\ \phi_k^H \Phi_{k+1}^{-H/2} & \rho_{k+1}^{1/2} \end{bmatrix}$$

▷ Atualização do Vetor θ através de "substituição para trás"

$${\theta^H}_{k+1} = \omega_{k+1}^H \Phi_{k+1}^{-1/2}$$

▷ Recuperação da matriz S a partir do vetor θ

$$n_\theta = (n + n_e)(n + n_e + 1)/2$$

$$S^{\theta_{k+1}} \leftarrow \begin{bmatrix} \theta_1 & \theta_2/2 & \cdots & \theta_{(n+n_e)}/2 \\ \theta_2/2 & \theta_{(n+n_e)+1} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \theta_{n_{\theta-1}}/2 \\ \theta_{(n+n_e)}/2 & \theta_{2(n+n_e)-1}/2 & \cdots & \theta_{n_\theta} \end{bmatrix}$$

▷ Atualização da Política (Ganho Ótimo de Realimentação K) K

$$K_{k+1} \leftarrow (S_{uu}^{\theta_{k+1}})^{-1}(S_{ux}^{\theta_{k+1}})$$

---

if $k\%n_{revit} = 0$

then

$x_{k+1} \leftarrow x_{revit}$

End - Fim do Processo Iterativo

4.4 Algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR

Nessa seção, apresenta-se um método alternativo, via decomposição UDU^T, aos problemas de convergência e perda de estabilidade numérica relacionados ao mal condicionamento da matriz de covariância da abordagem RLS para aproximações da função valor de ação em esquemas de ADHDP.

Uma versão modificada das Eqs.(4.13)-(4.15) pode ser obtida supondo-se que $\boldsymbol{\Gamma}_k = \boldsymbol{U}_k\boldsymbol{D}_k\boldsymbol{U}_k^T$, sendo $\boldsymbol{U}_k$ uma matriz triangular superior com todos os elementos diagonais unitários e $\boldsymbol{D}_k$ uma matriz diagonal.

Sejam $\boldsymbol{U}_k = [u_{1,k} \ \cdots \ u_{n_\theta,k}]$ e $\boldsymbol{D}_k = diag(d_{1,k}, \ldots, d_{n_\theta,k})$. Supondo-se que $\boldsymbol{\Gamma}_{k-1}$ admite a decomposição $\boldsymbol{\Gamma}_{k-1} = \boldsymbol{U}_{k-1}\boldsymbol{D}_{k-1}\boldsymbol{U}_{k-1}^T$, as Eqs.(4.14) e (4.15) podem ser rearranjadas para

$$\boldsymbol{L}_k = \boldsymbol{U}_{k-1}\boldsymbol{g}\beta^{-1} \tag{4.30}$$

e

$$\boldsymbol{\Gamma}_k = \mu^{-1}\boldsymbol{U}_{k-1}[\boldsymbol{D}_{k-1} - \beta^{-1}\boldsymbol{g}\boldsymbol{g}^T]\boldsymbol{U}_{k-1}^T, \tag{4.31}$$

Uma vez que o produto de duas matrizes triangulares superiores unitárias é ainda uma matriz triangular superior unitária, para obter-se fatores $\boldsymbol{U}$-$\boldsymbol{D}$ de $\boldsymbol{\Gamma}_k$ é suficiente obter-se fatores $\boldsymbol{U}$-$\boldsymbol{D}$ para a expressão entre colchetes na Eq.(4.31). Os detalhes dos parâmetros da fatoração são apresentados no Apêndice A.

O desenvolvimento do algoritmo RLS$_\mu$-ADHDP-DLQR com fatoração UDU^T, denominado RLS-UDU^T-ADHDP-DLQR, é baseado nas Eqs. (4.32)-(4.43) que estão dispostas a seguir:

$$\varepsilon_k = d(\cdot) - \boldsymbol{\phi}_k^T\boldsymbol{\theta}_{k-1}, \tag{4.32}$$

$$\boldsymbol{e} = \boldsymbol{U}_{k-1}^T\boldsymbol{\phi}_k, \tag{4.33}$$

$$\boldsymbol{g} = \boldsymbol{D}_{k-1}\boldsymbol{e}, \tag{4.34}$$

$$\beta_0 = \mu. \tag{4.35}$$

Para j de 1 a n_θ faz-se

$$\beta_j = \beta_{j-1} + \boldsymbol{e}_j \boldsymbol{g}_j, \tag{4.36}$$

$$d_{j,k} = d_{j,k-1} \frac{\beta_{j-1}}{\beta_j \mu}, \tag{4.37}$$

$$\kappa_j = \boldsymbol{g}_j, \tag{4.38}$$

$$\tau_j = \frac{-\boldsymbol{e}_j}{\beta_{j-1}}. \tag{4.39}$$

Se $j \neq 1$ faz-se para m de 1 a $j-1$,

$$u_{mj,k} = u_{mj,k-1} + \tau_j \kappa_m \tag{4.40}$$

$$\kappa_{m,new} = \kappa_{m,old} + u_{mj,k-1} \kappa_j \tag{4.41}$$

A atualização do vetor de ganho é dada por

$$\boldsymbol{L}_k = \frac{\begin{bmatrix} \kappa_1 & \kappa_2 & \cdots & \kappa_{n_\theta} \end{bmatrix}^T}{\beta_{n_\theta}}. \tag{4.42}$$

A atualização do vetor de parâmetros é

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{L}_k \varepsilon_k. \tag{4.43}$$

Essas equações são agrupadas, passando a integrar o bloco 2 do algoritmo 2, de modo que o passo de avaliação de política que aproxima a matriz S é realizado via estimação RLS com fatoração UDU^T. O conjuto de Eqs.(4.32)-(4.43) substitui os passos 31-34 do algoritmo 2, o que corresponde a maior diferença entre as abordagens dos algoritmos RLS$_\mu$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR.

ALGORITMO 4 - RLS$_\mu$-UDU^T-ADHDP-DLQR

▷ **Bloco 0 - Inicialização**
▷ Matrizes de Ponderação e do Sistema Dinâmico - Q, R, A_d, B_d
▷ Selecione -Política Inicial - K_0.
▷ Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
▷ Selecione - Estado Inicial - x_0.
▷ Selecione - Revitalização dos Estados - x_{revit}, n_{revit}
▷ Parâmetros do RLS$_\mu$-UDU^T: θ_0, U_0, D_0.
▷ Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.
▷ Selecione - Número de Iterações - N.

▷ Processo Iterativo
for $k \leftarrow 0 : N$
do
▷ **Bloco 1 - Simulação do Ambiente**
▷ Ruído de Controle (componente de "exploração"do sinal de controle)
$\epsilon_k \leftarrow [\quad]$
▷ Ação de Controle
$u_k \leftarrow -K_k x_k + \epsilon_k$
▷ Estado Seguinte
$x_{k+1} \leftarrow A_d x_k + B_d u_k$
▷ Ação de Controle Seguinte
$u_{k+1} \leftarrow -K_k x_{k+1}$

▷ **Bloco 2 - Atualização da Função Valor e da Política**
▷ Montagem do Alvo
$d(r) \leftarrow x_k^T Q x_k + u_k^T R u_k$
▷ Vetor de Funções de Base - Produto de Kronecker
$\phi_k \leftarrow [x_{1,k}^2; x_{1,k}x_{2,k}; \ldots; x_{1,k}u_{1,k}; \ldots; x_{1,k}u_{n_e,k}; \ldots; x_{n,k}^2;$
$x_{n,k}u_{1,k}; \ldots; x_{n,k}u_{n_e,k}; u_{1,k}^2; \ldots; u_{n_e,k}^2;] - \gamma[x_{1,k+1}^2;$
$x_{1,k}x_{2,k+1}; \ldots; x_{1,k+1}u_{1,k+1}; \ldots; x_{1,k+1}u_{n_e,k+1}; \ldots;$
$x_{n,k+1}^2; x_{n,k+1}u_{1,k+1}; \ldots; x_{n,k+1}u_{n_e,k+1}; u_{1,k+1}^2; \ldots; u_{n_e,k+1}^2;]$

▷ Atualiza-se θ usando as Eqs.(4.32)-(4.43) da estimação RLS com Fatoração UDU^T

▷ Recuperação da matriz S a partir do vetor θ

$$S^{\theta_{k+1}} \leftarrow \begin{bmatrix} \theta_1 & \theta_2/2 & \cdots & \theta_{(n+n_e)}/2 \\ \theta_2/2 & \theta_{(n+n_e)+1} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \theta_{n_{\theta-1}}/2 \\ \theta_{(n+n_e)}/2 & \theta_{2(n+n_e)-1}/2 & \cdots & \theta_{n_\theta} \end{bmatrix}$$

▷ Atualização da Política (Ganho Ótimo de Realimentação K) K
$K_{k+1} \leftarrow (S_{uu}^{\theta_{k+1}})^{-1}(S_{ux}^{\theta_{k+1}})$

---

if $k\%n_{revit} = 0$
then
$x_{k+1} \leftarrow x_{revit}$

---

end - Fim do Processo Iterativo

4.5 Considerações Finais

Neste capítulo, a caracterização do problema de estimação RLS para a solução da equação HJB-*Riccati* associada com o problema DLQR via ADP foi apresentada. A formulação do problema é apresentada como fusão de três metodologias para projeto *online* de controle ótimo baseada na solução da equação HJB: programação dinâmica (iteração de política), aprendizagem por reforço (diferenças temporais) e aproximação da função valor via estrutura paramétrica

RLS.

O problema principal está relacionado com o mal condicionamento da matriz de covariância da abordagem RLS para estimar a solução da equação HJB-*Riccati*, a qual resulta em dificuldades numéricas na atualização das variáveis da recursão do RLS em ambientes de precisão finita.

Com o intuito de minimizar esse problema, propôs-se a inserção de métodos de decomposição $\mathcal{QR}$ e UDU^T no algoritmo padrão para o projeto *online* de controle ótimo DLQR. Tal abordagem foi desenvolvida para melhorar o comportamento do número de condição e parâmetro de positividade da matriz de covariância que são avaliados para monitorar e guiar o processo de convergência e estabilidade numérica, resultando assim nos algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR (variantes do algoritmo RLS$_\mu$-ADHDP-DLQR).

5

COMPLEXIDADE COMPUTACIONAL

5.1 Introdução

A teoria da complexidade de algoritmos tem sido parte central do projeto de algoritmos eficientes desde os anos iniciais da computação. O estudo desse tópico apresenta duas motivações principais. Uma delas é a necessidade de comparar algoritmos de forma idônea, com independência dos recursos computacionais disponíveis, da eficiência das técnicas computacionais existentes e até da capacidade técnica dos programadores que os implementem. E a outra, a baixa quantidade de recursos computacionais existentes nos primeiros anos da computação tornavam essa necessidade teórica também uma exigência do ponto de vista prático. A noção de complexidade assintótica (HARTMANIS; STEARNS, 1965), assim, desempenha um papel central na análise da eficiência de algoritmos.

Em Matemática, a análise assintótica consiste de uma técnica de aproximação de funções. Em programação, princípios de análise assintótica são utilizados para avaliação de complexidade de algoritmos em termos de tempo e espaço gastos para realização de suas respectivas tarefas. Sendo a teoria da complexidade computacional um ramo da teoria da computação em ciência da computação teórica e matemática que se concentra em classificar problemas computacionais de acordo com sua dificuldade inerente e relacionar essas classes entre si (DOWNEY; FELLOWS; STEGE, 1999), identificando que tipos de problemas podem, em princípio, ser resolvidos através de algoritmos. Nesse contexto, um problema computacional é entendido como uma tarefa que é, em princípio, possível de ser resolvida por um computador (uma vez que o problema pode ser descrito por um conjunto de instruções matemáticas).

A complexidade de um algoritmo é, dessa forma, calculada como uma função que relaciona o tamanho das instâncias do problema ao qual o algoritmo é aplicado com a quantidade de instruções computacionais requeridas pelo algoritmo para concluir sua execução. Em geral, esse tipo de cálculo baseia-se na ideia de que à medida que o tamanho dos dados de entrada de um problema cresce, a complexidade do algoritmo que o resolve estabiliza-se numa função

de proporcionalidade simples relacionada com alguma função conhecida (por exemplo, função linear, quadrática, logarítmica, dentre outras).

A análise da complexidade assintótica de algoritmos é particularmente importante para o projeto de novos algoritmos. Uma vez que os padrões de crescimento de funções mais comuns são bem conhecidos, torna-se possível examinar a eficiência de algoritmos e prever seu comportamento em diferentes tamanhos de instâncias.

Uma vez que muitos problemas possuem frequentemente vários algoritmos com funções equivalentes, isto é, dois algoritmos são funcionalmente equivalentes quando, pelo menos em princípio, fornecem os mesmos resultados para um mesmo problema. Mas, o fato de dois algoritmos serem funcionalmente equivalentes não quer dizer que eles sejam equivalentes em termos de eficiência. Ao contrário, existem algoritmos que são funcionalmente equivalentes e que, ao mesmo tempo, divergem bastante em termos de eficiência.

Campos intimamente relacionados com a ciência da computação teórica são a análise de algoritmos e teoria da computabilidade. Uma distinção chave entre a análise de algoritmos e teoria da complexidade computacional é que a primeira é dedicada a analisar a quantidade de recursos necessários para um determinado algoritmo resolver um problema, enquanto a segunda interpela de modo mais geral sobre todos os possíveis algoritmos que podem ser usados para resolver o mesmo problema (GOLDREICH, 2008), sendo responsável por classificar os problemas que podem ou não serem resolvidos com os recursos devidamente restritos.

5.2 Avaliação de Algoritmos

Apesar da noção de algoritmo ser utilizada há pelo menos dois mil anos, apenas no século vinte o conceito computacional foi formalmente definido e estudado. Com base portanto na teoria define-se um algoritmo como uma sequência finita de instruções sem ambiguidade, cada uma das quais pode ser executada automaticamente em um período de tempo finito e com uma quantidade de esforço finita (GOLDREICH, 2008).

Um problema de ordenação, que visa colocar em ordem decrescente uma sequência de números, pode ser utilizado para ilustrar o conceito de algoritmo. Deste modo, uma sequência numérica é tida como uma instância do problema, consistindo na entrada (que satisfaz a quaiquer restrições impostas no enunciado do problema) necessária para se calcular uma solução para o problema (CORMEN et al., 2001).

Assim, diz-se que um algoritmo correto resolve o problema computacional dado, uma

vez que para cada instância de entrada, ele para com a saída correta. Um algoritmo incorreto pode não parar em algumas instâncias de entrada, ou então pode parar com outra resposta que não seja a desejada.

Diferentes algoritmos podem realizar a mesma tarefa usando um conjunto diferenciado de instruções em diferentes tempos, espaços ou esforços. Tais diferenças podem ser reflexo da complexidade computacional aplicada, que depende de estruturas de dados adequados ao algoritmo (SIPSER, 2006).

Analisar um algoritmo significa avaliar os recursos sob os quais o algoritmo necessitará. Tipicamente, pode-se avaliar um algoritmo quanto a parâmetros como:

1. Velocidade de Execução, sendo um dos pontos importantes em algoritmos para processamento em tempo real.

2. Utilização de Memória; os avanços na área de informática elevou a disponibilidade de memória diminuindo a preocupação com esse parâmetro.

3. Estabilidade, de forma a garantir o funcionamento do algoritmo de maneira regular ao passar do tempo de execução.

O presente trabalho envolve a avaliação dos algoritmos desenvolvidos tendo como referência três importantes parâmetros: Custo Computacional, Estabilidade Numérica e Tempo de Convergência, os quais são discutidos a seguir.

5.2.1 Custo Computacional

A complexidade computacional de um algoritmo é medida como uma função do volume de dados que ele processa. Isto é, se o tamanho dos dados (por exemplo, o número de elementos de uma matriz) for n, sua complexidade é associada a alguma função $f(n)$, cujo domínio é o conjunto dos números naturais e cuja imagem é o conjunto dos números reais não negativos. Podendo ser entendida assim como a estimativa do esforço (custo) computacional despendido para a execução de um algoritmo que resolve um problema com entrada de tamanho n.

Em análise de algoritmo este esforço normalmente é medido por uma função da forma $O(f(n))$ (notação matemática denominada "Ó" grande), a qual indica a ordem de complexidade de tempo de execução do algoritmo que está associado ao número de operações aritméticas realizadas pelo mesmo em função da dimensão n da entrada do problema. Em alguns algoritmos a função de complexidade $f(n)$ pode relacionar mais de uma variável, como $f(n, m)$, $f(n, m, p)$,

dentre outros. Dessa forma, algoritmos que necessitam de um número reduzido de operações aritméticas são preferíveis na prática, pois implicam em custos computacionais menores.

Para algoritmos, cujo objetivo está condicionado à realização de longas sequências de cálculos, pode-se ter uma ideia do tempo de execução ou complexidade, através da contagem do número de operações de pontos flutuantes realizadas por segundo (do inglês *Floating-point Operations Por Second*). Usa-se o acrônimo *flops* para designar uma operações básica de ponto flutuante como a soma, subtração, multiplicação, divisão ou extração de raiz quadrada, para determinar o custo computacional dos algoritmos propostos nesse trabalho .

Para os dispositivos computacionais atuais, cuja capacidade de processamento é demasiadamente grande, adotam-se os múltiplos de $flops$ como unidade de medida. Os múltiplos mais utilizados são: *megaflops* (Mflop/s), *gigaflops* (Gflop/s), *teraflops* (Tflop/s), *petaflops* (Pflop/s) e *exaflops* (Eflop/s).

Pode-se exemplificar o cálculo de tais medidas segundo a formulção $Gflops = v \times np \times nc \times oc$, sendo: *Gflops* corresponde a bilhões de operações por segundo, *v* a velocidade da CPU em GHz, *np* o número de CPU's, *nc* o números de núcleos (*cores*) por CPU e *oc* a quantidade de operações de ponto flutuante capazes de serem realizadas por ciclo (4 em geral).

Assim, uma maquina/computador (1 CPU) com velocidade de 3.6 GHz, 4 núcleos de processamento, capaz de realizar 4 $flops$ por ciclo, executaria em tese, 57 $Gflops$ ou 57 bilhões de operações de ponto flutuante por segundo. Dessa forma, uma matriz quadrada de ordem $n = 5.000$, precisa em torno de 83 bilhões de operações para que sua decomposição LU seja computada com um tempo aproximado de $1,4$ segundos, uma vez que todos os núcleos são utilizados.

Em uma maneira geral, quando trabalha-se com sistemas não ideais, os tempos são maiores, pois há restrições de acesso à memória, tempo gasto para transição dos dados da memória para o processador, dentre outras limitações de *hardware*. Por isso, a busca por otimizar os custos computacionais dos algoritmos.

Esta subseção propõe-se à descrever os custos computacionais dos algoritmos abordados nesse estudo. Para tal, faz-se necessário introduzir algumas notações, que serão apresentadas a seguir.

Sejam α_1 e α_2 dois escalares quaisquer em $\mathbb{R}$. As notações $op_\alpha^{(+)}$, $op_\alpha^{(\cdot)}$ e $op_\alpha^{(\div)}$ serão usadas para denotar as operações de adição $\alpha_1 + \alpha_2$, multiplicação $\alpha_1 \cdot \alpha_2$ e divisão α_1/α_2, respectivamente. Tais operações elementares descritas, equivalem à um $flop$ (medida de com-

plexidade aritmética adotada).

As Tabelas 5.1 e 5.2 apresentam algumas operações elementares de vetores e matrizes que são encontradas nas etapas de avaliação e melhoria de política dos algoritmos RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR. Nessas tabelas o número de $flops$ (operações de ponto flutuante por segundo) para cada uma das operações básicas pode ser consultado.

Algumas observações relevantes são:

1. De forma genérica, a multiplicação de duas matrizes M_1 e M_2 com $M_1 \in \mathbb{R}^{m\times p}$ e $M_2 \in \mathbb{R}^{p\times m}$ envolve $2mnp - mn\ flops$ e será denotada por $op^{(*)}_{M_{mpn}}$.

2. Dada uma Matriz M com $M \in \mathbb{R}^{m\times m}$ inversível, a obtençãi da inversa de M, M^{-1}, por meio de transformações de Gauss (eliminação Gaussiana) requer aproximadamente $\frac{2}{3}m^3\ flops$ ($\frac{1}{3}m^3\ flops$ para matrizes definidas positivas). A operação de inversão de uma matriz definida positiva será denotada por $op^{(-1)}_{M_{DP}}$.

Tabela 5.1: Operações básicas com vetores em $\mathbb{R}^{n_\theta}$

Operação	Descrição	Complexidade
$op_v^{(+)}$	Adição: $v_1 + v_2$	n_θ
op_v^{dot}	Produto escalar: $v_1^T v_2$	$2n_\theta - 1$
op_v^{out}	Produto externo: $v_1 v_2^T$	n_θ^2
$op_{v,a}^{(\cdot)}$	Multiplicação de um escalar α por um vetor: αv	n_θ

Tabela 5.2: Operações básicas com matrizes em $\mathbb{R}^{n_\theta\times n_\theta}$

Operação	Descrição	Complexidade
$op_M^{(+)}$	Adição	n_θ^2
$op_M^{(\cdot)}$	Multiplicação	$2n_\theta^3 - n_\theta^2$
$op_{M,\alpha}^{(\cdot)}$	Multiplicação de um escalar por uma matriz	n_θ^2
$op_{M,v}^{(\cdot)}$	Multiplicação de uma matriz por um vetor	$2n_\theta^2 - n_\theta$
$op_{M_{Triangular},v}^{(\cdot)}$	Multiplicação de uma matriz triangular por um vetor	n_θ^2
$op_{M_{Diagonal},v}^{(\cdot)}$	Multiplicação de uma matriz diagonal por um vetor	n_θ

RLS$_\mu$-ADHDP-DLQR

Etapa de Avaliação de Política

Na descrição do custo computacional em uma iteração k de avaliação de política realizada pelo RLS padrão, tem-se:

1. Contrução do vetor de funções de base $\phi_k = \bar{z}_k - \gamma \bar{z}_{k+1}$ via produto de Kronecker do par estado-ação. Considerando que a formação de cada vetor $\bar{z}_k$ envolve $n_\theta[op_\alpha^{(\cdot)}]$, o que equivale à $n_\theta\ flops$, tem-se:

   $op^{(\cdot)} \rightarrow 2n_\theta[op_\alpha^{(\cdot)}] + op_{v,a}^{(\cdot)} \rightarrow 2n_\theta + n_\theta = 3n_\theta\ flops$

   $op^{(+)} \rightarrow op_v^{(+)} \rightarrow n_\theta\ flops$

2. Cálculo do ganho $\boldsymbol{L}_k$ da Eq.(4.14):

   $op^{(\cdot)} \rightarrow 2op_{M,v}^{(\cdot)} + op_v^{dot} + op_{v,\alpha}^{(\cdot)} \Rightarrow 2(2n_{n_\theta}^2 - n_\theta) + (2n_\theta - 1) + n_\theta = 4n_\theta^2 + n_\theta - 1\ flops,$

   $op^{(+)} \rightarrow op_\alpha^{(+)} = 1\ flops,$

   $op^{(\div)} \rightarrow op_\alpha^{(\div)} = 1\ flops.$

3. Atualização do vetor de parâmetros $\hat{\boldsymbol{\theta}}_k$ da Eq.(4.13):

   $op^{(\cdot)} \rightarrow op_v^{dot} + op_v^{(\cdot)} \Rightarrow (2n_\theta - 1) + n_\theta = 3n_\theta - 1\ flops,$

   $op^{(+)} \rightarrow op_\alpha^{(+)} + op_v^{(+)} = n_\theta + 1\ flops.$

4. Atualização da matriz de covariância $\boldsymbol{\Gamma}_k$ da Eq.(4.15):

   $op^{(\cdot)} \rightarrow op_v^{out} + op_M^{(\cdot)} + op_{M,\alpha}^{(\cdot)} \Rightarrow n_\theta^2 + (2n_\theta^3 - n_\theta^2) + n_\theta^2 = n_\theta^3 + n_\theta^2\ flops,$

   $op^{(+)} \rightarrow op_M^{(+)} \Rightarrow n_\theta^2\ flops,$

   $op^{(\div)} \rightarrow op_\alpha^{(\div)} \Rightarrow 1\ flops.$

5. Recuperação da matriz S a partir do vetor $\hat{\boldsymbol{\theta}}$ gerado pelo RLS convencional é dado por:

   $[(n + n_e)^2 - (n + n_e)]op_\alpha^{(\div)} \Rightarrow (n + n_e)^2 - (n + n_e)\ flops.$

RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

Etapa de Avaliação de Política

Na descrição do custo computacional em uma iteração k de avaliação de política realizada pelo método RLS-$_\mu$-$\mathcal{QR}$ com rotações de Givens, tem-se:

1. Construção do vetor de funções de base $\phi_k = \bar{z}_k - \gamma \bar{z}_{k+1}$:
   Segue-se o mesmo procedimento do algoritmo RLS$_\mu$-ADHDP-DLQR, o qual representa $op^{(\cdot)} \rightarrow 3n_\theta\ flops$ e $op^{(+)} \rightarrow n_\theta\ flops$.

2. Cálculo de $C_{\mathcal{QR}}$ e $S_{\mathcal{QR}}$ (Givens):

   $op^{(\cdot)} \rightarrow 4op_{\alpha}^{(\cdot)} = 4\ flops$

   $op^{(+)} \rightarrow 2op_{\alpha}^{(+)} = 2\ flops$

   $op^{(\div)} \rightarrow 2op_{\alpha}^{(\div)} = 2\ flops$

   $op^{(sqrt)} \rightarrow 2op_{\alpha}^{(sqrt)} = 2\ flops$

3. Cálculo da Matriz $\mathcal{R}$:

   $op^{(\cdot)} \rightarrow op_{M}^{(\cdot)} = 2n_{\theta}^{3} - n_{\theta}\ flops$

4. Atualização do vetor de parâmetros $\boldsymbol{\theta}_k$ (Back Substitution)

   $op_{backsub} \rightarrow n_{\theta}^{2} = n_{\theta}^{2}\ flops$

5. Recuperação da matriz S a partir do vetor $\boldsymbol{\theta}$ gerado pelo RLS-$\mathcal{QR}$ é dado por:

   $[(n+n_e)^2 - (n+n_e)]op_{\alpha}^{(\div)} \Rightarrow (n+n_e)^2 - (n+n_e)\ flops.$

RLS$_{\mu}$-UDU^T-ADHDP-DLQR

Etapa de Avaliação de Política

Por fim, é investigado o custo computacional de uma iteração k de avaliação de política realizada pelo método RLS-$_{\mu}$-UDU^T-ADHDP-DLQR. Considere os seguintes passos:

1. Construção do vetor de funções de base $\phi_k = \bar{z}_k - \gamma\bar{z}_{k+1}$:

   Segue-se o mesmo procedimento do algoritmo RLS$_{\mu}$-ADHDP-DLQR, o qual representa $op^{(\cdot)} \rightarrow 3n_{\theta}\ flops$ e $op^{(+)} \rightarrow n_{\theta}\ flops.$

2. Cálculo do erro de estimação de Eq.(4.32):

   $op^{(\cdot)} \rightarrow op_{v}^{(dot)} \Rightarrow 2n_{\theta} - 1\ flops,$

   $op^{(+)} \rightarrow op_{\alpha}^{(+)} \Rightarrow 1\ flops.$

3. Cálculo do vetor $\boldsymbol{e}$ de Eq.(4.33):

   $op^{(\cdot)} \rightarrow op_{M_{Triangular},v}^{(\cdot)} \Rightarrow n_{\theta}^{2}\ flops.$

4. Cálculo do vetor $\boldsymbol{g}$ de Eq.(4.34)

   $op^{(\cdot)} \rightarrow op_{M_{Diagonal},v}^{(\cdot)} \Rightarrow n_{\theta}\ flops.$

5. As operações necessárias a atualização dos escalares $\beta,\ d$ e τ, são obtidas a partir das Eqs.(4.36)-(4.39).

   Tem-se que para Eq.(4.36):

   $op^{(\cdot)} \rightarrow n_\theta[op_\alpha^{(\cdot)}] \Rightarrow n_\theta\ flops,$

   $op^{(+)} \rightarrow n_\theta[op_\alpha^{(+)}] \Rightarrow n_\theta\ flops.$

   Para Eq.(4.37), segue:

   $op^{(\cdot)} \rightarrow n_\theta[2op_\alpha^{(\cdot)}] \Rightarrow 2n_\theta\ flops,$

   $op^{(\div)} \rightarrow n_\theta[op_\alpha^{(\div)}] \Rightarrow n_\theta\ flops,$

   Para a Eq.(4.39), tem-se:

   $op^{(\div)} \rightarrow n_\theta[op_\alpha^{(\div)}] \Rightarrow n_\theta\ flops,$

6. O custo total para a atualização de todos os elementos u_{ml} superiores à diagonal principal da matriz U e para atualização do escalar $\kappa_{m,new}$ é dado por $\sum_{l=2}^{n_\theta} 2(l-1)$, considerando que $m = 1 : l-1$, com $l \neq 1$, e uma vez que $l = 1 : n_\theta$ cada equação requer:

   $op^{(\cdot)} \rightarrow (l-1)[2op_\alpha^{(\cdot)}] \Rightarrow 2(l-1) = n_\theta^2 - n_\theta\ flops.$

   De modo análogo, tem-se que:

   $op^{(+)} \rightarrow (l-1)[2op_\alpha^{(+)}] \Rightarrow 2(l-1) = n_\theta^2 - n_\theta\ flops.$

7. Atualização do vetor de ganho $\boldsymbol{L}_{k+1}$ de Eq.(4.42)

   $op^{(\cdot)} \rightarrow op_{v,\alpha}^{(\cdot)} \Rightarrow n_\theta\ flops,$

   $op^{(\div)} \rightarrow op_\alpha^{(\div)} \Rightarrow 1\ flops.$

8. Atualização do vetor de parâmetros $\boldsymbol{\theta}_{k+1}$ da Eq.(4.43):

   $op^{(\cdot)} \rightarrow op_{v,\alpha}^{(\cdot)} \Rightarrow n_\theta\ flops,$

   $op^{(+)} \rightarrow op_v^{(+)} \Rightarrow n_\theta\ flops.$

9. Custo da recuperação da matriz S a partir do vetor $\boldsymbol{\theta}_{k+1}$ gerado pela fatoração RLS-UDU^T é dado por

   $(n_\theta^2 - n_\theta)op_\alpha^{(\div)} \Rightarrow n_\theta^2 - n_\theta\ flops.$

Etapa de Melhoria de Política

O custo computacional descrito em uma iteração k da etapa de melhoria de política para todas as abordagens propostas, segue:

$$op^{(-1)}_{M_{DP}} + op^{(+)}_{M_{n_e n_e n}} \Rightarrow \frac{1}{3}n_e^3 + (2n_e^2 n - n_e n) = \frac{1}{3}n_e^3 + 2n_e^2 n - n_e n \; flops.$$

Tendo em vista os resultados expostos acima e com o objetivo de melhor avaliar a eficiência dos algoritmos propostos neste trabalho em termos de sua complexidade, apresenta-se um comparativo entre o Custo Computacional dos algoritmos RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR, sendo esta apreciação realizada apenas para a etapa de avaliação de política, por ser a única etapa que possui variações entre os algoritmos.

Como o tempo e execução de um algoritmo está intimamente relacionado a fatores inerentes à máquina onde este é implementado, uma maneira simples de comparar os algoritmos propostos se concentra nas operações de ponto flutuante por segundo (flops) realizadas pelo algoritmo a fim de cumprir determinado estágio. Dessa forma, o esforço computacional dos algoritmos RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR, é analisado segundo a quantidade de flops que cada algoritmo necessita para a realização das operações de adição, multiplicação e divisão, denotados por $op^{(+/-)}$, $op^{(\cdot)}$ e $op^{(\div)}$, um resumo dessas operações é apresentado a seguir.

RLSμ - ADHDP - DLQR

$$op^{(+/-)} \rightarrow \frac{m^4}{4} + \frac{m^3}{2} + \frac{5m^2}{4} + m + 2$$
$$op^{(\cdot)} \rightarrow \frac{m^6}{4} + \frac{3m^5}{4} + m^4 + \frac{11m^3}{4} + \frac{19m^2}{4} + \frac{7m}{2} - 2$$
$$op^{(\div)} \rightarrow m^2 - m + 2$$

RLSμ - $\mathcal{QR}$ - ADHDP - DLQR

$$op^{(+/-)} \rightarrow \frac{m^2}{2} + \frac{m}{2} + 2$$
$$op^{(\cdot)} \rightarrow \frac{m^6}{8} + \frac{3m^5}{8} + \frac{m^4}{4} - \frac{3m^3}{8} + 2m^2 + \frac{3m}{4} + 2$$
$$op^{(\div)} \rightarrow m^2 - m + 2$$

RLSμ - $\mathbf{UDU}^T$- ADHDP - DLQR

$$op^{(+/-)} \rightarrow \frac{m^4}{4} + \frac{m^3}{4} + \frac{5m^2}{4} + m$$
$$op^{(\cdot)} \rightarrow \frac{m^4}{2} + m^3 + \frac{11m^2}{2} + 5m - 1$$
$$op^{(\div)} \rightarrow m^2 + 1$$

Para fins de simplificação, adota-se $m = n+n_e$, com $n_\theta = 1/2(n+n_e)(n+n_e+1)$, onde n corresponde ao número de estados e n_e o número de ações de controle. O passo de melhoria de política apresenta o mesmo custo para todos os algoritmos, o qual foi descrito anteriormente, e determinado como $\frac{1}{3}n_e^3 + 2n_e^2 n - n_e n$ *flops*. Substituindo o valor de n por 4 e n_e por 1 para um sistema de quarta ordem com uma entrada de controle, o número de *flops* necessários a execução de cada algoritmo é contabilizado, e o resultado é descrito na Tabela 5.3.

A análise dos dados obtidos corrobora a afirmação de que o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR necessita de um número menor de $flops$ para sua execução. Isso pode ser explicado pelo uso direto de matrizes esparsas (triangular e diagonal) no núcleo de operações do algoritmo. Tais matrizes apresentam uma quantidade significativa de elementos cujos valores são iguais a zero, o que é tomado como vantagem, já que o algoritmo deixa de realizar essas multiplicações por zero, impactando positivamente no custo computacional do algoritmo.

Os resultados referentes ao tempo de execução são apresentados na Tabela 5.4, onde os algoritmos são implementados em M-código, usando MATLAB® 2.8 (64 Bits) em um microcomputador com processador 4-core e frequência de 2.1 GHz, 4 GB de memória RAM, e performance computacional de 67.2 Gigaflop/s.

Tabela 5.3: Resultado da avaliação de custo computacional

Algoritmo	*flops*
RLS$_\mu$-ADHDP-DLQR	7632
RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR	3329
RLS$_\mu$-UDU^T-ADHDP-DLQR	849

Tabela 5.4: Tempo de execução das operações/algoritmos

Algoritmo	Tempo em *ns*
$op^{(+/-)}$	0.014
$op^{(\cdot)}$	0.0784
$op^{(\div)}$	0.084
RLS$_\mu$-ADHDP-DLQR	106.848
RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR	46.606
RLS$_\mu$-UDU^T-ADHDP-DLQR	11.886

É possível concluir através dos dados obtidos que os algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR, proposto nesse trabalho, realizam significativamente menor esforço computacional, que o algoritmo padrão RLS$_\mu$-ADHDP-DLQR. Nos passos que compõem a rede crítica adaptativa a estimação RLS$_\mu$-$\mathcal{QR}$ apresenta redução de custo computacional de 54.73% sendo 2.3 vezes mais rápido que a estimação padrão, e a abordagem RLS$_\mu$-UDU^T apresenta um custo 88.9% menor e 9 vezes mais rápido que o apresentado pelo algoritmo sem decomposição.

