Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Programa de Pós-Graduação em Engenharia de Eletricidade

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

---

Patrícia Helena Moraes Rêgo

São Luís
2014

Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Programa de Pós-Graduação em Engenharia de Eletricidade

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

---

Patrícia Helena Moraes Rêgo

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos necessários para a obtenção do título de Doutora em Engenharia de Eletricidade na área de Automação e Controle.

São Luís
2014

Rêgo, Patrícia Helena Moraes

Aprendizagem por reforço e programação dinâmica aproximada para controle ótimo: uma abordagem para o projeto online do regulador linear... / Patrícia Helena Moraes Rêgo. - São Luís, 2014.

291f.

Impresso por computador (fotocópia).

Orientador: João Viana da Fonseca Neto.

Tese (Doutorado) - Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade, 2014.

1. Programação dinâmica. 2. Aprendizagem por reforço. 3. Controle multivariável.. I.Título.

CDU 621.3: 004

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

Patrícia Helena Moraes Rêgo

Tese aprovada em 24 de julho de 2014.

Prof. João Viana da Fonseca Neto, Dr.
(Orientador)

Prof. Raimundo Carlos Silvério Freire, Dr.
(Membro da Banca Examinadora)

Prof. Roberto Célio Limão de Oliveira, Dr.
(Membro da Banca Examinadora)

Prof. Ginalber Luiz de Oliveira Serra, Dr.
(Membro da Banca Examinadora)

Prof. Francisco das Chagas de Souza, Dr.
(Membro da Banca Examinadora)

"A ciência vive de sucessivas soluções dadas a porquês cada vez mais sutis, cada vez mais próximos à essência dos fenômenos". **Pasteur**

À minha mãe (*in memoriam*), Enilde Rodrigues Moraes Rêgo, pela sua incansável dedicação dispensada a mim, e pelo amor, compreensão e carinho que sempre estiveram presentes nas nossas relações, e neste sentido, venho enfatizar que os seus ensinamentos são parâmetros que utilizo na condução de meus objetivos e o quanto é satisfatório render homenagens a quem está sempre viva em nossa memória.

Agradecimentos

A Deus minha fonte de inspiração, conhecimento e sabedoria.

À minha mãe (*in memoriam*) que me proporcionou os valores que deram base para o meu futuro, e aos meus familiares.

Ao Prof. Dr. João Viana da Fonseca Neto pelo grande empenho como orientador, desde o projeto de dissertação do mestrado, fato que enfatizo a sua participação com grande relevância no que tange a discussão da pesquisa científica como mola propulsora para geração do conhecimento, agradeço também, pelo incentivo que ele me despertou para esta difícil linha de pesquisa na área de Automação e Controle.

À Universidade Estadual do Maranhão (UEMA) por ter oportunizado a minha participação e inclusão ao PPGEE e aos meus colegas do Departamento de Matemática e Informática, pelo incentivo, e finalmente aos professores que compõem a banca examinadora, e meus amigos de laboratórios (LCP e LABSECI) que contribuíram para realização deste trabalho cujo fim é a promoção do bem comum.

Enfim, a todos que colaboraram para que este propósito se concretizasse, fazendo sugestões, trocando idéias e apoiando-nos.

Resumo

Apresenta-se nesta tese uma proposta de uma abordagem unificada de teorias de programação dinâmica, aprendizagem por reforço e aproximação de função que tem por objetivo o desenvolvimento de métodos e algoritmos para projeto *online* de sistemas de controle ótimo. Esta abordagem é apresentada no contexto de programação dinâmica aproximada que permite aproximar a solução de realimentação ótima de modo a reduzir a complexidade computacional associada com métodos convencionais de programação dinâmica para controle ótimo de sistemas multivariáveis. Especificamente, no quadro de programação dinâmica heurística e programação dinâmica heurística dependente de ação, esta proposta é orientada para o desenvolvimento de soluções aproximadas *online*, numericamente estáveis, da equação de *Hamilton-Jacobi-Bellman* do tipo *Riccati* associada ao problema do regulador linear quadrático discreto que tem por base uma formulação que combina estimativas da função valor por meio de uma estrutura RLS (do inglês *Recursive Least-Squares*), diferenças temporais e melhorias de política. O desenvolvimento das metodologias propostas, neste trabalho, tem seu foco principal voltado para a fatoração UDU^T que é inserida neste quadro para melhorar o processo de estimação RLS de políticas de decisão ótimas do regulador linear quadrático discreto, contornando-se problemas de convergência e estabilidade numérica relacionados com o mal condicionamento da matriz de covariância da abordagem RLS.

Palavras-Chave: Programação Dinâmica, Aprendizagem por Reforço, Programação Dinâmica Heurística, Controle Multivariável, Controle Ótimo, Regulador Linear Quadrático Discreto, Mínimos Quadrados Recursivos.

Abstract

In this thesis a proposal of an unified approach of dynamic programming, reinforcement learning and function approximation theories aiming at the development of methods and algorithms for design of optimal control systems is presented. This approach is presented in the approximate dynamic programming context that allows approximating the optimal feedback solution as to reduce the computational complexity associated to the conventional dynamic programming methods for optimal control of multivariable systems. Specifically, in the state and action dependent heuristic dynamic programming framework, this proposal is oriented for the development of online approximated solutions, numerically stable, of the Riccati-type Hamilton-Jacobi-Bellman equation associated to the discrete linear quadratic regulator problem which is based on a formulation that combines value function estimates by means of a RLS (Recursive Least-Squares) structure, temporal differences and policy improvements. The development of the proposed methodologies, in this work, is focused mainly on the UDU^T factorization that is inserted in this framework to improve the RLS estimation process of optimal decision policies of the discrete linear quadratic regulator, by circumventing convergence and numerical stability problems related to the covariance matrix ill-conditioning of the RLS approach.

Keyword: Dynamic Programming, Reinforcement Learning, Heuristic Dynamic Programming, Multivariable Control, Optimal Control, Discrete Linear Quadratic Regulator, Recursive Least-Squares.

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas e Siglas

AC	*Adaptive Critic* (Crítico Adaptativo)
ACDs	*Adaptive Critic Designs* (Projetos Críticos Adaptativos)
AD	*Action Dependent* (Dependente de Ação)
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*
IP	Iteração de Política
LQ	*Linear Quadratic*(Linear Quadrático)
LQG/LTR	*Linear Quadratic Gaussian/Loop Transfer Recovery* (Gaussiano Linear Quadrático/ Recuperação da Malha de Transferência)
LQR	*Linear Quadratic Regulator* (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

Capítulo 1

Introdução

A complexidade dos sistemas dinâmicos, devido aos requisitos de desempenho do projeto, exige controladores de malha fechada de alto desempenho para executar as tarefas programadas, tais como aeronaves e robôs de alta velocidade e precisão que sejam capazes de rejeitarem as perturbações externas e incertezas. Uma alternativa para atender a esses requisitos de projeto é dada na teoria de controle ótimo que fornece os meios para desenvolver soluções para sistemas dinâmicos que demandam controladores mais complexos.

Métodos clássicos de controle ótimo têm alcançado grande sucesso para sistemas lineares via equações do tipo *Riccati*, mas este sucesso é restrito à algumas aplicações *online*. Em uma maneira geral os métodos são *offline* e requerem o modelo do sistema dinâmico. Diversos conceitos matemáticos e métodos de otimização dinâmica têm sido agregados para desenvolver sistemas de controle ótimo, os quais destacam-se o princípio do mínimo de *Pontryagin* e a equação de *Euler-Lagrange*, (Bryson e Ho, 1975) e (Athans e Falb, 1966), e os métodos de *Bellman* da Programação Dinâmica com a subsequente equação HJB (*Hamilton-Jacobi-Bellman*), (Bellman, 1957), (Bellman, 1958), (Bellman, 2003), (Bertsekas, 1995*a*), (Bertsekas, 1987), (Howard, 1960).

Sabe-se que as formulações de *Bellman* fornecem a solução mais abrangente para controle ótimo que variam desde sistemas lineares baseados em funções de critério quadráticas para os sistemas não-lineares mais complexos, contudo os recursos computacionais associados com a realização dessas formulações são de alto custo para implementação *online*, (Lendaris, 2009), especialmente para sistemas

de ordem elevada. Uma alternativa para contornar esta complexidade computacional são os métodos de aproximação de programação dinâmica que empregam um tipo de crítico adaptativo da aprendizagem por reforço (*Reinforcement Learning* - RL). Esta abordagem também tem sido denominada programação dinâmica aproximada/adaptativa (*Approximate/Adaptive Dynamic Programming* - ADP).

Um esquema típico de crítico adaptativo é ilustrado na Figura 1.1, representando o sistema de controle com realimentação dentro de um contexto relacionado à controle adaptativo que é realizado pelos componentes ator e crítico. O ator estabelece uma lei de controle (política de controle) parametrizada, enquanto que o crítico fornece uma representação parametrizada da função valor. Esta função avalia o efeito do controle sobre seu desempenho futuro e fornece diretrizes sobre como melhorar a lei de controle. A principal característica do *crítico adaptativo* que permite a aplicação *online* é que ele efetivamente resolve as equações de otimização HJB em uma maneira "para frente no tempo", enquanto a programação dinâmica tradicional requer um procedimento "para trás no tempo", (Werbos, 1990*b*).

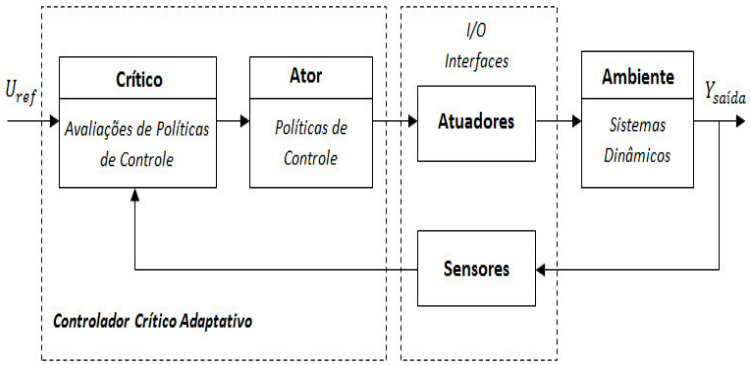

Figura 1.1: Estrutura de Aprendizado por Reforço com Ator/Crítico.

As variações de métodos críticos adaptativos mais comuns na literatura são os métodos de diferenças temporais, (Sutton, 1988), e aprendizagem Q, (Watkins, 1989), (Watkins e Dayan, 1992). Estes, por sua vez, têm sido classificados por Werbos (Werbos, 1992), (Werbos, 1990*b*) em programação dinâmica heurística (*Heuristic Dynamic Programming* - HDP) e programação heurística dual (*Dual Heuristic Programming* - DHP). Existem versões modificadas chamadas críticas dependentes de ação de cada uma delas, nominalmente Programação Dinâmica Heurística Dependente de Ação (*Action Dependent Heuristic Dynamic Program-*

ming - ADHDP) e Programação Heurística Dual Dependente de Ação (*Action Dependent Dual Heuristic Programming* - ADDHP). Algumas vantagens do desenvolvimento de métodos baseados em programação dinâmica aproximada é que alguns dos críticos adaptativos não necessitam da identificação explícita do sistema, e os sistemas estocásticos podem ser inseridos neste quadro.

Prokhorov e Wunsch (Prokhorov e Wunsch, 1997) forneceram uma discussão detalhada das estruturas críticas adaptativas de Werbos e sugeriram dois novos métodos, que são programação heurística dual global (*Global Dual Heuristic Programming* - GDHP) e programação heurística dual global dependente de ação (*Action Dependent Global Dual Heuristic Programming* - ADGDHP). As aproximações são classificadas de acordo com as abordagens estabelecidas para estimar a função valor $V(x)$. 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. Na GDHP, o crítico aproxima tanto a função valor quanto seu gradiente.

Resultados teóricos de estabilidade e convergência foram obtidos para vários projetos críticos adaptativos para solução de problemas de controle ótimo. Veja, por exemplo, o trabalho de Bradtke e outros (Bradtke *et al.,* 1994) em que aprendizagem Q é demonstrada convergir ao usar uma aproximação linear da função valor. Especificamente, esta abordagem foi baseada sobre uma importante condição de persistência de excitação dos métodos RLS. Mais trabalho foi realizado por Landelius e Knutsson (Landelius e Knutsson, 1996) que estudaram as quatro arquiteturas críticas adaptativas de Werbos. Eles demonstraram resultados de convergência para todos os quatro casos para o problema do regulador linear quadrático utilizando-se métodos do gradiente. Al-Tamimi e outros (Al-Tamimi *et al.,* 2007*a*) apresentaram um esquema crítico adaptativo baseado em aprendizagem Q para o jogo de soma zero linear quadrático de tempo discreto relacionado à controle ótimo H^{∞}. Mais recentemente, Vamvoudakis e Lewis (Vamvoudakis e Lewis, 2010) sugeriram um algoritmo baseado em iteração de política que aprende *online* soluções aproximadas da equação HJB para problemas de controle ótimo com custo de horizonte infinito para sistemas não-lineares de tempo contínuo e

com dinâmicas conhecidas.

Programação dinâmica aproximada tem expandido bastante o horizonte do campo de controle ótimo para aplicações *online* e tem superado a necessidade por um modelo dinâmico completo exigido nos métodos clássicos de controle ótimo, viabilizando aprendizagem em ambientes desconhecidos e não-lineares. Por exemplo, Jiang e outros (Jiang e Jiang, 2011) fornecem um quadro de programação dinâmica aproximada robusta que garante, sob certas condições, robustez à incertezas dinâmicas via aprendizagem *online*. Em (Liu *et al.,* 2011), os autores propõem um esquema de controle ótimo inteligente para sistemas não-lineares de tempo discreto com dinâmicas desconhecidas, e os autores (Dierks e Jagannathan, 2011) fornecem uma classe de aproximadores *online* para resolver problemas de controle ótimo de regulação e de rastreamento de horizonte infinito via ADP para sistemas não-lineares de tempo discreto na presença de dinâmica interna desconhecida.

Este capítulo introdutório está organizado como segue. Nas Seções 1.1 e 1.2 abordam-se os objetivos desta tese e a principal motivação que conduziu o desenvolvimento desta pesquisa. Segue-se então, com a Seção 1.3 onde se discute uma abordagem moderna de aprendizagem por reforço alicerçada em teorias de programação dinâmica e aproximação de função no contexto de controle ótimo. Na Seção 1.4 mostra-se a importância de métodos de programação dinâmica aproximada/adaptativa para o projeto *online* de sistemas de controle ótimo. Na Seção 1.5, apresentam-se perspectivas relativas ao desenvolvimento teórico e aspectos práticos de ADP. Na Seção 1.6 são apresentadas as principais contribuições fornecidas por esta pesquisa e, finalmente, na Seção 1.7, descreve-se a organização da tese.

1.1 Objetivos

O objetivo geral desta tese é propor métodos alternativos para estabelecer políticas de controle ótimo baseadas na forma HJB da equação algébrica de *Riccati* discreta (HJB-*Riccati*) para realizações *online* de projeto de sistemas de controle ótimo.

Os desenvolvimentos apresentados são orientados por meio de esquemas críti-

cos adaptativos em que a função valor é modelada em termos de recompensas em longo prazo. Especificamente, a programação dinâmica heurística e sua estratégia dependente de ação são formuladas no contexto de controle ótimo e aprendizagem por reforço para aproximar a solução da equação HJB via RLS e diferenças temporais por métodos numericamente estáveis, quando submetidos a uma iteração de política.

O objetivo principal é apresentar uma proposta para resolver, via fatoração UDU^T, os problemas de convergência e estabilidade numérica que estão relacionados com o mal condicionamento da matriz de covariância da abordagem RLS para aproximações *online* da solução da equação HJB do tipo *Riccati* do problema DLQR. Propor uma metodologia RL ator-crítico que combina métodos RLS, aprendizagem TD(λ) e melhorias de políticas, tendo em vista a formulação de estratégias para acelerar iteração de política aproximada para solução *online* de controle ótimo.

1.2 Motivação

Com o crescente desenvolvimento tecnológico e industrial tem-se exigido sistemas de controle com especificações de projeto muito mais rigorosas e que sejam capazes de rejeitarem as perturbações e incertezas, como por exemplo: aeronaves de alto desempenho e robôs de alta velocidade e precisão. Juntamente com a demanda por sistemas de controle de alto desempenho, sistema de controle MIMO (do inglês *Multiple-Input and Multiple-Output*) muitas vezes tornam-se preferidos devido à disponibilidade de sensores e atuadores mais baratos e confiáveis. Sistemas de controle não-lineares, por sua vez, cobrem a maior parte dos sistemas do mundo real. O desafio para projeto de controle é combinar todos esses requisitos e informações para alcançar o melhor desempenho possível de sistema de controle *online*.

Nos últimos anos, os métodos de controle ótimo baseados em programação dinâmica aproximada e aprendizagem por reforço tem se destacado como uma ferramenta muito útil para melhorar o desempenho de sistemas do mundo real e contribuido para viabilizar as implementações em tempo real de controladores ótimos. Programação dinâmica aproximada e aprendizagem por reforço pode ser

vista em (Werbos, 2012) na perspectiva de um campo de pesquisa promissor para desenvolver controladores inteligentes inspirados em estruturas neurais biológicas que fornecem capacidades para lidar de forma eficaz com o grau de complexidade de sistemas não-lineares, incertos e parcialmente observáveis. Em (Fu *et al,* 2011) e (Buşoniu *et al.,* 2010*a*), os autores apresentam uma visão do projeto de controladores inteligentes para sistemas MIMO que tanto aprendem quanto exibem comportamento ótimo empregando programação dinâmica adaptativa e aprendizagem por reforço.

O campo de programação dinâmica aproximada tem passado por uma enorme expansão e tem se destacado em uma variedade de setores tecnológicos, por exemplo, teoria de controle, inteligência artificial, redes neurais e lógica fuzzy. Nos setores industriais, o foco tem sido sobre aeronaves de alto desempenho ou sistemas de controle de mísseis, (Bertsekas *et al.,* 2000), autopiloto, (Ferrari e Stengel, 2004), (Lin, 2005), geradores (Park *et al.,* 2003), sistemas elétricos, (Mohagheghi *et al.,* 2007), (Qiao *et al.,* 2009), sistemas de comunicação, (Liu *et al.,* 2005), processos bioquímicos, (Iyer e Wunsch, 2001), etc. Em (Enns e Si, 2003), os autores desenvolveram um controle de rastreamento de helicóptero usando programação dinâmica neural, e em (Zhao e Hu, 2011) os autores apresentam um algoritmo de programação dinâmica adaptativa supervisionada para um sistema de controle adaptativo de cruzeiro.

1.3 Aprendizagem por Reforço Moderna

A aprendizagem por reforço tem sido reconhecida por fisiologistas desde a época de Ivan Pavlov (Pavlov, 1927) que utilizou estímulos simples de recompensa e punição para modificar padrões do comportamento de cães por indução de reflexos condicionados. Em sua forma clássica, a aprendizagem por reforço está estreitamente relacionada à aprendizagem a partir da experiência combinada com o princípio de recompensa e punição emulada de seres vivos (humanos e animais) tendo por objetivo alcançar um comportamento altamente qualificado ao longo do tempo. A ideia de aprendizagem por reforço é também incorporada na obra clássica de Charles Darwin: "A origem das Espécies", tendo por base adaptações estabelecidas por organismos vivos por meio de interações com o ambiente no qual

eles estão inseridos, levando a seleção natural e sobrevivência do mais forte.

A aprendizagem por reforço está bem enraizada em Ciência da Computação e Inteligência Artificial, e tem sido um foco principal para pesquisadores no campo de redes neurais (*Neural Networks* - NN), especialmente nas áreas de reconhecimento de padrão, regressão não-linear e identificação de sistema não-linear. Segundo Sutton e Barto (Sutton e Barto, 1998), a aprendizagem por reforço é um formalismo da inteligência artificial que permite aos sistemas aprenderem a tomar boas decisões observando o seu próprio comportamento, e usarem estratégias embutidas para melhorar suas ações através de um mecanismo por reforço. Um quadro detalhado de literatura de pesquisa de aprendizagem por reforço tem sido autorado por (Kael *et al.,* 1996). O trabalho deles está principalmente focado sobre os esquemas RL em ciências da computação. Além disso, (Sutton, 1999) e também (Thrun e Littman, 2000) apresentaram excelentes panoramas de RL.

Na abordagem moderna, a aprendizagem por reforço está intimamente relacionada com a programação dinâmica (*Dynamic Programming* - DP), que foi desenvolvida por *Bellman* (Bellman, 1957) no contexto de controle ótimo. A programação dinâmica fornece o formalismo matemático para a tomada de decisão multiestágio que envolve o compromisso entre um baixo custo no presente e altos custos indesejáveis no futuro, levando à um planejamento ótimo. O objetivo é estabelecer políticas de tomada de decisão que minimizem o custo total sobre um número de estágios. Tais problemas são primariamente 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, 1998).

Seguindo a terminologia de Bertsekas e Tsitsiklis (Bertsekas e Tsitsiklis, 1996), a abordagem moderna de aprendizagem por reforço é referida como programação neurodinâmica (*Neuro-Dynamic Programming* - NDP), tendo em vista a dependência dos conceitos tanto de DP quanto de redes neurais, enquanto a DP fornece a fundamentação teórica, a NN fornece a capacidade de aprendizagem. Como discutido pelos autores, métodos NDP podem ser determinantes para solução de problemas de tomada de decisão sequencial complexos e de larga escala onde métodos tradicionais de DP seriam dificilmente aplicáveis e outros métodos heurísticos teriam potencial limitado. Em (Bertsekas e Tsitsiklis, 1996), Bertsekas e Tsitsiklis

fornecem um tratamento completo de programação neurodinâmica, incluindo teoremas de convergência geral para métodos de aproximação estocástica como o alicerce para análise de vários algoritmos NDP. Alí, os autores sugerem muitas metodologias úteis para aplicações, como simulação de *Monte Carlo*, métodos de diferença temporal, algoritmo de aprendizagem Q, métodos de iteração de política otimística, métodos de erros de *Bellman*, programação linear aproximada, programação dinâmica aproximada com "função de custo para avançar", etc.

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), (Werbos, 1990*b*). O conceito de crítico adaptativo é essencialmente uma combinação de idéias da aprendizagem por reforço e programação dinâmica. Contribuições anteriores importantes à aprendizagem por reforço moderna incluem os trabalhos de Samuel (Samuel, 1959) sobre o seu célebre programa de jogo de damas, de Barto e outros (Barto *et al.,* 1983) sobre sistemas críticos adaptativos, de Sutton (Sutton, 1988) sobre métodos de diferença temporal e de Watkins (Watkins, 1989) sobre a aprendizagem Q. O manual de controle inteligente de White e Sofge (White e Sofge, 1992) apresenta material sobre controle ótimo por White e Jordan, sobre aprendizagem por reforço e métodos críticos adaptativos por Barto e sobre programação dinâmica heurística por Werbos.

Todas as abordagens de aprendizagem por reforço compartilham o mesmo objetivo que é aprender otimalidade ao longo do tempo. De acordo com (Williams, 2009), a aprendizagem por reforço moderna é uma combinação de métodos de diferença temporal, controle ótimo e teorias de aprendizagem de estudos de animais. Como explicado em (Gosavi, 2009), a aprendizagem por reforço moderna surgiu de uma síntese das ideias de quatro campos diferentes: a) programação dinâmica clássica, b) inteligência artificial (diferenças temporais), c) aproximação estocástica, e d) aproximação de função (regressão, erro de *Bellman* e redes neurais).

Em (Si *et al.,* 2004), são discutidas as relações de ADP com a inteligência artificial, teoria de aproximação, teoria de controle e pesquisas operacionais. Em (Powell, 2007), mostra-se como ADP acoplada com programação matemática pode resolver (aproximadamente) problemas de otimização determinística e estocás-

tica e apresentam-se as direções melhoradas de ADP. Literatura mais recente de Bertsekas (Bertsekas, 2012) apresenta uma discussão aprofundada dos esquemas de ADP.

1.4 Controle Ótimo *Online*

Sabe-se que controle ótimo e controle adaptativo representam diferentes filosofias para o projeto de controladores. Controladores ótimos são comumente projetados para minimizarem funções de desempenho prescritas pelo usuário com respeito aos critérios definidos para os objetivos de projeto. Não obstante, no final, após o controlador ser projetado, é colocado em serviço sem mecanismos associados para modificar seu projeto em resposta às mudanças no contexto (planta, ambiente, medidas de desempenho associadas). Os projetos de controle são realizados *offline* e, de fato, tendo por base um modelo detalhado da dinâmica da planta. Entretanto, na prática, muitas vezes é importante projetar-se controladores *online*, em tempo real, sem ter conhecimento completo da dinâmica da planta. Incertezas de modelagem podem existir, incluindo os parâmetros imprecisos, dinâmicas não-modeladas de alta freqüência, e perturbações. Além disso, tanto a dinâmica do sistema quanto os objetivos de desempenho podem mudar com o tempo.

Controladores adaptativos aprendem *online* para controlar sistemas com dinâmicas desconhecidas usando dados em tempo real medidos ao longo das trajetórias do sistema, no entanto, eles não são geralmente projetados com a qualidade de serem ótimos no sentido de otimização matemática (em relação a critérios especificados pelo usuário). Segundo Werbos (Werbos, 2012), o controle adaptativo toma uma abordagem similar a de uma estrutura neural a qual se baseia inteiramente em aprendizagem em tempo real, em algum nível. Em cada instante de tempo k, observa-se a saída do sistema dinâmico, decide-se sobre a tomada de ação $u(k)$ e, então, adaptam-se as redes em nosso cérebro e, finalmente, avança-se para o instante de tempo seguinte $k+1$. Para tentar emular a inteligência do cérebro do "animal"é necessário desenvolver sistemas ADP que podem operar com sucesso neste modo.

Vários pesquisadores têm fornecido uma visão unificada de controle ótimo e

controle adaptativo baseada em paradigmas RL e ADP. De acordo com Werbos (Werbos, 2012), ADP tanto pode ser vista como uma extensão de controle adaptativo que, devido ao *lookahead* implícito, alcança estabilidade sob condições muito mais fracas que as formas bem conhecidas de controle adaptativo direto e indireto, como também ela pode ser formulada como uma parte de controle ótimo que busca métodos gerais computacionalmente viáveis para o caso não-linear estocástico. O trabalho por Sutton e outros (Sutton *et al.,* 1992) discute aprendizagem por reforço (aprendizagem-Q) como uma abordagem de controle ótimo adaptativo direto. Barto, Bradtke e Singh (Barto *et al.,* 1995) discutiram esquemas de aprendizagem baseada em programação dinâmica tendo em vista trazer tais técnicas mais próximas de aprendizagem/planejamento em tempo real e teoria de controle (principalmente controle ótimo e controle adaptativo). Entretanto, a estrutura ator-crítico adaptativa fundamental pode ser encontrada no trabalho de Barto e Sutton (Barto *et al.,* 1983). Detalhe significante sobre a teoria de controle adaptativo ótimo via ADP pode ser encontrado no livro de Bertsekas (Bertsekas, 2012).

A maior parte do trabalho no que diz respeito a controle ótimo adaptativo está consagrado em ADP que usa métodos ator-crítico adaptativos. Por exemplo, o trabalho de He e Jagannathan (He e Jagannathan, 2007) apresenta um esquema adaptativo ótimo na presença de restrições de entrada. Estruturas ator-crítico adaptativas são os métodos mais populares e provavelmente mais eficientes até o momento para produzir controle adaptativo ótimo baseado em aprendizagem por reforço. Outras referências úteis são (Lewis e Vrabie, 2009b) e (Lewis e Vrabie, 2009a). Lee e outros (Lee *et al.,* 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. O esquema proposto deles pode resolver *online* o problema de controle linear quadrático na presença tanto de incertezas internas quanto de sinais de excitação conhecidos. O trabalho mais recente de Lewis e Vamvoudakis (Lewis e Vamvoudakis, 2011) apresenta um esquema de controle adaptativo ótimo baseado em aprendizagem por reforço (ADP) para um sistema desconhecido usando realimentação de saída medida. O esquema é baseado em uma estrutura ator-crítico adaptativa usando duas redes neurais, uma rede neural para o crítico

estimar a função valor de ação e uma outra para o ator estimar a política de controle ótima; o esquema é similar em natureza ao trabalho por (Stingu e Lewis, 2010).

Uma visão geral das técnicas RL e seus avanços focados em controle ótimo adaptativo é apresentado em (Khan *et al.,* 2012), onde os autores destacam aplicações de RL em robótica, particularmente em relação ao campo de controle. Entretanto, o escopo de RL engloba problemas de uma variedade de outros campos, incluindo, por exemplo, ciência da computação, inteligência artificial, pesquisa operacional e economia. Lendaris (Lendaris, 2009) comenta os desenvolvimentos e benefícios de métodos ADP para o campo de controle ótimo adaptativo e discute algumas tendências de pesquisa em ADP para esta década. Balakrishnan, Ding e Lewis (Balakrishnan *et al.,* 2008) fornecem uma boa discussão sobre assuntos de estabilidade para controladores de realimentação baseados em redes neurais e técnicas ADP (críticos adaptativos). Eles forneceram esquemas ADP baseados em modelo bem como abordagens ADP livres de modelo. Um trabalho recente por Busoniu (Buşoniu *et al.,* 2011) fornece um estudo de esquemas de aprendizagem por reforço e programação dinâmica bem como assuntos de implementação relacionados.

1.5 Perspectivas sobre Programação Dinâmica Aproximada

Esta seção descreve perspectivas no que tange ao desenvolvimento teórico de Programação Dinâmica Aproximada (ou Adaptativa) e aborda aspectos práticos de empregar métodos ADPs para realização de projetos de controle, tomando-se as concepções de vários pesquisadores no campo de ADP e Controle Ótimo. Discute-se, inicialmente, assuntos relativos à aproximação da função valor, e comenta-se sobre "paradigmas dependentes de modelo"e "paradigmas independentes de modelo"para o projeto de sistemas de controle ótimo. Discutem-se, em seguida, as compensações entre projeto de controlador *offline* e aprendizagem *online* e abordam-se, por fim, problemas sobre observabilidade parcial.

1.5.1 Aproximações de Funções Valor

O problema de se aproximar uma função valor em programação dinâmica é uma tendência de pesquisa com uma longa história. Antes da época de ADP moderna[1], muitos pesquisadores tentariam representar a função valor ótima $V^*(x)$ usando-se tabelas de consulta (representações tabulares) ou árvores de decisão. Não há dúvida de que as representações tabulares fornecem as soluções mais exatas possíveis para se calcular a política de controle ótima. No entanto, isto levou a famosa "maldição da dimensionalidade". Por exemplo, se o estado x consiste de 10 variáveis contínuas, e se são considerados apenas 20 níveis possíveis para cada uma destas varíaveis, a representação tabular conteria 20^{10} números. Isso seria uma enorme tarefa computacional irrealista para preencher essa tabela de consulta, mas 20 níveis possíveis ainda pode ser muito rude para um desempenho útil.

Para um sistema ADP de propósito geral, desejaría-se poder aproximar a função valor V^* ou seu gradiente ∇V^* com um aproximador universal não-linear de função. Muitos destes tais aproximadores foram provados existirem, para uma grande classe de possíveis funções. Por exemplo, série de Taylor multivariável foi vista muitas vezes como os mais consideráveis dos aproximadores universais, devido a sua longa história. Entretanto, Andrew Barron (Barron, 1993) demonstrou que tais aproximadores lineares de funções de base apresentam o mesmo problema básico que as tabelas de consulta: eles requerem um crescimento exponencial em complexidade (número de pesos) à medida que o número de varíaveis de estado cresce, para uma dada qualidade de aproximação de uma função suave. Além disso, devido aos muitos termos polinomiais correlacionarem fortemente um com os outros, aproximações por séries de Taylor normalmente têm sérios problemas com condicionamento numérico (Sauer, 2011).

Para muitas aplicações, portanto, a melhor escolha para um Crítico é o Perceptron Multicamadas (Multilayer Perceptron - MLP), a locomotiva tradicional, simples de engenharia de redes neurais. Barron provou (Barron, 1993), (Barron,

[1] A ADP moderna retrocede, no mínimo, ao ano de 1968 (Werbos, 1968) quando Werbos propôs a construção de sistemas de aprendizagem por reforço por meio de aproximações adaptativas para a equação HJB. Antes de 1968, pesquisa em RL e pesquisa relacionada à programação dinâmica eram duas áreas independentes.

1994) que o nível exigido de complexidade de uma MLP cresce somente como uma baixa potência do número de variáveis de estado na aproximação de uma função suave. Isto funcionou muito bem em aplicações não-lineares difíceis que requerem, digamos, 10-20 variáveis. Pode-se usar aproximações mais simples como Funções Gaussianas ou Funções de Base Radial ou CMAC (do inglês *Cerebellar Model Articulation Controller*) diferenciável (White e Sofge, 1992) ou métodos de *Kernel* ou teoria de ressonância magnética em casos onde existem poucas variáveis de estado ou onde o sistema parece ser restrito na prática a alguns agrupamentos dentro do espaço de estado maior. Existe uma razão para suspeitar que lógica fuzzy elástica (Werbos, 1995) pode oferecer desempenho similar às MLPs na aproximação de função, uma vez que ela parece uma versão exponenciada da MLP, mas isto tem ainda que ser comprovado ou não.

Em 1996, Bertsekas e Tsitsiklis (Bertsekas e Tsitsiklis, 1996) propuseram uma outra maneira de aproximar a função valor, usando funções de base φ_i especificadas pelo usuário:

$$\widehat{V}(x,\theta) = \sum_{i=1}^{n_\theta} \varphi_i(x)\theta_i.$$

Esta aproximação é ainda governada, em princípio, pelos resultados de Barron sobre aproximadores de função de base lineares, mas se os usuários fornecerem funções de base apropriadas para seu problema particular, isto pode permitir melhor desempenho na prática. Isto é análogo a aqueles pacotes de software estatísticos que permitem aos usuários estimarem modelos que são "lineares nos parâmetros mas não-lineares nas variáveis". Isso é também análogo ao uso de "características"definidas pelo usuário, tipo características HOG (do inglês *Histogram of Oriented Gradients*) ou SIFT (do inglês *Scale Invariant Feature Transform*), em processamento de imagem tradicional. Isso também abre as portas para muitos casos especiais de métodos ADP de propósito geral, e novos métodos de próposito especial para o caso linear, tais como o uso de programação linear para estimar o peso θ (De Farias e Roy, 2003).

O senso comum e teoremas de Barron sugerem que desempenho bem melhor poderia ser alcançado se as funções de base φ_i pudessem ser aprendidas, ou adaptativamente melhoradas, ao invés de serem mantidas fixadas, mas isso traz de volta, em princípio, ao problema de como adaptar um crítico não-linear geral

$\widehat{V}(x,\theta)$ ou $\nabla\widehat{V}(x,\theta)$, que os métodos existentes já abordam. Em processamento de imagem, "aprendizagem aprofundada (*deep learning*)"de redes neurais já levou à grandes avanços, desbancando métodos antigos baseados em características desenvolvidos durante décadas de intenso trabalho de domínio específico, e tem algumas vezes reduzido o erro por um fator de dois ou mais comparados com métodos tradicionais (Kavukcuoglu *et al.,* 2010), (Lecun *et al.,* 2010). Esperaría-se que aprendizagem aprofundada produzisse benefícios similares, especialmente se houvesse mais pesquisa nesta área.

1.5.2 Projeto de Controlador Ótimo Independente de Modelo

Nesta subseção, os paradigmas dependentes de modelo e independentes de modelo, bem como suas versões híbridas são enfatizados no contexto de projeto de sistemas de controle.

Sabe-se que na abordagem DHP, algum tipo de modelo de sistema dinâmico é necessário para treinar a rede crítica e a rede de ação. Mesmo com a HDP, um modelo da função de transição de estado do sistema é requerido para encontrar as ações de controle ou treinar a rede de ação. Por outro lado, a ADHDP (aprendizagem Q) não necessita de tal modelo. Sob quais condições seria vantajoso usar-se um paradigma independente de modelo ao invés de um paradigma dependente de modelo?

Bradtke (Bradtke, 1994) apresenta dois argumentos principais para adotar uma abordagem independente de modelo. Primeiro, tal abordagem pode ser mais barata para encontrar o controlador ótimo (ou pelo menos um aceitável) através da interação direta com o sistema do que por meio da construção de um modelo, e então, obter-se um controlador a partir desse modelo. Este argumento tem tido por base estudos experimentais realizados por Gullapalli (Gullapalli, 1992). Um modelo da função de transição de estados pode ser relativamente fácil de ser construido para alguns sistemas dinâmicos, tais como sistemas lineares. Entretanto, um modelo preciso pode ser muito difícil de ser obtido para outros sistemas, tais como controle de vôo. Além disso, mesmo que um modelo preciso do sistema seja conhecido, muitas vezes será o caso de que a obtenção de um controlador ótimo a partir desse modelo seja analiticamente e computacionalmente intratável.

Considere, por exemplo, o jogo de Gamão (Tesauro, 1989), (Tesauro, 1994). A função de transição de estado para o gamão é especificada pelas regras do jogo. Entretanto, estima-se que o gamão tenha pelo menos 10^{20} estados. Tentar encontrar a política ótima para o jogo de gamão usando-se uma abordagem baseada em modelo é um problema intratável.

Werbos (Werbos, 2012) discute o que parece mais plausível quanto a uma escolha entre HDP e ADDHP. Segundo ele, a melhor abordagem, em princípio, seria combinar os dois tipos de paradigmas, tendo em vista que em ADHDP, escolhe-se ações de controle $u(k)$ que são similares à ações que têm funcionado bem no passado, enquanto que em HDP escolhe-se ações que devem levar a melhores resultados no passo de tempo seguinte $k + 1$. Isto exige um tipo de modelo de transição de estado, mas também requer alguma forma de ser robusto com respeito às incertezas naquele modelo.

Ainda de acordo com Werbos, (Werbos, 2012), a melhor informação que se tem até agora sugere que o desempenho de DHP não é tão dependente na prática aos detalhes do modelo de transição de estados, e que DHP desenvolve cada vez mais vantagem em relação à ADHDP à medida que o número de variáveis de estado cresce. Contudo, mais pesquisa seria útil em fornecer mais informação sistemática e analítica sobre essas compensações. Essa será uma importante área para pesquisa, especialmente quando as ferramentas próprias para DHP e HDP tornarem-se mais amplamente disponíveis e mais facéis de serem usadas.

1.5.3 Aprendizagem *Online* de Controladores Ótimos

Nesta subseção, discutem-se as compensações entre aprendizagem *online* e aprendizagem *offline* de controladores ótimos. Uma discussão acerca destas compensações é apresentada em (Werbos, 2012), onde o autor destaca o trabalho realizado por Miller (Miller *et al.,* 1990) que mostrou como se poderia treinar uma rede neural por controle adaptativo para empurrar um carrinho instável para adiante em torno de uma pista no formato do número 8 (oito). Quando ele duplicou o peso sobre o carrinho, a rede aprendeu a equilibrar e recuperar depois de apenas duas voltas ao redor da pista. Isso pareceu impressionante, mas, conforme citado pelo autor, esse tipo de tarefa poderia ser realizada bem melhor treinando-se uma rede recorrente *offline*. Dado um banco de dados do comportamento do carrinho

em uma ampla faixa de variações nos pesos, a rede recorrente pode aprender a detectar e responder imediatamente à variações no peso, mesmo que não se observe o peso do carrinho diretamente. Este truque, de "aprender *offline* para ser adaptativa *online*", sublinha aos grandes sucessos da Ford em muitas aplicações. Presumivelmente, o próprio cérebro inclui uma combinação de neurônios recorrentes para fornecer este tipo de resposta adaptativa rápida (como a "memória em funcionamento") estudada por Goldmann-Rakic e outros neurocientistas, juntamente com aprendizagem em tempo real para lidar com mais novos tipos de mudanças no mundo. Ele também inclui alguma habilidade de aprender a reviver memórias do passado (Werbos, 2011), que também poderiam ser usadas para melhorar os sistemas de predição usados em engenharia e ciências sociais.

1.5.4 Observabilidade Parcial

É importante abordar processos dinâmicos nos quais nem todos os estados podem ser observados diretamente da planta. Na literatura ADP, tais processos têm sido referidos como *parcialmente observáveis*, e eles surgem na maioria das aplicações em sistemas de controle no mundo real quando um agente sofre de capacidades sensoriais limitadas que o impede de recuperar um sinal de estado a partir de suas percepções.

Em controle ótimo linear quadrático para processos parcialmente observáveis, métodos de controle dual no qual o estado x é estimado por um filtro de *Kalman* tem desempenhado um papel importante na prática. No caso do LQR, a inserção do filtro de Kalman tem sido bastante útil, mas isto tem ocasionado perdas nas propriedades de margens de fase e ganho garantidas pelo LQR. Entretanto, tais propriedades de robustez tem sido frequentemente recuperadas por meio da metodologia LTR (*Loop Transfer Recovery*).

Métodos de ADP geralmente requerem, de alguma forma, a disponibilidade da informação completa dos estados internos do processo dinâmico a ser controlado. Processos parcialmente observáveis têm sido extensivamente estudados por pesquisadores interessados em construir agentes autônomos que aprendem por meio da interação com seu ambiente (Ng e Jordan, 2000), (Hauskrecht, 2000), (Baxter e Bartlett, 2001), (Porta *et al.,* 2006). O trabalho de Lewis e Vamvoudakis (Lewis e Vamvoudakis, 2011) apresenta desenvolvimentos em estruturas ator-crítico para

processos dinâmicos parcialmente observáveis baseados somente na realimentação da saída do sistema.

Werbos (Werbos, 2012) fornece uma discussão sobre o que acontece quando um paradigma independente de modelo, tal como ADHDP, é aplicado diretamente a um processo parcialmente observável. Se ações de controle u são escolhidas de modo a maximizar a função valor Q, o crítico simplesmente não teria a informação necessária para tomar as melhores decisões. Com efeito, a verdadeira função Q é uma função do estado x e da ação u. O procedimento óbvio é criar algum tipo de estimativa atualizada de x, digamos $\widehat{x}$. Werbos sugere algumas formas padrões para desenvolver uma estimativa de estado, a qual pode ser usada como a entrada principal para uma rede crítica ou uma rede de ação: filtro de *Kalman* estendido, filtro de partícula, treinamento de uma rede TLRN (*time-lagged recurrent network*) para predizer x a partir da saída Y em dados simulados, extração da saída dos nodos recorrentes de uma rede neural usados para modelar a planta. Por exemplo, o sucesso do mundo real de White e Sofge (White e Sofge, 1992) ao usar ADHDP dependeu do fato que eles usaram filtragem de *Kalman* estendida para criar esse tipo de estimativa. Eles usaram ADHDP para treinar um crítico o qual aproximou Q como uma função de $\widehat{x}$ e u.

1.6 Contribuições

As principais contribuições propostas nesta tese são explicitadas nos Capítulos 5 e 6. Três principais contribuições podem ser identificadas:

- A caracterização e solução do problema de estimacão paramétrica RLS formulado no quadro de programação dinâmica aproximada e aprendizagem por reforço para obter soluções aproximadas *online*, numericamente estáveis, da equação HJB-*Riccati* associada ao problema de controle ótimo DLQR. O fenômeno da instabilidade numérica é caracterizado aqui e se refere à uma classe de problemas onde as variáveis atualizadas na implementação computacional dos métodos RLS para solução *online* de controle ótimo DLQR perdem suas propriedades teóricas. Exemplos de tais propriedades são a simetria e a positividade da matriz de covariância da abordagem RLS. Devido

à problemas de mal condicionamento dessa matriz, a solução para uma política de decisão ótima para um dado ponto de operação pode não convergir.

- O desenvolvimento de algoritmos RL ator-crítico para a solução *online* de sistemas de controle ótimo de tempo discreto. Especificamente, os algoritmos *online* consistem em uma abordagem baseada em estimação paramétrica RLS, métodos de diferenças temporais e versões aproximadas de iteração de política que fazem uso da informação de estado de tempo discreto para resolver em uma maneira *online* a equação HJB-*Riccati* subjacente ao problema DLQR no contexto de programação dinâmica aproximada.

- Uma metodologia RL ator-crítico baseada na fusão de métodos RLS, aprendizagem TD(λ) e melhorias de políticas para solução *online* de controle DLQR, tendo em vista a formulação de estratégias de iteração de política para acelerar o processo de aprendizagem da política de controle ótima DLQR. As estratégias são avaliadas em termos do efeito do parâmetro λ do vetor de elegibilidade por meio de estatísticas de primeira e segunda ordem.

Pesquisas anteriores relacionadas à paradigmas ator-crítico adaptativos baseados em aproximadores RLS da função valor podem ser encontradas em (Khan *et al.,* 2012), (Al-Tamimi, 2007), (Al-Tamimi *et al.,* 2007*a*), (Bradtke e Barto, 1996) e (Bradtke *et al.,* 1994). Contudo, estabilidade numérica dos métodos RLS é um assunto que não tem sido discutido no contexto de aprendizagem por reforço e programação dinâmica aproximada para controle ótimo *online*. Sob esta óptica, este é o primeiro trabalho onde se propõe a formulação e solução de problemas de estabilidade numérica relacionados com o mal condicionamento da matriz de covariância da abordagem RLS para aproximações da função valor.

O método de solução proposto, neste trabalho, para resolver problemas de convergência e estabilidade numérica dos métodos RLS, via fatoração UDU^T, é visto como uma melhoria no processo de estimação de políticas de decisão ótima DLQR uma vez que a fatoração UDU^T contorna problemas relacionados à perda da simetria e positividade da matriz de covariância do RLS. A contribuição aqui é vista em dois estágios. No primeiro, avalia-se o comportamento do processo de estimação da solução da equação HJB-*Riccati* do problema DLQR por meio do número de condição e parâmetro de positividade da matriz de covariância do

RLS. Estes valores são usados em uma estratégia para avaliar o comportamento do processo iterativo da solução da equação HJB-*Riccati*. No segundo estágio, a fatoração UDU^T é inserida no processo de inversão da matriz de autocorrelação do método RLS.

Um desenvolvimento anterior sobre aprendizagem TD(λ) com aproximação da função valor usando a abordagem RLS é apresentado no trabalho de Xu e outros (Xu *et al.,* 2002). Eles propuseram e analisaram aprendizagem RLS-TD(λ) para solução de problemas de aprendizagem por reforço. Contudo, os resultados de convergência fornecidos por eles são válidos para avaliações de políticas de controle que são fixadas ao longo do tempo. Mais recentemente, Cheng e outros (Cheng *et al.,* 2012) sugeriram um novo algoritmo ator-crítico incremental baseado em diferenças temporais, aplicando-se aprendizagem RLS-TD(λ) para avaliação do crítico. No entanto, ainda não existem provas de convergência para paradigmas de iteração de política otimística baseados em aprendizagem RLS-TD(λ).

A abordagem proposta nesta tese foca-se sobre os princípios de iteração de política otimística estabelecida por Bertsekas e Tsitsiklis, (Bertsekas e Tsitsiklis, 1996), um dos assuntos mais promissores no campo de Programação Neurodinâmica. Aqui são fornecidas as primeiras ideias sobre métodos de iteração de política otimística baseados na fusão de aprendizagem TD(λ), métodos RLS e melhorias de políticas, e aplicações são realizadas sobre o problema de controle DLQR *online*. Esta proposta é direcionada para aproximações *online* de iteração de política para solução DLQR no sentido que as melhorias de política são realizadas em cada passo de tempo ao longo da realização de trajetória de estado em direção à política ótima DLQR.

1.7 Organização da Proposta de Tese

O trabalho está organizado em capítulos que abordam formulações de controle ótimo desde as abordagens de *Bellman* de Programação Dinâmica até formulações que empregam aproximações de programação dinâmica por meio de críticos adaptativos.

Duas abordagens críticas adaptativas são apresentadas, HDP e ADHDP. Ao longo do texto, as formulações de HDP e ADHDP para solução *online* de pro-

blemas de controle ótimo são desenvolvidas em domínio de tempo discreto. Tais abordagens discretas são a consideração básica para controle digital. Grande parte das pesquisas sobre metodologias ADPs existentes nas literaturas foram conduzidas por conceitos de aprendizagem por reforço formulados no quadro de processos de decisão markovianos (PDMs) para os sistemas que operam em tempo discreto.

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.

O problema de controle ótimo é apresentado no Capítulo 3 no contexto da formulação de *Bellman* de programação dinâmica, onde a ênfase é dada à problemas de controle ótimo do tipo linear quadrático. No Capítulo 4, o desenvolvimento de esquemas HDP e ADHDP é apresentado para determinar a política de controle ótima para o problema DLQR que está relacionado com a solução da equação HJB-*Riccati*. O problema da maldição da dimensionalidade é discutido no contexto de realizações *online*.

O capítulo 5 dedica-se ao foco principal do trabalho que diz respeito à tese. Apresenta-se o desenvolvimento de algoritmos de iteração de política aproximada, com suas variantes otimísticas, para controle DLQR *online*. Apresenta-se uma proposta para resolver via métodos de fatoração UDU^T os problemas de convergência e estabilidade numérica que estão relacionados com o mal condicionamento da matriz de covariância da abordagem RLS para aproximações *online* da solução da equação HJB-*Riccati* em esquemas HDP. Na sequência são discutidos métodos RLS-TD(λ) para problemas de controle ótimo *online*. Experimentos computacionais são apresentados no Capítulo 6 que verificam a melhoria dos algoritmos RLS$_\mu$-HDP-DLQR desenvolvidos ao usar fatoração UDU^T e avaliam o desempenho de métodos RLS-TD(λ) para parametrizações DLQR.

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.

1.8 Artigos Publicados

1.8.1 Artigos Publicados em Periódicos

1. Rêgo, P.H.M., Fonseca Neto, J.V. e Ferreira, E.F.M. (2013), "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*, pp.1-19, DOI: 10.1080/00207721.2013.844283.

2. Fonseca Neto, J.V. e Rêgo, P.H.M. (2014), "QR-Tunning and Approximate-LS Solutions of the HJB Equation for Online DLQR Design Via State and Action-Dependent Heuristic Dynamic Programming", *International Journal of Innovative Computing, Information and Control*, Vol.10, No.3, pp.1-26.

1.8.2 Artigos Publicados em Anais de Congressos

1. Queiroz, J.A., Rêgo, P.H.M., Fonseca Neto, J.V., Silva, Cristiane da, Santana, E. e Barros, A.K. (2013), "Estimators Based on Non-Squares Loss Functions to Approximate HJB-Riccati Equation Solution for DLQR Design via HDP", in *Systems, Man, and Cybernetics, 2013 IEEE International Conference on*, 13-16 October 2013, Manchester, pp.3238-3243, doi=10.1109/SMC.2013.552.

2. Fonseca Neto, J.V., Ferreira, E.F.M. e Rêgo, P.H.M. (2013), "Online Optimal DLQR-DFIG Control System Design via Recursive Least-Square Approach and State Heuristic Dynamic Programming for Approximate Solution of the HJB Equation". in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, 13-16 October 2013, Manchester, pp.3174-3179, doi=10.1109/SMC.2013.541.

3. Andrade, G., Fonseca Neto, J.V. e Rêgo, P.H.M. (2014), "RLS Estimator State Space Basis for Solution of HJB-Riccati Equation in Approximate Dynamic Programming", in *Advances in Control and Optimization of Dy-*

namical Systems, 13-15 March 2014, Kanpur, Índia, Vol.3, pp.1153-1160, doi=10.3182/20140313-3-IN-3024.00215.

4. Queiroz, J.A., Rêgo, P.H.M., Fonseca Neto, J.V., Silva, Cristiane da, Santana, E. e Barros, A.K. (2014), "Convergence Analysis Using Non-squares Estimators to Approximate the Solution of HJB-Riccati Equation for the Design DLQR via HDP", in *International Conference on Computer Modelling and Simulation, UKSim-AMSS 16th*, 26-28 March 2014, Cambridge, United Kingdom, pp.63-68, doi=10.1109/UKSim.2014.107.

5. Santos, W.R.M., Queiroz, J.A., Fonseca Neto, J.V., Rêgo, P.H.M., Santana, E. e Andrade, G. (2014), "RLS Algorithms and Convergence Analysis Method for Online DLQR Optimal Control Design via Heuristic Dynamic Programming", in *International Conference on Computer Modelling and Simulation, UKSim-AMSS 16th*, 26-28 March 2014, Cambridge, United Kingdom, pp.77-83, doi=10.1109/UKSim.2014.109.

Capítulo 2

Aprendizagem por Reforço

2.1 Introdução

Nas abordagens convencionais de aprendizagem, os parâmetros do sistema são ajustados com base em um conjunto relevante de exemplos entrada-saída fornecido por um "professor"(agente externo). Em termos conceituais, o professor é considerado como tendo conhecimento prévio sobre o ambiente, sendo capaz de fornecer ao sistema uma resposta desejada (ações ótimas a serem realizadas pelo sistema) para um dado padrão de entrada. O sistema, por sua vez, responde a partir do ambiente com saídas que são comparadas com as saídas desejadas. Alguma função do erro na saída do sistema (medida de desempenho do sistema, como por exemplo o erro quadrático médio) é, então, minimizada para ajustar os parâmetros do sistema. Daí porque tal processo de aprendizagem se enquadra como um tipo de aprendizagem por correção de erro, denominada aprendizagem supervisionada. A Figura 2.1(b) ilustra de forma simplificada o processo de aprendizagem supervisionda.

Mas, quando se deseja que o agente tenha autonomia total, ou seja, aprendizagem precisa ser realizada sem a tutela de um professor que fornece informação sobre o ambiente em cada passo do processo de aprendizagem, o agente terá que ser capaz de aprender com base em outras informações como, por exemplo, recompensas ou reforços fornecidos por um "crítico"ou pelo próprio ambiente. A Figura 2.1(a) ilustra uma forma de aprendizagem por reforço construída em torno de um crítico que converte um sinal de reforço primário recebido do ambiente em

um sinal de reforço de melhor qualidade, denominado sinal de reforço heurístico, sendo ambos entradas escalares.

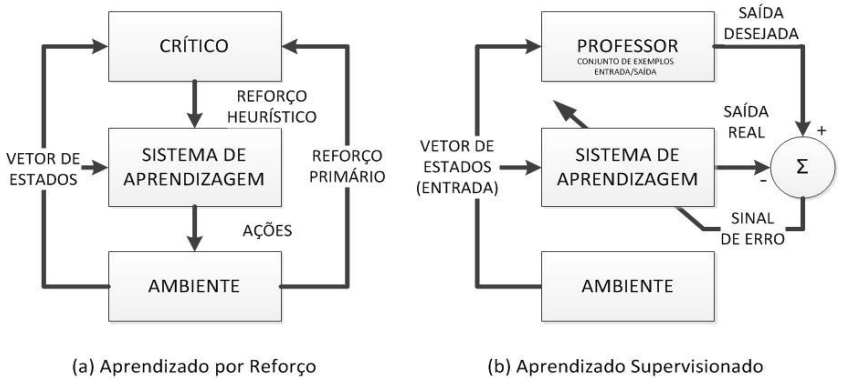

Figura 2.1: Comparativo Aprendizagem por Reforço e Aprendizagem Supervisionada.

O sistema de aprendizagem por reforço é projetado para aprender por reforço atrasado, o que significa que o sistema observa uma sequência temporal de estímulos (i.e., vetores de estado) também recebidos do ambiente, que eventualmente resultam na geração do sinal de reforço heurístico. O objetivo da aprendizagem é maximizar uma função valor, definida como a expectativa da recompensa cumulativa de ações tomadas ao longo de uma sequência de passos, em vez simplesmente da recompensa imediata. Pode acontecer que certas ações tomadas anteriormente naquela sequência de passos de tempo sejam de fato os melhores determinantes do comportamento global do sistema, (Haykin, 1998).

O propósito do presente capítulo é apresentar, de forma introdutória, os fundamentos teóricos relacionados à aprendizagem por reforço considerando uma abordagem de tempo discreto. Os conceitos e métodos são apresentados no contexto da formulação de *Bellman* de programação dinâmica. Na Seção 2.2, define-se formalmente o problema de aprendizagem por reforço em termos de um processo de decisão markoviano de uma forma genérica. Na Seção 2.3, apresentam-se o princípio básico e alguns métodos da programação dinâmica para processos de decisão markovianos. Na Seção 2.4, discutem-se alguns métodos independentes de modelo como uma alternativa à programação dinâmica convencional que é baseada no conhecimento do modelo do processo de decisão markoviano.

2.2 O Problema de Aprendizagem por Reforço

Nesta seção, apresentam-se os elementos fundamentais de um problema de aprendizagem por reforço, bem como sua definição formalizada em termos de um processo de decisão markoviano.

No quadro de aprendizagem por reforço um agente age em um ambiente cujo estado pode ser observado e ocasionalmente recebe alguma penalidade (sinal RL negativo) ou recompensa (sinal RL positivo) baseada em seu estado e ação. Sua tarefa de aprendizagem é encontrar uma política para seleção de ação que maximize sua recompensa a longo prazo. Esta tarefa requer não somente a escolha de ações que estão associadas com alta recompensa no estado atual, mas também a escolha de ações que levarão o agente à partes mais lucrativas do espaço de estado. No contexto de controle, recompensa implica em uma diminuição de custo de controle, enquanto penalidade causa um 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:

- Uma política (decisão), a qual é definida como o comportamento de um agente em um instante de tempo particular. A política é uma parte muito importante de esquemas 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, a política é calculada por maximização da função valor). O processo RL levará a uma política ótima (comportamento) ao longo do tempo.

- Uma função de utilidade que é definida sobre a base do objetivo do problema de aprendizagem por reforço e ela 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 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 $V(x)$, a qual é uma predição da soma esperada em longo prazo de recompensas descontadas futuras 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 tomar 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 fatores Q), a qual é uma predição de recompensas descontadas total esperadas associadas com pares estado-ação (x, u) iniciais, normalmente representada por $Q(x, u)$. Como explicado em (Sutton e Barto, 1998), uma função de utilidade avalia a recompensa imediata enquanto a função valor é a predição da recompensa futura total, daí, decisões sobre tomar ações são realizadas sobre a base da função valor de estado ou ação.

2.2.1 Formulação do Problema

Um problema RL é geralmente formulado por um processo de decisão markoviano (PDM) de tempo discreto $\wp = (X, U, \widetilde{f}, r)$, sendo X o espaço de estados, U o espaço de ações, $\widetilde{f} : X \times U \times X \to [0, \infty)$ é uma função densidade de probabilidade de transição de estados dada por

$$P_r(x_{k+1} \in X_{k+1} \mid x_k, u_k) = \int_{X_{k+1}} \widetilde{f}(x_k, u_k, x')dx',$$

sendo P_r o operador de probabilidade do estado seguinte x_{k+1}, este pertencer à uma região $X_{k+1} \subseteq X$ devido à ação u_k aplicada no estado atual x_k, $r : X \times U \times X \to \Re$; $r(x_k, u_k, x_{k+1}) = r_k$ é uma função de utilidade que determina a recompensa instantânea r_k recebida de uma transicão do estado x_k para o estado x_{k+1} sob à ação u_k realizada no passo de tempo k, (Buşoniu *et al.*, 2010*e*). Para cada estado $x_k \in X$ existe um subconjunto $U(x_k) \subseteq U$ de ações admissíveis a partir do qual u_k deve ser escolhida. Devido à propriedade de Markov (veja

Definição A.1.3 no Apêndice A), o estado corrente x_k e a ação correspondente u_k são suficientes para determinar a densidade de probabilidade do estado seguinte.

Uma política é uma sequência $\pi = \{h_i\}_{i=k}^{\infty}$, sendo cada $h_i : X \to U$ uma função que produz a ação u_i a ser tomada no passo de tempo i, com $h_i(x_i) \in U(x_i)$ para todos os estados x_i. Para uma dada política π, define-se a função valor de estado, ou simplesmente função valor, $V^{\pi} : X \to \Re$, por

$$V^{\pi}(x_k) = E_{x_{i+1} \sim \widetilde{f}(x_i, h_i(x_i), \cdot)} \left[\sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h_i(x_i), x_{i+1}) \,|\, x_k \right], \tag{2.1}$$

que representa a recompensa esperada em longo prazo (obediência aos termos de compromisso para avaliar o objetivo de aprendizagem), com $0 \leq \gamma \leq 1$ um fator de desconto, E denota o operador esperança com respeito à variável aleatória x_{i+1}, e a notação $x_{i+1} \sim \widetilde{f}(x_i, h_i(x_i), \cdot)$ significa que o estado x_{i+1} é determinado de acordo com a densidade $\widetilde{f}$.

O objetivo de um PDM é determinar uma política, π^*, que é ótima no sentido de promover a maior coleção possível de recompensas imediatas esperadas, que satisfaz

$$V^{\pi^*}(x_k) \geq V^{\pi}(x_k), \tag{2.2}$$

para todo $x_k \in X$ e para todas as políticas π, (Bradtke e Barto, 1996).

O problema de avaliar uma dada política ou determinar uma política ótima para o caso onde todo o conhecimento das funções $\widetilde{f}$ e r vem da interação entre o tomador de decisão e o próprio processo, ou de um modelo gerador do processo[1] é conhecido como aprendizagem por reforço, e inclui observações de estados, ações e recompensas, (Lagoudakis e Parr, 2003). Estas observações são geralmente organizadas em tuplas da forma (x_k, u_k, r_k, x_{k+1}) observadas ao longo da realização de trajetória do estado.

Em particular se X é um conjunto discreto e finito, uma função de probabilidade de transição de estados é dada por $\widetilde{f} : X \times U \times X \to [0, 1]$; $P_r(x_{k+1} = x' \mid x_k, u_k) = \widetilde{f}(x_k, u_k, x')$, sendo P_r o operador de probabilidade de transição de

[1] Um modelo gerador de um PDM é um simulador do processo, isto é, uma "caixa preta"que dados um estado x_k e uma ação u_k, gera uma recompensa r_k e um estado seguinte x_{k+1} amostrados de acordo com a dinâmica do processo.

x_k para x_{k+1} sob à ação u_k. De particular interesse em problemas de aprendizagem por reforço são as políticas estacionárias, ou seja, aquelas que não dependem do passo de tempo k: $h_i = h_j, \forall i, j$. Em outras palavras, uma política estacionária especifica exatamente a mesma ação cada vez que um estado particular for visitado. Por brevidade, a política $\pi = (h, h, \ldots)$ será referida como a política estacionária h.

2.3 Programação Dinâmica

A metodologia conhecida como programação dinâmica encontra-se fundamentada no chamado princípio da otimalidade de *Bellman*. Tal princípio sugere que uma sequência ótima de ações (política ótima) possui a propriedade que independente das decisões tomadas em instantes de tempo anteriores ao atual, as decisões futuras devem constituir uma política ótima em relação ao estado resultante das decisões anteriores. Em programação dinâmica distingue-se entre os problemas de horizonte finito, em que a recompensa se acumula sobre um número finito de estágios, e problemas de horizonte infinito, em que a recompensa se acumula indefinidamente. A fim de fornecer uma formulação matemática do princípio da otimalidade de *Bellman*, considere um problema de horizonte finito, conforme definido em (Bertsekas, 1995*a*), para o qual a função valor de uma política π é dada por

$$V_0^{\pi}(x_0) = E\left[r_N(x_N) + \sum_{k=0}^{N-1} r_k(x_k, h_k(x_k), x_{k+1})\right], \tag{2.3}$$

sendo N o horizonte (número de estágios), x_0 o estado inicial, $r_N(x_N)$ uma recompensa terminal incorrida no final do processo, e o valor esperado $E(\cdot)$ é com respeito aos estados restantes $x_1, \ldots, x_{N-1}$. O problema é definido para se determinar uma política ótima $\pi^* = \{h_0^*, h_1^*, \ldots, h_{N-1}^*\}$ que maximize (2.3) para todos os estados iniciais x_0, isto é,

$$\pi^* = \arg\max_{\pi \in \prod} V_0^{\pi}(x_0), \ \forall\, x_0, \tag{2.4}$$

sendo Π o conjunto de políticas admissíveis.

Princípio da Otimalidade de *Bellman* (Bertsekas, 1995*a*): *Seja* $\pi^* = \{h_0^*, h_1^*, \ldots, h_{N-1}^*\}$ *uma política ótima para o problema (2.4), e suponha que ao utilizar* π^*, *um dado estado* x_k *ocorre no passo de tempo* k *com probabilidade positiva. Considere o subproblema em que o ambiente está no estado* x_k *no passo de tempo* k *e suponha que se deseja maximizar a função valor correspondente*

$$V_k^{\pi}(x_k) = E\left[r_N(x_N) + \sum_{i=k}^{N-1} r_i(x_i, h_i(x_i), x_{i+1})\right], \tag{2.5}$$

para $k = 0, 1, \ldots, N-1$. *Então, a política truncada* $\{h_k^*, h_{k+1}^*, \ldots, h_{N-1}^*\}$ *é ótima para o subproblema.*

Proposicão (Algoritmo de Programação Dinâmica) (Bertsekas, 1995*a*): *Para todo estado inicial* x_0, *a função valor ótimo* $V^*(x_0)$ *associada ao problema (2.4) é igual a* $V_0(x_0)$, *sendo a função* V_0 *obtida do último passo do seguinte algoritmo, o qual prossegue para trás no tempo, do passo de tempo* $N-1$ *para o passo de tempo* 0:

$$V_N(x_N) = r_N(x_N) \tag{2.6}$$

$$V_k(x_k) = \max_{h_k} \underset{x_{k+1}}{E} \left[r_k(x_k, h_k(x_k), x_{k+1}) + V_{k+1}(x_{k+1})\right]. \tag{2.7}$$

Além disso, se h_k^* *maximiza o lado direito da Eq.(2.7) para cada* x_k *e* k, *então a política* $\pi^* = \{h_0^*, h_1^*, \ldots, h_{N-1}^*\}$ *é ótima.*

Demonstração:

Seja $\pi = \{h_0, h_1, \ldots, h_{N-1}\}$ uma política admissível. Para cada $k = 0, 1, \ldots, N-1$, denote $\pi^k = \{h_k, h_k + 1, \ldots, h_{N-1}\}$, e suponha que $V_k^*(x_k)$ seja a função valor ótima para o problema de $N-k$ estágios que inicia no estado x_k e passo de tempo k, e termina no passo de tempo N, isto é,

$$V_k^*(x_k) = \max_{\pi^k} \underset{(x_{k+1}, \ldots, x_{N-1})}{E} \left[r_N(x_N) + \sum_{i=k}^{N-1} r_i(x_i, h_i(x_i), x_{i+1})\right] \tag{2.8}$$

que representa a forma ótima da Eq.(2.5).

Para $k = N$, defina $V_N^*(x_N) = r_N(x_N)$. Prova-se por indução que as funções V_k^* são iguais às funções V_k geradas pelo algoritmo DP, de modo que para $k = 0$, obtém-se o resultado desejado.