Isto é explicado pela falta de operações matriciais (Inversão, Multiplicação) de grande ordem em algoritmos com transformações ortogonais. Assim, pode-se concluir a partir dos resultados apresentados que os algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR, têm custo computacional inferior ao custo do algoritmo RLS$_\mu$-ADHDP-DLQR.

5.3 Estabilidade Numérica

Em abordagens convencionais, cuja estimação dos parâmetros ocorre via algoritmos RLS, a degradação de exatidão numérica é um problema nativo ao método e que está associado geralmente ao mal condicionamento da matriz de covariância $\mathbf{\Gamma}_k$ (POTTER, 1963). Tal comportamento instável do RLS é resultante de imprecisões numéricas devido ao uso de ambientes

de precisão finita, e pode acarretar na perda de positividade da matriz $\mathbf{\Gamma}_k$, esse fenômeno é definido como "instabilidade numérica". Afim de solucionar tal problema diferentes procedimentos são sugeridos para melhorar a exatidão numérica e preservar a positividade da matriz de covariância do RLS. Em Bierman (1977) são discutidas maneiras de se implementar algoritmos RLS em ambientes de precisão finita para a melhoria de propriedades numéricas. Formas distintas de análise foram desenvolvidas em diversos trabalhos para explicar a origem e a propagação do fenômeno da instabilidade numérica, (LIAVAS; REGALIA, 1999; LJUNG; LJUNG, 1985; SLOCK, 1992; VERHAEGEN, 1989).

Um método alternativo para solucionar problemas de convergência e instabilidade numérica inerentes a abordagem RLS envolve a redução da matriz de covariância à forma de fatoração UDU^T. Em (GOLUB; LOAN, 2012; RÊGO; NETO; FERREIRA, 2013; WILKINSON, 1968) constata-se que o uso da fatoração UDU^T garante propriedades numéricas desejáveis. Como proposto, esse trabalho faz uso da decomposição $\mathcal{QR}$, assim como da fatoração UDU^T, com o objetivo de superar os problemas resultantes da abordagem padrão do RLS.

O problema da convergência e instabilidade numérica para a estimação RLS é abordado nesta seção em termos das avaliações do número de condição da matriz de covariância $\mathbf{\Gamma}_{(\cdot)}$ da estimativa RLS e do fator Cholesky da matriz $\mathbf{\Phi}_{(\cdot)}$, assim como do parâmetro de positividade.

5.3.1 O Número de Condição

O número de condição da matriz de covariância Γ_k é definido pela relação entre o maior σ_{max} e o menor σ_{min} valores singulares de Γ_k que é dado por

$$cond(\mathbf{\Gamma}_k) = \frac{\sigma_{max}(\mathbf{\Gamma}_k)}{\sigma_{min}(\mathbf{\Gamma}_k)}. \tag{5.1}$$

Os métodos aqui propostos, buscam superar o problema de instabilidade numérica para o algoritmo RLS$_\mu$-ADHDP-DLQR usando transformações unitárias numericamente estáveis a cada iteração do algoritmo RLS. De modo singular, a matriz $\mathbf{\Gamma}_{(\cdot)}$ é propagada numa forma de raiz quadrada usando a fatoração de Cholesky

$$\mathbf{\Gamma}_k = \mathbf{\Gamma}_k^{1/2}\mathbf{\Gamma}_k^{H/2}, \tag{5.2}$$

em que $\mathbf{\Gamma}_k^{1/2}$ é uma matriz triangular inferior, e $\mathbf{\Gamma}_k^{H/2}$ é sua transposta Hermitiana. Métodos de fatoração UDU^T são do tipo raiz quadrada no sentido que os fatores raiz quadrada são propa-

gados, ao invés da própria matriz Γ_k. Essa mudança torna as matrizes melhores condicionadas como pode ser visto em (BIERMAN, 1977; RÊGO; NETO; FERREIRA, 2013).

Deve-se ressaltar que o produto matricial $\mathbf{\Gamma}_k^{1/2}\mathbf{\Gamma}_k^{H/2}$ não assume a condição de semi-definido positivo ou indefinido, uma vez que o produto de qualquer matriz quadrada por sua transposta Hermitiana é sempre definida positiva. Assim, mesmo sob a presença de erros de arredondamento, o condicionamento numérico do fator Cholesky $\mathbf{\Gamma}_{(k)}^{1/2}$ apresenta-se, de modo geral, em melhor estado do que a da própria matriz de covariância $\mathbf{\Gamma}_k$

Uma relação entre os números de condição da matriz de covariância Γ_k com a do seu fator Cholesky $\Gamma_k^{1/2}$ é dada da seguinte maneira:

$$\begin{aligned} cond(\mathbf{\Gamma}_k) &= cond(\mathbf{\Gamma}_k^{1/2}\mathbf{\Gamma}_k^{H/2}) \\ &= [cond(\mathbf{\Gamma}_k^{1/2})]^2. \end{aligned} \tag{5.3}$$

A partir da relação (5.3), observa-se que a diferença entre os números de condição das matrizes $\Gamma_k^{1/2}$ e Γ_k torna-se significativa à medida que $cond(\Gamma_k)$ aumenta. Tal resultado acarreta em uma melhor robustez numérica dos métodos *raiz quadrada*. Em (GOLUB; LOAN, 1996; WILKINSON, 1968) mais detalhes podem ser vistos sobre estabilidade computacional e exatidão numérica para os métodos propostos.

5.3.2 Parâmetro de positividade

A estabilidade numérica do algoritmo RLS$_\mu$-ADHDP-DLQR também pode ser avaliada mediante análise do parâmetro de positividade ρ. O qual é definido como

$$\rho_k = 1 - \boldsymbol{L}_k^T \boldsymbol{\phi}_k. \tag{5.4}$$

O uso da definição do vetor de ganho L_k dado pela Eq.(2.44), assim como o fato de Γ_k ser simétrica, isto é $\Gamma_k^T = \Gamma_k$, após simplificações a Eq.(5.4) pode ser reescrita como

$$\rho_k = \frac{\mu}{\mu + \boldsymbol{\phi}_k^T \mathbf{\Gamma}_{k-1} \boldsymbol{\phi}_k} \tag{5.5}$$

Observando que a forma simétrica na Eq.(5.5) possui uma valor real não-negativo, tem-se $0 < \rho \leq 1$. Portanto, uma condição necessária para que a matriz Γ_k seja definida positiva é dada por $0 < \rho \leq 1$. A ocorrência de um valor fora desta faixa indica que a matriz $\mathbf{\Gamma}_k$ perdeu a

propriedade definida positiva.

Em (ROMANO, 1995) o parâmetro de positividade ρ_k é interpretado como uma razão entre os erros *a priori* e *a posteriori* associados à estimação dos parâmetros de $\boldsymbol{\theta}$. Sendo utilizado como forma de prever a ocorrência de divergências causadas por erros de quantização.

5.4 Tempo de Convergência

A preocupação com o tempo tornou-se um dos paradigmas dominantes na sociedade da informação, visto que um número crescente de aplicações de importância para a sociedade atual, cuja rapidez nas decisões nas comunicações e nas atividades em geral são indispensável, apresentam abordagens baseadas em restrições temporais. Alguns exemplos dessas aplicações se encontram no controle de plantas industriais, de tráfego aéreo ou ferroviário, nas telecomunicações, na eletrônica embarcada em carros e aviões, na robótica, em sistemas de multimídia, dentre outras.

Essas aplicações, que apresentam a característica adicional de estarem sujeitas a restrições temporais, são agrupadas no que é normalmente identificado como Sistemas de Tempo Real, ou seja, um sistema computacional que deve reagir a estímulos oriundos do seu ambiente em prazos específicos (BUTTAZZO, 2011). Dessa forma tais aplicações exigem uma demanda de algoritmos de suporte computacional e de metodologias para sua implementação que garantam requisitos sob restrições temporais, que são verdadeiros desafios aos projetistas desse tipo de sistemas.

Como a noção de tempo em sistemas computacionais é difícil de ser expressa, o enfoque adotado neste trabalho corresponde ao Tempo de Convergência de um algoritmo, sendo esse o tempo decorrido necessário para um algoritmo encontrar uma solução aceitável, onde o erro de regime tem valor menor que o erro de projeto.

Sendo o tempo de convergência diretamente relacionado à capacidade computacional da máquina que está executando esse algoritmo, em casos que não se pode controlar essa velocidade de processamento esse parâmetro deve ser medido por iterações necessárias para encontrar a solução. Assim, quanto menos iterações o algoritmo precisar para encontrar a solução, menor será seu tempo de convergência e mais rápido será sua resposta para o sistema.

A rapidez nos cálculos visa melhorar o desempenho de um sistema computacional, minimizando o tempo médio de resposta para um conjunto de tarefas, enquanto que o objetivo de um cálculo em tempo real é o atendimento dos requisitos temporais das atividades de processa-

mento caracterizadas nesse sistema (STANKOVIC, 1988).

Um conjunto de fatores ligados à arquitetura de hardware, ao sistema operacional e às linguagens de programação são também importantes. Ou seja, os tempos gastos, no pior caso, na execução de códigos da aplicação (tempo de computação) e em funções de suportes devem ser perfeitamente conhecidos ou determinados previamente. Esses tempos limites são necessários nas análises e verificações do comportamento temporal do sistema de tempo real.

A previsibilidade é o requisito necessário que deve ser introduzido em paradigmas de computação clássica para se poder atender aplicações de tempo real, em particular quando estas apresentam requisitos temporais. Para um sistema de tempo real ser considerado previsível no domínio temporal, independente de possíveis variações, o comportamento do sistema deve ser antecipado antes de sua execução. Ou seja, o sistema é previsível quando podemos antecipar que todos os prazos estabelecidos a partir das interações com o ambiente serão atendidos, só assim pode-se, então, almejar realizar controle em tempo real.

5.5 Considerações Finais

Neste Capítulo foram apresentados conceitos de complexidade computacional com enfoque no estudo dos parâmetros de avaliação: Custo Computacional, Estabilidade Numérica e Tempo de Convergência, que servem de métricas para a análise dos algoritmos apresentados no Capítulo 4, com o intuito de que os algoritmos propostos detenham características necessárias a sua aplicação em sistemas de tempo real, sendo esses sistemas microprocessados como: FPGA's (Field Programmable Gate Array), DSP's (Digital Signal Processor) e PSoC's (Programmable System-on-Chip).

6

EXPERIMENTOS COMPUTACIONAIS

6.1 Introdução

Os procedimentos para avaliar o desempenho dos algoritmos RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR para o projeto *online* de controladores ótimos via ADHDP foram estabelecidos levando em consideração a inicialização do processo iterativo, bem como a política de referência, que é estabelecida via solução *offline* da equação HJB-*Riccati* pelo método de Schur. As avaliações verificam o comportamento da convergência dos estimadores RLS$_\mu$-ADHDP, tendo como referência a complexidade computacional e a estabilidade numérica dos estimadores acima mencionados.

Assim, os testes computacionais foram realizados via MATLAB® para os algoritmos propostos nesse estudo e, em seguida, foram realizadas comparações para avaliar a exatidão das soluções fornecidas por esses algoritmos. As políticas de referência para avaliar a exatidão das soluções aproximadas dos controladores (políticas de decisões) são estabelecidas pelo método de Schur, que fornece os valores verdadeiros dos parâmetros associados à solução da equação HJB-*Riccati*, (LAUB, 1979). Tais valores são usados para mostrar que a solução aproximada da equação HJB-*Riccati* possui a habilidade não somente para alcançar uma solução suficientemente próxima da solução exata, mas também alcançar a solução exata da equação HJB-*Riccati*.

Com o intuito de garantir que o desempenho dos algoritmos RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR sejam comparados de forma plausível no que diz respeito a variações no valor do fator de esquecimento, os outros parâmetros foram mantidos constantes e iguais para todos os algoritmos. Por exemplo, para os valores iniciais das matrizes de correlação e de correlação inversa (covariância), uma equivalência é

mantida de acordo com a seguinte relação

$$\mathbf{\Gamma}_k \sim \mathbf{\Phi}_k^{1/2}.$$

Os algoritmos têm uma condição de revitalização de estado devido aos problemas de estado nulo que levam a uma matriz de regressores com *rank* nulo. Neste caso, a revitalização é aplicada periodicamente após cada intervalo n_{revit}, que é dada pela dimensão do vetor de regressão ϕ, que muda de acordo com a ordem do sistema.

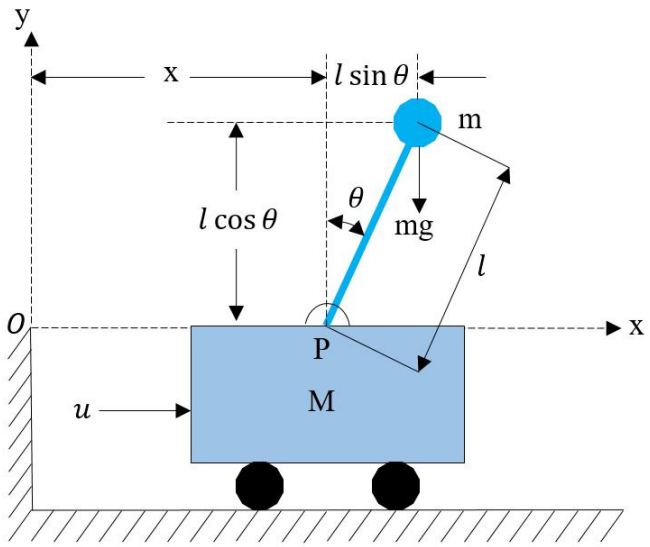
Os experimentos de simulação são apresentados para dois sistemas dinâmicos multivariáveis, um SIMO e outro MIMO, que avaliam a funcionalidade dos estimadores RLS baseados em decomposições $\mathcal{QR}$ e UDU^T da função valor de ação para o projeto *online* do sistema de controle DLQR via ADHDP, e verificam a estabilidade e convergência dos algoritmos RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR. No primeiro sistema dinâmico os resultados foram explorados em um sistema de quarta ordem para o projeto de estabilização de um pêndulo invertido montado sobre uma base móvel (carrinho) (CHEN, 1995; PRASAD; GUPTA; TYAGI, 2014). Por fim, um modelo de sexta ordem para o controle de um Helicóptero 3-DOF Quanser (helicóptero com três graus de liberdade) (LOPES, 2007) é apresentado.

6.2 Modelo do Pêndulo Invertido sobre Base Móvel

O sistema do pêndulo invertido montado sobre uma base móvel (carrinho) tem sua configuração apresentada na Fig.6.1. O sistema consiste de um carrinho de massa M que se movimenta sobre uma superfície horizontal no eixo x, com uma esfera de massa m na extremidade de uma haste rígida, de massa desprezível e comprimento l. O carrinho e a esfera foram tratados como massas pontuais, com o pivô no centro do carrinho. São desconsiderados o atrito e a resistência do ar. Apresenta-se como entrada uma força horizontal u aplicada à base móvel. Os outros sinais apresentados são o ângulo θ e a posição da esfera $y = x + l\sin\theta$.

As Leis de *Newton* para o movimento, mais especificamente a segunda Lei de *Newton*, descrevem as forças que agem sobre a base móvel assim como fornece o equilíbrio de forças inerentes ao movimento rotacional do pêndulo. Considera-se o diagrama de corpo livre mostrado na Fig.6.1, e assume-se que as coordenadas do centro de gravidade do pêndulo, $(x_G,\ y_G)$,

Figura 6.1: Diagrama esquemático de um Sistema de Pêndulo Invertido de 4ª Ordem.



são dadas por

$$x_G = x + l\sin\theta \;\text{ e }\; y_G = l\cos\theta, \tag{6.1}$$

sendo l o comprimento do pêndulo até seu centro de gravidade e x a coordenada da posição do carrinho.

Pelo movimento horizontal do carro, a segunda Lei de Newton do movimento, descreve as forças que agem sobre o carro, como

$$\sum F = M\frac{d^2x}{dt^2} + m\frac{d^2x_G}{dt^2}. \tag{6.2}$$

Para a segunda equação do movimento considera-se todas as forças agindo sob a massa do pêndulo, o emprego da segunda lei de Newton nos fornece o equilíbrio de forças no caso do movimento rotacional do pêndulo

$$\sum m_G = mlF_x\cos\theta - mlF_y\sin\theta. \tag{6.3}$$

O comportamento do sistema é descrito pelo seguinte conjunto de equações diferenciais não-lineares

$$(M+m)\ddot{x} + ml(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta) = u, \tag{6.4}$$

$$m(\ddot{x}\cos\theta + l\ddot{\theta}) = mg\sin\theta. \tag{6.5}$$

Estas podem ser linearizadas em torno do ponto de equilíbrio $(x, \theta) = (0, 0)$

$$(M+m)\ddot{x} + ml\ddot{\theta} = u, \tag{6.6}$$

$$\ddot{x} + l\dot{\theta} - g\theta = \frac{1}{m}. \tag{6.7}$$

Com o estado definido como $x = [\theta\ \dot{\theta}\ x\ \dot{x}]^T$, segue que: $\dot{x}_1 = \dot{\theta},\ \dot{x}_2 = \ddot{\theta},\ \dot{x}_3 = \dot{x}$ e $\dot{x}_4 = \ddot{x}$. Dessa forma, x_1 e x_3 são variáveis de estado que representam as posições angular e linear, e as variáveis x_2 e x_4 são as velocidades, angular e linear, respectivamente.

A descrição em espaço de estados correspondente ao modelo das Eqs.(6.6) e (6.7) é dado por:

$$\dot{x} = Ax + Bu,\ \ k \geq k_0, \tag{6.8}$$

sendo A a matriz de estado,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{(M+m)g}{Ml} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-mg}{M} & 0 & 0 & 0 \end{bmatrix}, \tag{6.9}$$

a ponderação B do vetor de entrada e a matriz de saída C são dadas, respectivamente, por

$$B = \begin{bmatrix} 0 \\ \frac{-1}{Ml} \\ 0 \\ \frac{1}{M} \end{bmatrix} \text{ e } C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6.10}$$

As matrizes do sistema dinâmico para os valores dos parâmetros $M = 2\,kg$, $m = 0.1\,kg$,

$l = 0.5\ m$; $g = 9.81\ m/s^2$ são dadas por

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 20,601 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -0,4905 & 0 & 0 & 0 \end{bmatrix}, \text{ e } B_c = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} \tag{6.11}$$

Já as matrizes do sistema discretizado para o intervalo de amostragem $T_{amost} = 0.01s$ são obtidas pelo método de discretização do segurador de ordem zero - ZOH (do inglês *Zero-order hold*).

Condições Iniciais e *Setup* da Simulação

Para a implementação dos algoritmos RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR as condições iniciais adotadas foram: uma política inicial K_0 admissível, fator de desconto $\gamma = 1$, estado do sistema $x_0 = [-0.13\ 0.13\ 0.005\ 0.01]^T$ e matrizes quadráticas da função de custo respectivamente dadas por $Q = [10^2\ 0\ 0\ 0; 0\ 1\ 0\ 0; 0\ 0\ 1\ 0; 0\ 0\ 0\ 1]$ e $R = 1$. Os parâmetros que inicializam o estimador RLS proposto são dados pelo vetor $\boldsymbol{\theta}_0 = 10^3[1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1]^T$ e matriz $\Gamma_0 = 10^3 I_{15\times 15}$, sendo I uma matriz identidade de ordem n_θ.

Para as simulações realizadas, a influência do fator de esquecimento μ pôde ser verificada no processo de convergência e na estabilidade numérica dos estimadores RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR para as soluções aproximadas da equação HJB-*Riccati* com diferentes valores de μ. Dentre os diversos fatores testados, escolheu-se considerar os valores 0.634, 0.739, 0.824, 0.878 e 0.934 de μ para a fase de simulações e discussões.

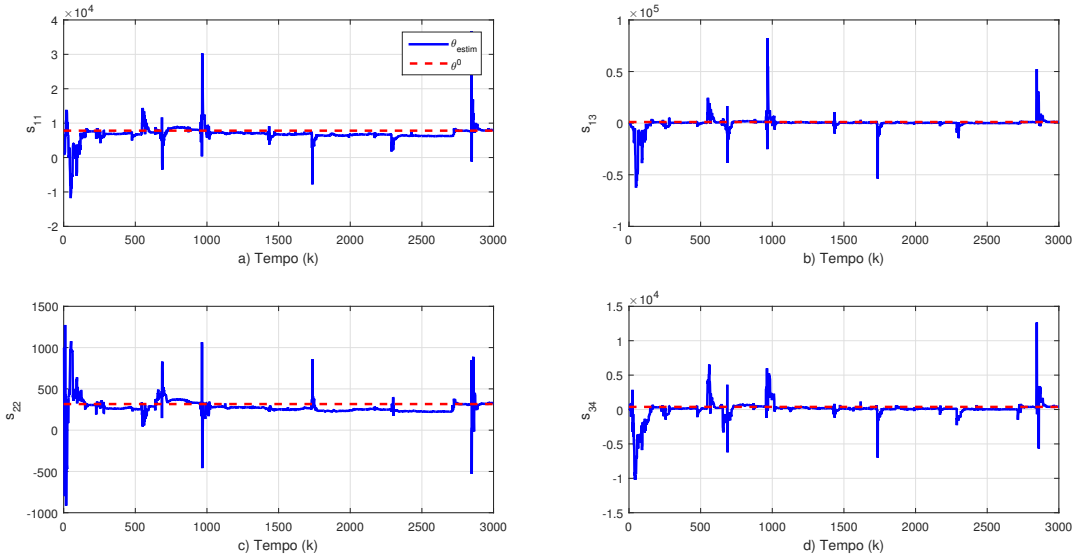
6.2.1 Simulação com $\mu = 0.634$

RLS$_\mu$-ADHDP-DLQR

A evolução do processo iterativo para a solução da equação HJB-*Riccati* é ilustrada na Fig.6.2 para um ciclo de 3000 iterações, com fator de esquecimento $\mu = 0.634$, para o algoritmo RLS$_\mu$-ADHDP-DLQR. As curvas (a)-(d) da Fig.6.2 representam o comportamento de convergência dos elementos $s_{11},\ s_{13},\ s_{22}$ e s_{34} da matriz S correspondentes respectivamente aos

componentes $\theta_1,\ \theta_3,\ \theta_6$ e θ_{11} do vetor de parâmetros $\boldsymbol{\theta}$. Percebe-se claramente que os elementos não convergem para o valor de referência, apresentando constantes oscilações dos valores estimados. Tal comportamento deve-se a uma maior sensibilidade aos efeitos de perturbação causados por problemas de estabilidade numérica.

Figura 6.2: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.634$ - RLS$_\mu$-ADHDP-DLQR.

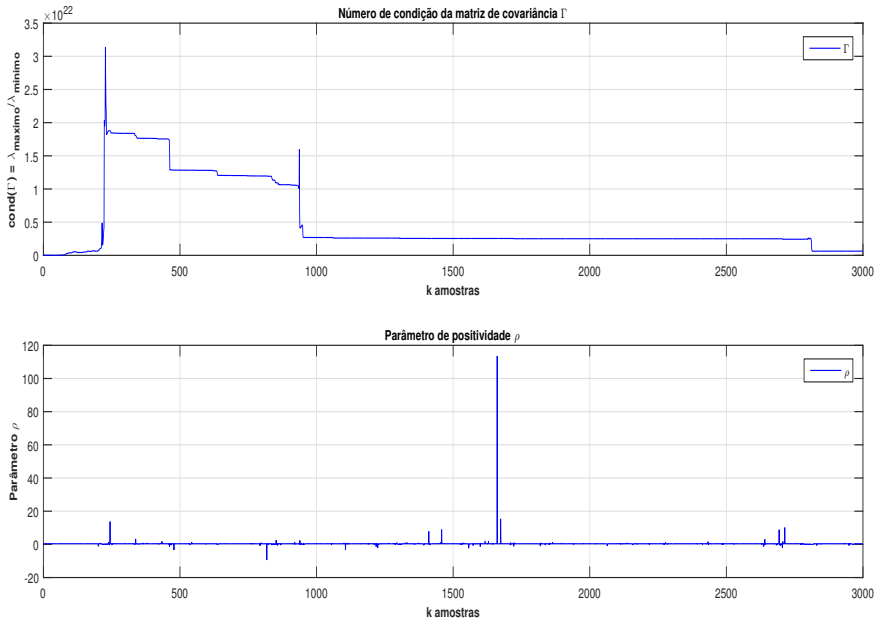


A Fig.6.3 mostra o número de condição da matriz de covariância $\boldsymbol{\Gamma}_k$ e o parâmetro de positividade ρ do processo RLS$_\mu$-ADHDP-DLQR para a estimação em questão. Observa-se que os valores de ρ não se encontram dentro da faixa de referência, $0 < \rho < 1$, apresentando suas primeiras oscilações fora de faixa por volta da iteração 200 a 250, mesmo instante de tempo em que o número de condição da matriz $\boldsymbol{\Gamma}_k$ apresenta um pico considerável em seus valores. Essas duas condições corroboram os problemas de convergência sofridos pelos elementos s_{ij} da matriz S, indicando por sua vez a perda de características de estabilidade numérica da estimação.

RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

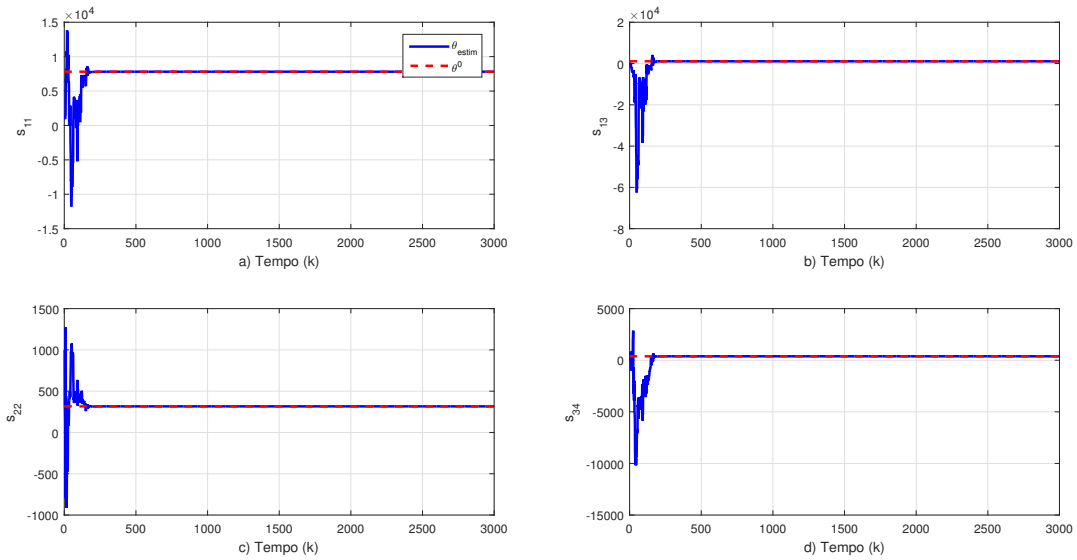
Na Fig.6.4 apresenta-se a evolução do processo iterativo da solução da equação HJB-$Riccati$ para um ciclo de 3000 iterações com fator de esquecimento de $\mu = 0.634$ via algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. As curvas (a)-(d) da Fig.6.4 representam o comportamento de convergência dos elementos $s_{11},\ s_{22},\ s_{33}$ e s_{44} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_6,\ \theta_{10}$ e θ_{13} do vetor de parâmetros $\boldsymbol{\theta}$. Observa-se que o sistema entra em

Figura 6.3: Número de condição da matriz de covariância $\mathbf{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento $\mu = 0.634$ - RLS$_\mu$-ADHDP-DLQR.



fase de acomodação por volta da iteração 400, de modo a não apresentar perturbações quando comparado ao valor de referência (solução Schur) e a solução aproximada dada pelo algoritmo padrão.

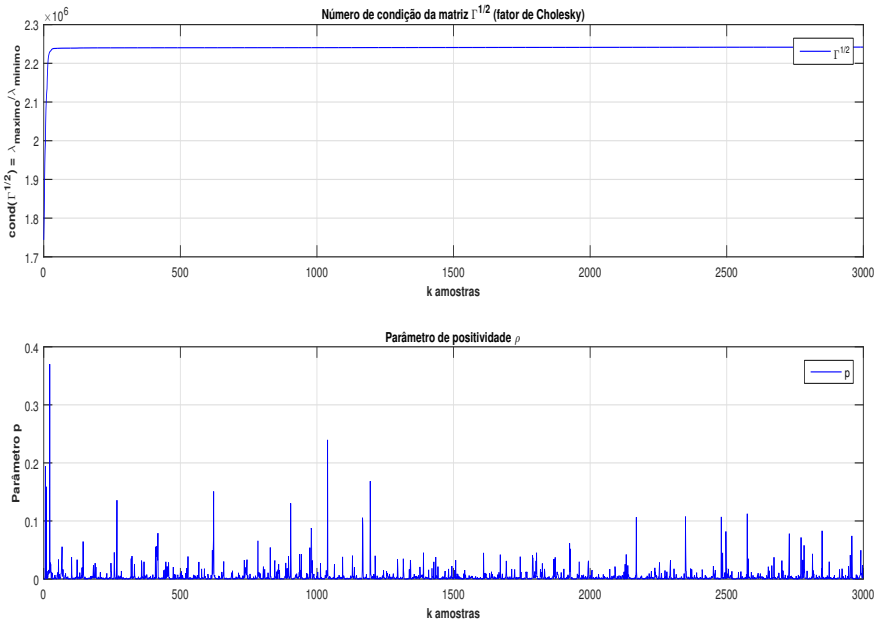
Figura 6.4: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.634$ - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



Na Fig.6.5, apresentam-se o número de condição do fator Cholesky e o parâmetro de

positividade ρ para o processo de estimação da solução da equação HJB-*Riccati*. Percebe-se pelos dados apresentados que o parâmetro de positividade ρ encontra-se dentro da faixa de valores esperados e que o número de condição do fator Cholesky exibe um comportamento sem mudanças abruptas e com amplitude em uma faixa de ocorrência de 2.25×10^6, valor significativamente menor em relação aos apresentados pela estimação RLS$_\mu$-ADHDP-DLQR.

Figura 6.5: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento $\mu = 0.634$ - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

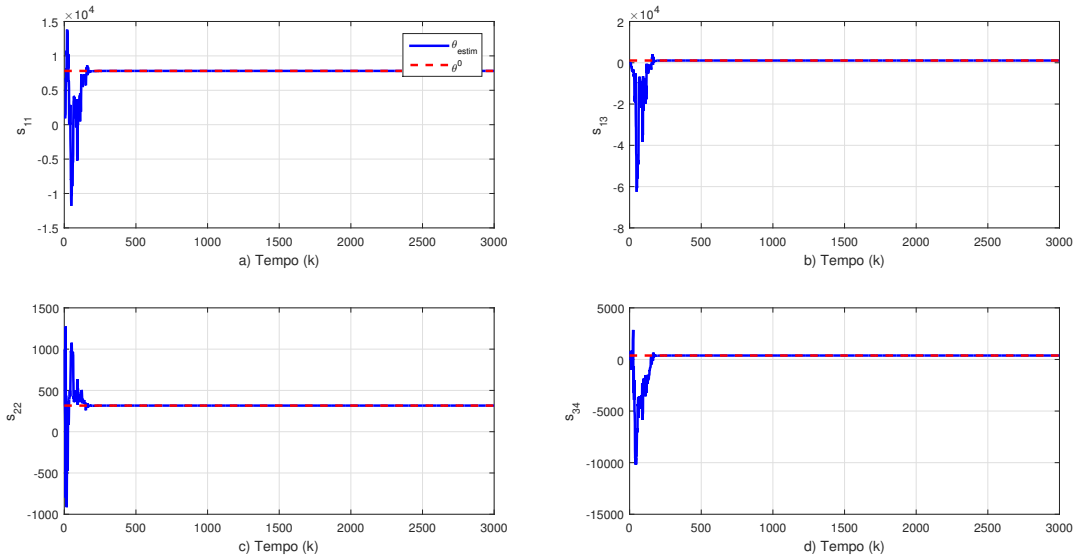


RLS$_\mu$-UDU^T-ADHDP-DLQR

Os resultados dos experimentos para o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR são apresentados também para um ciclo de 3000 iterações, como mostrado na Fig.6.6. Para o fator de esquecimento $\mu = 0.634$, observa-se que o sistema entra em fase de acomodação logo após a iteração 400, com um comportamento muito similar ao apresentado pelos resultados obtidos para a estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR, de modo a não apresentar perturbações quando comparados ao valor de referência (solução Schur).

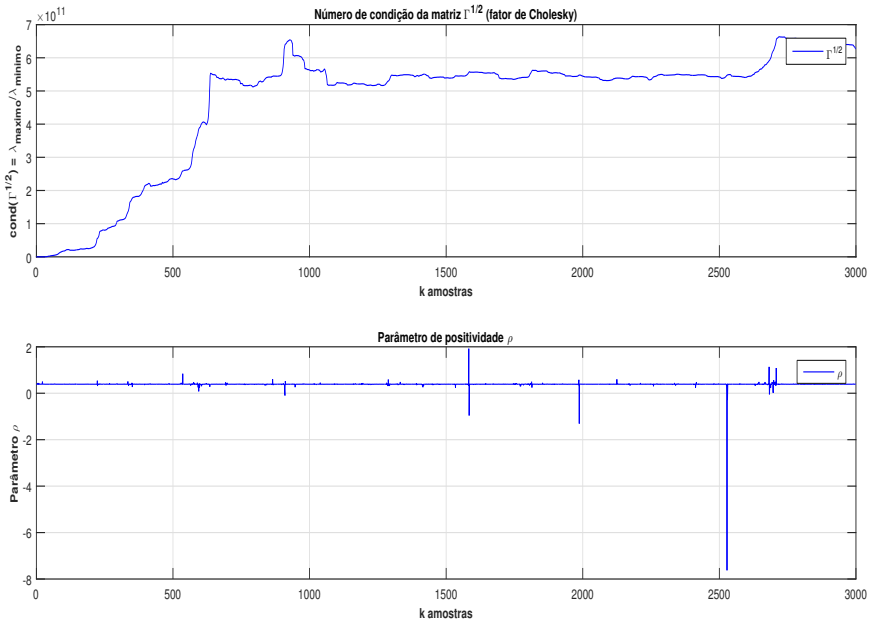
A Fig.6.7 apresenta o número de condição do fator Cholesky e o parâmetro de positividade para o processo de estimação da solução da equação HJB-*Riccati* via algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR. Observa-se que o número de condição do fator Cholesky exibe um comportamento com amplitudes maiores em relação às apresentadas pelas estimações RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR, mas consideravelmente mais baixas do que as evidenciadas pelo algo-

Figura 6.6: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.634$ - RLS$_\mu$-UDU^T-ADHDP-DLQR.



ritmo padrão.

Figura 6.7: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento $\mu = 0.634$ - RLS$_\mu$-UDU^T-ADHDP-DLQR.