De fato, tem-se por definição $V_N^* = V_N = r_N$. Agora suponha que para um dado k e para todo x_{k+1} tem-se $V_{k+1}^*(x_{k+1}) = V_{k+1}(x_{k+1})$. Reconhecendo que $\pi^k = (h_k, \pi^{k+1})$, e expandindo parcialmente o somatório no lado direito da Eq.(2.8), pode-se escrever para todo x_k

$$\begin{aligned}
V_k^*(x_k) &= \max_{(h_k,\pi^{k+1})} \underset{(x_{k+1},\ldots,x_{N-1})}{E} \Big[r_k(x_k, h_k(x_k), x_{k+1}) \\
&\qquad + r_N(x_N) + \sum_{i=k+1}^{N-1} r_i(x_i, h_i(x_i), x_{i+1})\Big] \\
&= \max_{h_k} \underset{x_{k+1}}{E} \Big\{r_k(x_k, h_k(x_k), x_{k+1}) \\
&\qquad + \max_{\pi^k} \underset{(x_{k+2},\ldots,x_{N-1})}{E} \Big[r_N(x_N) + \sum_{i=k+1}^{N-1} r_i(x_i, h_i(x_i), x_{i+1})\Big]\Big\} \qquad (2.9) \\
&= \max_{h_k} \underset{x_{k+1}}{E} \big[r_k(x_k, h_k(x_k), x_{k+1}) + V_{k+1}^*(x_{k+1})\big] \\
&= \max_{h_k} \underset{x_{k+1}}{E} \big[r_k(x_k, h_k(x_k), x_{k+1}) + V_{k+1}(x_{k+1})\big] \\
&= V_k(x_k),
\end{aligned}$$

sendo que na terceira equação usou-se a definição da Eq.(2.8) com $k+1$ no lugar de k.

Em problemas de horizonte infinito, a recompensa total esperada associada a um estado inicial x_0 e a uma política $\pi = (h_0, h_1, \ldots)$ é dada por

$$V^\pi(x_0) = \lim_{N\to\infty} E\left[\sum_{k=0}^{N-1} \gamma^k r(x_k, h_k(x_k),\, x_{k+1}\,|x_0\right]. \qquad (2.10)$$

A função valor ótima correspondente é o limite das recompensas ótimas esperadas à N-estágios quando $N \to \infty$, isto é, $V^*(x_0) = \max_\pi V^\pi(x_0)$.

2.3.1 Equação de *Bellman* e Política Ótima

A programação dinâmica para processos de decisão markovianos baseia-se na propriedade de que a função valor de uma dada política h (estacionária) satisfaz a seguinte relação de recorrência

$$V^h(x_k) = E_{x_{k+1}\sim\widetilde{f}(x_k,h(x_k),\cdot)}\left[r(x_k,h(x_k),x_{k+1}) + \gamma V^h(x_{k+1})\right], \tag{2.11}$$

para cada $x_k \in X$, (Buşoniu *et al.*, 2010*e*). A Eq.(2.11) é conhecida como equação de *Bellman*. Ela expressa a função valor no estado x_k de uma política h como o valor esperado da soma da recompensa imediata recebida pela execução da ação $u_k = h(x_k)$ com a função valor no estado x_{k+1} sob a política h em relação à todos os estados seguintes possíveis x_{k+1}. A Eq.(2.11) é verdadeira para todas as políticas, incluindo política ótimas. Supondo que h^* é uma política ótima, a Eq.(2.11) para h^* pode ser reescrita como

$$V^{h^*}(x_k) = E_{x_{k+1}\sim\widetilde{f}(x_k,h^*(x_k),\cdot)}\left[r(x_k,h^*(x_k),x_{k+1}) + \gamma V^{h^*}(x_{k+1})\right]. \tag{2.12}$$

Mas, todas as políticas ótimas conduzem a mesma função valor, V^*, e $V^*(x_k) \geq V^h(x_k)$ para todas as políticas h. Portanto, a ação $u_k = h^*(x_k)$ escolhida por uma política ótima deve maximizar a expressão

$$E_{x_{k+1}\sim\widetilde{f}(x_k,u_k,\cdot)}\left[r(x_k,u_k,x_{k+1}) + \gamma V^*(x_{k+1})\right]. \tag{2.13}$$

Combinando todas essas relações, resulta na Equação da Otimalidade de *Bellman*

$$V^*(x_k) = \max_{u_k\in U(x_k)}\left\{E_{x_{k+1}\sim\widetilde{f}(x_k,u_k,\cdot)}\left[r(x_k,u_k,x_{k+1}) + \gamma V^*(x_{k+1})\right]\right\}. \tag{2.14}$$

Uma vez que V^* é conhecida, uma política ótima é determinada por meio da seguinte equação

$$h^*(x_k) = \arg\max_{u_k\in U(x_k)}\left\{E_{x_{k+1}\sim\widetilde{f}(x_k,u_k,\cdot)}\left[r(x_k,u_k,x_{k+1}) + \gamma V^*(x_{k+1})\right]\right\}. \tag{2.15}$$

2.3.2 Iteração de Política

A iteração de política (IP) consiste de um processo iterativo de busca da política ótima h^*, em que para cada iteração do processo, realiza-se uma etapa de avaliação de política que determina a função valor de estado V^{h_j} associada

à política atual h_j como a solução da equação de *Bellman* (2.11). Esta etapa funciona no *loop* interno de cada iteração de política. Em seguida, realiza-se uma etapa de melhoria de política, na qual estabelece-se uma nova política de controle h_{j+1} que é dada por

$$h_{j+1}(x_k) = \arg \max_{u_k \in U(x_k)} \left\{ E_{x_{k+1} \sim \tilde{f}(x_k, u_k, \cdot)} \left[r(x_k, u_k, x_{k+1}) + \gamma V^{h_j}(x_{k+1}) \right] \right\}. \quad (2.16)$$

Este procedimento é descrito por Howard (Howard, 1960) que prova que a nova política h_{j+1} é melhorada com respeito à h_j no sentido que $V^{h_{j+1}}(x_k) \geq V^{h_j}(x_k)$. Os resultados de convergência para iteração de política baseiam-se sobre uma tabela de consulta que fornece as representações exatas das políticas e das funções valor. Se as etapas de avaliação e melhoria de políticas são realizadas corretamente, o processo convergirá para h^* após vários ciclos de operação, sob uma política inicial admissível.

2.3.3 Iteração Gulosa

Na estratégia IP padrão a política atual h 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, $\widehat{V}^*$, é atualizada de acordo com a equação da otimalidade de *Bellman*, Eq.(2.14), que é dada por

$$\widehat{V}^*(x) = E_{x' \sim \tilde{f}(x, \widehat{h}^*(x), \cdot)}[r(x, \widehat{h}^*(x), x') + \gamma \widehat{V}^*(x')]. \quad (2.17)$$

$$\widehat{h}^*(x) = \arg \max_{u \in U(x)} \left\{ E_{x' \sim \tilde{f}(x, u, \cdot)}[r(x, u, x') + \gamma \widehat{V}^*(x')] \right\}. \quad (2.18)$$

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.(2.17) e (2.18) são avaliadas na ordem inversa pois o foco é sobre a política parametrizada $\widehat{h}^*(x,\theta)$, 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 e Tsitsiklis, 1996) que atualiza a política atual após cada transição de estado x_k para x_{k+1}.

2.3.4 Iteração de Valor

A iteração de valor é um método de aproximações sucessivas para encontrar a função valor ótima sem requerer o conhecimento prévio de uma política ótima. A política ótima é obtida a partir da função valor ótima. O método atualiza as estimativas da função valor de acordo com seguinte equação

$$V_{k+1}(x_k) = \max_{u_k \in U(x_k)} \left\{ E_{x_{k+1} \sim \widetilde{f}(x_k, u_k, \cdot)} \left[r(x_k, u_k, x_{k+1}) + \gamma V_k(x_{k+1}) \right] \right\}, \qquad (2.19)$$

para todos os estados x_k. A iteração de valor é garantida convergir para uma função que satisfaz a Eq.(2.14) baseada em uma representação tabular da função valor, (Bertsekas, 1995*b*).

2.4 Diferenças Temporais

Os métodos tradicionais de programação dinâmica são baseados em um modelo do processo de decisão markoviano, ou seja, os métodos requerem conhecimento dos modelos $\widetilde{f}$ da função de probabilidade de transição e r da função de utilidade. Entretanto, às vezes um modelo do sistema não pode ser obtido em sua plenitude, porque a obtenção de um modelo é muito cara, se não impossível fornecer estimativas das probabilidades de transição de estados. Neste caso, métodos de diferenças temporais (*Temporal Differences* - TD) (Sutton e Barto, 1998) são úteis, uma vez que funcionam utilizando apenas os dados obtidos a partir do sistema, sem requerer um modelo do seu comportamento. Métodos TD originaram-se de um estudo de algoritmos de aprendizagem por reforço e do problema de atribuição de crédito, (Sutton, 1984), (Sutton, 1988), (Anderson, 1988), (Barto *et al.,* 1983).

2.4.1 Avaliação de Política - Predição por Diferenças Temporais

Nesta subseção, apresenta-se uma classe de métodos independentes de modelo baseados em diferenças temporais para estimar a função valor V^h para uma dada política h. A abordagem envolve a correção das predições anteriores e constrói estimativas da função valor V^h baseadas em tuplas de dados da forma $(x_k, x_{k+1}, r(x_k, h(x_k), x_{k+1}))$ observadas ao longo da realização de trajetória do estado, (Sutton e Barto, 1998). O método, em sua versão à um passo, atualiza V^h após cada transição de estado de acordo com a equação que é dada por

$$V_{k+1}(x_k) = V_k(x_k) + \alpha_k \Delta_k, \tag{2.20}$$

com V^h inicializado para algum valor arbitrário V_0, V_k é a k-ésima estimativa de V^h, α_k é um fator de aprendizagem, e Δ_k é a diferença temporal que é dada por

$$\Delta_k = r(x_k, h(x_k), x_{k+1}) + \gamma V_k(x_{k+1}) - V_k(x_k), \tag{2.21}$$

expressando a diferença entre a estimativa atualizada $r(x_k, u_k, x_{k+1}) + \gamma V_k(x_{k+1})$ da função valor V^h para o estado x_k e a estimativa atual $V_k(x_k)$ associada com a transição de x_k para x_{k+1}. De acordo com a Eq.(2.11) de *Bellman* se as estimativas de valor geradas pelo algoritmo de Eq.(2.20) forem exatas, a diferença temporal média Δ_k é zero. Consequentemente, os valores Δ_k podem ser visto como uma indicação de ajuste para corresponder à equação de *Bellman* ao longo das trajetórias amostradas.

A regra de Eq.(2.20) é um caso especial de uma classe mais geral de métodos de diferenças temporais, chamados TD(λ), com $\lambda = 0$. A aprendizagem TD(λ) foi introduzida em (Sutton, 1988). Excelentes estudos sobre TD(λ) podem ser encontrados em (Dayan, 1992), (Bertsekas e Tsitsiklis, 1996), (Sutton e Barto, 1998). TD(λ) é uma abordagem que usa recompensas truncadas à m-passos corrigidas $R_k^{(m)}$, que são definidas por

$$R_k^{(m)} = r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + \ldots + \gamma^m V_k(x_{k+m}), \tag{2.22}$$

sendo $\gamma^m V_k(x_{k+m})$ o termo aproximadamente corrigido para o truncamento, o qual estima o valor do desempenho no próximo m-ésimo estado, isto é, o valor de $\sum_{l=m}^{\infty} \gamma^l r_{k+l}$. A família TD($\lambda$) parametrizada pelo parâmetro λ, com $0 \leq \lambda \leq 1$, combina recompensas à m passos ponderadas proporcionalmente à λ^{m-1}. Desta forma, uma média ponderada de recompensas truncadas à m-passos corrigidas é definida por

$$R_k^\lambda = (1-\lambda) \sum_{m=1}^{\infty} \lambda^{m-1} R_k^{(m)} \tag{2.23}$$

$$= (1-\lambda) \sum_{m=1}^{T-k-1} \lambda^{m-1} R_k^{(m)} + \lambda^{T-k-1} R_k, \tag{2.24}$$

sendo que o fator $1-\lambda$ é aplicado para normalizar a soma dos pesos para 1. A segunda equação é obtida de modo que a recompensa à m passos após um estado terminal é o Retorno de *Monte Carlo*, denotado por R_k. Desta forma, se $\lambda = 1$, $R_k^\lambda = R_k$, isto é, as atualizações são baseadas em toda a seqüência de recompensas observadas até o final de um episódio. Se $\lambda = 0$, $R_k^\lambda = R_k^{(1)}$, isto é, as atualizações são baseadas simplesmente nas amostras de recompensas imediatas como em todos os algoritmos RL à um passo. O parâmetro λ determina a influência de recompensas truncadas à m passos sobre a função valor $V(x)$ em uma taxa exponencial. A regra de atualização é determinada a partir da Eq.(2.24), a qual é dada por

$$V_{k+1}(x) = V_k(x) + \alpha_k (R_k^\lambda - V_k(x)). \tag{2.25}$$

O problema em se implementar a Eq.(2.25) é que teria que se esperar indefinidamente para calcular $R_k^{(\infty)}$. Esta visão é útil para análise teórica e entendimento de backups à m passos. Ela é denominada visão "para frente"do algoritmo TD(λ). Entretanto, a maneira usual para implementar estes tipos de atualizações é chamada visão "para trás"do algoritmo TD(λ) e isto é feito usando-se traços de elegibilidade, a qual é uma implementação incremental da mesma idéia.

Traços de elegibilidade podem ser vistos como um artifício algébrico pelo qual TD(λ) propaga recompensas para trás durante a trajetória atual sem ter que lembrar da trajetória explicitamente. Isto é uma maneira de realizar backups à m

passos. Para cada estado $x \in X$, uma elegibilidade $\xi_k(x)$ é mantida na memória, informando quais estados foram recentemente visitados. Traços de elegibilidade são incrementados a cada passo de tempo de acordo com

$$\xi_k(x) = \begin{cases} \gamma\lambda\xi_{k-1}(x) & se \quad x \neq x_k \\ \gamma\lambda\xi_{k-1}(x) + 1 & se \quad x = x_k, \end{cases} \tag{2.26}$$

com $\xi_0(x) = 0$, $\forall x \in X$. O traço de elegibilidade para cada estado é acumulado cada vez que o estado for visitado. Caso contrário, diminuirá exponencialmente até que o estado seja revisitado. Em cada passo, um incremento para o valor do estado é dado proporcionalmente ao seu respectivo traço de elegibilidade de acordo com

$$V_{k+1}(x) = V_k(x) + \alpha_k \Delta_k \xi_k(x), \tag{2.27}$$

sendo Δ_k a diferença temporal no passo de tempo k definida na Eq.(2.21).

2.4.2 Aprendizagem Q

A aprendizagem Q originou-se no trabalho de (Watkins, 1989), (Watkins e Dayan, 1992). Duas formas básicas de aprendizagem Q são apresentadas nesta seção: Aprendizagem Q ótima e Aprendizagem Q baseada em política. Esses métodos podem ser vistos como um tipo de aproximação estocástica (Bertsekas e Tsitsiklis, 1996) para construir estimativas da função Q (veja definição a seguir) para qualquer política ótima ou não-ótima.

Função Q

A função Q, ou função valor de ação, é definida por $Q^h : X \times U \to \Re$,

$$Q^h(x_k, u_k) = E_{x_{k+1} \sim \tilde{f}(x_k, u_k, \cdot)} \left[r(x_k, u_k, x_{k+1}) + \gamma V^h(x_{k+1}) \right]. \tag{2.28}$$

A partir desta definição e da Eq.(2.11), observa-se que $Q^h(x_k, h(x_k)) = V^h(x_k)$, e, portanto, a função Q^h pode ser expressa na forma de *Bellman* como

$$Q^h(x_k, u_k) = E_{x_{k+1} \sim \widetilde{f}(x_k, u_k, \cdot)} \left[r(x_k, u_k, x_{k+1}) + \gamma Q^h(x_{k+1}, h(x_{k+1})) \right]. \quad (2.29)$$

Assim, a função Q_L ótima deve satisfazer

$$\begin{aligned} Q^*(x_k, u_k) &= E_{x_{k+1} \sim \widetilde{f}(x_k, u_k, \cdot)} \left[r(x_k, u_k, x_{k+1}) + \gamma V^*(x_{k+1}) \right] & (2.30) \\ &= E_{x_{k+1} \sim \widetilde{f}(x_k, u_k, \cdot)} \left[r(x_k, u_k, x_{k+1}) + \gamma Q^*(x_{k+1}, h^*(x_{k+1})) \right]. & (2.31) \end{aligned}$$

Desta forma, uma equação da otimalidade de *Bellman* para funções Q é dada por

$$Q^*(x_k, u_k) = E_{x_{k+1} \sim \widetilde{f}(x_k, u_k, \cdot)} \left[r(x_k, u_k, x_{k+1}) + \gamma \max_{a \in U(x_k)} Q^*(x_{k+1}, a) \right]. \quad (2.32)$$

Esta equação caracteriza Q^*, e declara que o valor ótimo de uma ação u_k aplicada no estado x_k é o valor esperado da soma da recompensa imediata com o valor ótimo descontado obtido pela melhor ação no estado seguinte. Consequentemente, uma política ótima h^* pode ser determinada a partir de Q^* por

$$h^*(x_k) = \arg\max_u Q^*(x_k, u). \quad (2.33)$$

Uma vez que a função Q depende 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. Neste caso, para inferir a qualidade das transições, estas devem ser explicitamente levadas em consideração. Consequentemente, um modelo do MDP é necessário na forma da dinâmica $\widetilde{f}$ e da função de utilidade r, enquanto que na formulação usando função Q o problema de decisão markoviano é tratado sem referência à tais modelos.

Aprendizagem Q ótima

Aprendizagem Q é um método de política-*off*, o que siginifica que a função valor ótima Q^* é estimada, independentemente da política atual (exploração) que está sendo utilizada para gerar as trajetórias amostradas. A regra de aprendizagem Q ótima pode ser expressa como:

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha_k \left[r(x_k, u_k, x_{k+1}) + \gamma \max_{a \in U(x_k)} Q_k(x_{k+1}, a) - Q_k(x_k, u_k) \right], \quad (2.34)$$

sendo Q_k a k-ésima estimativa para Q^* e α_k os tamanhos de passos (*stepsizes*).

Admitindo-se uma representação tabular da função Q, aprendizagem Q converge com probabilidade 1 para o valor ótimo Q^* quando $k \to \infty$, sob as seguintes suposições, (Watkins e Dayan, 1992):

1. Uma sequência decrescente apropriada de tamanhos dos passos α_k que satisfaz as condições $\sum_{k=0}^{\infty} \alpha_k = \infty$ e $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$, e
2. Todos os pares estado-ação são visitados, assintoticamente, infinitas vezes.

Por exemplo, uma sequência padrão de tamanhos de passos que garante a convergência da aprendizagem Q é dada por

$$\alpha_k = \frac{\tau}{\eta + k}, \quad k = 1, 2, \ldots, \quad (2.35)$$

sendo τ e η números positivos.

A condição 2) pode ser satisfeita se o agente de aprendizagem selecionar com probabilidade não-nula uma ação aleatória em cada estado visitado (exploração) e, além disso, explotar seu conhecimento atual para obter um bom desempenho, por exemplo, selecionando-se ações gulosas com respeito à sua função Q atual (explotação). Uma abordagem típica que lida com exploração-explotação em algoritmos RL é a chamada exploração gulosa ε (Sutton e Barto, 1998), a qual seleciona ações de acordo com a seguinte regra:

$$u_k = \begin{cases} u \in \arg \max_a Q_k(x_k, a), \textit{ com probabilidade } 1 - \varepsilon_k \\ \textit{uma ação uniformemente aleatória em } U(x_k), \textit{ com probabilidade } \varepsilon_k, \end{cases} \quad (2.36)$$

sendo $\varepsilon_k \in (0, 1)$ a probabilidade de exploração no passo de tempo k.

Aprendizagem baseada em política

Aprendizagem Q baseada em política tenta aprender Q^h, a função Q para alguma política projetada h. A política h pode ser ou não ser a política que de fato está sendo seguida durante o treinamento. A regra de aprendizagem baseada em política é dada por

$$\begin{aligned} Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) \\ + \alpha_k \left[r(x_k, u_k, x_{k+1}) + \gamma Q_k(x_{k+1}, h(x_{k+1})) - Q_k(x_k, u_k) \right], \end{aligned} \tag{2.37}$$

sendo Q_k a k-ésima estimativa para Q^h.

Pouco depois dos trabalhos de Watkins e Dayan (Watkins, 1989), (Watkins e Dayan, 1992), Peng e Williams (Peng e Williams, 1996) apresentaram um método de aprendizagem Q multi-passos incremental, $Q(\lambda)$, uma combinação de aprendizagem Q a um passo e TD(λ). Sutton e Barto (Sutton e Barto, 1998) realizaram uma comparação entre aprendizagem Q multi-passos, $Q(\lambda)$ e o método sugerido por Watkins (Watkins, 1989). Similarmente, Kuzmin (Kuzmin, 2002) sugeriu uma modificação à aprendizagem Q tabular tradicional. Ele propõe um perceptron multicamadas para atuar como um aproximador de avaliação de aprendizagem Q. Ele denominou esta combinação, "conexionismo". Antes do trabalho dele, um esquema similar para aproximação de função valor de estado foi proposto por Pipe (Pipe, 1998), onde o método tabular tradicional foi substituído por uma aproximação de função via uma função de base radial NN. No esquema proposto, ele denominou a função valor de estado como um campo potencial. Deng e Er (Deng e Er, 2004) propuseram aprendizagem Q Fuzzy. Neste esquema, a aprendizagem Q é usada para gerar e ajustar automaticamente regras fuzzy para uma tarefa de navegação de robô móvel. Bertsekas e Yu (Bertsekas e Yu, 2010) propuseram aprendizagem Q e melhoram iteração de política em programação dinâmica com desconto.

2.4.3 SARSA (Estado-Ação-Recompensa-Estado-Ação)

Proposto por (Rummery e Niranjan, 1994), SARSA é similar à aprendizagem Q, entretanto, é um método TD de política-*on*, daí, a função valor de ação, $Q(x, u)$, é estimada para a política e para o par estado-ação atuais, isto é, SARSA

avalia a política que atualmente está sendo usada para controlar o processo, diferente de aprendizagem-Q, a qual age sobre o processo usando uma política e avalia uma outra política. Por exemplo, em aprendizagem Q ótima, a política usada para controlar o sistema tipicamente inclui exploração, enquanto que o método implicitamente avalia uma política que é gulosa na função Q atual (política que escolhe ações que maximizam Q), uma vez que os valores máximos de Q são usados nas atualizações da função Q^*. Para selecionar ações, SARSA combina uma política gulosa na função Q atual com exploração, usando-se, por exemplo, exploração gulosa ε ou exploração *Boltzmann*, (Buşoniu *et al.,* 2010*e*). Devido ao componente guloso, SARSA realiza implicitamente uma melhoria de política em cada etapa de tempo, e é portanto um tipo de iteração de política *online*. Tal esquema de iteração de política, o qual melhora a política após cada amostra, é algumas vezes chamado otimístico.

2.4.4 Métodos Ator-Crítico

Os métodos ator-crítico fornecem procedimentos "para frente no tempo"para determinação de políticas de controle ótimas que podem ser implementadas em tempo real. Trata-se de uma forma de aprendizagem TD de política-*on* que representa as políticas de forma independente da função valor.

Há uma interpretação interessante dos métodos ator-crítico quando versões aproximadas de iteração de política são levadas em consideração. O método consiste de duas estruturas paramétricas separadas (por exemplo redes neurais), uma para avaliação de política, a outra estrutura de rede é para a melhoria de política. A etapa de avaliação da política é vista como o trabalho de um crítico, que avalia o desempenho da política atual, isto é, calcula uma estimativa de V^{h_k}. A etapa de melhoria de política é vista como o trabalho de um ator, que leva em conta a avaliação mais recente do crítico, isto é, a estimativa de V^{h_k}, e age de acordo com a política melhorada h_{k+1}. No contexto de controle, o ator é um agente interagindo com o ambiente, isto é, o ator é o controlador realizando ações de controle. A ação tomada pelo ator (controlador) é avaliada pelo crítico (função de recompensa e função de custo) usando uma abordagem de diferença temporal.

Um esquema ator-crítico pode ser visto na Figura 2.2. O crítico assume a forma de um erro TD, sendo calculado como na Eq.(2.21). A etapa de atualização

da função valor V é realizada de acordo com a Eq.(2.20). O crítico avalia o desempenho da ação selecionada sobre a base de Δ_k.

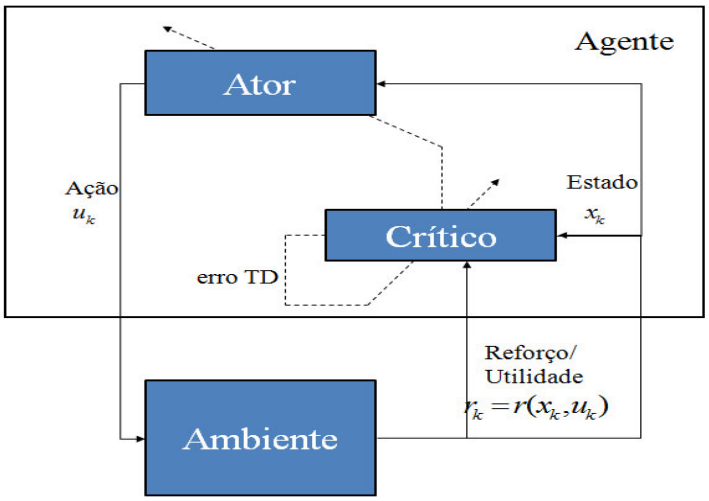

Figura 2.2: Estrutura de Aprendizado por Reforço com Ator/Crítico (Dissertação de Leandro Rocha Lopes - PPGEE - UFMA).

2.5 Programação Neurodinâmica

Durante várias décadas, a programação dinâmica clássica tem sido reconhecida como computacionalmente intratável para resolver problemas de decisão markovianos de larga escala por causa do enorme tamanho do espaço de estado que deve ser explorado. Sua complexidade computacional cresce proporcionalmente ao número de estados que por sua vez cresce exponencialmente com a dimensionalidade do espaço de estado (para uma quantização fixada), um problema conhecido como a "maldição da dimensionalidade de *Bellman*". A programação neurodinâmica usa aproximações por redes neurais para superar a "maldição da dimensionalidade", que tem sido um gargalo para a aplicação prática de programação dinâmica e controle estocástico para problemas complexos.

Uma estrutura paramétrica de aproximação envolve um vetor de parâmetros θ que deve ser sintonizado de modo a fornecer um melhor ajuste entre o mapeamento entrada/saída realizado pela rede neural e um dado conjunto de pares representativos $(x_k, V^*(x_k))$ de estados x_k e estimativas da função valor $V^*(x_k)$. O problema de aprendizagem ou treinamento é um problema de otimização tipica-

mente da forma de mínimos quadrados. Assim, dependendo do tipo de estrutura de rede neural, o objetivo é minimizar uma função de perda dada por

$$\sum_k \left(V^*(x_k) - \sum_i \varphi_i(x_k)\theta_i \right)^2,$$

ou sua versão não linear

$$\sum_k \left(V^*(x_k) - \widehat{V}(x_k, \theta) \right)^2.$$

Entretanto, no contexto dos problemas de programação dinâmica, tal conjunto de dados de treinamento não está disponível para treinar a rede neural. Uma possibilidade é utilizar métodos de simulação, tais como simulação de Monte Carlo ou método de diferenças temporais (Bertsekas e Tsitsiklis, 1996). A simulação possibilita o uso de métodos de programação neurodinâmica para projetar sistemas para os quais modelos explícitos não estão disponíveis. Para estes sistemas, as técnicas de programação dinâmica tradicionais são inaplicáveis, bem como é muito dispendioso se não impossível fornecer estimativas das probabilidades de transição de estados. A programação neurodinâmica permite que os sistemas aprendam sobre o seu comportamento por meio da simulação, e melhorem o seu desempenho por meio de reforço iterativo.

A função $\widehat{V}$ é chamada *função de escore* ou *função valor aproximada*. Em uma representação compacta típica, somente o vetor θ e a estrutura geral da função de escore $\widehat{V}(\cdot, \theta)$ são armazenados. Os escores $\widehat{V}(x_k, \theta)$ são gerados somente quando necessário. De acordo com (Bertsekas e Tsitsiklis, 1996), a função de escore $\widehat{V}(\cdot, \theta)$ geralmente envolve uma descrição dimensional menor do estado x_k em termos de suas "características significantes". A determinação da função de escore envolve duas escolhas importantes: a decisão sobre a estrutura de rede neural da função $\widehat{V}(\cdot, \theta)$ que se adpta ao problema em questão, e a escolha de um algoritmo de aprendizagem ou treinamento de forma a minimizar em algum sentido o erro entre as funções $V^*(\cdot)$ e $\widehat{V}(\cdot, \theta)$.

No quadro de iteração de política, aproximações podem ser introduzidas tanto na representação de funções valor quanto na representação das políticas. Considerando-se, por exemplo funções valor de ação, a representação tabular da função valor $Q^h(x, u)$ é substituída por uma aproximação paramétrica apropriada

$\widehat{Q}^h(x,u,\theta)$, em que θ é o vetor de parâmetros ajustáveis da aproximação e a representação tabular da política $h(x)$ é substituída por uma aproximação paramétrica $\widehat{h}(x,\kappa)$, em que κ é o vetor de parâmetros ajustáveis da aproximação. Apresenta-se a seguir um esboço geral do algoritmo de iteração de política aproximada com avaliação de política de função Q (Algoritmo 1).

Algoritmo 1 - Algoritmo de Iteração de Política Aproximada para Funções Q

1 ▷ - **Bloco 0 - Inicialização**
2 Selecione - Política Admissível - $\widehat{h}_0$.
3 Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
4 $j \leftarrow 0$
5 ▷ - **Processo Iterativo**
6 Para cada iteração j de política
7 **do**
8 ▷ - **Bloco 1 - Avaliação de Política**
9 Cálculo da Função Valor Aproximada $\widehat{Q}^{\widehat{h}_j}$
10 ▷ - **Bloco 2 - Melhoria de Política**
11 Estabeleça - Política Melhorada
12 $\widehat{h}_{j+1}(x) \approx \arg\max_u \widehat{Q}^{\widehat{h}_j}(x,u), \ \ \forall x \in X.$
13 **until** $\widehat{h}_{j+1}$ seja satisfatória
14 **then** Fim do Processo Iterativo
15 **Fim do Algoritmo 1**.

Uma questão natural é se a seqüência de políticas e funções valor geradas por um algoritmo de iteração de política aproximada converge para uma política e uma função valor próximas dos valores ótimos. Desde que os erros de avaliação e melhoria de política sejam limitados, a iteração de política aproximada *offline* produz políticas com uma subotimalidade limitada. Este resultado é formalizado pelo teorema a seguir, adaptado a partir de (Bertsekas e Tsitsiklis, 1996), considerando-se o caso em que as funções valor de ação são usadas.

Teorema 2.5.1. *Seja $\{\widehat{h}_j\}_{j=0}^{\infty}$ a sequência de políticas geradas por um algoritmo de iteração de política aproximada e seja $\{\widehat{Q}^{\widehat{h}_j}\}_{j=0}^{\infty}$ a sequência de funções valor aproximadas correspondentes. Sejam ϵ_Q e ϵ_h escalares positivos que limitam o erro em todas as aproximações (em todas as iterações) para funções valor e políticas, respectivamente. Se*

$$\left\| \widehat{Q}^{\widehat{h}_j} - Q^{\widehat{h}_j} \right\|_{\infty} \leq \varepsilon_Q, \qquad \forall\, j \geq 0, \tag{2.38}$$

e

$$\left\| T^{\widehat{h}_{j+1}}(\widehat{Q}^{\widehat{h}_j}) - T^*(\widehat{Q}) \right\|_\infty \leq \varepsilon_h, \qquad \forall\, j \geq 0. \tag{2.39}$$

[2] *Então, esta sequência eventualmente produz políticas cujo desempenho se encontra a uma distância limitada do ótimo:*

$$\limsup_{j\to\infty} \left\| \widehat{Q}^{\widehat{h}_j} - Q^* \right\|_\infty \leq \frac{\varepsilon_h + 2\gamma\varepsilon_Q}{(1-\gamma)^2}. \tag{2.40}$$

2.6 Aprendizagem TD(λ) com Aproximação da Função Valor

Métodos TD(λ) ajustam incrementalmente os coeficientes de $\widehat{V}^h(\cdot,\theta)$ para cada estado sobre cada trajetória observada em relação à novos valores objetivo. Os valores alvo dependem do parâmetro $0 \leq \lambda \leq 1$. No caso limite $\lambda = 1$, o alvo em cada estado visitado x_k é o retorno de *Monte Carlo*, isto é, a soma observada real de recompensas descontadas futuras $r_k + \gamma r_{k+1} + \ldots + \gamma^N r_N$. Isto é uma amostra não-polarizada de $\widehat{V}^h(x_k,\theta)$, mas tem variância significativa uma vez que ela depende de uma longa sequência estocástica de recompensas. No outro limite, $\lambda = 0$, o valor alvo é estabelecido por um *lookahead* à um passo amostrado $r_k + \gamma\widehat{V}^h(x_{k+1},\theta)$. Este valor têm variância menor, uma vez que o único componente aleatório é uma única transição de estado, mas é polarizado devido a inexatidão potencial da estimativa *lookahead* de V^h. O parâmetro λ pode ser visto como uma compensação entre bias e variâncias.

Vários estudos empíricos têm encontrado valores intermediários de λ para realizar melhor o processo de aprendizagem da função valor V^h. Por exemplo, (Sutton, 1988), (Sutton e Barto, 1998) sugerem que valores menores de λ contribuem na redução da variância das estimativas do parâmetro θ. Resultados sólidos têm

[2] T^h e T^* são operadores de *Bellman* definidos de acordo com as Eqs.(2.29) e (2.32), respectivamente, por

$$\left[T^h(Q)\right](x_k,u_k) = E_{x_{k+1}\sim\widetilde{f}(x_k,u_k,\cdot)}\left[r(x_k,u_k,x_{k+1}) + \gamma Q^h(x_{k+1},h(x_{k+1}))\right]$$
$$\left[T^*(Q)\right](x_k,u_k) = E_{x_{k+1}\sim\widetilde{f}(x_k,u_k,\cdot)}\left[r(x_k,u_k,x_{k+1}) + \gamma \max_{a\in U(x_k)} Q^*(x_{k+1},a)\right]$$

sido obtidos por (Tsitsiklis e Roy, 1997) que prova a convergência de TD(λ) com probabilidade 1 para problemas de decisão markovianos com espaço de estados discreto (possivelmente infinito), supondo-se uma sequência decrescente apropriada de tamanhos dos passos (*stepsizes*) e um aproximador linear de função no vetor de parâmetros ajustáveis θ.

O método TD(λ) atualiza θ_k após cada transição de estado de acordo com a equação

$$\theta_{k+1} = \theta_k + \alpha_k \Delta_k \sum_{i=0}^{k} (\gamma\lambda)^{k-i} \nabla \widehat{V}^h(x_i, \theta_k), \tag{2.41}$$

com θ inicializado para algum vetor arbitrário, α_k é um fator de aprendizagem, $\nabla \widehat{V}(x_i, \theta_k)$ é o vetor gradiente com respeito às componentes de θ_k e Δ_k é a diferença temporal entre a estimativa atual $\widehat{V}^h(x_k, \theta_k)$ de custo e a estimativa atualizada $r(x_k, u_k) + \gamma \widehat{V}^h(x_{k+1}, \theta_k)$ no estado x_k correspondente à transição de x_k para x_{k+1}, ou seja,

$$\Delta_k = r(x_k, u_k) + \gamma \widehat{V}^h(x_{k+1}, \theta_k) - \widehat{V}^h(x_k, \theta_k). \tag{2.42}$$

A Eq.(2.41) fornece um contínuo de algoritmos parametrizado por λ referido como TD(λ). No caso limite $\lambda = 0$, usando a convenção $0^0 = 1$, a Eq.(2.41) reduz-se à forma

$$\theta_{k+1} = \theta_k + \alpha_k \Delta_k \nabla \widehat{V}^h(x_k, \theta_k), \tag{2.43}$$

com $k = 0,\ 1,\ 2,\ \cdots$. Esta é a equação de atualização de aprendizagem da função valor V à um passo, referida como aprendizagem TD(0).

Uma representação viável para realização de aprendizagem *online* é obtida definindo-se uma sequência de vetores de elegibilidade ξ_k, de dimensão n_θ, por

$$\xi_k = \sum_{i=0}^{k} (\gamma\lambda)^{k-i} \nabla \widehat{V}^h(x_i, \theta_k), \tag{2.44}$$

sendo cada ξ_k uma função do conjunto inteiro de observações passadas até o passo de tempo k, que grava os gradientes passados e atuais da estimação da função valor.

No caso de aproximadores lineares de função, a função $\widehat{V}^h$ assume a forma dada por

$$\widehat{V}^h(x_k, \theta) = \varphi^T(x_k)\widehat{\theta}, \tag{2.45}$$

sendo $\varphi(x_k) = [\varphi_1(x_k)\ \varphi_2(x_k)\ \ldots\ \varphi_{n_\theta}(x_k)]^T$ o vetor de funções de base e $\widehat{\theta} = [\widehat{\theta}_1\ \widehat{\theta}_2\ \ldots\ \widehat{\theta}_{n_\theta}]^T$ o vetor de parâmetro estimado. Desta forma, o vetor de elegibilidade de Eq.(2.44), reduz-se à

$$\xi_k = \sum_{i=0}^{k} (\gamma\lambda)^{k-i}\varphi(x_i). \tag{2.46}$$

Com estas notações, as atualizações de TD(λ) são dadas por

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \alpha_k \Delta_k \xi_k, \tag{2.47}$$

e os vetores de elegibilidade podem ser atualizados de acordo com

$$\xi_{k+1} = \gamma\lambda\xi_k + \varphi(x_{k+1}), \quad \xi_{-1} = 0. \tag{2.48}$$

Da definição (2.44), observa-se que o vetor de elegibilidade no instante k depende da história da trajetória e do parâmetro λ, sendo $i = 0$ o instante em que a trajetória atual iniciou. No caso de TD(0), somente as características do estado atual são elegíveis a serem atualizadas, enquanto que em TD(1), as características de todos os estados visitados até o instante atual k sobre a trajetória corrente são elegíveis.

De acordo com (Bertsekas e Tsitsiklis, 1996), o limite de convergência θ^* existe e satisfaz à seguinte equação

$$\lim_{\theta \to \theta^*} (C\theta + d) = 0, \tag{2.49}$$

sendo $C = \mathrm{E}_0[C(X_k)]$, $d = \mathrm{E}_0[d(X_k)]$, $X_k = (x_k, x_{k+1}, \xi_{k+1})$ forma um processo markoviano, $k = 1, 2, \ldots$, $\mathrm{E}_0[\cdot]$ denota a esperança com respeito a única distribuição invariante de X_k, $C(X_k) = \xi_k(\gamma\varphi^T(x_{k+1}) - \varphi^T(x_k))$ e $d(X_k) = \xi_k r_k$.

ALGORITMO 2 - ALGORITMO DE AVALIAÇÃO DE POLÍTICA APROXIMADA USANDO-SE TD(λ)

▷ - **Inicialização**
Selecionar a política de controle h a ser avaliada.
Selecionar o fator de desconto, $0 < \gamma \leq 1$.
Selecionar as funções de base $\varphi_1, \cdots, \varphi_{n_\theta}$ e uma
sequência de stepsizes $\{\alpha_k\}_{k=0}^{\infty}$.
Escolher estado inicial x_0
$\theta_0 \leftarrow 0$
$\xi_0 \leftarrow 0$
$k \leftarrow 0$
▷ - **Processo Iterativo**
Para cada passo de tempo k
    do
        Para o estado atual x_k, observe a transição de estado
        de x_k para x_{k+1} e a recompensa imediata r_k.
        $\widehat{\theta}_{k+1} = \widehat{\theta}_k + \alpha_k \Delta_k \xi_k$
        $\xi_{k+1} = \gamma\lambda\xi_k + \varphi(x_{k+1})$
    until $\widehat{\theta}_{k+1}$ seja satisfatória
    then Fim do Processo Iterativo
Fim do Algoritmo 2.

2.7 Considerações Finais

Este capítulo forneceu um quadro de conceitos e métodos em aprendizagem por reforço formulado no contexto de processos de decisão markoviano e programação dinâmica. Dois problemas fundamentais em aprendizagem por reforço são o problema de avaliação de política e a síntese de política de decisão ótima. O capítulo revisou MDPs, definindo-se um problema de decisão sequencial ótima, em que decisões são tomadas em estágios de um processo que evolui ao longo do tempo. Na sequência, apresentou-se a programação dinâmica que é baseada em duas formas da equação de *Bellman*: a Eq.(2.11) da avaliação de política e a Eq.(2.14) da otimalidade. Associados à elas tem-se os métodos convencionais de iteração de política e iteração de valor que são métodos dependentes de modelo para resolver problemas de decisão sequencial ótima funcionando-se "para trás no tempo". A programação dinâmica tradicional é uma técnica de solução *offline* que não pode ser implementada *online* em uma maneira "para frente no tempo". Na sequência, o capítulo descreveu métodos de diferenças temporais que fornece 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 observados do sistema ao longo de sua trajetória.

Capítulo 3

Controle Ótimo

3.1 Introdução

Controle ótimo é uma abordagem de controle moderno que se propõe à determinação de estratégias de ação, ou leis de controle, que otimizem uma métrica de desempenho desejado. Um conceito fundamental em tal abordagem é aquele de uma função de critério ou função de utilidade a qual incorpora os requisitos de projeto para o sistema controlado e para o contexto de problema. A escolha de uma função de critério caracteriza uma etapa crítica para o projeto de controladores ótimos, uma vez que ela é a única fonte de informação disponível para o processo de projeto (automatizado) para os objetivos do controlador que ele projetar, e mais ainda, devido aos atributos do controlador resultante e a qualidade de desempenho estarem intrinsecamente relacionados com tal função.

Em geral, a formulação de uma função de utilidade envolve um compromisso entre uma avaliação significativa do desempenho do sistema e um problema matemático tratável. Uma função de custo total ou função valor, definida em termos da função de utilidade, é usada para realizar o processo de otimização. Especificações de desempenho podem envolver energia mínima para controlar o sistema dinâmico, tempo mínimo para atingir um dado estado de operação, entre outras. O sistema que é o resultado final de um projeto ótimo não é meramente suposto ser estável, ter uma determinada resposta transitória, largura de banda, ou satisfazer qualquer uma das restrições associadas com controle clássico (rejeição de perturbação, e robustez à variações da planta ou incertezas), mas é suposto ser o

melhor sistema possível de um tipo particular.

Uma abordagem para solução de problemas de controle ótimo é baseada na teoria conhecida como cálculo variacional que deduz condições necessárias para otimalidade na forma das equações de Euler-Lagrange. Uma alternativa à abordagem variacional para solução de controle ótimo surgiu no final da década de 50 com Richard Bellman, referida como programação dinâmica, (Bellman, 1957), (Bellman e Dreyfus, 1962), (Bellman e Kalaba, 1965). Uma papel fundamental no desenvolvimento desta abordagem é desempenhado pelo princípio da otimalidade de *Bellman*: *Uma trajetória ótima possui a propriedade que em um ponto intermediário, independente da forma de como este ponto é alcançado, a trajetória restante também é ótima, conforme planejada e tendo o ponto intermediário como ponto de partida.*

A programação dinâmica fornece um formalismo matemático para tratar problemas relativos à inconsistência de tempo e problemas estocásticos sob condições muito amplas que são difíceis de lidar usando-se o cálculo variacional. Sua aplicação à problemas de controle ótimo tem implicado tratamentos para sistemas tanto a tempo contínuo quanto a tempo discreto, sistemas invariantes no tempo ou variantes no tempo, sistemas lineares ou não-lineares. O princípio da otimalidade de *Bellman* desempenha um papel similar à aquele desempenhado pelo princípio do mínimo de *Pontryagin* na abordagem variacional, e serve para limitar, potencialmente, o número de estratégias de controle ótimo que devem ser investigadas. Sua importância está no fato de que viabiliza a otimização sobre somente um vetor de controle de cada vez. As referências para este tópico incluem (Lewis *et al.,* 2012), (Kirk, 1970), (Bryson e Ho, 1975) e (Athans e Falb, 1966).

O presente capítulo enfatiza problemas de controle ótimo sob a óptica de programação dinâmica. Inicia-se o capítulo com uma formulação geral do problema de controle ótimo, e na sequência, são abordados os problemas quadráticos lineares, dando uma atenção especial ao problema do Regulador Linear Quadrático. Apresenta-se a descrição em espaços de estados por modelos de tempo discreto.

3.2 O Problema de Controle Ótimo

Considere um sistema dinâmico descrito por uma equação de estado não-linear e variante no tempo

$$x_{k+1} = f(x_k, u_k, k), \tag{3.1}$$

sendo $x_k \in \Re^n$ o vetor de estado, $u_k \in \Re^{n_e}$ o vetor de entrada de controle, e o estado inicial x_i é dado.

Uma abordagem razoavelmente geral de controle ótimo consiste na determinação de uma lei de controle u_k sobre o intervalo $[i, N]$ que minimiza o índice de desempenho que é dado por

$$V(x_i, u_{(\cdot)}, i) = \psi(x_N, N) + \sum_{k=i}^{N-1} \ell(x_k, u_k, k), \tag{3.2}$$

tendo como restrição o sistema dinâmico de Eq.(3.1). A função $\psi(\cdot)$ penaliza o estado final x_N no horizonte de tempo $k = N$, e a função $\ell(\cdot)$ penaliza os estados x_k e as entradas de controle u_k de $k = i$ até $k = N - 1$.

Uma abordagem para otimização da Eq.(3.2) sujeita à Eq.(3.1) é baseada na teoria de multiplicadores de Lagrange (Cálculo Variacional). Desta forma, defina inicialmente um vetor adjunto (ou co-estado) λ_k e a função de custo estendida V_e

$$V_e = \psi(x_N, N) + \sum_{k=i}^{N-1} \big[\ell(x_k, u_k, k) + \lambda_{k+1}^T (f(x_k, u_k, k) - x_{k+1})\big]. \tag{3.3}$$

Defina agora a função de Hamilton

$$H(x_k, u_k, \lambda_k, k) = \ell(x_k, u_k, k) + \lambda_k^T f(x_k, u_k, k). \tag{3.4}$$

Pode-se, então, escrever

$$V_e = \psi(x_N, N) - \lambda_N^T x_N + H(x_i, u_i, i) + \sum_{k=i+1}^{N-1} \big[H(x_k, u_k, k) + \lambda_k^T x_k)\big]. \tag{3.5}$$

Suponha agora que as funções f e ℓ sejam suficientemente diferenciáveis. Denote por dV_e o incremento em V_e devido aos incrementos em todas as variáveis independentes x_k, λ_k, e u_k. De acordo com a teoria de multiplicadores de Lagrange, o mínimo restrito de V é alcançado no mínimo irrestrito de V_e, isto é, quando $dV_e = 0$. Portanto,

$$\begin{aligned} dV_e = &\left(\tfrac{\partial \psi}{\partial x_N} - \lambda_N\right)^T dx_N + \left(\tfrac{\partial H_i}{\partial x_i}\right)^T dx_i + \left(\tfrac{\partial H_i}{\partial u_i}\right)^T du_i \\ &+ \sum_{k=i+1}^{N} \left[\left(\tfrac{\partial H_k}{\partial x_k} - \lambda_k\right)^T dx_k + \left(\tfrac{\partial H_k}{\partial u_k}\right)^T du_k\right] \\ &+ \sum_{k=i+1}^{N} \left(\tfrac{\partial H_{k-1}}{\partial \lambda_k} - x_k\right)^T d\lambda_k, \end{aligned} \tag{3.6}$$

sendo $H_k \triangleq H(x_k, u_k, \lambda_k, k)$. Consequentemente, condições necessárias são dadas por

$$\frac{\partial H_k}{\partial \lambda_{k+1}} = x_{k+1}, \tag{3.7}$$

$$\frac{\partial H_k}{\partial x_k} = \lambda_k, \tag{3.8}$$

$$\frac{\partial H_k}{\partial u_k} = 0, \tag{3.9}$$

$$\left(\frac{\partial \psi}{\partial x_N} - \lambda_N\right)^T dx_N = 0, \tag{3.10}$$

$$\left(\frac{\partial H_i}{\partial x_i}\right)^T dx_i = 0, \tag{3.11}$$

com $k = i, \ldots, N-1$.

As condições (3.7)-(3.9) decorrem dos termos de dentro dos somatórios e do coeficiente de du_i que juntamente com as condições (3.10) e (3.11) definem um problema de valor de contorno de dois pontos para o horizonte de tempo N fixo.

A solução deste problema constitue um dos principais tópicos de interesse na teoria de controle ótimo. No caso mais geral, o qual inclui restrições às variáveis de controle, condições necessárias para otimalidade de soluções de problemas de controle ótimo correspondem ao teorema conhecido por princípio do mínimo de *Pontryagin*, declarando que sobre a trajetória ótima, a função de Hamilton satisfaz a condição de mínimo

$$H(x_k^*, u_k^*, \lambda_k^*, k) \leq H(x_k^*, u_k, \lambda_k^*, k), \tag{3.12}$$

para todo u admissível, sendo x^* e λ^* valores ótimos de estado e co-estado, respectivamente.

3.2.1 Solução via Programação Dinâmica

A solução ao problema de controle ótimo é apresentada nas formas da Equação de *Hamilton-Jacobi-Bellman*, uma abordagem apoiada no princípio da otimalidade de *Bellman*.

Sejam várias trajetórias resultantes de diferentes controles, todas iniciando-se no estado x_k no instante de tempo k. Seja $V^*(x_k, k)$ o índice de desempenho ótimo associado com um estado inicial x_k no instante k. Supõe-se que são conhecidos, para todos os estados possíveis x_{k+1}, o custo ótimo $V^*(x_{k+1}, k+1)$ do instante $k+1$ ao instante terminal N e as sequências de controle ótimo de realimentação do instante $k+1$ para N. O índice de desempenho ótimo é obtido quando a sequência de controle ótimo u_{k+1}^*, u_{k+2}^*, ..., u_{N-1}^* é aplicada à planta com um estado x_{k+1}. Se um controle arbitrário u_k for aplicado no instante k, e, então, a sequência de controle ótimo a partir de $k+1$ for usada, o custo incorrido no instante k é dado por

$$\ell(x_k, u_k, k) + V^*(x_{k+1}, k+1). \tag{3.13}$$

De acordo com o princípio da otimalidade de *Bellman*, o custo ótimo para trajetórias iniciando-se no instante k e terminando-se no instante N é incorrido minimizando-se a soma do custo na transição de x_k para x_{k+1} e o custo ótimo daí em diante, isto é,

$$V^*(x_k, k) = \min_{u_k}\{\ell(x_k, u_k, k) + V^*(x_{k+1}, k+1)\}. \tag{3.14}$$

O controle ótimo u_k^* no instante k é dado por u_k que alcança este mínimo. A Eq.(3.14) é a equação de *Hamilton-Jacobi-Bellman* para sistemas de tempo discreto.

3.3 Controle Ótimo Linear Quadrático

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*. A planta controlada supõe-se ser linear e o índice de desempenho associado é quadrático nas variáveis de controle e variáveis de erro de regulação/rastreamento. Tais métodos conduzem à leis de controle ótima lineares.

Uma importante vantagem das metolodogias de controle ótimo especificamente do tipo linear quadrático é o fato de que nos projetos de controle onde se supõe que todos os estados da planta são mensuráveis e disponíveis para realimentação acabam por possuírem um número de propriedades, outras do que simplesmente otimalidade de um índice de desempenho, as quais o controle clássico sugere serem atrativas. Exemplos de tais propriedades são boa margem de ganho e margem de fase, e boa tolerância à não linearidades. Um caso especial de controle LQ que exibe boas propriedades refere-se à metodologia denominada regulador linear quadrático (LQR), a qual promove robustez garantida: margem de ganho infinitamente crescente, margem de fase entre $\pm 60^\circ$ e boa tolerância à não-linearidades, (Maciejowski, 1989). Tais propriedades de robustez podem ser frequentemente alcançadas mesmo quando a estimação de estado for necessária por meio da metodologia LQG/LTR.

Na maioria dos projetos de controle baseados em controle ótimo linear quadrático, as matrizes de ponderações que constituem os índices de desempenho são parâmetros de projeto. Essas matrizes precisam ser selecionadas pelo projetista na tentativa de impor as características de desempenho desejadas do sistema, tal como autoestrutura desejada, projetando sistemas que particularmente apresentam boas características de estabilidade e desempenho, (Stein, 1979), (Medanic

et al., 1988), (Kawasaki e Shimemura, 1983),(Graupe, 1972) e (Harvey e Stein, 1978). Essa seleção é uma das dificuldades do projeto LQ, pois tais matrizes normalmente são determinadas por métodos de tentativa e erro, ou relações analíticas que se adequam somente à sistemas SISO e de baixa ordem, (Johnson e Grimble, 1987). Contudo, as referências (Liu, 1998), (Bottura e Fonseca Neto, 1999), (Fonseca Neto, 2000), (Ferreira *et al.,* 2003) e (Fonseca Neto *et al.,* 2008) apresentam uma alternativa para superar esta dificuldade que são procedimentos baseados em computação evolutiva.

3.4 O Problema LQ Discreto

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

$$x_{k+1} = A_k x_k + B_k u_k, \tag{3.15}$$

com estado inicial x_i dado, x_k é o vetor de estado de dimensão n e u_k é o vetor de entrada de controle de dimensão n_e. A lei $u_{(.)}$ para controlar o sistema de Eq.(3.15) é estabelecida para tentar fazê-lo tão próximo quanto possível de uma trajetória desejada de estado, a qual é dada por uma função prescrita $\widetilde{x}_{(.)}$ do tempo, em um intervalo de tempo especificado $[i, N]$ levando em consideração a energia mínima de controle utilizada. Dessa forma, o problema LQ é caracterizado como uma estrutura de otimização com o objetivo de determinar uma lei de controle u_k, $t \in [t_0, N]$, que minimiza o índice de desempenho (função de custo) dado por

$$\begin{aligned} V(x_i, u_{(.)}, i) &= (x_N - \widetilde{x}_N)^T S (x_N - \widetilde{x}_N) + \\ &+ \sum_{k=i}^{N-1} \left[(x_k - \widetilde{x}_k)^T Q_k (x_k - \widetilde{x}_k) + u_k^T R_k u_k \right] \end{aligned} \tag{3.16}$$

e que tenha como restrição dinâmica o sistema de Eq.(3.15). As matrizes $Q = Q^T \geq 0$ e $R = R^T > 0$, com dimensões apropriadas, representam ponderações do erro entre o estado $x_{(.)}$ e a trajetória desejada $\widetilde{x}_{(.)}$, e do controle, respectivamente.

A matriz $S = S^T \geq 0$ pondera o desvio de estado final no instante N. O problema definido pelas Eqs.(3.15) e (3.16) 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 à propriedades de estabilidade e detectabilidade do sistema. Para mais discussão sobre este tópico, veja (Kirk, 1970), (Bryson e Ho, 1975) e (Anderson e Moore, 1990).

3.4.1 O Problema do Regulador Linear Quadrático Discreto e a Equação Algébrica de *Riccati* Discreta

O problema LQR 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 dado por

$$V_i = \frac{1}{2}x_N^T P_N x_N + \frac{1}{2}\sum_{k=i}^{N-1}(x_k^T Q_k x_k + u_k^T R_k u_k), \tag{3.17}$$

e tem como restrição a equação de estado (3.15), sendo $[i, N]$ o intervalo de tempo de interesse, $Q = Q^T \geq 0$ e $R = R^T > 0$ são matrizes de ponderações do estado e do controle, respectivamente, e $P_N = P_N^T \geq 0$ é uma matriz que pondera o estado final, todas com dimensões apropriadas.

Observa-se a partir do índice de desempenho (3.17) que a trajetória desejada $\widetilde{x}$ é simplesmente o estado nulo. Especificamente, a lei de controle ótima deve conduzir a planta de um estado não-nulo x_i no instante i para o estado nulo no instante N levando em consideração a energia mínima de controle.

Para começar, seja $k = N$ e escreva

$$V_N^* = \frac{1}{2}x_N^T P_N x_N, \tag{3.18}$$

que é a penalidade para começar no estado x_N no instante N.

Agora, reduz-se k para $N-1$ e escreva

$$V_{N-1} = \frac{1}{2}x_{N-1}^T Q_{N-1} x_{N-1} + \frac{1}{2}u_{N-1}^T R_{N-1} u_{N-1} + \frac{1}{2}x_N^T P_N x_N. \tag{3.19}$$

De acordo com Eq.(3.14), é preciso se determinar u_{N-1}^* por minimização da Eq.(3.19). Dessa forma, usando-se a equação de estado, tem-se

$$\begin{aligned} V_{N-1} = {} & \tfrac{1}{2}x_{N-1}^T Q_{N-1} x_{N-1} + \tfrac{1}{2}u_{N-1}^T R_{N-1} u_{N-1} \\ & + \tfrac{1}{2}(A_{N-1}x_{N-1} + B_{N-1}u_{N-1})^T P_N \left(A_{N-1}x_{N-1} + B_{N-1}u_{N-1}\right). \end{aligned} \tag{3.20}$$

Uma vez que não existem restrições, o mínimo de V_{N-1} é encontrado estabelecendo-se

$$0 = \frac{\partial V_{N-1}}{\partial u_{N-1}} = R_{N-1}u_{N-1} + B_{N-1}^T P_N (A_{N-1}x_{N-1} + B_{N-1}u_{N-1}). \tag{3.21}$$

Resolvendo-se para o controle ótimo, tem-se

$$u_{N-1}^* = -\left(B_{N-1}^T P_N B_{N-1} + R_{N-1}\right)^{-1} B_{N-1}^T P_N A_{N-1} x_{N-1}. \tag{3.22}$$

Definindo

$$K_{N-1} = \left(B_{N-1}^T P_N B_{N-1} + R_{N-1}\right)^{-1} B_{N-1}^T P_N A_{N-1}, \tag{3.23}$$

pode-se escrever

$$u_{N-1}^* = -K_{N-1}x_{N-1}. \tag{3.24}$$

O custo ótimo no instante $k = N-1$ é encontrado substituindo-se Eq.(3.24) em Eq.(3.20) que após simplificações, resulta

$$\begin{aligned} V_{N-1}^* = {} & \tfrac{1}{2}x_{N-1}^T \Big[(A_{N-1} - B_{N-1}K_{N-1})^T P_N \left(A_{N-1} - B_{N-1}K_{N-1}\right) \\ & + K_{N-1}^T R_{N-1} K_{N-1} + Q_{N-1}\Big] x_{N-1}. \end{aligned} \tag{3.25}$$

Definindo

$$\begin{aligned} P_{N-1} = (A_{N-1} - B_{N-1}K_{N-1})^T P_N (A_{N-1} - B_{N-1}K_{N-1}) \\ + K_{N-1}^T R_{N-1} K_{N-1} + Q_{N-1}, \end{aligned} \tag{3.26}$$

a Eq.(3.25) pode ser escrita como

$$V_{N-1}^* = \frac{1}{2} x_{N-1}^T P_{N-1} x_{N-1}. \tag{3.27}$$

Agora reduz-se para $k = N-2$. Então

$$V_{N-2} = \frac{1}{2} x_{N-2}^T Q_{N-2} x_{N-2} + \frac{1}{2} u_{N-2}^T R_{N-2} u_{N-2} + \frac{1}{2} x_{N-1}^T P_{N-1} x_{N-1} \tag{3.28}$$

são os custos admissíveis para $N-2$, uma vez que estes são os custos que são ótimos a partir de $k = N-1$. Para determinar u_{N-2}^*, de acordo com (3.14), minimiza-se (3.28).

Observe que (3.28) é da mesma forma que (3.19). O controle ótimo e o custo ótimo no instante $k = N-2$ são portanto dados por (3.23), (3.24), (3.26) e (3.27) com N substituído por $N-1$.

Continuando-se a reduzir k e aplicando-se o princípio da otimalidade, o resultado para cada k, $k = N-1, \cdots, 1, 0$ é

$$K_k = \left(B_k^T P_{k+1} B_k + R_k\right)^{-1} B_k^T P_{k+1} A_k, \tag{3.29}$$

$$u_k^* = -K_k x_k, \tag{3.30}$$

$$P_k = (A_k - B_k K_k)^T P_{k+1} (A_k - B_k K_k) + K_k^T R_k K_k + Q_k, \tag{3.31}$$

$$V_k^* = \frac{1}{2} x_k^T P_k x_k, \tag{3.32}$$

sendo a condição final P_N para (3.31) é dada em (3.17). Esta formulação implica que estratégias de controle ótimo devem ser determinadas recursivamente no tempo a partir do estágio final (procedimento "para trás no tempo").

Uma observação importante refere-se à Eq.(3.31). Se o ganho K_k é dado por uma sequência arbitrária de ganhos estabilizantes, então (3.31) é simplesmente uma equação de *Lyapunov*, a qual é linear em P_k. Por outro lado, se a sequência de ganhos ótimos de Eq.(3.29) for aplicada na Eq.(3.31), esta torna-se a equação matricial de *Riccati* em sua versão estabilizada de *Joseph*. Não é difícil verificar que, substituindo-se (3.29) em (3.31) e após tranformações, obtém-se a forma

$$P_k = Q_k + A_k^T P_{k+1} A_k - A_k^T P_{k+1} B_k (R_k + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k. \tag{3.33}$$

Esta equação é uma declaração da equação HJB discreta para o caso DLQR.

Um caso particular do problema 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, isto é, $(N-k) \to \infty$. Neste caso, condições de existência e estabilidade para a solução do regulador podem ser garantidas supondo-se que o par (A, B) é estabilizável e (A, C) é completamente observável para qualquer $C^T C = Q$. Então, para $N \to \infty$, tem-se que $\overline{P} \triangleq P_{k+1} = P_k$ é constante e é a única solução definida positiva da equação

$$P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A, \tag{3.34}$$

que é equação algébrica de *Riccati* de tempo discreto (*Discrete Algebraic Riccati Equation* - DARE), também referida neste texto como equação HJB-*Riccati*. Além disso, o sistema de malha fechada

$$x_{k+1} = (A - B K_\infty) x_k \tag{3.35}$$

é assintoticamente estável, sendo K_∞ o ganho em regime permanente que é dado por

$$K_\infty = (B^T P B + R)^{-1} B^T P A. \tag{3.36}$$

3.5 Considerações Finais

Neste capítulo, discutiu-se brevemente a abordagem variacional e o princípio do mínimo de *Pontryagin* para solução de problemas de controle ótimo. Na sequência apresentou-se os formalismos relacionados à programação dinâmica como uma abordagem alternativa ao cálculo variacional. Ênfase foi dada ao princípio da otimalidade de *Bellman* e à obtenção da equação de *Hamilton-Jacobi-Bellman* para o problema do regulador linear quadrático. Entretanto, um importante aspecto limitador da programação dinâmica é que sua aplicação não tem sido computacionalmente tratável do ponto de vista de implementações *online*, uma vez que é um método "para trás no tempo". Na maioria das vezes é necessário uma grande quantidade de requisitos computacionais e armazenagem para obter a solução da equação de *Hamilton-Jacobi-Bellman* associada. Para superar esta dificuldade, métodos de programação dinâmica aproximada/adaptativa têm sido aplicados para obter a solução *online* da equação HJB indiretamente, usando-se uma estrutura de função tal como redes neurais para aproximar a função de custo.

Capítulo 4

Programação Dinâmica Heurística Dependente de Estado e Ação para Soluções Aproximadas da Equação HJB-*Riccati*

4.1 Introdução

Nos últimos anos, muito tem sido feito para o desenvolvimento de soluções aproximadas da equação HJB por meio de métodos conhecidos como programação dinâmica aproximada/adaptativa, objetivando encontrar soluções *online* para problemas de controle ótimo. Por exemplo, uma prova de convergência de um algoritmo de programação dinâmica heurística baseado em iteração de valor para resolver a equação HJB de tempo discreto do problema de controle ótimo para sistemas não-lineares de tempo discreto é fornecida em (Al-Tamimi e Lewis, 2007), (Al-Tamimi *et al.,* 2008), e os autores (Al-Tamimi *et al.,* 2007*b*) deduzem um método "para frente no tempo"para resolver a equação de *Riccati* do problema de controle ótimo H_∞ de tempo discreto via programação dinâmica heurística e programação heurística dual.

Uma abordagem de redes neurais para solução aproximada da equação HJB

para uma função valor baseada em funcionais não-quadráticos é apresentada por Abu-Khalaf e Lewis (Abu-Khalaf e Lewis, 2005) para o projeto de controladores de realimentação quase-ótima com entradas restritas para atuadores saturados. Chen e Jagannathan (Chen e Jagannathan, 2008) apresentam um método por aproximações sucessivas de mínimos quadrados baseado em redes neurais para solução da equação HJB generalizada para o projeto de controladores quase-ótimos de sistemas não-lineares de tempo discreto sob a suposição de pequenas perturbações. Cheng e outros (Cheng *et al.,* 2007) fornecem resultados de convergência para controle ótimo de tempo final fixo de sistemas não-lineares baseado em redes neurais para resolver equações HJB com funções de custo variantes no tempo. Em (Jin *et al.,* 2007), os autores apresentam uma abordagem ADP para sistemas não-lineares de tempo discreto baseada em redes neurais WBF (*wavelet base funtion*) que fornece velocidade de treinamento mais rápido quando comparada com redes RBF (*radial base function*).

Uma estratégia para programação dinâmica adaptativa que combina técnicas de computação *soft* e computação *hard* para aprender uma lei de controle ótima em tempo real para um sistema não-linear estabilizável com dinâmicas desconhecidas é proposta por Murray e outros (Murray *et al,* 2002). Em (Liu *et al.,* 2001), os autores propõem uma abordagem crítica adaptativa dependente de ação direcionada para aplicações de controle de aprendizagem *online* utilizando redes neurais. Si e Wang (Si e Wang, 2001) fornecem um tratamento sistemático para desenvolver um sistema de controle de aprendizagem *online* em ambientes estocásticos e não-lineares para programação dinâmica neural livre de modelo. Liu e Balakrishnan (Liu e Balakrishnan, 2000) estabelecem condições necessárias de convergência para solução ótima de um crítico adaptativo consistindo de duas redes neurais as quais fornecem valores de controle e multiplicadores de Lagrange associados com controle ótimo. Outros autores, tais como Landelius (Landelius, 1997) apresenta uma análise de convergência para vários projetos críticos adaptativos para sistemas de controle LQR, e Lendaris (Lendaris e Paintz, 1997) discute vários procedimentos para a metodologia de programação dinâmica heurística dual e investiga seu desempenho com respeito à velocidade de convergência e qualidade de projeto de controlador resultante.

Em 1977, Werbos (Werbos, 1977) introduziu uma abordagem para ADP que

no início da década de 90 foi chamada Projetos Críticos Adaptativos (ACDs), (Werbos, 1992), (Werbos, 1990*b*). A família de métodos ACDs inclui: Programação Dinâmica Heurística (HDP), Programação Heurística Dual (DHP), e Programação Heurística Dual Global (GDHP). Existem versões chamadas críticas dependentes de ação de cada uma delas, nominalmente, ADHDP (também conhecida como Aprendizagem Q, (Watkins, 1989), (Watkins e Dayan, 1992), ADDHP e ADGDHP. Os ACDs são fortemente enraizados no campo de controle ótimo baseado em aprendizagem por reforço e surgem sob a categoria de métodos RL TD ator-crítico.