A partir da Fig.6.7 percebe-se também que o parâmetro de positividade para o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR apresenta um comportamento fora da faixa de referência, $0 < \rho < 1$, indicando a perda de positividade do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$. Mas mesmo sob tal

condição os parâmetros estimados da matriz S não apresentam perturbações. Em hipótese, isso pode ser explicado, pelo fato de que os erros das estimativas tendem a se anular, a norma do vetor $\boldsymbol{L}_k$ pode assumir uma gama muita ampla de valores sem que ocorram grandes alterações nas componentes do parâmetro $\boldsymbol{\theta}$. Isto faz com que o produto $\boldsymbol{L}_k\xi_k$ permaneça em limites relativamente aceitáveis, mantendo os estimadores indefinidamente estáveis.

Analisando os resultados para o fator de esquecimento $\mu = 0.634$, percebe-se que somente os algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR alcançaram a convergência, ambos em torno da iteração 400, de forma suave, já o algoritmo padrão RLS$_\mu$-ADHDP-DLQR mostrou-se completamente instável, comportamento causado pela falta de estabilidade numérica. A perda de estabilidade numérica também ocorre para a estimação RLS$_\mu$-UDU^T-ADHDP-DLQR, mas tal fato não impede a convergência dos parâmetros do vetor $\boldsymbol{\theta}$, por razões discutidas anteriormente.

6.2.2 Simulação com $\mu = 0.739$

RLS$_\mu$-ADHDP-DLQR

O processo iterativo para a solução da equação HJB-*Riccati* tem sua evolução ilustrada na Fig.6.8 para um ciclo de 3000 iterações, com fator de esquecimento $\mu = 0.739$, para o algoritmo RLS$_\mu$-ADHDP-DLQR. As curvas (a)-(d) da Fig.6.8 representam o comportamento de convergência dos elementos $s_{11},\ s_{13},\ s_{22}$ e s_{34} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_3,\ \theta_6$ e θ_{11} do vetor de parâmetros $\boldsymbol{\theta}$. Percebe-se que os parâmetros estimados tendem a seguir o valor de referência um pouco após a iteração 1500, comportamento que não é mantido, uma vez que os elementos passam a apresentar oscilações leves após a iteração 2000 e grandes picos oscilatórios logo após a iteração 2500. Tal comportamento deve-se a uma maior sensibilidade aos efeitos de perturbação causados por problemas de estabilidade numérica.

O número de condição da matriz de covariância $\boldsymbol{\Gamma}_k$, e o parâmetro de positividade ρ do processo RLS$_\mu$-ADHDP-DLQR para a estimação desejada são apresentados na Fig.6.9. Observa-se que os valores de ρ não se encontram dentro da faixa de referência, $0 < \rho < 1$, o que sugere, e em muitos casos antecede irregularidades na estimação dos elementos do vetor $\boldsymbol{\theta}$, que é o caso desta simulação. O número de condição da matriz $\boldsymbol{\Gamma}_k$ mostrado apresenta comportamento altamente irregular em seus valores.

Figura 6.8: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.739$ - RLS$_\mu$-ADHDP-DLQR.

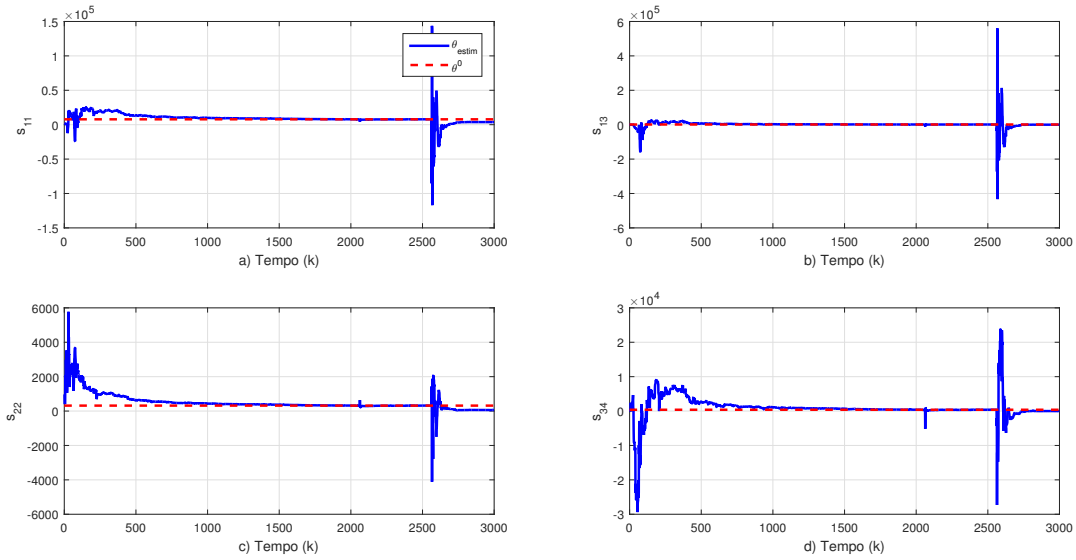
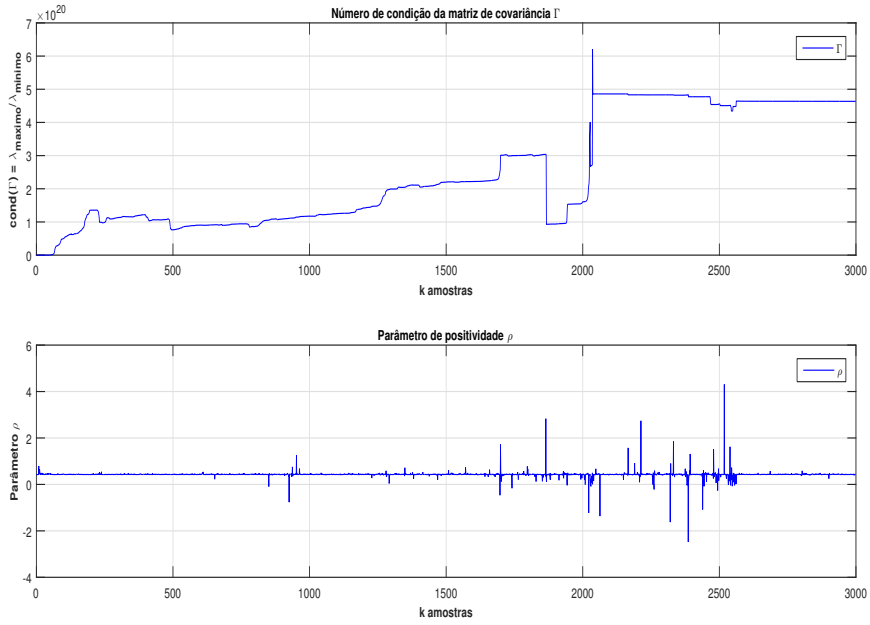


Figura 6.9: Número de condição da matriz de covariância $\mathbf{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento $\mu = 0.739$ - RLS$_\mu$-ADHDP-DLQR.

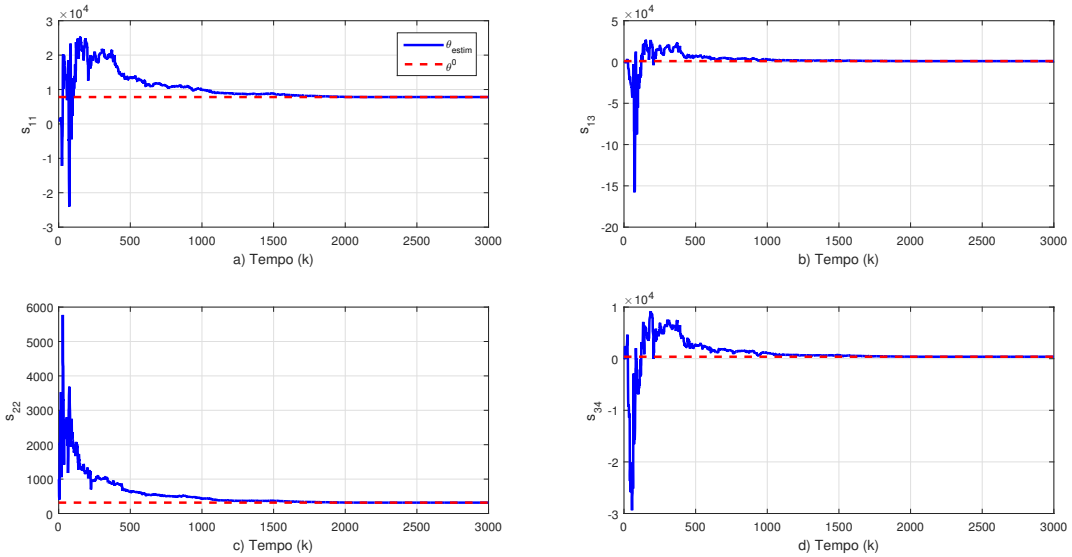


RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

As curvas (a)-(d) da Fig.6.10 representam o comportamento de convergência dos elementos $s_{11},\ s_{22},\ s_{33}$ e s_{44} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_6,\ \theta_{10}$ e θ_{13} do vetor de parâmetros $\boldsymbol{\theta}$ para a evolução do processo iterativo da solução da equação

HJB-$Riccati$ com um ciclo de 3000 iterações, e com fator de esquecimento de $\mu = 0.739$ via estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. Observa-se que o sistema entra em fase de acomodação de maneira suave por volta da iteração 2250, de modo a não apresentar perturbações quando comparados ao valor de referência (solução Schur) e da solução aproximada dada pelo algoritmo padrão.

Figura 6.10: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.739$ - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



Na Fig.6.11 são apresentados o número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e o parâmetro de positividade ρ para o processo de estimação da solução da equação HJB-*Riccati* via RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. Observa-se que tanto o número de condição do fator Cholesky como o fator de positividade exibem um comportamento sem mudanças abruptas, estando ambos em faixas estritas de ocorrência. Para $\mathbf{\Gamma}_k^{1/2}$ esses valores estão estimados na vizinhança de 1.8×10^6, valores significativamente menores em relação aos apresentados pela estimação RLS$_\mu$-ADHDP-DLQR padrão, e para ρ uma faixa de $0 < \rho < 1$.

RLS$_\mu$-UDU^T-ADHDP-DLQR

Os resultados dos experimentos para o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR são apresentados para um ciclo de 3000 iterações, como mostrado na Fig.6.12. Para o fator de esquecimento $\mu = 0.739$, percebe-se um comportamento de convergência dos elementos do vetor $\boldsymbol{\theta}$ estritamente semelhante aos apresentados pela estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

Figura 6.11: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento $\mu = 0.739$ - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

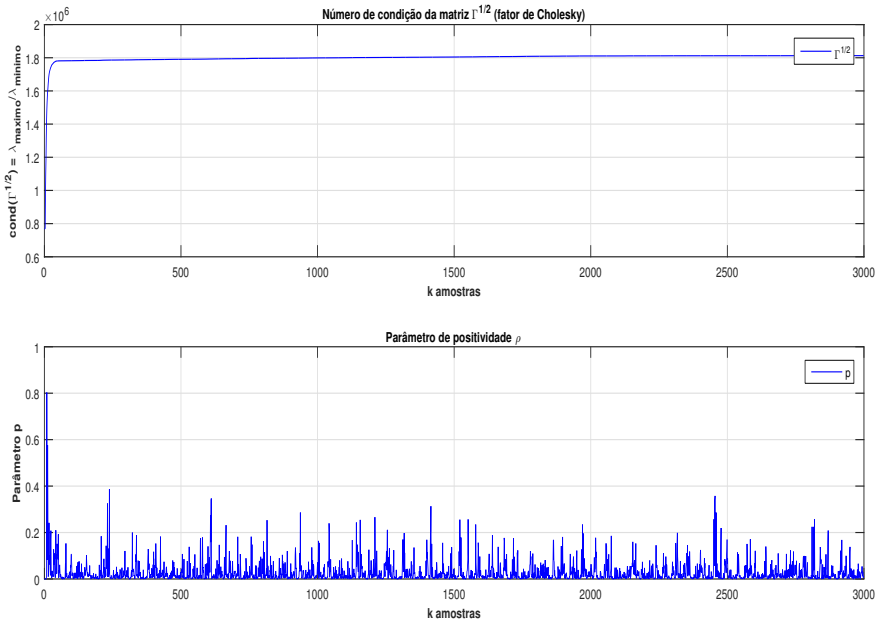
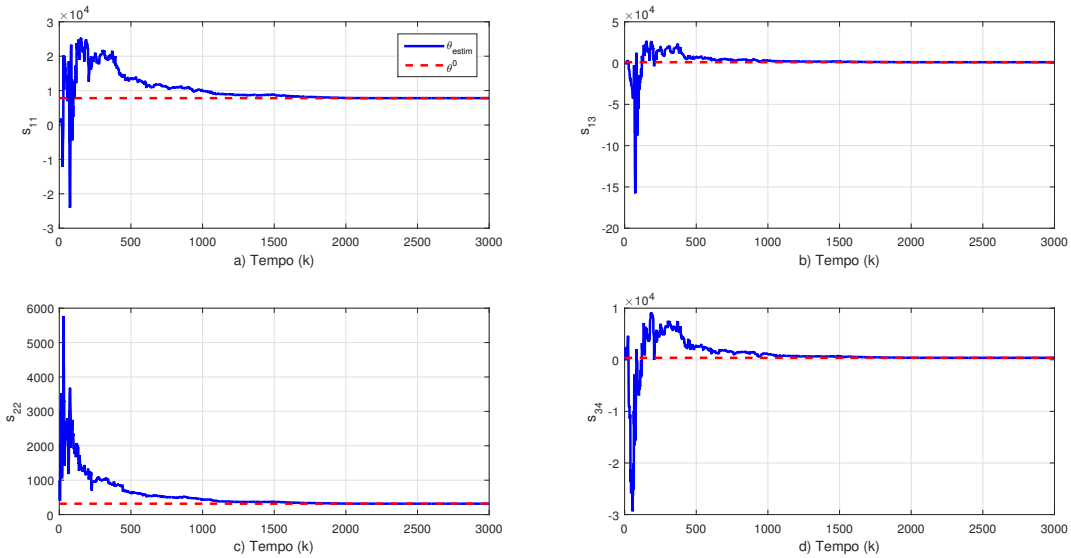


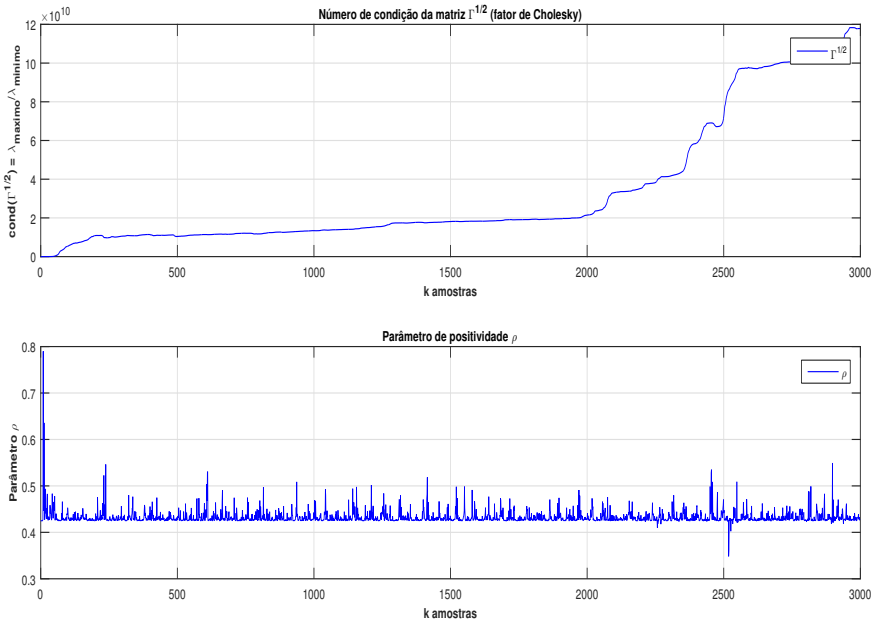
Figura 6.12: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.739$ - RLS$_\mu$-UDU^T-ADHDP-DLQR.



A Fig.6.13 apresenta tanto o número de condição do fator Cholesky quanto o parâmetro de positividade ρ para o processo de estimação da solução da equação HJB-*Riccati* via algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR. Observa-se que o número de condição do fator Cholesky exibe um comportamento com amplitudes menores em relação às apresentadas pela estimação RLS$_\mu$-ADHDP-DLQR, mas com tendência crescente o que pode ou não deixar de ocorrer para o

aumento no número de iterações da estimação. O parâmetro de positividade encontra-se bem comportado para a simulação realizada.

Figura 6.13: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento $\mu = 0.739$ - RLS$_\mu$-UDU^T-ADHDP-DLQR.



Os resultados obtidos para o fator de esquecimento $\mu = 0.739$, nos mostram que somente os algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR alcançaram a convergência, ambos por volta da iteração 2500, de modo suave, em situação contrária ao algoritmo padrão RLS$_\mu$-ADHDP-DLQR que mostrou-se instável, comportamento causado pela perda de estabilidade numérica do método.

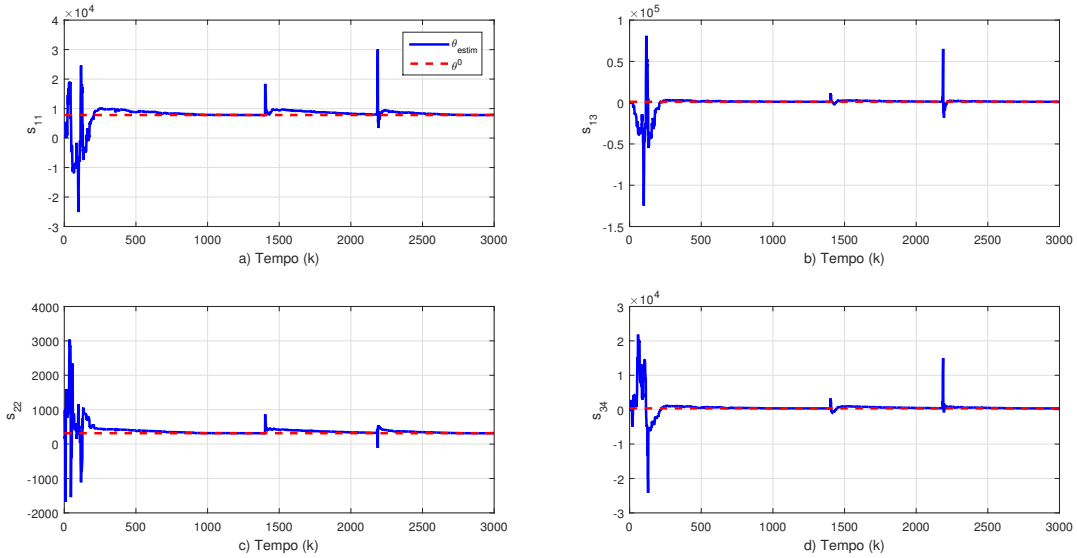
6.2.3 Simulação com $\mu = 0.824$

RLS$_\mu$-ADHDP-DLQR

A implementação do algoritmo RLS$_\mu$-ADHDP-DLQR tem a evolução de seu processo iterativo para a solução da equação HJB-*Riccati* mostrada na Fig.6.14 para um ciclo de 3000 iterações, com fator de esquecimento $\mu = 0.824$. As curvas (a)-(d) da Fig.6.14 representam o comportamento de convergência dos elementos $s_{11},\ s_{13},\ s_{22}$ e s_{34} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_3,\ \theta_6$ e θ_{11} do vetor de parâmetros $\boldsymbol{\theta}$. Nota-se que logo após os parâmetros de S alcançarem a fase de acomodação (por volta da iteração 1000) desajustes significativos ocorrem em seus valores entre as iterações 1200 e 2500. Esse período,

após a estabilização, caracteriza uma fase crítica para a estabilidade numérica do algoritmo, já que neste momento o estimador apresenta uma maior sensibilidade aos efeitos de perturbação causados por problemas numéricos.

Figura 6.14: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.824$ - RLS$_\mu$-ADHDP-DLQR.



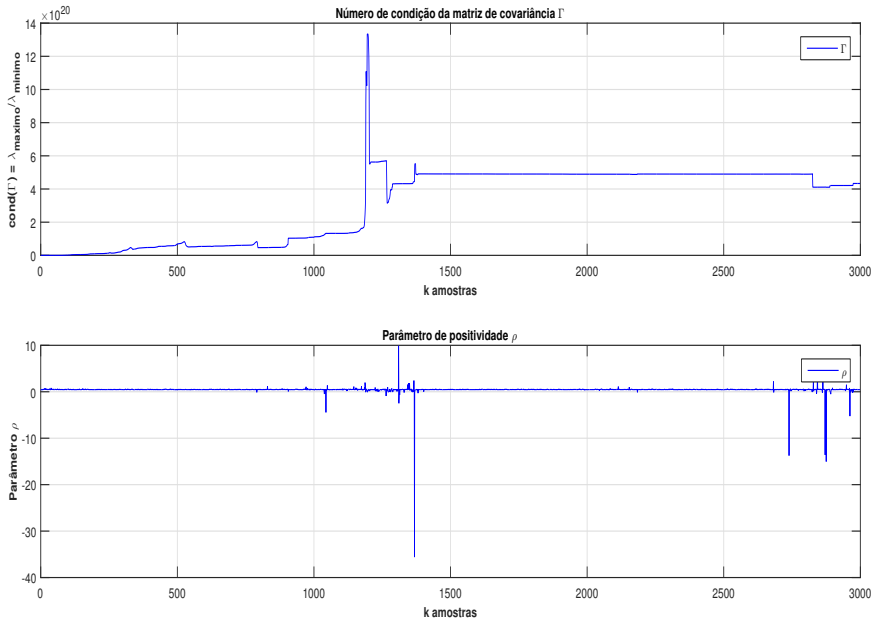
Esse fenômeno pode ser explicado pela observação da Fig.6.15, o número de condição da matriz de covariância Γ_k do estimador RLS$_\mu$-ADHDP padrão apresenta mudanças abruptas em seu valor (a presença de picos que podem levar a situações de não convergência), principalmente em torno das iterações 1200 e 1500, que se reflete nos valores dos parâmetros que em um primeiro momento sofrem perda de estabilidade entre as iterações 1200 e 1500.

Observa-se também que um pouco antes da iteração 1500, o parâmetro de positividade tende a assumir valores dentro do intervalo de validade, para o período de tempo ajustado, a medida em que o estimador tende à recuperar o regime permanente. No entanto pequenos desajustes neste parâmetro logo após a iteração 2100, podem sugerir a ocorrência de perda de estabilidade dos componentes de S.

RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

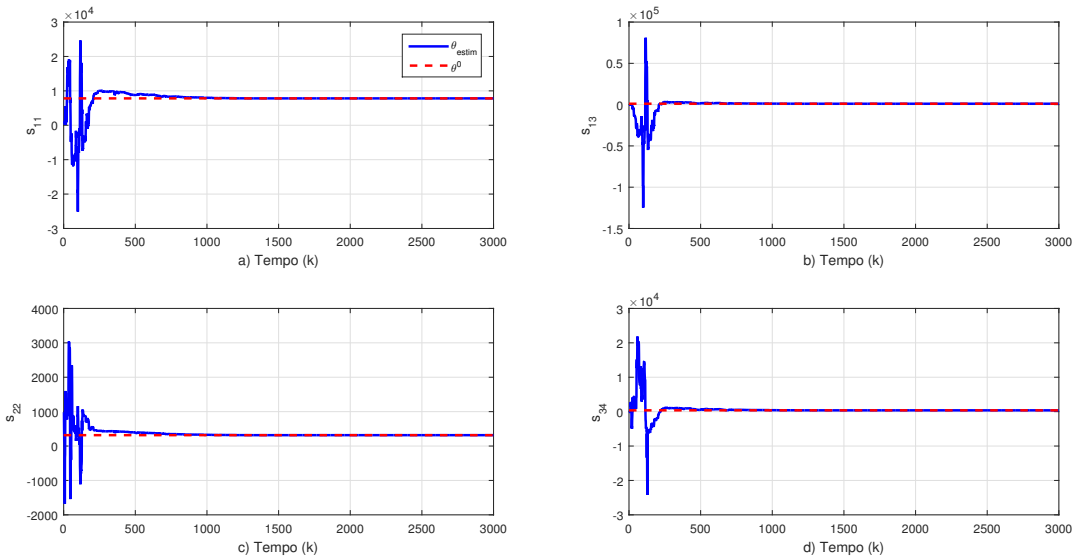
Na Fig.6.16, apresenta-se a evolução do processo iterativo da solução da equação HJB-*Riccati* para um ciclo de 3000 iterações com fator de esquecimento de $\mu = 0.824$. As curvas (a)-(d) da Fig. 6.16 representam o comportamento de convergência dos elementos $s_{11},\ s_{22},\ s_{33}$

Figura 6.15: Número de condição da matriz de covariância $\mathbf{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento $\mu = 0.824$ - RLS$_\mu$-ADHDP-DLQR.



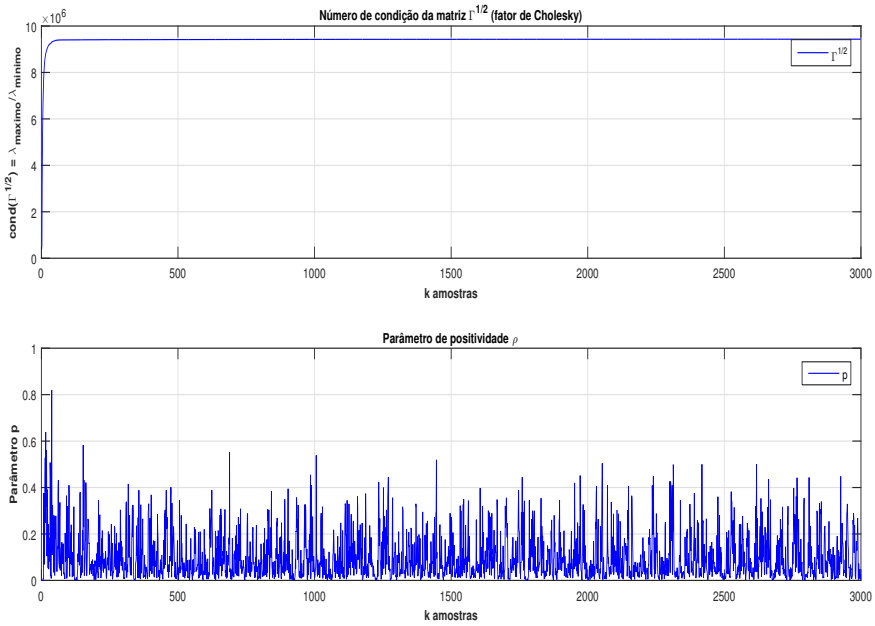
e s_{44} da matriz S correspondentes respectivamente aos parâmetros θ_1, θ_6, θ_{10} e θ_{13} do vetor de parâmetros $\boldsymbol{\theta}$. Observa-se que o sistema entra em fase de acomodação por volta da iteração 1300, de modo a não apresentar perturbações quando comparados ao valor de referência (solução Schur) e a solução aproximada dada pelo algoritmo padrão.

Figura 6.16: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.824$ - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



A Fig.6.17 apresenta o número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e o parâmetro de positividade ρ para o processo de estimação da solução da equação HJB-*Riccati* via RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. Observa-se que o número de condição do fator Cholesky exibe um comportamento sem mudanças abruptas e com amplitudes significativamente menores em relação as apresentadas pela estimação RLS$_\mu$-ADHDP-DLQR padrão, assim como, o parâmetro de positividade encontra-se dentro de faixa de referência.

Figura 6.17: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento $\mu = 0.824$ - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

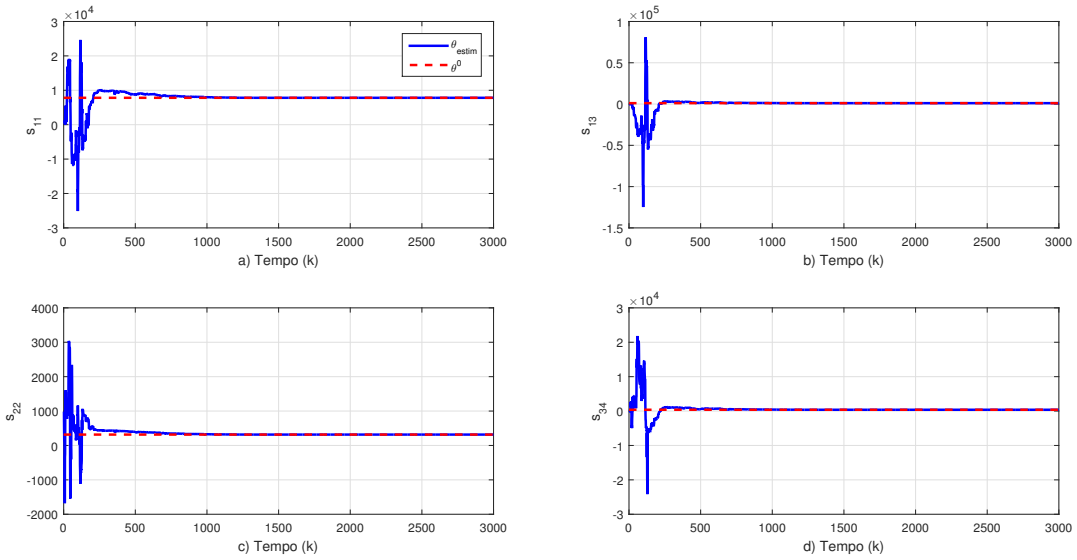


RLS$_\mu$-UDU^T-ADHDP-DLQR

De maneira análoga os resultados dos experimentos para o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR são apresentados para um ciclo de 3000 iterações, como mostrado na Fig. 6.18. Para o fator de esquecimento $\mu = 0.824$, observa-se também que o sistema entra em fase de acomodação logo após a iteração 1300, de modo a não apresentar perturbações quando comparados ao valor de referência (solução Schur).

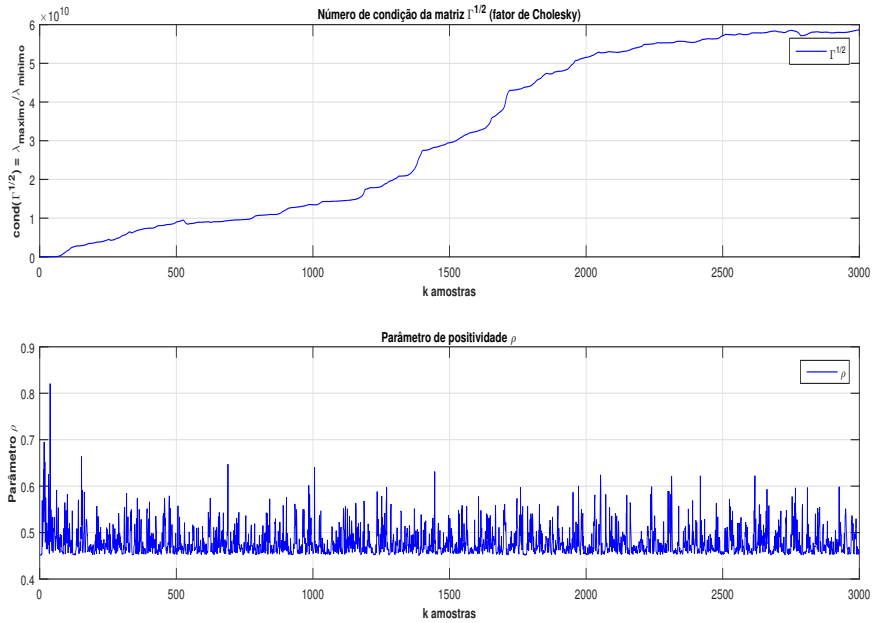
A Fig. 6.19 apresenta o número de condição do fator Cholesky e o parâmetro de positividade para o processo de estimação da solução da equação HJB-*Riccati* via RLS$_\mu$-UDU^T-ADHDP-DLQR. Observa-se que o parâmetro de positividade encontra-se dentro do intervalo $0 < \rho < 1$ e que o número de condição do fator Cholesky exibe um comportamento sem mudanças abruptas, mas ligeiramente elevado quando comparado ao apresentado pela estimação

Figura 6.18: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 3000 iterações, com um fator de esquecimento $\mu = 0.824$ - RLS$_\mu$-UDU^T-ADHDP-DLQR.



RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

Figura 6.19: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de $\mu = 0.824$ - RLS$_\mu$-UDU^T-ADHDP-DLQR.



Comparando os resultados apresentados pelas estimações em questão, para o fator de esquecimento $\mu = 0.824$, percebe-se que todos os algoritmos apresentam uma resposta moderada para tal fator, tendo a estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR um comportamento li-

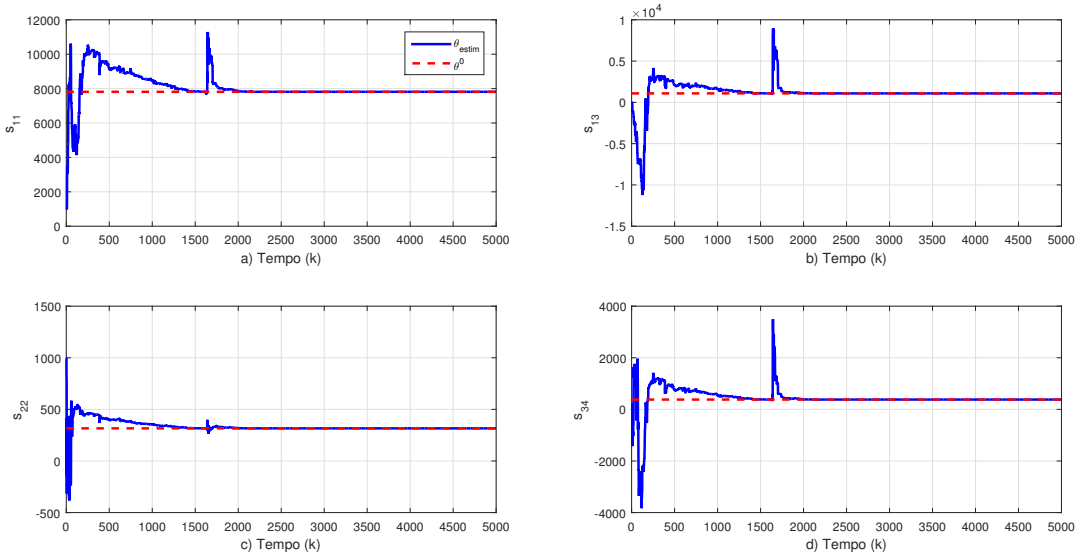
geiramente melhor que RLS$_\mu$-UDU^T-ADHDP-DLQR e superior ao algoritmo padrão RLS$_\mu$-ADHDP-DLQR. Já o número de condição do fator Cholesky $\Gamma_k^{1/2}$ apresenta significativa melhora em relação a matriz de covariância Γ_k adotada no algoritmo padrão.

6.2.4 Simulação com $\mu = 0.878$

RLS$_\mu$-ADHDP-DLQR

O algoritmo RLS$_\mu$-ADHDP-DLQR tem a evolução de seu processo iterativo para a solução da equação HJB-*Riccati* mostrada na Fig.6.20 para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.878$. As curvas (a)-(d) da Fig.6.20 representam o comportamento de convergência dos elementos $s_{11},\ s_{13},\ s_{22}$ e s_{34} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_3,\ \theta_6$ e θ_{11} do vetor de parâmetros $\boldsymbol{\theta}$. Percebe-se que logo após os parâmetros de S alcançarem a fase de acomodação (por volta da iteração 1500) um grande pico ocorre para os valores estimados, entre as iterações 1600 e 2000. Após esse período o processo tende a se estabilizar alcançado novamente a convergência por volta da iteração 2500. De modo a não apresentar sensibilidade aos efeitos de perturbação causados por problemas numéricos.