Na literatura, existem vários sinônimos usados para "Projetos Críticos Adaptativos", (Liu *et al.,* 2001), (Prokhorov e Wunsch, 1997), (Balakrishnan e Biega, 1996), (Dalton e Balakrishnan, 1996), (Javaherian *et al.,* 2004), (Kulkarni e KrishnaKumar, 2003), (Prokhorov *et al.,* 1995), (Venayagamoorthy *et al.,* 2002), incluindo: Programação Dinâmica Aproximada, (Powell, 2007), (Si *et al.,* 2004), (Werbos, 1992), Programação Dinâmica Assintótica, (Saeks *et al.,* 1997), Programação Dinâmica Adaptativa, (Murray *et al,* 2002), (Murray *et al.,* 2003), Programação Dinâmica Heurística, (Lendaris e Paintz, 1997), (Werbos, 1990*a*), Programação Neuro-Dinâmica, (Bertsekas e Tsitsiklis, 1996), Programação Dinâmica Neural, (Si *et al.,* 2004), (Yang *et al.,* 2003), Aprendizagem por Reforço, (Sutton e Barto, 1998).

Prokhorov e Wunsch em (Prokhorov e Wunsch, 1997), apresentaram mais algoritmos de acordo com os ACDs e discutiram as famílias de projetos de HDP, DHP e GDHP. Eles sugeriram algumas novas melhorias ao projeto original de GDHP úteis para muitas aplicações de engenharia nas áreas de otimização e controle ótimo. Baseada em uma dessas modificações, apresentaram uma abordagem unificada para todos os ACDs. Isto levou a um procedimento de treinamento generalizado para ACDs. Trabalhos posteriores realizados por Werbos, (Werbos, 2004), (Werbos, 2007), (Werbos, 2009), (Werbos, 2008), têm tomado mais ainda as ideias de ADP.

Todas as abordagens críticas adaptativas de Werbos podem realizar a mesma função que é obter a política de controle ótima e são compostas por uma rede, chamada rede de ação ou ator, que representa um mapeamento entre as variáveis de estado de um sistema dinâmico e as variáveis de controle. Uma diferença

entre as abordagens reside nas entradas e saídas de uma segunda rede, chamada o crítico. Na estrutura HDP o crítico fornece uma aproximação da função valor V_k. Em DHP a rede crítica fornece uma aproximação do gradiente de V_k. Na GDHP, o crítico aproxima tanto V_k quanto seu gradiente. Nessas abordagens as entradas do crítico tipicamente são as variáveis de estado. Nas estruturas dependentes de ação, incluem-se também as variáveis de controle como entradas para a rede crítica. Dependendo da estrutura ADP, não é necessário um modelo da planta para o treinamento de uma ou ambas as redes. Por exemplo, em paradigmas de HDP algum tipo de modelo de sistema dinâmico é necessário para determinação das políticas de controle, mas para se treinar a rede crítica não é requerido tal modelo. Por outro lado, em paradigmas de ADHDP a regra de decisão (política de controle) é completamente independente de um modelo do sistema dinâmico.

Neste capítulo, a programação dinâmica aproximada, denominada crítica adaptativa, é apresentada levando em consideração a programação dinâmica heurística e programação dinâmica heurística dependente de ação que são orientadas para a realização *online* de controle ótimo, especificamente do tipo linear quadrático discreto. O objetivo principal é aproximar soluções de programação dinâmica usando-se métodos de aproximação e esquemas de iteração de política para aproximar a solução da equação HJB-*Riccati*. Os elementos necessários para o desenvolvimento dos métodos HDP e ADHDP, tais como processo de decisão markoviano, programação dinâmica, diferenças temporais, iteração de política e iteração gulosa, foram apresentados no Capítulo 2. Apesar desses elementos fornecerem um tratamento para sistemas dinâmicos estocásticos, as formulações serão apresentadas em um contexto determinístico. Aqui a função densidade de probabilidade $\widetilde{f} : X \times U \times X \to [0, \infty)$ será substituída por uma função de transição de estados determinística dada por $f : X \times U \to X$; $f(x_k, u_k) = x_{k+1}$, e a função de utilidade $r : X \times U \times X \to \Re$; $r(x_k, u_k, x_{k+1}) = r_k$ será substituída por $r : X \times U \to \Re$; $r(x_k, u_k) = r_k$.

4.2 Programação Dinâmica Heurística

A programação dinâmica heurística (HDP) é estabelecida por um conjunto de metodologias para o desenvolvimento de procedimentos que estimam a função

valor V. A estimação da função valor V para uma dada política requer somente amostras da função de recompensa instantânea r, enquanto os modelos do ambiente e da recompensa instantânea são necessários para determinar a função valor V correspondente à política ótima.

Considere a versão determinística da equação de *Bellman* (2.11) apresentada no Capítulo 2, a qual é dada pela seguinte equação

$$\begin{aligned} V^h(x_k) &= r(x_k, h(x_k)) + \gamma V^h(x_{k+1}) \\ &= r(x_k, h(x_k)) + \gamma V^h(f(x_k, h(x_k))). \end{aligned} \tag{4.1}$$

O aprendizado supervisionado pode ser introduzido na Eq.(4.1) usando-se um esquema iterativo que tem V^h aproximado por modelos parametrizados. Em sua forma geral, esses modelos tem como ponto de partida o modelo parametrizado dado por

$$\Theta(r, f, \theta) = r(x, h(x)) + \gamma V(f(x, h(x)), \theta), \tag{4.2}$$

sendo θ o vetor de parâmetros da aproximação. Este vetor deve minimizar a média do quadrado do erro entre o valor estimado $V(x, \theta)$ e o valor medido $\Theta(\cdot)$, produzindo-se um novo vetor que é dado por

$$\theta_{k+1} = \arg\min_{\theta} E\{|V(x, \theta) - \Theta(r, f, \theta_k)|^2\}, \tag{4.3}$$

sendo que $E(\cdot)$ denota o valor esperado.

Para se determinar os parâmetros correspondentes à função valor ótima é necessário usar-se esquemas de iteração de política e uma parametrização da política $h(x) = h(x, \kappa)$. O valor apropriado do parâmetro κ é determinado de acordo com a equação da otimalidade de *Bellman*, e baseado na estimativa de V. O parâmetro ótimo k^* é dado por

$$\kappa^* = \arg\max_{\kappa} \{r(x, h(x, \kappa)) + \gamma V(f(x, h(x, \kappa)), \theta)\}. \tag{4.4}$$

Os parâmetros ótimos de Eq.(4.4) são obtidos por meio da solução da equação do gradiente com respeito à κ que é dada por

$$\frac{\partial V}{\partial \kappa} = 0. \tag{4.5}$$

Claramente, as derivadas dos modelos parametrizados f, r, V e h são necessárias para determinar o gradiente $\frac{\partial V}{\partial \kappa}$. Dessa forma, o máximo na Eq.(4.4) deve satisfazer

$$\frac{\partial r}{\partial h}\frac{\partial h}{\partial \kappa} + \gamma \frac{\partial V}{\partial f}\frac{\partial f}{\partial h}\frac{\partial h}{\partial \kappa} \;\; = \;\; 0. \tag{4.6}$$

4.3 Programação Dinâmica Heurística Dependente de Ação

A programação dinâmica heurística dependente de ação é baseada sobre a abordagem de aprendizagem Q, a qual será denotada por Q_L, e consiste de um método para estimar a função Q_L para qualquer política ótima ou não ótima. A formulação ADHDP requer somente amostras da função de recompensa instantânea r.

A abordagem ADHDP é apresentada aqui em duas instâncias. Na primeira, a aprendizagem Q_L é formulada em termos da equação de *Bellman* e formas de política de decisão ótima. Na segunda instância, aborda-se a aproximação da função Q_L para obter a política de controle ótima.

4.3.1 Aprendizagem Q_L

A função Q_L, ou função valor de ação, é definida por $Q_L^h : X \times U \rightarrow \Re$,

$$Q_L^h(x_k, u_k) = r(x_k, u_k) + \gamma V^h(f(x_k, u_k)). \tag{4.7}$$

A partir desta definição e da Eq.(4.1), observa-se que $Q_L^h(x_k, h(x_k)) = V^h(x_k)$, e, portanto, a função Q_L pode ser expressa na forma de *Bellman* como

$$Q_L^h(x_k, u_k) = r(x_k, u_k) + \gamma Q_L^h(f(x_k, u_k), h(f(x_k, u_k))). \tag{4.8}$$

Assim, a função Q_L ótima deve satisfazer

$$\begin{aligned} Q_L^*(x_k, u_k) &= r(x_k, u_k) + \gamma V^*(x_{k+1}) && (4.9)\\ &= r(x_k, u_k) + \gamma Q_L^*(x_{k+1}, h^*(x_{k+1})). && (4.10) \end{aligned}$$

Esta formulação permite que o problema de decisão markoviano seja resolvido independente dos modelos f and r. Consequentemente, a política ótima é dada por

$$h^*(x_k) = \arg\max_u Q_L^*(x_k, u). \tag{4.11}$$

4.3.2 Aproximações da Função Q_L

O lado direito da Eq.(4.8) é usado como objetivo para aproximar Q_L por um modelo parametrizado, o qual é dado por

$$\Theta(r, f, h(f), \theta) = r(x, u) + \gamma Q_L(f(x, u), h(f(x, u)), \theta), \tag{4.12}$$

$$\theta_{k+1} = \arg\min_\theta E\left\{|Q_L(x, u, \theta) - \Theta(r, f, h(f), \theta_k)|^2\right\}. \tag{4.13}$$

A iteração gulosa é a política usada para determinar a função Q_L ótima. A política é parametrizada por $h(x) = h(x, \kappa)$. A lei de controle é obtida por meio da equação da otimalidade de *Bellman* para a função Q_L, que é dada por $u^* = \arg\max_u Q_L(x, u)$.

Os erros quadráticos entre o valor estimado $Q_L(x, u, \theta)$ e o valor medido $\Theta(\cdot)$ são minimizados, tendo em vista melhorar por meio da iteração gulosa a estimativa ótima dos parâmetros. Os novos parâmetros produzem novos alvos, desta forma o processo é computacionalmente realizável.

4.4 Solução do LQR Discreto via HDP e ADHDP

A programação dinâmica heurística e a programação dinâmica heurística dependente de ação são apresentadas no contexto da solução do problema DLQR. As abordagens de controle discreto, tais como DLQR, são destacadas nas referências (Kuo, 1980), (Jacquot, 1994) e (Lewis *et al.,* 2012).

4.4.1 DLQR via HDP

Para o LQR discreto, tem-se que as funções parametrizadas são classificadas em parâmetros Q e R do custo instantâneo (função de utilidade) e nos parâmetros A e B do sistema dinâmico (ambiente). As parametrizações do ambiente são dadas pelo sistema dinâmico que é representado por

$$f(x, h(x)) \quad = \quad Ax + Bu, \tag{4.14}$$

sendo $A \in \Re^{n \times n}$, n é a ordem do sistema, $B \in \Re^{n \times n_e}$ e n_e é a quantidade de entradas do sistema. A política de controle, formada a partir da combinação linear de estados, é dada por

$$h(x) \quad = \quad -K(.)x, \tag{4.15}$$

sendo $K(.) \in \Re^{n_e \times n}$ a matriz de ganhos da realimentação de estado.

A parametrização do custo instantâneo é a função de utilidade em k, representando os desvios dos estados e a energia da política do estado $k-1$ para o estado k, é dada por

$$r(x, u) \quad = \quad x^T Q x + u^T R u, \tag{4.16}$$

sendo $Q \in \Re^{ne \times ne} \geq 0$ é a matriz de ponderação do estado e $R \in \Re^{n \times n} > 0$ é a matriz de ponderação do controle

A parametrização da função de utilidade de Eq.(4.16) é substituída na Eq.(2.1) para obter a função de custo DLQR parametrizada, sendo a política de controle

dada por uma matriz de ganhos de realimentação constante K, Eq.(4.15). Especificamente, o objetivo agora é determinar a política K que minimize a soma dos custos instantâneos descontados incorridos ao longo do tempo que é dada por

$$V^K(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k}(x_i^T Q x_i + u_i^T R u_i). \tag{4.17}$$

Para $u_i = -Kx_i$, tem-se que

$$V^K(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} x_i^T (Q + K^T R K) x_i, \ \forall x_k \in X. \tag{4.18}$$

A solução ótima do custo é uma forma quadrática do estado parametrizada por P que é dada por

$$V(x) = x^T P x, \tag{4.19}$$

com $P \in \Re^{n \times n}$ simétrica. Esta forma é aproximada por uma transformação que utiliza o produto de *Kronecker* para obtenção de um sistema de equações lineares em P (solução de *Riccati*).

Aplicando-se o produto de *Kronecker* (Brewer, 1978) e as definições de vetorização, tem-se que o valor de custo ótimo, forma quadrática de Eq.(4.19), é representado por

$$\begin{aligned} x^T P x &= (x^T \otimes x^T) vec(P) && (4.20)\\ &= (x_1 x^T \ x_2 x^T \ \ldots \ x_n x^T) vec(P) && (4.21)\\ &= vec(x x^T)^T vec(P), && (4.22) \end{aligned}$$

sendo *vec* a funcão de vetorização.

Os parâmetros ótimos da aproximação satisfazem a condição de Eq.(4.5), com $\kappa = vec(K)$, isto é, o parâmetro κ é uma vetorização da matriz de ganho K. Supondo $\frac{\partial h}{\partial \kappa} \neq 0$, Eq.(4.6) reduz-se à

$$\frac{\partial r}{\partial h} + \gamma \frac{\partial V}{\partial f} \frac{\partial f}{\partial h} = 0. \tag{4.23}$$

Substituindo-se as derivadas $\frac{\partial r}{\partial h} = 2u^T R$, $\frac{\partial V}{\partial f} = 2f^T P$ e $\frac{\partial f}{\partial h} = B$ na Eq.(4.23), tem-se que a política ótima u é dada por

$$\begin{aligned} u &= -K(P^\theta)x \quad (4.24) \\ &= -Kx, \quad (4.25) \end{aligned}$$

sendo $K = \gamma(R + \gamma B^T P^\theta B)^{-1} B^T P^\theta A$ o ganho ótimo.

A abordagem crítica adaptativa é usada para determinar os parâmetros da aproximação da função valor. A Eq.(4.19) que minimiza o erro quadrático médio é parametrizada por

$$P_{k+1} = \arg\min_P E\{|x^T P x - \Theta(r, f, P_k)|^2\}. \quad (4.26)$$

A solução da Eq.(4.26) por aproximação e vetorização é representada por

$$\theta_{k+1} = \arg\min_\theta E\{|\bar{x}^T \theta - \Theta(r, f, \theta_k)|^2\}, \quad (4.27)$$

sendo $\bar{x} \in \Re^{n(n+1)/2}$ definido de acordo com o produto de *Kronecker* que é dado por $\bar{x}^T = [x_1^2 \ \ldots \ x_1 x_n \ \ x_2^2 \ \ldots \ x_{n-1} x_n \ \ x_n^2]$.

A função $\theta = vec(P) \in \Re^{n(n+1)/2}$ da matriz quadrada P é um vetor contendo as n entradas diagonais de P e as $n(n+1)/2 - n$ somas distintas $p_{ij} + p_{ji}$. Supondo uma ordenação entre o vetor $\bar{x}$ e a vetorização $vec(P)$ para representar a forma quadrática $x^T P x = \bar{x}^T vec(P)$, a estimativa paramétrica de mínimos quadrados é dada por

$$\theta_{k+1} = (E\{\bar{x}\bar{x}^T\})^{-1} E\{\bar{x}\Theta(r, f, P_k)\}. \quad (4.28)$$

A função $\Theta(r, f, P_k)$ da Eq.(4.28) é o valor desejado que consiste do custo atual e da função de custo a partir do estado seguinte $f(x, u)$, como pode-se observar na Eq.(4.2). As parametrizações do ambiente de Eq.(4.14), da política de controle de Eq.(4.15) e do custo instantâneo de Eq.(4.16) estabelecem a aproximação da função $\Theta(r, f, P)$ parametrizada para o DLQR que é dada por

$$\begin{aligned}\Theta(r,f,P) &= x^TQx + u^TRu + \\ &+ \gamma(Ax+Bu)^TP(Ax+Bu). \end{aligned} \tag{4.29}$$

As propriedades de produto de *Kronecker* contribuem para a solução da equação algébrica de *Riccati* discreta que é aproximada por um sistema de equações lineares para os coeficientes da matriz P. Consequentemente, a aproximação da solução da HJB-*Riccati* é uma solução de mínimos quadrados (Astrom e Wittenmark, 1994) dada por

$$\overline{x}^T\theta = \Theta(r,f,P^\theta), \tag{4.30}$$

sendo θ o vetor correspondente aos elementos da matriz de *Riccati*. O vetor de regressão e o custo ótimo são dados por

$$\overline{x} = [x_{1,k}^2 \ \ldots \ x_{1,k}x_{n,k} \ x_{2,k}^2 \ \ldots \ x_{n-1,k}x_{n,k} \ x_{n,k}^2]^T \tag{4.31}$$

$$\Theta(r,f,P^\theta) = x_k^TQx_k + u_k^TRu_k + \gamma x_{k+1}^TP^\theta x_{k+1}, \tag{4.32}$$

sendo x_{lk} a l-ésima, $l = 1, 2, ..., n$, componente do vetor de estado x no passo de tempo k.

4.4.2 DLQR via ADHDP

A função Q_L do DLQR é uma parametrização da Eq.(4.7) que leva em consideração as parametrizações do sistema dinâmico de Eq.(4.14), a política de controle de Eq.(4.15) e o custo instantâneo de Eq.(4.16). Consequentemente, a parametrização do lado direito da equação de *Bellman* é dada por

$$Q_L(x,u) = x_k^TQx + u^TRu + \gamma(Ax+Bu)^TP(Ax+Bu). \tag{4.33}$$

Após manipulações algébricas, obtém-se uma forma quadrática que é dada por

$$Q_L(x,u) = z^TQ_{LV}z, \tag{4.34}$$

sendo $z^T = [x^T \ u^T]$ e $Q_{LV} \in \Re^{(n+n_e)\times(n+n_e)}$ é a matriz de aprendizagem associada à função Q_L, que é dada por

$$Q_{LV} = \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix}, \tag{4.35}$$

sendo $Q_{xx} = Q + \gamma A^T P A$, $Q_{xu} = \gamma A^T P B$, $Q_{ux} = \gamma B^T P A$ e $Q_{uu} = R + \gamma B^T P B$ as matrizes que representam as ponderações do estado x e da política u.

A minimização da função Q_L parametrizada fornece os meios para determinação da política u ótima. A equação do gradiente $\partial Q_L / \partial u = 0$ quando resolvida para u fornece a política ótima

$$\begin{aligned} \frac{\partial Q_L}{\partial u} &= \frac{\partial}{\partial u}\left\{ [x^T \ \ u^T] \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \right\} \\ &= \frac{\partial}{\partial u}\left(x^T Q_{xx} x + 2u^T Q_{ux} x + u^T Q_{uu} u \right) \\ &= 2Q_{ux}x + 2Q_{uu}u \\ &= 0. \end{aligned}$$

Consequentemente, a lei de controle ótimo é dada por

$$u = -Q_{uu}^{-1} Q_{ux} x \tag{4.36}$$

$$= -K(Q_{LV})x \tag{4.37}$$

$$= -Kx. \tag{4.38}$$

A parametrização de $Q_L(Q_{LV})$ induz uma parametrização da política $h(Q_{LV})$. De acordo com a parametrização $u = h(x, \kappa)$ da política de controle, o vetor de parâmetros κ é uma vetorização da matriz de ganhos K, isto é, $\kappa = vec(K)$. Comparando-se com a abordagem HDP, não são necessários parâmetros além dos já existentes na função Q_L. Uma diferença entre as duas abordagens é claramente observada a partir das Eqs.(4.24)-(4.25) e Eqs.(4.36)-(4.38) para atualizações das políticas HDP e ADHDP, respectivamente. Enquanto as Eqs.(4.24)-(4.25) expressam ter uma dependência explícita dos modelos do ambiente e da recompensa instantânea por meio das matrizes A, B e R, por outro lado, nas Eqs.(4.36)-(4.38) não são exibidos tais modelos.

4.4.3 Convergência de HDP-DLQR

Na análise de convergência de HDP avalia-se o seguinte algoritmo do gradiente para resolver o problema de minimização na Eq.(4.26) com respeito à P, a matriz correspondente ao vetor de parâmetros θ

$$\begin{aligned} P_{k+1} &= P_k - \eta \frac{\partial}{\partial P'} E\Big\{ |x_k^T P' x_k - \Theta(r, f, P_k)|^2 \Big\}_{P'=P_k} \\ &= P_k - \eta E\Big\{ \frac{\partial}{\partial P'} |x_k^T P' x_k - \Theta(r, f, P_k)|^2 \Big\}_{P'=P_k} \\ &= P_k - 2\eta E\{x_k x_k^T (x_k^T P_k x_k - \Theta(r, f, P_k))\}, \end{aligned} \tag{4.39}$$

sendo η um fator de aprendizagem.

Introduzindo-se a notação $\upsilon = vec(xx^T)$, $p = vec(P)$, $\alpha = 2\eta$, e usando-se o fato de que uma forma quadrática pode ser expressa como um produto escalar, conforme descrito na Eq.(4.22), a estimativa da função valor V na forma vetorizada é atualizada por

$$p_{k+1} = p_k - \alpha E\{\upsilon_k (\upsilon_k^T p_k - \Theta(r, f, P_k))\}. \tag{4.40}$$

O uso da função *vec* permite substituir o valor esperado na Eq.(4.40) pelo valor instantâneo da derivada. Fazendo assim, obtém-se a seguinte versão estocástica da Eq.(4.40)

$$\begin{aligned} p_{k+1} &= p_k - \alpha \upsilon_k (\upsilon_k^T p_k - \Theta(r(x_k, u_k), x_{k+1}, P_k)) \\ &= p_k - \alpha \upsilon_k (\upsilon_k^T p_k - [x_k^T Q x_k + u_k^T R u_k + (Ax_k + Bu_k)^T P_k (Ax_k + Bu_k)]) \\ &= p_k - \alpha \upsilon_k (\upsilon_k^T p_k - [x_k^T (Q + K_k^T R K_k) x_k + x_k^T (A - BK_k)^T P_k (A - BK_k) x_k]) \\ &= p_k - \alpha \upsilon_k \upsilon_k^T vec(P_k - [Q + K_k^T R K_k + (A - BK_k)^T P_k (A - BK_k)]). \end{aligned} \tag{4.41}$$

Observe que toda informação necessária por HDP para produzir o alvo é dada pela recompensa atual e pela função valor avaliada no estado seguinte do sistema, e o fator de desconto γ foi estabelecido igual a 1. Agora, examinando-se a equação de atualização na média, tem-se

$$E\{vec(P_{k+1})\} = E\{vec(P_k)\} - \alpha C_v E\{vec(\Delta P_k)\}, \tag{4.42}$$

sendo $C_v = E\{vv^T\} > 0$ uma matriz de covariância simétrica. Pode-se provar que se uma sequência $w_{k+1} = w_k + \alpha \Delta w_k$ converge, então a sequência $w_{k+1} = w_k + \alpha C \Delta w_k$ também converge, com $C > 0$ simétrica.

Reformulando-se a Eq.(4.42) em termos de matrizes, com $W = E\{P\}$ e $C_v = I$, resulta na seguinte equação para a atualização dos parâmetros da função valor na média:

$$W_{k+1} = W_k - \alpha[W_k - (Q + K_k^T R K_k + (A - BK_k)^T W_k (A - BK_k))], \tag{4.43}$$

sendo a matriz de realimentação dada por

$$K_k = K(P_k) \;\; = \;\; (R + B^T P_k B)^{-1} B^T P_k A. \tag{4.44}$$

A prova de que a sequência $\{W_k\}_{k=0}^{\infty}$ de fato converge para os parâmetros correspondentes à função valor ótima $V^*(x) = x^T P^* x$ pode ser encontrada em (Landelius e Knutsson, 1996). As condições de convergência da Eq.(4.43) são garantidas se as seguintes relações são satisfeitas

$$\alpha \quad \in \quad (0, 1] \tag{4.45}$$

$$R \quad > \quad 0 \tag{4.46}$$

$$Q \quad \geq \quad 0 \tag{4.47}$$

$$\text{O par} \quad (A, B) \quad \text{é controlável.} \tag{4.48}$$

O valor ótimo W^* é a única solução semidefinida positiva da equação HJB-*Riccati*

$$W^* \;\; = \;\; Q + A^T W^* A - A^T W^* B (R + B^T W^* B)^{-1} B^T W^* A. \tag{4.49}$$

4.4.4 Convergência de ADHDP-DLQR

De modo análogo à análise de convergência de HDP, considera-se o seguinte algoritmo do gradiente para resolver o problema de minimização na Eq.(4.13) com respeito à Q_{LV}, a matriz correspondente ao vetor de parâmetros θ

$$\begin{aligned} Q_{LV_{k+1}} &= Q_{LV_k} - \eta \frac{\partial}{\partial Q'_{LV}} E\Big\{ |z_k^T Q'_{LV} z_k - \Theta(r, f, h(f), Q_{LV_k})|^2 \Big\}_{Q'_{LV}=Q_{LV_k}} \\ &= Q_{LV_k} - \eta E\left\{ \frac{\partial}{\partial Q'_{LV}} |z_k^T Q'_{LV} z_k - \Theta(r, f, h(f), Q_{LV_k})|^2 \right\}_{Q'_{LV}=Q_{LV_k}} \\ &= Q_{LV_k} - 2\eta E\{ |z_k z_k^T (z_k^T Q_{LV_k} z_k - \Theta(r, f, h(f), Q_{LV_k}))\}, \end{aligned} \tag{4.50}$$

sendo η um fator de aprendizagem.

Introduzindo-se a notação $\upsilon = vec(zz^T)$, $q_{LV} = vec(Q_{LV})$ e $\alpha = 2\eta$, a estimativa da função valor Q_L na forma vetorizada é atualizada por

$$q_{LV_{k+1}} = q_{LV_k} - \alpha E\{\upsilon_k (\upsilon_k^T q_{LV_k} - \Theta(r, f, h(f), Q_{LV_k}))\}. \tag{4.51}$$

O objetivo dado na Eq.(4.12) para o algoritmo de aprendizagem ADHDP pode ser expresso por

$$\begin{aligned} &\Theta(r, f, h(f), Q_{LV_k}) \\ &= r(x,u) + \gamma Q_L(f(x,u), h(f(x,u)), Q_{LV_k}) \\ &= x^T Q x + u^T R u + \gamma \begin{bmatrix} f^T & h^T(f) \end{bmatrix} Q_{LV_k} \begin{bmatrix} f \\ h(f) \end{bmatrix} \\ &= z^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} z + \gamma \begin{bmatrix} (Ax+Bu)^T & -K^T(Ax+Bu)^T \end{bmatrix} Q_{LV_k} \begin{bmatrix} Ax+Bu \\ -K(Ax+Bu) \end{bmatrix} \\ &= z^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} z + \gamma z^T \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix}^T Q_{LV_k} \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix} z. \end{aligned} \tag{4.52}$$

Substitue-se agora esta última expressão da função objetivo na Eq.(4.51) para obter a seguinte versão estocástica da Eq.(4.51)

$$
\begin{aligned}
q_{LV_{k+1}} &= q_{LV_k} - \alpha \upsilon_k (\upsilon_k^T q_{LV_k} - \Theta(r(z_k), z_{k+1}, Q_L(z_{k+1}, Q_{LV_k}))) \\
&= q_{LV_k} - \alpha \upsilon_k \left\{ \upsilon_k^T q_{LV_k} - \left(z_k^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} z_k + \right.\right. \\
&\qquad \left.\left. + z_k^T \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix}^T Q_{LV_k} \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix} z_k \right) \right\} \qquad (4.53) \\
&= q_{LV_k} - \alpha \upsilon_k \upsilon_k^T vec \left\{ Q_{LV_k} - \left(\left[\begin{matrix} Q & 0 \\ 0 & R \end{matrix} \right] + \right.\right. \\
&\qquad \left.\left. + \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix}^T Q_{LV_k} \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix} \right) \right\}.
\end{aligned}
$$

Observe que a única informação que ADHDP necessita para produzir o objetivo é dada pela recompensa atual e pela função Q_L avaliada no estado e ação de controle seguintes, e o fator de desconto γ foi estabelecido igual a 1. Agora, examinando-se a equação de atualização vetorizada na média, tem-se

$$E\{vec(Q_{LV_{k+1}})\} = E\{vec(Q_{LV_k})\} - \alpha C_\upsilon E\{vec(\Delta Q_{LV_k})\}, \qquad (4.54)$$

sendo $C_\upsilon = E\{\upsilon\upsilon^T\} > 0$ uma matriz de covariância simétrica. Pode-se provar que se uma sequência $w_{k+1} = w_k + \alpha \Delta w_k$ converge, então a sequência $w_{k+1} = w_k + \alpha C \Delta w_k$ também converge, com $C > 0$ simétrica.

Reformulando-se a Eq.(4.54) em termos de matrizes, com $W = E\{Q_{LV}\}$ e $C_\upsilon = I$, resulta na seguinte equação para a atualização dos parâmetros da função valor na média:

$$
\begin{aligned}
W_{k+1} = W_k - \alpha \left\{ W_k - \left(\left[\begin{matrix} Q & 0 \\ 0 & R \end{matrix} \right] + \right.\right. \\
\left.\left. + \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix}^T Q_{LV_k} \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix} \right) \right\},
\end{aligned} \qquad (4.55)
$$

sendo a matriz de realimentação dada por

$$K_k = K(P_k) \;\; = \;\; (R + B^T P_k B)^{-1} B^T P_k A. \qquad (4.56)$$

A prova de que a sequência $\{W_k\}_{k=0}^{\infty}$ de fato converge para os parâmetros correspondentes à função Q_L ótima $Q_L^*(x) = x^T Q_{LV}^* x$ pode ser encontrada em (Landelius e Knutsson, 1996).

4.5 Considerações Finais

Neste capítulo, os métodos de programação dinâmica heurística e programação dinâmica heurística dependente de ação, os quais estão inseridos na abordagem crítica adaptativa, foram apresentados para a determinação de soluções aproximadas da equação de *Hamilton-Jacobi-Bellman* e políticas de decisão ótimas para viabilizar o desenvolvimento de projetos *online* de sistemas de controle ótimo. Especificamente, a aproximação da função valor para uma dada política de decisão usou uma formulação apoiada em teoria de álgebra de Kronecker e métodos aproximados de descida do gradiente, e aplicações foram realizadas sobre o problema DLQR. As parametrizações da equação de *Bellman*, função de utilidade e sistema dinâmico montam um quadro para a solução do problema DLQR. A aproximação foi apresentada para se determinar os coeficientes da matriz P e a matriz de ganhos do controlador ótimo DLQR para HDP e ADHDP, respectivamente.

Uma diferença entre as abordagens HDP e ADHDP pode ser verificada na regra das melhorias de políticas, em que na ADHDP a regra de decisão é completamente independente do modelo da planta, enquanto que na HDP, a regra de decisão é baseada em modelo, entretanto, para a atualização dos parâmetros do crítico não se necessita de tal modelo. Especificamente, no contexto de controle ótimo DLQR, uma diferença entre as abordagens é dada em termos do cálculo da matriz P. Na ADHDP, ao contrário da HDP, esta matriz não é obtida diretamente, pois o ganho K é determinado a partir das submatrizes Q_{uu} e Q_{ux} da função parametrizada Q_L para o DLQR.

É importante salientar também que os métodos críticos adaptativos diferem-se da forma como o problema DLQR é abordado usando técnicas clássicas de controle ótimo. Esta diferença torna-se bastante evidente quando a abordagem ADHDP é aplicada. Aqui somente amostras da função de utilidade juntamente com os parâmetros da função Q_L são necessários para alcançar o controlador ótimo. Enquanto que nos métodos clássicos de controle ótimo, a função de utilidade assume-se ser

conhecida e os parâmetros do ambiente necessitam serem estimados e usados na solução da equação de *Riccati* antes que o controlador ótimo seja calculado.

Capítulo 5

Aproximadores RLS para Solução da Equação HJB-*Riccati* e Programação Dinâmica Heurística Dependente de Estado e Ação

5.1 Introdução

Um aspecto relacionado à abordagem ator-crítico é que as estimativas da função valor de uma política são atualizadas em cada passo de tempo usando-se dados observados do sistema, sem qualquer conhecimento da dinâmica interna. No contexto de aprendizagem incremental ator-crítico, diferenças temporais (TD) destacam-se como uma importante classe de métodos independentes de modelo para projetos de controle *online*. A eficiência de métodos TD é analisada por (Bradtke e Barto, 1996) que propõe uma abordagem baseada em mínimos quadrados (*Least Squares* - LS) denominada LSTD para melhorar a eficiência de dados e superar a dificuldade do projeto *step-size* com aproximação linear da função valor. Embora os métodos LSTD exigem mais computação por passo de tempo comparados aos métodos TD, eles tipicamente requerem um número menor de etapas de tempo para alcançar uma dada exatidão da função valor. Uma

extensão da abordagem LSTD que incorpora o conceito de traços de elegibilidade foi fornecida por (Boyan, 2002).

Vários resultados usando os métodos de mínimos quadrados em conjunção com iteração de política têm sido apresentados para solução *online* de problemas de controle ótimo, (Li *et al.,* 2009), (Buşoniu *et al.,* 2010*d*). Melhorias nos métodos de iteração de política de mínimos quadrados *online* podem ser observadas em (Buşoniu *et al.,* 2010*c*) e (Buşoniu *et al.,* 2010*b*). Alí, os autores propõem o uso de conhecimento à priori para acelerar iteração de política de mínimos quadrados *online*, o que é um paradoxo pois métodos RL são vistos como um paradigma para funcionarem sem qualquer conhecimento à priori do sistema. Em (Buşoniu *et al.,* 2012), os autores apresentam um estado da arte sobre métodos de mínimos quadrados para iteração de política aproximada. Sob este escopo, métodos RLS também têm sido usados para viabilizar a realização *online* em uma forma mais eficiente, (Xu *et al.,* 2002), (Cheng *et al.,* 2012), bem como reduzir a carga computacional dos métodos LSTD.

Pesquisa sobre controle quadrático linear adaptativo usando-se esquemas ator-crítico baseados em diferenças temporais com aproximadores RLS de função pode ser encontrada em (Bradtke *et al.,* 1994), (Bradtke, 1994). Contudo, os resultados de convergência de Bradtke são limitados à ADHDP e métodos de iteração de política, em que as melhorias de políticas são determinadas baseadas sobre uma avaliação completa da política atual, isto é, a avaliação de política é processada até alcançar (próximo) a convergência, diferentes das estratégias discutidas neste capítulo, em que as melhorias de política são realizadas a cada amostra de transição de estado antes que uma avaliação da política atual seja concluída. Um esquema de iteração de política de mínimos quadrados para aprendizagem por reforço em espaço de estados contínuos foi proposto por (Xu *et al.,* 2007). O esquema deles é baseado em funções de *Kernel* e é orientado para realização de controle de realimentação adaptativa de sistemas dinâmicos incertos.

O desenvolvimento de métodos de diferenças temporais com aproximação de função valor baseados na abordagem RLS é ainda um campo ativo de pesquisa. Por exemplo, Pietquin e outros (Pietquin *et al.,* 2011) apresentaram um avanço recente nos métodos TD como diferença temporal de Kalman (*Kalman Temporal Differences* - KTD). Nos esquemas propostos, um filtro de Kalman foi incorporado

na estimação do processo de aproximação da função valor usando uma representação de espaço de estados. Um desenvolvimento anterior sobre paradigmas KTD é apresentado no trabalho de Geist e outros (Geist *et al,* 2009) para processos de decisão markovianos determinísticos. Eles propuseram aprendizagem-Q baseada em KTD (aprendizagem KTD-Q) e SARSA baseada em KTD (KTD-SARSA). A abordagem proposta por eles é válida para parametrização linear e não-linear.

Neste trabalho, os esquemas RL ator-crítico são baseados em estimação paramétrica RLS formulados no quadro de programação dinâmica aproximada, especificamente programação dinâmica heurística dependente de estado e ação, tendo em vista a solução *online* de problemas de controle ótimo. Sob este escopo, uma contribuição principal nesta pesquisa é a formulação e solução dos problemas de convergência e estabilidade numérica que estão relacionados com o mal-condicionamento da matriz de covariância da abordagem RLS para aproximações *online* da solução da equação HJB-*Riccati* associada ao problema DLQR via esquemas RL ator-crítico. Devido à problemas de estabilidade numérica, a solução para a política de decisão ótima para um dado ponto de operação pode não convergir. Dessa forma, a abordagem 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 UDU^T e 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 em uma 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.

Outro foco importante dado aqui é a proposta de uma metodologia baseada na fusão de métodos RLS, aprendizagem TD(λ) e melhorias de políticas para o desenvolvimento de algoritmos RL ator-crítico para solução DLQR, tendo em vista a formulação de estratégias que promovam melhorias no processo de aprendizagem no sentido de acelerar a busca de política de controle ótima DLQR. As estratégias são avaliadas em termos do efeito do parâmetro λ do vetor de elegibilidade por meio de estatísticas de primeira e segunda ordem. A abordagem é direcionada para aproximações *online* de métodos de iteração de política para solução DLQR

no sentido que as melhorias de política são realizadas em cada passo de tempo ao longo da realização de trajetória de estado em direção à política ótima DLQR.

5.2 Caracterização e Formulação do Problema

A dificuldade em implementar o projeto *online* de sistemas de controle ótimo reside na complexidade computacional envolvida na solução da equação de *Bellman* (4.1) devido à "maldição da dimensionalidade", (Lendaris, 2009). Os resultados de convergência para a aprendizagem da função valor de acordo com as Eqs.(2.20) e (2.21) assumem um PDM com espaço de estado finito e são baseados em uma representação tabular da função valor, em que os valores de V^h são armazenados para todos os estados x_k, (Bertsekas e Tsitsiklis, 1996). Tais representações não são viáveis para PDMs com um grande número de estados. Os métodos ADP propõem uma viabilização do projeto *online* dos sistemas de controle ótimo que combinam aproximação da função valor e métodos de simulação, tais como diferenças temporais, para superar a maldição da dimensionalidade.

Motivado pelo desafio exposto acima, o desenvolvimento de métodos e algoritmos para o projeto de sistemas de controle *online* tem sido proposto, tendo em vista aplicações em controle ótimo ou filtragem ótima no contexto da abordagem RL e ADP. As soluções propostas baseiam-se na aproximação da solução da equação HJB via métodos paramétricos, tais como LS ou um de seus derivados: RLS, Projeção, Kaczmarz e Filtro de Kalman, (Astrom e Wittenmark, 1994). Então, o problema de aproximação da equação HJB é formulado como um problema de regressão ou estimação de parâmetro.

No presente estudo, a estrutura paramétrica para representar a função valor assume a forma dada por

$$\widehat{V}^h(x_k,\theta) = \varphi^T(x_k)\widehat{\theta}, \tag{5.1}$$

em que $\varphi(x_k) = [\varphi_1(x_k)\ \varphi_2(x_k)\ \ldots\ \varphi_{n_\theta}(x_k)]^T$ é o vetor de funções de base e $\widehat{\theta} = [\widehat{\theta}_1\ \widehat{\theta}_2\ \ldots\ \widehat{\theta}_{n_\theta}]^T$ é o vetor de peso estimado. A investigação em curso leva em consideração o método RLS para o problema de estimação do parâmetro θ. A abordagem RLS considerada é para viabilizar a realização da aprendizagem *online*

para projetos de controle ótimo, uma vez que as estimativas da função de custo de uma dada política de controle precisam ser construídas, incrementalmente, a partir de dados $(x_k, x_{k+1}, r(x_k, h(x_k)))$ observados do sistema ao longo de sua trajetória.

Observou-se que o projeto *online* tornou-se instável durante o processo de estimação RLS para determinação das ações de decisão. A situação é caracterizada como um problema de estabilidade numérica associado com a matriz de covariância do RLS que se origina na formação ou natureza dos vetores de regressores (estados) os quais são governados pelo processo, gerando vetores linearmente dependentes ou fracamente linearmente independentes quando associados a RL e Controle Ótimo.

Tem-se observado que o problema da perda de estabilidade numérica ocorre durante o comportamento em regime permanente do processo iterativo da estimação paramétrica, enquanto que durante o comportamento transitório, tem-se informação o suficiente para gerar a base de vetores de regressores. Pergunta-se: o que origina este fenômeno? À priori, parte-se da premissa que os elementos que compõem o vetor de regressores tendem à tornarem-se linearmente dependentes após o sistema atingir o regime permanente. Este tipo de fenômeno inviabiliza o projeto *online* dos controladores já que o sistema perde a estabilidade numérica.

O problema de estabilidade numérica é conceituado nesta seção em termos do problema de inversão da matriz Φ do método RLS que é dado por $\Gamma(k) = \Phi^{-1}(k)$, em que $\Gamma(\cdot)$ é a matriz de covariância. O problema de avaliação do comportamento do vetor de regressores na sua geração é solucionado pelo número de condição e o parâmetro de positividade da matriz de covariância $\Gamma(\cdot)$. A solução proposta via fatoração UDU^T 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 e diferenças temporais no contexto da abordagem RL.

5.2.1 RLS com Fator de Esquecimento

Considere o seguinte modelo de regressão linear dado por

$$y(k) = \phi^T(k)\theta, \tag{5.2}$$

sendo k o índice de iteração, $y(k)$ a variável observada, $\theta = [\theta_1 \quad \theta_2 \quad \dots \quad \theta_{n_\theta}]^T$ o vetor de parâmetros a ser determinado e $\phi^T(k) = [\phi_1(k) \quad \phi_2(k) \quad \dots \quad \phi_{n_\theta}(k)]^T$ o vetor de funções conhecidas, geralmente chamadas de variáveis de regressão ou regressores.

O problema é determinar uma estimativa do vetor de parâmetros θ a partir de um dado conjunto de pares de observações e regressores $\{(y(k), \phi(k)),\ k = 1, 2, \ \cdots, \ N\}$. A estimativa de mínimos quadrados de θ é definida como o vetor que minimiza a seguinte função de perda

$$J(\theta, N) = \sum_{k=1}^{N} \mu^{N-k} \left[y(k) - \phi^T(k)\theta\right]^2, \tag{5.3}$$

sendo μ o fator de esquecimento que é definido no intervalo $0 < \mu \leq 1$. Este parâmetro pondera dinamicamente a influência de dados amostrados na função de perda, descontando os dados mais antigos, de modo a tornar θ representativo para as propriedades atuais do sistema, (Astrom e Wittenmark, 1994). Em outras palavras, o fator de esquecimento daria menos peso a observações feitas em regimes dinâmicos (pontos de operação) "distantes"daquele para o qual o modelo está sendo identificado. Nessa situação, ainda que todas as observações tenham a mesma precisão àquelas menos representativas para o ponto de operação em questão podem naturalmente receber menor peso.

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 dada por

$$\theta_{MQ}(N) = \Phi^{-1}(N)\ \Theta(N), \tag{5.4}$$

sendo

$$\Phi(N) = \sum_{k=1}^{N} \mu^{N-k} \phi(k)\phi^T(k) \tag{5.5}$$

e

$$\Theta(N) = \sum_{k=1}^{N} \mu^{N-k} \phi(k) y(k). \tag{5.6}$$

Por manipulação simples, as Eqs.(5.4)-(5.6) são desenvolvidas em uma forma recursiva dada por

$$\theta(k) = \theta(k-1) + \Phi^{-1}(k)\phi(k)(y(k) - \phi^T(k)\theta(k-1)), \tag{5.7}$$

sendo $\Phi(k)$ uma forma recursiva que é dada por

$$\Phi(k) = \mu\Phi(k-1) + \phi(k)\phi^T(k). \tag{5.8}$$

Introduzindo-se $\Gamma(k) = \Phi^{-1}(k)$ e aplicando-se o lema da inversão de matriz à Eq.(5.8), a estimativa RLS na forma (5.7)-(5.8) pode ser reescrita por

$$\theta(k) = \theta(k-1) + L(k)(y(k) - \phi^T(k)\theta(k-1)), \tag{5.9}$$

sendo

$$L(k) = \frac{\Gamma(k-1)\phi(k)}{\mu + \phi^T(k)\Gamma(k-1)\phi(k)}, \tag{5.10}$$

e

$$\Gamma(k) = \mu^{-1}\left[\Gamma(k-1) - L(k)\phi^T(k)\Gamma(k-1)\right]. \tag{5.11}$$

5.2.2 RLS com Fatoração UDU^T

Uma versão modificada de (5.9)-(5.11) pode ser obtida supondo-se que $\Gamma(k) = U(k)D(k)U^T(k)$, sendo $U(k)$ uma matriz triangular superior com todos os elementos diagonais unitários e $D(k)$ uma matriz diagonal.

Sejam $U(k) = [u_1(k) \ \cdots \ u_{n_\theta}(k)]$ e $D(k) = diag(d_1(k), \ \cdots, \ d_{n_\theta}(k))$, com $u_j(k) = [u_{1j}(k) \ \cdots \ u_{j-1j}(k) \ 1 \ 0 \ \cdots \ 0]^T$. Supondo-se que $\Gamma(k-1)$ admite a

decomposição $\Gamma(k-1) = U(k-1)D(k-1)U^T(k-1)$, as Equações (5.10) e (5.11) podem ser rearranjadas para

$$L(k) \ = U(k-1)g\beta^{-1} \tag{5.12}$$

e

$$\Gamma(k) = \mu^{-1}U(k-1)\left[D(k-1) - \beta^{-1}gg^T\right]U^T(k-1), \tag{5.13}$$

sendo

$$g = D(k-1)e \ , \ e = U^T(k-1)\phi(k) \tag{5.14}$$

e

$$\beta = \mu + e^T D(k-1)e = \mu + e^T g. \tag{5.15}$$

Uma vez que o produto de duas matrizes triangulares superiores unitárias é ainda uma matriz triangular superior unitária, para obter-se fatores U-D de $\Gamma(k)$ é suficiente obter-se fatores U-D para a expressão entre colchetes na Eq.(5.13). Os detalhes da dedução dos parâmetros da fatoração são apresentados no Apêndice B.

5.2.3 O Problema da Inversão de $\Phi(N)$

O cálculo da inversa de $\Phi(N)$ na Equação (5.4) pode ser realizado de forma recursiva pela Eq.(5.11) para o RLS com fator de esquecimento ou pelo RLS com fatoração UDU^T pela Eq.(5.13). Desta forma, o problema proposto é estabelecido como a determinação da inversa de $\Phi(k)$ para métodos RLS que é dada por

$$\Gamma(k) \ = \ \Phi^{-1}(k). \tag{5.16}$$

O problema dado pela relação (5.16) é investigado em termos do número de condição da matriz $\Gamma(k)$ e do parâmetro de positividade, tendo em vista avaliar a convergência e a estabilidade numérica do método RLS para problema de estimação da solução da HJB-*Riccati* via programação dinâmica aproximada.

5.2.4 Convergência e Estabilidade Numérica

Uma questão muito importante em estimação de parâmetros via RLS refere-se à sua convergência e estabilidade numérica em ambientes de precisão finita. Para os algoritmos RLS convencionais, Potter (Potter, 1963) observou que a degradação de exatidão numérica quando ocorre geralmente está associada com problemas de mal condicionamento da matriz de covariância, a qual perde a sua propriedade definida positiva. Questões de como embutir algoritmos RLS em ambientes de precisão finita para melhores propriedades numéricas são discutidas em (Bierman, 1977), e diferentes procedimentos são sugeridos para melhorar a exatidão numérica e preservar a positividade da matriz de covariância do RLS. Diversas formas de análise foram desenvolvidas para explicar a origem e a propagação do fenômeno da instabilidade numérica, (Ljung e Ljung, 1985), (Liavas e Regalia, 1999), (Verhaegen, 1989), (Slock, 1992), (Cioffi, 1987).

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 à forma de fatoração UDU^T. O uso de fatoração UDU^T na estimação RLS apresenta propriedades numéricas desejáveis conforme pode ser constatado em (Golub e Charles, 1996) e (Wilkinson, 1968).

O problema da convergência e estabilidade numérica para o RLS é abordado nesta seção em termos das avaliações do número de condição da matriz de covariância da estimativa e do parâmetro de positividade. Analisa-se também a robustez numérica dos métodos RLS convencionais (sem fatoração UDU^T) e RLS com fatoração UDU^T com respeito à propagação de uma pequena matriz de erro adicionada na matriz de covariância $\Gamma(k)$ no instante de recursão $k-1$ para as recursões seguintes.

Número de condição

O número de condição da matriz de covariância $\Gamma(k)$ é definido pela relação entre o maior σ_{max} e o menor σ_{min} valores singulares de $\Gamma(k)$ que é dado por

$$cond(\Gamma(k)) \quad = \quad \frac{\sigma_{max}(\Gamma(k))}{\sigma_{min}(\Gamma(k))}. \tag{5.17}$$

Métodos de fatoração UDU^T são do tipo *raiz quadrada* no sentido que os

fatores *raiz quadrada* são propagados, ao invés da própria matriz $\Gamma(k)$. Como discutido em (Bierman, 1977), isso torna as matrizes melhores condicionadas. Com efeito, mostremos a relação entre os números de condição da matriz de covariância $\Gamma(k)$ com o do seu fator *Cholesky* G, o qual é dado por

$$G(k) \;=\; U(k)D^{1/2}(k), \tag{5.18}$$

sendo

$$D^{1/2}(k) = diag\left(\sqrt{d_1(k)},\ \cdots,\ \sqrt{d_{n_\theta}(k)}\right). \tag{5.19}$$

O número de condição da matriz de covariância $\Gamma(k)$ pode ser expresso por

$$\begin{aligned} cond(\Gamma(k)) &= cond(U(k)D(k)U^T(k)) && (5.20)\\ &= cond(G(k)G^T(k)) && (5.21)\\ &= [cond(G(k))]^2. && (5.22) \end{aligned}$$

Da relação (5.22), observa-se que a diferença entre os números de condição das matrizes $G(k)$ e $\Gamma(k)$ torna-se siginificativa à medida que $cond(\Gamma(k))$ aumenta. Isto conduz a uma robustez numérica melhorada dos métodos *raiz quadrada*. Assuntos sobre estabilidade computacional e exatidão dos métodos de fatoração UDU^T podem ser encontrados em (Golub e Charles, 1996) e (Wilkinson, 1968).

Parâmetro de positividade

Uma medida do "bom comportamento"da matriz $\Gamma(k)$ pode ser dada pelo parâmetro de positividade p definido por

$$p(k) = 1 - L^T(k)\phi(k). \tag{5.23}$$

Usando a definição do vetor de ganho de Eq.(5.10) e o fato de que $\Gamma(k)$ é simétrica, isto é $\Gamma^T(k) = \Gamma(k)$, após simplificações a Eq.(5.23) pode ser reescrita por

$$p(k) = \frac{\mu}{\mu + \phi(k)^T \Gamma(k-1)\phi(k)}. \tag{5.24}$$

Observando que a forma simétrica em (5.24) possui um valor real não-negativo, tem-se $0 < p \leq 1$. Portanto, uma condição necessária para que a matriz $\Gamma(k)$ seja definida positiva é dada por $0 < p < 1$. A ocorrência de um valor fora desta faixa indica que a matriz $\Gamma(k)$ perdeu a propriedade definida positiva.

O parâmetro de positividade $p(k)$ tem sido também interpretado como uma razão entre os erros de predição a posteriori e a priori associados à estimativa do parâmetro θ, veja por exemplo (Romano, 1995), em que ele é utilizado para prever a acorrência da divergência provocada por erros de quantização (erros de arredondamento).

Análise de propagação de erro na matriz de covariância $\Gamma(k)$

Lema 5.2.1. *Seja n um inteiro positivo e seja $\Re^{n\times n}$ o espaço vetorial das matrizes $n \times n$ sobre $\Re$. Seja W_1 o subespaço das matrizes simétricas, isto é, $W_1 = \left\{M \in \Re^{n\times n};\, M^T = M\right\}$. Seja W_2 o subespaço das matrizes anti-simétricas, isto é, $W_2 = \left\{M \in \Re^{n\times n};\, M^T = -M\right\}$. Então $\Re^{n\times n} = W_1 \oplus W_2$, em que $\oplus$ denota a soma direta de subespaços*[1]*. Além disso, se M é uma matriz arbitrária em $\Re^{n\times n}$, a única expressão para M como uma soma de duas matrizes, uma em W_1 e outra em W_2, é*

$$M = M_1 + M_2, \tag{5.25}$$

com

$$M_1 = \frac{1}{2}(M + M^T) \tag{5.26}$$

e

[1]Se $W_1, \cdots, W_r$ são subespaços de $\Re^{n\times n}$, então a soma deles é o susbespaço definido por $W = \{a_1 + a_2 + \cdots + a_r;\, a_i \in W_i\,,\, 1 \leq i \leq r\}$. W é uma soma direta se cada $v \in W$ tem uma única representação da forma $v = a_1 + \cdots + a_r$, com $a_i \in W_i$. Neste caso, escreve-se $W = W_1 \oplus \cdots \oplus W_r$.

$$M_2 = \frac{1}{2}(M - M^T). \tag{5.27}$$

Demonstração:

De fato, uma dada matriz $M \in \Re^{n\times n}$ pode ser ser escrita sob a forma $M = M_1 + M_2$, $M_i \in W_i$, de uma única maneira. Isto resulta do fato de que se também $M = N_1 + N_2$ com $N_i \in W_i$, então

$$M_1 + M_2 = N_1 + N_2 \tag{5.28}$$

de modo que aplicando-se a transposição a ambos os membros da Eq.(5.28), obtém-se

$$M_1 - M_2 = N_1 - N_2. \tag{5.29}$$

A partir das Eqs.(5.28) e (5.29), é fácil ver que $M_1 = N_1$ e $M_2 = N_2$.

As expressões (5.26) e (5.27) podem ser deduzidas do fato de que para qualquer vetor $y \in \Re^n$, tem-se

$$y^T My = (y^T My)^T = y^T M^T y \tag{5.30}$$

e assim

$$y^T My = \frac{y^T My + y^T M^T y}{2} = y^T \frac{M + M^T}{2} y. \tag{5.31}$$

Note que a expressão $\frac{M+M^T}{2}$ que aparece na Eq.(5.31) é uma matriz simétrica. A componente anti-simétrica é dada pela solução da equação $M = \frac{M+M^T}{2} + Y$, a qual é dada por $Y = \frac{M-M^T}{2}$.

Observação 5.2.1. Se M é uma matriz anti-simétrica, então a forma quadrática $y^T My$ é nula para todo $y \in \Re^n$. Isto decorre diretamente da Eq.(5.30), uma vez que da última igualdade desta equação, segue que $y^T My = -y^T My$.

Teorema 5.2.1. *Seja $\delta\Gamma$ uma pequena matriz de erro adicionada à matriz de covariância $\Gamma(\cdot)$ no instante de recursão $k-1$ e defina*

$$\widetilde{\Gamma}(k-1) = \Gamma(k-1) + \delta\Gamma. \tag{5.32}$$

Suponha

$$\delta\Gamma = \Gamma_S + \Gamma_{AS}, \tag{5.33}$$

em que Γ_S e Γ_{AS} são as componentes simétricas e anti-simétricas de $\delta\Gamma$. Supondo que não existem erros adicionais na matriz de covariância no próximo instante de recursão k, então a matriz de erro $\delta\Gamma$ se propaga para o instante k de acordo com a seguinte equação

$$\begin{aligned}\Delta\Gamma(k) = \mu\Gamma(k)\Phi(k-1)\Gamma_S\Phi(k-1)\Gamma(k) + \tfrac{1}{\mu}\Gamma_{AS} + \\ + \tfrac{1}{\mu}[L(k)\phi^T(k)\Gamma_{AS} + \Gamma_{AS}^T\phi(k)L^T(k)].\end{aligned} \tag{5.34}$$

Demonstração:

Substituindo o vetor de ganho de Eq.(5.10) na recursão (5.11) e utilizando a matriz modificada de Eq.(5.32), obtém-se

$$\begin{aligned}\widetilde{\Gamma}(k) = \tfrac{1}{\mu}\{\Gamma(k-1) + \delta\Gamma - \tfrac{1}{\mu+\phi^T(k)[\Gamma(k-1)+\delta\Gamma]\phi(k)} \\ [\Gamma(k-1) + \delta\Gamma]\phi(k)\phi^T(k)[\Gamma(k-1) + \delta\Gamma^T]\}.\end{aligned} \tag{5.35}$$

Considere a seguinte aproximação

$$\begin{aligned}\tfrac{1}{\mu+\phi^T(k)[\Gamma(k-1)+\delta\Gamma]\phi(k)} \approx \\ \tfrac{1}{\mu+\phi^T(k)\Gamma(k-1)\phi(k)}\left[1 - \tfrac{\phi^T(k)\delta\Gamma\phi(k)}{\mu+\phi^T(k)\Gamma(k-1)\phi(k)}\right].\end{aligned} \tag{5.36}$$

Substituindo esta aproximação na Eq.(5.35), usando a Eq.(5.33), desprezando os termos de segunda ordem e considerando que o produto $\phi^T(k)\Gamma_{AS}\phi(k)$ é um escalar nulo, obtém-se

$$\begin{aligned}\widetilde{\Gamma}(k) &= \Gamma(k) + \tfrac{1}{\mu}[\delta\Gamma - L(k)\phi^T(k)\delta\Gamma^T - \delta\Gamma\phi(k)L^T(k) + \\ &\qquad L(k)\phi^T(k)\Gamma_S\phi(k)L^T(k)] \\ &= \Gamma(k) + \tfrac{1}{\mu}[I - L(k)\phi^T(k)]\Gamma_S[I - \phi(k)L^T(k)] + \\ &\qquad + \tfrac{1}{\mu}\Gamma_{AS} + \tfrac{1}{\mu}[L(k)\phi^T(k)\Gamma_{AS} + \Gamma_{AS}^T\phi(k)L^T(k)].\end{aligned} \tag{5.37}$$

Usando a equação

$$[I - L(k)\phi^T(k)] = \mu\Gamma(k)\Phi(k-1),$$

obtém-se finalmente o modelo de propogação do erro de Eq.(5.34). Se a sequência de regressores $\phi(k)$ é estacionária e $\mu < 1$, a aproximação

$$\Gamma(k)\Phi(k-1) \approx I \tag{5.38}$$

pode ser usada em regime permanente na Eq.(5.34) (Eleftheriou e Falconer, 1986). Desta forma, esta equação sugere que com respeito às componentes simétrica e anti-simétrica de $\delta\Gamma$, a componente Γ_S produz uma perturbação que decai exponencialmente com o tempo, enquanto que a componente Γ_{AS} leva à uma perturbação que cresce sem limites. A base do decrescimento exponencial é igual ao fator de esquecimento. O Teorema 5.2.1 é uma reformulação do trabalho de (Romano, 1995), onde o autor utiliza as definições das componentes Hermitiana e Anti-Hermitiana de uma matriz.

Como uma consequência do Teorema 5.2.1, supondo que a matriz de erro $\delta\Gamma$ seja puramente simétrica ($\Gamma_{AS} = 0$), e que seja aplicada na matriz $\Gamma(\cdot)$ no instante k_0, tem-se que a perturbação observada em $k > k_0$ é dada por

$$\Delta\Gamma(k) = \mu^{k-k_0}[\Gamma(k)\Phi(k_0)\Gamma_S\Phi(k_0)\Gamma(k)]. \tag{5.39}$$

Considerando novamente uma sequência estacionária de regressores $\phi(k)$ e $\mu < 1$, tem-se, na condição de regime, $\Gamma(k)\Phi(k_0) \approx I$. A Eq.(5.39) demonstra que a perturbação resultante de uma matriz de erro simétrica tende a desaparecer com o tempo. Além disso, enquanto seus elementos decaem, a matriz de perturbação

se mantém simétrica, não perdendo suas características de simetria. Portanto, $\Gamma(k)$ também se mantém rigorosamente simétrica.

Por outro lado, quando a matriz de erro $\delta\Gamma$ possui uma parte anti-simétrica ($\Gamma_{AS} \neq 0$), a matriz $\Gamma(k)$ é perturbada não apenas por uma componente anti-simétrica, que cresce com $1/\mu$, mas também por uma componente simétrica, correspondente ao último termo do segundo membro da Eq.(5.34). A primeira perturbação não altera a definição positiva da matriz $\Gamma(k)$, mas a segunda, que evolui de uma forma acoplada com a primeira, é a grande responsável pela alteração da definição positiva da matriz $\Gamma(k)$. Portanto, o artifício necessário para manter o algoritmo RLS estável consiste em se garantir que a matriz $\Gamma(k)$ permaneça simétrica ao longo do tempo.

O próximo teorema, o qual é uma contribuição desta tese, fornece uma análise da propagação de uma matriz de erro em esquemas RLS-UDU^T, motivando o método de fatoração UDU^T como uma forma de preservar a simetria da matriz $\Gamma(k)$. Nesta análise, erros de arredondamento locais possivelmente ocorridos durante as recursões não são considerados, mas somente a propagação de dois erros anteriores aplicados aos fatores $U(\cdot)$ e $D(\cdot)$ no instante de recursão $k-1$.

Teorema 5.2.2. *Sejam δU e δD matrizes de erro adicionadas aos fatores $U(\cdot)$ e $D(\cdot)$, respectivamente, no instante de recursão $k-1$. Assumindo que não existem erros de arredondamento locais nas matrizes $U(k)$ e $D(k)$, então, δU e δD induzem um erro $\Delta\Gamma(k)$ sobre a matriz de covariância $\Gamma(\cdot)$ no instante k como segue:*

$$\begin{aligned}\Delta\Gamma(k) = \mu^{-1}[&U(k-1)\delta DU^T(k-1) + \delta U D(k-1)U^T(k-1) + \\ &+ \delta U\delta DU^T(k-1) - \beta^{-1}\delta U g g^T U^T(k-1) + \\ &+ U(k-1)D(k-1)\delta U^T + U(k-1)\delta D\delta U^T - \\ &- U(k-1)\beta^{-1} g g^T \delta U^T + \delta U D(k-1)\delta U^T + \\ &+ \delta U\delta D\delta U^T - \beta^{-1}\delta U g g^T \delta U^T].\end{aligned} \tag{5.40}$$

Demonstração:

Seguindo o mesmo raciocínio do Teorema 5.2.1, defina as seguintes quantidades errôneas

$$\widetilde{U}(k-1) = U(k-1) + \delta U \tag{5.41}$$

$$\widetilde{D}(k-1) = D(k-1) + \delta D. \tag{5.42}$$

Substituindo os erros de Eqs.(5.41) e (5.42) na recursão (5.13), obtém-se

$$\widetilde{\Gamma}(k) = \mu^{-1}[U(k-1) + \delta U][D(k-1) + \delta D - \beta^{-1} g g^T][U(k-1) + \delta U]^T. \tag{5.43}$$

Desenvolvendo este produto, tem-se

$$\begin{aligned}\widetilde{\Gamma}(k) = \mu^{-1}[&U(k-1)D(k-1)U^T(k-1) + U(k-1)\delta D U^T(k-1) - \\ &- U(k-1)\beta^{-1} g g^T U^T(k-1) + \delta U D(k-1)U^T(k-1) + \\ &+ \delta U \delta D U^T(k-1) - \beta^{-1}\delta U g g^T U^T(k-1) + \\ &+ U(k-1)D(k-1)\delta U^T + U(k-1)\delta D \delta U^T - \\ &- U(k-1)\beta^{-1} g g^T \delta U^T + \delta U D(k-1)\delta U^T + \\ &+ \delta U \delta D \delta U^T - \beta^{-1}\delta U g g^T \delta U^T].\end{aligned} \tag{5.44}$$

Baseada novamente na recursão de Eq.(5.13), observe que a expressão $U(k-1)$ $D(k-1)U^T(k-1) - U(k-1)\beta^{-1} g g^T U^T(k-1)$ que aparece entre colchetes na Eq.(5.44) é dada por $\mu\Gamma(k)$, obtendo-se, desta forma, o modelo de propagação de erro de Eq.(5.40). De acordo com este modelo, a propagação de uma única matriz de erro em esquemas RLS-UDU^T apresenta propriedade de estabilidade numérica diferente daquela caracterizada pelo Teorema 5.2.1 para esquemas RLS convencionais. O modelo de erro de Eq.(5.40) indica que os esquemas de estimação RLS-UDU^T são estáveis uma vez que a simetria da matriz de covariância $\Gamma(k)$ não é perdida.

Uma análise teórica sobre o fenômeno da propagação de erros de arredondamento da implementação de esquemas de estimação RLS é realizada no trabalho de Verhaegen (Verhaegen, 1989), onde é mostrado que sob certas condições, a preservação da simetria de $\Gamma(k)$ garante que esta se mantém definida positiva, não obstante o acúmulo de erros. Alí, o autor estabelece uma condição suficiente para manter a propriedade definida positiva da matriz de covariância $\Gamma(k)$, que é dada pelo seguinte teorema:

Teorema 5.2.3. *Se a matriz de erro* $\delta\Gamma$ *aplicada à matriz* $\Gamma(k-1)$ *é simétrica e preserva sua propriedade definida positiva, então a matriz* $\widetilde{\Gamma}(k)$ *atualizada pela equação* (5.11) *permanece definida positiva.*

Demonstração:

Motivado pela definição de ser definida positiva uma dada matriz simétrica, o seguinte produto será examinado

$$z^T\widetilde{\Gamma}(k)z, \tag{5.45}$$

para algum vetor $z \neq 0$. De acordo com a equação de atualização (5.11) e da definição do vetor de ganho de Eq.(5.10), a Eq.(5.45) pode ser expressa como

$$\mu^{-1}\left[z^T\widetilde{\Gamma}(k-1)z - \frac{z^T\widetilde{\Gamma}(k-1)\phi(k)\phi^T(k)\widetilde{\Gamma}(k-1)z}{\mu + \phi^T(k)\widetilde{\Gamma}(k-1)\phi(k)}\right], \tag{5.46}$$

que é equivalente à

$$\begin{aligned} z^T\widetilde{\Gamma}(k-1)z + \mu^{-1}(z^T\widetilde{\Gamma}(k-1)z\phi^T(k)\widetilde{\Gamma}(k-1)\phi(k) - \\ - z^T\widetilde{\Gamma}(k-1)\phi(k)\phi^T(k)\widetilde{\Gamma}(k-1)z). \end{aligned} \tag{5.47}$$

Uma vez que $\widetilde{\Gamma}(k-1)$ é definida positiva, seu fator Cholesky $\widetilde{\Gamma}_{k-1}^{1/2}$ existe. Desta forma, pode-se definir

$$\sigma = \widetilde{\Gamma}^{1/2}(k)z \tag{5.48}$$

$$\tau = \widetilde{\Gamma}^{1/2}(k)\phi(k), \tag{5.49}$$

o qual reduz (5.47) a

$$\|\sigma\|_2^2 + \mu^{-1}\left\{\|\sigma\|_2^2\|\tau\|_2^2 - \left(\sigma^T\tau\right)^2\right\}. \tag{5.50}$$

Uma vez que $\sigma^T\tau = \|\sigma\|_2\|\tau\|_2\cos(\alpha)$, o segundo termo em (5.50) é maior ou igual à zero, e daí é estritamente positiva, o que conclui a prova.

5.3 Algoritmos IPA e Aproximadores RLS$_\mu$-HDP

Nesta seção, o processo de iteração de política aproximada (IPA), Subseção 5.3.1, é orientado para a atualização da política de decisão ou controle em esquemas HDP. A estratégia IPA é baseada em diferenças temporais e na estimação paramétrica RLS para aprendizagem da função valor V. O cálculo da função valor V^h para uma dada política h demanda somente amostragem da recompensa instantânea r_k e dos estados. Um desafio da abordagem é que os modelos f do ambiente e r da recompensa são necessários para calcular a função valor V correspondente à política ótima. Desta forma, a função valor V correspondente à política ótima é estimada durante o processo iterativo. Nas Subseções 5.3.2 e 5.3.3, apresenta-se o desenvolvimento dos algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR, respectivamente.

5.3.1 Algoritmo IPA com Função Valor V

Neste quadro de aproximações, a função valor V^h é substituída por uma aproximação paramétrica $\widehat{V}^h : X \times \Re^{n_\theta} \to \Re$, em que $\theta \in \Re^{n_\theta}$ é o vetor de parâmetros ajustáveis para minimizar em algum sentido a diferença temporal que é dada por

$$\Delta_k = r(x_k, h(x_k)) + \gamma \widehat{V}^h(x_{k+1}, \theta) - \widehat{V}^h(x_k, \theta). \tag{5.51}$$

Os valores Δ_k podem ser vistos como uma indicação de ajuste para corresponder à equação de *Bellman* (4.1) de modo que Δ_k precisa tender para zero.

O algoritmo IP consiste de um processo em que, para cada iteração j, dados a política de controle atual h_j e o vetor de parâmetros θ^{h_j} correspondente à h_j, então uma nova política de controle h_{j+1} é calculada por

$$h_{j+1}(x_k) = \arg \max_{u \in U(x_k)} \{r(x_k, u) + \gamma \widehat{V}^{h_j}(x_{k+1}, \theta^{h_j})\}. \tag{5.52}$$

Tal política é uma política gulosa com respeito a $\widehat{V}^{h_j}$, pois ela escolhe a ação que determina o maior conjunto possível de recompensas, de acordo com a equação da otimalidade de *Bellman*.

As principais etapas do algoritmo IP geral para a função valor são apresentadas no algoritmo 3 que é chamado Algoritmo de Iteração de Política Geral. O algoritmo tem etapas ativas que pertencem aos blocos de avaliações e melhorias de políticas.

Algoritmo 3 - Algoritmo de Iteração de Política Geral

1 ▷ - **Bloco 0 - Inicialização**
2 Selecione - $\forall$ Política Admissível - h_0.
3 Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
4 $j \leftarrow 0$
5 ▷ - **Processo Iterativo**
6 Para cada iteração j de política
7 **do**
8 ▷ - **Bloco 1 - Avaliação de Política Aproximada**
9 Cálculo da Função Valor Aproximada $\widehat{V}^{h_j}$
10 ▷ - **Bloco 2 - Melhoria de Política**
11 Estabeleça - Política Gulosa Melhorada h_{j+1} of Eq.(5.52).
12 **until** h_{j+1} seja satisfatória
13 **then** Fim do Processo Iterativo
14 **Fim do Algoritmo 3**.

Na etapa 9, a avaliação aproximada de política é calculada baseada na estimação do parâmetro θ^{h_j} de modo que a diferença temporal Δ_k seja minimizada. A etapa 11 implementa a política gulosa melhorada h_{j+1} como uma função da aproximação $\widehat{V}^{h_j}(x_{k+1}, \theta^{h_j})$. A inicialização do processo de IP aproximada requer a escolha de uma política inicial h_0 admissível, isto é, uma política estabilizante e que produz um valor $\widehat{V}^{h_0}(x_k)$ finito.

5.3.2 Algoritmo RLS$_\mu$-HDP-DLQR

A idéia principal por trás do método RLS$_\mu$-HDP para o DLQR é a estimação da função de custo V^{K_j} para uma dada política K_j que requer somente amostras da recompensa instantânea r_k, enquanto os modelos do ambiente e da função de utilidade são necessários para determinar a função de custo correspondente à política ótima.

Inicialmente, considera-se que o problema de aproximação da função de custo DLQR pode ser formulado por uma representação paramétrica $\widehat{V}^{K_j} : X \times \Re^{n_\theta} \rightarrow \Re$, sendo θ_j o vetor de parâmetros ajustáveis para estimar V^{K_j}, dada por

$$V^{K_j}(x_k) = \varphi^T(x_k)\theta_j, \tag{5.53}$$

e, portanto, precisa satisfazer a condição de consistência que é dada por

$$V^{K_j}(x_k) = r(x_k, h_j(x_k)) + \gamma V^{K_j}(f(x_k, h_j(x_k))). \tag{5.54}$$

A forma quadrática da função de custo, Eq.(4.19), é representada em termos do produto de *Kronecker* de estados e em termos da vetorização da matriz P que é a solução da equação de *Hamilton-Jacobi-Bellman*, Eq.(3.34). A solução ótima em sua forma vetorizada é dada por

$$V^{K_j}(x_k) = x_k^T P_j x_k = \overline{x}_k^T vec(P_j), \tag{5.55}$$

sendo $\overline{x}_k \in \Re^{n(n+1)/2}$ um vetor que é definido de acordo com o produto de *Kronecker* dado por

$$\begin{aligned} \overline{x}_k^T &= x_k^T \otimes x_k^T & (5.56)\\ &= [x_{1,k}^2 \ \ldots \ x_{1,k}x_{n,k} \ x_{2,k}^2 \ \ldots \ x_{n-1,k}x_{n,k} \ x_{n,k}^2], & (5.57) \end{aligned}$$

e $vec(P_j) \in \Re^{n(n+1)/2}$ resulta da vetorização. Este vetor tem os n elementos diagonais de P_j e as $\frac{n(n+1)}{2} - n$ somas não-repetidas $p_{rs} + p_{sr}$. Os elementos de $\overline{x}_k$ e $vec(P_j)$ são ordenados de modo a representar a forma quadrática de Eq.(5.55).

A Eq.(5.54) é escrita em termos da recompensa instantânea e substituindo o custo $V^{K_j}(x_k)$ pelo lado direito da Eq.(5.55), obtém-se uma relação para o cálculo da recompensa dada por

$$\begin{aligned} r(x_k, h_j(x_k)) &= \overline{x}_k^T vec(P_j) - \gamma \overline{x}_{k+1}^T vec(P_j) & (5.58)\\ &= \phi_k^T \theta_j, & (5.59) \end{aligned}$$

sendo $\phi_k = \overline{x}_k - \gamma \overline{x}_{k+1}$.

Observa-se que a recompensa instantânea $r_{(.)}$ é o valor desejado para estimação do vetor de parâmetros θ_j no sentido de mínimos quadrados. O algoritmo recursivo para estimação de θ_j gera uma sequência $\{\widehat{\theta}_j(i)\}_{i=1}^N$ de vetores que resolve N subproblemas parciais de mínimos quadrados envolvendo os i-ésimos termos quadráticos, isto é

$$\widehat{\theta}_j(i) = \arg\min_{\theta} \sum_{m=1}^{i} \mu^{i-m} \left| r_{k-i+m} - \phi_{k-i+m}^T \theta \right|^2, \tag{5.60}$$

sendo μ o fator de esquecimento e N é a quantidade de dados amostrados.

Realizando-se a minimização na Eq.(5.60), obtém-se

$$\widehat{\theta}_j(i) = \Phi_j^{-1}(i)\Theta_j(i), \tag{5.61}$$

sendo

$$\Phi_j(i) = \sum_{m=1}^{i} \mu^{i-m} \phi_{k-i+m} \phi_{k-i+m}^T \tag{5.62}$$

e

$$\Theta_j(i) = \sum_{m=1}^{i} \mu^{i-m} \phi_{k-i+m} r_{k-i+m} \ . \tag{5.63}$$

Por manipulações algébricas simples nas Eqs.(5.61)-(5.63) e aplicação do Lema da Inversão de Matriz, a estimação recursiva de θ_j é dividida em três equações que são a matriz de ganho, o vetor de parâmetros e a inversa da matriz $\Phi_j(i)$, a qual é denotada por $\Gamma_j(i)$. O ganho RLS na etapa i é dado por

$$L_j(i) = \Gamma_j(i)\phi_k = \frac{\Gamma_j(i-1)\phi_k}{\mu + \phi_k^T \Gamma_j(i-1)\phi_k} \ , \tag{5.64}$$

sendo j o índice de atualização de política, i é o índice da recorrência atual do estimador RLS, k é o tempo discreto. A estimação de parâmetro na etapa i é dada por

$$\widehat{\theta}_j(i) = \widehat{\theta}_j(i-1) + L_j(i)(r_k - \phi_k^T \widehat{\theta}_j(i-1)), \tag{5.65}$$

sendo $\theta_j = vec(P_j)$ o parâmetro verdadeiro para a função V^{K_j}, $\widehat{\theta}_j(i)$ é a i-ésima estimativa de θ_j. A matriz de covariância na etapa i é dada por

$$\Gamma_j(i) = \mu^{-1}\left[\Gamma_j(i-1) - L_j(i)\phi_k^T\Gamma_j(i-1)\right], \tag{5.66}$$

sendo $\Gamma_0 = \delta I$ para alguma constante positiva δ, $\Gamma_{j+1}(0) = \Gamma_j$ e $\Gamma_j(0) = \Gamma_0$.

Observa-se a partir da Eq.(5.65) que a estimativa $\widehat{\theta}_j(i)$ é obtida por uma correção da estimativa anterior $\widehat{\theta}_j(i-1)$ que é proporcional à diferença temporal $\Delta_k = r_k - \phi_k^T\widehat{\theta}_j$. As componentes do vetor de ganho L_j são ponderações da correção do parâmetro $\widehat{\theta}_j$ na Eq.(5.65). Observe que a correção é baseada sobre a aproximação da recompensa instantânea.

As equações para o desenvolvimento do algoritmo de iteração de política aproximada para o DLQR são baseadas na aproximação da solução da equação de *Bellman* (5.54) associada à política atual K_j conforme as regras que são estabelecidas para o incremento Δ_k nas Eqs.(5.64)-(5.66) da estimação RLS e no procedimento de melhoria de política de acordo com a Eq.(5.52). Então, tem-se na etapa de avaliação de política

$$\phi_k^T\theta_j = r(x_k, h_j(x_k)) = x_k^T(Q + K_j^T R K_j)x_k, \tag{5.67}$$

sendo $\phi_k = \overline{x}_k - \gamma\overline{x}_{k+1}$. Esta equação é resolvida para o parâmetro θ usando-se relações de recorrência RLS (5.64)-(5.66).

Em particular, o algoritmo busca por melhorias na política de decisão que são estabelecidas por

$$K_{j+1} = \gamma(R + \gamma B^T P^{\widehat{\theta}_j} B)^{-1} B^T P^{\widehat{\theta}_j} A. \tag{5.68}$$

O primeiro bloco do IP aproximado para o DLQR é constituído pelos parâmetros do sistema e o setup da simulação. O segundo bloco implementa as regras para avaliação de IP aproximada. O terceiro bloco implementa as melhorias da política. Essas três etapas são mostradas no algoritmo 4 que implementa o IP aproximado para o DLQR.

ALGORITMO 4 - IP APROXIMADO - DLQR

1 ▷ - **Bloco 0 - Inicialização**
2 ▷ Matrizes do Sistema Dinâmico e de Ponderação
3 $[Q, R, A, B] \leftarrow$ [ ]
4 Selecione - $\forall$ Política Admissível, K_0.
5 Selecione - Fator de Desconto, $0 < \gamma \leq 1$.
6 $j \leftarrow 0$
7 ▷ - **Processo Iterativo**
8 Para cada iteração j de política
9 **do**
10 ▷ - **Bloco 1 - Avaliação de Política Aproximada**
11 Cálculo da solução RLS via relações (5.64)-(5.66) para
12 $\phi_k^T \theta_j = r(x_k, h_j(x_k)) = x_k^T (Q + K_j^T R K_j) x_k$.
13 ▷ - **Bloco 2 - Melhoria de Política**
14 $K_{j+1} = \gamma (R + \gamma B^T P^{\widehat{\theta}_j} B)^{-1} B^T P^{\widehat{\theta}_j} A$.
15 **until** K_{j+1} seja satisfatória
16 **then** Fim do Processo Iterativo
17 **Fim do Algoritmo 4**.

Os parâmetros do sistema dinâmico, a ponderação da função de custo e o fator de desconto são associados com as etapas 3, 4 e 5, respectivamente. A regra de avaliação da etapa 12 fornece uma solução aproximada $\widehat{P}^{K_j}$ da equação de *Lyapunov* via relações de recorrência (5.64)-(5.66). A decisão de controle de etapa 14 é a melhoria de política. A decisão é dada pela lei de controle ótima de Eq.(5.68) que é a parametrização da Eq.(5.52).

O algoritmo RLS$_\mu$-HDP-DLQR possui dois laços principais que são dos processos iterativos da HDP e do RLS para determinação da política de decisão (controle) baseada nos métodos de aprendizado por reforço. Nesta situação o algoritmo customizado é formado pelos seguintes passos principais:

ALGORITMO 5 - RLS$_\mu$-HDP-DLQR

1 ——————————————————————————
2 ▷ - **Bloco 0 - Inicialização**
3 ▷ Matrizes do Sistema Dinâmico e de Ponderação
4 $[Q, R, A, B] \leftarrow [\quad\quad]$
5 Selecione - $\forall$ Política de Controle Admissível - K_0.
6 Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
7 Selecione - Estado Inicial - x_0.
8 ▷ Parâmetros do RLS
9 $\widehat{\theta}_0(0) \leftarrow \widehat{\theta}_0$
10 $\Gamma_0(0) \leftarrow \Gamma_0$
11 Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.
12 ——————————————————————————
13 $k \leftarrow 0$, $j \leftarrow 0$
14 ▷ - Processo Iterativo
15 Para cada iteração j de política
16 **do**
17 **for** $i \leftarrow 1 : N$
18 **do**
19 ——————————————————————————
20 ▷ **Bloco 1 - Simulação do Ambiente**
21 ▷ Ação de Controle
22 $u_k \leftarrow -K_j x_k$
23 ▷ Estados
24 $x_{k+1} \leftarrow A_d x_k + B_d u_k$
25 ——————————————————————————
26 ▷ **Bloco 2 - Avaliação de Política Aproximada**
27 ▷ Montagem do Alvo
28 $r_k \leftarrow x_k^T Q x_k + u_k^T R u_k$
29 ▷ Vetor de Funcões de Base - Produto de *Kronecker*
30 $\phi_k \leftarrow [x_{1,k}^2; x_{1,k}x_{2,k}; x_{1,k}x_{3,k}; x_{1,k}x_{4,k}; x_{2,k}^2;$
31 $x_{2,k}x_{3,k}; x_{2,k}x_{4,k}; x_{3,k}^2; x_{3,k}x_{4,k}; x_{4,k}^2]$
32 $-\gamma[x_{1,k+1}^2; x_{1,k+1}x_{2,k+1}; x_{1,k+1}x_{3,k+1};$
33 $x_{1,k+1}x_{4,k+1}; x_{2,k+1}^2; x_{2,k+1}x_{3,k+1};$
34 $x_{2,k+1}x_{4,k+1}; x_{3,k+1}^2; x_{3,k+1}x_{4,k+1}; x_{4,k+1}^2]$
35 Atualize $\widehat{\theta}_j(i)$ via relações de recorrência (5.64)-(5.66) do RLS
36 $k \leftarrow k + 1$
37 **End**
38 Associação - Matriz $P^{\widehat{\theta}_j}$ com $\widehat{\theta}_j$
39 $P^{\widehat{\boldsymbol{\theta}}_j} \leftarrow [\widehat{\theta}_{1,j} \quad \widehat{\theta}_{2,j}/2 \quad \widehat{\theta}_{3,j}/2 \quad \widehat{\theta}_{4,j}/2;$
40 $\widehat{\theta}_{5,j} \quad \widehat{\theta}_{6,j}/2 \quad \widehat{\theta}_{7,j}/2;$
41 $\widehat{\theta}_{8,j} \quad \widehat{\theta}_{9,j}/2;$
42 $\widehat{\theta}_{10,j}]$
43 ——————————————————————————
44 ▷ - **Bloco 3 - Melhoria de Política**
45 $K_{j+1} = \gamma(R + \gamma B^T P^{\widehat{\theta}_j} B)^{-1} B^T P^{\widehat{\theta}_j} A$
46 ——————————————————————————
47 ▷ Reinicialização - Parâmetros do RLS
48 $\widehat{\theta}_{j+1}(0) = \widehat{\theta}_j$
49 $\Gamma_{j+1}(0) = \Gamma_j$
50 **until** K_{j+1} seja satisfatória
51 **then** Fim do Processo Iterativo
52 **Fim do Algoritmo 5**.

O algoritmo RLS$_\mu$-HDP-DLQR (algoritmo 5) possui dois blocos principais que são os blocos de avaliações de políticas e melhorias de políticas. Neste algoritmo, a diferença principal, dentre outras, está no bloco de avaliações de políticas, em que o produto de *Kronecker* e as equações de estimação RLS são realizadas. O bloco 0 no algoritmo 5, passos 3-11, realiza a inicialização de parâmetros fixos do problema de otimização que são as ponderações Q e R, as matrizes do sistema dinâmico A e B, e o fator de desconto γ. Para o problema RLS, as condições iniciais são o fator de esquecimento, o parâmetro θ e a matriz Γ.

Os passos 21-28 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 r_k (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 29-35 do algoritmo realizam a vetorização dos estados e a aproximação da matriz P via estimação RLS. Os passos 38-42 realizam a associação da aproximação $P^{\widehat{\theta}_j}$ que é usada no passo 45 para realização de melhorias de políticas K_{j+1}.

5.3.3 Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR

Nesta seção, apresenta-se uma solução via métodos de fatoração UDU^T ao problema de convergência e estabilidade numérica que está relacionado com o mal condicionamento da matriz de covariância da abordagem RLS para aproximações da função valor em esquemas HDP. Especificamente, apresenta-se o algoritmo RLS$_\mu$-HDP-DLQR com fatoração UDU^T que é orientado para a avaliação de soluções aproximadas, numericamente estáveis, da equação HJB-*Riccati* associada ao problema DLQR.

O desenvolvimento do algoritmo RLS$_\mu$-HDP-DLQR com fatoração UDU^T é baseado nas Equações (5.69)-(5.80) que são agrupadas aos procedimentos algoritmicos anteriores para montar o algoritmo 6, chamado Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR. O bloco 2 realiza a avaliação aproximada de política baseada nas seguintes equações do estimador RLS com fatoração UDU^T

$$\varepsilon_j(i) = r_k - \phi_k^T \widehat{\theta}_j(i-1), \tag{5.69}$$

$$e = U_j^T(i-1)\phi_k, \tag{5.70}$$

$$g = D_j(i-1)e, \tag{5.71}$$

$$\beta_0 = \mu. \tag{5.72}$$

Para l de 1 a n_θ faz-se

$$\beta_l = \beta_{l-1} + e_l g_l, \tag{5.73}$$

$$d_l(i) = d_l(i-1)\frac{\beta_{l-1}}{\beta_l \mu}, \tag{5.74}$$

$$\kappa_l = g_l, \tag{5.75}$$

$$\tau_l = \frac{-e_l}{\beta_{l-1}}. \tag{5.76}$$

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

$$u_{ml}(i) = u_{ml}(i-1) + \tau_l \kappa_m, \tag{5.77}$$

$$\kappa_{m,new} = \kappa_{m,old} + u_{ml}(i-1)\kappa_l. \tag{5.78}$$

Atualização do vetor de ganho

$$L_j(i) = \frac{\left[\kappa_1 \;\; \kappa_2 \;\; \ldots \;\; \kappa_{n\theta}\right]^T}{\beta_{n_\theta}}. \tag{5.79}$$

Atualização do vetor de parâmetros

$$\widehat{\theta}_j(i) = \widehat{\theta}_j(i-1) + L_j(i)\varepsilon_j(i). \tag{5.80}$$

ALGORITMO 6 - RLS$_\mu$-UDU^T-HDP-DLQR

1 ——————————————————————
2 ▷ - **Bloco 0 - Inicialização**
3 ▷ Matrizes do Sistema Dinâmico e de Ponderação
4 $[Q, R, A, B] \leftarrow$ [ ]
5 Selecione - $\forall$ Política de Controle Admissível - K_0.
6 Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
7 Selecione - Estado Inicial - x_0.
8 ▷ Parâmetros do RLS
9 $\widehat{\theta}_0(0) \leftarrow \widehat{\theta}_0$
10 $D_0(0) \leftarrow D_0$
11 $U_0(0) \leftarrow U_0$
12 Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.
13 ——————————————————————
14 $k \leftarrow 0$, $j \leftarrow 0$
15 ▷ - Processo Iterativo
16 Para cada iteração j de política
17 **do**
18 **for** $i \leftarrow 1 : N$
19 **do**
20 ——————————————————————
21 ▷ **Bloco 1 - Simulação do Ambiente**
22 ▷ Ação de Controle
23 $u_k \leftarrow -K_j x_k$
24 ▷ Estados
25 $x_{k+1} \leftarrow A_d x_k + B_d u_k$
26 ——————————————————————
27 ▷ **Bloco 2 - Avaliação de Política Aproximada**
28 ▷ Montagem do Alvo
29 $r_k \leftarrow x_k^T Q x_k + u_k^T R u_k$
30 ▷ Vetor de Funções de Base - Produto de *Kronecker*
31 $\phi_k \leftarrow [x_{1,k}^2; x_{1,k}x_{2,k}; x_{1,k}x_{3,k}; x_{1,k}x_{4,k}; x_{2,k}^2;$
32 $x_{2,k}x_{3,k}; x_{2,k}x_{4,k}; x_{3,k}^2; x_{3,k}x_{4,k}; x_{4,k}^2]$
33 $-\gamma[x_{1,k+1}^2; x_{1,k+1}x_{2,k+1}; x_{1,k+1}x_{3,k+1};$
34 $x_{1,k+1}x_{4,k+1}; x_{2,k+1}^2; x_{2,k+1}x_{3,k+1};$
35 $x_{2,k+1}x_{4,k+1}; x_{3,k+1}^2; x_{3,k+1}x_{4,k+1}; x_{4,k+1}^2]$
36 Atualize $\widehat{\theta}_j(i)$ usando Eqs.(5.69)-(5.80) da estimação
37 do RLS com fatoração UDU^T
38 $k \leftarrow k + 1$
39 **End**
40 Associação - Matriz $P^{\widehat{\theta}_j}$ com $\widehat{\theta}_j$
41 $P^{\widehat{\boldsymbol{\theta}}_j} \leftarrow [\widehat{\theta}_{1,j} \quad \widehat{\theta}_{2,j}/2 \quad \widehat{\theta}_{3,j}/2 \quad \widehat{\theta}_{4,j}/2;$
42 $\widehat{\theta}_{5,j} \quad \widehat{\theta}_{6,j}/2 \quad \widehat{\theta}_{7,j}/2;$
43 $\widehat{\theta}_{8,j} \quad \widehat{\theta}_{9,j}/2;$
44 $\widehat{\theta}_{10,j}]$
45 ——————————————————————
46 ▷ - **Bloco 3 - Melhoria de Política**
47 $K_{j+1} = \gamma(R + \gamma B^T P^{\widehat{\theta}_j} B)^{-1} B^T P^{\widehat{\theta}_j} A$
48 ——————————————————————
49 ▷ Reinicialização - Parâmetros do RLS
50 $\widehat{\theta}_{j+1}(0) = \widehat{\theta}_j$
51 $D_{j+1}(0) = D_j$
52 $U_{j+1}(0) = U_j$
53 **until** K_{j+1} seja satisfatória
54 **then** Fim do Processo Iterativo
55 **Fim do Algoritmo 6**.

5.3.4 Custo Computacional

A princípio, complexidade computacional se refere à estimativa do esforço computacional (custo computacional) despendido para execução de um algoritmo que resolve um problema com entrada de tamanho n. Este esforço normalmente é medido por uma função da forma $O(f(n))$, a qual indica a ordem de complexidade de tempo de execução do algoritmo que está associado ao número de operações aritméticas realizadas pelo algoritmo em função da dimensão n da entrada do problema. Em alguns algoritmos a função de complexidade $f(n)$ pode relacionar mais de uma váriável, como $f(n,m)$, $f(n,m,p)$, etc. Algoritmos que requerem menos operações aritméticas são preferidos na prática pois ocasionam em custos computacionais menores.

Neste ponto, vale ressaltar uma outra questão, que é da estabilidade numérica. Sabe-se que imprecisões e erros de arredondamento são inevitáveis na implementação computacional de qualquer algoritmo. Algoritmos que sofrem de instabilidade numérica em face de tais erros não são adequados na prática. Os algoritmos RLS$_\mu$-HDP-DLQR quando implementados de acordo com as Eqs.(5.64)-(5.66), instabilidade numérica pode ser um problema uma vez que os erros de arredondamento e precisão finita podem levar à matriz de covariância $\Gamma(\cdot)$ da recursão de Eq.(5.66) perder sua positividade.

O problema da perda da positividade da matriz $\Gamma(\cdot)$ na implementação do algoritmo RLS$_\mu$-HDP-DLQR tem sido contornado empregando-se métodos do tipo raiz quadrada (Fatoração UDU^T) (Rêgo *et al.,* 2013). Esta abordagem tem um custo computacional menor com respeito ao número de operações aritméticas, requeridas na implementação do algoritmo, que se reflete na redução dos erros de arredondamento, tornando os algoritmos RLS$_\mu$-HDP-DLQR menos sensíveis à perda da positividade da matriz $\Gamma(\cdot)$.

Um dos compromissos interessantes aqui está em se selecionar apropriadamente os parâmetros do projeto RLS$_\mu$-HDP-DLQR para estimação da função valor de modo a melhorar o ajuste desejado embutido por Eq.(5.1) e ao mesmo tempo obter melhores estimativas em termos de velocidade de convergência. Por exemplo, existe uma compensação entre se usar um fator de esquecimento μ que rastreia mais rapidamente a verdadeira trajetória do parâmetro θ e que garanta um limite aceitável para estabilidade numérica de estimadores RLS$_\mu$-HDP-DLQR.

Tendo em vista as dicussões acima e com o objetivo de melhor avaliar a eficiência dos algoritmos propostos neste trabalho em termos de sua complexidade, quando estabilidade numérica e tempo de convergência são também levados em consideração, é natural se pensar em uma medida de complexidade que seja capaz de representar o compromisso entre esforço computacional, estabilidade numérica e tempo de convergência. Sob este ponto de vista, uma proposta de medida para avaliar a complexidade computacional é definida da seguimte forma:

$$\Delta v_c : (n, \rho, O, \zeta, \psi) \rightarrow \Re, \tag{5.81}$$

sendo Δv_c uma função que mapeia a cada quádrupla $(n, \rho, O, \zeta, \psi)$ um número real, em que ρ representa o tipo de fatoração utilizada no processo recursivo do algoritmo (por exemplo, ρ pode representar as formas de fatoração UDU^T e QR na decomposição da matriz de covariância da abordagem RLS), $O = O(f_c(n, n_e, n_\theta, \rho))$ indica a quantidade de operações aritméticas realizadas pelo algoritmo em função da dimensão n do vetor de estado, da dimensão n_e do vetor de entrada de controle do sistema, da dimensão $n_\theta(n, n_e)$ do vetor de funções de base e do parâmetro ρ (observe a Tabela 5.1 que apresenta o mapeamento da influência do parâmetro ρ na complexidade numérica dos algoritmos RLS$_\mu$-HDP-DLQR na etapa de avaliação de política), $\zeta(\pi)$ é o número de laços do algoritmo definido pelo tipo de política π adotada (iteração de política padrão, iteração de política gulosa, iteração de valor), e ψ representa um conjunto de parâmetros preponderantes, tais como fator de esquecimento μ, fator de desconto γ e parâmetro λ (bem como parâmetros não modelados presentes no processo recursivo), que controlam estabilidade e taxa de convergência do algoritmo. Um mapeamento dos parâmetros μ, γ e λ na função Δv_c é definido pela seguinte composição:

$$\psi_B \circ \psi_A : (\mu, \gamma, \lambda) \rightarrow \{0, 0.5, 0.8, 1\}, \tag{5.82}$$

sendo ψ_A uma função que avalia a qualidade do algoritmo em termos de estabilidade numérica e velocidade de convergência, a qual pode ser determinada por meio da observação do comportamento de convergência do algoritmo ilustrado em gráficos e tabelas. A função ψ_B representa os escores que são estabelecidos afim de quantificar a qualidade do algoritmo que segue o seguinte critério de pontuação: se o desempenho do algoritmo for ótimo atribui-se o escore 1; se for bom

atribui-se o escore 0.8; se for médio atribui-se 0.5, e ruim atribui-se o valor 0.

O custo computacional do algoritmo RLS$_\mu$-HDP-DLQR (Algoritmo 5) por cada iteração i da etapa de avaliação de política realizada pelo RLS sem fatoração UDU^T é $O(2n_\theta^3 + 6n_\theta^2 + 9n_\theta + 1)$, ou simplesmente $O(n_\theta^3)$, considerando um *flop* como uma operação de ponto flutuante (adição, multiplicação e divisão), medida de complexidade aritmética adotada em (Golub e Charles, 1996). Além disso, o custo computacional associado com a recuperação da matriz P a partir do vetor $\widehat{\theta}_j$ gerado pelo RLS é dado por $O(n^2 - n)$ por cada iteração j de iteração de política. O custo computacional em uma iteração j da etapa de melhoria de política é dado por $O(\frac{1}{3}n_e^3 + 6n_e n^2 + 4n_e^2 n + 2n_e^2 - 4n_e n)$. Para mais detalhes do número de *flops* obtidos, veja Apêndice E.

Os Algoritmos RLS$_\mu$-UDU^T-HDP-DLQR requerem um custo computacional menor com respeito às operações matemáticas realizadas em uma iteração i do "loop"de avaliação de política. Os resultados são apresentados na Tabela 5.1 em termos das operações de ponto flutuante para os algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR. O número de *flops* necessários para a construção do vetor de funções de base e para a recuperação da matriz P é equivalente ao do algoritmo RLS$_\mu$-HDP-DLQR. Como também a realização da etapa de melhoria de política que necessita do mesmo número de flops.

Tabela 5.1: Complexidade - Etapa de Avaliação de Política

	Complexidade Numérica	
Operações	RLS$_\mu$-HDP	RLS$_\mu$-UDU^T-HDP
Adições	$n_\theta^3 + 2n_\theta^2 + 2n_\theta$	$\frac{3}{2}n_\theta^2 + \frac{5}{2}n_\theta$
Multiplicações	$n_\theta^3 + 4n_\theta^2 + 7n_\theta$	$\frac{3}{2}n_\theta^2 + \frac{19}{2}n_\theta$
Divisões	1	$2n_\theta + 1$

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

O método de programação dinâmica heurística dependente de ação (ADHDP) somente requer amostras da recompensa instantânea r_k para realizar aproximações da função de custo para qualquer política ótima ou não ótima. A estrutura de

aproximação é similar àquela da HDP de Eq.(5.55). Os conceitos de produto de *Kronecker* e vetorização são aplicados na matriz de aprendizagem Q_{LV}. A aproximação de função de custo é dada por

$$Q_L^K(x_k, u_k) = z_k^T Q_{LV}^K z_k = \overline{z}_k^T vec(Q_{LV}^K), \tag{5.83}$$

sendo $\overline{z}_k \in \Re^{(n+n_e)(n+n_e+1)/2}$ definido de acordo com o produto de *Kronecker* que é dado por

$$\begin{aligned} \overline{z}_k^T &= z_k^T \otimes z_k^T & (5.84)\\ &= [z_{1,k}^2 \ \ldots \ z_{1,k}z_{n+n_e,k} \ z_{2,k}^2 \ z_{2,k}z_{3,k} \ \ldots \ z_{n+n_e-1,k}z_{n+n_e,k} \ z_{n+n_e,k}^2], & (5.85) \end{aligned}$$

sendo z_{lk} a l-ésima, $l = 1, 2, ..., n + n_e$, componente do vetor $z_k = [x_k^T \ u_k^T]^T$, e $vec(Q^K) \in \Re^{(n+n_e)(n+n_e+1)/2}$ a vetorização da matriz Q^K. Este vetor possui os $n + n_e$ elementos diagonais e os $\frac{(n+n_e)(n+n_e+1)}{2} - n$ elementos fora da diagonal da matriz Q^K, respectivamente. Os últimos elementos são as somas não-repetidas $Q_{ij}^K + Q_{ji}^K$. Os elementos de $\overline{z}_k$ e $vec(Q_{LV}^K)$ são ordenados de modo a representar a forma quadrática de Eq.(5.83).

Substituindo-se a vetorização de Eq.(5.83) na equação de *Bellman* dada por Eq.(4.8), em termos de recompensa instantânea para o projeto DLQR, obtém-se a relação para a estimação de recompensa que é dada por

$$\begin{aligned} r(x_k, u_k) &= Q_L^K(x_k, u_k) - \gamma Q_L^K(x_{k+1}, Kx_{k+1}) & (5.86)\\ &= \overline{z}_k^T vec(Q^K) - \gamma \overline{z}_{k+1}^T vec(Q^K) & (5.87)\\ &= {\phi_k}^T \theta^K, & (5.88) \end{aligned}$$

sendo $\phi_k = \overline{z}_k - \gamma \overline{z}_{k+1}$ e $\theta^K = vec(Q^K)$. As relações de recorrência para estimação de θ^K são similares àquelas do algoritmo RLS$_\mu$-HDP-DLQR.

No algoritmo de iteração de política aproximada baseada sobre funções Q_L para o DLQR, o ator deve buscar por uma política melhorada que satisfaz a Eq.(4.34) por minimização de Eq.(4.33) com respeito à u. Após manipulações algébricas, a política de controle ótima é dada por

$$K_{j+1} = (\widehat{Q}_{uu}^{K_j})^{-1}\widehat{Q}_{ux}^{K_j}. \tag{5.89}$$

No contexto de programação dinâmica heurística, o algoritmo dependente de ação, baseado sobre a política de decisão ótima (lei de controle) que é dada pela Eq.(5.89), realiza as ações do ator sobre o esquema ator-crítico da aprendizagem por reforço. A estrutura básica do algoritmo RLS$_\mu$-ADHDP-DLQR é similar àquela do algoritmo RLS$_\mu$-HDP-DLQR. Entretanto, as principais diferenças estão na dimensão do produto de *Kronecker* e no cálculo direto de melhorias de política. Neste último, a solução P da equação HJB está ímplicita na função Q_L, como pode ser observado na Eq.(5.89). As principais etapas do procedimento ADHDP para o projeto DLQR são dadas no algoritmo RLS$_\mu$-ADHDP-DLQR (algoritmo 7).

ALGORITMO 7 - RLS_{μ}-ADHDP-DLQR

1 ——————————————————————————
2 ▷ - **Bloco 0 - Inicialização**
3 ▷ Matrizes do Sistema Dinâmico e de Ponderação
4 $[Q, R, A, B] \leftarrow [\quad]$
5 Selecione - $\forall$ Política de Controle Admissível - K_0.
6 Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
7 Selecione - Estado Inicial - x_0.
8 ▷ Parâmetros do RLS
9 $\widehat{\theta}_0(0) \leftarrow \widehat{\theta}_0$
10 $\Gamma_0(0) \leftarrow \Gamma_0$
11 Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.
12 ——————————————————————————
13 $k \leftarrow 0$
14 $j \leftarrow 0$
15 ▷ Processo Iterativo
16 Para cada iteração j de política
17 **do**
18 **for** $i \leftarrow 1 : N$
19 **do**
20 ——————————————————————————
21 ▷ **Bloco 1 - Simulação do Ambiente**
22 ▷ Ruído de Controle (componente de "exploração"do sinal de controle)
23 $e_k \leftarrow [\quad]$
24 ▷ Ação de Controle
25 $u_k \leftarrow -K_j x_k + e_k$
26 ▷ Estado Seguinte
27 $x_{k+1} \leftarrow Ax_k + Bu_k$
28 ▷ Ação de Controle Seguinte
29 $u_{k+1} \leftarrow -K_j x_{k+1}$
30 ——————————————————————————
31 ▷ **Bloco 2 - Avaliação de Política Aproximada**
32 ▷ Montagem do Alvo
33 $r_k \leftarrow x_k^T Q x_k + u_k^T R u_k$
34 ▷ Vetor de Funções de Base - Produto de Kronecker
35 $\phi_k \leftarrow [x_{1k}^2;$
36 $x_{1,k}x_{2,k}; x_{1,k}x_{3,k}$
37 $x_{1,i}u_{1,k}; x_{1,k}u_{2,k}$
38 $x_{2,k}^2; x_{2,k}x_{3,k}; x_{2,k}u_{1,k}$
39 $x_{2,k}u_{2,k}; x_{3,k}^2; x_{3,k}u_{1,k}$
40 $x_{3,k}u_{2,k}; u_{1,k}^2; u_{1,k}u_{2,k}$
41 $u_{2,k}^2] - \gamma[x_{1,k+1}^2;$
42 $x_{1,k+1}x_{2,k+1}; x_{1,k+1}x_{3,k+1}$
43 $x_{1,k+1}u_{1,k+1}; x_{1,k+1}u_{2,k+1}$
44 $x_{2,k+1}^2; x_{2,k+1}x_{3,k+1}; x_{2,k+1}u_{1,k+1}$
45 $x_{2,k+1}u_{2,k+1}; x_{3,k+1}^2; x_{3,k+1}u_{1,k+1}$
46 $x_{3,k+1}u_{2,k+1}; u_{1,k+1}^2; u_{1,k+1}u_{2,k+1}$
47 $u_{2,k+1}^2];$
48 Atualize $\widehat{\theta}_j(i)$ via relações de recorrência (5.64)-(5.66) do RLS
49 $k \leftarrow k + 1$
50 **End**
51 Associação - Matriz $Q^{\widehat{\theta}_j}$ com $\widehat{\theta}_j$
52 $Q^{\widehat{\theta}_j} \leftarrow [\widehat{\theta}_{1,j} \quad \widehat{\theta}_{2,j}/2 \quad \widehat{\theta}_{3,j}/2 \quad \widehat{\theta}_{4,j}/2 \quad \widehat{\theta}_{5,j}/2;$
53 $\widehat{\theta}_{6,j}, \quad \widehat{\theta}_{7,j}/2 \quad \widehat{\theta}_{8,j}/2 \quad \widehat{\theta}_{9,j}/2$
54 $\theta_{10,j}, \quad \widehat{\theta}_{11,j}/2 \quad \widehat{\theta}_{12,j}/2$
55 $\widehat{\theta}_{13,j} \quad \widehat{\theta}_{14,j}/2$
56 $\widehat{\theta}_{15,j}]$
57 ——————————————————————————
58 ▷ - **Bloco 3 - Melhoria de Política**
59 $K_{j+1} = (Q_{uu}^{\widehat{\theta}_j})^{-1} Q_{ux}^{\widehat{\theta}_j}$
60 ——————————————————————————
61 ▷ Reinicialização - Parâmetros do RLS
62 $\widehat{\theta}_{j+1}(0) = \widehat{\theta}_j$
63 $\Gamma_{j+1}(0) = \Gamma_j$
64 **until** K_{j+1} seja satisfatória
65 **then** Fim do Processo Iterativo
66 **Fim do Algoritmo 7**.

As inicializações e parâmetros do algoritmo são solicitados nos passos 1 à 10. Na etapa 34, mostra-se o produto de *Kronecker* do vetor de estado e entrada de controle. Este produto é usado na etapa 48 para atualizar o vetor de parâmetros, de acordo com as Eqs.(5.64)-(5.66) do RLS. A etapa 55 atualiza a política K_{j+1} que é baseada na etapa 48.

5.4.1 Convergência de RLS$_\mu$-ADHDP para o DLQR

Os resultados de convergência para esquemas ADHDP usando-se aproximadores RLS foram provados por Bradtke (Bradtke *et al.*, 1994) para problemas LQR discreto. O próximo Teorema pode ser encontrado em (Bradtke *et al.*, 1994) e mostra que a sequência $\{K_j\}_{j=1}^{\infty}$ de políticas gerada pelo esquema de iteração de política aproximada converge à política ótima.

Teorema 5.4.1. *Suponhamos que $\{A, B\}$ é um par controlável, K_0 é um controle estabilizante, e o vetor ϕ_k é excitado persistentemente segundo*

$$\varepsilon_0 I \leq \frac{1}{N}\sum_{i=1}^{N} \phi_{k-i}\phi_{k-i}^T \leq \overline{\varepsilon}_0 I, \tag{5.90}$$

para todo $k \geq N_0$, $N \geq N_0$, $\varepsilon_0 \leq \overline{\varepsilon}_0$, com ε_0 e $\overline{\varepsilon}_0$ inteiros positivos e N_0 uma constante positiva. Então, existe um intervalo de estimação $N < \infty$ de modo que o esquema de iteração de política aproximada gera uma sequência $\{K_j\}_{j=1}^{\infty}$ de controles estabilizantes, tal que

$$\lim_{j\to\infty} \|K_j - K^*\| = 0,$$

sendo K^* a matriz de controle de realimentação ótima.

5.5 Variante Otimística do Algoritmo RLS$_\mu$-HDP-DLQR

Na abordagem IP aproximada apresentada na Subseção 5.3.2, a política atual h_j é fixada e uma aproximação $\widehat{V}^{h_j}(\cdot, \theta)$ de V^{h_j} é construída usando-se as relações de recorrência RLS (5.64)-(5.66). Em uma iteração típica, a política h_j é

fixada por um período de tempo suficientemente longo até que a convergência pelo crítico seja alcançada. Em seguida, uma atualização de política é realizada determinando-se uma nova política h_{j+1}, a qual é gulosa com respeito à $\widehat{V}^{h_j}(\cdot,\theta)$, isto é,

$$h_{j+1}(x_k) = \arg\min_{u\in U(x_k)} \{r(x_k,u) + \gamma \widehat{V}^{h_j}(x_{k+1},\theta^{h_j})\}. \tag{5.91}$$

Nesta subseção, considera-se uma variante do algoritmo RLS$_\mu$-HDP-DLQR, em que as atualizações de política são realizadas baseadas em uma avaliação incompleta da política atual h_j. Uma atualização de política é realizada subsequente a cada atualização pelo algoritmo de avaliação de política. Mais precisamente, nesta estratégia, uma atualização de política é realizada subsequente à geração de cada transição de estado de x_k para x_{k+1}, calculando-se uma ação u_k que é gulosa com respeito à k-ésima estimativa de θ, $\widehat{\theta}_k$. Neste caso, as relações de recorrência RLS são dadas por

$$L_{k+1} = \Gamma_{k+1}\phi_k = \frac{\Gamma_k\phi_k}{\mu + \phi_k^T\Gamma_k\phi_k}, \tag{5.92}$$

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + L_{k+1}(r(x_k,u_k) - \phi_k^T\widehat{\theta}_k) \tag{5.93}$$

e

$$\Gamma_{k+1} = \mu^{-1}\left(\Gamma_k - \frac{\Gamma_k\phi_k\phi_k^T\Gamma_k}{\mu + \phi_k^T\Gamma_k\phi_k}\right), \tag{5.94}$$

sendo $u_k = \arg\min_{u\in U(x_k)} \{r(x_k,u) + \gamma\widehat{V}(x_{k+1},\widehat{\theta}_k)\}$, $\widehat{\theta}_k$ é a k-ésima estimativa de θ e $\Gamma(0) = \Gamma_0$, com $\Gamma_0 = \delta I$ para alguma constante positiva δ.

Bertsekas e Tsitsiklis, (Bertsekas e Tsitsiklis, 1996), propuseram algumas variantes otimísticas baseadas em TD(λ) do método de iteração de política em programação dinâmica aproximada com função *cost-to-go*, em que atualizam a política atual após cada transição de estado. Um estudo por Landelius, (Landelius e Knutsson, 1996), lida com críticos adaptativos gulosos para problemas LQR, em

que a política de controle ótima é determinada por *certainty equivalence control*. Provas de convergência são apresentadas para um número de críticos adaptativos usando iterações gulosas. Ambas as referências usam métodos do gradiente incremental para resolver o problema de mínimos quadrados associado com a aproximação *cost-to-go*, ao invés de métodos RLS.

O algoritmo 8 fornece as etapas correspondentes à variante otimística do algoritmo RLS$_\mu$-HDP-DLQR, denominado RLS$_\mu$-HDP-DLQR Otimístico, para determinar a política de controle ótima e é desenvolvido de acordo com as Eqs.(5.92)-(5.94) para o núcleo principal. Observa-se que neste algoritmo a política é atualizada em um único *loop*, em direção à política ótima em cada passo de tempo k. Em contraste ao algoritmo RLS$_\mu$-HDP-DLQR, em que um índice de iteração i é usado para avaliação de política, o qual funciona no *loop* interno de cada iteração de política j (*offline*). Uma vez que a convergência é alcançada pelo algoritmo de avaliação de política, a política é atualizada em um *loop* externo.

ALGORITMO 8 - RLS$_\mu$-HDP-DLQR OTIMÍSTICO

1 ——————————————————————————
2 ▷ - **Bloco 0 - Inicialização**
3 ▷ Matrizes do Sistema Dinâmico e de Ponderação
4 $[Q, R, A, B] \leftarrow [\quad\quad]$
5 ▷ Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
6 ▷ Parâmetros do RLS: $\widehat{\theta}_0$, Γ_0
7 Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.
8 ▷ Selecione - Estado Inicial - x_0.
9 ▷ Selecione - Política Inicial
10 $K_0 = \gamma(R + \gamma B^T P^{\widehat{\theta}_0} B)^{-1} B^T P^{\widehat{\theta}_0} A$.
11 ——————————————————————————
12 $k \leftarrow 0$
13 ▷ - Processo Iterativo
14 Para cada passo de tempo k
15 **do**
16 ——————————————————————————
17 ▷ **Bloco 1 - Simulação do Ambiente**
18 ▷ Ação de Controle
19 $u_k \leftarrow -K_k x_k$
20 ▷ Estados
21 $x_{k+1} \leftarrow Ax_k + Bu_k$
22 ——————————————————————————
23 ▷ **Bloco 2 - Atualização da Política e da Função Valor**
24 ▷ Montagem do Alvo
25 $r_k \leftarrow x_k^T Q x_k + u_k^T R u_k$
26 ▷ Vetor de Funções de Base - Produto de *Kronecker*
27 $\phi_k \leftarrow [x_{1,k}^2; x_{1,k}x_{2,k}; x_{1,k}x_{3,k}; x_{1,k}x_{4,k}; x_{2,k}^2;$
28 $x_{2,k}x_{3,k}; x_{2,k}x_{4,k}; x_{3,k}^2; x_{3,k}x_{4,k}; x_{4,k}^2]$
29 $-\gamma[x_{1,k+1}^2; x_{1,k+1}x_{2,k+1}; x_{1,k+1}x_{3,k+1};$
30 $x_{1,k+1}x_{4,k+1}; x_{2,k+1}^2; x_{2,k+1}x_{3,k+1};$
31 $x_{2,k+1}x_{4,k+1}; x_{3,k+1}^2; x_{3,k+1}x_{4,k+1}; x_{4,k+1}^2]$
32 Atualize $\widehat{\theta}_k$ via Relações de Recorrência (5.92)-(5.94) do RLS
33 Associação - Matriz $P^{\widehat{\theta}_{k+1}}$ com $\widehat{\theta}_{k+1}$
34 $P^{\widehat{\boldsymbol{\theta}}_{k+1}} \leftarrow [\widehat{\theta}_{1,k+1} \quad \widehat{\theta}_{2,k+1}/2 \quad \widehat{\theta}_{3,k+1}/2 \quad \widehat{\theta}_{4,k+1}/2;$
35 $\widehat{\theta}_{5,k+1} \quad \widehat{\theta}_{6,k+1}/2 \quad \widehat{\theta}_{7,k+1}/2;$
36 $\widehat{\theta}_{8,k+1} \quad \widehat{\theta}_{9,k+1}/2;$
37 $\widehat{\theta}_{10,k+1}]$
38 ▷ Atualização da Política
39 $K_{k+1} = \gamma(R + \gamma B^T P^{\widehat{\theta}_{k+1}} B)^{-1} B^T P^{\widehat{\theta}_{k+1}} A$
40 ——————————————————————————
41 $k \leftarrow k + 1$
42 **until** K_{k+1} seja satisfatória
43 **then** Fim do Processo Iterativo
44 **Fim do Algoritmo 8**.

Considerando-se as peculiaridades de cada algoritmo, uma variante otimística do algoritmo RLS$_\mu$-UDU^T-HDP-DLQR é desenvolvida em uma maneira similar à do algoritmo RLS$_\mu$-HDP-DLQR otimístico. As principais etapas do procedimento

otimístico para RLS$_\mu$-UDU^T-HDP-DLQR são dadas no algoritmo 10 (Apêndice C), chamado algoritmo RLS$_\mu$-UDU^T-HDP-DLQR otimístico.

5.6 Abordagem RLS para Aprendizagem TD(λ)

Para melhorar a eficiência de algoritmos TD(λ), Boyan (Boyan, 2002) propõe o algoritmo TD(λ) de mínimos quadrados, ou LSTD(λ), o qual converge para o mesmo coeficiente θ^* que TD(λ). Entretanto, ao invés de realizar descida do gradiente, LSTD(λ) constrói estimativas explicítas da matriz C e do vetor d associados à solução da Eq.(2.49), e então resolve a Eq.(2.49) diretamente. O problema de mínimos quadrados associado à Eq.(2.49) tem a seguinte função de perda

$$J(\theta, k) = \left\| \sum_{i=0}^{k} C(X_i)\theta + \sum_{i=0}^{k} d(X_i) \right\|^2, \tag{5.95}$$

sendo $\|\cdot\|$ a norma Euclidiana, e a estimativa de mínimos quadrados do vetor de pesos θ no passo de tempo k é dada por

$$\widehat{\theta}(k) = \Lambda^{-1}(k)\Theta(k), \tag{5.96}$$

sendo

$$\Lambda(k) = -\sum_{i=0}^{k} C(X_i) = \sum_{i=0}^{k} \xi_i \phi_i^T \tag{5.97}$$

e

$$\Theta(k) = \sum_{i=0}^{k} d(X_i) = \sum_{i=0}^{k} \xi_i r_i, \tag{5.98}$$

com $\phi_i = \varphi(x_i) - \gamma\varphi(x_{i+1})$. Para facilitar a computação iterativa do algoritmo LSTD(λ), a solução de mínimos quadrados, Eqs.(5.96)-(5.98), é reescrita em uma forma recursiva

$$L(k) = \Gamma(k)\xi_k = \frac{\Gamma(k-1)\xi_k}{1 + \phi_k^T \Gamma(k-1)\xi_k}, \tag{5.99}$$

$$\widehat{\theta}(k) = \widehat{\theta}(k-1) + L(k)(r_k - \phi_k^T \widehat{\theta}(k-1)) \tag{5.100}$$

e

$$\Gamma(k) = \left(\Gamma(k-1) - \frac{\Gamma(k-1)\xi_k \phi_k^T \Gamma(k-1)}{1 + \phi_k^T \Gamma(k-1)\xi_k}\right), \tag{5.101}$$

sendo $\Gamma(k) = \Lambda^{-1}(k)$, $\widehat{\theta}(k)$ a k-ésima estimativa de θ e $\Gamma(0) = \Gamma_0$, com $\Gamma_0 = \delta I$ para alguma constante positiva δ. O vetor de ganho $L(k)$ mantém o controle de quanto o vetor de parâmetros $\widehat{\theta}$ deve ser ajustado, a fim de modificar adequadamente estimativas anteriores quando uma diferença temporal $\Delta_k = r_k - \phi_k^T \widehat{\theta}$ é observada.

5.6.1 Convergência de RLS-TD(λ)

Na análise de convergência do método RLS-TD(λ) considera-se cadeias de Markov ergódicas (veja definição no Apêndice A, Seção A.4). Sob esta suposição lida-se com uma cadeia de Markov $\{x_k\}$, cujas probabilidades de estado estacionário $\pi(i)$ satisfazem

$$\pi^T M = \pi^T, \tag{5.102}$$

sendo M a matriz de probabilidade de transição de estados e π é o vetor cuja i-ésima componente é $\pi(i)$ (veja Teorema A.5.1 no Apêndice A). Seja $\mathrm{E}_0[\cdot]$ o operador esperança com respeito a esta distribuição. Defina a seguinte notação: $\Phi \in \Re^{card(X)\times n_\theta}$ é uma matriz cuja j-ésima coluna, $j = 1, \cdots, n_\theta$, é a função base φ_j da representação linear $\widehat{V}^h(x_k, \theta) = \varphi^T(x_k)\theta$ (ou na notação vetorial $\widehat{V}^h(\theta) = \Phi\theta$). A convergência de RLS-TD(λ) requer as seguintes suposições:

A.1 Os custos de transição r_k satisfazem $\mathrm{E}_0[r_k^2] < \infty$.

A.2 A matriz Φ tem rank coluna completo, isto é, o conjunto de funções $\{\varphi_j \mid j = 1, \ldots, n_\theta\}$ é linearmente independente. Além disso, para cada j, a função base φ_j satisfaz $\mathrm{E}_0[\varphi_j^2(x_k)] < \infty$.

A.3 A matriz $(\Gamma_0^{-1} + \frac{1}{k}\sum_{i=1}^{k}\xi_i\phi_i^T)$ é não-singular para todo tempo $k > 0$.

Sob as suposições A.1-A.3, tem-se o seguinte teorema de convergência para RLS-TD(λ)

Teorema 5.6.1. (Xu *et al.*, 2002) *Para uma cadeia de Markov ergódica que satisfaz as condições A.1-A.3, a estimativa assintótica encontrada pelo RLS-TD(λ) converge, com probabilidade 1, para θ^* determinado pela Eq.(2.49).*

Demonstração:

Inicialmente considera-se um processo estacionário $X_k = \{x_k, x_{k+1}, \xi_k\}$ como segue (construído similarmente à idéia de (Tsitsiklis e Roy, 1997)). Seja $\{x_k\}$ uma cadeia de Markov, $-\infty < k < \infty$, que evolui de acordo com a matriz de transição M e que já está em seu estado estacionário no sentido que $P_r\{x_k = i\} = \pi(i)$, para todo i e k. Dada qualquer trajetória amostral desta cadeia de Markov, define-se

$$\xi_k = \sum_{\tau=-\infty}^{k} (\gamma\lambda)^{k-\tau}\varphi(x_\tau). \tag{5.103}$$

Observe que ξ_k é construído tomando-se o processo estacionário $\varphi(x_k)$, cuja variância é finita (Suposição A.2). Seja $\mathrm{E}_0[\cdot]$ a esperança com respeito à distribuição de estado estacionário de X_k. Segue do Lema D.0.1 (Apêndice D) que $\mathrm{E}_0[\mathrm{C}(\mathrm{X_k})]$ e $\mathrm{E}_0[\mathrm{d}(\mathrm{X_k})]$ são dadas por expressões bem definidas e finitas. Além disso, de acordo com o Lema D.0.3 (Apêndice D), $\mathrm{E}_0[\mathrm{C}(\mathrm{X_k})]$ é definida negativa.

Agora verifica-se como o limite da estimativa $\widehat{\theta}_k$ determinada pelo RLS-TD(λ) se comporta quando $k \to \infty$. Tem-se

$$\begin{aligned}
\widehat{\theta}_k &= \left(\sum_{i=0}^{k}\xi_i\phi_i^T\right)^{-1}\sum_{i=0}^{k}\xi_i r_i \\
&= \left(\Gamma_0^{-1} + \sum_{i=1}^{k}\xi_i\phi_i^T\right)^{-1}\left(\Gamma_0^{-1}\theta_0 + \sum_{i=1}^{k}\xi_i r_i\right) \\
&= \left(\tfrac{1}{k}\Gamma_0^{-1} - \tfrac{1}{k}\sum_{i=1}^{k}C(X_i)\right)^{-1}\left(\tfrac{1}{k}\Gamma_0^{-1}\theta_0 + \tfrac{1}{k}\sum_{i=1}^{k}d(X_i)\right).
\end{aligned} \tag{5.104}$$

Uma vez que

$$\mathrm{E}_0[C(X_k)] = \lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{k} C(X_i), \tag{5.105}$$

$$\mathrm{E}_0[d(X_k)] = \lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{k} d(X_i) \tag{5.106}$$

e $-\mathrm{E}_0[C(X_k)]$ é invertível, segue-se que

$$\lim_{k\to\infty} \widehat{\theta}_k = (-\mathrm{E}_0[C(X_k)])^{-1}\mathrm{E}_0[d(X_k)] = \theta^*. \tag{5.107}$$

A condição especificada pela suposição A.3 pode ser satisfeita estabelecendo-se $\Gamma_0 = \delta I$ apropriadamente. De acordo com o Teorema 5.6.1, RLS-TD(λ) converge para o mesmo limite que TD(λ), dado pela Eq.(2.49). Assim, o limite de convergência pode ser caracterizado pelo seguinte teorema, cuja demonstração pode ser encontrada em (Tsitsiklis e Roy, 1997).

Teorema 5.6.2. *O limite de convergência θ^* determinado pela Eq.(2.49) é a única solução da equação*

$$\Pi T^{(\lambda)}(\Phi\theta^*) = \Phi\theta^*, \tag{5.108}$$

sendo Π o operador projeção sobre um subespaço S de aproximação que é gerado pelas funções base, isto é, $S = \{\Phi\theta \,|\, \theta \in \Re^{n_\theta}\}$, com respeito à uma norma Euclidiana ponderada $\|\cdot\|_D$ que é dada por

$$\|V\|_D^2 = V^T D V = \sum_{x_k \in X} \pi(x_k) V^2(x_k), \tag{5.109}$$

com D uma matriz diagonal cujos elementos são as probabilidades de estado estacionário $\pi(i)$ de x_k. $T^{(\lambda)}$ é um operador de Bellman multi-passos parametrizado pelo escalar $0 \leq \lambda \leq 1$ definido por

$$(T^{(\lambda)}\widehat{V}^h)(x_k) = (1-\lambda)\sum_{m=0}^{\infty}\lambda^m \mathrm{E}\left[\sum_{i=0}^{m}\gamma^i r_{k+i} + \gamma^{m+1}\widehat{V}^h(x_{k+m+1},\theta)\,|\,x_k\right] \quad (5.110)$$

para $\lambda < 1$, *e*

$$(T^{(1)}\widehat{V}^h)(x_k) = \mathrm{E}\left[\sum_{i=0}^{\infty}\gamma^i r_{k+i}\,|\,x_k\right] = V^h(x_k) \quad (5.111)$$

para $\lambda = 1$, *de modo que* $\lim_{\lambda\to 1}(T^{(\lambda)}V)(x_k) = (T^{(1)}V)(x_k)$.

Além disso, θ^* *satisfaz*

$$\left\|\Phi\theta^* - V^h\right\|_D \leq \frac{1-\lambda\gamma}{1-\gamma}\left\|\Pi V^h - V^h\right\|_D, \quad (5.112)$$

sendo $\Pi = \Phi(\Phi^T D\Phi)^{-1}\Phi^T D$.

Como discutido em (Tsitsiklis e Roy, 1997), o teorema acima mostra que a distância da função limite $\Phi\theta^*$ da função de valor verdadeiro V^h é limitada e o menor limite de erro de aproximação pode ser obtido quando $\lambda = 1$. Para cada $\lambda < 1$, o limite de fato se deteriora a medida que λ decresce. O pior limite é obtido quando $\lambda = 0$. Embora isso seja somente um limite, ele sugere fortemente que os valores mais altos de λ tendem a produzir aproximações mais exatas de V^h.

O Teorema 5.6.1 é válido para o caso em que as trajetórias de estado amostrais $\{x_k\}_{k=0}^{\infty}$ correspondem à uma política estacionária que é fixada ao longo do tempo. Contudo, no quadro de iteração de política otimística, este resultado precisaria ser estendido para o caso em que as trajetórias de estado são geradas por políticas gulosas melhoradas que são aplicadas a cada transição de estado x_k para x_{k+1}, com uma probabilidade positiva. Na Seção 6 são fornecidos alguns resultados de simulação referentes à iteração de política otimística baseada na abordagem RLS-TD(λ).

5.7 Considerações Finais

Neste capítulo, a caracterização e formulação do problema de estimação RLS para a solução da equação HJB-*Riccati* associada com o problema DLQR via ADP foram apresentados. 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*, o qual redunda em dificuldades numéricas na atualização das variáveis da recursão do RLS em ambientes de precisão limitada. As seguintes abordagens abrangem o método proposto para projeto *online* de controle ótimo:

1. Mínimos quadrados recursivos com fator de esquecimento.
2. Inserção da fatoração UDU^T no método RLS com fator de esquecimento.
3. Formulação do problema em termos da fatoração da matriz de ganhos da Eq.(5.12) que é obtida da Eq.(5.13). Como também, a Eq.(5.16) é definida em termos da fatoração UDU^T como

$$\Gamma(k, U, D, g, \beta) = \Phi^{-1}(k).$$

4. O número de condição e parâmetro de positividade que são avaliados para monitorar e guiar o processo de convergência e estabilidade numérica.

Os algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR e suas variantes otimísticas foram desenvolvidos para a solução *online* de sistemas de controle ótimo DLQR. Os algoritmos *online* consistem em uma abordagem RL ator-crítico baseada em estimação paramétrica RLS e versões aproximadas de métodos de iteração de política para solução da equação HJB-*Riccati* associada ao problema DLQR via programação dinâmica heurística.

Desenvolveu-se também uma abordagem baseada na fusão de métodos RLS com aprendizagem TD(λ), tendo em vista a formulação de estratégias que promovam a melhoria de funções valor associadas a sistemas de controle ótimo, bem como acelerar o processo de avaliação de políticas de controle DLQR.

Capítulo 6

Experimentos Computacionais

6.1 Introdução

Os procedimentos para avaliar o desempenho dos aproximadores RLS da função valor propostos neste trabalho são estabelecidos em dois tipos de experimentos computacionais: 1) Experimentos RLS$_\mu$-HDP-DLQR/RLS$_\mu$-UDU^T-HDP-DLQR, e 2) Experimentos RLS-TD(λ)-DLQR.

No primeiro, avalia-se a convergência e a estabilidade numérica dos algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR desenvolvidos com iteração de política otimística (gulosa) para solução *online* da equação HJB-*Riccati* do problema DLQR, bem como a adaptabilidade dos estimadores RLS$_\mu$-HDP sem fatoração UDU^T e com fatoração UDU^T em relação à variações paramétricas na planta. No segundo tipo, avalia-se o desempenho de métodos RLS-TD(λ) para solução *online* do DLQR por meio de estatísticas de primeira e segunda ordem para diferentes valores do parâmetro λ.

Na Figura 6.1 apresenta-se um diagrama do procedimento para avaliações do desempenho dos métodos de aproximação da função valor, propostos neste trabalho, os quais são orientados para viabilizar as implementações *online* de controlares ótimos para sistemas do mundo real. As avaliações verificam o comportamento de convergência dos estimadores RLS para a sintonia da HDP tendo como referência a complexidade computacional e estabilidade numérica, bem como as características de polarização e consistência de estimadores RLS. Do ponto de vista de controle, especificamente, as avaliações verificam o desempenho em termos das

trajetórias dos estados, ações de controle, e outros.

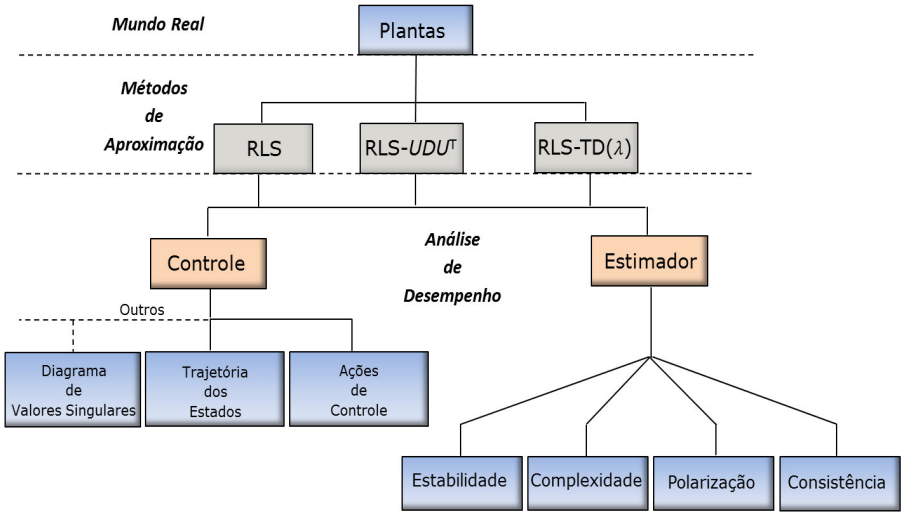

Figura 6.1: Diagrama de Blocos do Fluxo de Avaliação do Projeto *Online* de Controle.

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 da 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*.

Um modelo de circuito elétrico de quarta ordem com duas entradas e duas saídas é usado para avaliar o desempenho dos algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR no processo iterativo da solução da equação HJB-*Riccati*. Um modelo de terceira ordem multivariável que representa uma aeronave F-16 é usado para verificar o desempenho dos métodos RLS-TD(λ) para projeto *online* de controladores ótimos DLQR.

6.2 Experimentos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR

Esta seção está organizada em subseções que apresentam o desenvolvimento de procedimentos que avaliam o desempenho dos algoritmos RLS$_\mu$-HDP-DLQR

e RLS$_\mu$-UDU^T-HDP-DLQR para o projeto *online* de controladores ótimos via HDP. Inicialmente, os setups dos experimentos computacionais são apresentados na Subseção 6.2.1. Realiza-se, em seguida, comparações em termos dos transitórios dos comportamentos de convergência da solução da equação HJB-*Riccati* via algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR, Subseções 6.2.2 e 6.2.3. Na sequência, Subseção 6.2.4, avalia-se a estabilidade numérica do processo iterativo da solução da equação HJB-*Riccati* para situações em que a convergência já foi alcançada (regime permanente). Prossegue-se com a avaliação de desempenho dos algoritmos propostos em termos da consistência e polarização dos estimadores RLS quando associados com métodos HDP, Subseção 6.2.5. Depois, Subseção 6.2.6, apresentam-se os experimentos que verificam o efeito do fator de esquecimento no processo de convergência e estabilidade numérica e, por fim, apresentam-se os experimentos que avaliam a adaptabilidade dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR em relação à variações paramétricas na planta, Subseção 6.2.7.

6.2.1 Setup do Processo Iterativo

O setup do processo iterativo consiste em estabelecer quais são os melhores parâmetros para resolver uma dada aplicação, tendo como referência o empirismo do projetista ou as relações entre as variáveis do sistema que estão relacionadas com os métodos. Os parâmetros que desempenham um papel relevante na convergência do método são: ordem do sistema n ($n = 4$), intervalo de amostragem T_{amost} ($T_{amost} = 0.1s$), fator de esquecimento μ ($\mu = 0.92$) e matriz de covariância inicial Γ_0 ($\Gamma_0 = 10^3 I$).