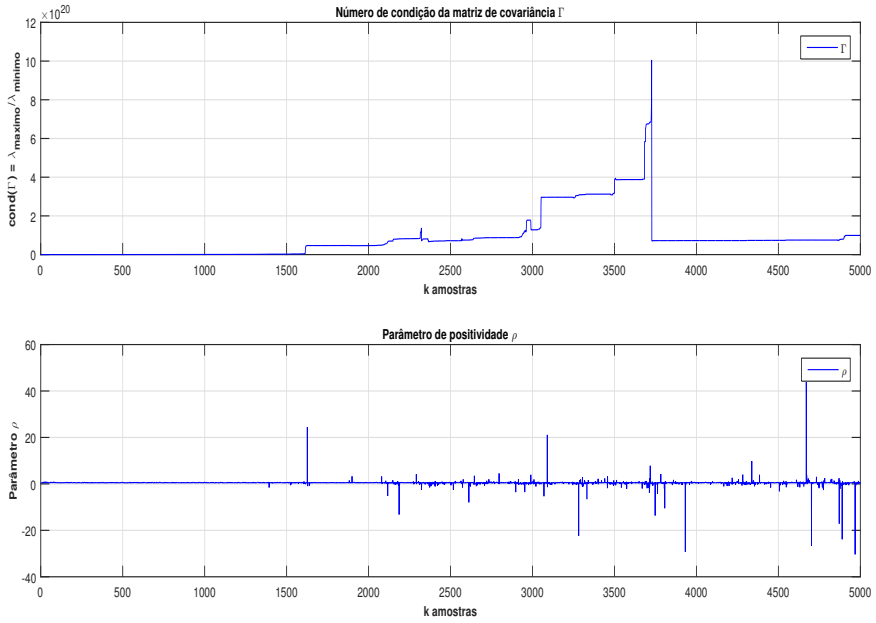
Figura 6.20: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.878 - RLS$_\mu$-ADHDP-DLQR.



Anormalidades no comportamento da matriz de covariância são evidenciadas logo após a iteração 1390, período em que o valor do parâmetro de positividade começa a apresentar grandes oscilações fora de sua faixa de validade, podendo levar a erros numéricos mais altos,

com um grande pico em torno das iterações 1640, como ilustrado na Fig.6.21, logo após esse período os elementos s_{ij} da matriz S apresentam perda de estabilidade. Percebe-se que durante o período que se inicia na iteração 2000 até a iteração 5000 o parâmetro ρ apresenta grande oscilação em seus valores, todas ultrapassando a faixa $0 < \rho < 1$, o que não afeta a retomada da estabilização dos elementos da matriz S com o valor de referência.

Figura 6.21: Número de condição da matriz de covariância $\mathbf{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento μ de 0.878 - RLS$_\mu$-ADHDP-DLQR.

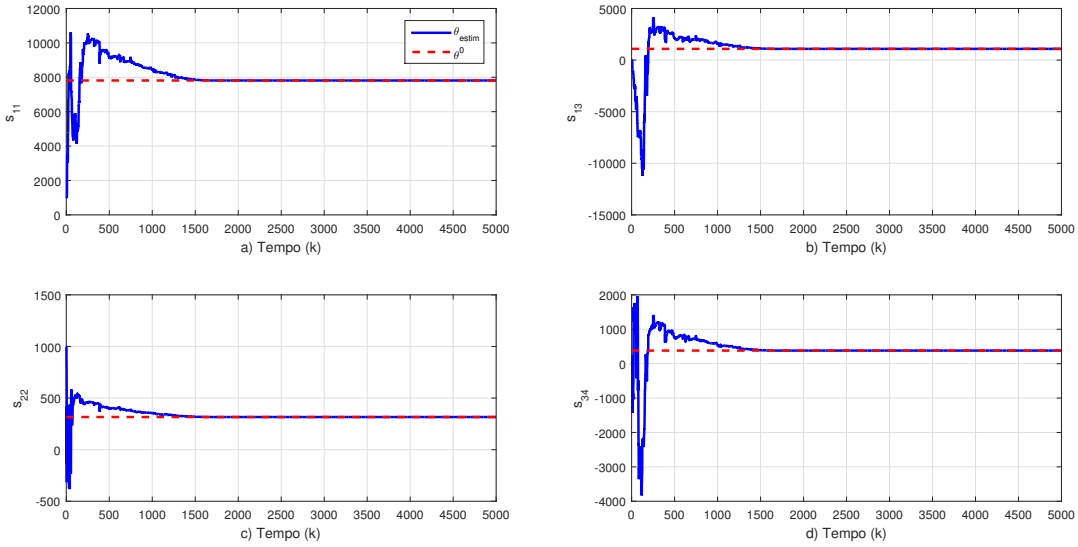


RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

Na Fig.6.22, apresenta-se a evolução do processo iterativo da solução da equação HJB-*Riccati* para um ciclo de 5000 iterações com fator de esquecimento de $\mu = 0.878$. As curvas (a)-(d) da Fig. 6.22 representam o comportamento de convergência dos elementos $s_{11},\ s_{22},\ s_{33}$ e s_{44} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_6,\ \theta_{10}$ e θ_{13} do vetor de parâmetros $\boldsymbol{\theta}$. Observa-se que o sistema entra em fase de acomodação suavemente por volta da iteração 1750, de modo a não apresentar perturbações quando comparados ao valor de referência (solução Schur).

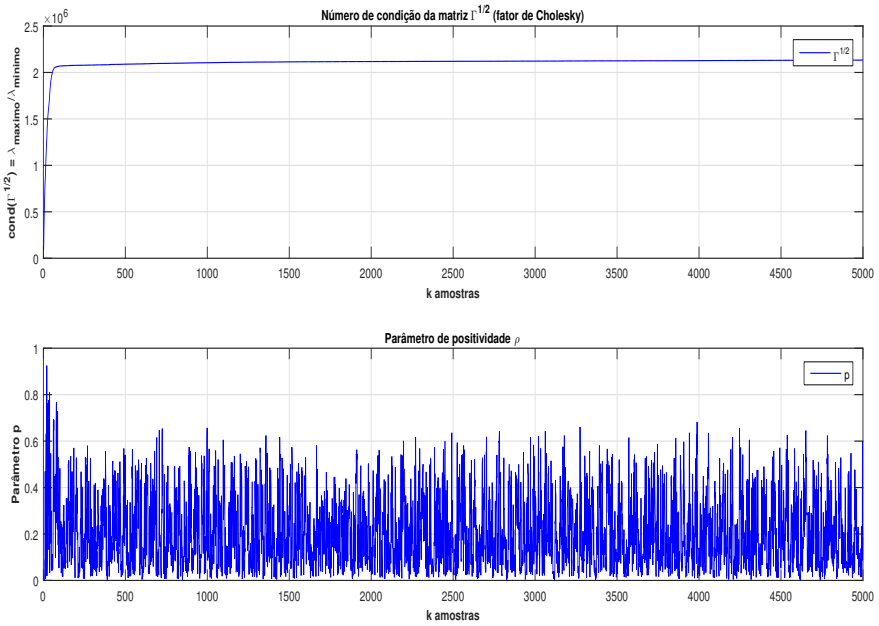
A Fig. 6.23 apresenta o número de condição do fator Cholesky e o parâmetro de positividade ρ para o processo de estimação da solução da equação HJB-*Riccati* via RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. Observa-se que o número de condição do fator Cholesky exibe um comportamento sem mudanças abruptas e o fator de positividade ρ não apresenta irregularidades perma-

Figura 6.22: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.878 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



necendo em sua faixa de referência.

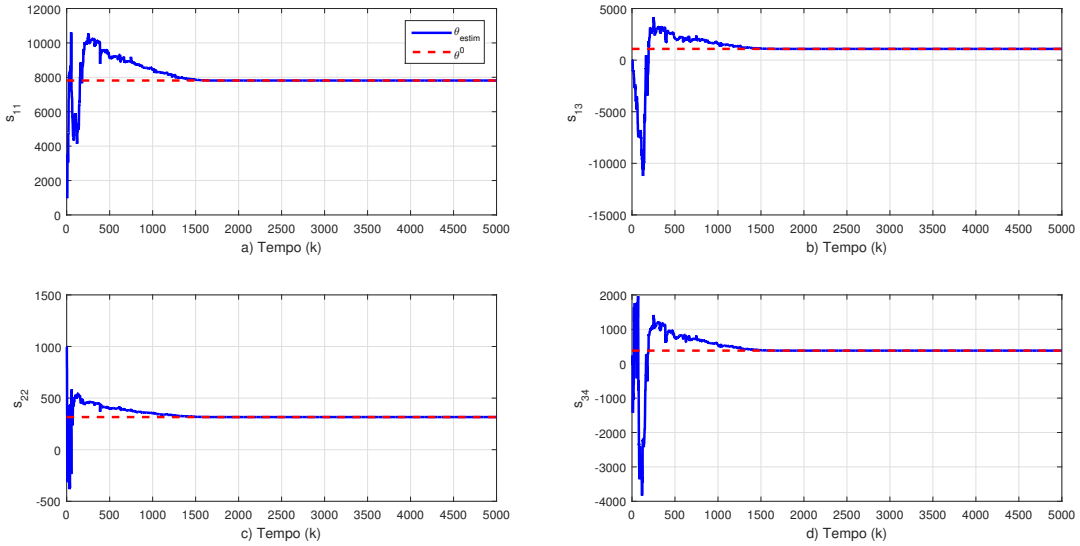
Figura 6.23: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento μ de 0.878 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



RLS$_\mu$-UDU^T-ADHDP-DLQR

De modo similar os resultados dos experimentos para o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR são apresentados para um ciclo de 5000 iterações, como mostrado na Fig.6.18. Os mesmos parâmetros já descritos para as estimações RLS$_\mu$-ADHDP-DLQR e RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR são analisados para o fator de esquecimento $\mu = 0.878$. Observa-se que o sistema entra em fase de acomodação logo após a iteração 1750, apresentando graficamente um comportamento estritamente similar ao resultado obtido pela estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR .

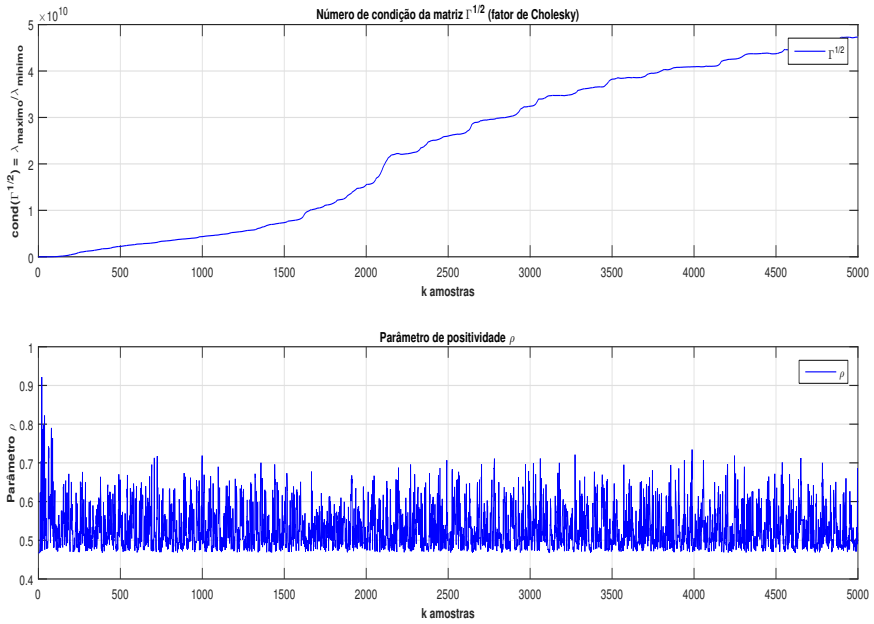
Figura 6.24: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.878 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



A Fig. 6.25 apresenta o número de condição do fator Cholesky e o parâmetro de positividade ρ para o processo de estimação da solução da equação HJB-*Riccati* via RLS$_\mu$-UDU^T-ADHDP-DLQR. Observa-se que o número de condição do fator Cholesky exibe um comportamento crescente, mas de modo suave e sem mudanças abruptas, com tendência a entrar em uma faixa estacionária de valores. Tal fato pode ser evidenciado pelo aumento no número de iterações realizadas para esse fator em mesmas condições de simulação. O parâmetro de positividade por sua vez encontra-se estritamente dentro da faixa de referência esperada.

Comparando os resultados para os elementos estimados pelos algoritmos apresentados nas Figs.6.20, 6.22 e 6.24 para o fator de esquecimento $\mu = 0.878$, percebe-se que todas as abordagens apresentaram uma resposta moderada no que diz respeito a convergência dos elementos estimados para tal fator de esquecimento, tendo a estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR um

Figura 6.25: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.878 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



comportamento ligeiramente melhor que RLS$_\mu$-UDU^T-ADHDP-DLQR, e ambas um comportamento superior ao algoritmo RLS$_\mu$-ADHDP-DLQR padrão.

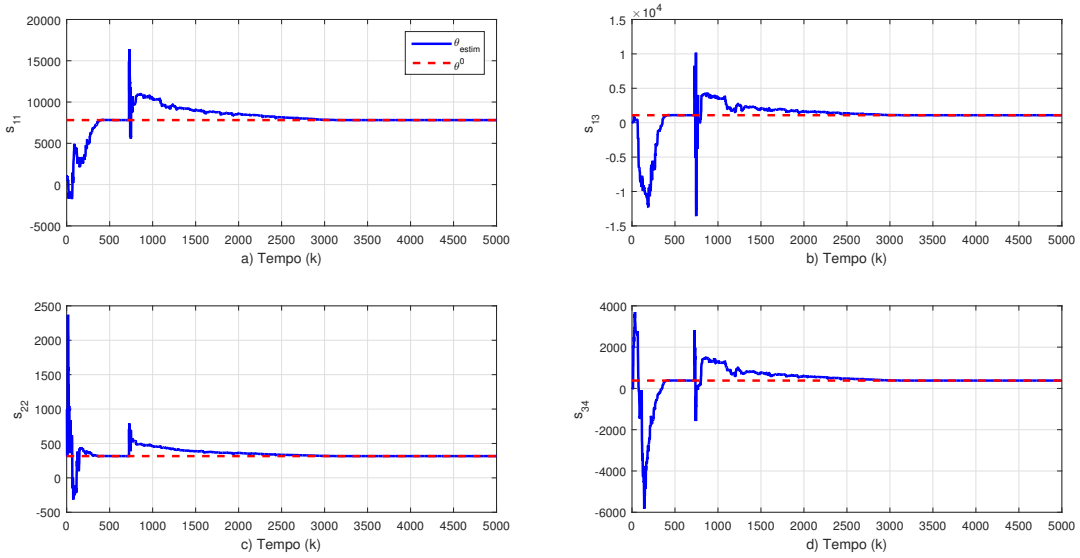
6.2.5 Simulação com $\mu = 0.934$

RLS$_\mu$-ADHDP-DLQR

A implementação do algoritmo RLS$_\mu$-ADHDP-DLQR tem a evolução de seu processo iterativo para a solução da equação HJB-*Riccati* ilustrada na Fig.6.26 para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.934$. As curvas (a)-(d) da Fig.6.26 representam o comportamento de convergência dos elementos $s_{11},\ s_{22},\ s_{33}$ e s_{44} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_6,\ \theta_{10}$ e θ_{13} do vetor de parâmetros $\boldsymbol{\theta}$. Observa-se que o estimador tende a alcançar o estado permanente rapidamente, por volta da iteração 400, mas apresenta um pico considerável logo após as 750 iterações, descaracterizando o estado estacionário.

Com relação ao comportamento de convergência do estimador RLS$_\mu$-ADHDP-DLQR padrão para o valor de $\mu = 0.934$, pode ser observado a partir da Fig.6.26 a relativa influência deste fator, uma vez que causa uma tendencia à convergência muito rápida nos parâmetros desse estimador (em torno da iteração 400) que é perdida por oscilações nos valores da matriz S em

Figura 6.26: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.934 - RLS$_\mu$-ADHDP-DLQR.



torno da iteração 750, sendo recuperada a fase de acomodação por volta das 3000 iterações. Uma característica anormal da matriz de covariância Γ_k evidencia-se entre as iterações 723 e 727, mesmo antes de o regime permanente ter sido novamente estabelecido no comportamento de S, como pode ser visto na Fig.6.27. O período entre as iterações 500 e 750, ilustrado também na Fig.6.27, mostra o parâmetro de positividade apresentando variações que excedem sua faixa de referência, tal comportamento caracteriza em muitos casos a perda de convergência dos elementos s_{ij} da matriz S.

RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

Na Fig.6.28, apresenta-se a evolução do processo iterativo da solução da equação HJB-$Riccati$ para um ciclo de 5000 iterações com fator de esquecimento de $\mu = 0.934$. As curvas (a)-(d) da Fig.6.28 representam o comportamento de convergência dos elementos $s_{11},\ s_{22},\ s_{33}$ e s_{44} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_6,\ \theta_{10}$ e θ_{13} do vetor de parâmetros $\boldsymbol{\theta}$. Observa-se que o sistema entra em fase de acomodação rapidamente por volta da iteração 400, de modo a não apresentar perturbações quando comparados ao valor de referência (solução Schur) e a solução aproximada dada pelo algoritmo RLS$_\mu$-ADHDP-DLQR.

Na Fig.6.29 são apresentados o número de condição do fator Cholesky e o parâmetro de positividade para o processo de estimação da solução da equação HJB-*Riccati* via RLS$_\mu$-$\mathcal{QR}$-

Figura 6.27: Número de condição da matriz de covariância $\mathbf{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.934 - RLS$_\mu$-ADHDP-DLQR.

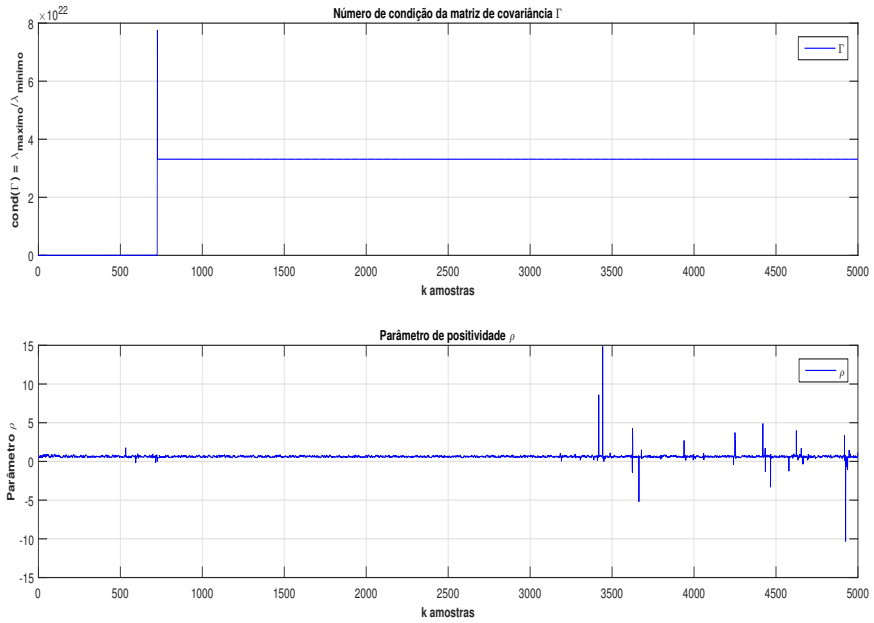
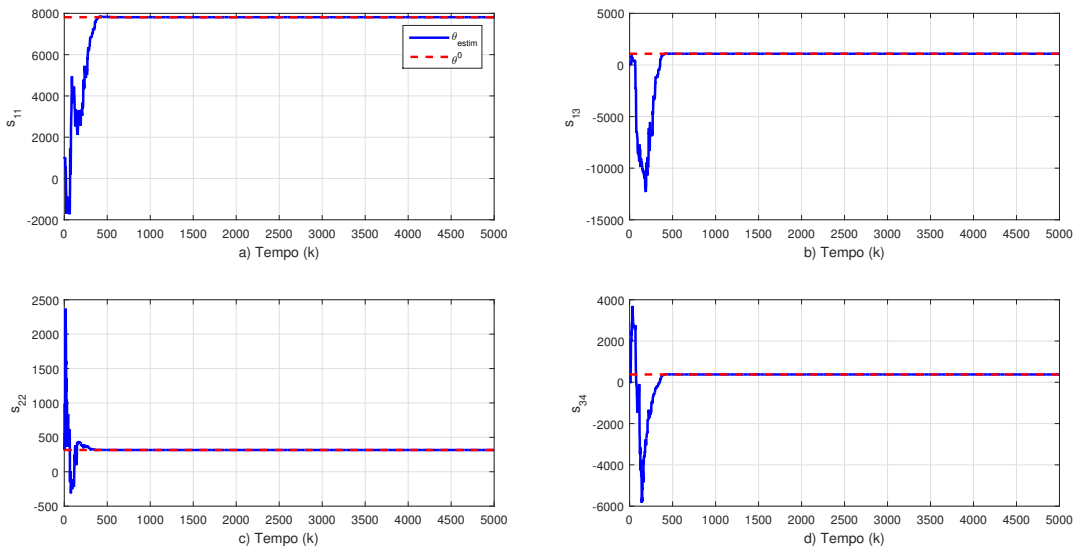
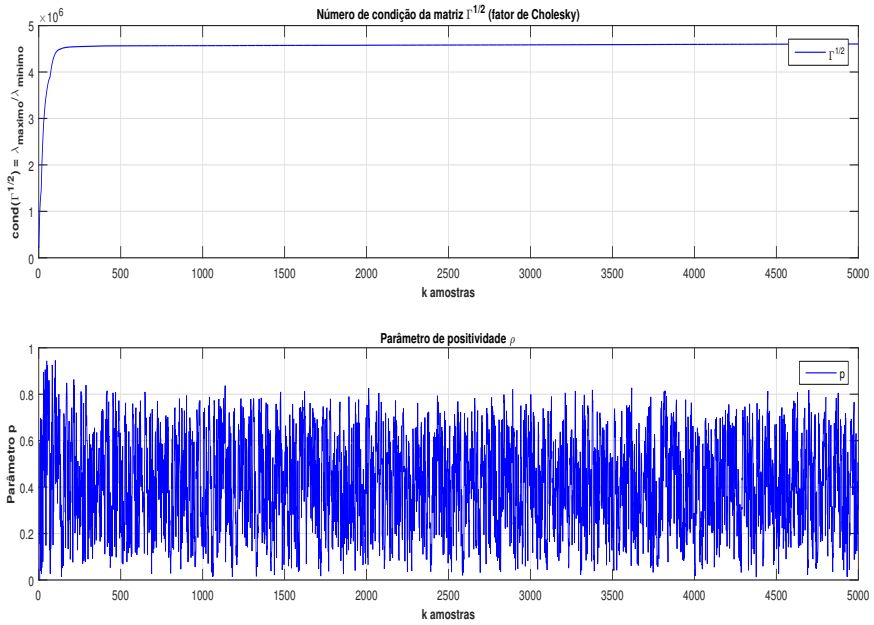


Figura 6.28: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.934 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



ADHDP-DLQR. Observa-se que tanto o número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ como o fator de positividade ρ exibem um comportamento suave sem mudanças abruptas e com amplitudes significativamente menores em relação as apresentadas pela estimação RLS$_\mu$-ADHDP-DLQR.

Figura 6.29: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.934 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



RLS$_\mu$-UDU^T-ADHDP-DLQR

O processo iterativo para a solução da equação HJB-*Riccati* via algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR é apresentado na Fig.6.30, também para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.934$. As curvas (a)-(d) da Fig.6.30 representam o comportamento de convergência dos elementos $s_{11},\ s_{22},\ s_{33}$ e s_{44} da matriz S correspondentes respectivamente aos parâmetros $\theta_1,\ \theta_6,\ \theta_{10}$ e θ_{13} do vetor de parâmetros $\boldsymbol{\theta}$. Para a estimação RLS$_\mu$ via fatoração UDU^T a convergência dos parâmetros da matriz S para o valor $\mu = 0.934$ é alcançada em torno da iteração 400, assim como ocorre no algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e de forma análoga, os valores estimados convergem para o valor verdadeiro dado pela solução Schur.

Quanto ao número de condição do fator Cholesky visto na Fig.6.31, observa-se um valor substancialmente menor para o fator Cholesky $\Gamma_k^{1/2}$ do algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR em comparação com o número de condição da matriz de covariância Γ_k do estimador RLS$_\mu$-ADHDP-DLQR, e este sendo ligeiramente maior que o apresentado pela estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. Na Fig.6.31, é ilustrado o comportamento do fator de positividade ρ inteiramente dentro da faixa de referência, sendo uma indicação de que o algoritmo tende a apresentar sensibilidade menor aos distúrbios causados por problemas numéricos.

Em uma maneira geral, fatores de esquecimento com $\mu > 0.9$ tendem a apresentar uma

Figura 6.30: Evolução do processo iterativo dos parâmetros s_{ij} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.934 - RLS$_\mu$-UDU^T-ADHDP-DLQR.

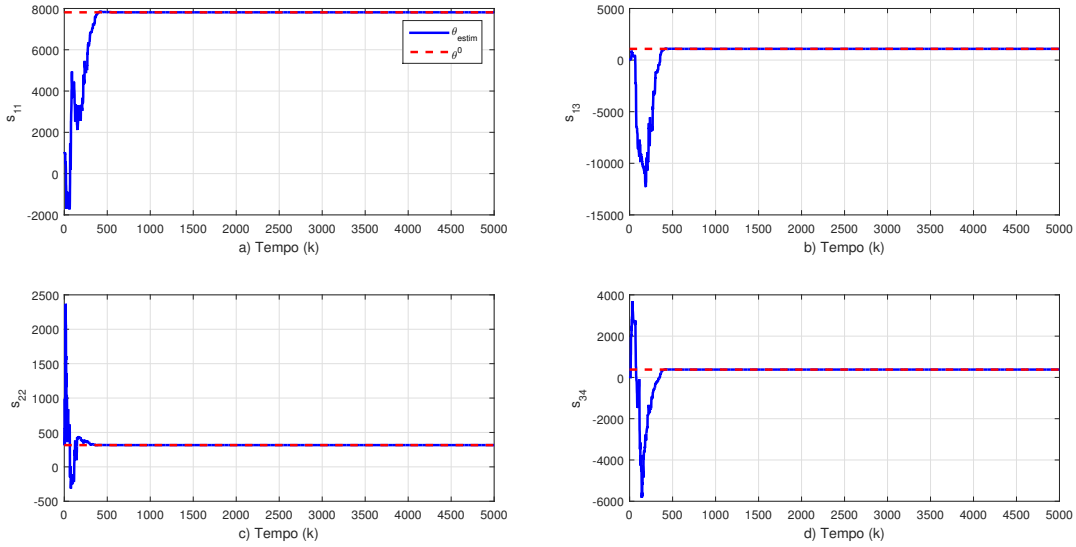
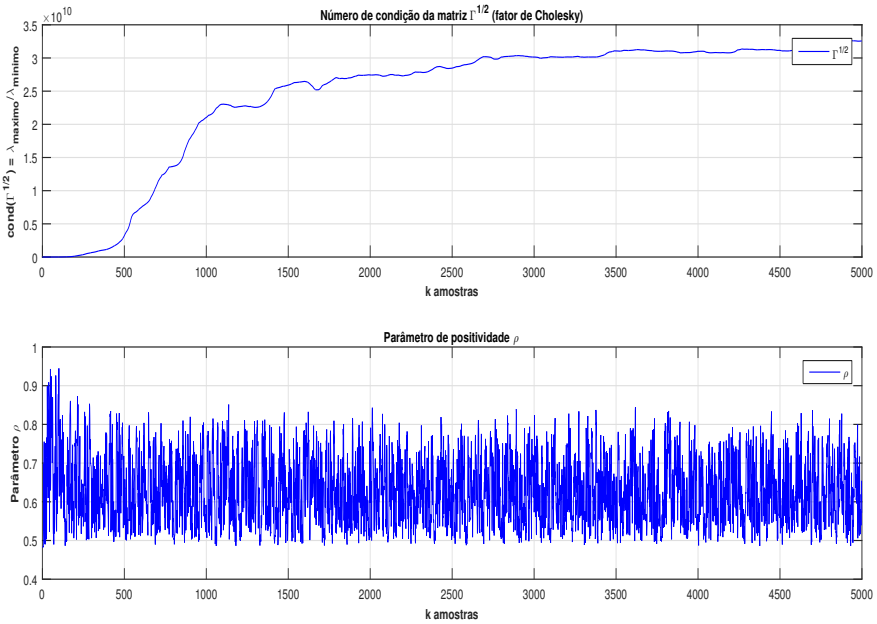


Figura 6.31: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.934 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



convergência mais lenta quando comparados a fatores relativamente menores. De fato isso não pode ser visto nessa análise uma vez que o fator $\mu = 0.934$ contribui significativamente para diminuir o processo de convergência do estimador RLS$_\mu$-ADHDP-DLQR padrão (em torno da iteração 400), sendo essa definitivamente alcançada somente em torno da iteração 3300, devido a perda de estabilidade numérica sofrida pela estimação.

Em relação aos estimadores RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR, pôde-se observar uma maior sensibilidade à escolha do fator de esquecimento $\mu = 0.934$ produzindo uma tendência a convergência com oscilações mais baixas e um tempo de acomodação consideravelmente mais rápido quando comparados com o estimador RLS$_\mu$-ADHDP-DLQR. Também se observa que os resultados apresentados nas Figs.6.29 e 6.31 indicam características de robustez para o fenômeno de instabilidade numérica quando o parâmetro de positividade se mantém dentro do intervalo de validade durante todo o processo iterativo e o número da condição mostra um comportamento em que não são observadas variações abruptas.

Nos demais fatores de esquecimento simulados como $0.634, 0.739$ e 0.824 não há convergência plena dos elementos s_{ij} da matriz S para a estimação RLS$_\mu$-ADHDP-DLQR. O uso dos estimadores RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR para esses casos mostra a capacidade da abordagem proposta em aproximar a solução estimada da equação HJB-*Riccati* dos valores de referência dados pela solução Schur de maneira suave e robusta, assim como a capacidade dos estimadores de superarem os problemas de estabilidade numérica inerentes a abordagem padrão RLS$_\mu$-ADHDP-DLQR.

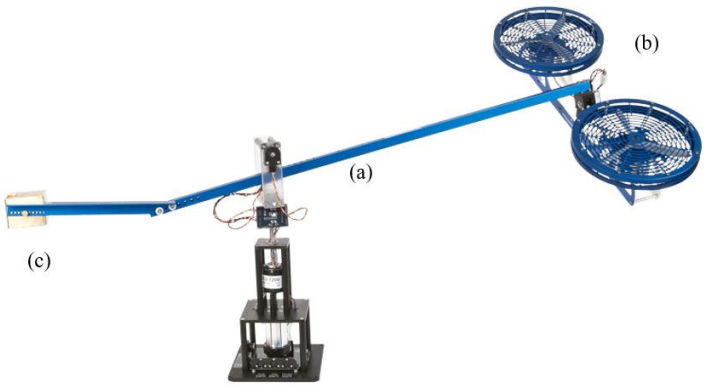
Pode-se afirmar que para todas as simulações realizadas o algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR apresenta ligeira vantagem sobre o estimador RLS$_\mu$-UDU^T-ADHDP-DLQR por apresentar um número de condição em uma menor faixa de valores, onde os valores do número de condição tendem a ser mantidos limitados a um intervalo no decorrer do tempo. Tal intervalo possivelmente está diretamente relacionado a composição dos limites que levam as condições necessárias para à estabilidade numérica do algoritmo proposto. Tal vantagem da estimação RLS$_\mu$-$\mathcal{QR}$ só se faz menor quando leva-se em consideração os resultados de custo computacional descritos no Capítulo 5.

Uma análise das condições para a estabilidade numérica da implementação de precisão finita de algoritmos RLS convencionais pode ser encontrada em (LIAVAS; REGALIA, 1999), onde os autores fornecem limites para a precisão relativa dos cálculos em termos do fator de esquecimento μ e do condicionamento do problema, isto é, o número de condição da matriz de covariância Γ_k. Ali, a precisão relativa mostra-se proporcional a $1-\mu$, o que significa que para μ muito próximo de 1, o acúmulo de erro de arredondamento é mais significativo. Além disso, um análise teórica sobre o fenômeno de propagação do erro de arredondamento na implementação de esquemas de estimação RLS pode ser vista em (LJUNG; LJUNG, 1985; VERHAEGEN, 1989).

6.3 Modelo do Helicóptero 3-DOF

A plataforma experimental utilizada nesta seção é um helicóptero de três graus de liberdade Quanser (3-DOF do inglês *3 Degree-of-Freedom*), cuja montagem é representada na Fig.6.32, e seu esquemático é descrito na Fig.6.33.

Figura 6.32: Helicóptero 3-DOF e seus componentes: (a) braço principal, (b) duplo rotor e (c) contrapeso.

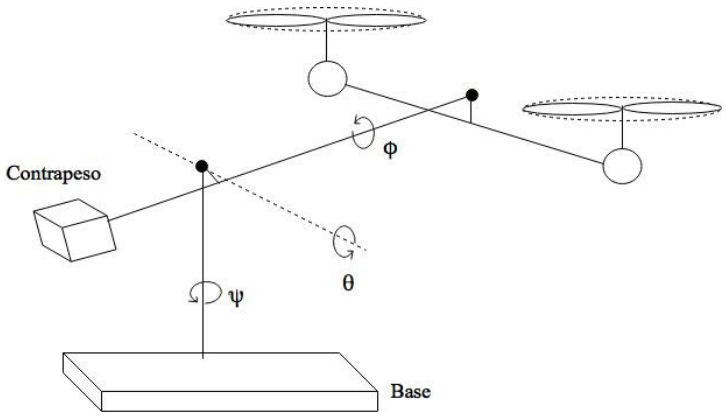


O helicóptero 3-DOF é montado sobre uma base fixa e seus componentes primários são o braço principal, o conjunto do rotor duplo e o contrapeso. O braço principal é montado de modo a permitir que o conjunto do rotor rotacione em círculos contínuos. Este movimento de rotação é chamado de deslocamento (*travel*). Isso ocorre em torno de um eixo vertical que passa por um anel de escorregamento e é perpendicular à base fixa. Na montagem do rolamento e do anel de escorregamento, existe um ponto pivô permitindo que o braço principal possa elevar-se e abaixar-se. Este movimento é descrito como elevação (*elevation*) e ele ocorre em torno do eixo que passa pela montagem do anel de escorregamento e é paralelo à base fixa. Na extremidade mais longa do braço principal, existe outro rolamento, cujo eixo é paralelo ao braço. Isso permite um conjunto de rotores duplos acionados por motores DC girar em torno desse rolamento. O movimento rotacional dos rotores é referido como arfagem (*pitch*) e ocorre em torno de um eixo que atravessa o braço principal. Os motores de corrente contínua podem fornecer tensão coletiva ou tensão diferencial (cíclica). A tensão coletiva resulta no movimento de elevação do braço principal e a tensão diferencial resulta no movimento de arfagem. O movimento de arfagem dos rotores, por sua vez, origina o movimento de deslocamento do equipamento. Na outra extremidade do braço principal, existe um contrapeso que reduz os requisitos de energia nos motores, reduzindo o peso efetivo do conjunto do rotor.