Modelo do Sistema Dinâmico

O circuito elétrico de quarta ordem usado para avaliar o desempenho da metodologia proposta é mostrado no diagrama de Figura 6.2. O circuito é um sistema de componentes elétricos passivos constituído por capacitores, indutores e resistores.

As leis de *Kirchoff* das correntes e das tensões são aplicadas para relacionar as variáveis de tensão e corrente, as fontes de energia do sistema e os elementos

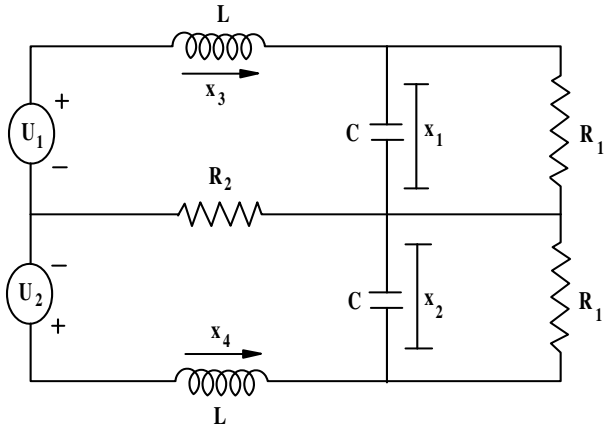

Figura 6.2: Diagrama Esquemático de um Circuito Elétrico de Ordem 4.

do circuito a parâmetros concentrados. O comportamento do sistema é modelado pelo seguinte conjunto de equações diferenciais.

$$C\frac{dx_1}{dt} + \frac{x_1}{R_1} - x_3 = 0 \tag{6.1}$$

$$C\frac{dx_2}{dt} + \frac{x_2}{R_1} - x_4 = 0 \tag{6.2}$$

$$L\frac{dx_3}{dt} + x_1 + R_2(x_3 + x_4) - u_1 = 0 \tag{6.3}$$

$$L\frac{dx_4}{dt} + x_2 + R_2(x_3 + x_4) - u_2 = 0, \tag{6.4}$$

em que x_1 e x_2 são variáveis de estado que representam as tensões nos capacitores. As variáveis de estado x_3 e x_4 são as correntes nos indutores. As matrizes do sistema dinâmico para a descrição do espaço de estado são dadas no Apêndice F.

Condições Iniciais para RLS

O vetor de parâmetros inicial θ_0 da equação de *Riccati/Lyapunov* estimado pelo método RLS é dado por

$$\theta = [\theta_1 \ \ldots \ \theta_i \ \ldots \ \theta_{10}]^T,$$

em que $\theta_i = 0$ para $i = 1, \ldots, 10$. A inicialização corresponde ao modelo do sistema dinâmico de quarta ordem.

A inicialização do processo de iteração de política aproximada requer a escolha de uma política inicial admissível K^0, então para tal, tem-se como matriz inicial da equação de *Riccati/Lyapunov* para o RLS

$$P_0 = \rho I_{4\times 4}, \tag{6.5}$$

em que $I_{4\times 4}$ é a matriz de identidade e ρ é um escalar real ($\rho = 10$). Esta matriz é associada aos parâmetros da aproximação RLS para o sistema dinâmico de quarta ordem, de acordo com a lei de formação de Eq.(5.55).

6.2.2 Experimentos RLS$_\mu$-HDP-DLQR

Os experimentos RLS$_\mu$-HDP-DLQR são realizados da implementação do algoritmo RLS$_\mu$-HDP-DLQR otimístico, conforme apresentado no Algoritmo 8. Este algoritmo representa o procedimento proposto que implementa o método RLS com fator de esquecimento para determinação da política de controle via regulador linear quadrático (LQR). O algoritmo tem uma condição de revitalização de estado devido à problemas de estado nulo que conduz à uma matriz de regressores com rank nulo. Neste caso, a revitalização é aplicada periodicamente, após cada intervalo n_{revit} ($n_{revit} = 10$).

A Solução da Equação Algébrica de *Riccati*

Nas Figuras 6.3, 6.4 e 6.5, apresenta-se a evolução do processo iterativo da solução da HJB-*Riccati* para um ciclo de 1150 iterações. As curvas (a)-(d) da Figura 6.3 representam o comportamento de convergência dos elementos p_{11}, p_{22}, p_{33} e p_{44} da matriz P correspondentes às componentes θ_1, θ_5, θ_8 e θ_{10} do vetor de parâmetros θ, respectivamente. As curvas (a)-(c) da Figura 6.4 mostram a evolução dos elementos p_{12}, p_{13}, e p_{14} da matriz P que estão associados aos elementos θ_2, θ_3 e θ_4 do vetor θ, respectivamente. O comportamento de convergência dos parâmetros p_{23}, p_{24} e p_{34} associados aos elementos θ_6, θ_7 e θ_9 é representado pelas curvas (a), (b) e (c) da Figura 6.5, respectivamente. Observa-se que a solução em torno da iteração 1100 já é admissível.

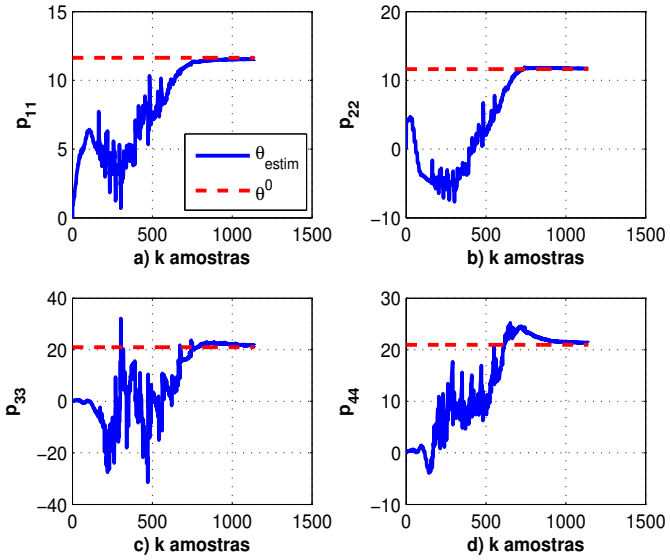

Figura 6.3: Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 1150 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

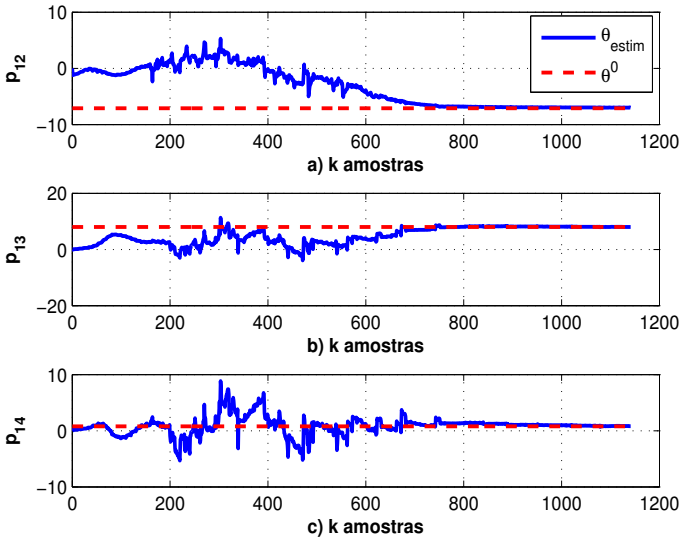

Figura 6.4: Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 1150 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

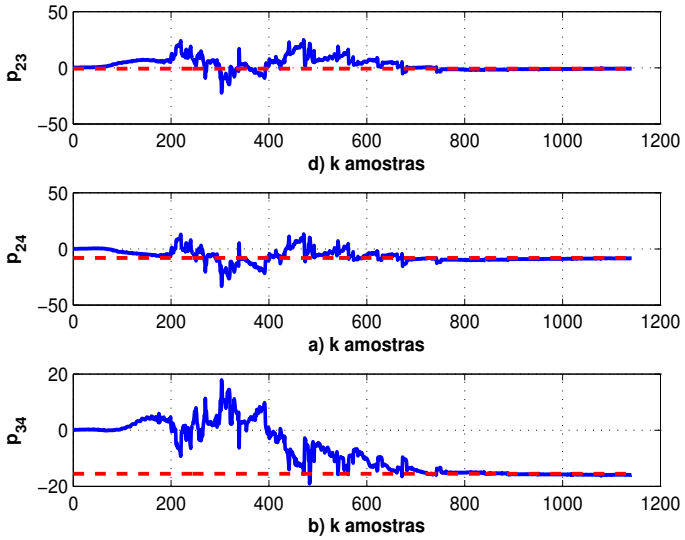

Figura 6.5: Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 1150 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Nas Tabelas 6.1-6.3 apresenta-se para cada componente do vetor de parâmetros θ a média, desvio padrão, mediana, valores máximos e mínimos da estimação RLS padrão para um ciclo de 1150 iterações no sentido de amostras do sensor. Estas estatísticas são avaliadas em termos da solução da equação HJB-*Riccati* para duas situações: a) Avaliar a estabilidade em torno de valores que são obtidos por meio de um modelo, e b) Avaliar a estabilidade em torno de valores que não estão previstos por modelos. A solução da HJB-*Riccati* pelo método de *Schur* é usada como referência para comparação de exatidão com os valores de P fornecidos pela abordagem HDP, supondo-se que os valores de *Shur* são os valores verdadeiros.

O processo de convergência é analisado sob a luz das equações que estão descritas no Algoritmo 8 para avaliar o condicionamento e a positividade da matriz de covariância do método RLS. Na Figura 6.6 apresenta-se o número de condição da matriz $\Gamma(k)$ de covariância e parâmetro de positividade p para o sistema de ordem 4. Observa-se que os valores de p situam-se no intervalo $0 < p < 1$ durante todo o processo iterativo do parâmetro θ. Desta forma, o comportamento de p não exibe irregularidades sobre o comportamento da matriz de covariância $\Gamma(k)$ de Eq.(5.94), pois esta é definida positiva durante todo o processo iterativo. Entretanto, observa-se uma elevada ordem de magnitude no condicionamento da

Tabela 6.1: Parâmetros θ_1, θ_5, θ_8 e θ_{10} - Estatísticas - Processo de estimação RLS$_\mu$-HDP-DLQR otimístico, para um ciclo de 1150 iterações.

Parâmetro	θ_1	θ_5	θ_8	θ_{10}
Associação $p_{ii} = \theta_s$	p_{11}	p_{22}	p_{33}	p_{44}
Verdadeiro (θ^0)	11.6401	11.6401	20.9535	20.9535
Estimado ($\widehat{\theta}$)	11.5521	11.7183	21.5801	21.3390
Média	8.2421	4.9255	7.8678	14.1208
Desvio padrão	3.1840	6.7779	13.2142	8.7209
Mediana	8.6833	6.0122	7.6874	15.8068
Mínimo	0.0000	-7.6627	-31.3833	-3.9063
Máximo	11.5521	11.9215	32.1060	25.2130

Tabela 6.2: Parâmetros θ_2, θ_3 e θ_4 - Estatísticas - Processo de estimação RLS$_\mu$-HDP-DLQR otimístico, para um ciclo de 1150 iterações.

Parâmetro	θ_2	θ_3	θ_4
Associação $p_{ij} = \theta_s/2$	p_{12}	p_{13}	p_{14}
Verdadeiro (θ^0)	-14.2285	16.0669	1.5776
Estimado ($\widehat{\theta}$)	-13.9831	16.0291	1.6335
Média	-6.6120	9.9739	1.7138
Desvio padrão	6.8720	6.0551	3.1860
Mediana	-6.7529	10.0996	1.9628
Mínimo	-13.9831	-7.8036	-10.6487
Máximo	10.5969	22.5650	17.6925

matriz de covariância e nota-se um aumento desse fator ao longo do processo iterativo. Intuitivamente, espera-se que este comportamento possa vir a comprometer a estabilidade numérica do algoritmo, uma hipótese que será verificada nas próximas seções.

Tabela 6.3: Parâmetros θ_6, θ_7 e θ_9 - Estatísticas - Processo de estimação RLS$_\mu$-HDP-DLQR otimístico, para um ciclo de 1150 iterações.

Parâmetro	θ_6	θ_7	θ_9
Associação $p_{ij} = \theta_s/2$	p_{23}	p_{24}	p_{34}
Verdadeiro (θ^0)	-1.5776	-16.0669	-31.0929
Estimado ($\widehat{\theta}$)	-1.4982	-16.7048	-31.8113
Média	5.2752	-13.1328	-15.2619
Desvio padrão	11.5294	10.7661	16.6948
Mediana	0.8135	-16.7924	-20.9395
Mínimo	-44.7021	-66.5781	-38.1626
Máximo	49.8375	26.3041	35.7785

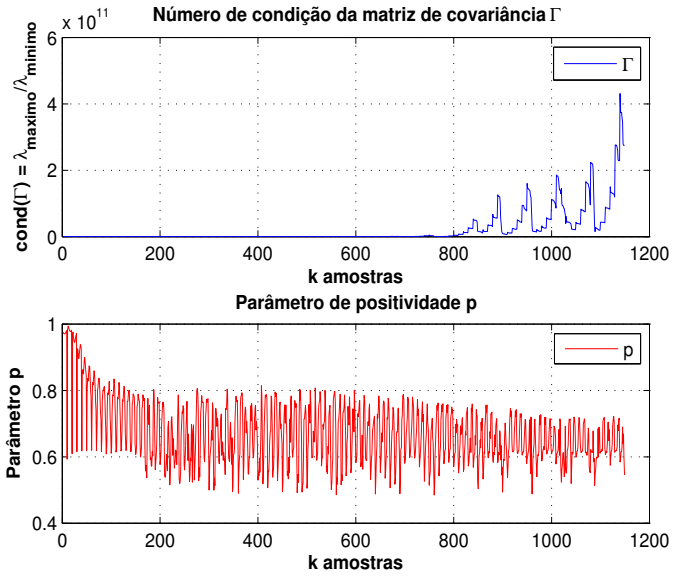

Figura 6.6: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade durante o processo de estimação RLS$_\mu$-HDP-DLQR otimístico com fator de esquecimento $\mu = 0.92$, para um ciclo de 1150 iterações.

6.2.3 Experimentos RLS$_\mu$-UDU^T-HDP-DLQR

Os resultados dos experimentos RLS$_\mu$-UDU^T-HDP-DLQR são apresentados de forma análoga aos experimentos RLS$_\mu$-HDP-DLQR. Para um ciclo de 1150 iterações, observa-se que o comportamento de convergência da solução da equação HJB-*Riccati* é similar ao do algoritmo RLS$_\mu$-HDP-DLQR otimístico, conforme ilustrado nas Figuras 6.7, 6.8 e 6.9 em comparação com as Figuras 6.3, 6.4 e 6.5,

respectivamente. Como também as estatísticas associadas com a evolução do processo iterativo da solução da HJB-*Riccati* do algoritmo RLS$_\mu$-UDU^T-HDP-DLQR otimístico apresentam propriedades semelhantes às apresentadas pelo algoritmo RLS$_\mu$-HDP-DLQR otimístico com respeito à média, desvio padrão, mediana, máximo e mínimo para cada um dos parâmetros estimados. Na Tabela 6.4 apresenta-se as estatísticas da estimação RLS com fatoração UDU^T que estão associadas com a evolução dos parâmetros p_{11}, p_{22}, p_{33} e p_{44} da solução da HJB-*Riccati*. Nas Tabelas 6.5 e 6.6 são apresentadas as estatísticas associadas com o comportamento dos parâmetros p_{12}, p_{13}, p_{14} e p_{23}, p_{24}, p_{34}, respectivamente.

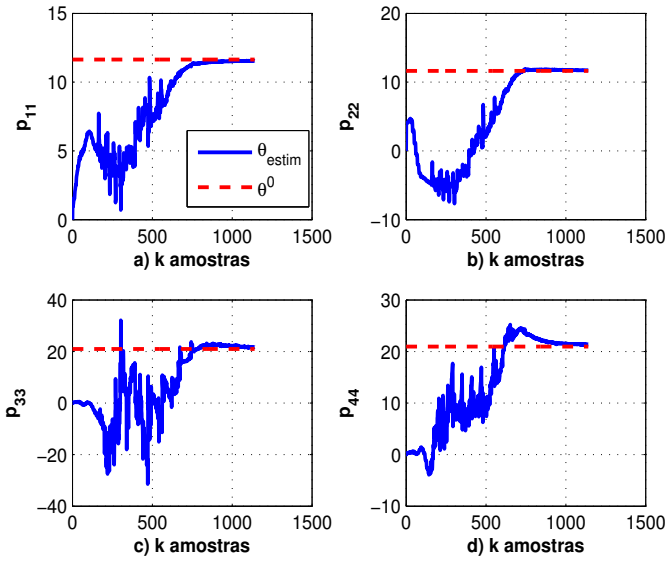

Figura 6.7: Comportamento de convergência dos parâmetros p_{ii} para um ciclo de 1150 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

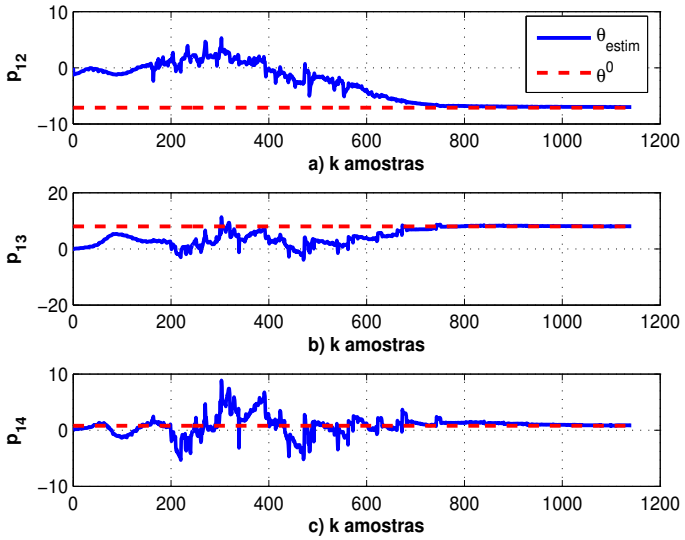

Figura 6.8: Comportamento de convergência dos parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 1150 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

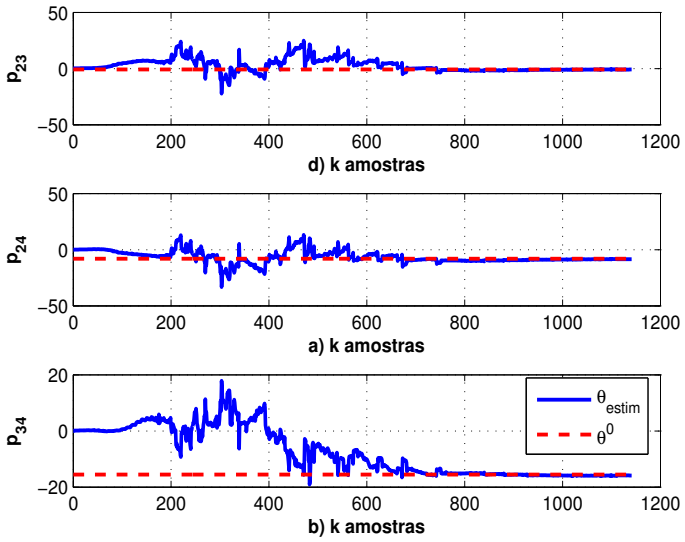

Figura 6.9: Comportamento de convergência dos parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 1150 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Na Figura 6.10 apresenta-se o número de condição do fator *Cholesky* $G(k)$

Tabela 6.4: Parâmetros θ_1, θ_5, θ_8 e θ_{10} - Estatísticas - Processo de estimação RLS$_\mu$-UDU^T-HDP-DLQR otimístico, para um ciclo de 1150 iterações.

Parâmetro	θ_1	θ_5	θ_8	θ_{10}
Associação $p_{ii} = \theta_s$	p_{11}	p_{22}	p_{33}	p_{44}
Verdadeiro (θ^0)	11.6401	11.6401	20.9535	20.9535
Estimado ($\widehat{\theta}$)	11.5506	11.7225	21.6826	21.3506
Média	8.2419	4.9268	7.8768	14.1222
Desvio padrão	3.1839	6.7792	13.2219	8.7218
Mediana	8.6837	6.0129	7.7079	15.8068
Mínimo	0.0000	-7.6627	-31.3812	-3.9063
Máximo	11.5582	11.9226	32.1060	25.1968

Tabela 6.5: Parâmetros θ_2, θ_3 e θ_4 - Estatísticas - Processo de estimação RLS$_\mu$-UDU^T-HDP-DLQR otimístico, para um ciclo de 1150 iterações.

Parameter	θ_2	θ_3	θ_4
Association $p_{ij} = \theta_s/2$	p_{12}	p_{13}	p_{14}
True (θ^0)	-14.2285	16.0669	1.5776
Estimated ($\widehat{\theta}$)	-13.9833	16.0895	1.7042
Expectation	-6.6114	9.9752	1.7138
Standard deviation	6.8713	6.0557	3.1854
Median	-6.7524	10.1058	1.9102
Minimum	-13.9984	-7.8027	-10.6487
Maximum	10.5969	22.5650	17.6925

da matriz de covariância $\Gamma(k)$ e parâmetro de positividade p do processo RLS$_\mu$-UDU^T-HDP-DLQR para estimação da solução DLQR. Observa-se que os valores de p satisfazem a condição necessária da propriedade definida positiva durante todo o processo iterativo do parâmetro θ, isto é $0 < p < 1$. Em relação ao número de condição, nota-se um valor substancialmente menor comparado com o número de condição da matriz de covariância do estimador RLS$_\mu$-HDP-DLQR. Isto é um indicativo de que o algoritmo tende a apresentar sensibilidades menores à efeitos de perturbações causadas por problemas numéricos.

Tabela 6.6: Parâmetros θ_6, θ_7 e θ_9 - Estatísticas - Processo de estimação RLS$_\mu$-UDU^T-HDP-DLQR otimístico, para um ciclo de 1150 iterações.

Parâmetro	θ_6	θ_7	θ_9
Associação $p_{ij} = \theta_s/2$	p_{23}	p_{24}	p_{34}
Verdadeiro (θ^0)	-1.5776	-16.0669	-31.0929
Estimado ($\widehat{\theta}$)	-1.6774	-16.8926	-31.6927
Média	5.2745	-13.1374	-15.2700
Desvio padrão	11.5279	10.7665	16.7022
Mediana	0.8135	-16.8651	-21.0662
Mínimo	-44.7021	-66.5780	-38.1577
Máximo	49.8345	26.3009	35.7785

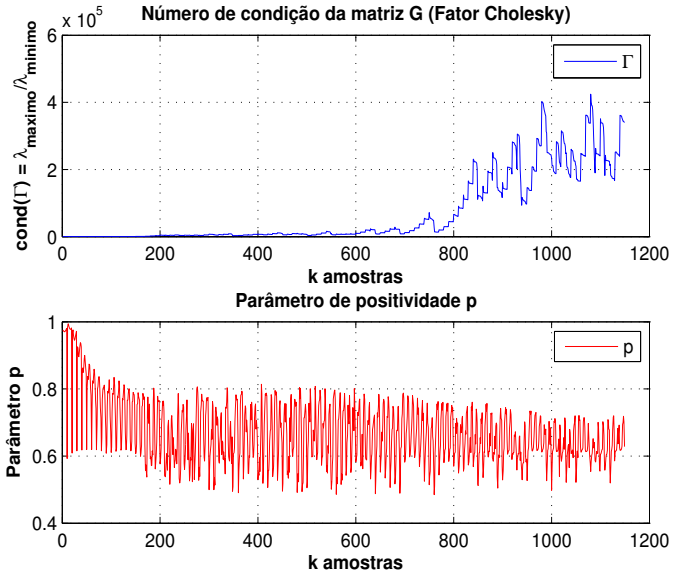

Figura 6.10: Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade durante o processo de estimação RLS$_\mu$-UDU^T-HDP-DLQR otimístico com fator de esquecimento $\mu = 0.92$, para um ciclo de 1150 iterações.

6.2.4 Experimentos sobre Estabilidade Numérica

O processo iterativo é avaliado para situações em que a convergência já foi atingida, restando avaliar a estabilidade numérica. Nas Figuras 6.11-6.13 apresenta-se a evolução do processo iterativo da solução da equação HJB-*Riccati* do algoritmo RLS$_\mu$-HDP-DLQR otimístico para um ciclo de 5000 iterações. Observa-se que depois que o parâmetro P alcança a fase de acomodação desajustes significativos

ocorrem nos seus valores entre as iterações 2500 e 3000, e menores desajustes entre as iterações 3000 e 5000. Este período caracteriza-se como uma fase crítica para a estabilidade numérica do algoritmo, pois neste momento o estimador apresenta uma maior sensibilidade à efeitos de perturbações causadas por problemas numéricos.

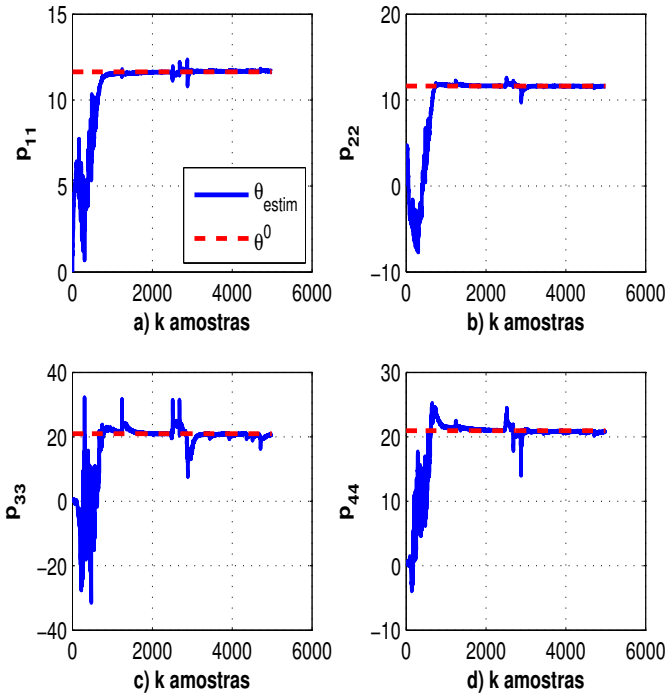

Figura 6.11: Comportamento de convergência dos parâmetros p_{ii} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

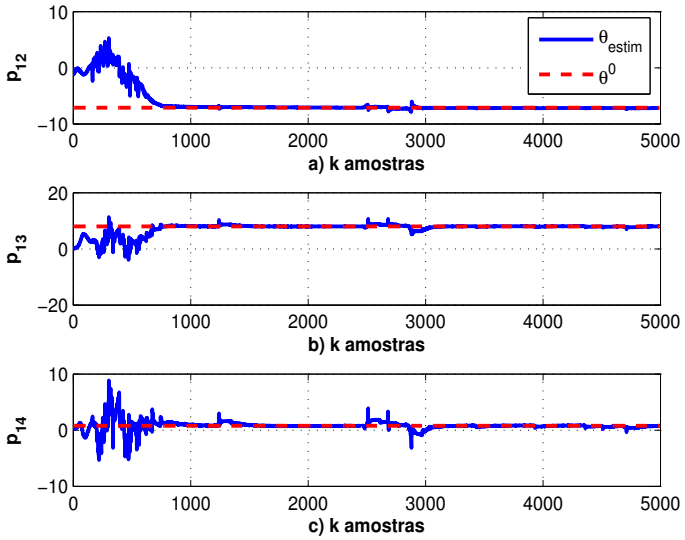

Figura 6.12: Comportamento de convergência dos parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

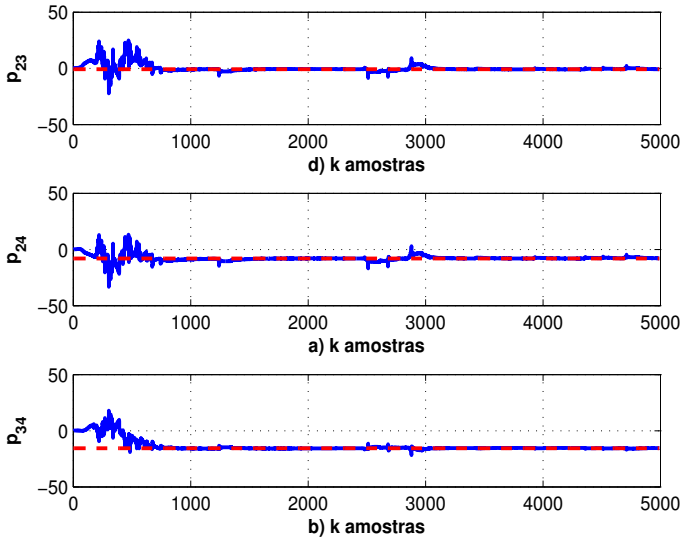

Figura 6.13: Comportamento de convergência dos parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Nas Figuras 6.14-6.16 apresenta-se a evolução do processo iterativo da solução da equação HJB-*Riccati* do algoritmo RLS$_\mu$-UDU^T-HDP-DLQR para um ciclo

de 5000 iterações. Na análise comparativa entre os comportamentos de convergência dos algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR, observa-se que quando o estimador RLS$_\mu$-UDU^T-HDP-DLQR ultrapassa a fase de acomodação, seus parâmetros estimados não sofrem nenhuma grande perturbação, mantendo-se estáveis em relação aos valores verdadeiros para o processo inteiro. Isto é uma indicação que o algoritmo RLS$_\mu$-HDP-DLQR com fatoração UDU^T recupera os desvios com respeito aos valores verdadeiros do parâmetro θ. Desta forma, observa-se-se que a fatoração UDU^T promove melhorias nas características de estabilidade numérica do processo de estimação da solução da equação HJB-*Riccati*. Este fenômeno pode ser explicado da seguinte forma: conforme mostrado na Figura 6.17, o número de condição da matriz de covariância $\Gamma(k)$ do estimador RLS$_\mu$-HDP-DLQR apresenta variações drásticas nos seus valores (presença de picos que podem levar à situações de não-convergência), principalmente em torno das iterações 2000 e 2500, o que reflete posteriormente nos valores dos parâmetros que sofrem perda de estabilidade entre as iterações 2500 e 3000. Enquanto isso, como mostrado na Figura 6.18, o número de condição do fator *Cholesky* exibe um comportamento sem variações abruptas, e com menores amplitudes em relação a aquelas apresentadas pelo RLS$_\mu$-HDP-DLQR, sendo mantido em uma determinada faixa de valores após o período transitório.

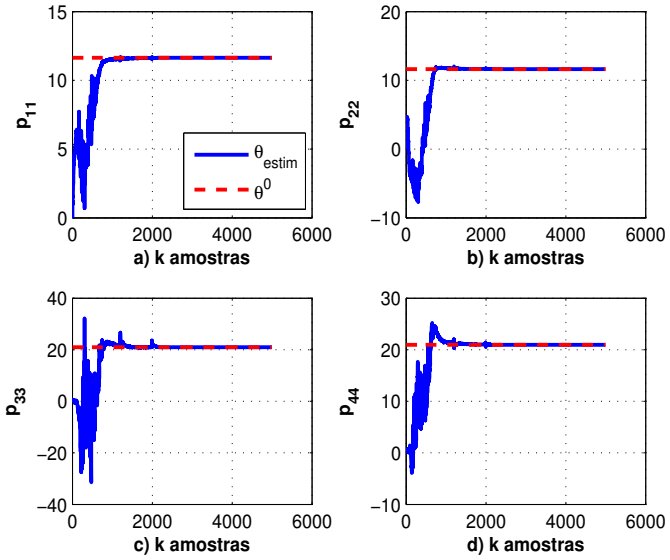

Figura 6.14: Comportamento de convergência dos parâmetros p_{ii} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

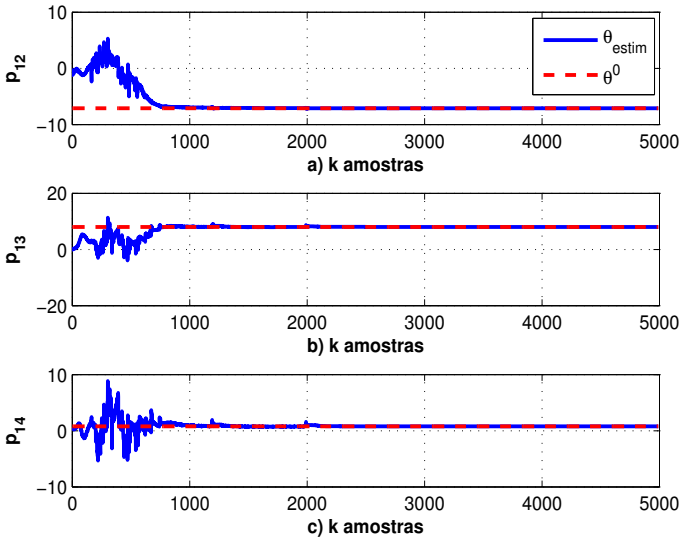

Figura 6.15: Comportamento de convergência dos parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

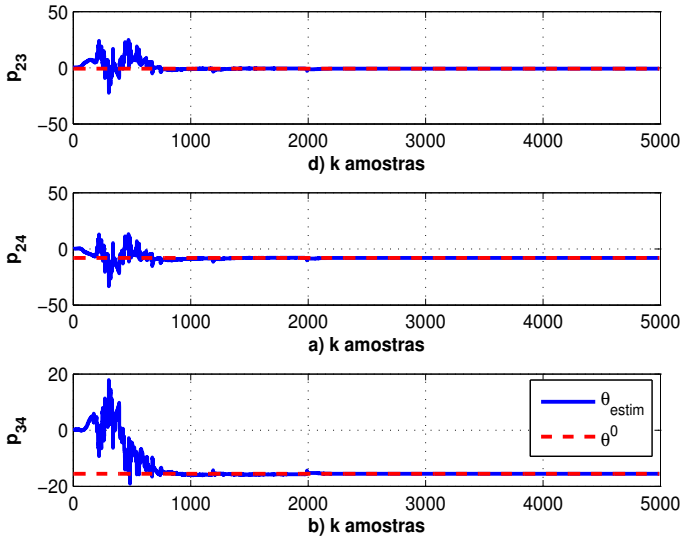

Figura 6.16: Comportamento de convergência dos parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Nas Figuras 6.17 e 6.18 apresentam-se os números de condição e os parâmetros de positividade dos processos de estimação RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR, respectivamente.

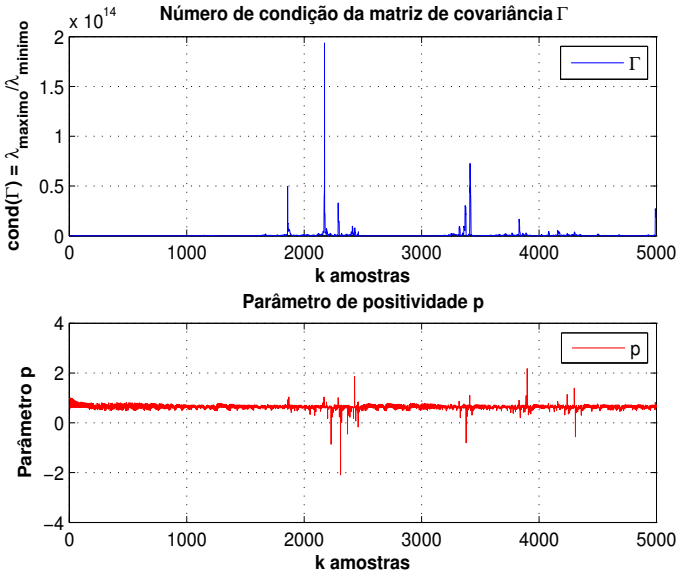

Figura 6.17: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade durante o processo de estimação RLS$_\mu$-HDP-DLQR otimístico com fator de esquecimento $\mu = 0.92$, para um ciclo de 5000 iterações.

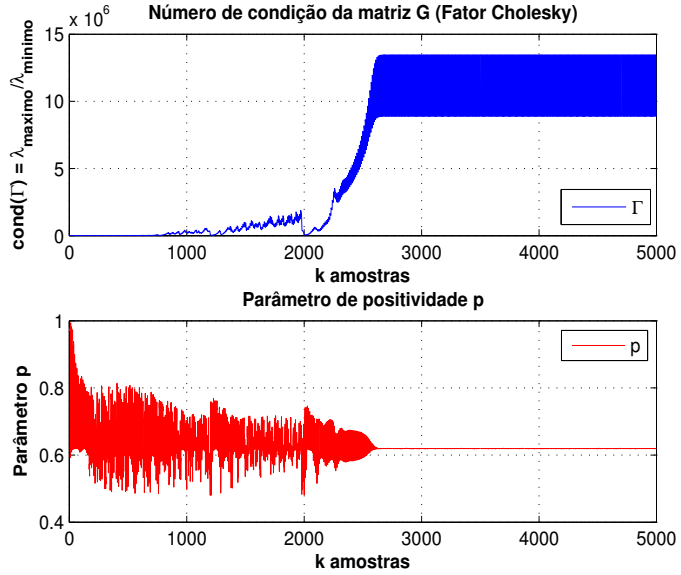

Figura 6.18: Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade durante o processo de estimação RLS$_\mu$-UDU^T-HDP-DLQR otimístico com fator de esquecimento $\mu = 0.92$, para um ciclo de 5000 iterações.

Com respeito aos parâmetros de positividade, observa-se que o estimador

RLS$_\mu$-HDP-DLQR sem fatoração UDU^T apresenta irregularidades quando seu parâmetro de positividade p assume valores fora do intervalo $0 < p < 1$ para algumas iterações do processo de estimação do parâmetro θ, indicando perda da positividade da matriz de covariância. Enquanto que os valores de θ para o estimador RLS$_\mu$-HDP-DLQR com fatoração UDU^T estão no intervalo de validade durante todo o processo iterativo.

6.2.5 Consistência dos Métodos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR

Experimentos foram realizados para investigar a média e desvio padrão para um número grande de iterações. Observou-se que à medida que se aumenta o número de iterações, a média tende para o valor verdadeiro θ^0 e o desvio padrão tende para zero. Esses resultados evidenciaram que os estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR são não-polarizados e consistentes.

Nas Tabelas 6.7-6.9 apresenta-se para cada componente do vetor de parâmetros θ a média e desvio padrão da estimação RLS$_\mu$-HDP-DLQR para um ciclo de 100000 iterações.

Tabela 6.7: Parâmetros θ_1, θ_5, θ_8 e θ_{10} - Estatísticas: Média e Desvio Padrão - Processo de estimação RLS$_\mu$-HDP-DLQR otimístico, para um ciclo de 100000 iterações.

Parâmetro	θ_1	θ_5	θ_8	θ_{10}
Associação $p_{ii} = \theta_s$	p_{11}	p_{22}	p_{33}	p_{44}
Verdadeiro (θ^0)	11.6401	11.6401	20.9535	20.9535
Estimado ($\widehat{\theta}$)	11.6401	11.6401	20.9535	20.9535
Média	11.6019	11.5631	20.7937	20.8728
Desvio padrão	0.4964	1.0170	2.0140	1.1849

Tabela 6.8: Parâmetros θ_2, θ_3 e θ_4 - Estatísticas: Média e Desvio Padrão - Processo de estimação RLS$_\mu$-HDP-DLQR otimístico, para um ciclo de 100000 iterações.

Parâmetro	θ_2	θ_3	θ_4
Associação $p_{ij} = \theta_s/2$	p_{12}	p_{13}	p_{14}
Verdadeiro (θ^0)	-14.2285	16.0669	1.5776
Estimado ($\widehat{\theta}$)	-14.2285	16.0669	1.5776
Média	-14.1433	15.9940	1.5757
Desvio padrão	1.0937	0.9292	0.3745

Tabela 6.9: Parâmetros θ_6, θ_7 e θ_9 - Estatísticas: Média e Desvio Padrão - Processo de estimação RLS$_\mu$-HDP-DLQR otimístico, para um ciclo de 100000 iterações.

Parâmetro	θ_6	θ_7	θ_9
Associação $p_{ij} = \theta_s/2$	p_{23}	p_{24}	p_{34}
Verdadeiro (θ^0)	-1.5776	-16.0669	-31.0929
Estimado ($\widehat{\theta}$)	-1.5776	-16.0669	-31.0929
Média	-1.4899	-16.0205	-30.9130
Desvio padrão	1.4912	1.2864	2.4582

Nas Tabelas 6.10-6.12 apresenta-se para cada componente do vetor de parâmetros θ a média e desvio padrão da estimação RLS$_\mu$-UDU^T-HDP-DLQR para um ciclo de 100000 iterações.

Tabela 6.10: Parâmetros θ_1, θ_5, θ_8 e θ_{10} - Estatísticas: Média e Desvio Padrão - Processo de estimação RLS$_\mu$-UDU^T-HDP-DLQR otimístico, para um ciclo de 100000 iterações.

Parâmetro	θ_1	θ_5	θ_8	θ_{10}
Associação $p_{ii} = \theta_s$	p_{11}	p_{22}	p_{33}	p_{44}
Verdadeiro (θ^0)	11.6401	11.6401	20.9535	20.9535
Estimado ($\widehat{\theta}$)	11.6401	11.6401	20.9535	20.9535
Média	11.6010	11.5638	20.8083	20.8767
Desvio padrão	0.4960	1.0165	1.9831	1.1813

Tabela 6.11: Parâmetros θ_2, θ_3 e θ_4 - Estatísticas: Média e Desvio Padrão - Processo de estimação RLS$_\mu$-UDU^T-HDP-DLQR otimístico, para um ciclo de 100000 iterações.

Parâmetro	θ_2	θ_3	θ_4
Associação $p_{ij} = \theta_s/2$	p_{12}	p_{13}	p_{14}
Verdaeiro (θ^0)	-14.2285	16.0669	1.5776
Estimado ($\widehat{\theta}$)	-14.2285	16.0669	1.5776
Média	-14.1406	15.9980	1.5799
Desvio padrão	1.0926	0.9156	0.3416

Tabela 6.12: Parâmetros θ_6, θ_7 e θ_9 - Estatísticas: Média e Desvio Padrão - Processo de estimação RLS$_\mu$-UDU^T-HDP-DLQR otimístico, para um ciclo de 100000 iterações.

Parâmetro	θ_6	θ_7	θ_9
Associação $p_{ij} = \theta_s/2$	p_{23}	p_{24}	p_{34}
Verdadeiro (θ^0)	-1.5776	-16.0669	-31.0929
Estimado ($\widehat{\theta}$)	-1.5776	-16.0669	-31.0929
Média	-1.5009	-16.0376	-30.9143
Desvio padrão	1.4324	1.1947	2.4521

6.2.6 Seleção do Fator de Esquecimento

Uma análise de condições para a estabilidade numérica da implementação de precisão finita dos algoritmos RLS convencionais (sem fatoração UDU^T) é encontrada em (Liavas e 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. Admissivelmente, explorando as consideráveis propriedades numéricas da fatoração UDU^T, os algoritmos RLS$_\mu$-UDU^T-HDP-DLQR podem oferecer uma convergência numericamente mais estável e, em determinadas situações, dependendo da escolha do fator de esquecimento, podem garantir uma convergência mais rápida do que os algoritmos RLS$_\mu$-HDP-DLQR.

Em inúmeras simulações realizadas neste trabalho, pôde-se verificar a influência do fator de esquecimento μ no processo de convergência e estabilidade numérica dos estimadores RLS$_\mu$-HDP-DLQR para solução da HJB-*Riccati* para diferentes valores de μ. Na análise comparativa dos desempenhos dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR em relação à variações no fator de es-

quecimento, os outros parâmetros foram mantidos constantes e iguais a aqueles dos experimentos anteriores em que se utilizou o valor $\mu = 0.92$, como por exemplo a matriz de covariância inicial que foi estabelecida $\Gamma_0 = 10^3 I$. Nos experimentos apresentados a seguir considera-se os seguintes valores de μ: 0.9, 0.93, 0.94, 0.95, 0.96 e 0.97.

Para o caso $\mu = 0.9$, observou-se a grande influência deste fator no comportamento dos estimadores RLS$_\mu$-HDP-DLQR, uma vez que este fator causou uma divergência abrupta nos parâmetros destes estimadores com poucas centenas de iterações, como pode ser verificado na Figura 6.19 que ilustra a evolução dos parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 1400 iterações. O fenômeno da divergência ocorreu antes mesmo que um regime permanente possa ter sido estabelecido. À priori, é necessário considerar como hipótese que o comportamento do parâmetro de positividade denuncia a iminência deste fenômeno quando seus valores extrapolam a faixa de validade, evidenciando uma anomalia, conforme pode ser observado na curva b) da Figura 6.20. Com também, nota-se a partir da curva a) da mesma figura variações abruptas no condicionamento da matriz de covariância.

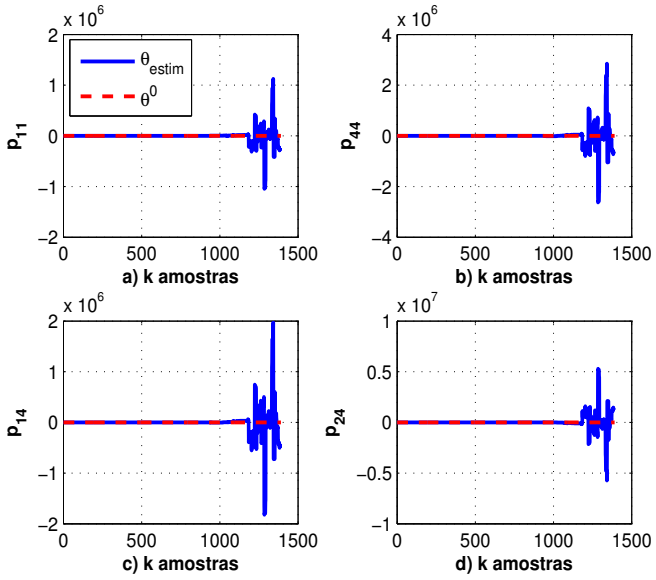

Figura 6.19: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 1400 iterações, com fator de esquecimento $\mu = 0.9$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

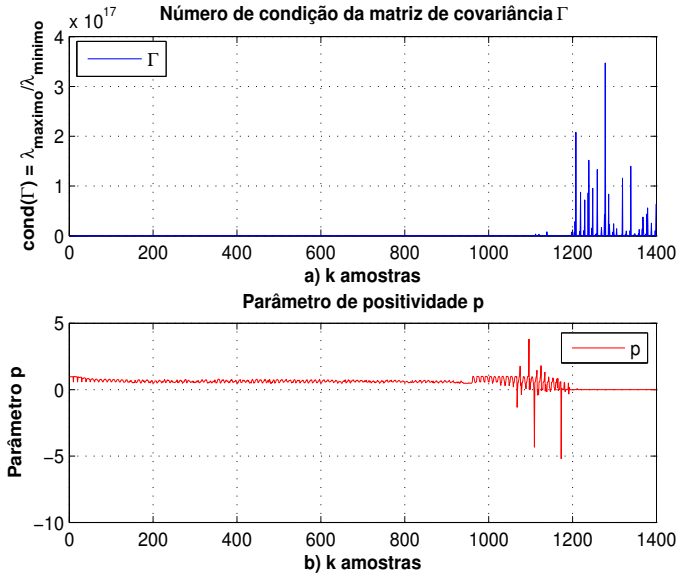

Figura 6.20: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 1400 iterações, com fator de esquecimento $\mu = 0.9$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

A Tabela 6.13 apresenta as ocorrências associadas com a perda da positividade da matriz de covariância Γ_k do processo de estimação RLS$_\mu$-HDP-DLQR para um ciclo de 1200 iterações, com fator de esquecimento $\mu = 0.9$, tendo por base o parâmetro de positividade p_k, em que k representa o instante de tempo de ocorrência do processo iterativo RLS, k_1 representa o "tag"associado com o tempo k, $\lambda_{min}(\Gamma_k)$ é o autovalor mínimo da matriz de covariância Γ_k no instante k, $cond(\Gamma_k)$ é o número de condição da matriz Γ_k no instante k, $f_{qd} = \phi_k^T \Gamma_{k-1} \phi_k$ é uma forma quadrática associada a matriz Γ_k, ε_k é o erro da estimativa que é dado por $r_k - \phi_k^T \theta_{k-1}$, $\|\Gamma_k\|_2$ é a norma 2 da matriz Γ_k, e $\|L_k\|_2$ é a norma 2 do vetor de ganho L_k do processo iterativo do RLS.

Tabela 6.13: Ocorrências associadas com a perda da positividade da matriz de covariância Γ_k para um ciclo de 1200 iterações - Processo de estimação RLS$_\mu$-HDP-DLQR, com fator de esquecimento $\mu = 0.9$.

$k1$	k	p_k	$\lambda_{min}(\Gamma_k)$	$cond(\Gamma_k)$	f_{qd}	ε_k	$\|\Gamma_k\|_2$	$\|L_k\|_2$
1	1068	-1.327	-1.088×10^{-3}	1.717×10^{13}	-1.578×10^{0}	1.005×10^{0}	1.867×10^{10}	1.576×10^{4}
2	1069	-0.008	-1.085×10^{3}	5.245×10^{11}	-1.102×10^{2}	3.205×10^{0}	6.149×10^{9}	1.884×10^{4}
3	1074	1.057	-2.004×10^{3}	4.345×10^{11}	-4.881×10^{-2}	4.935×10^{-2}	9.048×10^{9}	3.190×10^{3}
4	1075	1.765	-2.805×10^{3}	3.282×10^{11}	-3.900×10^{-1}	1.932×10^{-1}	1.000×10^{10}	8.356×10^{3}
5	1078	-0.362	-8.038×10^{3}	4.851×10^{12}	-3.388×10^{0}	2.513×10^{0}	1.048×10^{10}	1.289×10^{4}
6	1094	1.010	-3.666×10^{10}	1.357×10^{13}	-8.971×10^{-3}	8.693×10^{-2}	3.666×10^{10}	6.434×10^{4}
7	1096	3.805	-1.947×10^{11}	2.280×10^{13}	-6.635×10^{-1}	3.255×10^{-1}	1.947×10^{11}	3.315×10^{5}
8	1108	-0.050	-5.883×10^{10}	5.732×10^{12}	-1.900×10^{1}	7.611×10^{-3}	5.883×10^{10}	3.412×10^{4}
9	1109	-4.337	-6.530×10^{10}	5.981×10^{14}	-1.108×10^{0}	1.300×10^{-2}	6.530×10^{10}	8.220×10^{2}
10	1115	1.457	-2.682×10^{11}	1.095×10^{13}	-2.824×10^{-1}	4.355×10^{-3}	2.682×10^{11}	3.148×10^{5}
11	1116	-0.021	-2.906×10^{9}	8.174×10^{11}	-4.467×10^{1}	3.336×10^{-3}	6.118×10^{11}	1.103×10^{6}
12	1117	-0.412	-2.975×10^{9}	7.532×10^{13}	-3.084×10^{0}	9.649×10^{-3}	7.131×10^{11}	1.128×10^{4}
13	1119	-0.357	-3.661×10^{9}	2.629×10^{15}	-3.418×10^{0}	1.941×10^{-3}	8.832×10^{11}	3.335×10^{1}
14	1125	1.795	-1.221×10^{10}	3.773×10^{13}	-3.987×10^{-1}	1.545×10^{-2}	1.420×10^{12}	7.754×10^{4}
15	1126	-0.084	-1.355×10^{10}	8.386×10^{12}	-1.162×10^{1}	4.019×10^{-2}	1.632×10^{12}	1.993×10^{5}
16	1128	-0.004	-1.672×10^{10}	1.811×10^{13}	-2.023×10^{2}	9.724×10^{-3}	2.025×10^{12}	1.090×10^{4}
17	1129	-0.014	-1.858×10^{10}	3.494×10^{14}	-6.393×10^{1}	2.690×10^{-2}	2.251×10^{12}	4.505×10^{2}
18	1134	1.311	-4.962×10^{10}	2.154×10^{13}	-2.136×10^{-1}	1.477×10^{-3}	2.388×10^{12}	1.429×10^{5}
19	1135	-0.114	-4.543×10^{10}	3.720×10^{12}	-8.818×10^{0}	1.372×10^{-3}	2.992×10^{12}	1.025×10^{6}
20	1137	-0.042	-5.320×10^{10}	5.556×10^{13}	-2.240×10^{1}	5.927×10^{-3}	3.803×10^{12}	7.492×10^{3}
21	1138	-0.027	-5.911×10^{10}	2.250×10^{14}	-3.377×10^{1}	5.109×10^{-3}	4.226×10^{12}	2.181×10^{3}
22	1139	-0.167	-6.568×10^{10}	8.052×10^{15}	-6.291×10^{0}	1.139×10^{-3}	4.696×10^{12}	4.276×10^{2}
23	1147	-0.161	-2.849×10^{11}	2.459×10^{14}	-6.499×10^{0}	5.107×10^{-3}	4.883×10^{12}	6.638×10^{3}
24	1149	-0.019	-3.515×10^{11}	1.265×10^{15}	-4.745×10^{1}	1.120×10^{-3}	6.032×10^{12}	1.294×10^{2}
25	1158	-0.055	-2.187×10^{12}	4.567×10^{14}	-1.718×10^{1}	3.248×10^{-4}	4.333×10^{12}	3.579×10^{2}
26	1166	-0.083	-1.090×10^{13}	5.560×10^{13}	-1.174×10^{1}	1.777×10^{-4}	1.090×10^{13}	1.638×10^{4}
27	1169	-0.001	-1.495×10^{13}	1.722×10^{14}	-8.628×10^{2}	1.065×10^{-4}	1.495×10^{13}	1.711×10^{2}
28	1173	-5.194	-2.902×10^{13}	9.247×10^{12}	-1.073×10^{0}	9.320×10^{-4}	2.902×10^{13}	1.588×10^{6}
29	1178	-0.002	-4.691×10^{13}	2.308×10^{14}	-3.664×10^{2}	6.338×10^{-4}	4.691×10^{13}	3.820×10^{2}
30	1179	-0.001	-5.212×10^{13}	4.975×10^{14}	-1.044×10^{3}	1.015×10^{-4}	5.212×10^{13}	1.574×10^{2}
31	1182	-0.134	1.083×10^{2}	9.483×10^{12}	-7.605×10^{0}	7.873×10^{-4}	3.406×10^{14}	1.229×10^{7}
32	1183	-0.146	-2.668×10^{14}	1.397×10^{13}	-7.075×10^{0}	4.782×10^{-3}	2.668×10^{14}	1.344×10^{6}
33	1186	-0.000	-5.307×10^{14}	2.624×10^{12}	-1.989×10^{4}	5.008×10^{-2}	5.307×10^{14}	4.138×10^{5}
34	1189	-0.000	-6.916×10^{12}	9.812×10^{13}	-1.832×10^{5}	3.160×10^{1}	1.261×10^{15}	4.834×10^{3}
35	1191	-0.334	-2.627×10^{15}	1.224×10^{13}	-3.591×10^{0}	4.051×10^{-3}	2.628×10^{15}	1.836×10^{7}
36	1193	-0.000	-5.390×10^{8}	5.926×10^{12}	-3.304×10^{3}	3.431×10^{-2}	4.073×10^{16}	1.096×10^{8}
37	1194	-0.000	-4.999×10^{3}	5.374×10^{12}	-1.006×10^{4}	2.577×10^{0}	2.686×10^{16}	6.855×10^{5}
38	1196	-0.000	1.682×10^{4}	2.647×10^{12}	-3.152×10^{4}	1.812×10^{-1}	3.198×10^{16}	2.646×10^{5}
39	1197	-0.000	-5.711×10^{8}	3.610×10^{12}	-4.943×10^{6}	2.240×10^{-2}	3.593×10^{16}	8.192×10^{4}
40	1198	-0.000	-6.345×10^{8}	7.453×10^{15}	-1.129×10^{4}	3.778×10^{-2}	3.992×10^{16}	2.774×10^{2}
41	1199	-0.000	-7.050×10^{8}	5.526×10^{12}	-1.274×10^{8}	1.846×10^{-3}	4.436×10^{16}	1.132×10^{4}
42	1200	-0.027	-6.942×10^{8}	2.158×10^{13}	-3.477×10^{1}	8.474×10^{-2}	9.636×10^{16}	1.437×10^{7}

Similarmente, para o caso $\mu = 0.97$, os resultados apresentados nas Figuras 6.21 e 6.22 também mostram uma divergência abrupta dos estimadores RLS$_\mu$-HDP-DLQR associada com uma anomalia nos comportamentos do parâmetro de positividade e do número de condição. Contudo, o fenômeno da divergência é manifestado após a execução de milhares de iterações, ao contrário do caso $\mu = 0.9$ que contribuiu severamente para a aceleração do processo de divergência. O mesmo processo, executado com valores acima de 0.97, ocasionou também a divergência do estimador RLS$_\mu$-HDP-DLQR. Mais informação sobre as frequências de ocorrência relacionadas com a perda da positividade da matriz de covariância é fornecida a partir da Tabela 6.14, onde identifica-se um total de 42 ocorrências em um ciclo de 15080 iterações, 6 ocorrências observadas até a iteração 7086 e 36 ocorrências da iteração 13180 a 15080.

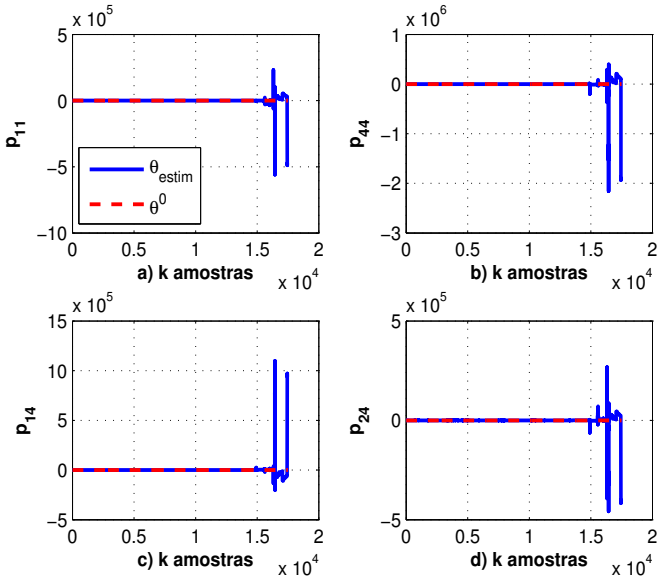

Figura 6.21: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 17500 iterações, com fator de esquecimento $\mu = 0.97$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

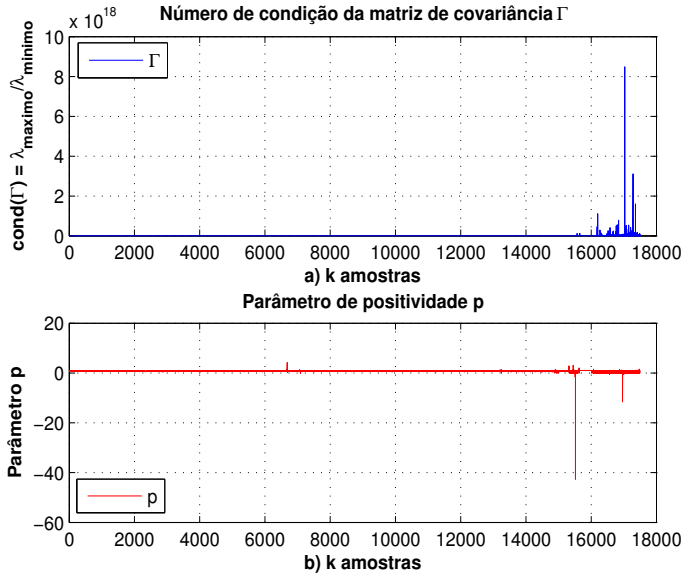

Figura 6.22: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 17500 iterações, com fator de esquecimento $\mu = 0.97$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Tabela 6.14: Ocorrências associadas com a perda da positividade da matriz de covariância Γ_k para um ciclo de 15072 iterações - Processo de estimação RLS$_\mu$-HDP-DLQR, com fator de esquecimento $\mu = 0.97$.

$k1$	k	p_k	$\lambda_{min}(\Gamma_k)$	$cond(\Gamma_k)$	f_{qd}	ε_k	$\|\Gamma_k\|_2$	$\|L_k\|_2$
1	6660	1.123	-3.590×10^{2}	1.239×10^{12}	-1.066×10^{-1}	2.914×10^{-9}	4.447×10^{14}	7.641×10^{6}
2	6670	1.248	-6.557×10^{2}	6.530×10^{11}	-1.927×10^{-1}	4.338×10^{-9}	4.280×10^{14}	9.903×10^{6}
3	6680	4.245	-1.491×10^{3}	3.070×10^{11}	-7.415×10^{-1}	7.736×10^{-9}	4.565×10^{14}	1.069×10^{7}
4	7030	1.100	-2.161×10^{2}	1.680×10^{12}	-8.810×10^{-2}	1.745×10^{-10}	3.630×10^{14}	4.218×10^{6}
5	7060	1.277	-1.199×10^{3}	1.880×10^{11}	-2.104×10^{-1}	1.807×10^{-9}	2.253×10^{14}	9.086×10^{6}
6	7086	-0.136	-2.224×10^{16}	1.423×10^{13}	-8.121×10^{0}	8.349×10^{-8}	2.224×10^{16}	4.411×10^{8}
7	13180	1.009	-1.198×10^{2}	2.833×10^{12}	-8.620×10^{-3}	6.283×10^{-10}	3.394×10^{14}	4.819×10^{6}
8	13190	1.142	-2.626×10^{2}	1.419×10^{12}	-1.207×10^{-1}	9.316×10^{-10}	3.725×10^{14}	4.863×10^{6}
9	13240	1.400	-1.165×10^{10}	3.413×10^{7}	-2.770×10^{-1}	3.157×10^{-6}	3.495×10^{10}	1.689×10^{5}
10	13251	-0.045	-1.112×10^{12}	1.570×10^{9}	-2.260×10^{1}	1.222×10^{-4}	1.112×10^{12}	4.991×10^{6}
11	14877	-0.001	-2.512×10^{-13}	1.961×10^{14}	-1.729×10^{3}	5.709×10^{7}	4.970×10^{1}	6.482×10^{-5}
12	14878	-0.000	-8.865×10^{-11}	3.668×10^{14}	-1.033×10^{5}	9.817×10^{8}	5.018×10^{1}	4.434×10^{-6}
13	14899	1.303	-1.006×10^{-8}	1.398×10^{14}	-2.257×10^{-1}	7.846×10^{5}	1.917×10^{-1}	1.143×10^{-2}
14	14924	1.147	-4.746×10^{-1}	5.605×10^{14}	-1.246×10^{-1}	3.320×10^{5}	4.746×10^{-1}	7.682×10^{-1}
15	14927	-0.002	-8.956×10^{-7}	9.977×10^{12}	-5.220×10^{2}	1.245×10^{8}	2.933×10^{-2}	7.982×10^{-4}
16	14928	-0.025	-5.333×10^{-6}	3.597×10^{13}	-3.980×10^{1}	4.409×10^{10}	1.755×10^{-4}	8.141×10^{-7}
17	14931	1.000	-5.845×10^{-6}	4.442×10^{14}	-5.109×10^{-12}	1.763×10^{1}	5.845×10^{-6}	1.168×10^{-8}
18	14932	1.000	-6.026×10^{-6}	4.500×10^{14}	-5.976×10^{-9}	1.346×10^{2}	6.026×10^{-6}	1.895×10^{-7}
19	14933	1.000	-6.212×10^{-6}	4.514×10^{14}	-5.627×10^{-7}	1.730×10^{3}	6.212×10^{-6}	1.924×10^{-6}
20	14934	1.000	-6.406×10^{-6}	4.481×10^{14}	-2.541×10^{-4}	1.460×10^{4}	6.406×10^{-6}	3.996×10^{-5}
21	14935	1.008	-6.657×10^{-6}	4.497×10^{14}	-7.933×10^{-3}	2.502×10^{5}	6.658×10^{-6}	2.335×10^{-4}
22	14950	1.000	-9.609×10^{-7}	5.629×10^{13}	-1.203×10^{-10}	5.409×10^{2}	9.609×10^{-7}	1.306×10^{-8}
23	14951	1.000	-9.906×10^{-7}	5.634×10^{13}	-2.244×10^{-10}	8.149×10^{1}	9.907×10^{-7}	1.577×10^{-8}
24	14952	1.000	-1.021×10^{-6}	5.632×10^{13}	-5.105×10^{-8}	1.727×10^{2}	1.021×10^{-6}	2.655×10^{-7}
25	14953	1.000	-1.053×10^{-6}	5.632×10^{13}	-1.733×10^{-5}	2.546×10^{3}	1.053×10^{-6}	4.985×10^{-6}
26	14954	1.006	-1.094×10^{-6}	5.670×10^{13}	-5.893×10^{-3}	1.037×10^{5}	1.094×10^{-6}	9.506×10^{-5}
27	14989	-0.084	-2.503×10^{-12}	8.419×10^{14}	-1.251×10^{1}	1.370×10^{13}	4.137×10^{-11}	3.999×10^{-10}
28	15001	1.000	-6.573×10^{-26}	5.103×10^{13}	7.143×10^{-19}	5.990×10^{0}	3.360×10^{-12}	1.384×10^{-15}
29	15011	1.000	-1.016×10^{-25}	2.941×10^{13}	6.685×10^{-18}	4.101×10^{-1}	2.989×10^{-12}	2.394×10^{-15}
30	15012	1.000	-1.047×10^{-25}	2.943×10^{13}	2.234×10^{-17}	7.888×10^{1}	3.081×10^{-12}	3.314×10^{-15}
31	15021	1.000	-1.436×10^{-25}	2.155×10^{13}	8.075×10^{-19}	3.637×10^{0}	3.094×10^{-12}	5.060×10^{-16}
32	15022	1.000	-1.480×10^{-25}	2.155×10^{13}	4.863×10^{-18}	7.596×10^{1}	3.190×10^{-12}	5.318×10^{-16}
33	15031	1.000	-2.263×10^{-25}	1.784×10^{13}	6.048×10^{-19}	1.855×10^{0}	4.034×10^{-12}	7.268×10^{-16}
34	15032	1.000	-2.331×10^{-25}	1.784×10^{13}	8.322×10^{-18}	7.436×10^{1}	4.159×10^{-12}	8.697×10^{-16}
35	15041	1.000	-3.017×10^{-25}	1.809×10^{13}	8.102×10^{-19}	1.901×10^{0}	5.459×10^{-12}	9.776×10^{-16}
36	15042	1.000	-3.110×10^{-25}	1.809×10^{13}	1.088×10^{-17}	7.421×10^{1}	5.628×10^{-12}	1.103×10^{-15}
37	15051	1.000	-3.934×10^{-25}	1.880×10^{13}	1.090×10^{-18}	1.922×10^{0}	7.395×10^{-12}	1.319×10^{-15}
38	15052	1.000	-4.057×10^{-25}	1.879×10^{13}	1.450×10^{-17}	7.416×10^{1}	7.624×10^{-12}	1.451×10^{-15}
39	15061	1.000	-5.968×10^{-25}	1.680×10^{13}	1.470×10^{-18}	1.932×10^{0}	1.002×10^{-11}	1.784×10^{-15}
40	15062	1.000	-6.151×10^{-25}	1.680×10^{13}	1.948×10^{-17}	7.413×10^{1}	1.033×10^{-11}	1.935×10^{-15}
41	15071	1.000	-8.604×10^{-25}	1.579×10^{13}	1.986×10^{-18}	1.938×10^{0}	1.359×10^{-11}	2.414×10^{-15}
42	15072	1.000	-8.876×10^{-25}	1.579×10^{13}	2.628×10^{-17}	7.412×10^{1}	1.401×10^{-11}	2.598×10^{-15}

Já os estimadores RLS$_\mu$-UDU^T-HDP-DLQR mostraram-se insensíveis à escolha do fator de esquecimento tanto para o caso $\mu = 0.9$ quanto para $\mu = 0,97$ que puderam suportar milhares de iterações sem que o fenômeno da divergência se manifestasse. As Figuras 6.23 e 6.25 ilustram os comportamentos de convergência dos parâmetros p_{11}, p_{44}, p_{14} e p_{24} para os casos $\mu = 0.9$ e $\mu = 0.7$, respectivamente. Os resultados mostrados nas Figuras 6.24 e 6.26 ilustram características mais robustas de estabilidade numérica quando o parâmetro de positividade se mantém na faixa de validade durante todo o processo iterativo e o número de condição exibe um comportamento no qual não se observa variações abruptas em seus valores, os quais são mantidos limitados à um determinado intervalo após decorrido um período de tempo à transição (em torno da iteração 3200).