A dinâmica do helicóptero pode ser descrita por um modelo não linear de 6ª ordem e para obter as equações do sistema, utiliza-se um sistema de coordenadas com sua origem no

conjunto do rolamento e do anel de escorregamento, sendo o deslocamento ψ o movimento circular do braço principal, arfagem ϕ o movimento do conjunto dos rotores e a elevação θ o movimento para cima e para baixo do braço principal, assim como suas respectivas velocidades (velocidade de deslocamento $\dot{\psi}$, velocidade de arfagem $\dot{\phi}$ e velocidade de elevação $\dot{\theta}$). Os ângulos correspondentes são mostrados na Fig.6.33.

Figura 6.33: Configuração do modelo experimental para o sistema do helicóptero.



Dessa forma, a equação de estados pode ser expressa como:

$$\dot{x} = f(x, u) \tag{6.12}$$

em que $x = [\phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}]$ e $u = [u_1\ u_2]^T$, sendo as variáveis de controle u_1 e u_2 correspondentes às tenções fornecidas aos motores esquerdo e direito, respectivamente.

Para simplificar a hipótese, considerou-se que a inércia do helicóptero pode ser representada por massas pontuais associadas ao corpo do helicóptero, ao contrapeso, ao centro de gravidade do braço de sustentação e à posição da massa de perturbação ativa. Além disso, foram considerados os efeitos de atrito viscoso nas equações que descrevem a dinâmica dos movimentos de arfagem e deslocamento. Afim de tornar mais realística a simulação, desprezaram-se o arrasto aerodinâmico e o momento angular das hélices, que giram no mesmo sentido. Para as forças geradas pelas hélices, considerou-se que estas não dependem do movimento relativo do corpo do helicóptero em relação ao ar. Por fim, desprezou-se a dinâmica eletromecânica dos conjuntos motor-hélice, que é muito mais rápida que as forças que a dinâmica de movimento do sistema como um todo.

O helicóptero 3-DOF tem sido objeto de estudo de diversos trabalhos na literatura, como por exemplo (ISHUTKINA, 2004; LOPES, 2007; SAPIŃSKI; ROSÓŁ, 2008; BREGANON, 2009). Dentre os modelos propostos para representar a dinâmica do helicóptero, optou-se por

utilizar aquele apresentado em (LOPES, 2007; MAIA, 2008). Tal modelo foi obtido através do formalismo de Lagrange, sendo não-linear e de sexta ordem. Através de algumas simplificações matemáticas, esse modelo pode ser expresso pelo seguinte conjunto de equações diferenciais:

$$
\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \xi_{16}.\{\xi_1(u_1^2 - u_2^2) + \xi_2(u_1 - u_2) - v_2 \cdot x_2\} \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= x_6^2 \cdot \{\xi_3 \cdot \sin(2x_3) + \xi_4 \cdot \cos(2x_3)\} + \xi_5 \cdot \sin(x_3) + \xi_6 \cdot \cos(x_3) + \\
&\quad + \{\xi_7(u_1^2 + u_2^2) + \xi_8(u_1 + u_2)\} \cdot \cos(x_1) \\
\dot{x}_5 &= x_6 \\
\dot{x}_6 &= \{\xi_{13} + \xi_{14} \cdot \sin(2x_3) + \xi_{15} \cdot \cos(2x_3)\}^{-1} \cdot \{v_1 - v_3 \cdot x_6 + \\
&\quad + [\xi_9(u_1^2 + u_2^2) + \xi_{10}(u_1 + u_2)] \cdot \sin(x_1) + x_4 \cdot x_6 \cdot [\xi_{11} \cdot \sin(2x_3) + \xi_{12} \cdot \cos(2x_3)]\}
\end{aligned}
\tag{6.13}
$$

em que x_1, x_3 e x_5 representam, os ângulos de deslocamento, arfagem e elevação (em rad), x_2, x_4 e x_6 representam suas respectivas derivadas (rad/s), e u_1 e u_2 representam as tensões de entrada dos motores esquerdo e direito respectivamente. Os demais parâmetros são constantes relacionadas às dimensões físicas e massas dos diversos componentes do helicóptero, assim como, as constantes relacionadas ao atrito viscoso estando seus valores utilizados apresentados na Tabela 6.1

Tabela 6.1: Valores dos Parâmetros

Parâmetro	Valor	Parâmetro	Valor
ξ_1	0.1117 N/V^2	ξ_{11}	1.0567 kg·m^2
ξ_2	0.0449 N/V	ξ_{12}	-0.2515 kg·m^2
ξ_3	-0,4843	ξ_{13}	0.5454 kg·m^2
ξ_4	0,1153	ξ_{14}	0.1258 kg·m^2
ξ_5	-1.0389 N/(m·kg)	ξ_{15}	0.5283 kg·m^2
ξ_6	-1.3170 N/(m·kg)	ξ_{16}	4.1832 (m·kg)$^{-1}$
ξ_7	0.0656 N/(m·kg·V^2)	v_1	0.107 N·m
ξ_8	0.0264 N/(m·kg·V)	v_2	0.18 N·s
ξ_9	-0.0718 N·m/V^2	v_3	0.47 N·m·s
ξ_{10}	-0.0289 N·m/V	-	-

Condições Iniciais e *Setup* da Simulação

Realizou-se a linearização para o ponto de operação $\bar{x}_1 = \bar{x}_2 = \bar{x}_4 = \bar{x}_5 = \bar{x}_6 = 0$ e $\bar{x}_3 = -7$, do modelo não-linear do helicóptero usando os parâmetros fornecidos na Tabela 6.1. As matrizes do modelo linear a tempo contínuo obtidas são expressas como

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.753 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -0.098 & 0 & -1.192 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1.257 & 0 & 0 & 0 & 0 & -0.457 \end{bmatrix} \text{ e } B = \begin{bmatrix} 0 & 0 \\ 2.814 & -2.814 \\ 0 & 0 \\ 0.394 & 0.394 \\ 0 & 0 \\ -0.035 & -0.035 \end{bmatrix}. \quad (6.14)$$

As variáveis de saída escolhidas foram os ângulos de deslocamento, arfagem e elevação ($y = [\psi\ \theta\ \phi]^T$). Além disso, como não há transmissão direta entre a entrada e a saída da planta, as matrizes C e D foram definidas como

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \text{ e } D = [0]. \quad (6.15)$$

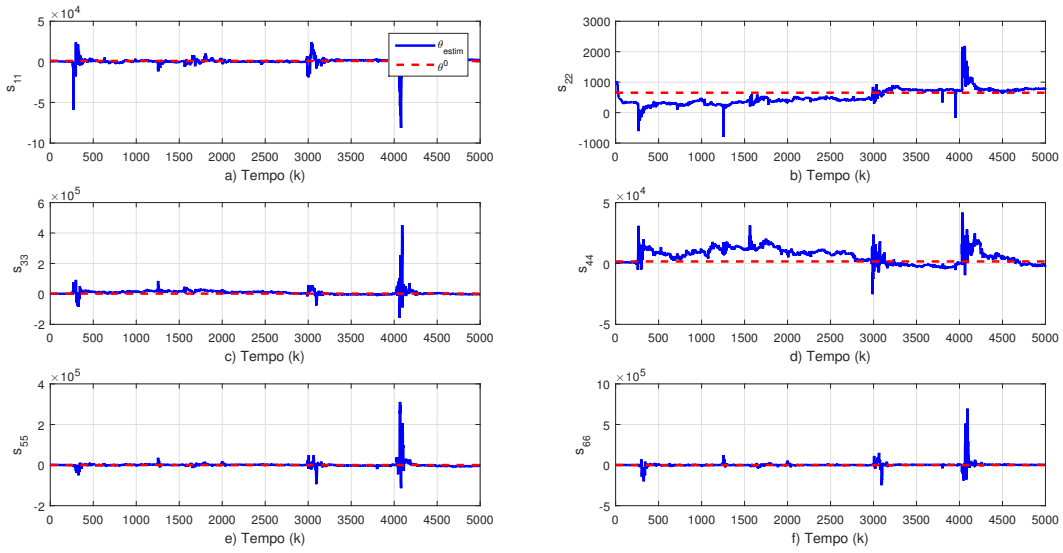
Para a implementação do algoritmo RLS$_\mu$-ADHDP-DLQR as condições iniciais adotadas são: uma política inicial K_0, admissível, fator de desconto $\gamma = 1$, estado do sistema $x_0 = [0.18\ -0.19\ -0.27\ -0.32\ 0.61\ -0.5]^T$ para os ângulos em radianos e matrizes de ponderação da função de custo dadas por $Q = [10^{-2}\ 0\ 0\ 0\ 0\ 0; 0\ 1\ 0\ 0\ 0\ 0; 0\ 0\ 10^{-2}\ 0\ 0\ 0; 0\ 0\ 0\ 1\ 0\ 0; \ldots$ $0\ 0\ 0\ 0\ 10^{-1}\ 0; 0\ 0\ 0\ 0\ 0\ 1]$ e $R = [10^2\ 0; 0\ 10^2]$. Os parâmetros que inicializam o estimador RLS proposto são dados pelo vetor $\boldsymbol{\theta}_0 = 10^3[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0$ $1\ 0\ 0\ 1\ 0\ 1]^T$; e matriz $\Gamma_0 = 10^{-3} I_{36\times 36}$, sendo I uma matriz identidade de ordem n_θ. As matrizes do sistema discretizado para o intervalo de amostragem $T_{amost} = 0.01s$ são obtidas pelo método do segurador de ordem zero (ZOH). Para as simulações diversos valores de esquecimento μ são utilizados, para fins de análise são considerados os valores: $0.782,\ 0.818,\ 0.874$ e 0.919.

6.3.1 Simulação com $\mu = 0.782$

RLSμ-ADHDP-DLQR

O processo iterativo para a solução da equação HJB-*Riccati* do algoritmo RLS$_\mu$-ADHDP-DLQR tem sua evolução ilustrada na Fig.6.34 para um ciclo de 5000 iterações, com o fator de esquecimento $\mu = 0.782$. As curvas (a)-(f) representam o comportamento de convergência dos elementos $s_{11}, s_{22}, s_{33}, s_{44}, s_{55}$ e s_{66} da matriz S correspondentes aos elementos $\theta_1, \theta_9, \theta_{16}, \theta_{22}, \theta_{27}$ e θ_{31} do vetor de parâmetros. As curvas (a)-(c) da Fig.6.35 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos θ_{11}, θ_{12} e θ_{32} do vetor $\boldsymbol{\theta}$. Observa-se em tais parâmetros constante oscilação que perdura para além do período transitório, e mesmo sob grande esforço o algoritmo não consegue estimar os parâmetros do vetor $\boldsymbol{\theta}$ de modo a seguir o valor de referência.

Figura 6.34: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.782 - RLS$_\mu$-ADHDP-DLQR.



Nota-se, da Fig.6.36, variações significativas fora da faixa de validade, do parâmetro de positividade da matriz de covariância. As primeiras grandes variações ocorrem justamente em um momento de esforço de ajuste com a solução Schur (valor de referência) no intervalo das iterações 200 e 280. Anormalidades no comportamento do número de condição da matriz de covariância também são evidenciadas na Fig.6.36, apresentando comportamento crescente em altas amplitudes.

Figura 6.35: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.782 - RLS$_\mu$-ADHDP-DLQR.

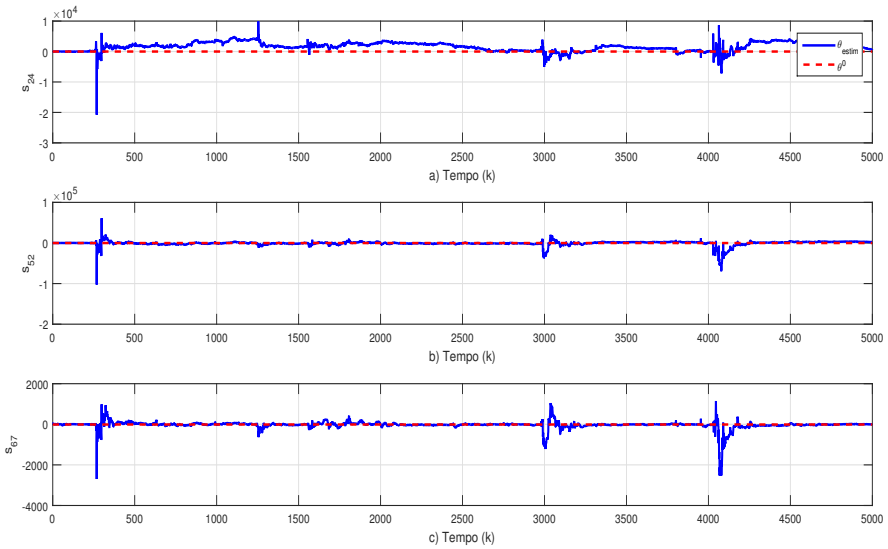
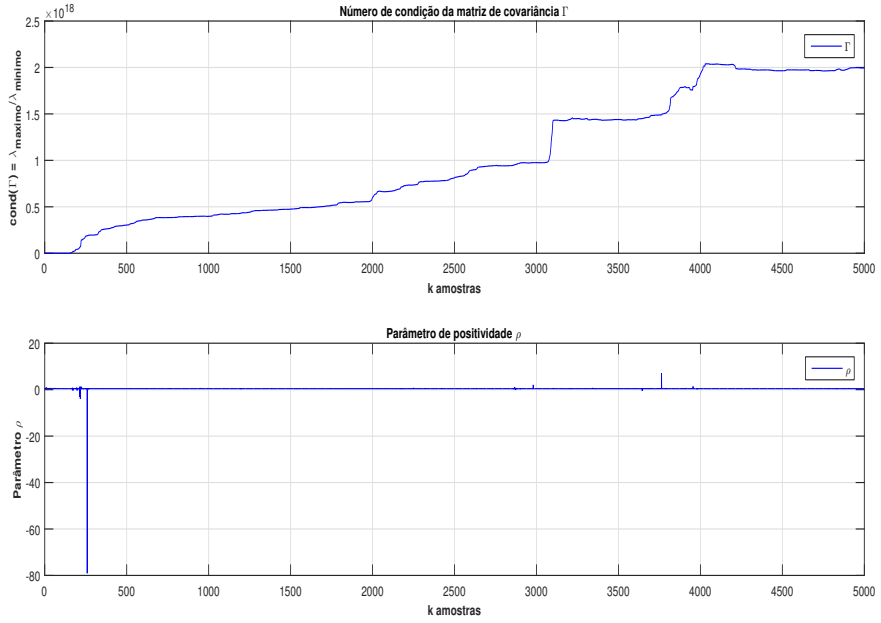


Figura 6.36: Número de condição da matriz de covariância $\mathbf{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.782 - RLS$_\mu$-ADHDP-DLQR.



RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

Nas Figs.6.37 e 6.38, apresenta-se a evolução do processo iterativo para solução da equação HJB-*Riccati* para um ciclo de 5000 iterações com o fator de esquecimento $\mu = 0.782$ via estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. As curvas (a)-(f) da Fig.6.37 representam o comportamento de convergência dos elementos $s_{11}, s_{22}, s_{33}, s_{44}, s_{55}$ e s_{66} da matriz S correspondentes

aos elementos $\theta_1,\ \theta_9,\ \theta_{16},\ \theta_{22},\ \theta_{27}$ e θ_{31} do vetor de parâmetros. Assim como, as curvas (a)-(c) da Fig.6.38 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos $\theta_{11},\ \theta_{12}$ e θ_{32} do vetor a ser estimado. Observa-se, pelas curvas citadas, um comportamento do período transitório mais suave quando comparado ao apresentado pelo algoritmo RLSμ-ADHDP-DLQR, a convergência dos parâmetros estimados para o valor de referência é alcançada próximo a iteração 3800.

Figura 6.37: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.782 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

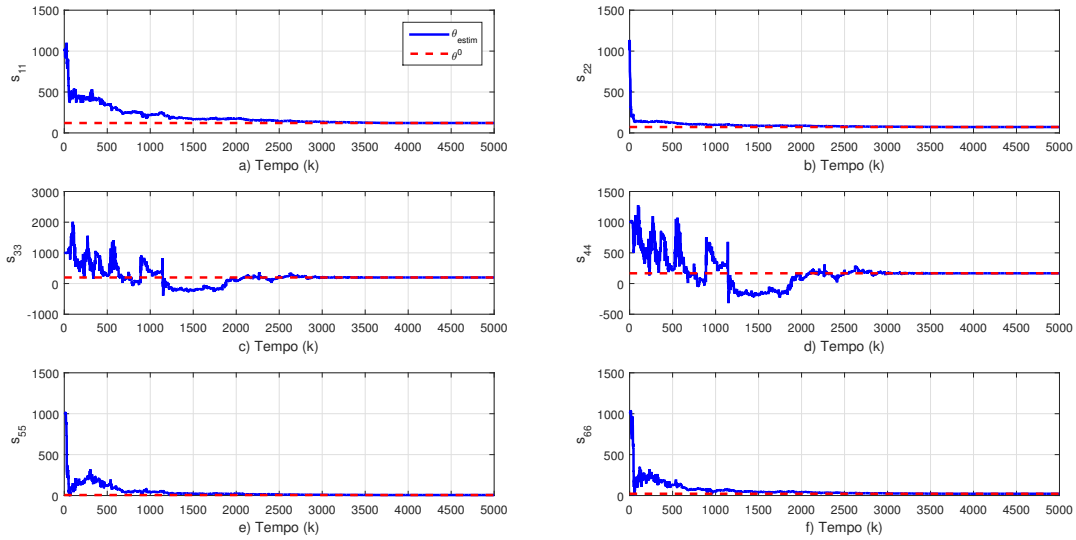
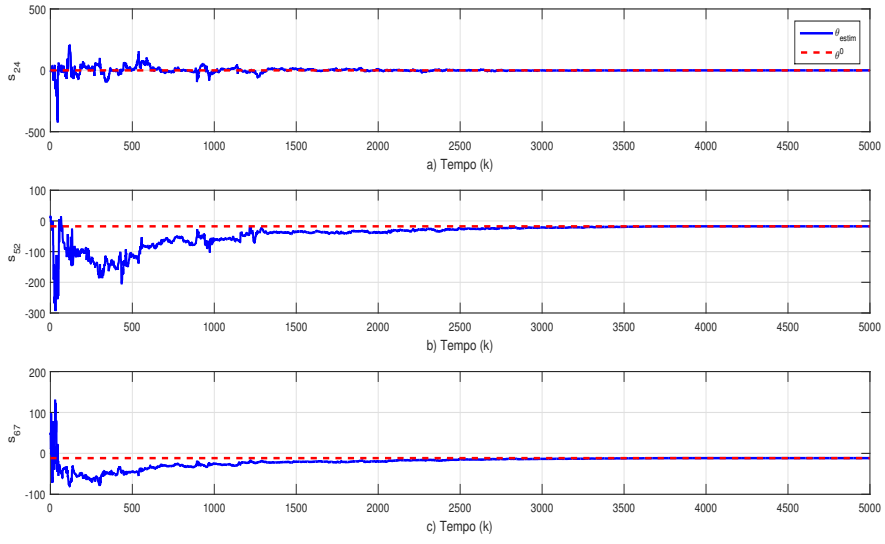
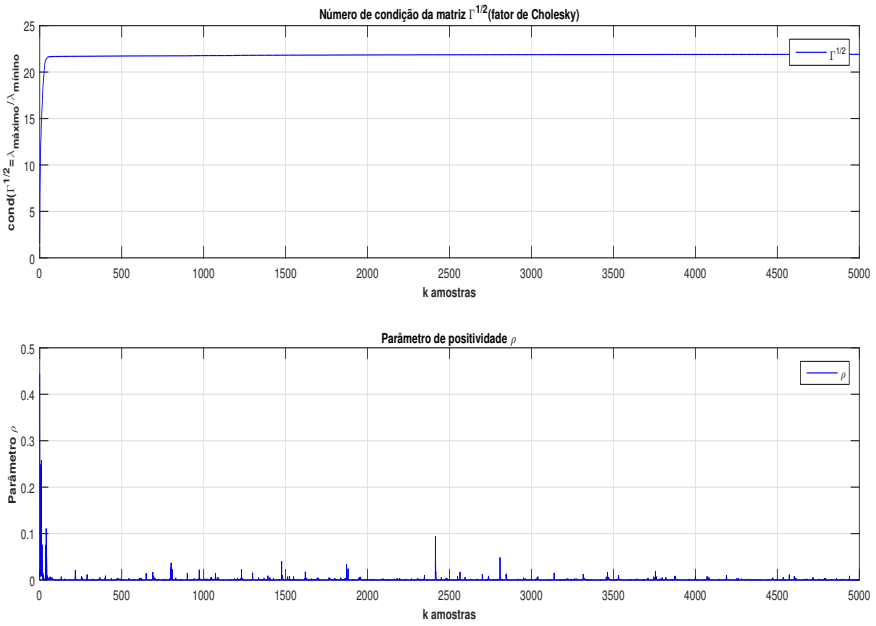


Figura 6.38: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.782 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



O comportamento do número de condição do Fator Cholesky, assim como, o comportamento do parâmetro de positividade ρ são ilustrados na Fig.6.39. Observa-se em ambos uma drástica redução nos valores apresentados quando comparados ao estimado RLS$_\mu$-ADHDP-DLQR. O Fator Cholesky apresenta um comportamento mais suave, sem mudanças abruptas de valores, e o parâmetro de positividade permanece durante todo o processo iterativo dentro da faixa de interesse ($0 < \rho < 1$). Tal condição caracteriza um melhor condicionamento numérico do método.

Figura 6.39: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.782 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



RLS$_\mu$-UDU^T-ADHDP-DLQR

De modo similar as simulações anteriores, as curvas (a)-(f) da Fig.6.40 representam o comportamento de convergência dos elementos $s_{11}, s_{22},\ s_{33},\ s_{44},\ s_{55}$ e s_{66} da matriz S correspondentes aos elementos $\theta_1,\ \theta_9,\ \theta_{16},\ \theta_{22},\ \theta_{27}$ e θ_{31} do vetor de parâmetros. Assim como, as curvas (a)-(c) da Fig.6.41 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos $\theta_{11},\ \theta_{12}$ e θ_{32} do vetor estimado para o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR. Observa-se, a partir do comportamento de estimação dos parâmetros algumas divergências quando esse comportamento é comparado com os resultados apresentados pela estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR, de modo que mesmo apresentando um período transitório com comportamento mais suave, a estimação RLS$_\mu$-UDU^T-ADHDP-DLQR se distância do

valor verdadeiro (Solução Schur) para alguns dos elementos estimados.

Figura 6.40: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.782 - RLS$_\mu$-UDU^T-ADHDP-DLQR.

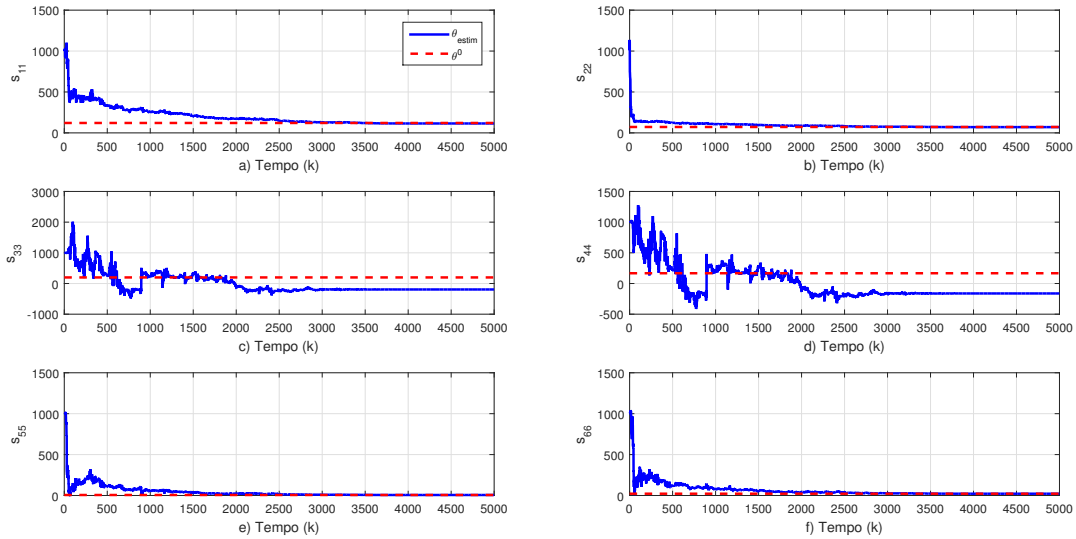
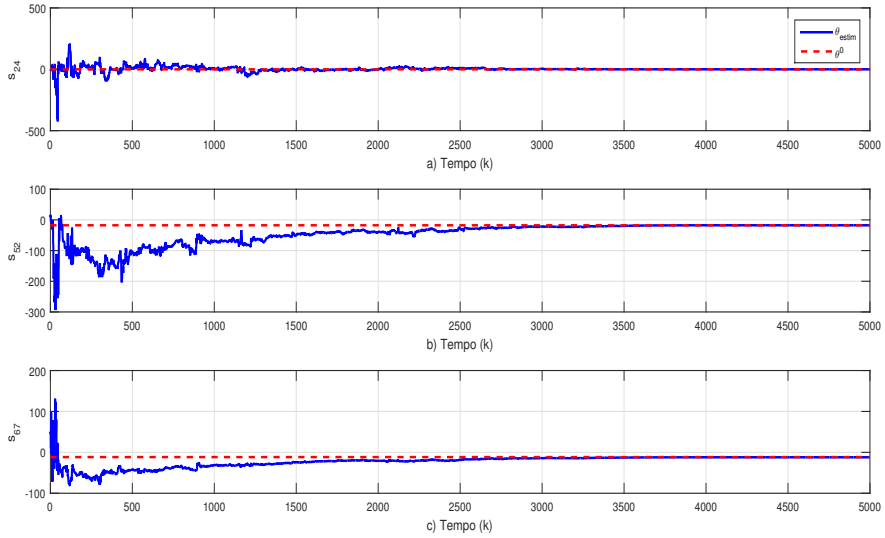
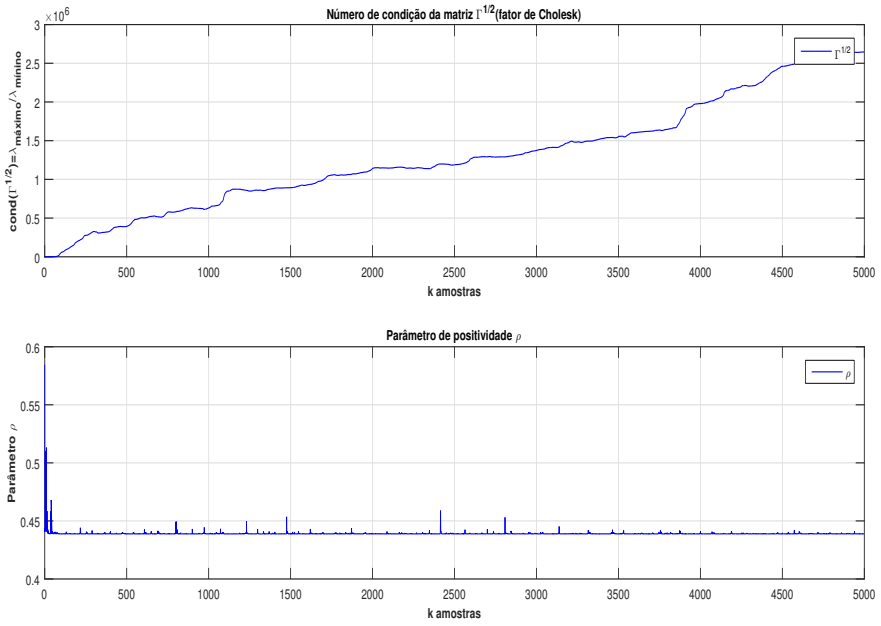


Figura 6.41: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.818 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



O número de condição do Fator Cholesky e o parâmetro de positividade do estimador RLS$_\mu$-UDU^T-ADHDP-DLQR são apresentados na Fig.6.42. Observa-se que o Fator Cholesky para o ciclo de iteração apresenta-se de modo mais suave e em uma amplitude de valores menores que os apresentados pela estimação RLS$_\mu$-ADHDP-DLQR, enquanto o parâmetro ρ, apresenta-se dentro do faixa de validade $0 < \rho < 1$.

Figura 6.42: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.782 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



Analisando os resultados para o fator de esquecimento $\mu = 0.782$, pode-se dizer que apenas a estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR alcança a convergência para todos os parâmetros estimados, o que ocorre próximo à iteração 3800. Após sua estabilização não demonstram sofrer com perturbações, isso é devido a uma matriz melhor condicionada pela decomposição $\mathcal{QR}$. O algoritmo padrão sem decomposição sofre com perturbações depois do período transitório, não conseguindo a convergência para o limite de simulação adotado que é de 5000 iterações. Já a estimação RLS$_\mu$-UDU^T-ADHDP-DLQR apresenta uma característica peculiar, mesmo mantendo comportamento regular em seus número de condição do fator Cholesky e parâmetro de positividade. Para alguns dos valores estimados do vetor $\boldsymbol{\theta}$ ocorrem divergências quanto a solução de referência.

6.3.2 Simulação com $\mu = 0.818$

RLSμ-ADHDP-DLQR

A evolução do processo iterativo para a solução da equação HJB-*Riccati* do algoritmo RLS$_\mu$-ADHDP-DLQR é mostrada na Fig. 6.43 para um ciclo de 5000 iterações, com o fator de esquecimento $\mu = 0.818$. As curvas (a)-(f) representam o comportamento de convergência dos elementos $s_{11}, s_{22}, s_{33}, s_{44}, s_{55}$ e s_{66} da matriz S correspondentes aos elementos

θ_1, θ_9, θ_{16}, θ_{22}, θ_{27} e θ_{31} do vetor de parâmetros. As curvas (a)-(c) da Fig.6.44 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos θ_{11}, θ_{12} e θ_{32} do vetor $\boldsymbol{\theta}$. Observa-se em tais parâmetros frequente oscilação durante o período transitório, e, mesmo sob grande esforço, o algoritmo não alcança a convergência.

Figura 6.43: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.818 - RLS$_\mu$-ADHDP-DLQR.

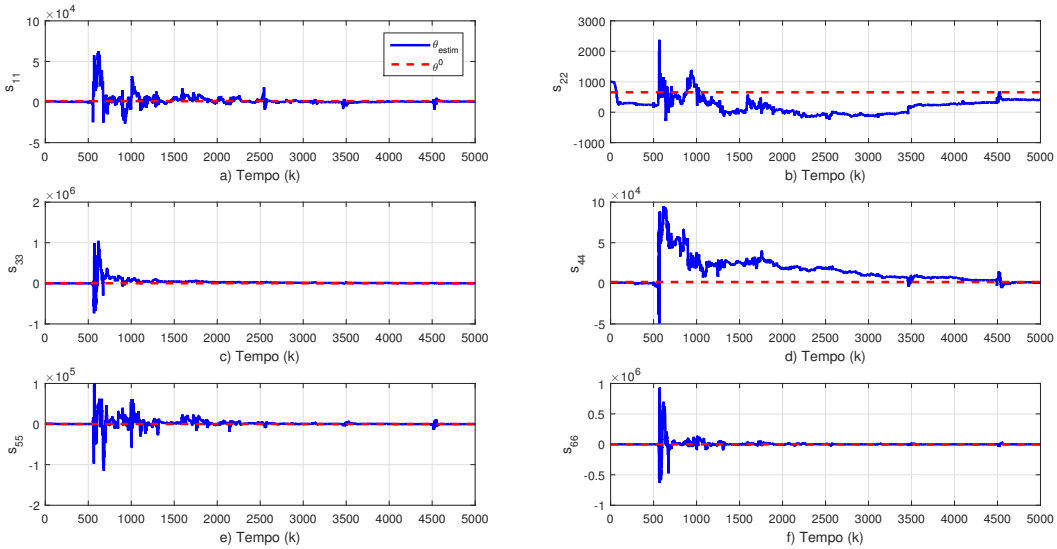
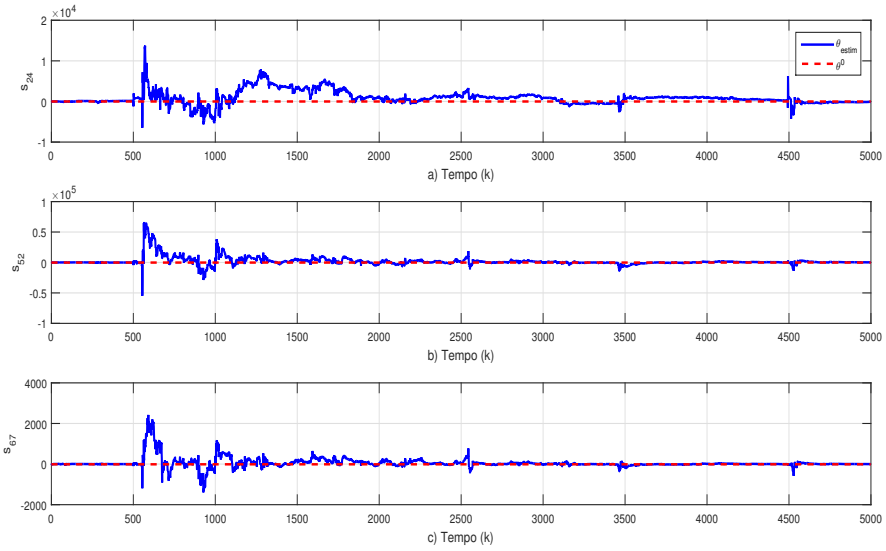


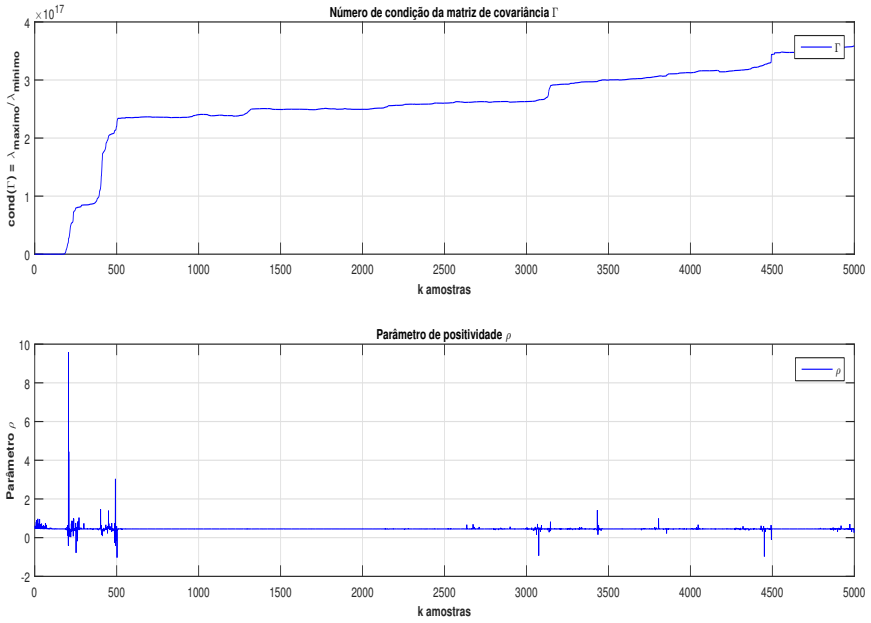
Figura 6.44: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.818 - RLS$_\mu$-ADHDP-DLQR.



Nota-se na Fig.6.45, variações no número de condição da matriz de covariância, bem como irregularidades no parâmetro de positividade ρ. Anormalidades no comportamento de tais

parâmetros são evidentes mesmo antes da iteração 250, de modo que, o número de condição apresenta um pico crescente e os valores de ρ oscilam drasticamente fora de faixa de interesse.

Figura 6.45: Número de condição da matriz de covariância $\mathbf{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.818 - RLS$_\mu$-ADHDP-DLQR.



RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

Nas Figs.6.46 e 6.47, apresenta-se a evolução do processo iterativo para solução da equação HJB-*Riccati* para um ciclo de 5000 iterações com o fator de esquecimento $\mu = 0.818$ via estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR. As curvas (a)-(f) da Fig.6.46 representam o comportamento da convergência dos elementos $s_{11}, s_{22},\ s_{33},\ s_{44},\ s_{55}$ e s_{66} da matriz S correspondentes aos elementos $\theta_1,\ \theta_9,\ \theta_{16},\ \theta_{22},\ \theta_{27}$ e θ_{31} do vetor de parâmetros. Assim como, as curvas (a)-(c) da Fig.6.47 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos $\theta_{11},\ \theta_{12}$ e θ_{32} do vetor $\boldsymbol{\theta}$. Para esta simulação observa-se pelas curvas citadas um comportamento do período transitório mais suave quando comparado ao apresentado pela solução estimada pelo algoritmo RLSμ-ADHDP-DLQR. A convergência dos parâmetros é alcançada logo após a iteração 2500.

Os comportamentos do número de condição do Fator Cholesky, assim como, do parâmetro de positividade ρ são ilustrados na Fig.6.48. Percebe-se que o Fator Cholesky tem um comportamento mais suave que o apresentado pela estimação RLS$_\mu$-ADHDP-DLQR, e o parâmetro de positividade exibe um comportamento dentro da faixa de validade, tais características

Figura 6.46: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.818 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

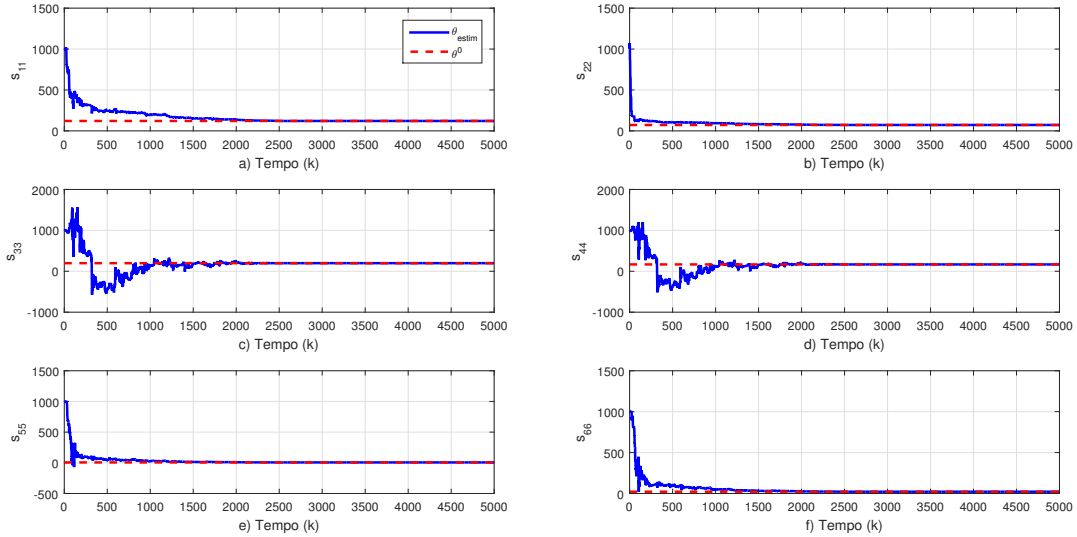
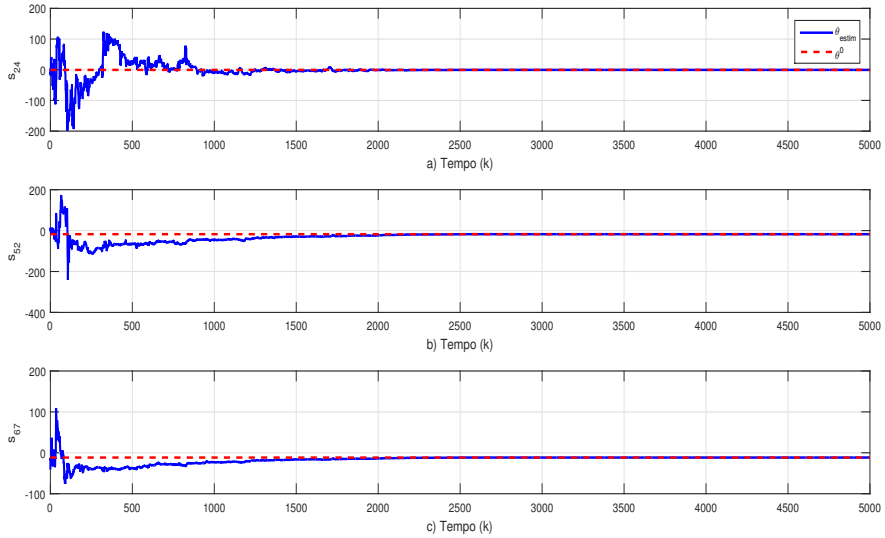


Figura 6.47: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.818 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

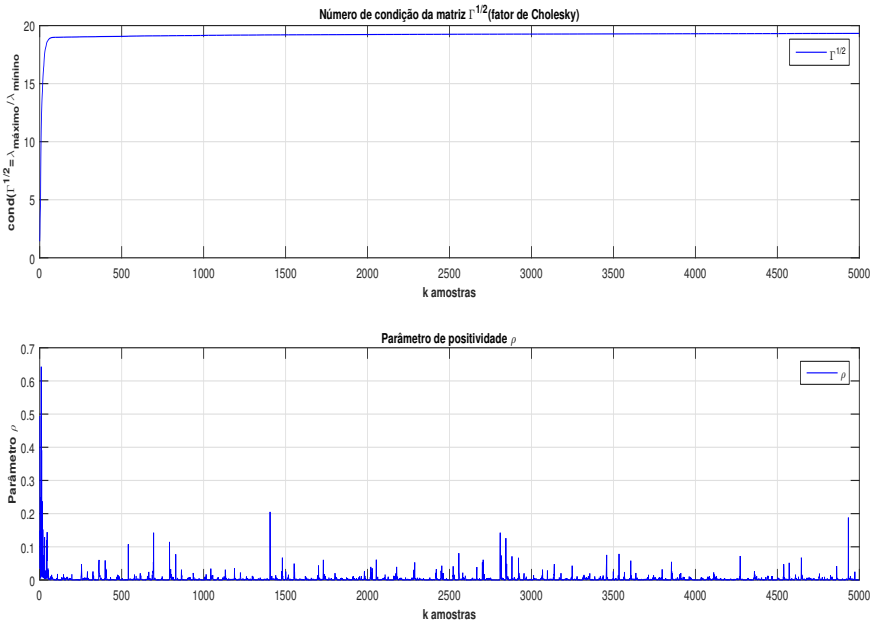


podem indicar um melhor condicionamento numérico do método proposto.

RLS$_\mu$-UDU^T-ADHDP-DLQR

Tal como nas simulações anteriores, as curvas (a)-(f) da Fig.6.49 representam o comportamento de convergência dos elementos $s_{11}, s_{22},\ s_{33},\ s_{44},\ s_{55}$ e s_{66} da matriz S correspondentes aos elementos $\theta_1,\ \theta_9,\ \theta_{16},\ \theta_{22},\ \theta_{27}$ e θ_{31} do vetor de parâmetros. Assim como, as curvas (a)-(c) da Fig.6.50 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados

Figura 6.48: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.818 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



aos elementos $\theta_{11},\ \theta_{12}$ e θ_{32} do vetor a ser estimado para o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR. O comportamento de convergência dos parâmetros apresentados para esta simulação demonstra sua semelhança com o obtido pela estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR, de modo que em ambas as estimações os parâmetros de $\boldsymbol{\theta}$ convergem para o valor verdadeiro dada pela solução Schur logo após a iteração 2500.

Figura 6.49: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.818 - RLS$_\mu$-UDU^T-ADHDP-DLQR.

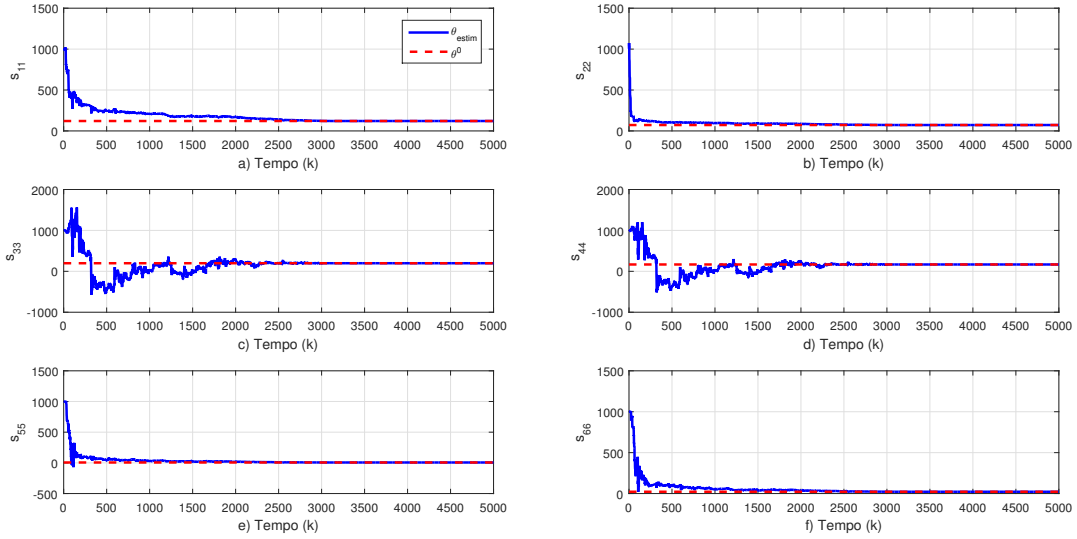
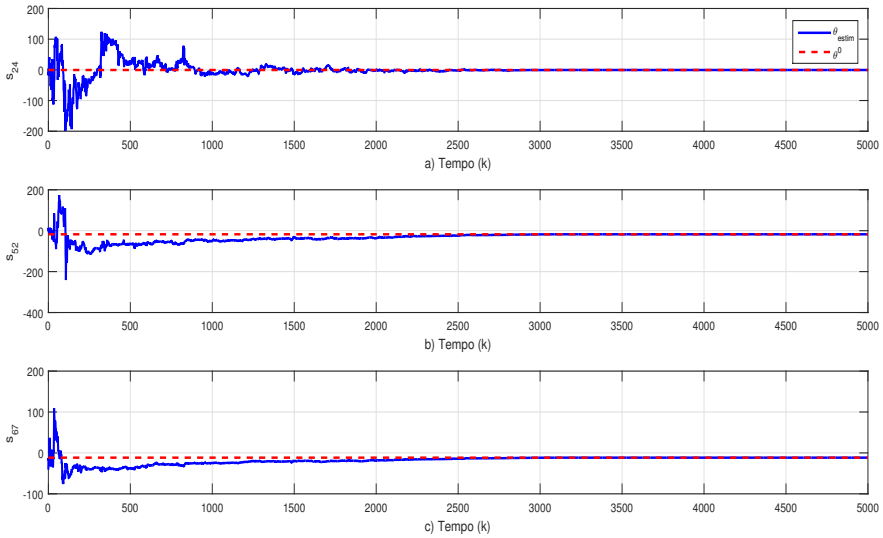
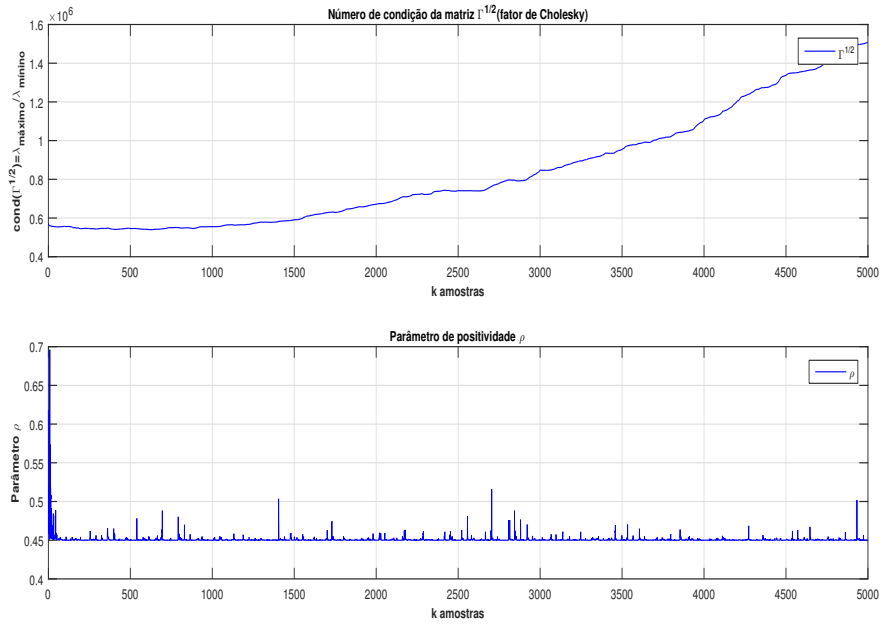


Figura 6.50: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.818 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



O número de condição do Fator Cholesky e o parâmetro de positividade do estimador RLS$_\mu$-UDU^T-ADHDP-DLQR são apresentados na Fig. 6.51. Observa-se que o Fator Cholesky para o intervalo de iteração tem um comportamento suave, sem variações abruptas em seus valores, e o parâmetro de positividade ρ apresenta-se dentro da faixa de interesse.

Figura 6.51: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.818 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



Analisando os resultados para o fator de esquecimento $\mu = 0.818$, pode-se dizer que os

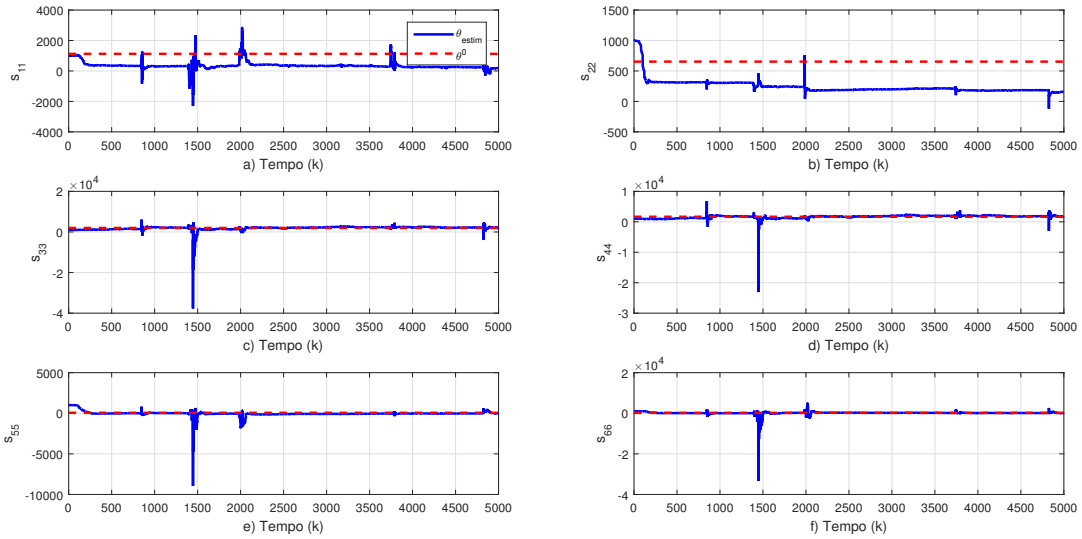
algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR alcançaram a convergência para os parâmetros estimados, isso logo após a iteração 2500, e após suas respectivas estabilizações demonstram não sofrer com perturbações, devido a uma matriz melhor condicionada por meio das decomposições $\mathcal{QR}$ e UDU^T das abordagens adotadas. Já o algoritmo padrão sem decomposição sofre com perturbações depois do período transitório mostrando-se instável durante toda a simulação.

6.3.3 Simulação com $\mu = 0.874$

RLSμ-ADHDP-DLQR

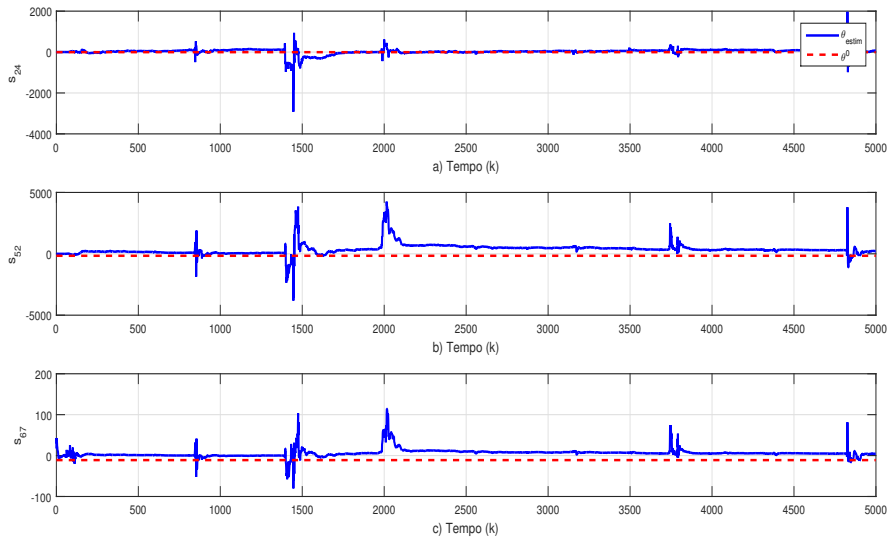
O processo iterativo para a solução da equação HJB-*Riccati* do algoritmo RLS$_\mu$-ADHDP-DLQR tem sua evolução apresentada na Fig.6.52 para um ciclo de 5000 iterações, com o fator de esquecimento $\mu = 0.874$. As curvas (a)-(f) representam o comportamento de convergência dos elementos $s_{11}, s_{22}, s_{33}, s_{44}, s_{55}$ e s_{66} da matriz S correspondentes aos elementos $\theta_1, \theta_9, \theta_{16}, \theta_{22}, \theta_{27}$ e θ_{31} do vetor de parâmetros. As curvas (a)-(c) da Fig.6.53 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos θ_{11}, θ_{12} e θ_{32} do vetor $\boldsymbol{\theta}$. Os parâmetros estimados apresentam constantes oscilações e mesmo sob esforço o algoritmo diverge sua aproximação da solução de referência para cada elemento apresentado.

Figura 6.52: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.874 - RLS$_\mu$-ADHDP-DLQR.



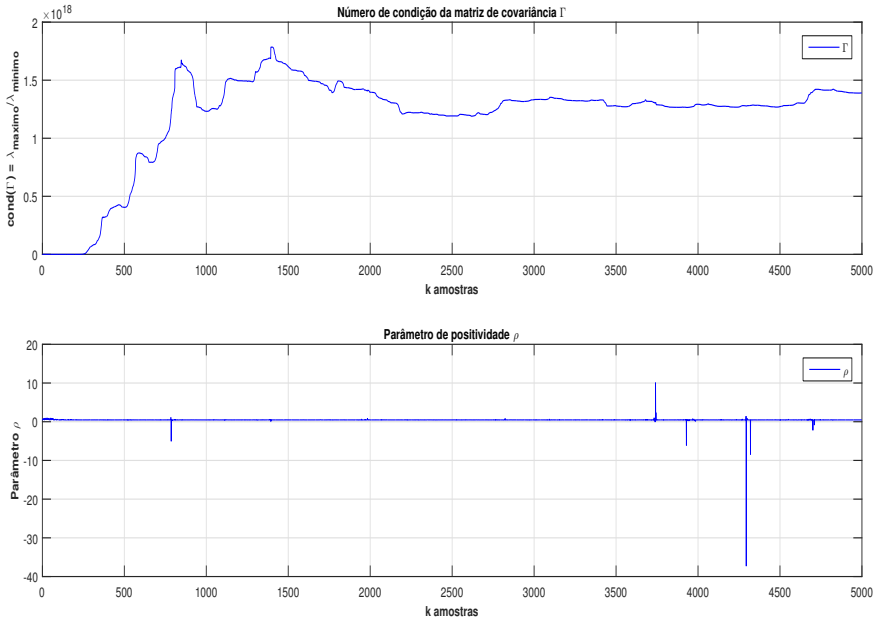
A partir da Fig.6.54, percebe-se grandes variações no número de condição da matriz de covariância, bem como irregularidades no parâmetro de positividade ρ. Tais irregularidades são

Figura 6.53: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.874 - RLS$_\mu$-ADHDP-DLQR.



expressivas, e em certo modo antecedem desajustes dos parâmetros estimados. Anormalidades no comportamento de tais parâmetros sugerem um pior condicionamento numérico da solução aproximada, o que se caracteriza pelas constantes oscilações dos parâmetros do vetor estimado $\boldsymbol{\theta}$.

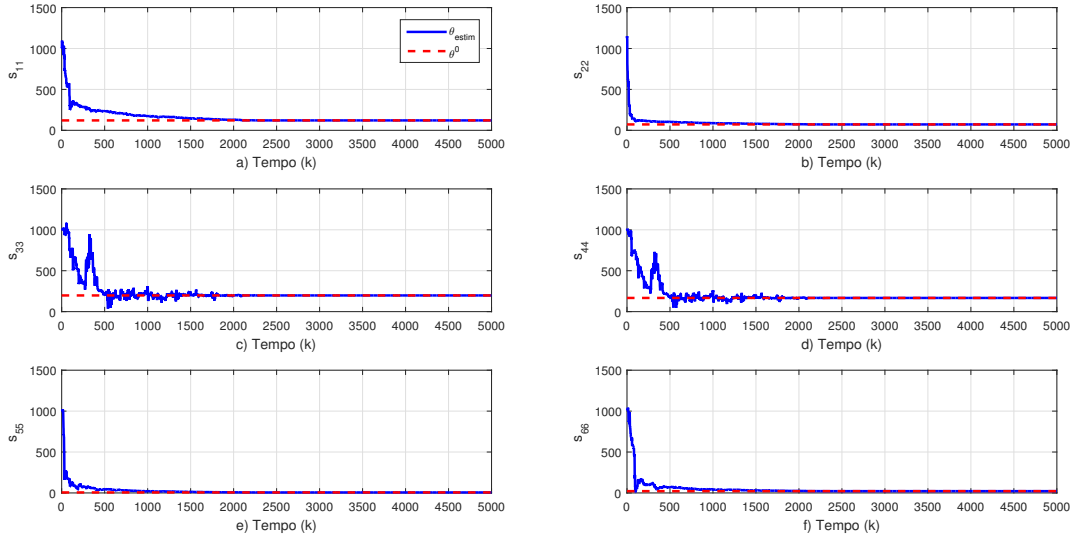
Figura 6.54: Número de condição da matriz de covariância $\boldsymbol{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento μ de 0.874 - RLS$_\mu$-ADHDP-DLQR.



RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

O processo iterativo para solução da equação HJB-*Riccati* para um ciclo de 5000 iterações com o fator de esquecimento $\mu = 0.874$ é apresentado nas Figs.6.55 e 6.56. As curvas (a)-(f) da Fig.6.55 representam o comportamento de convergência dos elementos $s_{11}, s_{22},\ s_{33},\ s_{44},\ s_{55}$ e s_{66} da matriz S correspondentes aos elementos $\theta_1,\ \theta_9,\ \theta_{16},\ \theta_{22},\ \theta_{27}$ e θ_{31} do vetor de parâmetros. Assim como, as curvas (a)-(c) da Fig.6.56 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos $\theta_{11},\ \theta_{12}$ e θ_{32} do vetor a ser estimado. Percebe-se pelas curvas citadas um comportamento mais regular e também mais suave dos elementos estimados quando comparados aos apresentados pelo algoritmo RLSμ-ADHDP-DLQR. A convergência dos parâmetros é alcançada logo após a iteração 2400.

Figura 6.55: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.874 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



Ilustram-se na Fig.6.57 tem-se os comportamentos do número de condição do Fator Cholesky, assim como, do parâmetro de positividade ρ. Observa-se em ambos uma suavização nos valores apresentados quando comparados ao estimado RLS$_\mu$-ADHDP-DLQR, exibindo um comportamento dentro de faixas de interesse, o que indica a robustez numérica para a estimação RLS$_\mu$-$\mathcal{QR}$.

RLS$_\mu$-UDU^T-ADHDP-DLQR

De modo similar as simulações anteriores, às curvas (a)-(f) da Fig.6.58, representam o comportamento de convergência dos elementos $s_{11}, s_{22},\ s_{33},\ s_{44},\ s_{55}$ e s_{66} da matriz S cor-

Figura 6.56: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.874 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

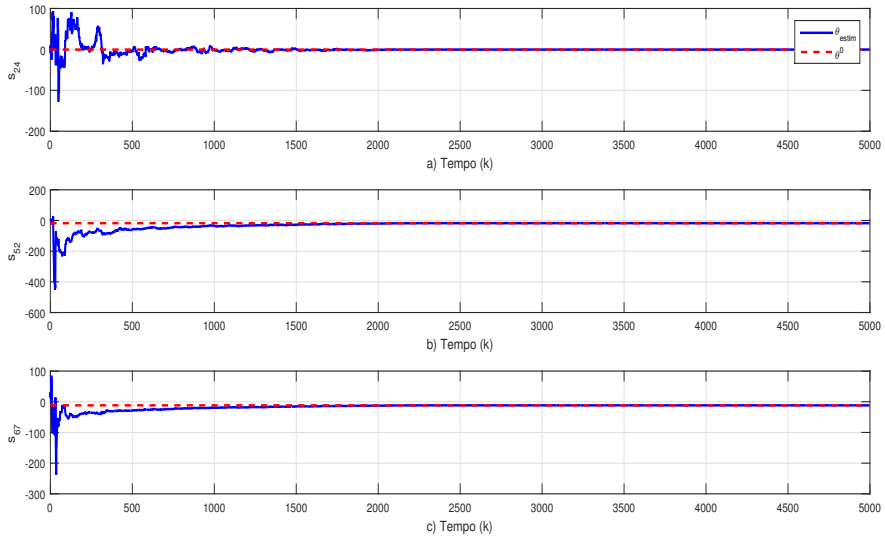
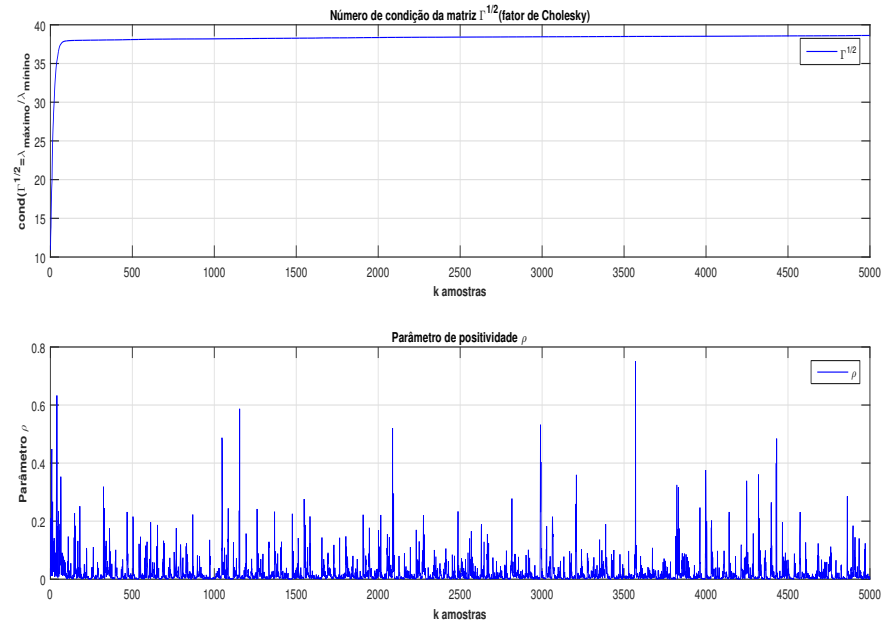


Figura 6.57: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento μ de 0.874 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



respondentes aos elementos $\theta_1,\ \theta_9,\ \theta_{16},\ \theta_{22},\ \theta_{27}$ e θ_{31} do vetor de parâmetros. Assim como, as curvas (a)-(c) da Fig.6.59 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos $\theta_{11},\ \theta_{12}$ e θ_{32} do vetor a ser estimado. Nota-se a partir do comportamento de convergência dos parâmetros sua semelhança com o apresentado pela estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR, de modo que em ambas as estimações os parâmetros de $\boldsymbol{\theta}$ conver-

gem para a vizinhança do valor verdadeiro dado pela solução Schur, o que ocorre logo após a iteração 2500.

Figura 6.58: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.874 - RLS$_\mu$-UDU^T-ADHDP-DLQR.

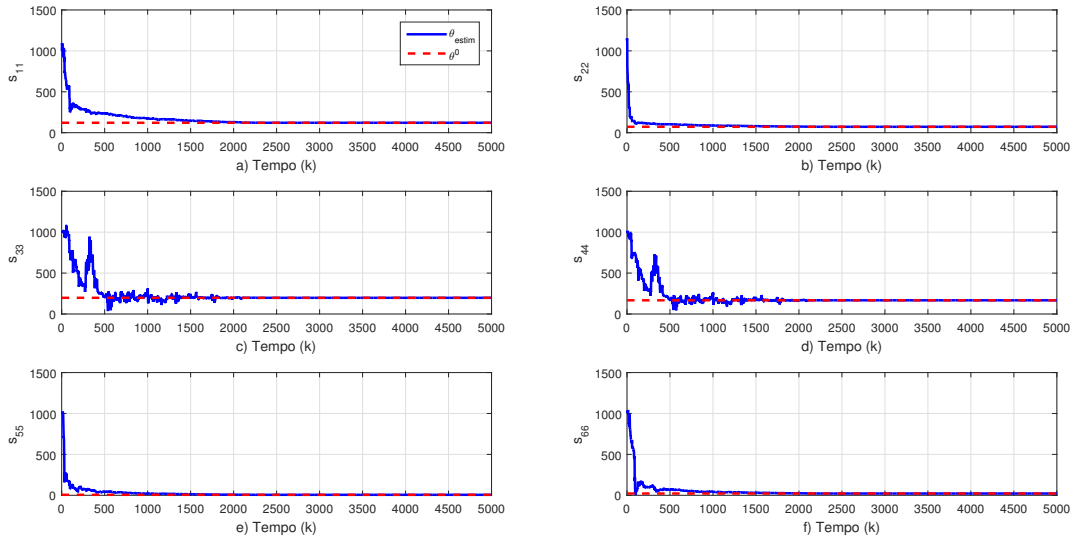
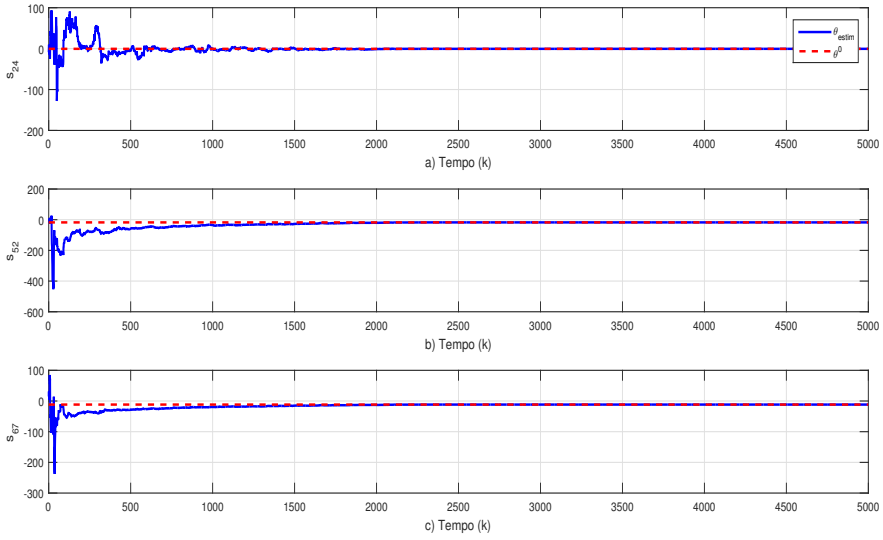


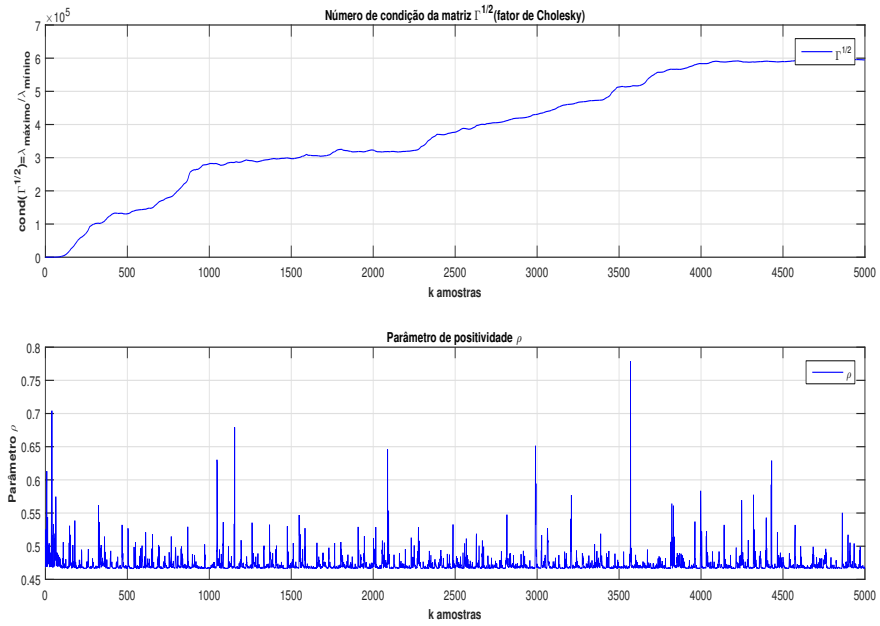
Figura 6.59: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.818 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



A Fig.6.60 apresenta o número de condição do Fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e o parâmetro de positividade ρ do estimador RLS$_\mu$-UDU^T-ADHDP-DLQR. Observa-se que o Fator Cholesky para o intervalo de iteração tende a entrar em uma faixa de valores constantes após um período de crescimento. Quanto ao parâmetro ρ, este apresenta-se dentro da faixa de interesse $0 < \rho <$

1.