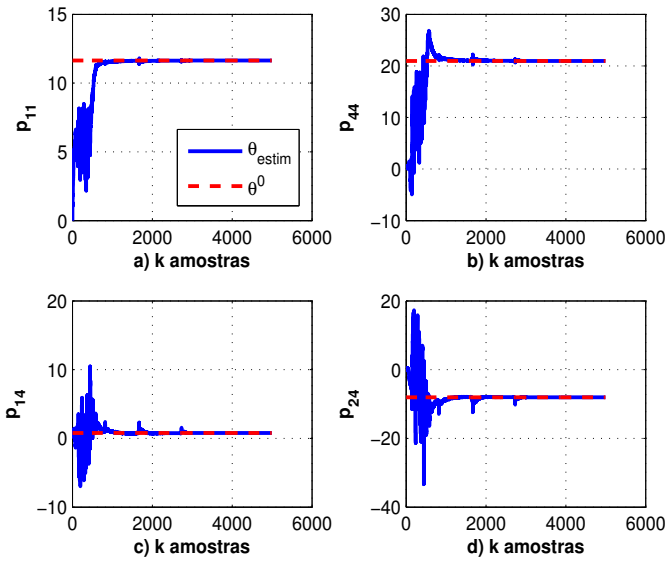

Figura 6.23: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.9$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

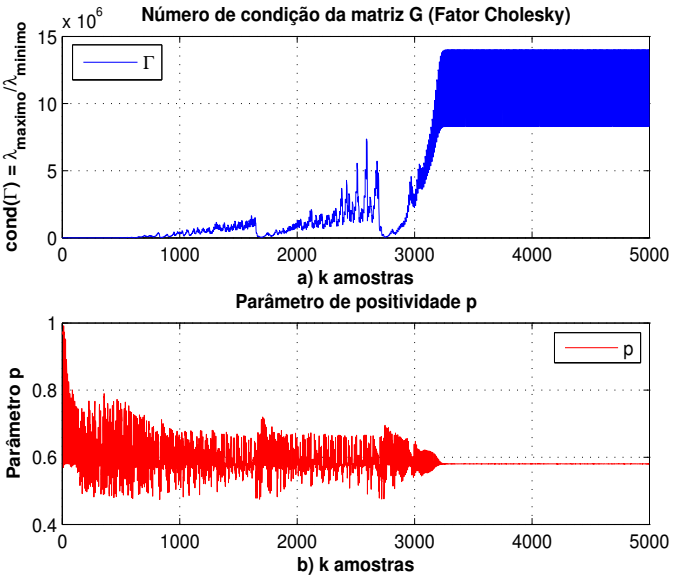

Figura 6.24: Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.9$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

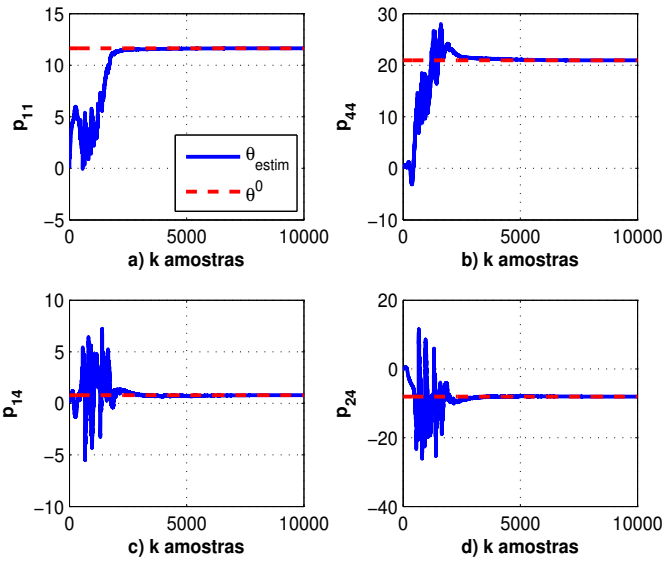

Figura 6.25: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.97$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

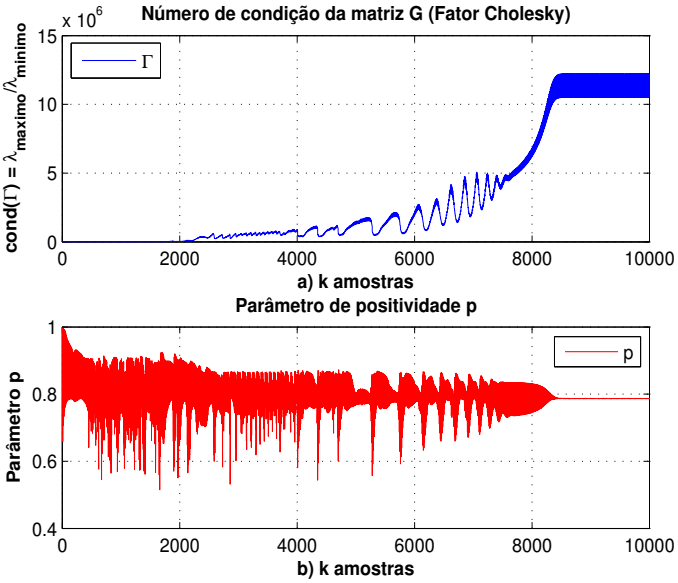

Figura 6.26: Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.97$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Em relação aos estimadores RLS$_\mu$-HDP-DLQR, pôde-se observar uma melhor convergência para os valores de $\mu = 0.95$ e $\mu = 0.96$, sem oscilações em torno do valor verdadeiro do parâmetro P durante o comportamento em regime permanente, como pode ser verificado nas Figuras 6.27 e 6.28 que ilustram a evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 5000 iterações para os casos $\mu = 0.95$ e $\mu = 0.96$, respectivamente.

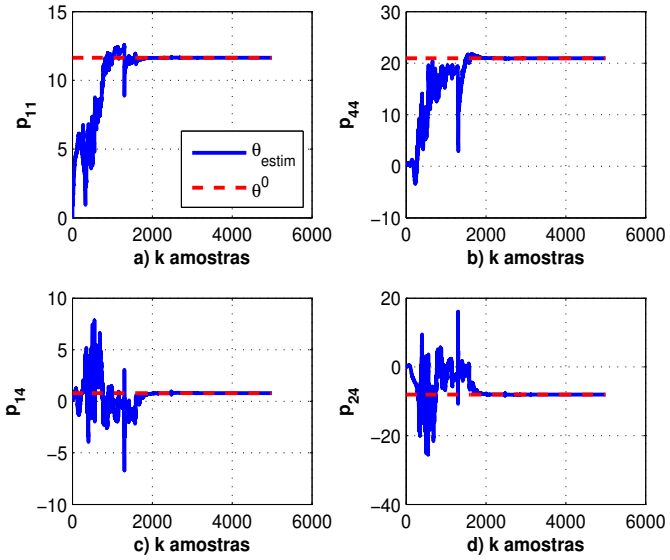

Figura 6.27: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.95$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

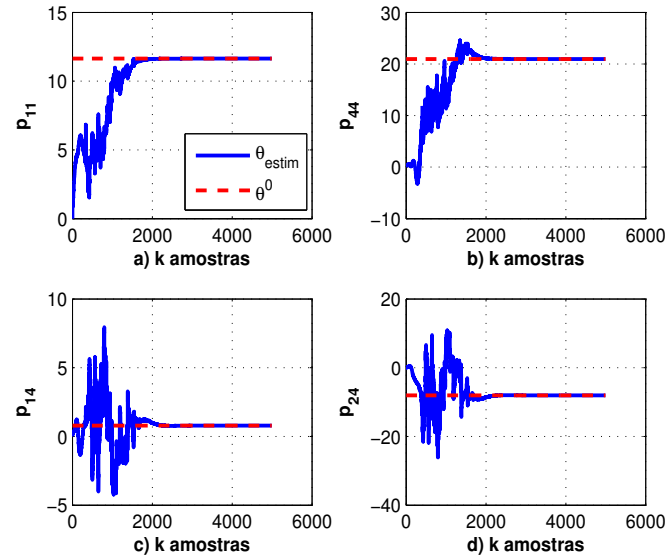

Figura 6.28: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.96$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Verifica-se que apesar da aparente normalidade desses comportamentos, os

resultados ilustrados nas Figuras 6.29-6.30 e 6.31 juntamente com os resultados apresentados nas Tabelas 6.15 e 6.16 mostram irregularidades no comportamento da matriz de covariância para ambos os valores de μ. No caso $\mu = 0.95$, evidencia-se a perda da positividade da matriz de covariância a partir da iteração 2000, enquanto que no caso $\mu = 0.96$, este fenômeno é manisfestado a partir da iteração 2320.

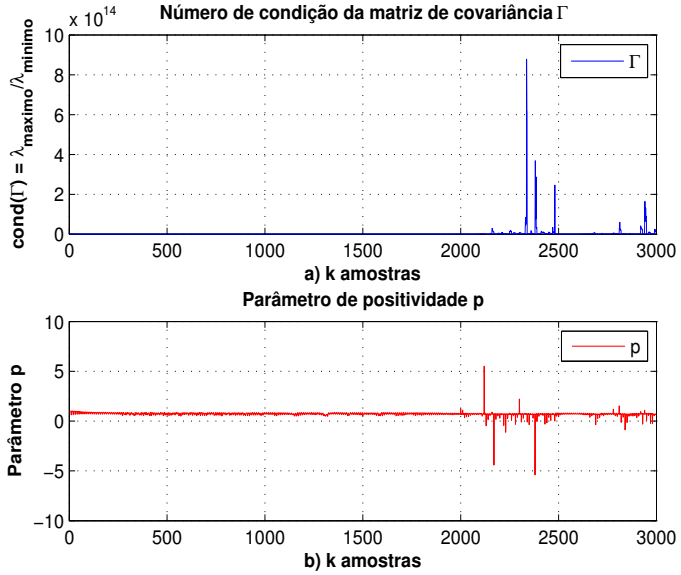

Figura 6.29: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 3000 iterações, com fator de esquecimento $\mu = 0.95$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

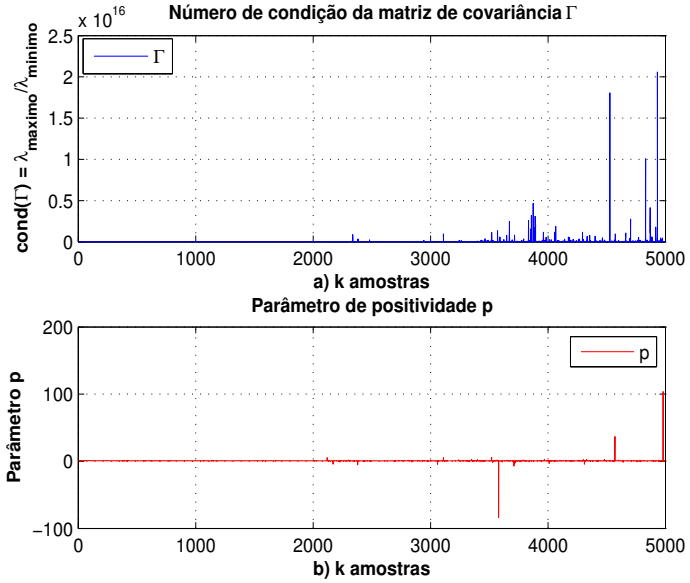

Figura 6.30: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.95$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Tabela 6.15: Ocorrências associadas com a perda da positividade da matriz de covariância Γ_k para um ciclo de 3000 iterações - Processo de estimação RLS$_\mu$-HDP-DLQR, com fator de esquecimento $\mu = 0.95$.

$k1$	k	p_k	$\lambda_{min}(\Gamma_k)$	$cond(\Gamma_k)$	f_{qd}	ε_k	$\|\Gamma_k\|_2$	$\|L_k\|_2$
1	2000	1.250	-5.271×10^{2}	1.015×10^{12}	-1.899×10^{-1}	4.692×10^{-9}	5.350×10^{14}	9.153×10^{6}
2	2010	1.087	-2.788×10^{3}	1.179×10^{11}	-7.597×10^{-2}	4.272×10^{-9}	3.284×10^{14}	1.818×10^{7}
3	2120	5.516	-1.105×10^{3}	1.891×10^{12}	-7.778×10^{-1}	2.873×10^{-10}	2.090×10^{15}	7.711×10^{6}
4	2130	-0.459	-5.201×10^{3}	6.764×10^{11}	-3.018×10^{0}	1.591×10^{-11}	3.353×10^{15}	4.871×10^{7}
5	2170	-4.420	-1.605×10^{3}	2.542×10^{12}	-1.165×10^{0}	5.872×10^{-12}	4.079×10^{15}	4.102×10^{6}
6	2220	-0.433	-4.353×10^{3}	1.380×10^{12}	-3.143×10^{0}	4.590×10^{-12}	6.007×10^{15}	1.038×10^{7}
7	2230	-1.117	-2.451×10^{3}	2.515×10^{12}	-1.800×10^{0}	4.117×10^{-11}	6.159×10^{15}	6.788×10^{6}
8	2280	-0.316	-5.514×10^{3}	7.466×10^{11}	-3.953×10^{0}	5.313×10^{-11}	4.113×10^{15}	9.901×10^{6}
9	2300	2.221	-7.270×10^{2}	8.052×10^{12}	-5.223×10^{-1}	6.611×10^{-11}	5.851×10^{15}	2.443×10^{6}
10	2310	-0.090	-1.294×10^{4}	7.047×10^{11}	-1.148×10^{1}	2.602×10^{-11}	6.727×10^{15}	1.380×10^{8}
11	2340	-0.091	-1.741×10^{4}	1.227×10^{12}	-1.135×10^{1}	6.134×10^{-11}	1.261×10^{16}	2.561×10^{7}
12	2350	-0.358	-5.122×10^{3}	2.755×10^{12}	-3.606×10^{0}	8.583×10^{-11}	1.409×10^{16}	3.129×10^{6}
13	2380	-5.396	-1.446×10^{3}	4.865×10^{12}	-1.126×10^{0}	1.151×10^{-10}	6.922×10^{15}	1.327×10^{7}
14	2390	-0.042	-3.382×10^{4}	4.396×10^{11}	-2.365×10^{1}	2.633×10^{-10}	4.596×10^{15}	3.849×10^{7}
15	2430	-0.311	-5.673×10^{3}	2.227×10^{12}	-4.010×10^{0}	1.574×10^{-12}	1.262×10^{16}	4.919×10^{6}
16	2440	-0.177	-8.876×10^{3}	1.984×10^{12}	-6.306×10^{0}	2.982×10^{-11}	1.741×10^{16}	6.398×10^{6}
17	2480	-0.003	-2.456×10^{16}	3.187×10^{12}	-3.089×10^{2}	1.232×10^{-11}	2.456×10^{16}	6.064×10^{8}
18	2690	-0.360	-5.118×10^{3}	7.013×10^{11}	-3.591×10^{0}	5.980×10^{-11}	3.587×10^{15}	1.431×10^{7}
19	2780	1.202	-2.690×10^{2}	6.216×10^{12}	-1.598×10^{-1}	3.117×10^{-10}	1.670×10^{15}	7.443×10^{6}
20	2810	1.519	-3.997×10^{2}	8.629×10^{12}	-3.246×10^{-1}	3.638×10^{-11}	3.418×10^{15}	9.591×10^{6}
21	2820	-0.123	-1.181×10^{4}	4.869×10^{11}	-8.673×10^{0}	9.790×10^{-12}	4.648×10^{15}	1.332×10^{7}
22	2840	-0.885	-2.805×10^{3}	2.274×10^{12}	-2.024×10^{0}	7.845×10^{-12}	6.349×10^{15}	8.495×10^{6}
23	2850	-0.144	-1.057×10^{4}	6.411×10^{11}	-7.533×10^{0}	3.297×10^{-10}	5.710×10^{15}	2.331×10^{7}
24	2930	-0.305	-5.878×10^{3}	6.013×10^{11}	-4.060×10^{0}	2.629×10^{-11}	3.473×10^{15}	1.688×10^{7}
25	2940	1.096	-1.229×10^{2}	4.041×10^{13}	-8.353×10^{-2}	1.442×10^{-11}	4.968×10^{15}	5.302×10^{6}
26	2970	-0.219	-6.640×10^{3}	2.115×10^{12}	-5.295×10^{0}	1.134×10^{-11}	1.295×10^{16}	9.193×10^{6}
27	2980	-0.250	-6.507×10^{3}	2.355×10^{12}	-4.747×10^{0}	9.620×10^{-11}	1.531×10^{16}	1.004×10^{7}

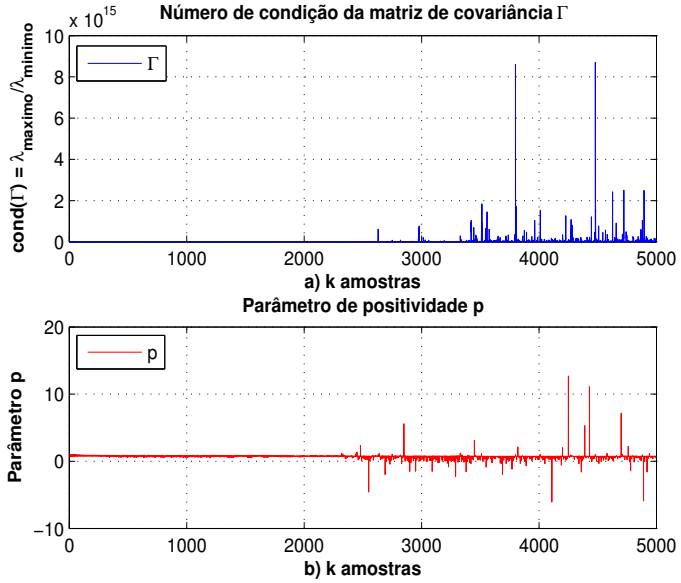

Figura 6.31: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.96$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Tabela 6.16: Ocorrências associadas com a perda da positividade da matriz de covariância Γ_k para um ciclo de 3000 iterações - Processo de estimação RLS$_\mu$-HDP-DLQR, com fator de esquecimento $\mu = 0.96$.

$k1$	k	p_k	$\lambda_{min}(\Gamma_k)$	$cond(\Gamma_k)$	f_{qd}	ε_k	$\|\Gamma_k\|_2$	$\|L_k\|_2$
1	2320	1.218	-4.658×10^{2}	1.090×10^{12}	-1.720×10^{-1}	8.547×10^{-10}	5.075×10^{14}	8.229×10^{6}
2	2330	1.083	-1.274×10^{3}	3.327×10^{11}	-7.341×10^{-2}	8.824×10^{-10}	4.238×10^{14}	1.625×10^{7}
3	2440	1.094	-1.476×10^{2}	1.687×10^{13}	-8.210×10^{-2}	5.896×10^{-11}	2.490×10^{15}	5.781×10^{6}
4	2450	1.379	-4.237×10^{2}	6.884×10^{12}	-2.638×10^{-1}	5.763×10^{-11}	2.915×10^{15}	9.797×10^{6}
5	2480	2.350	-7.675×10^{2}	6.452×10^{12}	-5.515×10^{-1}	3.028×10^{-11}	4.950×10^{15}	2.328×10^{6}
6	2550	-4.572	-1.629×10^{3}	3.054×10^{12}	-1.170×10^{0}	4.343×10^{-11}	4.971×10^{15}	6.511×10^{6}
7	2640	1.128	-1.648×10^{2}	1.477×10^{13}	-1.089×10^{-1}	6.324×10^{-12}	2.433×10^{15}	1.137×10^{6}
8	2690	-1.988	-2.058×10^{3}	4.725×10^{12}	-1.443×10^{0}	3.780×10^{-13}	9.723×10^{15}	3.128×10^{6}
9	2730	-0.431	-4.466×10^{3}	2.402×10^{12}	-3.186×10^{0}	9.288×10^{-12}	1.061×10^{16}	5.795×10^{6}
10	2750	1.054	-6.814×10^{1}	8.401×10^{13}	-4.878×10^{-2}	3.605×10^{-10}	5.703×10^{15}	2.512×10^{6}
11	2830	-0.365	-4.949×10^{3}	1.419×10^{12}	-3.587×10^{0}	1.500×10^{-12}	6.997×10^{15}	5.472×10^{6}
12	2850	5.589	-1.173×10^{3}	1.043×10^{13}	-7.882×10^{-1}	4.930×10^{-13}	1.217×10^{16}	4.611×10^{6}
13	2860	-0.012	-1.098×10^{5}	8.735×10^{11}	-7.813×10^{1}	2.928×10^{-12}	1.377×10^{16}	8.022×10^{7}
14	2890	-0.116	-1.314×10^{4}	1.471×10^{12}	-9.236×10^{0}	2.374×10^{-11}	1.330×10^{16}	4.829×10^{6}
15	2900	-1.508	-2.189×10^{3}	5.053×10^{12}	-1.597×10^{0}	9.950×10^{-11}	1.104×10^{16}	2.089×10^{6}
16	2910	-0.185	-8.708×10^{3}	1.169×10^{12}	-6.140×10^{0}	2.008×10^{-12}	9.427×10^{15}	1.019×10^{7}
17	2950	-1.491	-2.220×10^{3}	3.971×10^{12}	-1.604×10^{0}	1.800×10^{-12}	8.809×10^{15}	3.487×10^{6}

Entretanto, percebe-se que embora o parâmetro de positividade passa a assumir valores fora de sua faixa de validade e o número de condição apresenta variações drásticas em seus valores após um certo período de tempo em que os estimadores já alcançaram o regime permanente, esses comportamentos não tem mais qualquer efeito na perda da estabilidade do parâmetro θ. Esta forma de comportamento de convergência pode ser entendida da seguinte forma: A equação que atualiza o vetor de parâmetro θ, a qual é dada por

$$\theta_k = \theta_{k-1} + L_k \varepsilon_k, \tag{6.6}$$

sendo L_k o ganho RLS que é definido por

$$L_k = \frac{\Gamma_{k-1}\phi_k}{\mu + \phi^T{}_k \Gamma_{k-1}\phi_k}, \tag{6.7}$$

e ε_k é o erro da estimativa que é dado por

$$\varepsilon_k = y_k - \phi_k^T \theta_{k-1}, \tag{6.8}$$

mostra que, após haver convergido o parâmetro θ, os termos $L_k\varepsilon_k$ devem assumir valores muito pequenos, ou mesmo nulos. Como os erros das estimativas tendem a se anular, a norma do vetor L_k pode assumir uma gama muita ampla de valores sem que ocorram grandes alterações nas componentes do parâmetro θ. Isto faz com que o produto $L_k\varepsilon_k$ permaneça em limites relativamente aceitáveis, mantendo os estimadores indefinidamente estáveis. As Figuras 6.32 e 6.33 ilustram a evolução da norma de $L_k\varepsilon_k$ para os valores $\mu = 0.95$ e $\mu = 0.96$, respectivamente, para um ciclo de 10000 iterações.

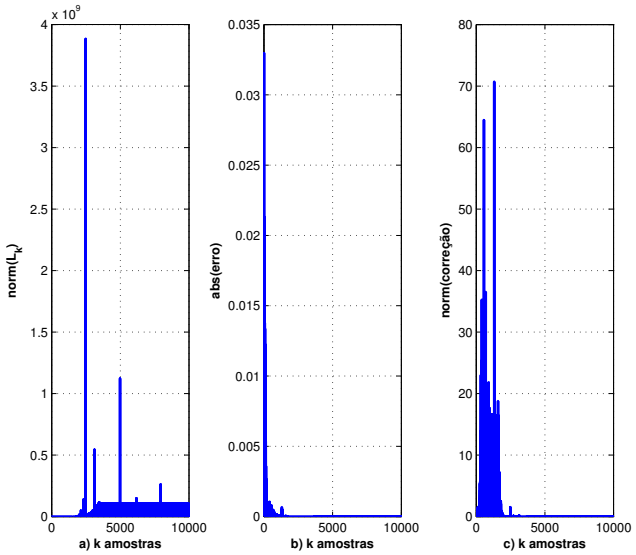

Figura 6.32: Norma do fator $L_k\varepsilon_k$ para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.95$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

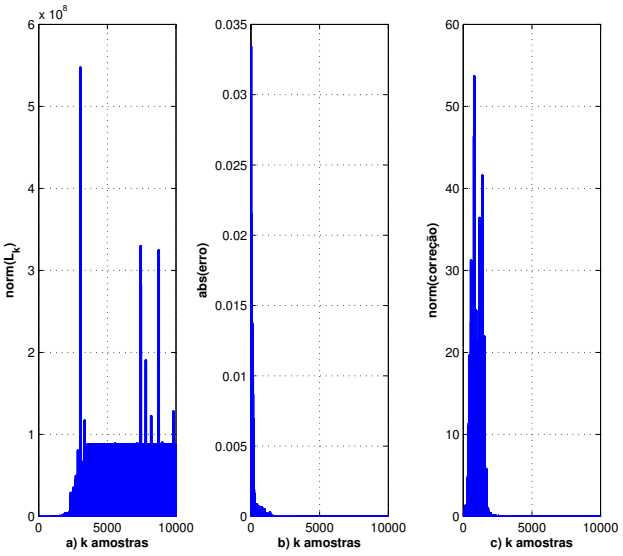

Figura 6.33: Norma do fator $L_k\varepsilon_k$ para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.96$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Já o fator $\mu = 0.94$ produziu resultados menos satisfatórios quando comparado com os casos $\mu = 0.95$ e $\mu = 0.96$, pois ocasionou grandes oscilações nos valores do parâmetro P por um período de tempo consideravelmente longo,

aproximadamente da iteração 7000 à 18000 (observe a Figura 6.34 que ilustra o comportamento de convergência dos parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 30000 iterações). Os resultados apresentados na Tabela 6.17 evidenciam anomalias no comportamento da matriz de covariância a partir da iteração 3750, com 25 ocorrências relacionadas com a perda da positividade da matriz de covariância da iteração 3750 à 6790, sendo que da iteração 6791 à 10279 não se verifica tal fenômeno.

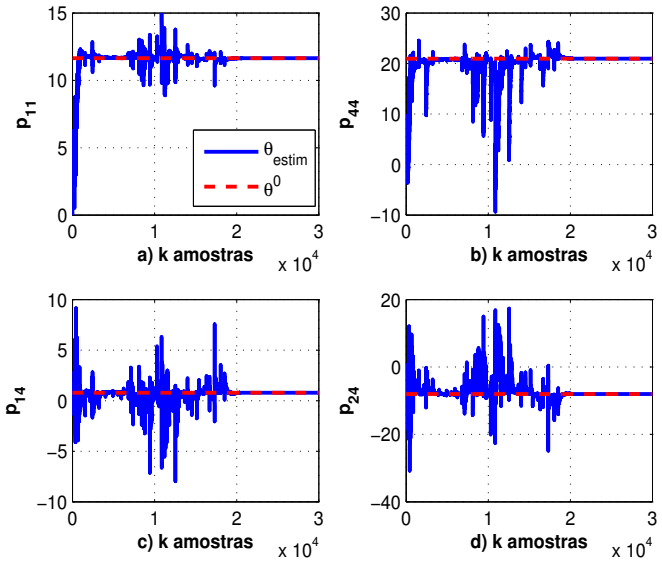

Figura 6.34: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 30000 iterações, com fator de esquecimento $\mu = 0.94$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

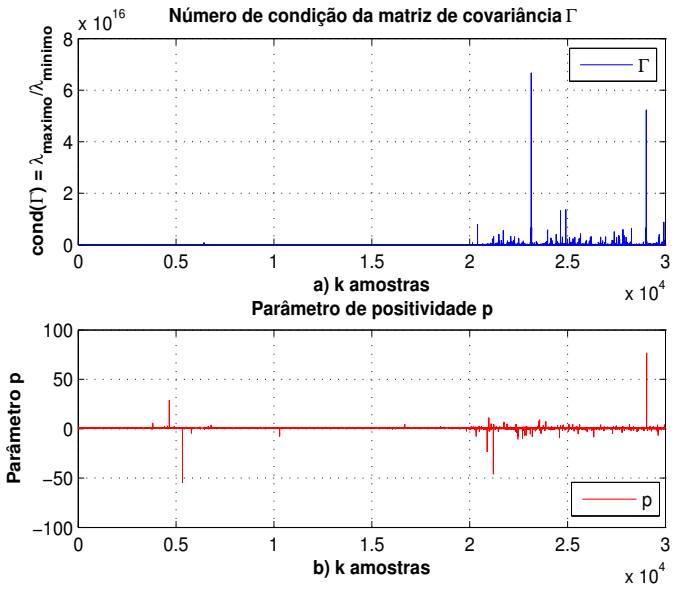

Figura 6.35: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 30000 iterações, com fator de esquecimento $\mu = 0.94$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Tabela 6.17: Ocorrências associadas com a perda da positividade da matriz de covariância Γ_k para um ciclo de 20000 iterações - Processo de estimação RLS$_\mu$-HDP-DLQR, com fator de esquecimento $\mu = 0.94$.

$k1$	k	p_k	$\lambda_{min}(\Gamma_k)$	$cond(\Gamma_k)$	f_{qd}	ε_k	$\|\Gamma_k\|_2$	$\|L_k\|_2$
1	3750	1.114	-3.142×10^{2}	2.153×10^{12}	-9.602×10^{-2}	2.560×10^{-9}	6.765×10^{14}	8.340×10^{6}
2	3760	1.156	-3.497×10^{2}	1.937×10^{12}	-1.268×10^{-1}	6.720×10^{-10}	6.771×10^{14}	8.427×10^{6}
3	3770	-0.643	-3.960×10^{3}	1.716×10^{11}	-2.402×10^{0}	7.295×10^{-10}	6.793×10^{14}	1.585×10^{7}
4	3790	1.082	-1.717×10^{2}	6.064×10^{12}	-7.139×10^{-2}	4.846×10^{-10}	1.041×10^{15}	7.116×10^{6}
5	3810	5.526	-1.248×10^{13}	1.978×10^{10}	-7.699×10^{-1}	3.333×10^{-9}	1.980×10^{14}	1.807×10^{7}
6	4400	1.164	-4.042×10^{2}	1.458×10^{12}	-1.326×10^{-1}	1.108×10^{-9}	5.893×10^{14}	8.544×10^{6}
7	4410	2.437	-1.148×10^{3}	6.436×10^{11}	-5.543×10^{-1}	1.035×10^{-9}	7.386×10^{14}	1.292×10^{7}
8	4560	2.747	-9.816×10^{2}	9.045×10^{11}	-5.978×10^{-1}	1.059×10^{-9}	8.875×10^{14}	9.181×10^{6}
9	4640	1.206	-2.720×10^{2}	2.845×10^{12}	-1.604×10^{-1}	7.927×10^{-10}	7.738×10^{14}	4.668×10^{6}
10	4650	28.808	-1.631×10^{3}	5.448×10^{11}	-9.074×10^{-1}	6.453×10^{-10}	8.876×10^{14}	1.461×10^{7}
11	4900	2.245	-8.078×10^{2}	1.556×10^{12}	-5.213×10^{-1}	2.884×10^{-10}	1.257×10^{15}	8.133×10^{6}
12	4920	1.100	-4.553×10^{14}	1.221×10^{12}	-8.574×10^{-2}	1.455×10^{-9}	4.556×10^{14}	1.563×10^{7}
13	5330	-54.442	-1.515×10^{3}	5.094×10^{11}	-9.573×10^{-1}	1.470×10^{-9}	7.715×10^{14}	9.089×10^{6}
14	5500	1.297	-4.680×10^{2}	2.048×10^{12}	-2.155×10^{-1}	3.053×10^{-10}	9.586×10^{14}	9.328×10^{6}
15	5530	1.553	-2.491×10^{15}	1.490×10^{12}	-3.349×10^{-1}	3.610×10^{-9}	2.491×10^{15}	6.284×10^{7}
16	5534	1.016	-5.777×10^{15}	2.679×10^{12}	-1.499×10^{-2}	1.580×10^{-8}	5.777×10^{15}	5.392×10^{7}
17	5770	1.001	-1.707×10^{1}	9.052×10^{13}	-6.971×10^{-4}	2.292×10^{-10}	1.546×10^{15}	4.370×10^{6}
18	5780	-4.491	-1.946×10^{3}	6.407×10^{11}	-1.149×10^{0}	3.787×10^{-10}	1.238×10^{15}	1.705×10^{7}
19	5920	1.406	-4.341×10^{2}	2.804×10^{12}	-2.716×10^{-1}	5.720×10^{-10}	1.217×10^{15}	6.338×10^{6}
20	6220	1.703	-6.337×10^{2}	1.820×10^{12}	-3.881×10^{-1}	4.996×10^{-10}	1.153×10^{15}	8.840×10^{6}
21	6430	1.392	-4.305×10^{2}	2.008×10^{12}	-2.646×10^{-1}	2.256×10^{-10}	8.645×10^{14}	5.810×10^{6}
22	6440	-0.590	-5.544×10^{3}	1.047×10^{11}	-2.532×10^{0}	5.166×10^{-10}	5.797×10^{14}	2.749×10^{7}
23	6640	2.378	-1.141×10^{3}	4.582×10^{11}	-5.447×10^{-1}	1.215×10^{-9}	5.230×10^{14}	1.092×10^{7}
24	6780	1.013	-6.052×10^{1}	2.135×10^{13}	-1.241×10^{-2}	2.148×10^{-10}	1.292×10^{15}	5.711×10^{6}
25	6790	3.445	-2.083×10^{3}	2.349×10^{11}	-6.671×10^{-1}	1.405×10^{-9}	4.889×10^{14}	1.590×10^{7}
26	10280	1.177	-3.453×10^{2}	2.447×10^{12}	-1.415×10^{-1}	7.988×10^{-10}	8.450×10^{14}	8.753×10^{6}
27	10290	-7.611	-1.614×10^{3}	3.991×10^{11}	-1.064×10^{0}	9.102×10^{-10}	6.442×10^{14}	7.835×10^{6}
28	15890	1.516	-6.867×10^{2}	4.757×10^{11}	-3.201×10^{-1}	1.536×10^{-9}	3.267×10^{14}	5.997×10^{6}
29	15900	-0.107	-2.286×10^{4}	2.760×10^{10}	-9.737×10^{0}	1.324×10^{-9}	3.440×10^{14}	3.056×10^{7}
30	16030	1.088	-2.767×10^{2}	2.209×10^{12}	-7.610×10^{-2}	1.041×10^{-9}	6.112×10^{14}	7.820×10^{6}
31	16040	1.051	-6.108×10^{2}	7.966×10^{11}	-4.575×10^{-2}	4.044×10^{-11}	4.865×10^{14}	1.193×10^{7}
32	16050	-0.392	-7.796×10^{3}	6.062×10^{10}	-3.341×10^{0}	1.029×10^{-9}	4.707×10^{14}	2.571×10^{7}
33	16650	1.773	-5.888×10^{2}	4.548×10^{12}	-4.098×10^{-1}	1.658×10^{-10}	2.669×10^{15}	8.845×10^{6}
34	16680	4.447	-2.354×10^{3}	7.278×10^{11}	-7.286×10^{-1}	2.915×10^{-9}	1.712×10^{15}	3.924×10^{7}
35	16695	1.154	-1.366×10^{13}	5.091×10^{9}	-1.251×10^{-1}	1.014×10^{-6}	1.366×10^{13}	3.011×10^{6}
36	18510	2.180	-1.770×10^{3}	1.951×10^{11}	-5.088×10^{-1}	2.860×10^{-9}	3.453×10^{14}	1.337×10^{7}
37	19200	1.245	-1.076×10^{3}	1.239×10^{11}	-1.848×10^{-1}	5.850×10^{-9}	1.333×10^{14}	6.743×10^{6}
38	19450	1.334	-6.742×10^{2}	7.106×10^{11}	-2.353×10^{-1}	2.281×10^{-9}	4.791×10^{14}	9.634×10^{6}
39	19460	-0.454	-1.439×10^{4}	2.219×10^{10}	-3.012×10^{0}	7.960×10^{-11}	2.731×10^{14}	3.392×10^{7}
40	19830	-2.405	-1.869×10^{3}	1.311×10^{12}	-1.331×10^{0}	9.369×10^{-11}	2.450×10^{15}	7.043×10^{6}
41	19990	1.096	-3.678×10^{2}	2.163×10^{12}	-8.202×10^{-2}	6.768×10^{-10}	7.953×10^{14}	1.025×10^{7}
42	20000	-0.825	-4.072×10^{3}	1.765×10^{11}	-2.079×10^{0}	8.366×10^{-10}	7.181×10^{14}	2.099×10^{7}

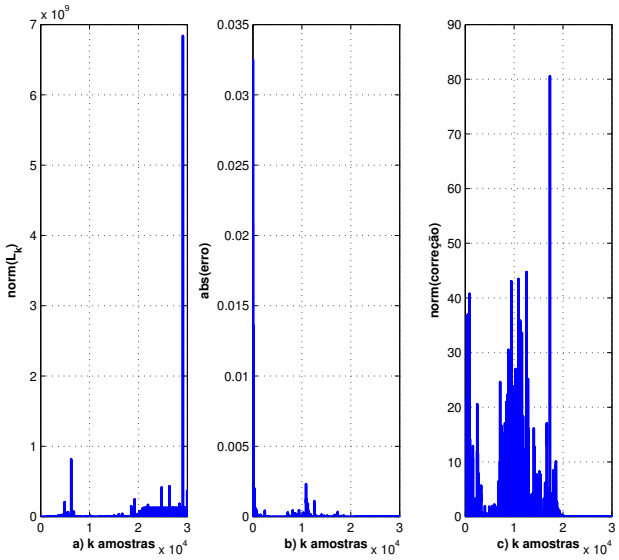

Figura 6.36: Norma do fator $L_k\varepsilon_k$ para um ciclo de 30000 iterações, com fator de esquecimento $\mu = 0.94$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Outros valores de μ como 0.93 contribuiu para acelerar o processo de convergência dos estimadores RLS$_\mu$-HDP-DLQR (em torno da iteração 1699), no entanto, os resultados ilustrados na Figura 6.37 mostram alguns desajustes em torno do valor verdadeiro entre as iterações 2180 e 3000. Nota-se também a partir da Figura 6.38 variações significativas no número de condição e irregularidades no parâmetro de positividade. A Tabela 6.18 fornece o número de ocorrências associadas com a perda da positividade da matriz de covariância para um ciclo de 3500 iterações, evidenciando anomalia no comportamento do parâmetro de positividade antes da iteração 2180.

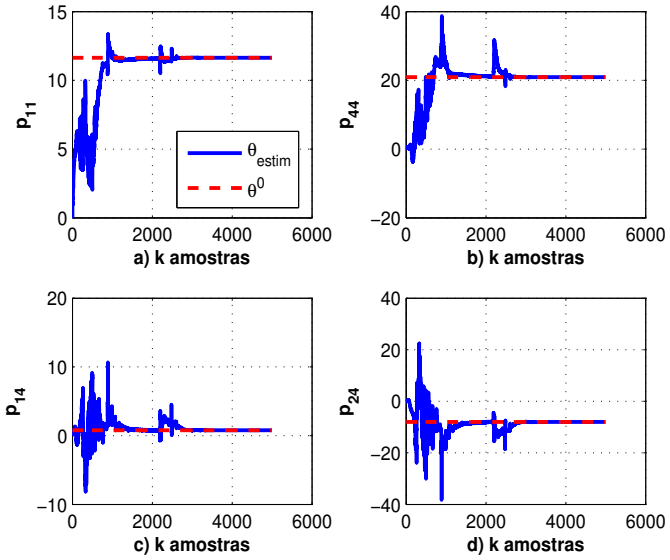

Figura 6.37: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.93$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

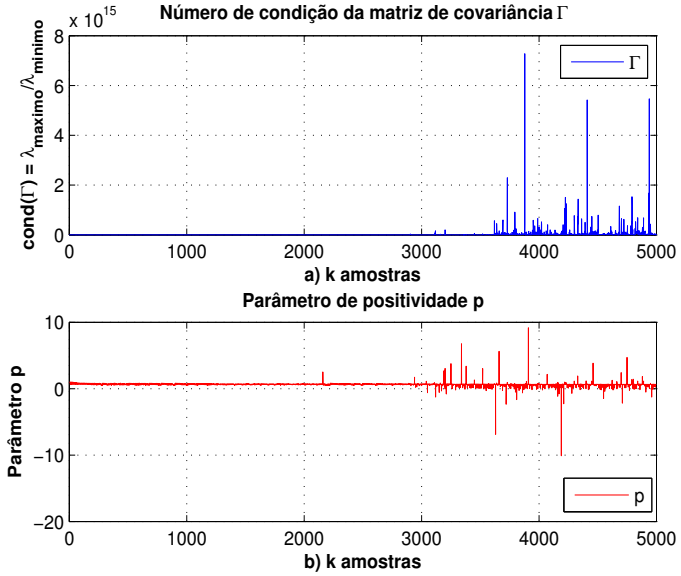

Figura 6.38: Número de condição da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.93$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Tabela 6.18: Ocorrências associadas com a perda da positividade da matriz de covariância Γ_k para um ciclo de 3500 iterações - Processo de estimação RLS$_\mu$-HDP-DLQR, com fator de esquecimento $\mu = 0.93$.

$k1$	k	p_k	$\lambda_{min}(\Gamma_k)$	$cond(\Gamma_k)$	f_{qd}	ε_k	$\|\Gamma_k\|_2$	$\|L_k\|_2$
1	2160	2.513	-1.070×10^{3}	6.256×10^{11}	-5.600×10^{-1}	1.791×10^{-9}	6.691×10^{14}	1.097×10^{7}
2	2940	1.725	-5.620×10^{2}	3.271×10^{12}	-3.909×10^{-1}	3.701×10^{-10}	1.838×10^{15}	5.675×10^{6}
3	3040	1.111	-3.077×10^{2}	2.175×10^{12}	-9.263×10^{-2}	1.972×10^{-9}	6.693×10^{14}	8.109×10^{6}
4	3050	-0.612	-4.622×10^{3}	1.299×10^{11}	-2.450×10^{0}	1.225×10^{-9}	6.006×10^{14}	1.773×10^{7}
5	3120	-1.268	-2.498×10^{3}	9.557×10^{11}	-1.664×10^{0}	3.700×10^{-10}	2.386×10^{15}	1.298×10^{7}
6	3150	-0.560	-3.600×10^{3}	9.772×10^{11}	-2.591×10^{0}	2.338×10^{-10}	3.518×10^{15}	1.108×10^{7}
7	3170	-0.359	-5.016×10^{3}	3.987×10^{11}	-3.518×10^{0}	3.457×10^{-10}	2.000×10^{15}	1.309×10^{7}
8	3190	2.675	-8.319×10^{2}	2.204×10^{12}	-5.824×10^{-1}	2.103×10^{-10}	1.833×10^{15}	6.240×10^{6}
9	3200	-0.045	-1.917×10^{15}	1.254×10^{14}	-2.143×10^{1}	1.261×10^{-10}	1.917×10^{15}	2.216×10^{8}
10	3201	3.038	-6.439×10^{15}	2.030×10^{14}	-6.238×10^{-1}	2.961×10^{-9}	6.439×10^{15}	7.757×10^{7}
11	3210	-0.810	-3.230×10^{3}	2.670×10^{11}	-2.079×10^{0}	7.514×10^{-9}	8.569×10^{14}	9.600×10^{6}
12	3250	3.773	-9.478×10^{2}	2.815×10^{12}	-6.835×10^{-1}	6.361×10^{-11}	2.668×10^{15}	4.594×10^{6}
13	3260	-0.016	-8.442×10^{4}	5.619×10^{11}	-5.927×10^{1}	1.458×10^{-11}	7.811×10^{15}	1.047×10^{8}
14	3290	-0.197	-8.086×10^{3}	9.417×10^{11}	-5.651×10^{0}	1.951×10^{-11}	7.560×10^{15}	1.407×10^{7}
15	3340	6.770	-1.083×10^{3}	8.320×10^{12}	-7.926×10^{-1}	1.758×10^{-11}	9.005×10^{15}	2.744×10^{6}
16	3370	-0.088	-1.592×10^{4}	8.779×10^{11}	-1.148×10^{1}	1.604×10^{-11}	1.330×10^{16}	1.086×10^{7}
17	3380	3.392	-9.022×10^{2}	1.482×10^{13}	-6.559×10^{-1}	6.360×10^{-11}	1.337×10^{16}	2.307×10^{6}
18	3450	1.310	-3.522×10^{2}	6.055×10^{13}	-2.200×10^{-1}	3.484×10^{-11}	2.133×10^{16}	7.616×10^{6}

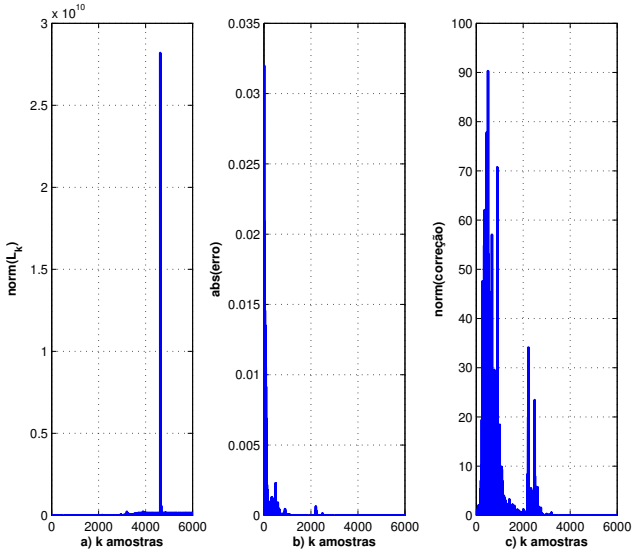

Figura 6.39: Norma do fator $L_k\varepsilon_k$ para um ciclo de 6000 iterações, com fator de esquecimento $\mu = 0.93$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

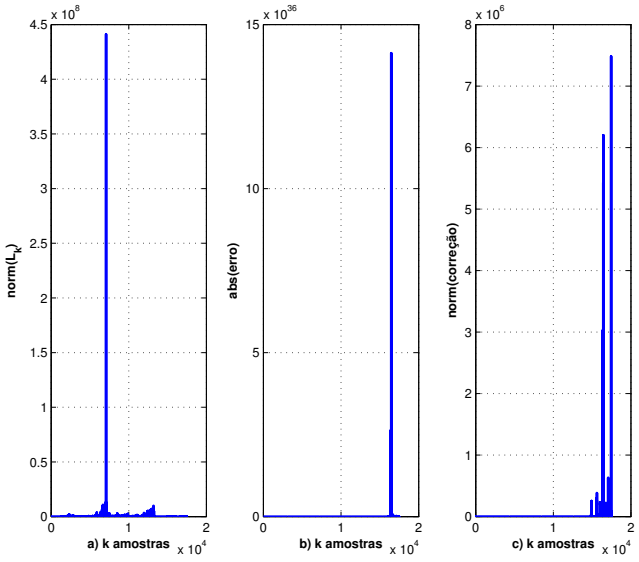

Figura 6.40: Norma do fator $L_k\varepsilon_k$ para um ciclo de 17500 iterações, com fator de esquecimento $\mu = 0.97$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

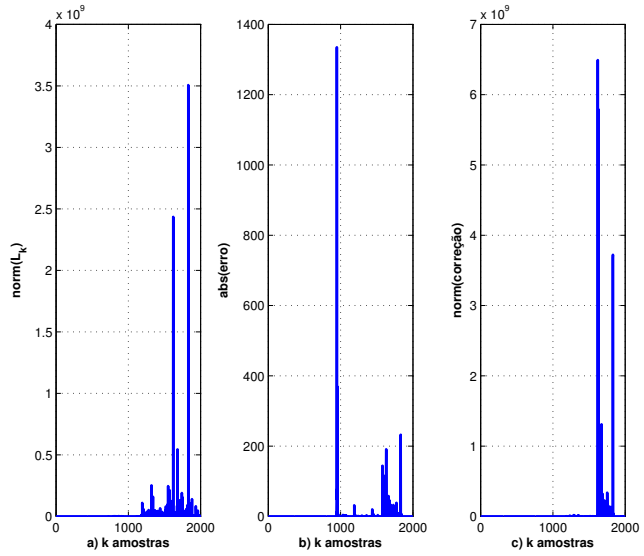

Figura 6.41: Norma do fator $L_k\varepsilon_k$ para um ciclo de 2000 iterações, com fator de esquecimento $\mu = 0.9$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Quanto ao comportamento dos estimadores RLS$_\mu$-UDU^T-HDP-DLQR para os casos $\mu = 0.95$ e $\mu = 0.96$, observa-se uma convergência mais lenta quando comparada com a convergência dos estimadores RLS$_\mu$-HDP-DLQR, conforme ilustrado

nas Figuras 6.42 e 6.44 em comparação com as Figuras 6.27 e 6.28. Entretanto, não se verifica anomalias no comportamento do parâmetro de positividade, como também percebe-se um comportamento de equilíbrio no número de condição, Figuras 6.43 e 6.45.

Já o fator $\mu = 0.94$ se destaca por produzir um comportamento de convergência com poucas oscilações e um tempo de acomodação consideravelmente mais rápido em relação ao do estimador RLS$_\mu$-HDP-DLQR (em torno da iteração 1977). De forma geral, os estimadores RLS$_\mu$-UDU^T-HDP-DLQR mostraram-se bem comportados para todos os valores de μ considerados nesta seção, garantindo uma maior estabilidade numérica.

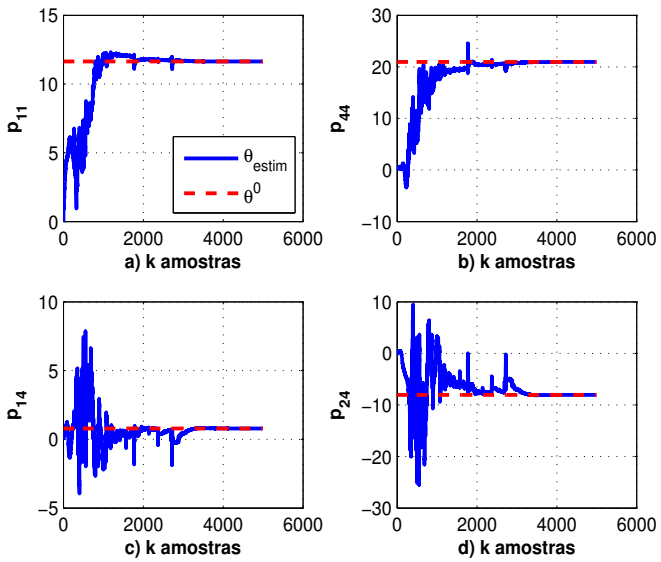

Figura 6.42: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.95$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

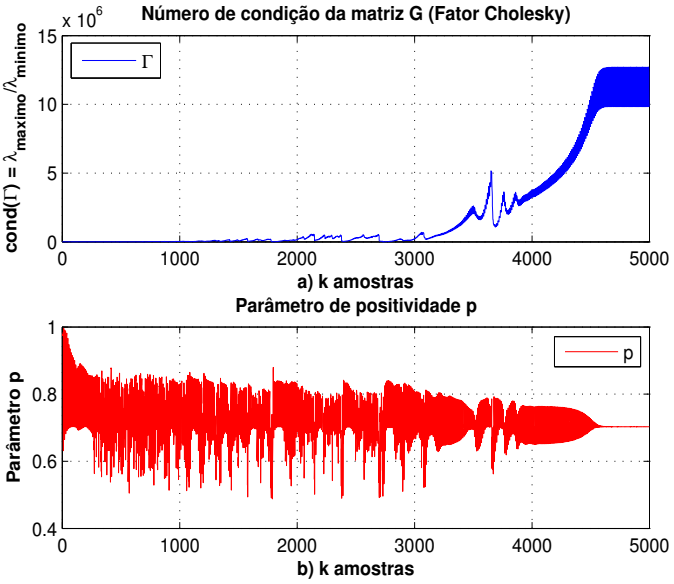

Figura 6.43: Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.95$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

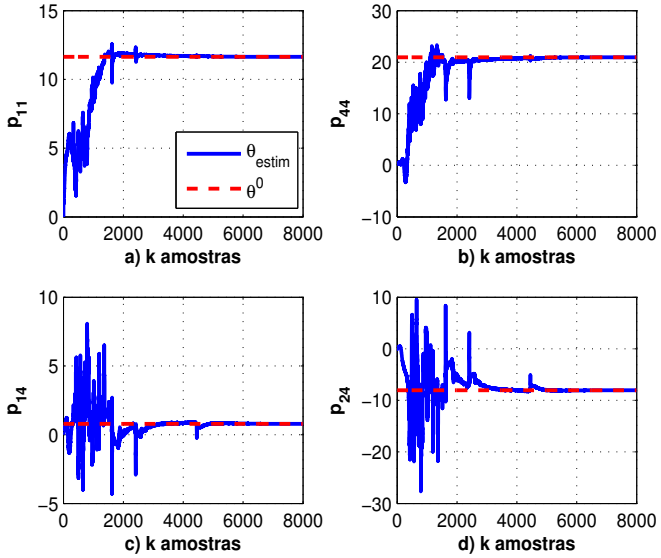

Figura 6.44: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.96$ - Algoritmo RLS-$_\mu$-UDU^T-HDP-DLQR Otimístico.

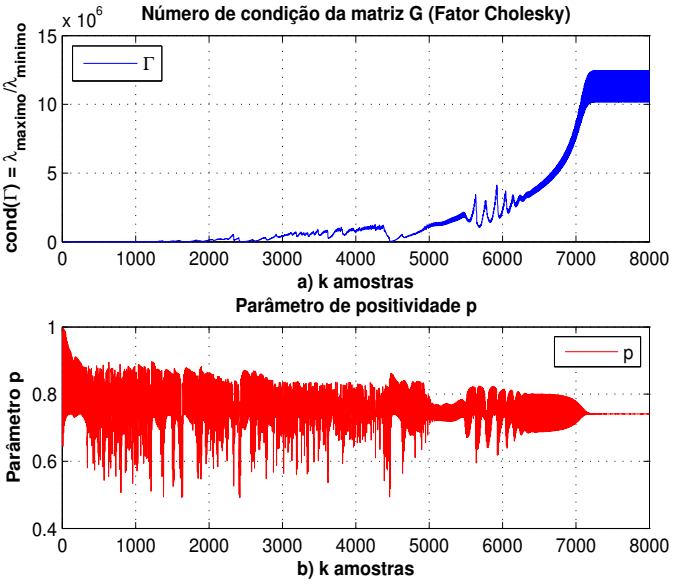

Figura 6.45: Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.96$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

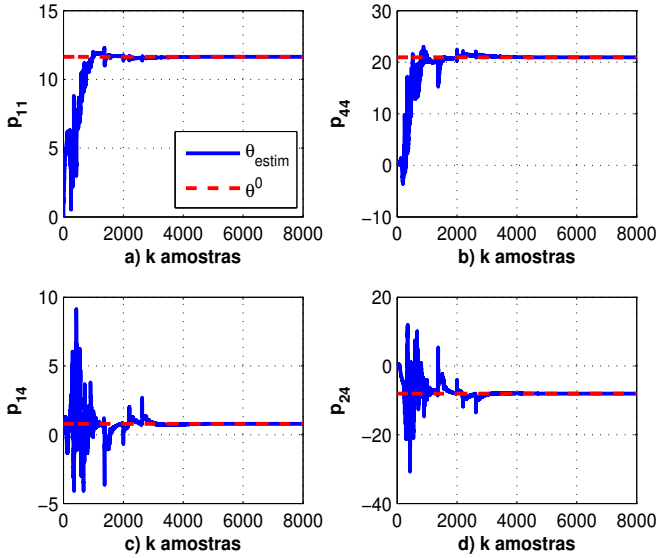

Figura 6.46: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.94$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

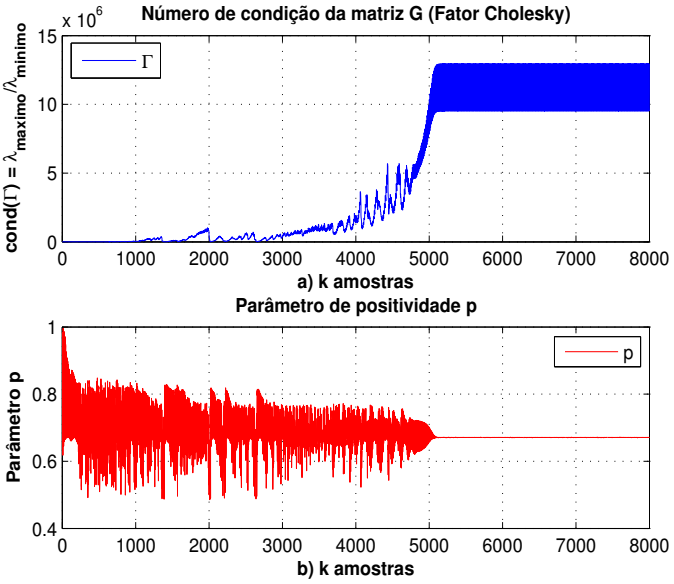

Figura 6.47: Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.94$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

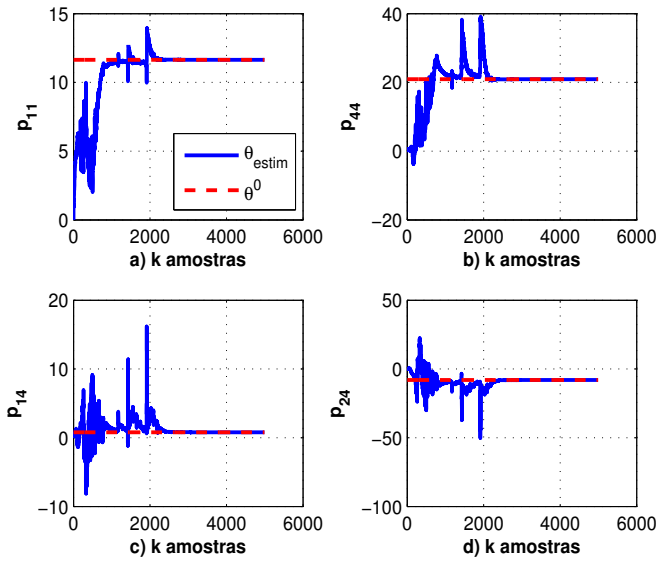

Figura 6.48: Evolução do processo iterativo para os parâmetros p_{11}, p_{44}, p_{14} e p_{24} para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.93$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

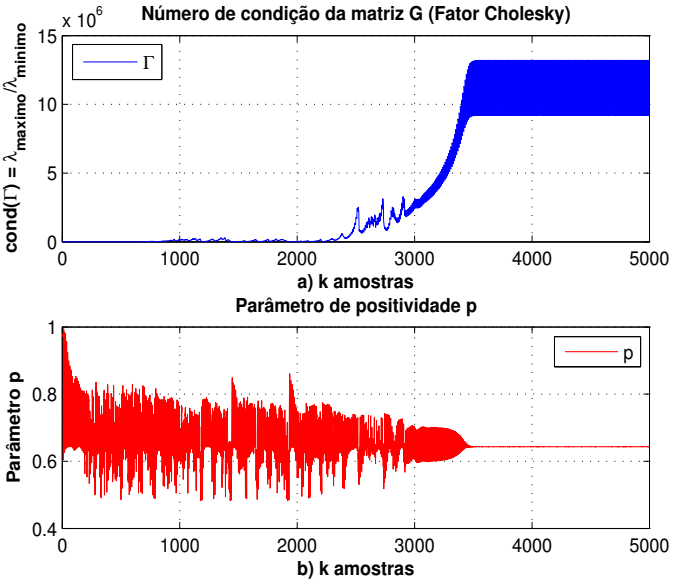

Figura 6.49: Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 5000 iterações, com fator de esquecimento $\mu = 0.93$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

6.2.7 Experimentos com Perturbações Paramétricas na Planta

Nesta seção, apresentam-se os resultados relacionados à perturbações paramétricas na planta para avaliar o desempenho dos métodos desenvolvidos neste trabalho para o projeto de controle DLQR *online* via HDP. Uma das questões de interesse na análise dos resultados é verificar, a partir de uma condição de operação anterior estável, a adaptabilidade dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR perante variações paramétricas na planta.

Considere um sistema dinâmico descrito pela seguinte equação de estado

$$\begin{aligned} \dot{x}(t) &= A(p(t))x(t) + B(p(t))u(t) \\ y(t) &= C(p(t))x(t), \end{aligned} \tag{6.9}$$

sendo $A \in \Re^{n\times n}$, $x \in \Re^n$ o vetor de estado, $B \in \Re^{n\times n_e}$, $u \in \Re^{n_e}$ o vetor de entrada de controle e p um parâmetro variante no tempo e exôgeno (estritamente independente do estado x). Em geral, o parâmetro p pode ser desconhecido à priori, podendo ser medido ou estimado sobre a operação do sistema. Suposições típicas à priori sobre p são os limites sobre sua magnitude e taxa de variação.

O modelo de Eq.(6.9) foi inspirado nos trabalhos de Shamma, (Shamma, 2012), (Shamma, 1996), (Shamma, 1988), que propôs um quadro de modelos lineares variantes nos parâmetros para análise e projeto de sistemas de controle de ganho programado. Tais sistemas consistem em uma abordagem de projeto de controle que contrói um controlador não-linear para uma planta não-linear a partir de uma coleção de controladores lineares indexada por um parâmetro exôgeno. Estes controladores são combinados em tempo-real de acordo com medições *online* disponíveis.

No intuito de avaliar o comportamento dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR em relação à variações governadas pelo parâmetro $p(t)$ no modelo (6.9), propõe-se as seguintes trajetórias para $p(t)$

$$p_1(t) = \begin{cases} 0 & se \quad t_0 \leq t < t_v \\ \alpha\, e^{-\beta(t-t_v)} & se \quad t \geq t_v \end{cases} \tag{6.10}$$

e

$$p_2(t) = \begin{cases} 0 & se \quad t_0 \leq t < t_v \\ \alpha - \alpha\, e^{-\beta(t-t_v)} & se \quad t \geq t_v, \end{cases} \tag{6.11}$$

em que $\alpha > 0$ e $\beta > 0$ são constantes. Uma motivação para se considerar variações paramétricas regidas por exponenciais do tipo p_1 e p_2 vem do fato de que estas podem representar situações do mundo real onde os sistemas estão sujeitos à mudanças mais suaves no seu ponto de operação. Contudo, observa-se a seguir alguns casos limites de p_1 e p_2 que podem representar variações mais abruptas do ponto de operação do sistema, as quais incluem perturbações regidas por um degrau e um pulso de duração muito rápida.

Casos limites das funções p_1 e p_2

a) p_1 quando $\beta \to \infty$: pulso de duração k.
b) p_2 quando $\beta \to \infty$:

$$p_2^{\infty}(t) = \begin{cases} \alpha & se \quad t \geq t_v \\ 0 & caso\ contrário \end{cases} \tag{6.12}$$

c) p_1 quando $\beta \to 0^+$: a função p_1 se aproxima de p_2^{∞}.

d) p_2 quando $\beta \to 0^+$: não existe variação paramétrica na planta.

Nas Tabelas 6.19 e 6.20, apresentam-se as variações paramétricas do tipo p_1 e p_2, respectivamente, para diferentes valores de α e β. Estas variações foram aplicadas nos parâmetros da planta do sistema do circuito elétrico, que segue o modelo de Eq.(6.9), conforme apresentado no Apêndice F. O símbolo ‡ indica se ocorre a não-adaptabilidade dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR para cada um dos casos.

Tabela 6.19: Perturbação paramétrica p_1

Caso	α	β	μ	RLS$_\mu$-UDU^T HDP	RLS$_\mu$-HDP
1	0.25	0^+	0.92	–	–
2	1	0^+	0.92	–	‡
3	1	0.01	0.92	–	‡
4	1	0.1	0.92	–	‡
5	1	1	0.92	–	‡
6	1	10	0.92	–	‡
7	1	100	0.92	–	‡
8	10	0.001	0.92	–	‡
9	10	0.1	0.92	–	‡
10	10	1	0.92	‡	‡
11	10	10	0.92	–	‡
12	10	100	0.92	–	‡
13	10	10^{13}	0.92	–	‡
14	10	∞	0.92	–	‡

Na análise comparativa entre os comportamentos de adaptabilidade dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR para perturbações do tipo p_2^∞, ou equivalentemente do tipo $p_1^{0^+}$, observa-se que o processo de aprendizagem do estimador RLS$_\mu$-HDP-DLQR se torna mais demorado e com a presença de maiores picos em relação ao do estimador RLS$_\mu$-UDU^T-HDP-DLQR, como pode ser verificado nas Figuras 6.103-6.105 e 6.106-6.108 que ilustram a evolução dos parâmetros dos estimadores RLS$_\mu$-UDU^T-HDP-DLQR e RLS$_\mu$-HDP-DLQR, respectivamente, para um ciclo de 8000 iterações, com a constante α igual à 0.25. Em outros experimentos realizados, observou-se que os estimadores RLS$_\mu$-UDU^T-HDP-DLQR apresentam adaptabilidade para uma faixa de variação em torno de $(0, 1.5]$ da constante α, enquanto que os estimadores RLS$_\mu$-HDP-DLQR apresen-

Tabela 6.20: Perturbação paramétrica p_2

Caso	α	β	μ	RLS$_\mu$-UDU^T HDP	RLS$_\mu$-HDP
1	1	0.001	0.92	–	–
2	1	0.01	0.92	–	–
3	1	0.1	0.92	–	–
4	1	1	0.92	–	‡
5	1	10	0.92	–	‡
6	1	100	0.92	–	‡
7	10	0.01	0.92	–	‡
8	10	0.1	0.92	–	‡
9	10	1	0.92	‡	‡
10	10	10	0.92	‡	‡
11	0.25	10	0.92	–	–
12	0.25	10^{13}	0.92	–	–
13	0.25	∞	0.92	–	–
14	1.5	∞	0.92	–	‡

tam adaptabilidade para uma faixa de valores de α em $(0, 0.25]$.

Em relação às perturbações do tipo p_2, para maiores valores de β, verifica-se um comportamento similar àquele causado pela perturbação do tipo p_2^∞. As Figuras 6.86-6.91 e 6.95-6.100 representam os comportamentos dos estimadores RLS$_\mu$-UDU^T-HDP-DLQR e RLS$_\mu$-HDP-DLQR ao longo do processo de aprendizagem frente às perturbações do tipo p_2, para valores de β iguais à 10 e 10^{13}, respectivamente, com $\alpha = 0.25$. Observa-se uma similaridade entre os comportamentos de cada um dos estimadores para os casos $\alpha = 0.25$, $\beta = 10^{13}$ e $\alpha = 0.25$, $\beta = \infty$.

Com respeito às perturbações do tipo p_1^∞, verificou-se que os estimadores RLS$_\mu$-UDU^T-HDP-DLQR são capazes de se adaptarem para uma faixa mais ampla de valores de α, $\alpha \in (0, 10]$, enquanto que os estimadores RLS$_\mu$-HDP-DLQR não apresentam adaptabilidade. De modo geral, os estimadores RLS$_\mu$-HDP-DLQR não respondem de forma satisfatória em relação às variações do tipo p_1, quando em seu processo de aprendizagem os estimadores não conseguem ajustar os seus parâmetros de modo que o sistema se adapte para uma nova condição de operação estável (por exemplo, caso $p_1^{0^+}$), ou retorne à condição de operação anterior estável (por exemplo, caso p_1^∞).

De forma similar às perturbações do tipo p_2, verifica-se que em relação às

perturbações do tipo p_1 com valores de β maiores ou iguais à 10, os estimadores RLS$_\mu$-UDU^T-HDP-DLQR apresentam comportamentos semelhantes entre si, apresentando um comportamento que tende àquele causado pela variação do tipo p_1^∞ à medida que β cresce. As Figuras 6.54-6.56 e 6.58-6.60 representam o comportamento dos estimadores RLS$_\mu$-UDU^T-HDP-DLQR ao longo do processo de aprendizagem frente às perturbações do tipo p_1, para valores de β iguais à 10 e 10^{13}, respectivamente, com $\alpha = 10$.

Percebe-se que os tipos de perturbações p_1 e p_2 que exercem maior influência sobre o comportamento dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR correspondem aos casos em que $\beta = 0.001$, como pode ser observado nas Figuras 6.50-6.52 e 6.62-6.67, pois o processo de convergência é mais demorado com relação a convergência dos casos destacados anteriormente.

Em geral, os estimadores RLS$_\mu$-UDU^T-HDP-DLQR retornaram melhores resultados quando comparados com os estimadores RLS$_\mu$-HDP-DLQR, apresentando um processo de adaptabilidade mais rápido em relação às variações do tipo p_1 e p_2, ao passo que os estimadores RLS$_\mu$-HDP-DLQR mostraram-se menos adaptáveis às variações na planta na maioria dos casos considerados nestes experimentos, especialmente para variações do tipo p_1 em que não ocorreu a adaptabilidade para nenhuma situação, exceto para o caso $p_1^{0^+}$, com $\alpha \in (0, 0.25]$. Entretanto, um caso interessante é ilustrado pelas Figuras 6.70-6.75 e 6.78-6.83 para os casos de perturbação p_2, com $\alpha = 1$, $\beta = 0.01$ e $\alpha = 1$, $\beta = 0.1$, respectivamente, em que os estimadores RLS$_\mu$-HDP-DLQR conseguiram se adaptar mais rapidamente às variações na planta.

O Apêndice I complementa os resultados desta seção relacionados às perturbações paramétricas do tipo p_1 e p_2 para outros valores das constantes α e β, conforme apresentados nas Tabelas 6.19 e 6.20. Com o propósito de avaliar o comportamento dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR do ponto de vista de sistemas de controle, no Apêndice H são apresentados a trajetória dos estados x_k e o sinal de controle u_k que foi aplicado ao modelo do circuito elétrico de quarta ordem durante a sintonia da HDP, como também são ilustrados os comportamentos dos traços da matriz de malha fechada e os autovalores associados.

Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.001$

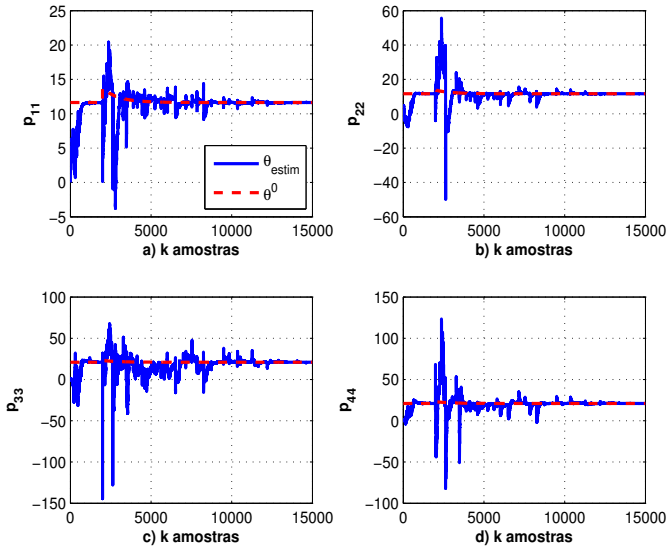

Figura 6.50: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

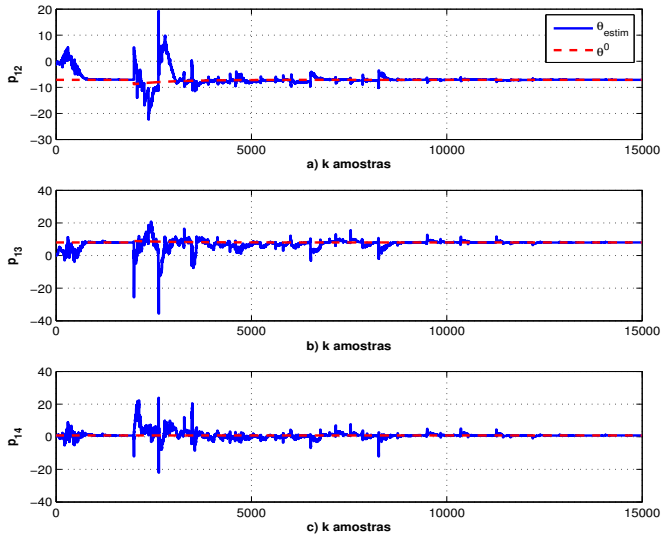

Figura 6.51: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

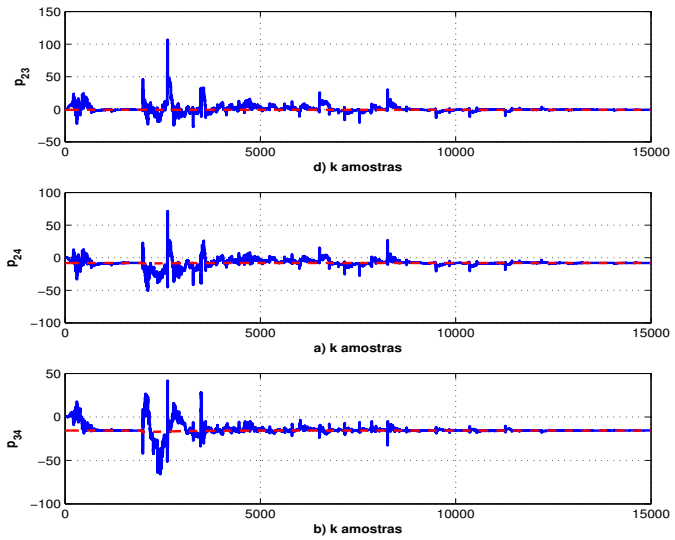

Figura 6.52: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

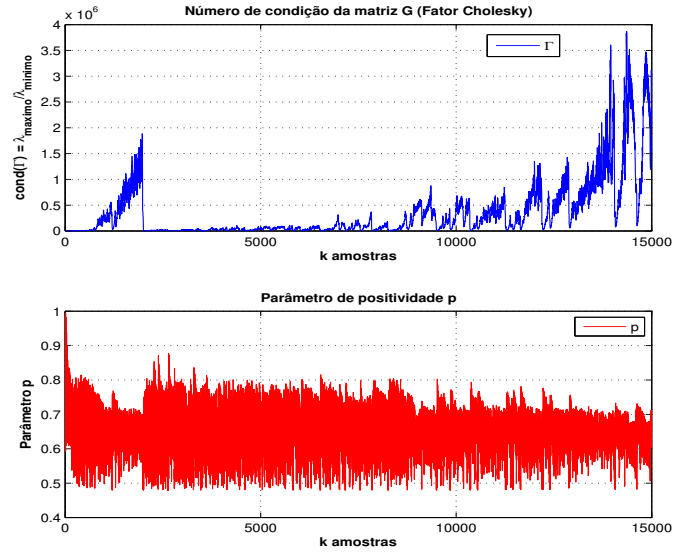

Figura 6.53: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.001$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$

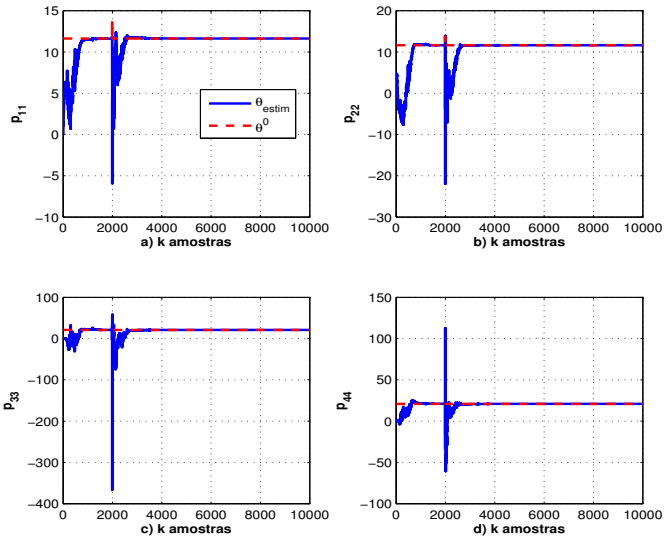

Figura 6.54: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

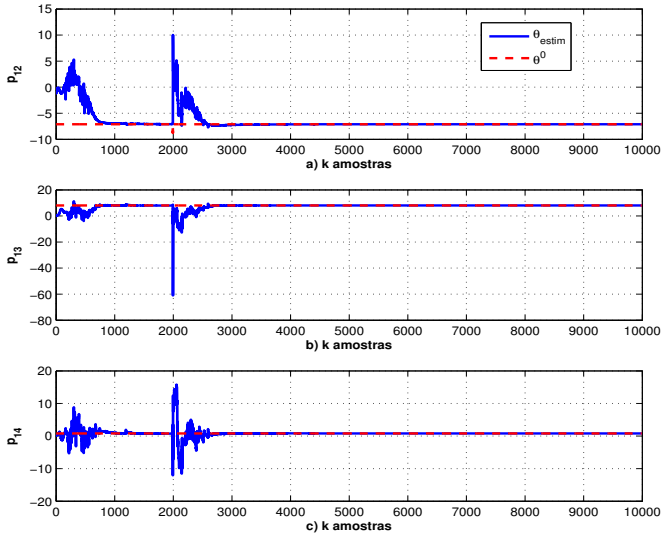

Figura 6.55: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

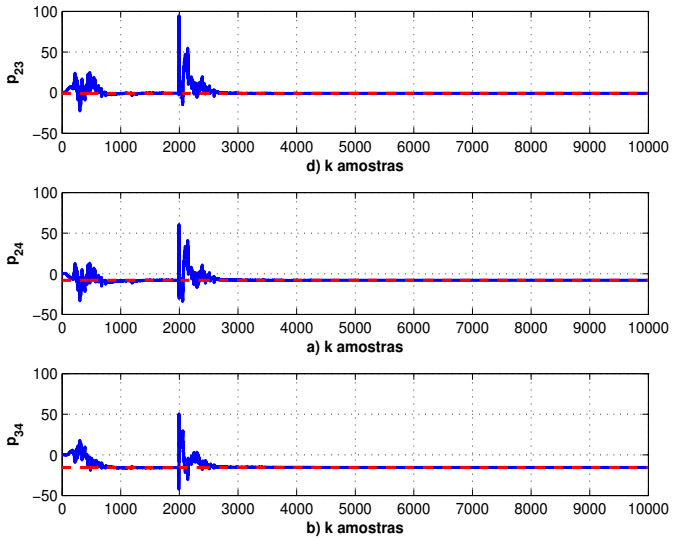

Figura 6.56: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

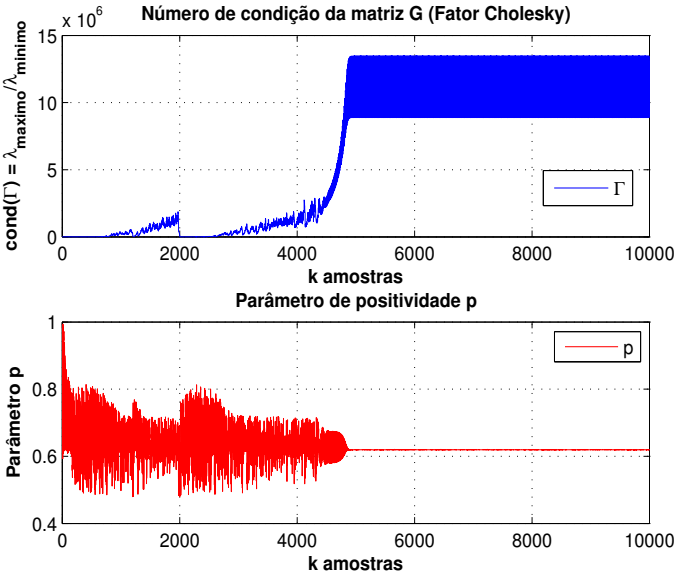

Figura 6.57: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10^{13}$

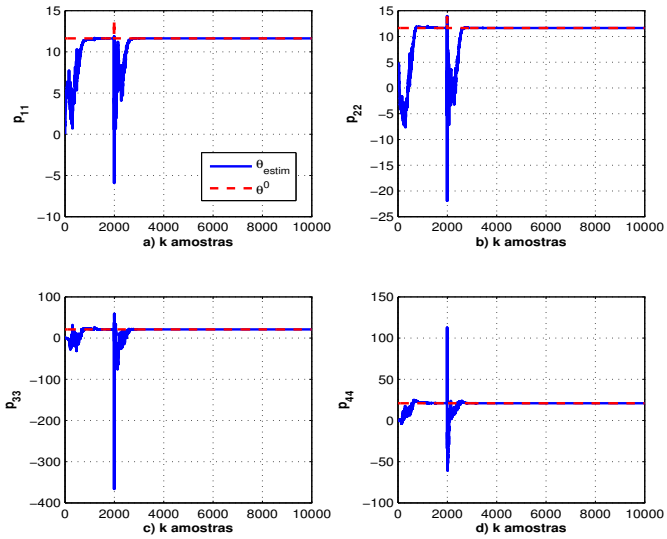

Figura 6.58: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

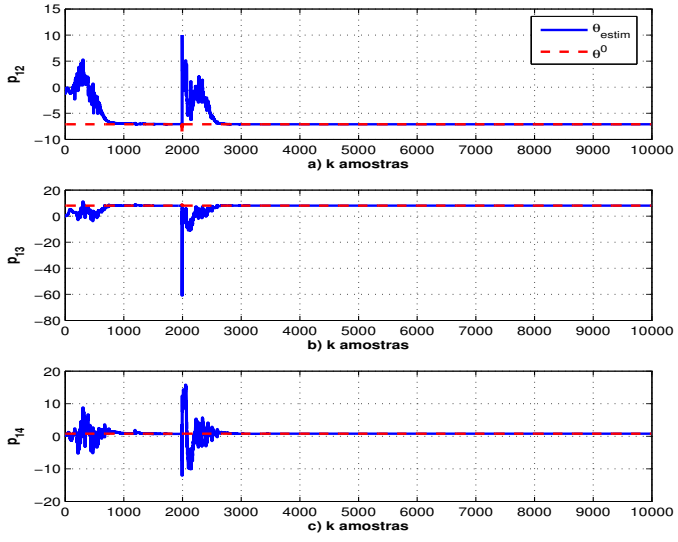

Figura 6.59: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

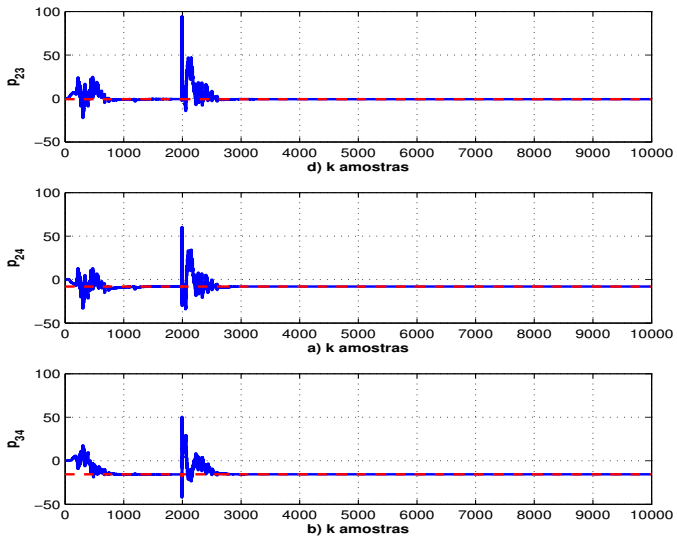

Figura 6.60: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

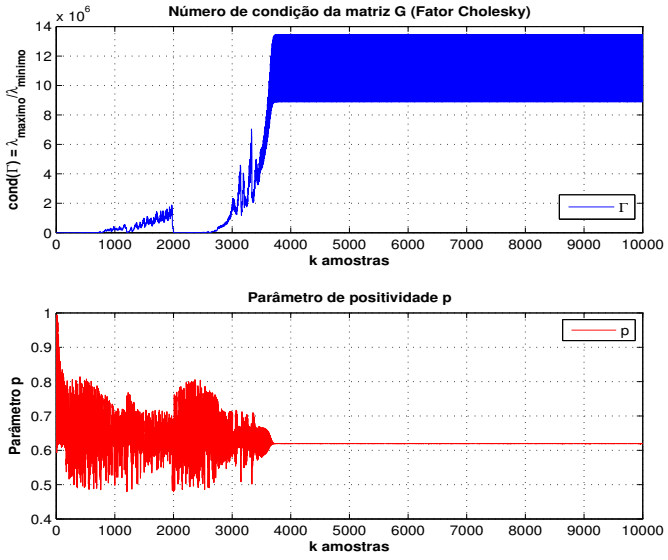

Figura 6.61: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10^{13}$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$

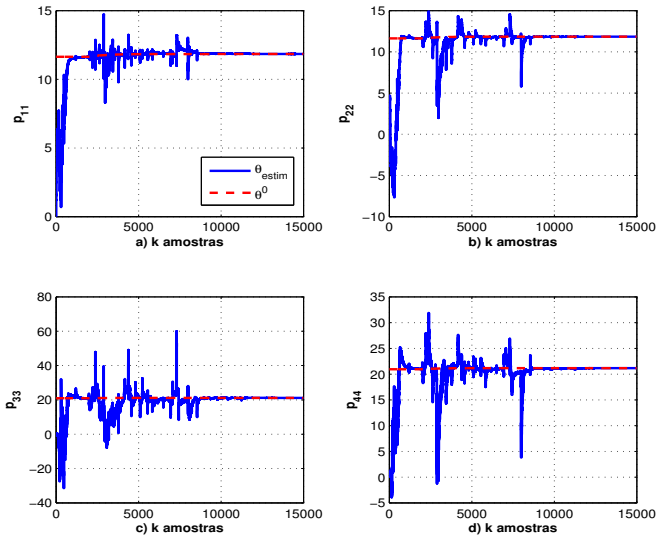

Figura 6.62: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

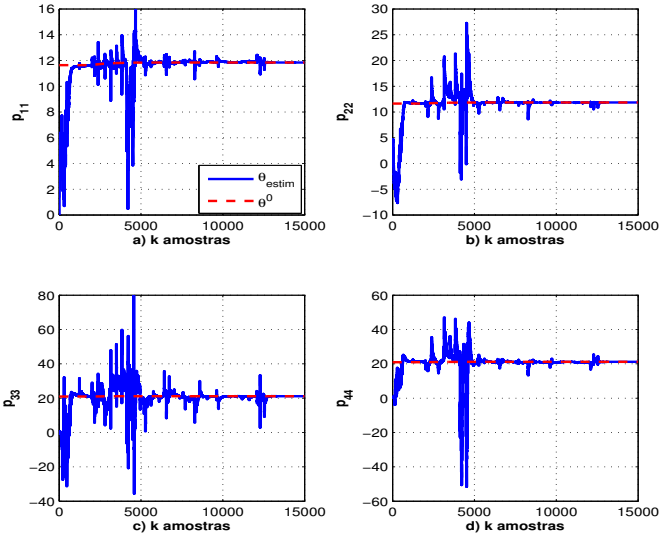

Figura 6.63: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

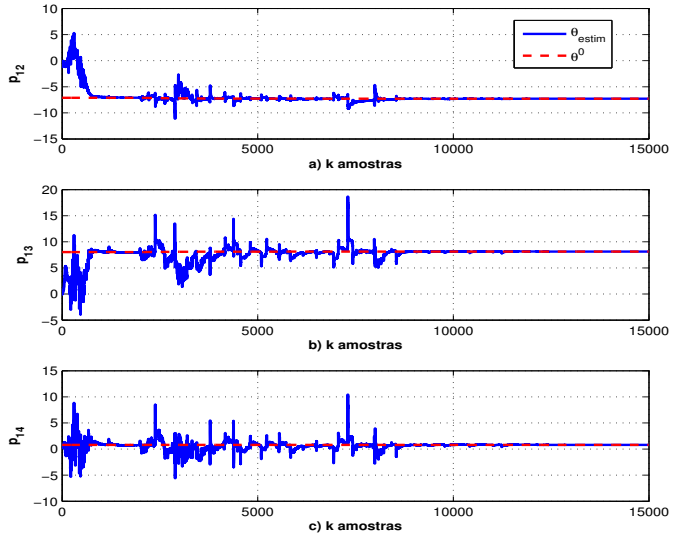

Figura 6.64: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

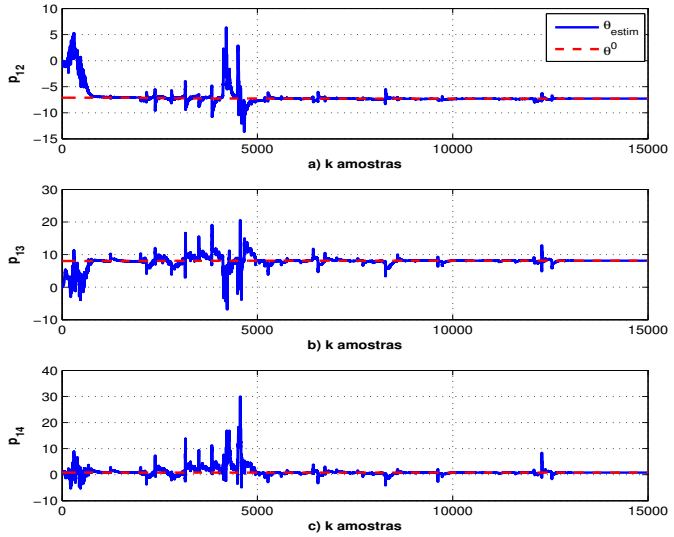

Figura 6.65: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

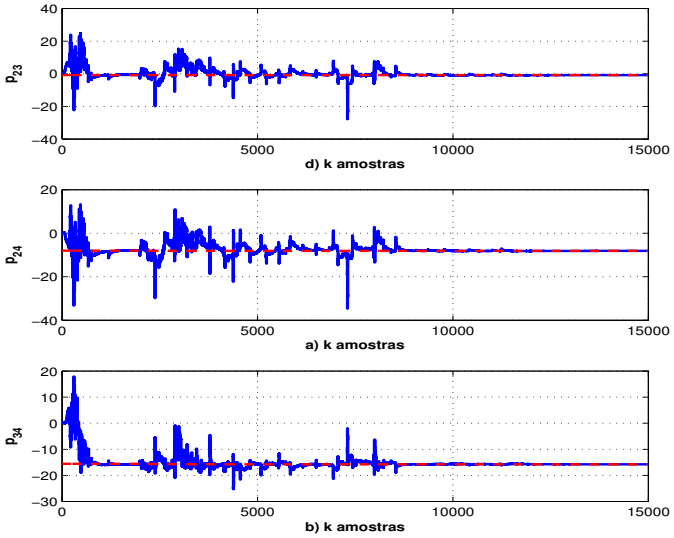

Figura 6.66: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

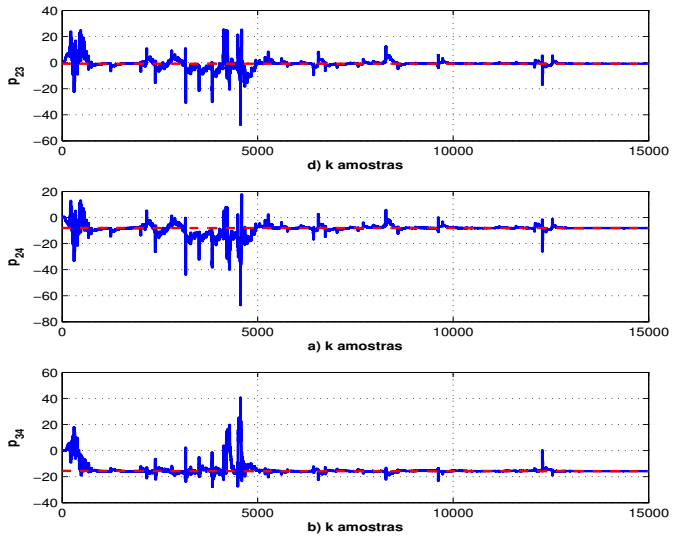

Figura 6.67: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

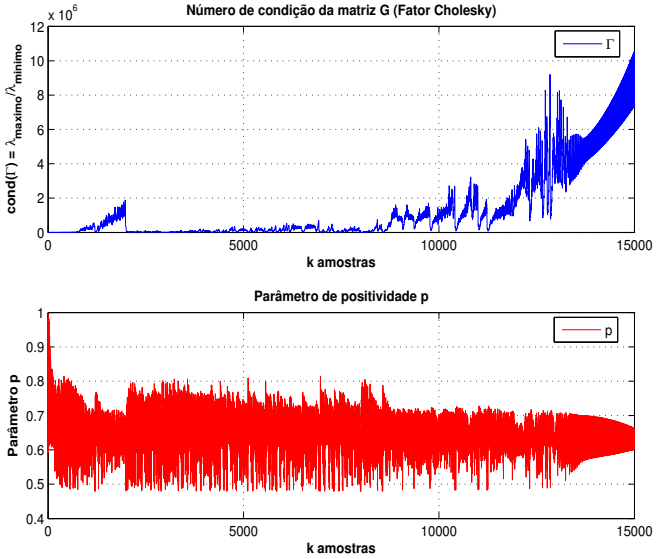

Figura 6.68: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

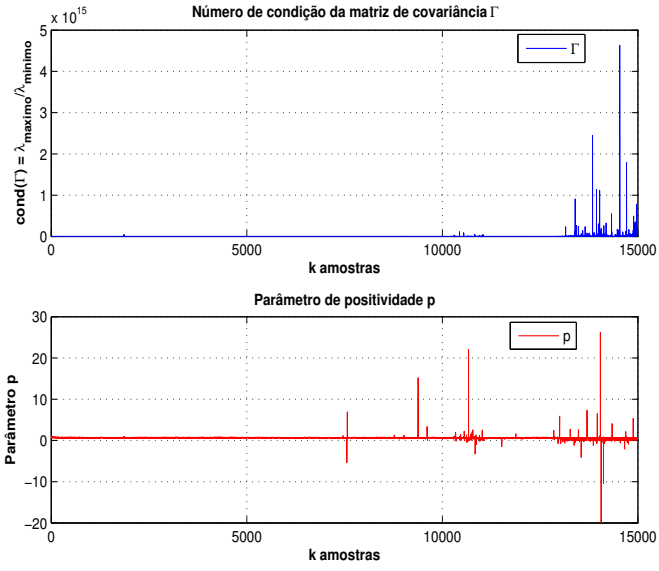

Figura 6.69: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$

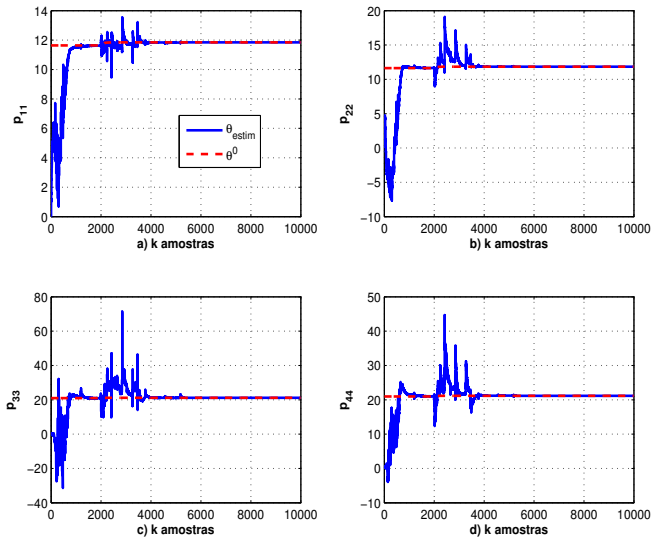

Figura 6.70: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

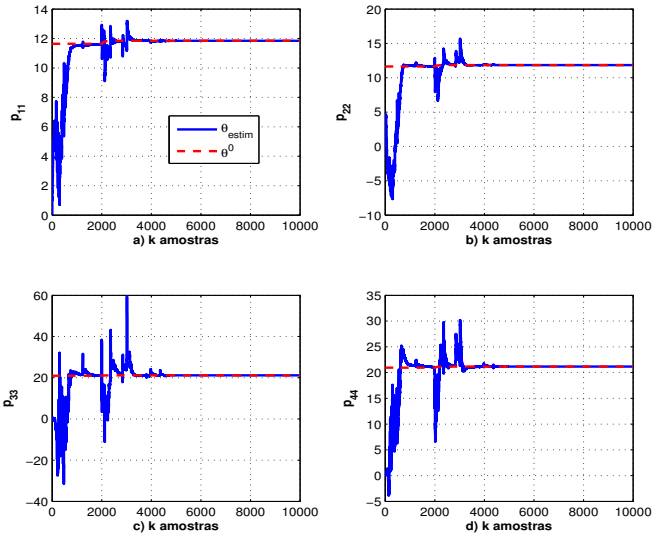

Figura 6.71: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

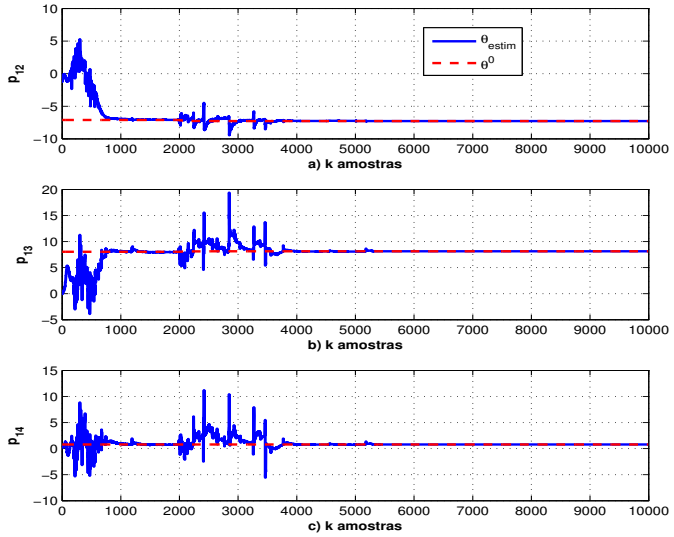

Figura 6.72: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

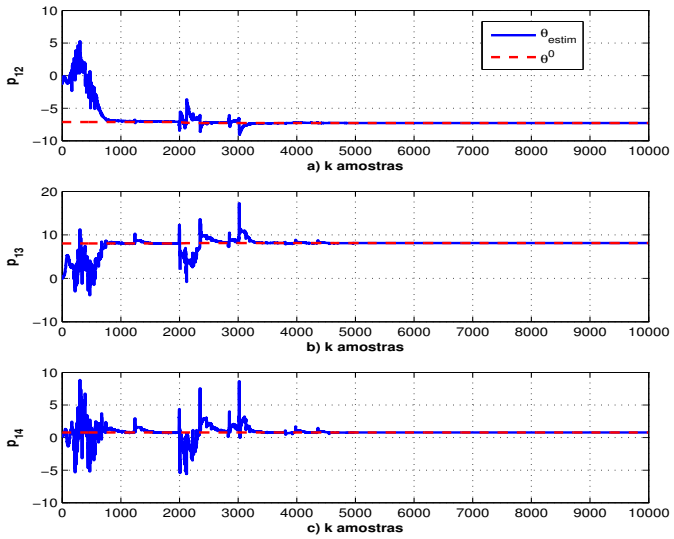

Figura 6.73: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

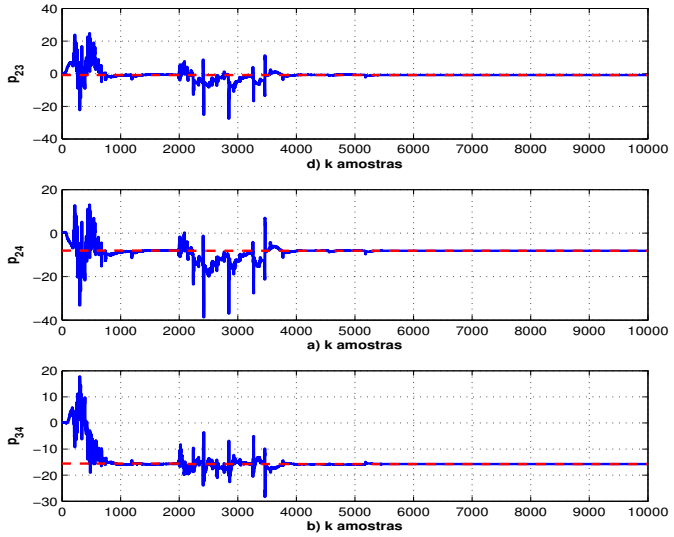

Figura 6.74: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

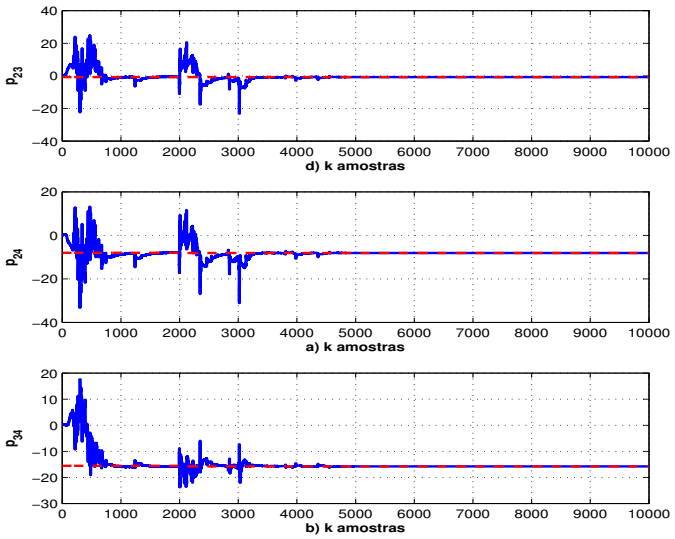

Figura 6.75: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

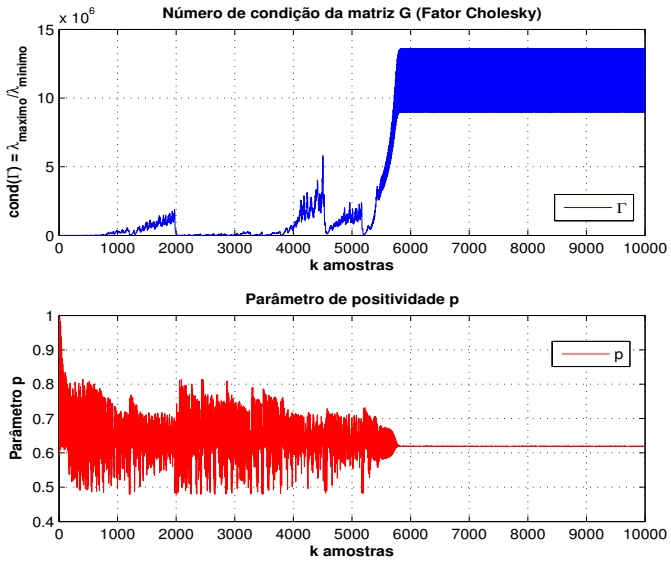

Figura 6.76: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

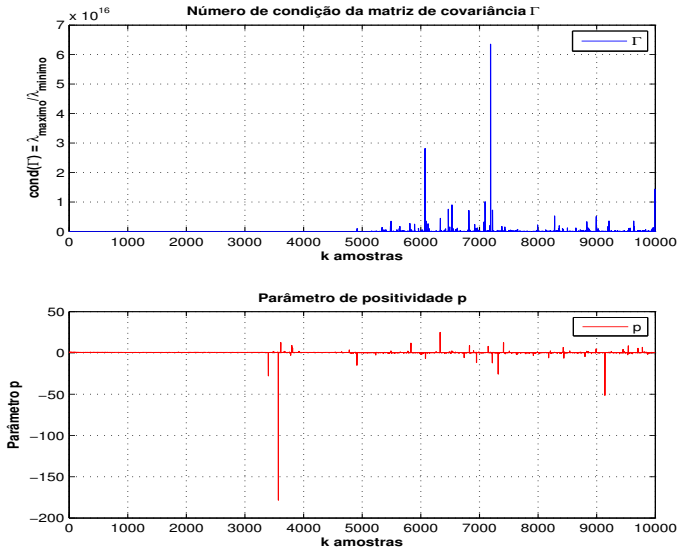

Figura 6.77: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.01$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$

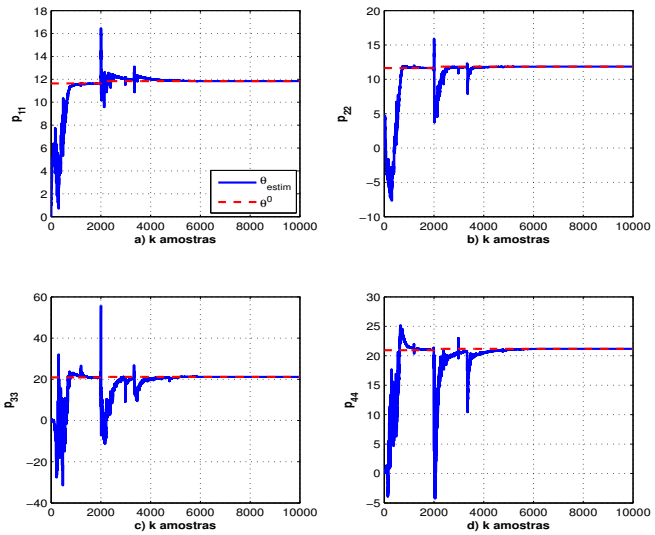

Figura 6.78: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

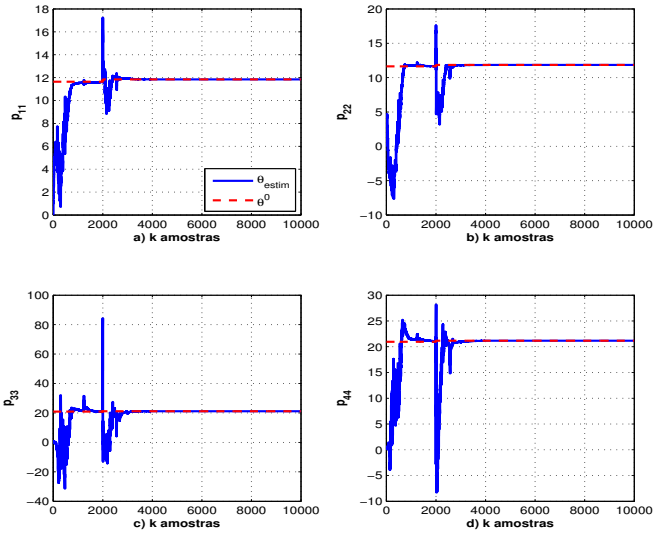

Figura 6.79: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

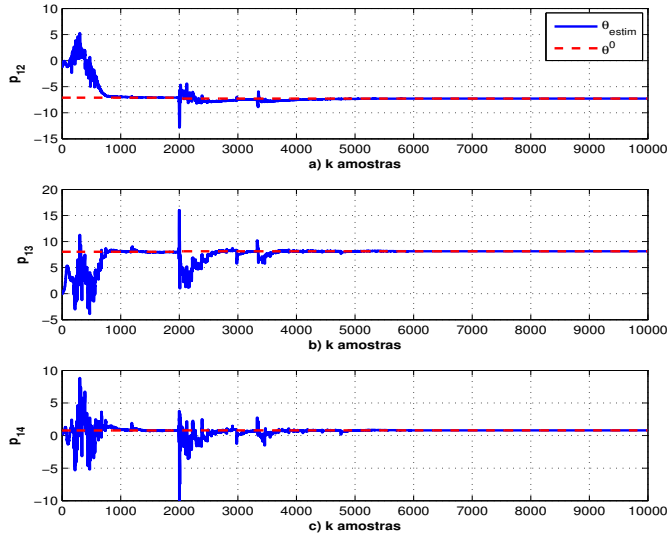

Figura 6.80: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

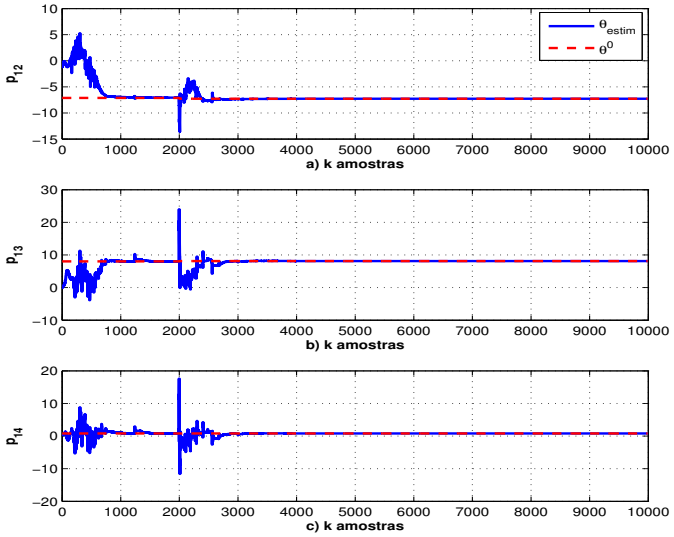

Figura 6.81: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

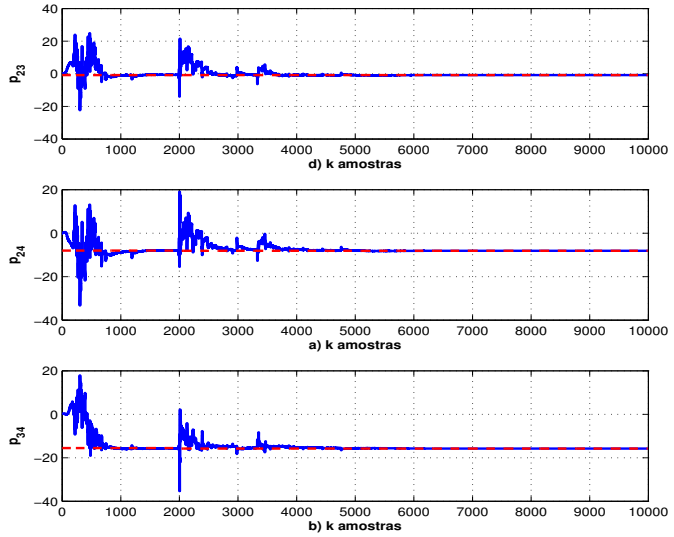

Figura 6.82: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

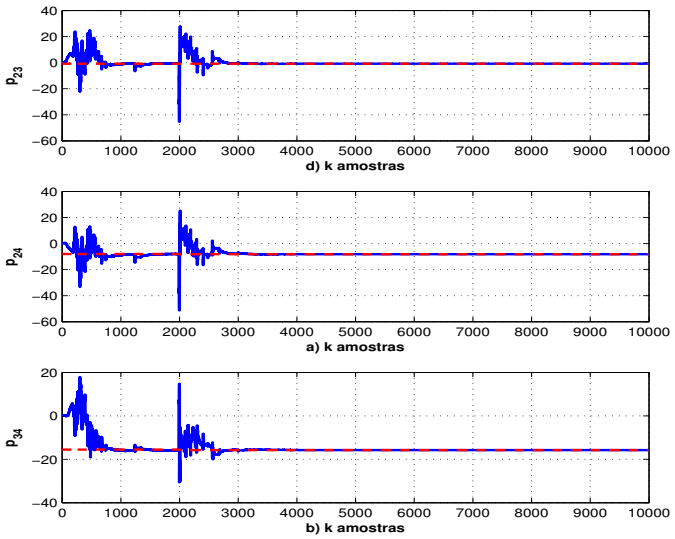

Figura 6.83: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

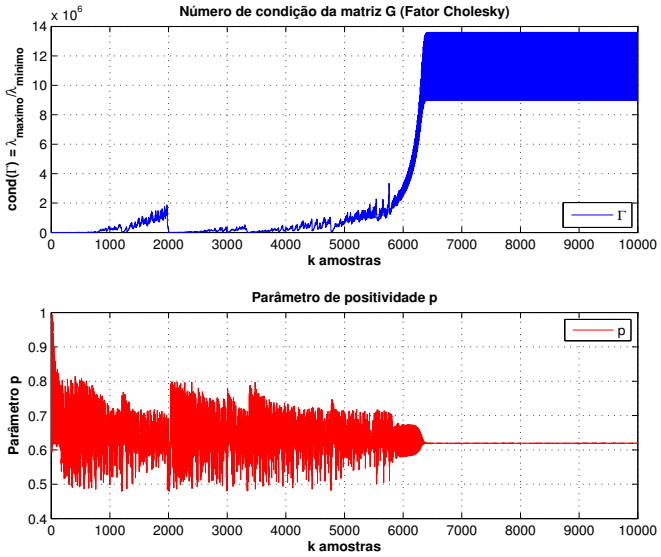

Figura 6.84: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

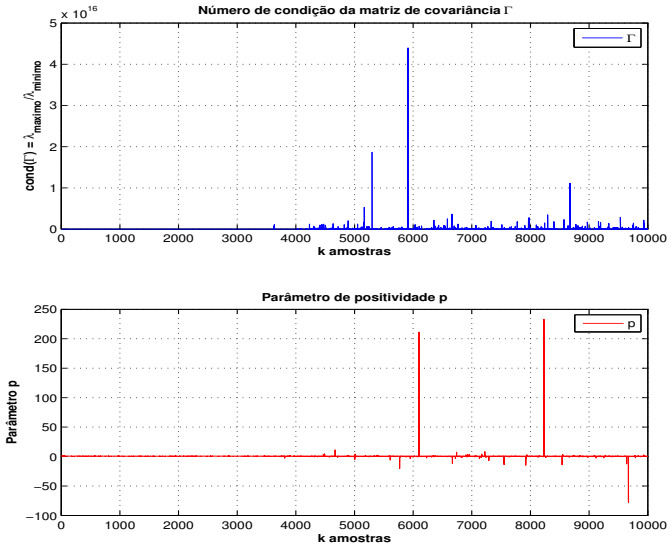

Figura 6.85: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.1$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$

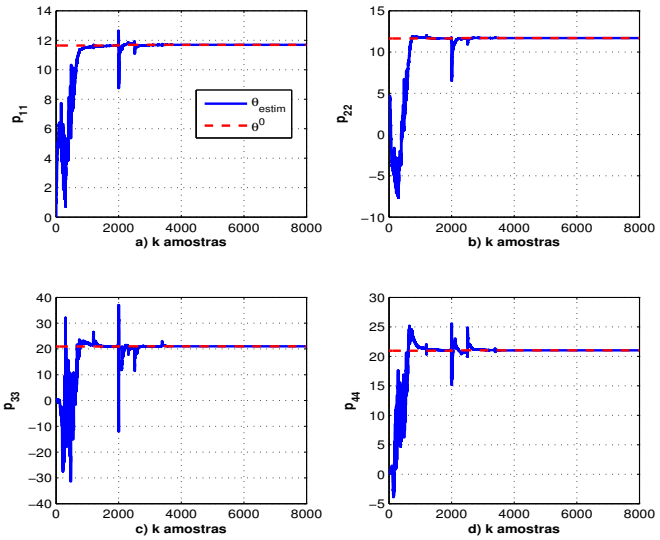

Figura 6.86: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

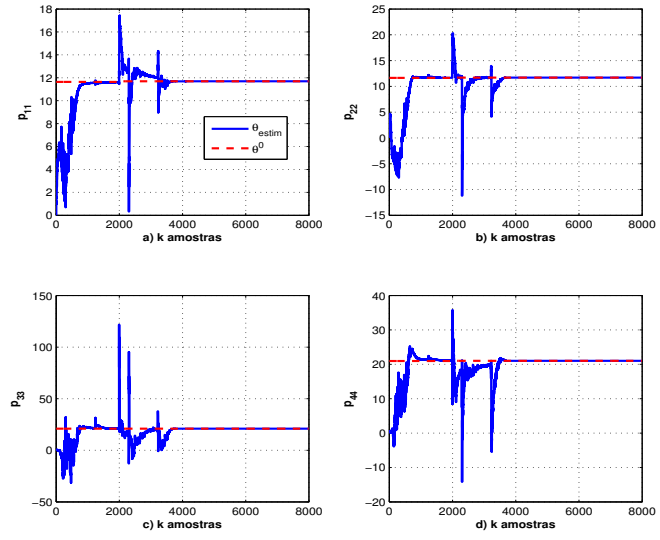

Figura 6.87: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

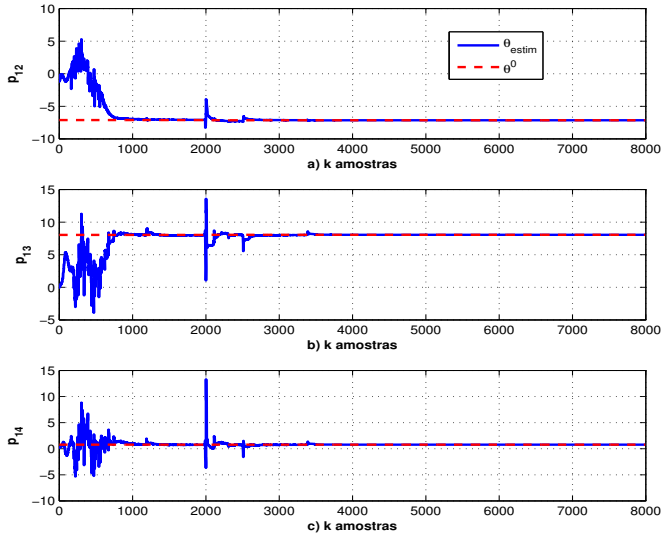

Figura 6.88: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

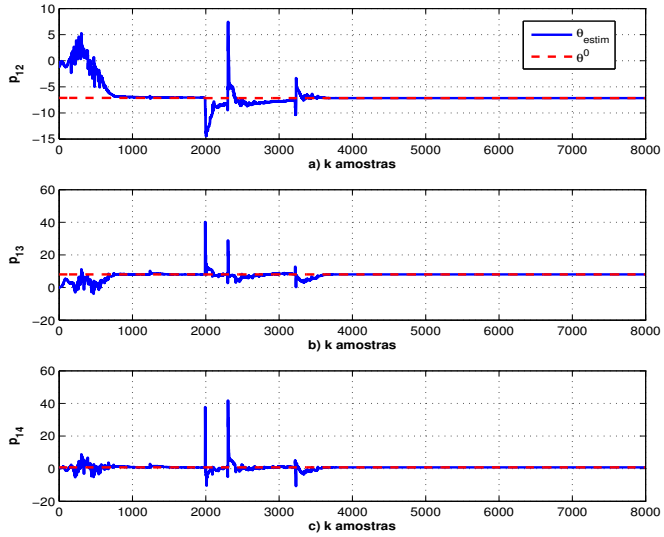

Figura 6.89: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

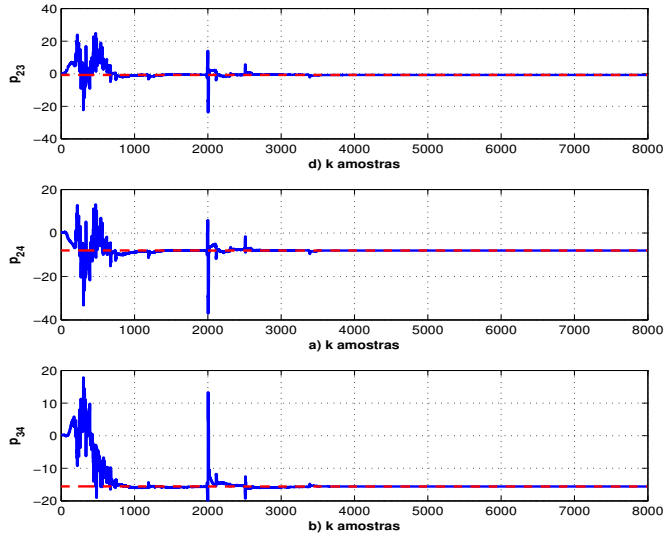

Figura 6.90: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

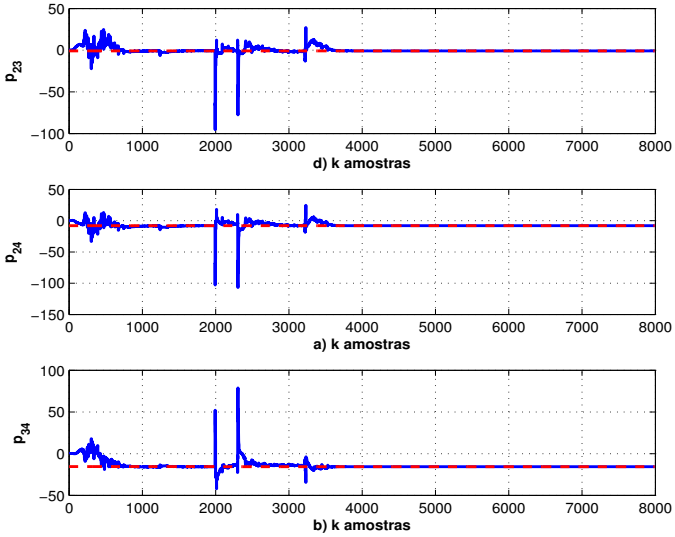

Figura 6.91: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

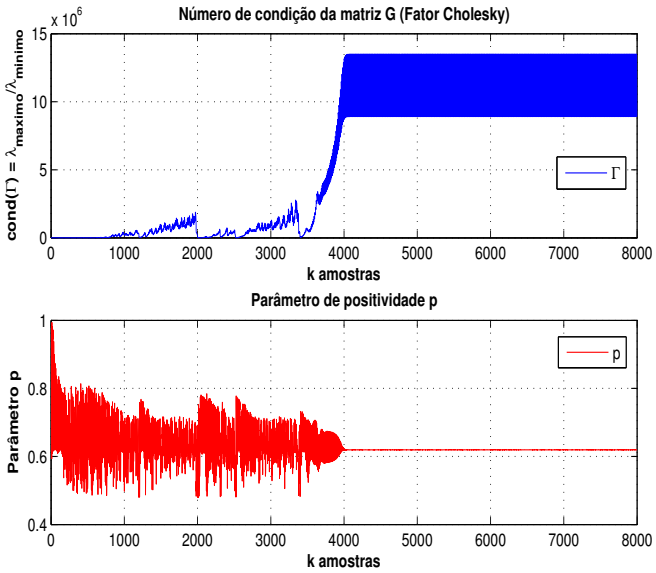

Figura 6.92: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

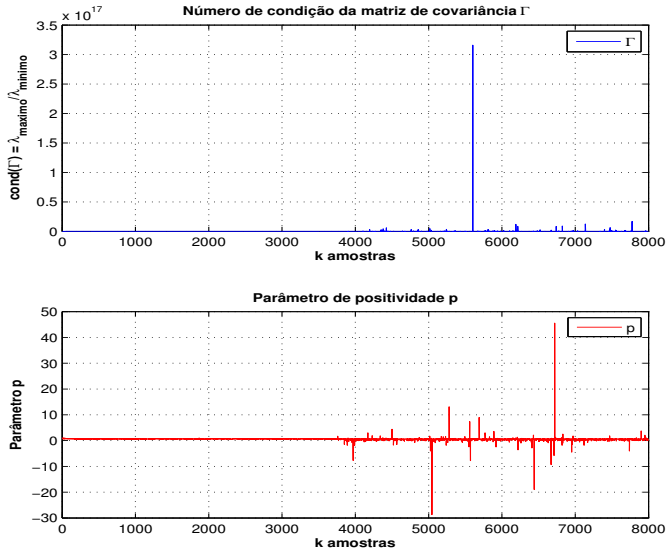

Figura 6.93: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

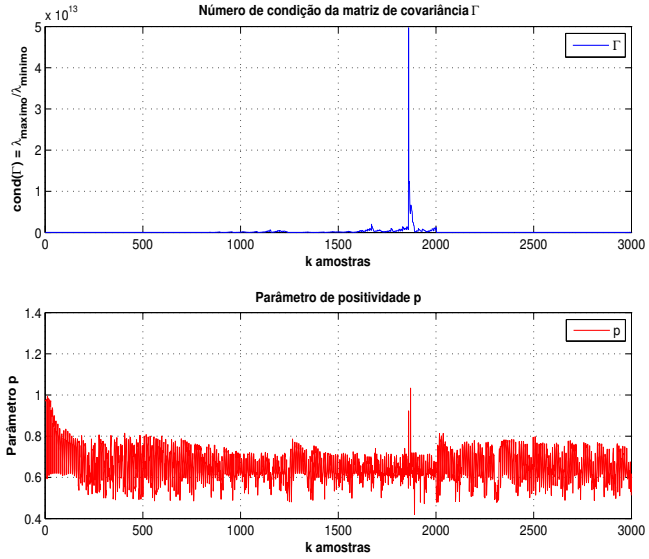

Figura 6.94: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 3000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$