Figura 6.60: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento μ de 0.874 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



A análise dos resultados para o fator de esquecimento $\mu = 0.874$ demonstra que os dois algoritmos, cujos núcleos tiveram o acréscimo das decomposições $\mathcal{QR}$ e UDU^T, respectivamente, alcançaram a convergência. Porém, nesse exemplo mostra-se a vantagem do algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR em relação ao algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR, uma vez que este apresenta um número de condição para o fator Cholesky em uma faixa de valores significativamente menor que o da estimação RLS$_\mu$-UDU^T-ADHDP-DLQR. Ambos os algoritmos alcançaram a convergência logo após as 2500 iterações, o que sugere uma matriz melhor condicionada (matriz correspondente ao fator Cholesky) pelas decomposições $\mathcal{QR}$ e UDU^T.

6.3.4 Simulação com $\mu = 0.919$

RLS$_\mu$-ADHDP-DLQR

As Figs.6.61 e 6.62 mostram a evolução do processo iterativo da solução da equação HJB-*Riccati* através do estimador RLS$_\mu$-ADHDP-DLQR para os parâmetros $s_{11}, s_{22}, s_{33}, s_{44}, s_{55}$ e s_{66}, assim como, para os parâmetros s_{24}, s_{52} e s_{67} da matriz S, respectivamente. Observa-se que não houve convergência dos parâmetros de S, os quais exibem comportamentos com perturbações abruptas e deslocamentos quando comparados a solução de referência para o pro-

cesso iterativo.

Figura 6.61: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.919 - RLS$_\mu$-ADHDP-DLQR.

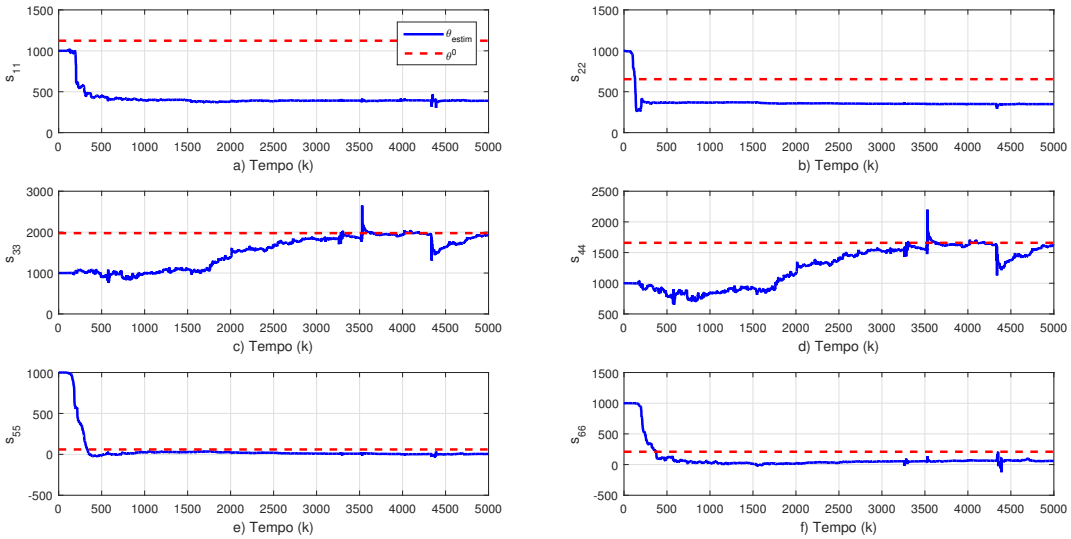
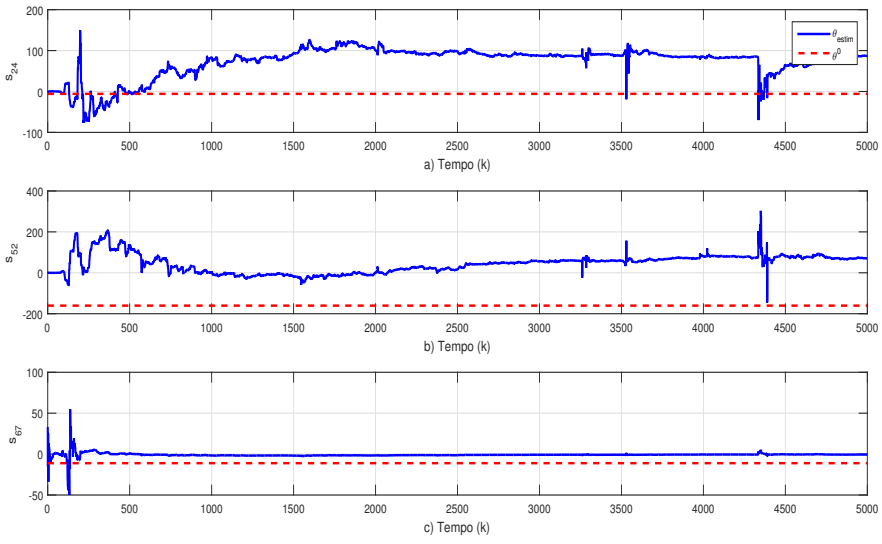


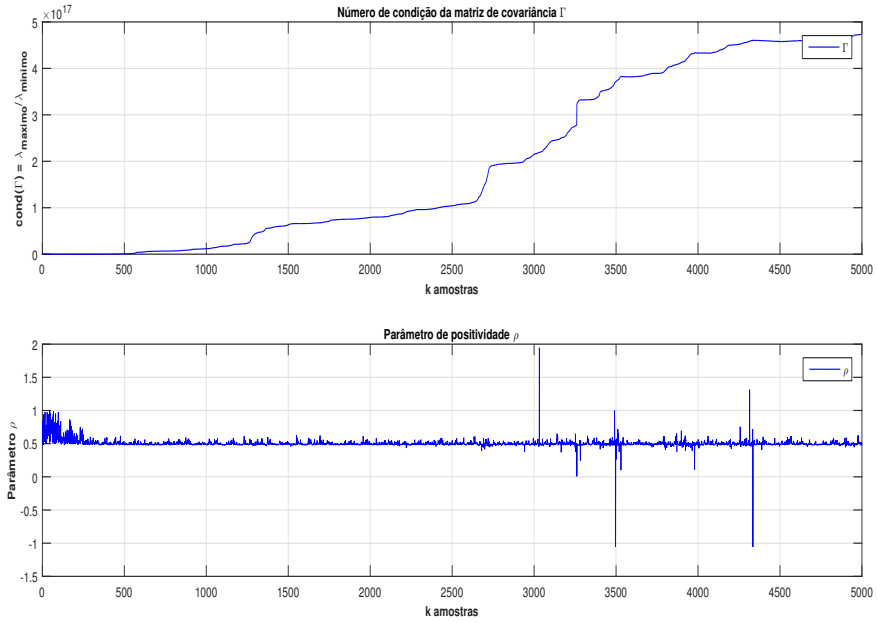
Figura 6.62: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.919 - RLS$_\mu$-ADHDP-DLQR.



A Fig.6.63 apresenta o número de condição e o parâmetro de positividade para o algoritmo RLS$_\mu$-ADHDP-DLQR. Como pode ser observado, para o algoritmo padrão os valores do parâmetro ρ apresentam oscilações que ultrapassando a faixa de referência $0 < \rho < 1$. Observa-se também a existência de uma correspondência entre os intervalos de variações do parâmetro ρ que excedem a faixa de referência e a perda de convergência dos elementos estimados do vetor

$\boldsymbol{\theta}$. Em geral, variações que excedem o intervalo $0 < \rho < 1$ para parâmetro de positividade, antecedem o comportamento irregular apresentado pelos elementos de $\boldsymbol{\theta}$.

Figura 6.63: Número de condição da matriz de covariância $\boldsymbol{\Gamma}_k$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.919 - RLS$_\mu$-ADHDP-DLQR.



RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

Nas Figs.6.64 e 6.65, apresenta-se a evolução do processo iterativo via algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR para solução da equação HJB-*Riccati* em um ciclo de 5000 iterações com fator de esquecimento de $\mu = 0.919$. As curvas (a)-(f) da Fig.6.64 representam o comportamento de convergência dos elementos $s_{11}, s_{22},\ s_{33},\ s_{44},\ s_{55}$ e s_{66} da matriz S correspondentes aos elementos $\theta_1,\ \theta_9,\ \theta_{16},\ \theta_{22},\ \theta_{27}$ e θ_{31} do vetor de parâmetros $\boldsymbol{\theta}$. Assim como, as curvas (a)-(c) da Fig.6.65 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos $\theta_{11},\ \theta_{12}$ e θ_{32} do vetor a ser estimado. Os elementos estimados não apresentam um grande *overshoot* para o início do processo iterativo, e os parâmetros tendem a seguir o sinal de referência de maneira suave atingindo a convergência logo após a iteração 2000.

O processo de estimação RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR tem seu número de condição do Fator Cholesky e parâmetro de positividade apresentados na Fig.6.66. Com respeito ao número de condição do Fator Cholesky percebe-se uma significativa redução quando comparado a estimação RLS$_\mu$-ADHDP-DLQR, assim como os valores do parâmetro ρ encontram-se no intervalo

Figura 6.64: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.919 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

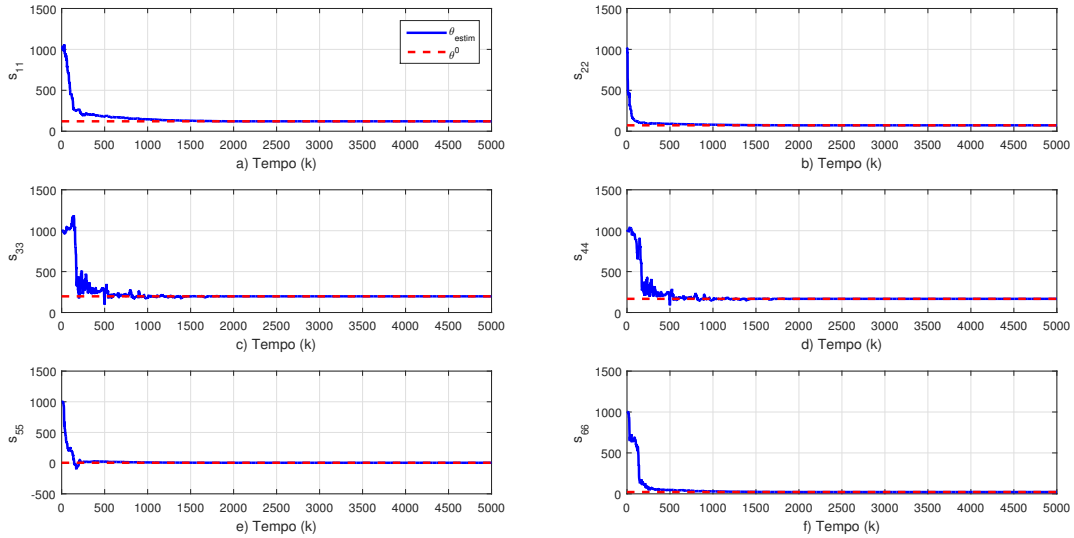
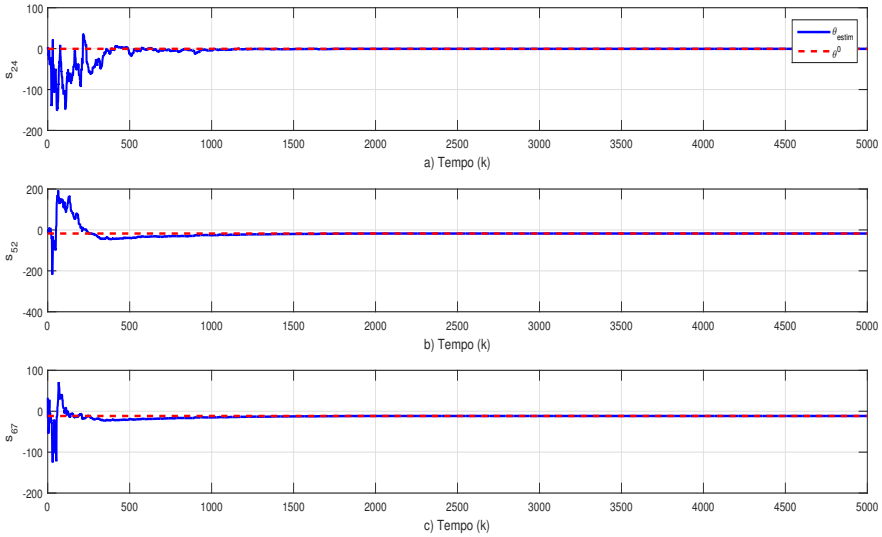


Figura 6.65: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.919 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

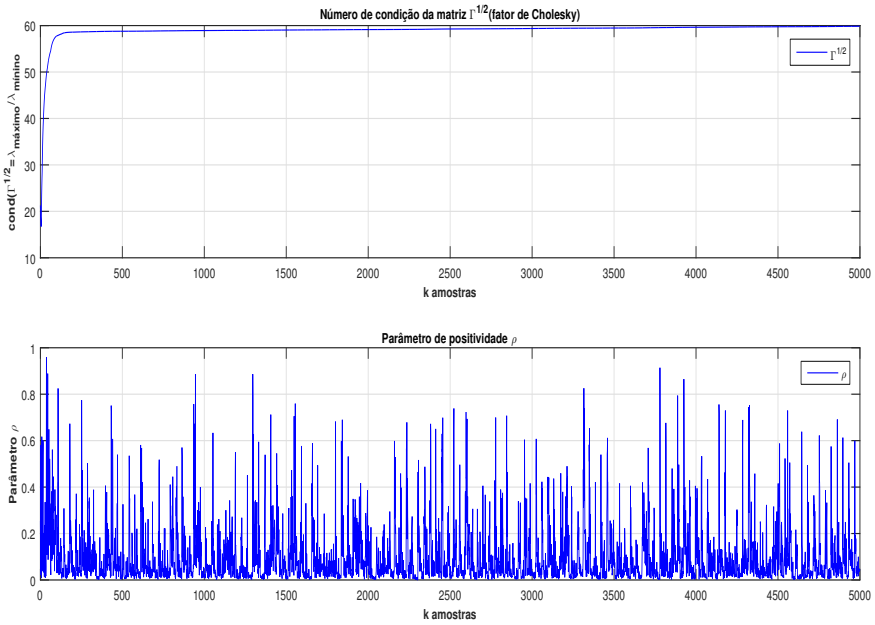


de validade durante todo o processo iterativo.

RLS$_\mu$-UDU^T-ADHDP-DLQR

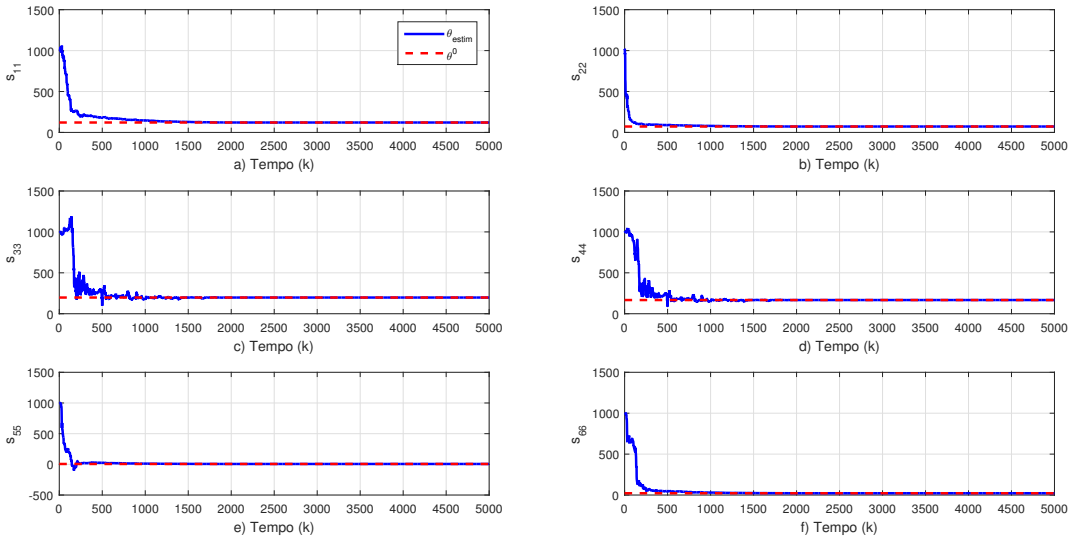
Nas Fig.6.67 e 6.68, apresenta-se a evolução do processo iterativo da solução da HJB-*Riccati* para um ciclo de 5000 iterações com fator de esquecimento $\mu = 0.919$ via algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR. As curvas (a)-(f) da Fig.6.67 representam o comportamento de convergência dos elementos $s_{11}, s_{22}, s_{33}, s_{44}, s_{55}$ e s_{66} da matriz S correspondentes às compo-

Figura 6.66: Número de condição do fator Cholesky $\boldsymbol{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.919 - RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.



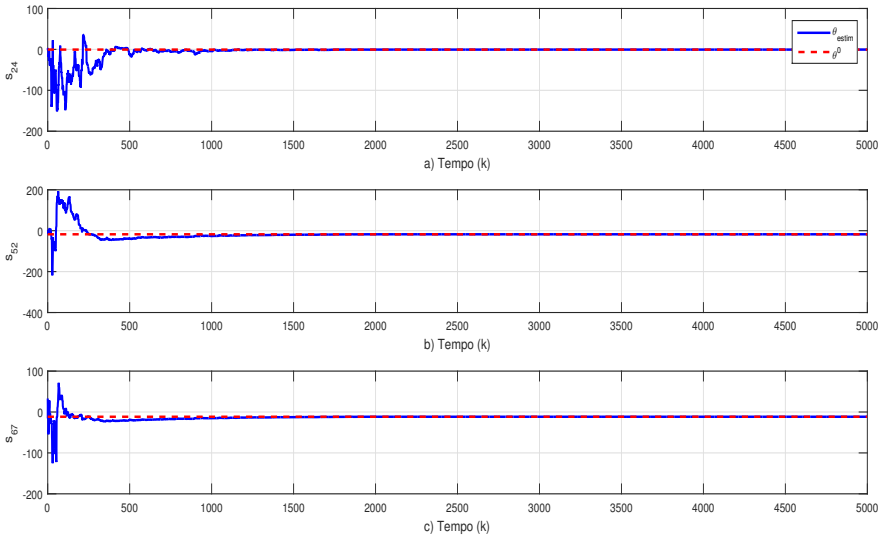
nentes $\theta_1,\ \theta_9,\ \theta_{16},\ \theta_{22},\ \theta_{27}$ e θ_{31} do vetor de parâmetros $\boldsymbol{\theta}$, respectivamente. As curvas (a)-(c) da Fig.6.68 mostram a evolução dos elementos s_{24}, s_{52} e s_{67} da matriz S que estão associados aos elementos $\theta_{11},\ \theta_{12}$ e θ_{32} do vetor $\boldsymbol{\theta}$, respectivamente.

Figura 6.67: Evolução do processo iterativo dos parâmetros s_{ii} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.919 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



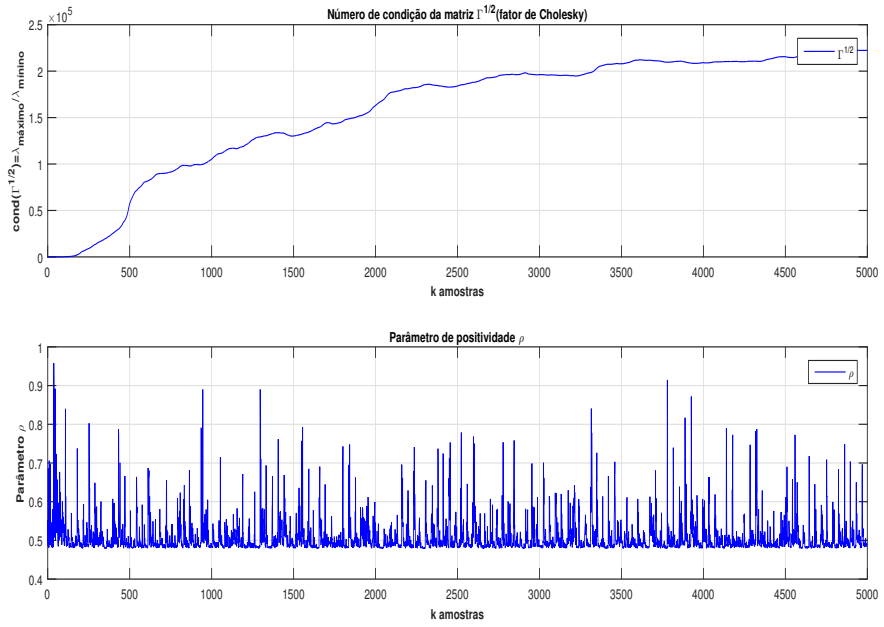
Ao análisar ao comportamento da matriz de covariância, não foram evidenciadas anormalidades, uma vez que o número de condição do Fator Cholesky, mesmo apresentando pico

Figura 6.68: Evolução do processo iterativo dos parâmetros s_{24}, s_{52} e s_{67} para um ciclo de 5000 iterações, com um fator de esquecimento μ de 0.919 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



crescente entre as iterações 250 à 3500, finda por permanecer numa faixa de oscilação na vizinhança do valor $\mathbf{\Gamma}^{1/2} = 2.25 \times 10^5$ logo após a iteração 3500. Quanto ao valor do parâmetro de positividade, este permanece sem extrapolar a faixa de validade, como ilustrado na Fig. 6.69.

Figura 6.69: Número de condição do fator Cholesky $\mathbf{\Gamma}_k^{1/2}$ e parâmetro de positividade ρ, com um fator de esquecimento de 0.919 - RLS$_\mu$-UDU^T-ADHDP-DLQR.



Analisando os resultados para o fator de esquecimento $\mu = 0.919$, pode-se dizer que os dois algoritmos munidos de esquemas de decomposições $\mathcal{QR}$ e UDU^T chegaram a conver-

gência dos elementos estimados e mais uma vez o algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR teve melhor desempenho em relação ao algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR, ambos chegaram a convergência próximo da iteração 2100, porém, o algoritmo com decomposição $\mathcal{QR}$ apresentou melhor desempenho, fato que só pode ser evidenciado quando se compara o número de condição da matriz $\mathbf{\Gamma}^{1/2}$ para ambas as abordagens, enquanto o algoritmo padrão não alcança a convergência dos elementos estimados.

Os resultados das simulações mostram claramente que as estimativas dadas pelos algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR para o sistema de quarta ordem apresenta, em geral, menor tempo de convergência (número de iterações necessárias para chegar a uma dada precisão) e custo computacional reduzido por iteração quando comparados com os sistemas de sexta ordem. Sendo necessário salientar que, para as plantas de ordem mais elevada, o custo computacional é maior e o estimador tende a apresentar maiores sensibilidades a perturbações provocadas por problemas numéricos, dessa forma o uso de métodos de decomposição para melhorar o comportamento do sistema torna-se imprescindível.

6.4 Considerações Finais

Neste Capítulo foram apresentados os resultados dos algoritmos propostos no Capítulo 4 em relação a duas plantas: Pêndulo Invertido sobre base móvel e Helicóptero 3-DOF. Os resultados mostraram que os algoritmos tem sua funcionalidade comprovada, podendo apresentar vantagens e desvantagens.

Levando em consideração os três parâmetros de avaliação de complexidade computacional têm-se:

- Custo Computacional: Conforme mostrado no Capitulo 5, o algoritmo que necessita de menor quantidade de $flops$ para sua execução é o RLS$_\mu$-UDU^T-ADHDP-DLQR. Isso garante que a execução de uma iteração desse algoritmo leva menos tempo do que a dos algoritmos RLS$_\mu$-ADHDP-DLQR e RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR.

- Estabilidade Numérica: Para este parâmetro, o algoritmo RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR, apresentou vantagem, garantindo em todos os casos, fator de positividade ρ dentro do intervalo fechado $[0, 1]$, e o número de condição do Fator Cholesky quando comparado com o dos algoritmos RLS$_\mu$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR mostrara-se sempre em menores amplitudes.

- Tempo de Convergência: Para este requisito, os algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR

e RLS$_\mu$-UDU^T-ADHDP-DLQR demostraram desempenho aprimorado, em detrimento da estimação padrão RLS$_\mu$-ADHDP-DLQR. Tal melhora dá-se em relação a quantidade de iterações necessárias para encontrar a solução desejada. Contudo, deve se lembrar que o algoritmo RLS$_\mu$-UDU^T-ADHDP-DLQR é mais rápido por iteração, por necessitar de menos $flops$, o que lhe fornece vantagem no cálculo real desse parâmetro.

Assim, pode-se concluir que as vantagens na execução dos algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR são maiores do que as do algoritmos RLS$_\mu$-ADHDP-DLQR.

7

CONCLUSÕES

Neste trabalho propõe-se um método crítico adaptativo, o qual envolve transformações unitárias e decomposições $\mathcal{QR}$ e UDU^T na rede crítica para melhorar o desempenho de algoritmos de ADHDP baseados em aprendizagem RLS para controle ótimo *online*. Os resultados de uma metodologia que investiga a convergência e estabilidade numérica dos algoritmos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR, desenvolvidos com iteração gulosa, para a solução DLQR *online* foram ressaltados.

Os algoritmos de programação dinâmica heurística dependente de ação foram formulados como uma aproximação da função valor de ação para uma dada política. A aproximação utilizou métodos incrementais RLS para resolver o problema de mínimos quadrados associado com a aproximação da função valor de ação sobre uma formulação apoiada pela teoria da vetorização de matrizes e álgebra de Kronecker. A aproximação foi apresentada para determinar os coeficientes da matriz S e os ganhos do controlador ótimo DLQR para ADHDP.

Em tentativa de aproximar, esses métodos de controle ótimo *online* a situações práticas do mundo real, buscou-se otimizar a execução dos algoritmos gerados por essas técnicas de controle a fim de tornar possível uma aplicação em tempo real em microcontroladores, presentes em sistemas de controle automático.

Na avaliação do desempenho dos algoritmos RLS$_\mu$-ADHDP-DLQR, RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR e RLS$_\mu$-UDU^T-ADHDP-DLQR, em relação à escolha do fator de esquecimento, observou-se uma grande influência desse fator no processo de convergência e estabilidade numérica na solução da equação HJB-*Riccati*. Verificou-se também que a escolha apropriada do fator juntamente com a decomposição $\mathcal{QR}$ e fatoração UDU^T podem contribuir significativamente para melhoria no processo de convergência da equação HJB-*Riccati*, assegurando ao mesmo tempo um limite aceitável para os problemas de instabilidade numérica dos estimadores RLS-ADHDP.

Os resultados dos experimentos RLS$_\mu$-$\mathcal{QR}$-ADHDP-DLQR revelam que a decomposi-

ção $\mathcal{QR}$ apresenta ligeira vantagem sobre a fatoração UDU^T e seu respectivo estimador RLS$_\mu$-UDU^T-ADHDP-DLQR, para uma seleção apropriada do fator de esquecimento. Mesmo não apresentando uma redução significativa do custo computacional, como foi o caso da estimação RLS$_\mu$-UDU^T, somente o processo de estimação munido de decomposição $\mathcal{QR}$ pôde incrementar melhorias substanciais nos métodos RLS$_\mu$-ADHDP-DLQR. Além disso, os resultados dos experimentos RLS$_\mu$-$\mathcal{QR}$-ADHDP evidenciaram que o ajuste adequado do parâmetro μ pode resolver os problemas de convergência e instabilidade numérica de maneira mais eficaz que a estimação RLS$_\mu$-UDU^T-ADHDP e RLS$_\mu$-ADHDP-DLQR para aprendizagem de política de controle ótima DLQR.

7.1 Trabahos Futuros

Para o desenvolvimento e direcionamento de futuros trabalhos, como extensão deste propõe-se:

- Realizar a implementação em *hardware* (Microcontroladores) dos algoritmos de controle propostos nesse trabalho;
- Avaliar a adaptabilidade dos estimadores propostos perante variações paramétricas na dinâmica da planta;
- Utilizar os métodos de estimação paramétrica RLS$_\mu$-$\mathcal{QR}$ e RLS$_\mu$-UDU^T em outros esquemas críticos adaptativos (DHP e ADDHP);
- Estender os métodos propostos para processos dinâmicos parcialmente observáveis nos quais nem todos os estados podem ser observados diretamente da planta;
- Estender os métodos desenvolvidos nesta dissertação para aproximadores de funções valor mais gerais. Neste caso, pode-se pensar em uma rede neural de múltiplas camadas treinada por métodos RLS.

7.2 Trabalhos Publicados e Submetidos

7.2.1 Congressos

1. **A Learning Controller Design Approach for a 3-DOF Helicopter System with Online Optimal Control**. International Conference on Electrical, Electronics, Computers, Communication, Mechanical and Computing (EECCMC) - IN (Aceito).

7.2.2 Periódicos

1. **Convergence and Numerical Stability of Action-Dependent Heuristic Dynamic Programming Algorithms based on RLS Learning for Online DLQR Optimal Control**, International Journal of Computational Science and Engineering, ISSN online: 1742-7193 (Submetido).

Apêndice A

Coeficientes da Fatoração UDU^T

Admitindo-se que

$$\boldsymbol{D}(k-1) - \beta^{-1}\boldsymbol{g}\boldsymbol{g}^T = \overline{\boldsymbol{U}}\,\overline{\boldsymbol{D}}\,\overline{\boldsymbol{U}}^T, \tag{A.1}$$

sendo $\overline{\boldsymbol{U}} = [\overline{\boldsymbol{u}}_1 \ \cdots \ \overline{\boldsymbol{u}}_{n_\theta}]$ e $\overline{\boldsymbol{D}} = diag(\overline{d}_1, \ \cdots, \ \overline{d}_{n_\theta})$, com $\overline{\boldsymbol{u}}_j = [\overline{u}_{1j} \ \cdots \ \overline{u}_{j-1j} \ 1 \ 0 \ \cdots \ 0]^T$, então os fatores $\boldsymbol{U}$-$\boldsymbol{D}$ de $\boldsymbol{\Gamma}(k)$ são dados por

$$\boldsymbol{U}(k) = \boldsymbol{U}(k-1)\overline{\boldsymbol{U}} \tag{A.2}$$

$$\boldsymbol{D}(k) = \overline{\boldsymbol{D}}\,\mu^{-1} \ . \tag{A.3}$$

A equação (A.1) pode ser transformada em

$$\sum_{i=1}^{n_\theta} \overline{d}_i \overline{\boldsymbol{u}}_i \overline{\boldsymbol{u}}_i^T = \sum_{i=1}^{n_\theta} d_i \boldsymbol{v}_i \boldsymbol{v}_i^T - \beta^{-1}\boldsymbol{g}\boldsymbol{g}^T, \tag{A.4}$$

sendo $d_i = d_i(k-1)$ e $\boldsymbol{v}_i$ é o i-ésimo versor.

Definindo-se

$$\beta_{n_\theta} = \beta \quad , \quad \beta_m = \mu + \sum_{i=1}^{m} e_i g_i \tag{A.5}$$

$$\boldsymbol{\psi}_{n_\theta} = \boldsymbol{g} \quad , \quad \boldsymbol{\psi}_{m-1} = [\psi_{m1} \ \ldots \ \psi_{mm-1} \ 0 \ \ldots \ 0]^T \ , \tag{A.6}$$

a Eq.(A.4) pode ser transformada para a forma

$$\sum_{i=1}^{n_\theta} \overline{d}_i \overline{\boldsymbol{u}}_i \overline{\boldsymbol{u}}_i^T - \sum_{i=1}^{n_\theta} d_i \boldsymbol{v}_i \boldsymbol{v}_i^T + \beta_{n_\theta}^{-1} \boldsymbol{\psi}_{n_\theta} \boldsymbol{\psi}_{n_\theta}^T = 0 \tag{A.7}$$

$$\sum_{i=1}^{n_\theta - 1} \overline{d}_i \overline{\boldsymbol{u}}_i \overline{\boldsymbol{u}}_i^T - \sum_{i=1}^{n_\theta - 1} d_i \boldsymbol{v}_i \boldsymbol{v}_i^T + \boldsymbol{M}_{n_\theta} = 0, \tag{A.8}$$

sendo

$$\boldsymbol{M}_{n_\theta} = \overline{d}_{n_\theta} \overline{\boldsymbol{u}}_{n_\theta} \overline{\boldsymbol{u}}_{n_\theta}^T - d_{n_\theta} \boldsymbol{v}_{n_\theta} \boldsymbol{v}_{n_\theta}^T + \beta_{n_\theta}^{-1} \boldsymbol{\psi}_{n_\theta} \boldsymbol{\psi}_{n_\theta}^T \tag{A.9}$$

Perceba que as matrizes sob o sinal dos somatórios na Eq.(A.8) possuem a n_θ-ésima linha e a n_θ-ésima coluna nulas. Portanto, a fim de que a Eq.(A.8), seja satisfeita deve-se estabelecer as seguintes condições sobre os elementos das respectivas fileiras da matriz $\boldsymbol{M}_{n_\theta}$

$$\overline{d}_{n_\theta} = d_{n_\theta} - \beta_{n_\theta}^{-1} \psi_{n_\theta n_\theta}^2 \tag{A.10}$$

$$\overline{u}_{i\, n_\theta} = \frac{-\psi_{n_\theta n_\theta} \psi_{n_\theta i}}{\beta_{n_\theta} \overline{d}_{n_\theta}} \tag{A.11}$$

Substituindo-se as Eqs.(A.10) e (A.11) na Eq.(A.9), a matriz $\boldsymbol{M}_{n_\theta}$ admite a forma

$$\begin{aligned} \boldsymbol{M}_{n_\theta} &= \overline{d}_{n_\theta} \frac{\psi_{n_\theta n_\theta}^2}{\overline{d}_{n_\theta}^2 \beta_{n_\theta}^2} \boldsymbol{\psi}_{n_\theta - 1} \boldsymbol{\psi}_{n_\theta - 1}^T + \beta_{n_\theta}^{-1} \boldsymbol{\psi}_{n_\theta - 1} \boldsymbol{\psi}_{n_\theta - 1}^T & \text{(A.12)} \\ &= \left(\frac{\psi_{n_\theta n_\theta}^2}{\overline{d}_{n_\theta}^2 \beta_{n_\theta}^2} + \frac{1}{\beta_{n_\theta}} \right) \boldsymbol{\psi}_{n_\theta - 1} \boldsymbol{\psi}_{n_\theta - 1}^T. & \text{(A.13)} \end{aligned}$$

ou, de acordo com as Eqs.(A.5), (A.6) e (A.10), tem-se

$$\begin{aligned} \boldsymbol{M}_{n_\theta} &= \frac{\psi_{n_\theta n_\theta}^2 + d_{n_\theta} \beta_{n_\theta} - \beta_{n_\theta}^{-1} \psi_{n_\theta n_\theta}^2 \beta_{n_\theta}}{\overline{d}_{n_\theta} \beta_{n_\theta}^2} \boldsymbol{\psi}_{n_\theta - 1} \boldsymbol{\psi}_{n_\theta - 1}^T & \text{(A.14)} \\ &= \frac{d_{n_\theta}}{d_{n_\theta} \beta_{n_\theta} - \beta_{n_\theta}^{-1} \psi_{n_\theta n_\theta}^2 \beta_{n_\theta}} \boldsymbol{\psi}_{n_\theta - 1} \boldsymbol{\psi}_{n_\theta - 1}^T & \text{(A.15)} \\ &= \frac{1}{\beta_{n_\theta} - \frac{\psi_{n_\theta n_\theta}^2}{d_{n_\theta}}} \boldsymbol{\psi}_{n_\theta - 1} \boldsymbol{\psi}_{n_\theta - 1}^T & \text{(A.16)} \\ &= \beta_{n_\theta - 1}^{-1} \boldsymbol{\psi}_{n_\theta - 1} \boldsymbol{\psi}_{n_\theta - 1}^T, & \text{(A.17)} \end{aligned}$$

e Eq.(A.7), pode assim ser expressa por

$$\sum_{i=1}^{n_\theta-1} \overline{d}_i \overline{\boldsymbol{u}}_i \overline{\boldsymbol{u}}_i^T - \sum_{i=1}^{n_\theta-1} d_i \boldsymbol{v}_i \boldsymbol{v}_i^T + \beta_{n_\theta-1}^{-1} \boldsymbol{\psi}_{n_\theta-1} \boldsymbol{\psi}_{n_\theta-1}^T = 0 \tag{A.18}$$

Deve-se enfatizar que as Eqs.(A.7) e (A.18) se diferem apenas pela variação do somatório e índices. Repetindo-se o mesmo raciocínio utilizado nas transformações (A.7)-(A.14) para índices variando de $n_\theta - 1$ a 1, é possível determinar todos os valores de d_i e u_{ij}.

A Eq.(A.11) é transformada usando as Eqs.(A.5), (A.6) e (A.10) como segue

$$\overline{u}_{i\,n_\theta} = \frac{-\psi_{n_\theta n_\theta}\psi_{n_\theta i}}{\beta_{n_\theta}\overline{d}_{n_\theta}} = \frac{-\psi_{n_\theta n_\theta}\psi_{n_\theta i}}{\beta_{n_\theta}\left(d_{n_\theta} - \frac{\psi_{n_\theta n_\theta}^2}{\beta_{n_\theta}}\right)} \tag{A.19}$$

$$= \frac{-\psi_{n_\theta n_\theta}\psi_{n_\theta i}}{d_{n_\theta}\left(\beta_{n_\theta} - \frac{\psi_{n_\theta n_\theta}^2}{d_{n_\theta}}\right)} = \frac{-\psi_{n_\theta n_\theta}\psi_{n_\theta i}}{d_{n_\theta}\beta_{n_\theta-1}} \tag{A.20}$$

Pondo-se

$$\tau_{n_\theta} = \frac{-\psi_{n_\theta n_\theta}}{d_{n_\theta}\beta_{n_\theta-1}} = \frac{-e_{n_\theta}}{\beta_{n_\theta-1}}, \tag{A.21}$$

obtém-se

$$\overline{u}_{i\,n_\theta} = -\tau_{n_\theta}\psi_{n_\theta i}. \tag{A.22}$$

Usando as Eqs.(A.5), (A.6) e (A.10), os valores dos elementos da matriz $\boldsymbol{D}$ são determinados da seguinte forma

$$d_i(k) = \overline{d}_i\mu^{-1} = \left(d_i - \frac{\psi_{ii}^2}{\beta_i}\right)\mu^{-1} \tag{A.23}$$

$$= d_i\frac{1}{\beta_i}\left(\beta_i - \frac{\psi_{ii}^2}{d_i}\right)\mu^{-1} = d_i\frac{\beta_{i-1}}{\beta_i\mu} \tag{A.24}$$

O numerador de (4.30) assume o seguinte forma

$$\boldsymbol{\kappa}(k) = \boldsymbol{U}(k-1)\boldsymbol{g} = \boldsymbol{U}(k-1)\boldsymbol{\psi}_{n_\theta} \tag{A.25}$$

Daí,

$$\kappa_i = \sum_{m=i}^{n_\theta} u_{im}(k-1)\psi_{n_\theta m} \tag{A.26}$$

$$= \sum_{m=i}^{n_\theta-1} u_{im}(k-1)\psi_{n_\theta m} + u_{i\,n_\theta}(k-1)\psi_{n_\theta\,n_\theta}, \tag{A.27}$$

ou, em forma de recorrência,

$$\kappa_{i,new} = \kappa_{i,old} + u_{in_\theta}(k-1)\psi_{n_\theta\,n_\theta}. \tag{A.28}$$

Os valores dos elementos da matriz $\boldsymbol{U}$ são determinados como segue

$$u_{ij}(k) = \sum_{m=i}^{j} u_{im}(k-1)\overline{u}_{mj} \tag{A.29}$$

$$= u_{ij}(k-1) + \sum_{m=i}^{j-1} u_{im}(k-1)\tau_j\psi_{jm} \tag{A.30}$$

$$= u_{ij}(k-1) + \tau_j \sum_{m=i}^{j-1} u_{im}(k-1)\psi_{jm} \tag{A.31}$$

$$= u_{ij}(k-1) + \tau_j\kappa_i \tag{A.32}$$

Apêndice B

Derivação dos valores dos blocos desconhecidos da pós-matriz B

Comparando-se os respectivos termos em ambos os lados da Eq.(4.27), as seguintes identidades são obtidas:

$$B_{11}^{H}(k)B_{11}(k) = \mu\boldsymbol{\Phi}(k-1) + \boldsymbol{\phi}(k)\boldsymbol{\phi}^{H}(k) = \boldsymbol{\Phi}(k) \tag{B.1}$$

$$b_{21}^{H}(i)B_{11}(k) = \mu\boldsymbol{\omega}^{H}(k-1)\boldsymbol{\Phi}^{H/2}(k-1) + d(k)\boldsymbol{\phi}^{H}(k) \tag{B.2}$$

$$b_{31}^{H}(k)B_{11}(k) = \boldsymbol{\phi}^{H}(k) \tag{B.3}$$

$$B_{11}^{H}(k)b_{21}(k) = \mu\boldsymbol{\Phi}^{1/2}(k-1)\boldsymbol{\omega}(k-1) + \boldsymbol{\phi}(k)d^{*}(k) \tag{B.4}$$

$$b_{21}^{H}(k)b_{21}(k) + b_{22}^{*}(k)b_{22}(k) = \mu\boldsymbol{\omega}^{H}(k-1)\boldsymbol{\omega}(k-1) + d(k)d^{*}(k) \tag{B.5}$$

$$b_{31}^{H}(k)b_{21}(k) + b_{32}^{*}(k)b_{22}(k) = d^{*}(k) \tag{B.6}$$

$$B_{11}^{H}(k)b_{31}(k) = \boldsymbol{\phi}(k) \tag{B.7}$$

$$b_{21}^{H}(k)b_{31}(k) + b_{22}^{*}(k)b_{32}(k) = d(k) \tag{B.8}$$

$$b_{31}^{H}(k)b_{31}(k) + b_{32}^{*}(k)b_{32}(k) = 1 \tag{B.9}$$

Desenvolvendo-se a Eq.(B.1), o valor $B_{11}(k)$ é encontrado:

$$B_{11}^H(k)B_{11}(k) = \Phi(k)$$

$$B_{11}^H(k) = \Phi^{1/2}(k) \tag{B.10}$$

Desenvolvendo-se a Eq.(B.2), o valor $b_{21}(k)$ é encontrado:

$$b_{21}^H(k)\mathbf{\Phi}^{H/2}(k) = \mu\boldsymbol{\omega}^H(k-1)\mathbf{\Phi}^{H/2}(k-1) + d(k)\boldsymbol{\phi}^H(k)$$

$$b_{21}^H(k)\mathbf{\Phi}^{H/2}(k) = \mu\hat{\boldsymbol{\theta}}^H(k-1)\mathbf{\Phi}^{1/2}(k-1)\mathbf{\Phi}^{H/2}(k-1) + d(k)\boldsymbol{\phi}^H(k)$$

$$b_{21}^H(k)\mathbf{\Phi}^{H/2}(k) = \mu\hat{\boldsymbol{\theta}}^H(k-1)\mathbf{\Phi}(k-1) + d(k)\boldsymbol{\phi}^H(k)$$

$$b_{21}^H(k)\mathbf{\Phi}^{H/2}(k) = \hat{\boldsymbol{\theta}}^H(k)\mathbf{\Phi}^H(k)$$

$$b_{21}^H(k) = \hat{\boldsymbol{\theta}}^H(k)\mathbf{\Phi}^{1/2}(k)$$

$$b_{21}^H(k) = \boldsymbol{\omega}^H(k) \tag{B.11}$$

Desenvolvendo-se a Eq.(B.3), o valor $b_{31}(k)$ é encontrado:

$$b_{31}^H(k)\mathbf{\Phi}^{H/2}(k) = \boldsymbol{\phi}^H(k)$$

$$b_{31}^H(k) = \boldsymbol{\phi}^H(k)\mathbf{\Phi}^{-H/2}(k) \tag{B.12}$$

Desenvolvendo a Eq.(B.9), o valor $b_{32}(k)$ é encontrado:

$$b_{32}^*(k)b_{32}(k) = 1 - b_{31}^H(k)b_{31}(k)$$

$$b_{32}^*(k)b_{32}(k) = 1 - [\boldsymbol{\phi}^H(k)\mathbf{\Phi}^{-H/2}(k)][\mathbf{\Phi}^{-1/2}(k)\boldsymbol{\phi}(k)]$$

$$b_{32}^{*}(k)b_{32}(k) = 1 - \boldsymbol{\phi}^{H}(k)\boldsymbol{\Phi}^{-1}(k)\boldsymbol{\phi}(k)$$

$$b_{32}^{*}(k)b_{32}(k) = 1 - \boldsymbol{\phi}^{H}(k)L(k)$$

$$b_{32}^{*}(k) = [1 - L^{H}(k)\boldsymbol{\phi}(k)]^{1/2}$$

fazendo $\rho(k) = 1 - L^{H}(k)\boldsymbol{\phi}(k)$ tem-se

$$b_{32}^{*}(k) = \rho^{1/2}(k) \tag{B.13}$$

Desenvolvendo a Eq.(B.8), o valor $b_{22}(k)$ é encontrado:

$$\boldsymbol{\omega}^{H}(k)\boldsymbol{\Phi}^{-1/2}(k)\boldsymbol{\phi}(k) + b_{22}^{*}(k)\left(\rho^{1/2}\right)^{*}(k) = d(k)$$

$$b_{22}^{*}(k)\left(\rho^{1/2}\right)^{*}(k) = d(k) - \boldsymbol{\omega}^{H}(k)\boldsymbol{\phi}(k)\boldsymbol{\Phi}^{1/2}(k)$$

$$b_{22}^{*}(k) = \left[d(k) - \boldsymbol{\omega}^{H}(k)\boldsymbol{\phi}(k)\boldsymbol{\Phi}^{1/2}(k)\right]\left(\rho^{-1/2}\right)^{*}(k)$$

$$b_{22}^{*}(k) = \left[d(k) - \boldsymbol{\phi}(k)\hat{\boldsymbol{\theta}}^{H}(k)\right]\left(\rho^{-1/2}\right)^{*}(k)$$

$$b_{22}^{*}(k) = \left\{d(k) - \boldsymbol{\phi}(k)\left[\hat{\boldsymbol{\theta}}^{H}(k-1) + L^{H}(k)\xi(k)\right]\right\}\left(\rho^{-1/2}\right)^{*}(k)$$

$$b_{22}^{*}(k) = \left[d(k) - \boldsymbol{\phi}(k)\hat{\boldsymbol{\theta}}^{H}(k-1) - \boldsymbol{\phi}(k)L^{H}(k)\xi(k)\right]\left(\rho^{-1/2}\right)^{*}(k)$$

$$b_{22}^{*}(k) = \left[\xi(k) - \boldsymbol{\phi}(k)L^{H}(k)\xi(k)\right]\left(\rho^{-1/2}\right)^{*}(k)$$

$$b_{22}^{*}(k) = \xi(k)\left[1 - \boldsymbol{\phi}(k)L^{H}(k)\right]\left(\rho^{-1/2}\right)^{*}(k)$$

$$b_{22}^{*}(k) = \xi(k)\rho(k)\left(\rho^{-1/2}\right)^{*}(k)$$

$$b_{22}^{*}(k) = \xi(k)\rho^{1/2}(k) \tag{B.14}$$

Apêndice C

Decomposição $\mathcal{QR}$

C.1 Introdução

Nesta seção se discutirá um tipo de decomposição de matrizes, que será utilizada na obtenção de soluções para problemas de mínimos quadrados.

A decomposição $\mathcal{QR}$ é uma operação elementar estável, que pode ser aplicada a qualquer tipo de matriz e consiste na fatoração de uma matriz dada em duas matrizes, uma matriz ortogonal e uma matriz triangular superior. Assim, a decomposição $\mathcal{QR}$ de uma matriz é apresentada como o produto de uma matriz ortogonal $\mathcal{Q}$ por uma matriz triangular superior $\mathcal{R}$.

Teorema C.1.1. *Seja A uma matriz $m \times n$ de posto n. Então*

$$A = \mathcal{QR},$$

em que $\mathcal{Q}$ é uma matriz $m \times n$ com colunas ortonormais e $\mathcal{R}$ é uma matriz $n \times n$ triangular superior com elementos diagonais positivos.

Demonstração: Sejam $v_1, \ldots, v_n$ as colunas da matriz A. Como essa matriz tem posto n, esses vetores são linearmente independentes em $\mathbb{C}^m$. Aplicando o processo de ortogonalização de Gram-Schmidt (BUENO, 2006) a esses vetores, obtemos os vetores ortonormais $q_1, \ldots, q_n \in \mathbb{C}^m$, dados por

$$q_k = \frac{1}{r_{kk}} \left(v_k - \sum_{i=1}^{k-1} r_{ik} q_i \right), \quad (k = 1, \ldots, n)$$

em que $r_{ik} = \langle v_k, q_i \rangle$ para $i = 1, \ldots, k-1$ e r_{kk} é a norma do vetor $v_k - \sum_{i=1}^{k-1} r_{ik} q_i$. Mas isso

quer dizer que

$$\begin{aligned} v_1 &= r_{11}q_1 \\ v_2 &= r_{12}q_1 + r_{22}q_2 \\ \vdots &\quad \vdots \\ v_n &= r_{1n}q_1 + \cdots + r_{nn}q_n. \end{aligned} \tag{C.1}$$

Definindo $\mathcal{Q}$ como a matriz cujas colunas são os vetores $q_1, \ldots, q_n$ e $\mathcal{R}$ a matriz triangular superior

$$\mathcal{R} = \begin{bmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ 0 & r_{21} & \ldots & r_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & r_{nn} \end{bmatrix} = [r_1\ r_2\ \ldots r_n], \tag{C.2}$$

temos que a j-ésima coluna da matriz $\mathcal{QR}$ é

$$\mathcal{QR}e_j = \mathcal{Q}r_j = r_{1j}q_1 + r_{2j}q_2 + \cdots + r_{jj}q_j + 0q_{j+1} + \cdots + 0q_n = v_j. \tag{C.3}$$

Isso mostra que $\mathcal{QR} = A$, completando a demonstração.

C.2 Rotações de Givens

Uma matriz de Givens Θ_{ij} é uma matriz ortogonal de dimensões $(n \times n)$ da forma:

$$\Theta_{ij} = \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & c & \cdots & -s & & \\ & & \vdots & \ddots & \vdots & & \\ & & s & \cdots & c & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{bmatrix} \tag{C.4}$$

onde $\theta_{ii} = \theta_{jj} = c$ e $\theta_{ij} = -\theta_{ji} = s$. Note que a intersecção das linhas i e j com as colunas i e j de Θ_{ij} formam assim uma ferramenta algébrica fundamental às Rotações de Givens, onde:

$$\Theta = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{C.5}$$

uma vez que c e s são parâmetros reais definidos por

$$c = \cos\phi \quad \text{e} \quad s = \sin\phi, \tag{C.6}$$

com a seguinte restrição trigonométrica $c^2 + s^2 = 1$.

Dessa forma a matriz Θ é ortogonal, pois

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} c^2 + s^2 & 0 \\ 0 & c^2 + s^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{C.7}$$

Podemos concluir, assim, que as matrizes Θ_{ij} são ortogonais. Essas matrizes são chamadas de rotações pois, se a e x satisfazem:

$$x \;=\; \Theta a = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a_i \\ a_k \end{bmatrix} = \begin{bmatrix} ca_i + sa_k \\ -sa_i + ca_k \end{bmatrix}, \tag{C.8}$$

então a transformação Θ rotaciona o vetor a em sentido horário para uma nova posição definida por x.

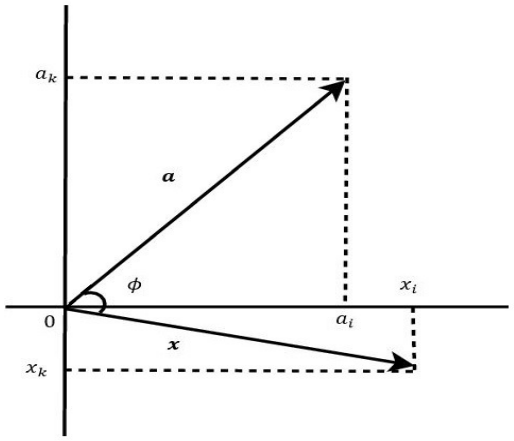


Figura C.1: Rotação plana de um vetor

Multiplicar um vetor por Θ_{ij} equivale a girar duas componentes do vetor pelo ângulo

$\phi = \arctan\left(\frac{s}{c}\right)$, sem alterações nas demais componentes.

Seja a um vetor com duas componentes e definimos

$$s = -\frac{a_k}{a_i}c \quad \text{e} \quad c = \frac{a_i}{\sqrt{a_i^2 + a_k^2}}. \tag{C.9}$$

Observe que $c^2 + s^2 = 1$ e

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a_i \\ a_k \end{bmatrix} = \begin{bmatrix} \sqrt{a_i^2 + a_k^2} \\ 0 \end{bmatrix},$$

Resultando em um processo iterativo que busca o zeramento de elementos da matriz através de multiplicações como visto na abaixo:

$$\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \rightarrow \Theta_{21}^T \begin{bmatrix} x & x & x \\ 0 & x & x \\ x & x & x \end{bmatrix} \rightarrow \Theta_{31}^T \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix} \rightarrow \Theta_{32}^T \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{bmatrix} \tag{C.10}$$

Dessa forma as matrizes $\mathcal{Q}$ e $\mathcal{R}$ são dadas por:

$$\mathcal{Q} = \Theta_{21}\Theta_{31}\ldots\Theta_{ij} \tag{C.11}$$

$$\mathcal{R} = \Theta_{ij}^T\ldots\Theta_{21}^T A \tag{C.12}$$

REFERÊNCIAS

AL-TAMIMI, A. et al. Model-free approximate dynamic programming schemes for linear systems. In: IEEE. *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*. [S.l.], 2007. p. 371–378.

ANDERSON, B.; MOORE, J. *Optimal Control: Linear Quadratic Methods*. [S.l.]: Dover Publications, 2007. (Dover Books on Engineering). ISBN 9780486457666.

ANDERSON, C. W. Strategy learning with multilayer connectionist representations. In: *Proceedings of the Fourth International Workshop on Machine Learning*. [S.l.: s.n.], 1987. p. 103–114.

ATHANS, M.; FALB, P. L. *Optimal control: an introduction to the theory and its applications*. [S.l.]: Dover Publications, 2007.

BARTO, A. G.; SUTTON, R. S.; ANDERSON, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, IEEE, n. 5, p. 834–846, 1983.

BELLMAN, R. *Dynamic programming and stochastic control processes*. *Information and control*, Elsevier, v. 1, n. 3, p. 228–239, 1958.

BELLMAN, R. *Dynamic programming*. [S.l.]: Dover Publications, INC., 2003.

BERTSEKAS, D. P. Dynamic programming: deterministic and stochastic models. Prentice-Hall, Inc., 1987.

BERTSEKAS, D. P. *Nonlinear programming*. [S.l.]: Athena scientific Belmont, 1999.

BERTSEKAS, D. P. et al. *Dynamic programming and optimal control*. [S.l.]: Athena scientific Belmont, MA, 1995. v. 1.

BERTSEKAS, D. P. et al. Missile defense and interceptor allocation by neuro-dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, IEEE, v. 30, n. 1, p. 42–51, 2000.

BERTSEKAS, D. P.; TSITSIKLIS, J. N. Neuro-dynamic programming: an overview. In: IEEE. *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*. [S.l.], 1995. v. 1, p. 560–564.

BIERMAN, G. J. Factorization methods for discrete sequential estimation. Academic Press, 1977.

BRADTKE, S. J.; YDSTIE, B. E.; BARTO, A. G. Adaptive linear quadratic control using policy iteration. In: IEEE. *American Control Conference, 1994*. [S.l.], 1994. v. 3, p. 3475–3479.

BREGANON, R. *Controle de arfagem e guinada de um sistema de hélices paralelas*. Tese (Doutorado) — Universidade de São Paulo, 2009.

BREWER, J. Kronecker products and matrix calculus in system theory. *IEEE Transactions on circuits and systems*, IEEE, v. 25, n. 9, p. 772–781, 1978.

BRYSON, A. E. *Applied optimal control: optimization, estimation and control*. [S.l.]: CRC Press, 1975.

BUENO, H. P. *Álgebra linear: um segundo curso*. [S.l.]: Sociedade Brasileira de Matemática, 2006. v. 1. ISBN 85-85818-31-X.

BUSONIU, L. et al. *Reinforcement learning and dynamic programming using function approximators*. [S.l.]: CRC press, 2010. v. 39.

BUTTAZZO, G. C. *Hard real-time computing systems: predictable scheduling algorithms and applications*. [S.l.]: Springer Science & Business Media, 2011. v. 24.

CAMPOS, M. L.; STRANG, G. Qr decomposition: An annotated bibliography. *QRD-RLS Adaptive Filtering*, Springer, p. 1, 2009.

CHEN, C.-T. *Linear system theory and design*. [S.l.]: Oxford University Press, Inc., 1995.

CHENG, Y.; FENG, H.; WANG, X. Efficient data use in incremental actor–critic algorithms. *Neurocomputing*, Elsevier, v. 116, p. 346–354, 2013.

CHU, B. et al. Passive dynamic walker controller design employing an rls-based natural actor–critic learning algorithm. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 21, n. 7, p. 1027–1034, 2008.

CORMEN, T. H. et al. *Introduction to algorithms second edition*. [S.l.]: The MIT Press, 2001.

DIERKS, T.; JAGANNATHAN, S. Online optimal control of nonlinear discrete-time systems using approximate dynamic programming. *Journal of Control Theory and Applications*, Springer, v. 9, n. 3, p. 361–369, 2011.

DOWNEY, R. G.; FELLOWS, M. R.; STEGE, U. Parameterized complexity: A framework for systematically confronting computational intractability. In: *Contemporary trends in discrete mathematics: From DIMACS and DIMATIA to the future*. [S.l.: s.n.], 1999. v. 49, p. 49–99.

FERRARI, S.; STENGEL, R. F. Online adaptive critic flight control. *Journal of Guidance Control and Dynamics*, New York: The Institute, 1982-, v. 27, n. 5, p. 777–786, 2004.

FERREIRA, E. F.; NETO, J. V. da F.; RÊGO, P. H. M. Computational performance of state-value function approximators based on rls-hdp estimators for online dlqr control system design. In: IEEE. *Computer Modelling and Simulation (UKSim), 2016 UKSim-AMSS 18th International Conference on*. [S.l.], 2016. p. 27–34.

FERREIRA, E. F.; RÊGO, P. H.; NETO, J. V. Numerical stability improvements of state-value function approximations based on rls learning for online hdp-dlqr control system design. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 63, p. 1–19, 2017.

FONSECA, J.; RÊGO, P. Qr-tuning and approximate-ls solutions of the hjb equation for online dlqr design via state and actiondependent heuristic dynamic programming. *International Journal of Innovative Computing, Information and Control (IJICIC).*, v. 10, n. 3, p. 1071–1094, 2014.

GEIST, M.; PIETQUIN, O.; FRICOUT, G. Kalman temporal differences: The deterministic case. In: *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL '09. IEEE Symposium on*. [S.l.: s.n.], 2009. p. 185–192.

GENTLEMAN, W. M. Least squares computations by givens transformations without square roots. *IMA Journal of Applied Mathematics*, Oxford University Press, v. 12, n. 3, p. 329–336, 1973.

GOLDREICH, O. Computational complexity: a conceptual perspective. *ACM Sigact News*, ACM, v. 39, n. 3, p. 35–39, 2008.

GOLUB, G. H.; LOAN, C. F. V. Matrix computations. 1996. *Johns Hopkins University, Press, Baltimore, MD, USA*, p. 374–426, 1996.

GOLUB, G. H.; LOAN, C. F. V. *Matrix computations*. [S.l.]: JHU Press, 2012. v. 3.

HANSELMANN, T.; MUSICKI, D.; PALANISWAMI, M. Adaptive target tracking in slowly changing clutter. In: IEEE. *Information Fusion, 2006 9th International Conference on*. [S.l.], 2006. p. 1–8.

HARTMANIS, J.; STEARNS, R. E. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, JSTOR, v. 117, p. 285–306, 1965.

HAYKIN, S. A comprehensive foundation. *Neural Networks*, v. 2, n. 2004, p. 41, 2004.

HAYKIN, S. S. *Adaptive filter theory*. [S.l.]: Pearson Education India, 2008.

HOWARD, R. A. Dynamic programming and markov processes.. 1960.

ISHUTKINA, M. A. *Design and implimentation of a supervisory safety controller for a 3DOF helicopter*. Tese (Doutorado) — Massachusetts Institute of Technology, 2004.

KAMINSKI, P.; BRYSON, A.; SCHMIDT, S. Discrete square root filtering: A survey of current techniques. *IEEE Transactions on Automatic Control*, IEEE, v. 16, n. 6, p. 727–736, 1971.

KHAN, S. G. et al. Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control*, Elsevier, v. 36, n. 1, p. 42–59, 2012.

KIRK, D. E. *Optimal control theory: an introduction*. [S.l.]: Courier Corporation, 2004.

KITAHARA, R. T.; SARIDIS, G. N. Computational aspects of performance-adaptive self-organizing control algorithms. In: IEEE. *Decision and Control, 1972 and 11th Symposium on Adaptive Processes. Proceedings of the 1972 IEEE Conference on*. [S.l.], 1972. v. 11, p. 642–646.

KUNG, H.; GENTLEMAN, W. Matrix triangularization by systolic arrays. 1982.

LANCASTER, P.; RODMAN, L. *Algebraic riccati equations*. [S.l.]: Clarendon press, 1995.

LANDELIUS, T. *Reinforcement learning and distributed local model synthesis*. Tese (Doutorado) — Linköping University Electronic Press, 1997.

LAUB, A. A schur method for solving algebraic riccati equations. *IEEE Transactions on automatic control*, IEEE, v. 24, n. 6, p. 913–921, 1979.

LEE, J. Y.; PARK, J. B.; CHOI, Y. H. Policy-iteration-based adaptive optimal control for uncertain continuous-time linear systems with excitation signals. In: IEEE. *Control Automation and Systems (ICCAS), 2010 International Conference on*. [S.l.], 2010. p. 464–651.

LENDARIS, G. G. Adaptive dynamic programming approach to experience-based systems identification and control. *Neural Networks*, Elsevier, v. 22, n. 5, p. 822–832, 2009.

LENDARIS, G. G. A retrospective on adaptive dynamic programming for control. In: IEEE. *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. [S.l.], 2009. p. 1750–1757.

LEWIS, F. L.; LIU, D. *Reinforcement learning and approximate dynamic programming for feedback control*. [S.l.]: John Wiley & Sons, 2012. v. 17.

LEWIS, F. L.; VRABIE, D. Adaptive dynamic programming for feedback control. In: IEEE. *Asian Control Conference, 2009. ASCC 2009. 7th*. [S.l.], 2009. p. 1402–1409.

LEWIS, F. L.; VRABIE, D. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE circuits and systems magazine*, IEEE, v. 9, n. 3, 2009.

LEWIS, F. L.; VRABIE, D.; VAMVOUDAKIS, K. G. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems*, IEEE, v. 32, n. 6, p. 76–105, 2012.

LI, L.; LITTMAN, M. L.; MANSLEY, C. R. Online exploration in least-squares policy iteration. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. [S.l.], 2009. p. 733–739.

LIAVAS, A. P.; REGALIA, P. A. On the numerical stability and accuracy of the conventional recursive least squares algorithm. *IEEE Transactions on Signal Processing*, IEEE, v. 47, n. 1, p. 88–96, 1999.

LIN, C.-K. Adaptive critic autopilot design of bank-to-turn missiles using fuzzy basis function networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 35, n. 2, p. 197–207, 2005.

LJUNG, S.; LJUNG, L. Error propagation properties of recursive least-squares adaptation algorithms. *Automatica*, Elsevier, v. 21, n. 2, p. 157–167, 1985.

LOPES, R. V. Modelagem e controle preditivo de um helicóptero com três graus de liberdade. *136f. Dissertations (Mestrado em Sistemas e Controle)-Instituto Tecnológico de Aeronáutica, São José dos Campos*, 2007.

MAIA, M. H. *Controle preditivo robusto de um helicóptero com três graus de liberdade sujeito a perturbações externas*. Tese (Doutorado) — Brasil. Ministério da Defesa. Comando-Geral de Tecnologia Aeroespacial (CTA). Instituto Tecnológico de Aeronáutica (ITA), 2008.

MCWHIRTER, J. Recursive least-squares minimization using a systolic array. *Real-Time Processing VI, Aug. 1983*, p. 105–112, 1983.

MCWHIRTER, J. Systolic array for recursive least-squares minimisation. *Electronics Letters*, IET, v. 19, n. 18, p. 729–730, 1983.

MURRAY, J. J. et al. Adaptive dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 32, n. 2, p. 140–153, 2002.

PAPOULIS, A.; PILLAI, S. U. *Probability, random variables, and stochastic processes*. [S.l.]: Tata McGraw-Hill Education, 2002.

PAVLOV, I. P.; ANREP, G. V. *Conditioned reflexes*. [S.l.]: Dover Publications, Inc., 2003.

PIETQUIN, O.; GEIST, M.; CHANDRAMOHAN, S. Sample efficient on-line learning of optimal dialogue policies with kalman temporal differences. In: *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*. AAAI Press, 2011. (IJCAI'11, v. 3), p. 1878–1883. ISBN 978-1-57735-515-1. Disponível em: <http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-314>.

POTTER, J. New statistical formulas,". *Space Guidance Analysis Memo*, v. 40, p. 1309–1314, 1963.

PRASAD, L. B.; GUPTA, H. O.; TYAGI, B. Adaptive optimal control of nonlinear inverted pendulum system using policy iteration technique. *IFAC Proceedings Volumes*, Elsevier, v. 47, n. 1, p. 1138–1145, 2014.

QUIRION, J.; GUNN, E.; GU, J. Optimal control of permanent magnet motors using dynamic programming. In: IEEE. *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*. [S.l.], 2004. v. 1, p. 364–369.

RÊGO, P. H. M. *Aprendizagem por Reforço e Programação Dinâmica Aproximada para Controle Ótimo: Uma Abordagem para o Projeto Online do Regulador Linear Quadrático Discreto com Programação Dinâmica Heurística Dependente de Estado E Ação*. Tese (Doutorado) — Universidade Federal do Maranhão, 2014.

RÊGO, P. H. M.; NETO, J. V. d. F.; FERREIRA, E. M. Convergence of the standard rls method and udu t factorisation of covariance matrix for solving the algebraic riccati equation of the dlqr via heuristic approximate dynamic programming. *International Journal of Systems Science*, Taylor & Francis, v. 46, n. 11, p. 2006–2028, 2013. Disponível em: <http://dx.doi.org/10.1080/00207721.2013.844283>.

ROMANO, J. M. T. *Sobre a Estabilidade Numérica dos Algoritmos de Mínimos Quadrados Rápidos*. Tese (Doutorado) — UNICAMP, Janeiro 1995.

SAPIŃSKI, B.; ROSÓŁ, M. Autonomous control system for a 3 dof pitch-plane suspension model with mr shock absorbers. *Computers & structures*, Elsevier, v. 86, n. 3, p. 379–385, 2008. Disponível em: <https://doi.org/10.1016/j.compstruc.2007.02.017>.

SI, J. *Handbook of learning and approximate dynamic programming*. [S.l.]: John Wiley & Sons, 2004. v. 2.

SIPSER, M. *Introduction to the Theory of Computation*. [S.l.]: Thomson Course Technology Boston, 2006. v. 2.

SLOCK, D. T. Backward consistency concept and round-off error propagation dynamics in recursive least-squares algorithms. *Optical Engineering*, International Society for Optics and Photonics, v. 31, n. 6, p. 1153–1169, 1992.

SOKOLOV, Y. et al. Complete stability analysis of a heuristic approximate dynamic programming control design. *Automatica*, Elsevier, v. 59, p. 9–18, 2015.

STANKOVIC, J. A. Misconceptions about real-time computing: A serious problem for next-generation systems. *Computer*, IEEE, v. 21, n. 10, p. 10–19, 1988.

STINGU, P. E.; LEWIS, F. L. Adaptive dynamic programming applied to a 6dof quadrotor. In: *Computational Modeling and Simulation of Intellect: Current State and Future Perspectives*. [S.l.]: IGI Global, 2011. p. 102–130.

SUTTON, R. S. Temporal credit assignment in reinforcement learning. 1984.

SUTTON, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, Springer, v. 3, n. 1, p. 9–44, 1988.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press Cambridge, 1998. v. 1.

SUTTON, R. S.; BARTO, A. G.; WILLIAMS, R. J. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems*, IEEE, v. 12, n. 2, p. 19–22, 1992.

VERHAEGEN, M. Round-off error propagation in four generally-applicable, recursive, least-squares estimation schemes. *Automatica*, Elsevier, v. 25, n. 3, p. 437–444, 1989.

VRABIE, D.; LEWIS, F. L. Adaptive optimal control algorithm for continuous-time nonlinear systems based on policy iteration. In: IEEE. *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. [S.l.], 2008. p. 73–79.

WANG, F.-Y.; ZHANG, H.; LIU, D. Adaptive dynamic programming: An introduction. *IEEE computational intelligence magazine*, IEEE, v. 4, n. 2, 2009.

WARD, C.; HARGRAVE, P.; MCWHIRTER, J. A novel algorithm and architecture for adaptive digital beamforming. *IEEE Transactions on Antennas and Propagation*, IEEE, v. 34, n. 3, p. 338–346, 1986.

WATKINS, C. J.; DAYAN, P. Q-learning. *Machine learning*, Springer, v. 8, n. 3-4, p. 279–292, 1992.

WATKINS, C. J. C. H. *Learning from delayed rewards*. Tese (Doutorado) — King's College, Cambridge, 1989.

WERBOS, P. J. A menu of designs for reinforcement learning over time. *Neural networks for control*, Cambridge, MA: MIT Press, p. 67–95, 1990.

WERBOS, P. J. Approximate dynamic programming for real-time control and neural modeling. *Handbook of intelligent control*, Van Nostrand Reinhold, p. 493–526, 1992.

WILKINSON, J. H. A priori error analysis of algebraic processes. 1968.

XU, X.; HE, H.-g.; HU, D. Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research*, v. 16, p. 259–292, 2002.

YEH, J.-W.; SU, S.-F. Learning analysis for correlation of fuzzy rules in applying rls for neural fuzzy systems. In: IEEE. *Granular Computing (GrC), 2012 IEEE International Conference on*. [S.l.], 2012. p. 609–613.

YEH, J.-W.; SU, S.-F.; RUDAS, I. Analysis of using rls in neural fuzzy systems. In: IEEE. *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*. [S.l.], 2011. p. 1831–1836.