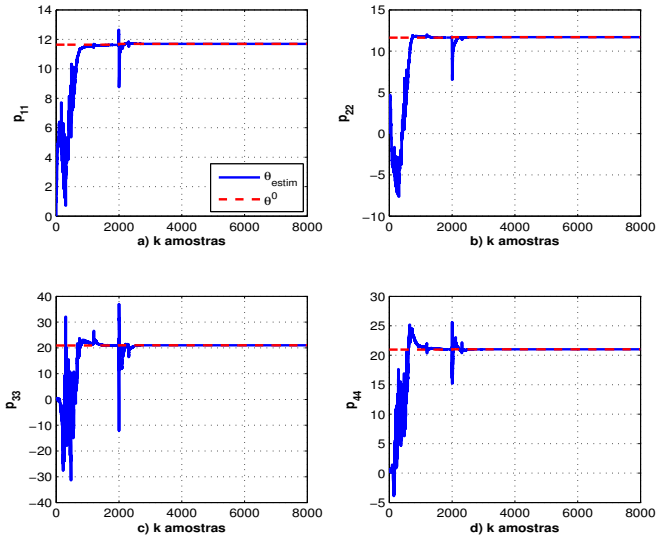

Figura 6.95: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

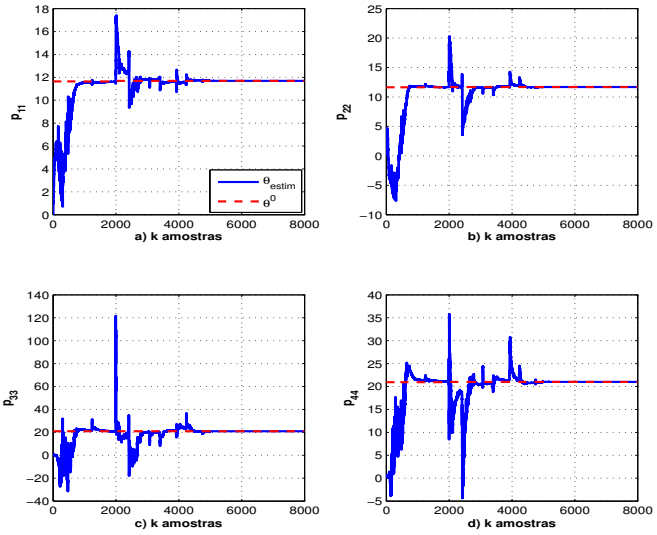

Figura 6.96: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

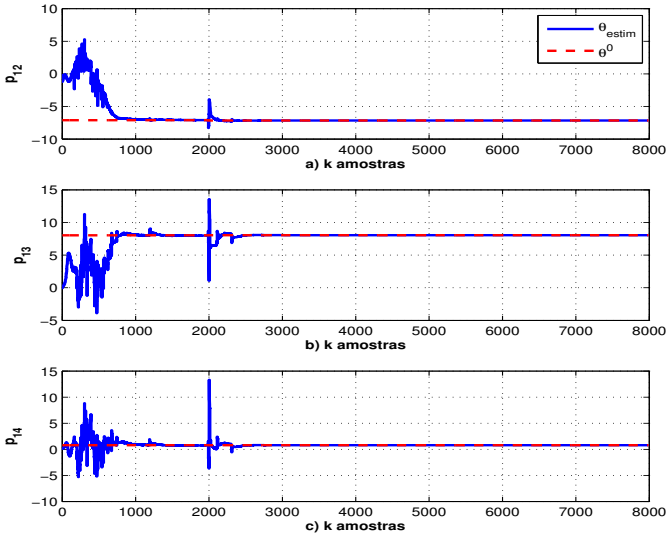

Figura 6.97: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

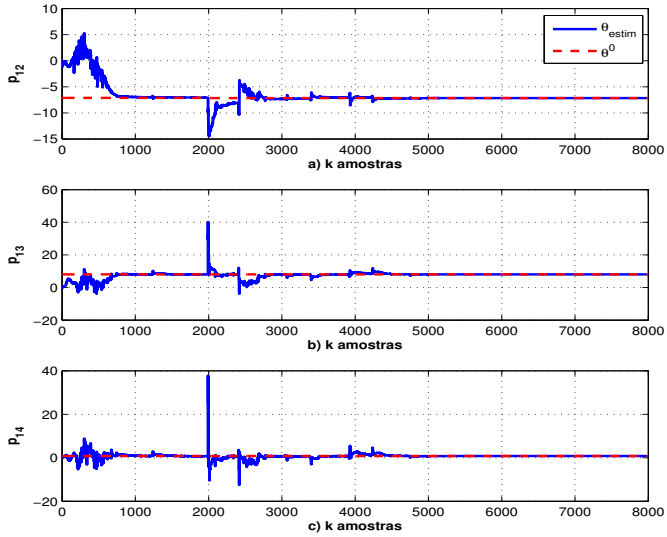

Figura 6.98: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

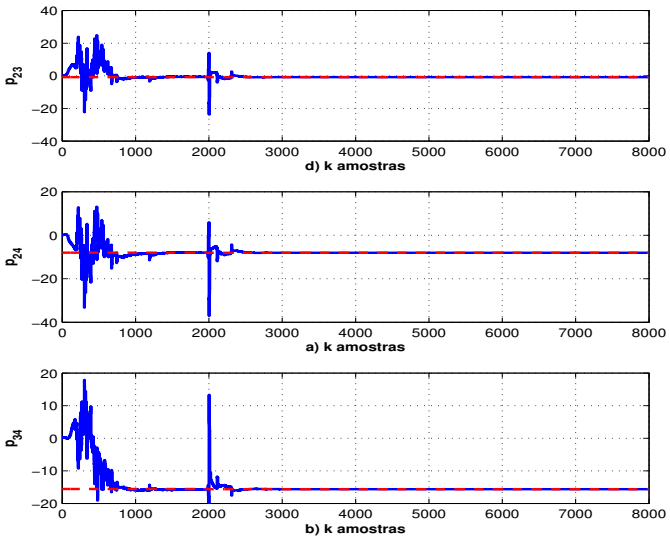

Figura 6.99: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

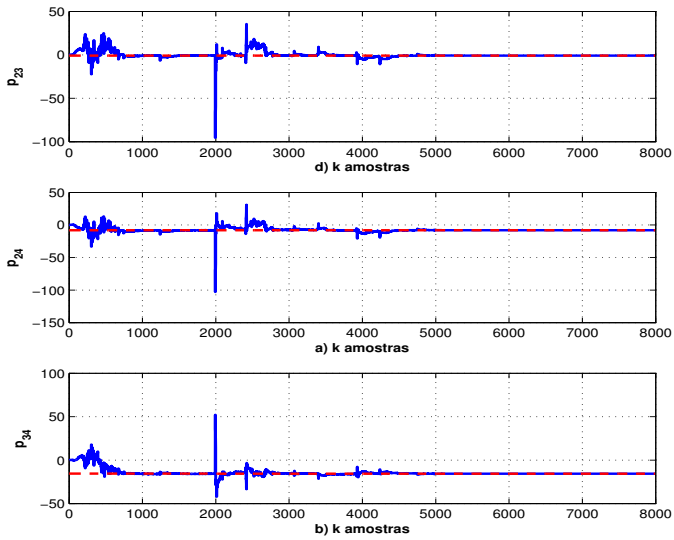

Figura 6.100: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

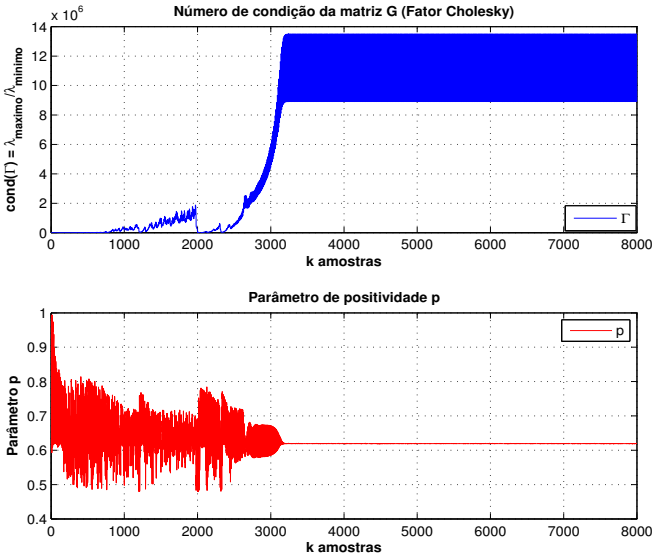

Figura 6.101: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

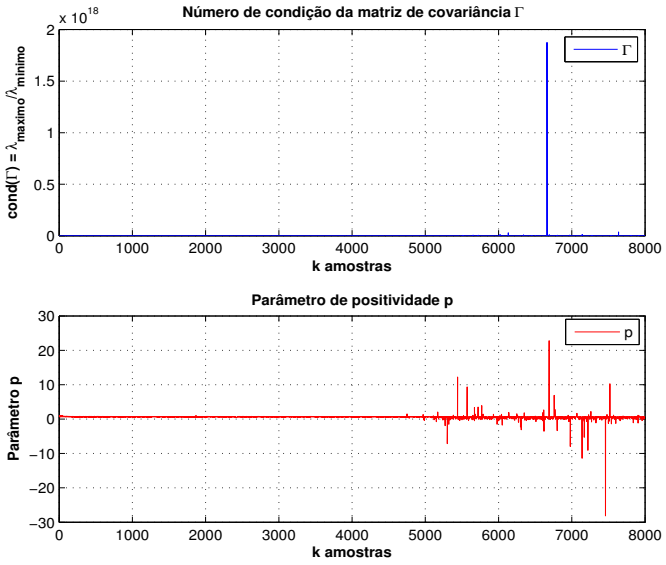

Figura 6.102: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$

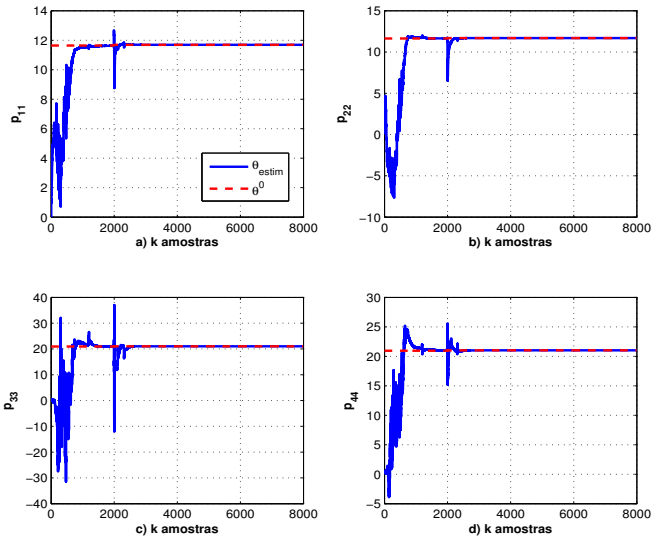

Figura 6.103: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

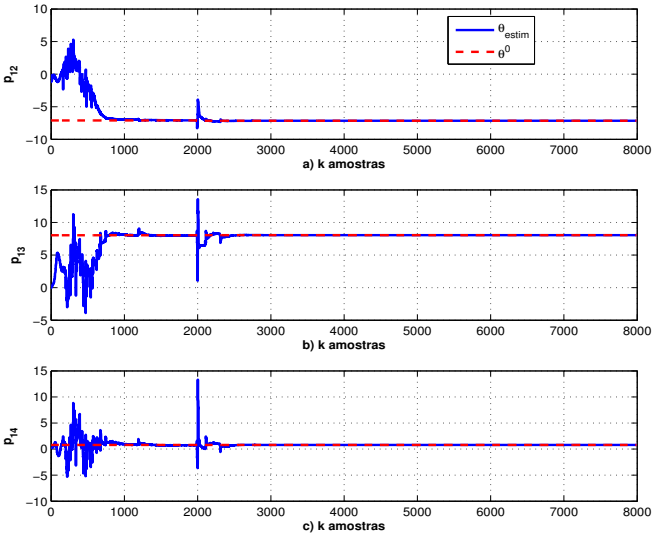

Figura 6.104: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

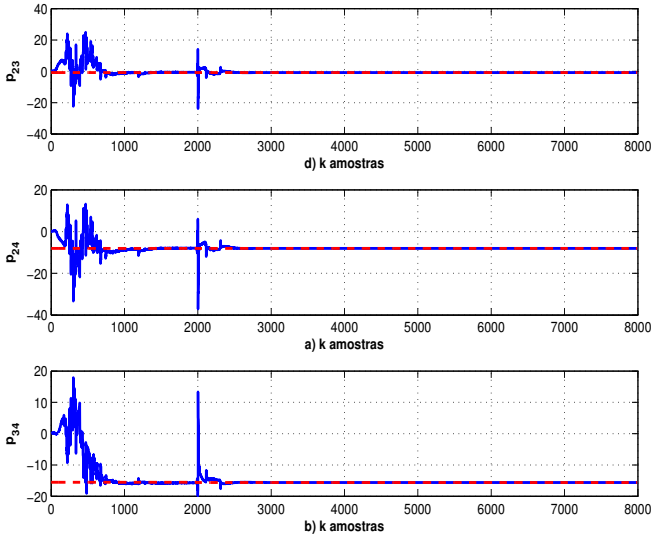

Figura 6.105: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

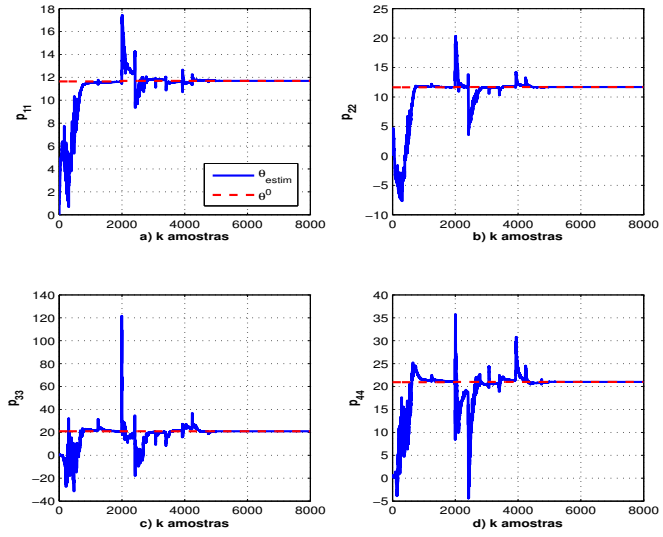

Figura 6.106: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

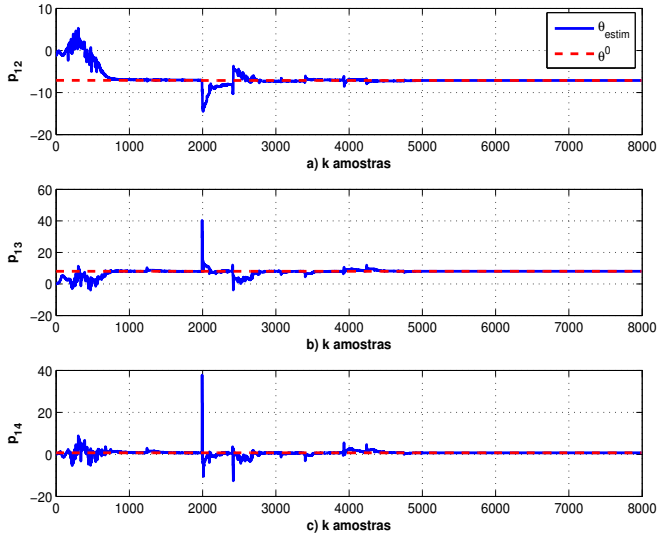

Figura 6.107: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

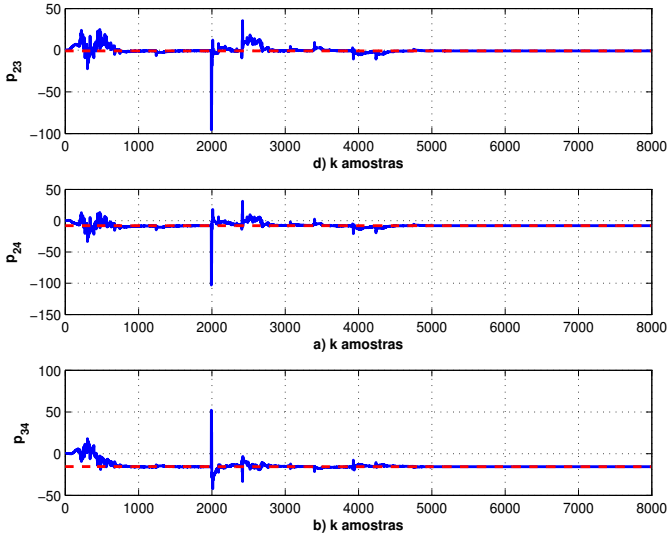

Figura 6.108: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

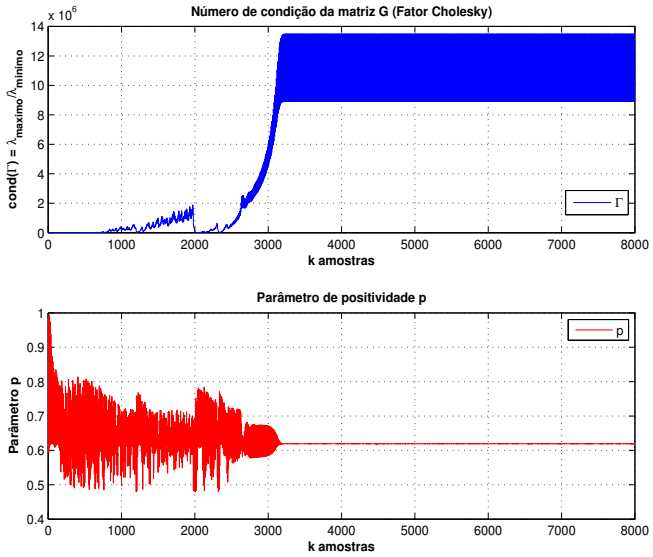

Figura 6.109: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

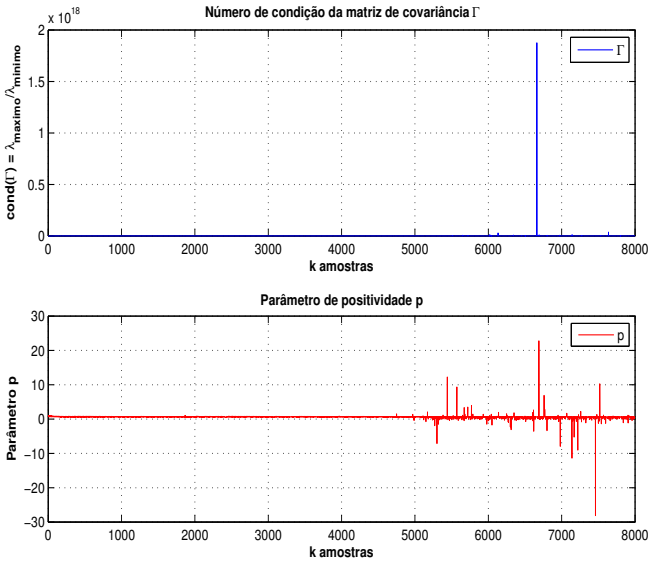

Figura 6.110: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = \infty$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

6.3 Experimentos RLS-TD(λ)-DLQR

Na aplicação de esquemas RLS-TD(λ) para solução do problema DLQR, um aspecto importante a ser investigado é a influência do parâmetro λ no processo de aproximações da solução da equação HJB-*Riccati*. Por exemplo, observou-se que alguns valores de λ, $0 < \lambda < 1$, proporcionaram estimativas com variâncias menores para solução da equação HJB-*Riccati* em relação ao valor limite $\lambda = 0$. Os resultados computacionais são apresentados para um sistema multivariável de terceira ordem que representa uma aeronave F-16 dada na referência (Lewis e Vrabie, 2009*a*). As matrizes do sistema dinâmico são mostradas no Apêndice G.

Nas Figuras 6.111-6.114 mostra-se a comparação de desempenho de algoritmos RLS-TD(λ) para diferentes valores de λ para um ciclo de 5000 iterações. Em todos os casos, o fator de esquecimento μ é estabelecido igual à 0.901 e a matriz de covariância inicial é dada por $\Gamma_0 = 10^5 I$, em que $I_{6\times 6}$ é a matriz identidade. O desempenho dos algoritmos é avaliado sob um ponto de vista estatístico por meio de momentos de primeira e segunda ordem[1] das estimativas do vetor de parâmetros θ associado com a matriz P da equação HJB-*Riccati*.

Observa-se para cada uma das componentes de θ que as estimativas fornecidas pelos algoritmos RLS-TD(0.402), RLS-TD(0.04) e RLS-TD(0.505) apresentam, em geral, menores variâncias quando comparadas com aquelas fornecidas pelo estimador RLS-TD(0), conforme ilustrado nas Figuras 6.113 e 6.114. Pecebe-se também que dentre os estimadores RLS-TD(0.04), RLS-TD(0.402) e RLS-TD(0.505), as menores variâncias são acentuadas para RLS-TD(0.04) e RLS-TD(0.402). Observando-se o comportamento das médias, Figuras 6.111 e 6.112, verifica-se que os estimadores RLS-TD(λ) tendem à um comportamento não-polarizado para todos os valores de λ considerados, entretanto observa-se que os resultados obtidos para os casos $\lambda = 0.04$ e $\lambda = 0.402$ se destacam devido as médias se aproximarem mais rapidamente para o valor verdadeiro.

[1]Momento de primeira ordem é definido pela média temporal das estimativas de θ, ou seja, a média ao longo do tempo de suas estimativas instantâneas. Quando o tempo t tende para o infinito, tem-se $\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} \widehat{\theta}(t)dt$

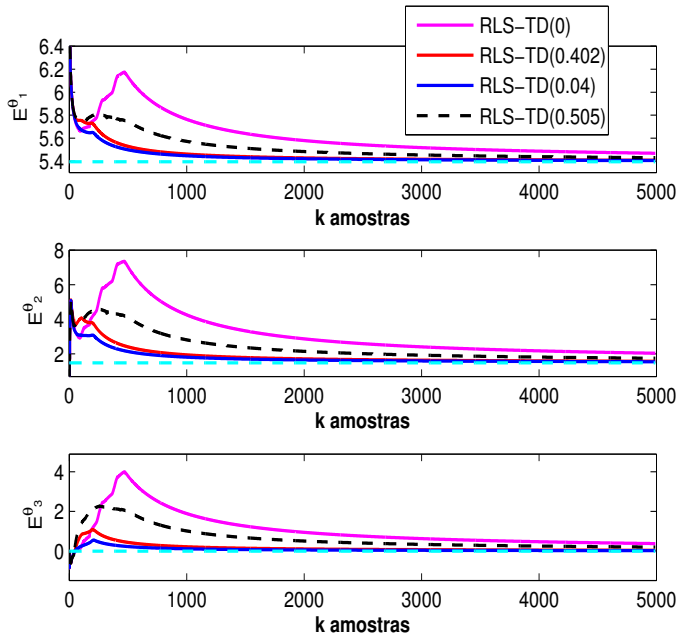

Figura 6.111: Média das estimativas para os parâmetros θ_1, θ_2 e θ_3.

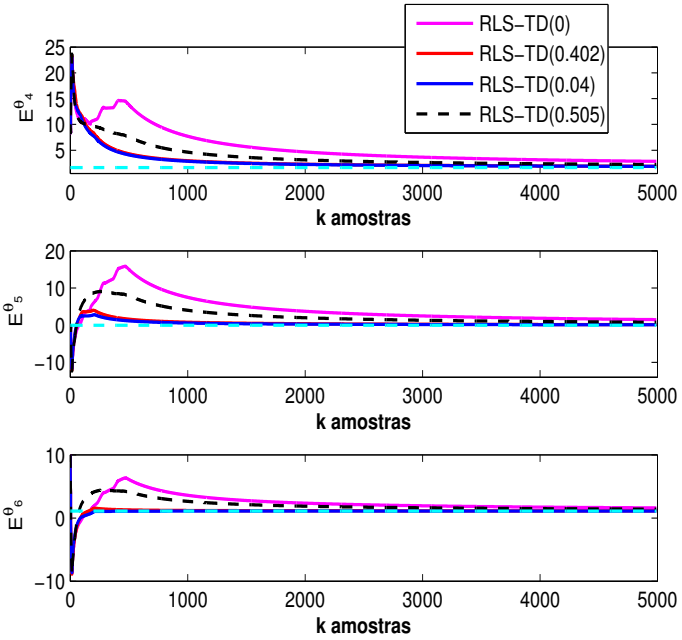

Figura 6.112: Média das estimativas para os parâmetros θ_4, θ_5 e θ_6.

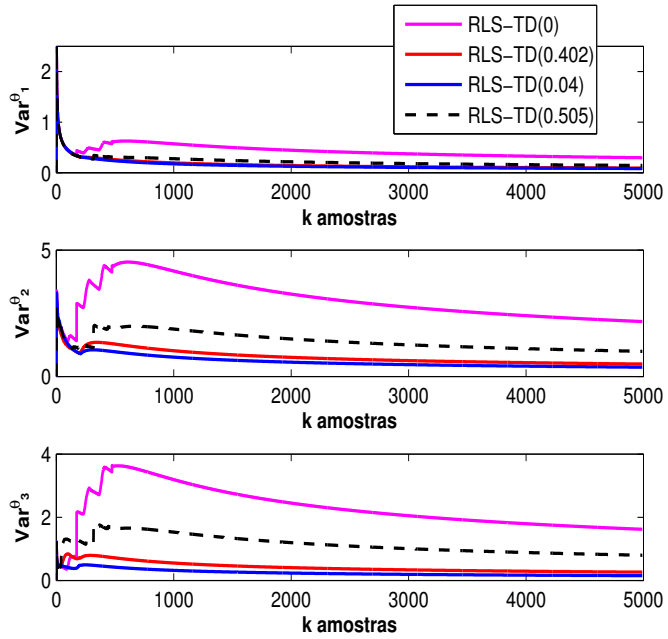

Figura 6.113: Comparação de desempenho de RLS-TD(0), RLS-TD(0.402), RLS-TD(0.04) e RLS-TD(0.505) para os parâmetros θ_1, θ_2 e θ_3.

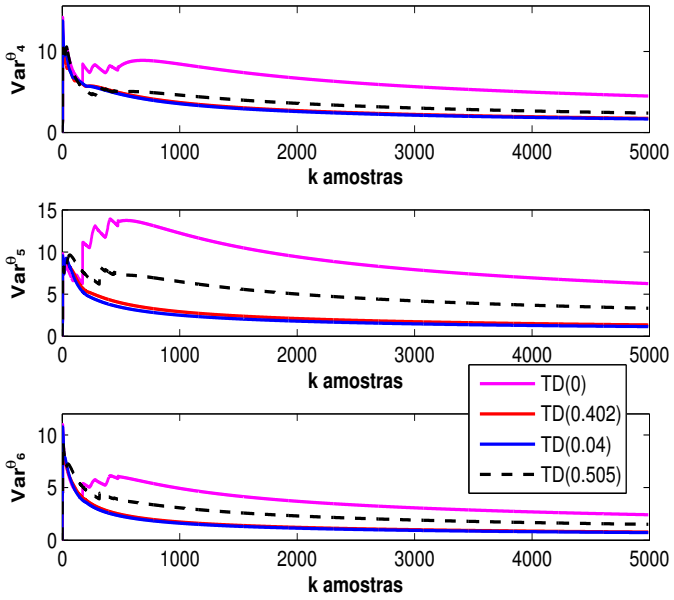

Figura 6.114: Comparação de desempenho de RLS-TD(0), RLS-TD(0.402), RLS-TD(0.04) e RLS-TD(0.505) para os parâmetros θ_4, θ_5 e θ_6.

6.4 Conclusão

Neste capítulo, os resultados de uma metodologia a qual investiga a convergência e a estabilidade numérica dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR desenvolvidos com iteração de política otimística para solução *online* do DLQR foram enfatizados. Apresentou-se também os resultados de uma metodologia baseada na fusão de métodos TD(λ) e RLS para estabelecer políticas ótimas via iteração de política otimística. O método proposto para análise de convergência da solução da equação HJB-*Riccati* foi baseado no número de condição e parâmetro de positividade da matriz de covariância da abordagem RLS, como também em estatísticas de primeira e segunda ordem que foram associadas com propriedades dos estimadores recursivos.

Na análise comparativa dos transitórios das evoluções dos processos iterativos da solução da equação HJB-*Riccati*, observou-se uma similaridade entre os comportamentos de convergência dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR durante as primeiras iterações do processo iterativo. Quanto à análise da estabilidade numérica, verificou-se que o problema da perda de estabilidade numérica ocorre durante o regime permanente da evolução do processo iterativo da solução da equação HJB-*Riccati*, enquanto durante a fase do transitório tem-se informação o suficiente para gerar a base de vetores de regressores.

Com respeito à adaptabilidade dos estimadores RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR em relação à variações paramétricas na planta, observou-se que, em geral, os estimadores RLS$_\mu$-UDU^T-HDP-DLQR retornaram melhores resultados quando comparados com os estimadores RLS$_\mu$-HDP-DLQR, apresentando um processo de adaptabilidade mais rápido em relação às variações do tipo p_1 e p_2, ao passo que os estimadores RLS$_\mu$-HDP-DLQR mostraram-se menos adaptáveis às variações na planta na maioria dos casos considerados nos experimentos, especialmente para variações do tipo p_1 em que não ocorreu a adaptabilidade para nenhuma situação, exceto para o caso $p_1^{0^+}$, com $\alpha \in (0, 0.25]$.

Na avaliação de desempenho dos estimadores RLS$_\mu$-HDP-DLQR em termos da seleção do fator de esquecimento, observou-se a grande influência deste fator no processo de convergência e estabilidade numérica da solução da equação HJB-*Riccati*. Verificou-se também que uma escolha apropriada deste fator em conjunção com os métodos de fatoração UDU^T pode contribuir significativamente para

acelerar o processo de convergência da solução da equação HJB-*Riccati*, garantindo ao mesmo tempo um limite aceitável para estabilidade numérica de estimadores RLS$_\mu$-HDP-DLQR. Os experimentos também mostraram que a estabilidade dos algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR decaem consideravelmente para valores de μ menores que 0.9 e maiores que 0.97.

Em uma maneira geral, observou-se que a inserção da fatoração UDU^T no processo de estimação RLS da solução da equação HJB-*Riccati* via HDP contribuiu para solucionar problemas de convergência e estabilidade numérica por uma escolha plausível do fator de esquecimento. A robustez numérica da fatoração UDU^T tornou-se bastante evidente quando as variações paramétricas do tipo p_1 e p_2 foram levadas em consideração. Os número de condição e parâmetro de positividade foram usados como medidas de desempenho para avaliar anomalias no comportamento da matriz de covariância associadas com o grau de dependência linear dos vetores de regressores, os quais devem formar uma base para a aproximação RLS da solução HJB-*Riccati*.

O desempenho de métodos RLS-TD(λ) foi avaliado para parametrizações do DLQR em um sistema dinãmico multivariável de terceira ordem. As propriedades do RLS foram avaliadas em termos de estatísticas de primeira e segunda ordem para estimação da função de custo DLQR. Verificou-se que uma escolha adequada do parâmetro λ pode incrementar melhorias no processo de aprendizagem da função valor ótima no sentido de acelerar o processo de busca de políticas de controle ótimas .

Capítulo 7

Conclusões

Métodos alternativos para estabelecer políticas de controle ótimas que são baseadas na forma HJB da equação algébrica de *Riccati* e teoria de aprendizagem por reforço foram apresentados neste trabalho. Os desenvolvimentos apresentados foram orientados por meio de esquemas críticos adaptativos em que a função valor é modelada em termos de recompensa em longo prazo. Especificamente, a programação dinâmica heurística e sua estratégia dependente de ação foram apresentadas para a solução *online* de controle ótimo discreto do tipo linear quadrático.

Os algoritmos de programação dinâmica heurística com suas versões dependentes de ação foram formulados como uma aproximação da função valor para uma dada política. A aproximação utilizou métodos incrementais tais como descida do gradiente e RLS para resolver o problema de mínimos quadrados associado com a aproximação da função valor sobre uma formulação apoiada pela teoria da vetorização de matrizes e álgebra de *Kronecker*. A aproximação foi apresentada para determinar os coeficientes da matriz P e os ganhos do controlador ótimo DLQR para HDP e ADHDP, respectivamente.

No contexto de projeto de controle ótimo *online* via aprendizagem por reforço, uma contribuição principal nesta pesquisa é o desenvolvimento de um novo método para melhorar o processo de estimação RLS de políticas de decisão ótimas sob uma iteração de política otimística. A formulação proposta tem por base os métodos de fatoração UDU^T que reduzem problemas de estabilidade numérica da implementação de precisão finita dos algoritmos RLS$_\mu$-HDP-DLQR. O processo de

convergência e estabilidade numérica de tais algoritmos é monitorado e guiado por medidas de desempenho que avaliam o comportamento da matriz de covariância face ao grau de dependência linear dos vetores de regressores que formam uma base para a aproximação RLS da solução da equação HJB-*Riccati*.

Um método para metrificar o desempenho das aproximações para funções valor para solução *online* da equação da otimalidade de *Bellman* foi apresentado. O método proposto foi baseado em momentos estatísticos de primeira e segunda ordem que foram associados com as propriedades dos estimadores recursivos RLS, com respeito à positividade e independência linear orientada para explorar as propriedades de estimadores não-polarizados.

Os resultados mostraram-se promissores para um modelo de sistema dinâmico multivariável, desde que os problemas relacionados à polarizações das observações da planta são levados em consideração durante as medições, tais como a relação entre estimador e sensoriamento do mundo real em termos de observações que levam à problemas de dependência linear. Os resultados dos experimentos RLS$_\mu$-UDU^T-HDP-DLQR revelam que a fatoração UDU^T, por uma seleção apropriada do fator de esquecimento, pode incrementar melhorias substanciais nos métodos RLS$_\mu$-HDP-DLQR. Além disso, os resultados dos experimentos RLS-TD(λ) evidenciam que o ajuste adequado do parâmetro λ pode acelerar o processo de aprendizagem de política de controle ótima DLQR.

7.1 Trabalhos Futuros

Para continuação e direcionamento de futuros trabalhos, convém destacar os seguintes tópicos:

- Investigar a solução de problemas de estabilidade numérica dos algoritmos RLS$_\mu$-HDP explorando outros métodos de fatoração da matriz de covariância da abordagem RLS, tais como fatoração QR que consiste na decomposição da matriz de covariância como um produto de uma matriz ortogonal Q por uma matriz triangular superior R.

- Avaliar a habilidade dos métodos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-ADHDP-DLQR para alocar autoestruturas no plano Z estável por meio de variações heurísticas nas matrizes de ponderação Q e R do projeto DLQR. No contexto de

projeto de controle *online*, uma abordagem interessante para estabelecer políticas de decisão ótimas foi desenvolvida em (Fonseca Neto e Rêgo, 2014), onde os autores propõem uma metodologia baseada na fusão do método de sintonia QR e soluções aproximadas LS da equação HJB via paradigmas HDP e ADHDP. O método de sintonia QR proposto é apresentado como uma metodologia para sintonizar controladores ótimos baseada em variações heurísticas das matrizes Q e R que são guiadas por uma relação aproximada destas matrizes. Em uma pesquisa futura, pode-se pensar na fusão do método de sintonia QR, paradigmas ADHDP e soluções aproximadas RLS da equação HJB-*Riccati* para impor margens de estabilidade de projetos de controle.

- Utilizar os métodos de estimação paramétrica RLS$_\mu$-UDU^T em outros esquemas críticos adaptativos (ADHDP, DHP e ADDHP).

- Avaliar o desempenho dos algorimos RLS$_\mu$-ADHDP-DLQR para o projeto *online* de sistema de controle ótimo. Uma vantagem da formulação ADHDP é que ela permite aprender a política de controle ótima sem usar conhecimento do modelo da dinâmica do sistema.

- Estender os métodos propostos nesta tese para processos dinâmicos parcialmente observáveis nos quais nem todos os estados podem ser observados diretamente da planta. No contexto de projeto *online* de sistemas de controle ótimo linear quadrático baseado em paradigmas de ADP, especificamente no projeto do regulador linear quadrático, um procedimento óbvio seria a inserção do filtro de *Kalman* para estimar os estados não-observáveis. Werbos sugere algumas formas padrões para desenvolver uma estimativa de estado, a qual pode ser usada como a entrada principal para uma rede crítica ou uma rede de ação: filtro de *Kalman* estendido, filtro de partícula, treinamento de uma rede TLRN (*time-lagged recurrent network*) para predizer x a partir da saída Y em dados simulados, extração da saída dos nodos recorrentes de uma rede neural usados para modelar a planta.

- Estender os métodos desenvolvidos nesta tese para aproximadores de funções valor mais gerais. Neste caso, pode-se pensar em uma rede neural de

múltiplas camadas treinada por métodos RLS. Os métodos desenvolvidos aqui tipicamente poderiam ser vistos como uma estrutura de uma rede neural com um único neurônio de processamento com função de ativação linear para aproximar a solução da equação HJB, sendo que as entradas da rede são dadas pelas componentes do vetor de regressores e os pesos sinápticos associados a aquele neurônio são as componentes do vetor de parâmetros θ da aproximação $\widehat{V}(\cdot, \theta)$.

Apêndice A

Cadeias de Markov

O objetivo deste apêndice é fornecer brevemente algumas definições básicas e resultados da teoria de cadeias de Markov que tem interesse sobre a análise de convergência dos métodos TD(λ) com aproximação de função. Para um tratamento mais detalhado consulte, por exemplo, o livro-texto (Iosifescu, 2007).

A.1 Definições

Definição A.1.1. Uma matriz quadrada A $m \times m$ diz-se ser não-negativa ($A \geq 0$) se $a_{ij} \geq 0$ para todo $i, j \in \{1, ..., m\}$. No caso de $a_{ij} > 0$ para todo $i, j \in \{1, ..., m\}$, diz-se que A é positiva ($A > 0$).

Definição A.1.2. Uma matriz não-negativa A $m \times m$ diz-se ser estocástica se $\sum_{j=1}^{m} a_{ij} = 1$ para todo $i \in \{1, ..., m\}$. Uma matriz estocástica A $m \times m$ que tem linhas idênticas é chamada estável.

Definição A.1.3. A cadeia de Markov neste contexto é determinada por um conjunto infinito enumerável ou finito S cujos elementos, chamados estados, podem ser enumerados em uma forma definida, uma distribuição de probabilidade $P^0 = (p_i)_{i \in S}$ sobre S, cujas componentes são chamadas probabilidades iniciais, e uma matriz estocástica P$= (p_{ij})_{i,j \in S}$, cujos elementos são chamadas probabilidades de transição. Portanto

$$p_i \geq 0, \quad i \in S, \quad \sum_{i \in S} p_i = 1$$

$$p_{ij} \geq 0, \quad i,j \in S, \quad \sum_{j \in S} p_{ij} = 1, \quad i \in S.$$

Diz-se que uma seqüência de variáveis aleatórias $\{X_t\}_{t \geq 0}$ de valores em S é uma cadeia de Markov (homogênea) com espaço de estado S, distribuição inicial P^0 e uma matriz de transição P se, e somente se,

$$P_r\left(X_{t+1} = i_{t+1} \,|X_t = i_t,\ \ldots\ , X_0 = i_0\right) = P_r\left(X_{t+1} = i_{t+1} \,|X_t = i_t\right) = p_{i_t\, i_{t+1}} \quad \text{(A.1)}$$

para todo $t \geq 0$ e $i_0, \ \ldots\ , i_{t+1} \in S$. A primeira igualdade é conhecida como propriedade de Markov e diz que a distribuição de X_{t+1} é independente de todos os estados prévios em que se encontrou a cadeia, dependendo apenas do estado atual. A homogeneidade é introduzida pela segunda igualdade que diz que a probabilidade condicional de X_{t+1} estar no estado i_{t+1} dado que X_t está no estado i_t tem um valor prescrito independente de t.

Escrevendo

$$p_{ij}^{(n)} = P_r\left(X_t = j \,|X_{t-n} = i\right) \quad \text{(A.2)}$$

para designar as probabilidades de transição a n-passos, tem-se

$$p_{ij}^{(1)} = p_{ij}, \qquad p_{ij}^{(n+1)} = \sum_k p_{ik}^{(n)} p_{kj}. \quad \text{(A.3)}$$

Denotando por $\mathrm{P}^{(n)}$ a matriz cujos elementos são as probabilidades de transição $p_{ij}^{(n)}$ a n passos, então por (A.3) tem-se a relação

$$\mathrm{P}^{(n)} = \mathrm{P}^n.$$

A.2 Classes de Estados

Os estados de uma cadeia de Markov podem ser agrupados em classes de acordo se outros estados de uma classe podem ser ou não alcançados a partir de qualquer estado dado da mesma classe. Diz-se que o estado i leva ao estado j ou,

equivalentemente, o estado j é acessível a partir do estado i, e escreve-se $i \to j$ se, e somente se, existe um $n \geq 1$ tal que $p_{ij}^{(n)} > 0$. Diz-se que o estado i se comunica com o estado j e escreve-se $i \leftrightarrow j$ se, e somente se, tem-se ambos $i \to j$ e $j \to i$. A relação "$\to$ "é transitiva, isto é, $i \to j$ e $j \to k$ implica $i \to k$. Isto é uma conseqüência imediata da desigualdade

$$p_{ik}^{(m+n)} \geq p_{ij}^{(m)} p_{jk}^{(n)}.$$

A relação $\leftrightarrow$ é também transitiva, e é obviamente simétrica, isto é, $i \leftrightarrow j$ implica $j \leftrightarrow i$. Entretanto, deve ser notado que a relação $\leftrightarrow$ não é reflexiva, uma vez que pode existir um estado que não se comunica com si próprio e portanto com nenhum outro estado (Tal estado é chamado "sem retorno".). Por exemplo, este é o caso do estado 0 na cadeia de Markov com a matriz de transição

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 1/4 & 3/4 \\ 0 & 1/2 & 1/2 \\ 0 & 1/3 & 2/3 \end{pmatrix} \end{array}.$$

Com exceção dos estados sem retorno a relação $\leftrightarrow$ é uma relação de equivalência sobre o conjunto dos estados restantes, isto é, $\leftrightarrow$ é reflexiva, simétrica e transitiva.

Um subconjunto M do espaço de estado diz-se ser fechado se, e somente se, $\sum_{j \in M} p_{ij} = 1$ para qualquer $i \in M$, isto é, a submatriz $(p_{ij})_{i,j \in M}$ da matriz de transição P é estocástica. Em outras palavras, M é fechado se, e somente se, não é possível deixar M dada que a cadeia iniciou em um estado pertencente a M. O exemplo mais simples de uma classe fechada é aquela que consiste de um único estado i e $p_{ii} = 1$. Tal estado diz-se ser absorvente. O outro caso extremo é aquele onde todo o espaço de estado S forma uma classe (fechada). Neste caso o espaço de estado (bem como a cadeia de Markov) diz-se ser irredutível.

Uma propriedade definida para todos os estados de uma cadeia de Markov diz-se ser uma propriedade de classe se, e somente se, sua pertinência para um estado i implica em sua pertinência por todos os estados da classe contendo i.

Um exemplo simples de uma propriedade de classe ocorre em conecção com o conceito do período de um estado. Se i é um estado de retorno, isto é, $p_{ii}^{(n)} > 0$ para algum $n \geq 1$, seu período d_i é definido como o maior divisor comum de todos os números naturais m tais que $p_{ii}^{(m)} > 0$. Um estado i diz-se ser periódico ou aperiódico de acordo se $d_i > 1$ ou $d_i = 1$.

Teorema A.2.1. *Dois estados distintos pertencentes à mesma classe tem o mesmo período. Em outras palavras, a propriedade de ter período d é uma propriedade de classe.*

O Teorema A.2.1 permite introduzir o conceito de "período de uma classe", e então classificar uma classe como periódica (cíclica) ou aperiódica (acíclica) de acordo se $d > 1$ ou $d = 1$.

A.3 Recorrência

De acordo se um estado ocorre ou não infinitamente geralmente dois tipos de estados são distinguidos. Primeiramente, defina o seguinte: para quaisquer dois estados i e j, as probabilidades de primeira passagem $f_{i,j}^{(n)}$ são dadas por

$$f_{ij}^{(n)} = P_r\left(X_t = j, X_{t-1} \neq j, \ \ldots, X_{t-n+1} \neq j \,|X_{t-n} = i\right),$$

sendo os tempos médios de primeira passagem (tempos médios de recorrência se $i = j$ definidos como

$$m_{ij} = \sum_{n=1}^{\infty} n f_{ij}^{(n)}, \tag{A.4}$$

desde que

$$\sum_{n=1}^{\infty} f_{ii}^{(n)} = 1. \tag{A.5}$$

Definição A.3.1. Um estado i diz-se ser recorrente ou transiente de acordo se $\sum_{n=1}^{\infty} f_{ii}^{(n)} = 1$ ou $\sum_{n=1}^{\infty} f_{ii}^{(n)} < 1$; se $m_{ii} < \infty$, diz-se que o estado i é um estado recorrente positivo.

Teorema A.3.1. *(fórmula de Doeblin): Quaisquer que sejam os estados i e j, tem-se*

$$\sum_{n=1}^{\infty} f_{ij}^{(n)} = \lim_{s\to\infty} \frac{\sum_{n=1}^{s} p_{ij}^{(n)}}{1+\sum_{n=1}^{s} p_{ij}^{(n)}}.$$

Corolário A.3.1. *O estado i é recorrente ou transiente de acordo se a série $\sum_{n=1}^{\infty} p_{ii}^{(n)}$ diverge ou converge.*

Corolário A.3.2. *Sejam i um estado arbitrário e j um estado transiente. Então a série $\sum_{n=1}^{\infty} p_{ij}^{(n)}$ converge, daí, em particular $\lim_{n\to\infty} p_{ij}^{(n)} = 0$.*

Corolário A.3.3. *O espaço de estado de uma cadeia de Markov contém pelo menos um estado recorrente.*

Teorema A.3.2. *Recorrência e transiência são propriedades de classe.*

Teorema A.3.3. *Seja i um estado recorrente. Se $i \to j$ então o estado j é também recorrente, e $\sum_{n=1}^{\infty} f_{ij}^{(n)} = \sum_{n=1}^{\infty} f_{ji}^{(n)} = 1$. Em outras palavras, um estado transiente não pode ser alcançado a partir de um estado recorrente.*

Teorema A.3.4. *Uma classe é recorrente se, e somente se, é fechada.*

A.4 Classificação de Cadeias de Markov Homogêneas

Uma cadeia de Markov pode ser classificada de acordo com o tipo de seus estados. Toda cadeia de Markov tem pelo menos um estado recorrente. Se não existem estados transientes, então os estados são enumerados de uma tal maneira que aqueles estados pertencentes à mesma classe tem números consecutivos. A matriz de transição pode ser escrita como

$$\mathrm{P} = \begin{pmatrix} P_1 & & & \\ & P_2 & & \\ & & \ddots & \\ & & & P_r \end{pmatrix},$$

em que $P_1, \ldots, P_r$ são matrizes estocásticas associadas com as classes recorrentes. Se $r > 1$ nenhuma comunicação é possível entre essas classes. De fato, tem-se r cadeias de Markov distintas que podem ser estudadas separadamente.

Uma cadeia de Markov sem estados transientes é chamada recorrente. Uma cadeia de Markov recorrente cujos estados formam somente uma classe é chamada irredutível. Uma cadeia de Markov em que todos os estados são aperiódicos é chamada aperiódica. Dado que a periodicidade é uma propriedade de classe, para que uma cadeia de Markov irredutível seja aperiódica é suficiente que um dos seus estados seja aperiódico. Cadeias de Markov irredutíveis, aperiódicas e com todos os estados recorrentes positivos são designadas por ergódicas.

Se a cadeia de Markov também tem estados transientes, então enumerando os estados de uma tal maneira que os estados transientes são colocados após os estados recorrentes e o último são tratados como no caso anterior, a matriz de transição pode ser escrita como

$$\mathrm{P}=\begin{pmatrix} P_1 & & & & \\ & P_2 & & 0 & \\ & & \ddots & & \\ & 0 & & P_r & \\ T_1 & T_2 & \cdots & T_r & T \end{pmatrix},$$

onde as matrizes $T_1, \ldots, T_r, T$ são associadas com os estados transientes: as primeiras r matrizes compreendem as probabilidades de transição dos estados transientes para os estados recorrentes e a última matriz as probabilidades de transição dentro do conjunto de estados transientes. As potências de P são, claramente, do mesmo tipo. Por exemplo

$$\mathrm{P}^2=\begin{pmatrix} P_1^2 & & & & \\ & P_2^2 & & 0 & \\ & & \ddots & & \\ & 0 & & P_r^2 & \\ T_1P_1+TT_1 & T_2P_2+TT_2 & \cdots & T_rP_r+TT_r & T^2 \end{pmatrix}.$$

Daí, para $i, j \in T$ as entradas $p_{ij}^{(m)}$ da matriz P^m coincide com as entradas correspondentes da matriz T^m. Isto é uma conseqüência imediata do fato de que

um estado transiente não pode ser alcançado a partir de um estado recorrente. Pelo Corolário A.3.2 do Teorema A.3.1, tem-se $\lim_{n\to\infty} T^n = 0$.

Observação A.4.1. Se uma cadeia de Markov inicia-se em um estado transiente então, com probabilidade 1, ele permanece no conjunto T de estados transientes por apenas um número finito de etapas após as quais ela entra em uma classe recorrente onde permanece para sempre. Por conta desta propriedade uma cadeia de Markov com estados transientes será chamada absorvente. Uma cadeia de Markov absorvente com somente uma classe recorrente aperiódica é chamada indecomponível.

A.5 Teorema do Limite Básico para Cadeias de Markov Ergódicas

Teorema A.5.1. *Seja uma cadeia de Markov ergódica caracterizada por uma matriz estocástica* P, então P^n converge quando $n \to \infty$ para uma matriz estocástica estável positiva $\overline{\mathrm{P}} = 1'v$, sendo $v = P^0 \lim_{n\to\infty} \mathrm{P}^n$ um vetor de probabilidade com entradas não nulas e é único apesar da distribuição inicial P^0.

O Teorema A.5.1 declara, em outras palavras, que as probabilidades de transição a n-passos $p_{ij}^{(n)}$ para cadeias de Markov ergódicas convergem quando $n \to \infty$ para limites independentes do estado inicial, isto é,

$$\lim_{n\to\infty} p_{ij}^{(n)} = v_j\,, \qquad \forall i, \tag{A.6}$$

sendo v_j a j-ésima componente do vetor linha v. Esta propriedade é chamada "ergodicidade". Mais ainda, v é o único vetor que satisfaz as equações de estacionaridade

$$v = vP. \tag{A.7}$$

Diz-se, nestas condições, que v é a distribuição estacionária (invariante) da cadeia.

Apêndice B

Coeficientes da Fatoração UDU^T

Admitindo-se que

$$D(k-1) - \beta^{-1} g g^T = \overline{U}\,\overline{D}\,\overline{U}^T, \tag{B.1}$$

sendo $\overline{U} = [\overline{u}_1 \ \cdots \ \overline{u}_{n_\theta}]$ e $\overline{D} = diag(\overline{d}_1, \ \cdots, \ \overline{d}_{n_\theta})$, com $\overline{u}_j = [\overline{u}_{1j} \ \cdots \ \overline{u}_{j-1j} \ 1 \ 0 \ \cdots \ 0]^T$, então os fatores U-D de $\Gamma(k)$ são dados por

$$U(k) = U(k-1)\overline{U} \tag{B.2}$$

e

$$D(k) = \overline{D}\mu^{-1} \ . \tag{B.3}$$

A Eq.(B.1) pode ser transformada em

$$\sum_{i=1}^{n_\theta} \overline{d}_i \overline{u}_i \overline{u}_i^T = \sum_{i=1}^{n_\theta} d_i v_i v_i^T - \beta^{-1} g g^T, \tag{B.4}$$

sendo $d_i = d_i(k-1)$ e v_i é o i-ésimo versor.

Definindo-se

$$\beta_{n_\theta} = \beta \quad , \quad \beta_m = \mu + \sum_{i=1}^{m} e_i g_i \tag{B.5}$$

$$\psi_{n_\theta} = g \quad , \quad \psi_{m-1} = [\psi_{m1} \ \ldots \ \psi_{mm-1} \ 0 \ \ldots \ 0]^T \ , \tag{B.6}$$

a Eq.(B.4) pode ser transformada em

$$\sum_{i=1}^{n_\theta} \overline{d}_i \overline{u}_i \overline{u}_i^T - \sum_{i=1}^{n_\theta} d_i v_i v_i^T + \beta_{n_\theta}^{-1} \psi_{n_\theta} \psi_{n_\theta}^T = 0 \tag{B.7}$$

$$\sum_{i=1}^{n_\theta - 1} \overline{d}_i \overline{u}_i \overline{u}_i^T - \sum_{i=1}^{n_\theta - 1} d_i v_i v_i^T + M_{n_\theta} = 0, \tag{B.8}$$

sendo

$$M_{n_\theta} = \overline{d}_{n_\theta} \overline{u}_{n_\theta} \overline{u}_{n_\theta}^T - d_{n_\theta} v_{n_\theta} v_{n_\theta}^T + \beta_{n_\theta}^{-1} \psi_{n_\theta} \psi_{n_\theta}^T. \tag{B.9}$$

Observe que as matrizes sob o sinal dos somatórios na Eq.(B.8) possuem a n_θ-ésima linha e a n_θ-ésima coluna nulas. Portanto, a fim de que a Eq.(B.8) seja satisfeita deve-se estabelecer as seguintes condições sobre os elementos das respectivas fileiras da matriz M_{n_θ}

$$\overline{d}_{n_\theta} = d_{n_\theta} - \beta_{n_\theta}^{-1} \psi_{n_\theta n_\theta}^2 \tag{B.10}$$

e

$$\overline{u}_{i n_\theta} = \frac{-\psi_{n_\theta n_\theta} \psi_{n_\theta i}}{\beta_{n_\theta} \overline{d}_{n_\theta}} \tag{B.11}$$

Substituindo-se as Eqs.(B.10) e (B.11) na Eq.(B.9), a matriz M_{n_θ} assume a forma

$$M_{n_\theta} = \overline{d}_{n_\theta}\frac{\psi^2_{n_\theta n_\theta}}{\overline{d}^2_{n_\theta}\beta^2_{n_\theta}}\psi_{n_\theta-1}\psi^T_{n_\theta-1} + \beta^{-1}_{n_\theta}\psi_{n_\theta-1}\psi^T_{n_\theta-1} \tag{B.12}$$

$$= \left(\frac{\psi^2_{n_\theta n_\theta}}{\overline{d}^2_{n_\theta}\beta^2_{n_\theta}} + \frac{1}{\beta_{n_\theta}}\right)\psi_{n_\theta-1}\psi^T_{n_\theta-1}. \tag{B.13}$$

ou, de acordo com as Eqs.(B.5), (B.6) e (B.10) tem-se

$$M_{n_\theta} = \frac{\psi^2_{n_\theta n_\theta} + d_{n_\theta}\beta_{n_\theta} - \beta^{-1}_{n_\theta}\psi^2_{n_\theta n_\theta}\beta_{n_\theta}}{\overline{d}_{n_\theta}\beta^2_{n_\theta}}\psi_{n_\theta-1}\psi^T_{n_\theta-1} \tag{B.14}$$

$$= \frac{d_{n_\theta}}{d_{n_\theta}\beta_{n_\theta} - \beta^{-1}_{n_\theta}\psi^2_{n_\theta n_\theta}\beta_{n_\theta}}\psi_{n_\theta-1}\psi^T_{n_\theta-1} \tag{B.15}$$

$$= \frac{1}{\beta_{n_\theta} - \frac{\psi^2_{n_\theta n_\theta}}{d_{n_\theta}}}\psi_{n_\theta-1}\psi^T_{n_\theta-1} \tag{B.16}$$

$$= \beta^{-1}_{n_\theta-1}\psi_{n_\theta-1}\psi^T_{n_\theta-1}, \tag{B.17}$$

e a Eq.(B.7) pode ser então expressa por

$$\sum_{i=1}^{n_\theta-1}\overline{d}_i\overline{u}_i\overline{u}^T_i - \sum_{i=1}^{n_\theta-1}d_i v_i v^T_i + \beta^{-1}_{n_\theta-1}\psi_{n_\theta-1}\psi^T_{n_\theta-1} = 0. \tag{B.18}$$

Deve-se observar com ênfase que as Eqs.(B.7) e (B.18) diferem uma da outra somente pela variação do somatório e índices. Repetindo-se o mesmo raciocínio utilizado nas tranformações (B.7)-(B.14) para índices variando de $n_\theta - 1$ a 1, é possível determinar todos os valores de d_i e u_{ij}.

A Eq.(B.11) é transformada usando-se Eqs.(B.5), (B.6) e (B.10) como segue

$$\overline{u}_{in_\theta} = \frac{-\psi_{n_\theta n_\theta}\psi_{n_\theta i}}{\beta_{n_\theta}\overline{d}_{n_\theta}} = \frac{-\psi_{n_\theta n_\theta}\psi_{n_\theta i}}{\beta_{n_\theta}\left(d_{n_\theta} - \frac{\psi^2_{n_\theta n_\theta}}{\beta_{n_\theta}}\right)} \tag{B.19}$$

$$= \frac{-\psi_{n_\theta n_\theta}\psi_{n_\theta i}}{d_{n_\theta}\left(\beta_{n_\theta} - \frac{\psi^2_{n_\theta n_\theta}}{d_{n_\theta}}\right)} = \frac{-\psi_{n_\theta n_\theta}\psi_{n_\theta i}}{d_{n_\theta}\beta_{n_\theta-1}}. \tag{B.20}$$

Pondo-se

$$\tau_{n_\theta} = \frac{-\psi_{n_\theta n_\theta}}{d_{n_\theta}\beta_{n_\theta-1}} = \frac{-e_{n_\theta}}{\beta_{n_\theta-1}}, \tag{B.21}$$

obtém-se

$$\overline{u}_{i\,n_\theta} = -\tau_{n_\theta}\psi_{n_\theta\,i}. \tag{B.22}$$

Usando-se Eqs.(B.5), (B.6) e (B.10), os valores dos elementos da matriz D são determinados como segue

$$d_i(k) = \overline{d}_i\mu^{-1} = \left(d_i - \frac{\psi_{ii}^2}{\beta_i}\right)\mu^{-1} \tag{B.23}$$

$$= d_i\frac{1}{\beta_i}\left(\beta_i - \frac{\psi_{ii}^2}{d_i}\right)\mu^{-1} = d_i\frac{\beta_{i-1}}{\beta_i\mu}. \tag{B.24}$$

O numerador de (5.12) assume a seguinte forma

$$\kappa(k) = U(k-1)g = U(k-1)\psi_{n_\theta}. \tag{B.25}$$

Daí,

$$\kappa_i = \sum_{m=i}^{n_\theta} u_{im}(k-1)\psi_{n_\theta m} \tag{B.26}$$

$$= \sum_{m=i}^{n_\theta-1} u_{im}(k-1)\psi_{n_\theta m} + u_{i\,n_\theta}(k-1)\psi_{n_\theta\,n_\theta}, \tag{B.27}$$

ou, em uma forma de recorrência,

$$\kappa_{i,new} = \kappa_{i,old} + u_{in_\theta}(k-1)\psi_{n_\theta\,n_\theta}. \tag{B.28}$$

Os valores dos elementos da matriz U são determinados como segue

$$u_{ij}(k) = \sum_{m=i}^{j} u_{im}(k-1)\overline{u}_{mj} \tag{B.29}$$

$$= u_{ij}(k-1) + \sum_{m=i}^{j-1} u_{im}(k-1)\tau_j\psi_{jm} \tag{B.30}$$

$$= u_{ij}(k-1) + \tau_j\sum_{m=i}^{j-1} u_{im}(k-1)\psi_{jm} \tag{B.31}$$

$$= u_{ij}(k-1) + \tau_j\kappa_i. \tag{B.32}$$

ALGORITMO 9 RLS-UDU^T

1 ——————————————————————————
2 ▷ - **Inicialização**
3 $D(0) = \delta I, \qquad \delta > 0$
4 $U(0) = I$
5 $\theta(0) = \theta_0.$
6 ——————————————————————————
7 ▷ - Processo Iterativo
8 **for** $k = 1 : N$
9 **do**
10 $\varepsilon(k) = y(k) - \phi^T(k)\theta(k-1)$
11 $e = U^T(k-1)\phi(k)$
12 $g = D(k-1)e$
13 $\beta_0 = \mu$
14 **for** $j = 1 : n_\theta$
15 **do**
16 $\beta_j = \beta_{j-1} + e_j\, g_j$
17 $d_j(k) = d_j(k-1)\frac{\beta_{j-1}}{\beta_j \mu}$
18 $\kappa_j = g_j$
19 $\tau_j = \frac{-e_j}{\beta_{j-1}}$
20 **if** $j \neq 1$
21 **do**
22 **for** $m = 1 : j-1$
23 $u_{mj}(k) = u_{mj}(k-1) + \tau_j \kappa_m$
24 $\kappa_{m,new} = \kappa_{m,old} + u_{mj}(k-1)\kappa_j$
25 **End**
26 **End**
27 **End**
28 Atualização do vetor de ganho
29 $L(k) = \frac{[\kappa_1 \quad \kappa_2 \quad \ldots \quad \kappa_{n_\theta}]^T}{\beta_{n_\theta}}$
30 Atualização do vetor de parâmetros
31 $\theta(k) = \theta(k-1) + L(k)\varepsilon(k)$
32 Atualização do vetor de regressores $\phi(k)$
33 **End**
34 **Fim do Algoritmo 9**

Apêndice C

RLS$_\mu$-UDU^T-HDP-DLQR Otimístico

Algoritmo 10 - RLS$_\mu$-UDU^T-HDP-DLQR Otimístico

▷ - **Bloco 0 - Inicialização**
▷ Matrizes do Sistema Dinâmico e de Ponderação
$[Q, R, A, B] \leftarrow [\quad\quad]$
▷ Selecione - Fator de Desconto - $0 < \gamma \leq 1$.
▷ Parâmetros do RLS: $\widehat{\theta}_0$, U_0, D_0
Selecione - Fator de Esquecimento - $0 < \mu \leq 1$.
▷ Selecione - Estado Inicial - x_0.
▷ Selecione - Política Inicial
$K_0 = \gamma(R + \gamma B^T P^{\widehat{\theta}_0} B)^{-1} B^T P^{\widehat{\theta}_0} A$.
$k \leftarrow 0$
▷ - Processo Iterativo
Para cada passo de tempo k
do
▷ **Bloco 1 - Simulação do Ambiente**
▷ Ação de Controle
$u_k \leftarrow -K_k x_k$
▷ Estados
$x_{k+1} \leftarrow Ax_k + Bu_k$
▷ **Bloco 2 - Atualização da Política e da Função Valor**
▷ 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*
$\phi_k \leftarrow [x_{1,k}^2; x_{1,k}x_{2,k}; x_{1,k}x_{3,k}; x_{1,k}x_{4,k}; x_{2,k}^2;$
$x_{2,k}x_{3,k}; x_{2,k}x_{4,k}; x_{3,k}^2; x_{3,k}x_{4,k}; x_{4,k}^2]$
$-\gamma[x_{1,k+1}^2; x_{1,k+1}x_{2,k+1}; x_{1,k+1}x_{3,k+1};$
$x_{1,k+1}x_{4,k+1}; x_{2,k+1}^2; x_{2,k+1}x_{3,k+1};$
$x_{2,k+1}x_{4,k+1}; x_{3,k+1}^2; x_{3,k+1}x_{4,k+1};$
$x_{4,k+1}^2]$
Atualize $\widehat{\theta}_k$ seguindo as etapas 10-31 do algoritmo 9
Associação - Matriz $P^{\widehat{\theta}_{k+1}}$ com $\widehat{\theta}_{k+1}$
$P^{\widehat{\theta}_{k+1}} \leftarrow [\widehat{\theta}_{1,k+1} \quad \widehat{\theta}_{2,k+1}/2 \quad \widehat{\theta}_{3,k+1}/2 \quad \widehat{\theta}_{4,k+1}/2;$
$\widehat{\theta}_{5,k+1} \quad \widehat{\theta}_{6,k+1}/2 \quad \widehat{\theta}_{7,k+1}/2;$
$\widehat{\theta}_{8,k+1} \quad \widehat{\theta}_{9,k+1}/2;$
$\widehat{\theta}_{10,k+1}]$
▷ Atualização da Política
$K_{k+1} = \gamma(R + \gamma B^T P^{\widehat{\theta}_{k+1}} B)^{-1} B^T P^{\widehat{\theta}_{k+1}} A$
$k \leftarrow k + 1$
until K_{k+1} seja satisfatória
then Fim do Processo Iterativo
Fim do Algoritmo 10.

Apêndice D

Provas de Lemas

Lema D.0.1. *Sob as suposições A.1 e A.2 apresentadas na Subseção 5.6.1, as seguintes relações se mantêm:*

(a) $\mathrm{E}_0[\varphi(x_k)\varphi^T(x_{k+m})] = \Phi^T D M^m \Phi$, *para* $m \geq 0$.

(b) $\mathrm{E}_0[\xi_k \varphi^T(x_k)] = \Phi^T D \sum_{m=0}^{\infty} (\gamma\lambda)^m M^m \Phi$.

(c) $\mathrm{E}_0[\xi_k r_k] = \sum_{m=0}^{\infty} (\gamma\lambda)^m \Phi^T D M^m \overline{r}$, *sendo* $\overline{r}$ *o vetor de dimensão* $card(X)$, *cuja* i*-ésima componente é igual a* $\mathrm{E}[r_k \,|\, x_k = i]$.

Demonstração:

Primeiramente, observa-se que para quaisquer V e $\overline{V}$, tem-se

$$\begin{aligned} &\mathrm{E}_0[V(x_k)\overline{V}(x_{k+m})] \\ &\quad = \sum_{i\in X} \pi(i) \sum_{j\in X} P_r(x_{k+m} = j \;|x_k = i) V(i)\overline{V}(j) \\ &\quad = \sum_{i\in X} \pi(i) V(i) [M^m \overline{V}](i) \\ &\quad = V^T D M^m \overline{V}, \end{aligned} \tag{D.1}$$

em que $[M^m\overline{V}](i)$ denota a i-ésima componente do vetor $M^m\overline{V}$. Em particular para o caso onde os vetores são da forma $V = \Phi\theta$, $\overline{V} = \Phi\overline{\theta}$, obtém-se

$$\mathrm{E}_0\left[\theta^T \varphi(x_k)\varphi^T(x_{k+m})\overline{\theta}\right] = \theta^T \Phi^T D M^m \Phi \overline{\theta}. \tag{D.2}$$

Uma vez que os vetores θ e $\overline{\theta}$ são arbitrários, segue-se que

$$\mathrm{E}_0\left[\varphi(x_k)\varphi^T(x_{k+m})\right] = \Phi^T D M^m \Phi. \tag{D.3}$$

Agora, verifica-se a parte (b) do lema. Observe que $\mathrm{E}_0\left[\xi_k\varphi^T(x_k)\right]$ é o mesmo para todo k, e basta provar o resultado para o caso $k = 0$. Tem-se

$$\begin{aligned}\mathrm{E}_0\left[\xi_0\varphi^T(x_0)\right] &= \mathrm{E}_0\left[\sum_{\tau=-\infty}^{0}(\gamma\lambda)^{-\tau}\varphi(x_\tau)\varphi^T(x_0)\right] \\ &= \sum_{\tau=-\infty}^{0}(\gamma\lambda)^{-\tau}\mathrm{E}_0[\varphi(x_\tau)\varphi^T(x_0)],\end{aligned} \tag{D.4}$$

onde a permuta da soma e a esperança é justificada pelo teorema de convergência dominada (Tsitsiklis e Roy, 1997). O resultado desejado segue usando-se o resultado da pate (a). Portanto,

$$\mathrm{E}_0\left[\xi_k\varphi^T(x_k)\right] = \Phi^T D \sum_{m=0}^{\infty}(\gamma\lambda)^m M^m \Phi. \tag{D.5}$$

Usando-se um argumento similar, obtém-se também

$$\mathrm{E}_0\left[\xi_k\varphi^T(x_{k+1})\right] = \Phi^T D \sum_{m=0}^{\infty}(\gamma\lambda)^m M^{m+1} \Phi. \tag{D.6}$$

Daí,

$$\begin{aligned}C &= \mathrm{E}_0\left[\xi_k(\gamma\varphi^T(x_{k+1}) - \varphi^T(x_k)\right] \\ &= \Phi^T D \sum_{m=0}^{\infty}(\gamma\lambda)^m(\gamma M^{m+1} - M^m)\Phi \\ &= \Phi^T D(G - I)\Phi,\end{aligned} \tag{D.7}$$

sendo $G = (1-\lambda)\sum_{m=0}^{\infty}\lambda^m(\gamma M)^{m+1}$.

Agora, a Eq.(D.1) é usada, com $V = \Phi\theta$ e $\overline{V} = \overline{r}$, para obter

$$\begin{aligned}\theta^T d &= \mathrm{E}_0[\theta^T\xi_0 r_k] \\ &= \mathrm{E}_0[\theta^T\xi_0\overline{r_k}] \\ &= \mathrm{E}_0\left[\theta^T\sum_{m=0}^{\infty}(\gamma\lambda)^m\varphi(x_{-m})\overline{r_k}\right] \\ &= \theta^T\Phi^T Dh,\end{aligned} \tag{D.8}$$

sendo $h = \sum_{m=0}^{\infty} (\gamma\lambda M)^m \overline{r}$. Uma vez que o vetor θ é arbitrário, segue a fórmula desejada para d.

Lema D.0.2. *Para qualquer $V \in \Re^n$, tem-se $\|MV\|_D \leq \|V\|_D$.*

Demonstração:

Seja m_{ij} a ij-ésima entrada de M. Então,

$$\begin{aligned}
\|MV\|_D^2 &= V^T M^T D M V \\
&= \sum_{i=1}^{n} \pi(i) \left(\sum_{j=1}^{n} m_{ij} V(j) \right)^2 \\
&\leq \sum_{i=1}^{n} \pi(i) \sum_{j=1}^{n} m_{ij} V^2(j) \\
&= \sum_{j=1}^{n} \sum_{i=1}^{n} \pi(i) m_{ij} V^2(j) \\
&= \sum_{j=1}^{n} \pi(j) V^2(j) \\
&= V^T D V \\
&= \|V\|_D^2 .
\end{aligned} \tag{D.9}$$

Na prova acima utilizou-se a desigualdade $\mathrm{E}[V(x_{k+1}) \mid x_k = i]^2 \leq \mathrm{E}[V(x_{k+1})^2 \mid x_k = i]$ (Desigualdade de Jensen), bem como a propriedade $\pi^T M = \pi^T$ do vetor de probabilidades invariantes.

Lema D.0.3.

(a)*Para todo $V \in \Re^n$, tem-se*

$$\|GV\|_D \leq \frac{\gamma(1-\lambda)}{1-\gamma\lambda} \|V\|_D. \tag{D.10}$$

(b)*A matriz C é definida negativa.*

Demonstração:

(a) Do Lema D.0.2, segue que $\|M^m V\|_D \leq \|V\|_D$, $\forall V, m \geq 0$. Usando a definição de G e a desigualdade triangular, obtém-se

$$\|GV\|_D \leq (1-\lambda)\sum_{m=0}^{\infty}\lambda^m\gamma^{m+1}\|V\|_D = \frac{\gamma(1-\lambda)}{1-\gamma\lambda}\|V\|_D.$$

(b) Denotando por $\|\cdot\|$ a norma Euclidiana, e usando a Desigualdade de Schwartz, tem-se

$$\begin{aligned} V^T DGV &= V^T D^{1/2} D^{1/2} GV \\ &\leq \left\|D^{1/2}V\right\| \left\|D^{1/2}GV\right\| \\ &= \|V\|_D \|GV\|_D \\ &\leq \gamma\|V\|_D \|V\|_D \\ &= \gamma V^T DV. \end{aligned}$$

Assim,

$$V^T D(G-I)V \leq -(1-\gamma)V^T DV < 0\,, \quad \forall V \neq 0,$$

que prova que $D(G-I)$ é definida negativa. Usando-se a suposição que a matriz Φ tem rank completo, segue que $C = \Phi^T D(G-I)\Phi$ é definida negativa.

Da definição de um operador T^λ estabelecida nas Eqs.(5.110) e (5.111), pode ser verificado usando-se álgebra simples que para $0 \leq \lambda \leq 1$, tem-se

$$\begin{aligned} (T^{(\lambda)}V)(x_k) &= (1-\lambda)\sum_{m=0}^{\infty}\lambda^m(\gamma M)^{m+1}V + \sum_{m=0}^{\infty}(\gamma\lambda M)^m\overline{r} \\ &= GV + h. \end{aligned} \tag{D.11}$$

Uma importante propriedade de T^λ é estabelecida no próximo lema, o qual declara que T^λ é uma contração com respeito à norma $\|\cdot\|_D$.

Lema D.0.4. *Para quaisquer $V, \overline{V} \in \Re^n$ e $0 \leq \lambda \leq 1$, tem-se*

$$\left\|T^{(\lambda)}V - T^{(\lambda)}\overline{V}\right\|_D \leq \frac{\gamma(1-\lambda)}{1-\gamma\lambda}\left\|V - \overline{V}\right\|_D. \tag{D.12}$$

Demonstração:

Do Lema D.0.3 e da Eq.(D.11), tem-se:

$$\begin{aligned}\left\|T^{(\lambda)}V - T^{(\lambda)}\overline{V}\right\|_D &= \left\|GV + h - (G\overline{V} + h)\right\|_D \\ &= \left\|G(V - \overline{V})\right\|_D \\ &\leq \tfrac{\gamma(1-\lambda)}{1-\gamma\lambda}\left\|V - \overline{V}\right\|_D.\end{aligned}$$

Uma vez que a sequência θ_k gerada pelas Eqs.(2.47) e (2.48) converge para a solução do sistema $C\theta + d = 0$, tem-se:

$$\begin{aligned}C\theta + d &= \Phi^T D(G - I)\Phi\theta + \Phi^T Dh \\ &= \Phi^T D(G\Phi\theta + h) - \Phi^T D\Phi\theta \\ &= \Phi^T DT^{(\lambda)}(\Phi\theta) - \Phi^T D\Phi\theta.\end{aligned}$$

A matriz $\Phi^T D\Phi$ é inversível pois D é uma matriz diagonal com elementos diagonais positivos e a matriz Φ assume-se ter rank completo. Daí, o sistema $C\theta + d = 0$ pode ser escrito por

$$\Phi(\Phi^T D\Phi)^{-1}\Phi^T DT^{(\lambda)}(\Phi\theta) = \Phi\theta,$$

ou

$$\Pi T^{(\lambda)}(\Phi\theta) = \Phi\theta,$$

sendo Π a matriz definida por $\Pi = \Phi(\Phi^T D\Phi)^{-1}\Phi^T D$.

O próximo lema fornece uma interpretação da matriz Π como uma projeção sobre o espaço de todos os vetores da forma $\Phi\theta$, com respeito à norma $\|\cdot\|_D$.

Lema D.0.5. *Suponha que a matriz Φ tenha rank coluna completo, isto é, as funções $\{\varphi_j \mid j = 1, \ldots, n_\theta\}$ são linearmente independentes, e seja $\Pi = \Phi(\Phi^T D\Phi)^{-1}\Phi^T D$. Então, para cada vetor V, tem-se*

$$\|\Pi V - V\|_D = \min_\theta \|\Phi\theta - V\|_D.$$

Demonstração:

Minimiza-se a função objetivo $\|\Phi\theta - V\|_D^2$, com respeito a θ, escrevendo-a na forma $\theta^T\Phi^T D\Phi\theta - 2\theta^T\Phi^T DV + V^T DV$. Toma-se o gradiente, e estabelece-o igual à zero, para obter que a solução ótima $\Phi\theta$ é dada por ΠV.

Apêndice E

Detalhes do Custo Computacional dos Algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR

Sejam α_1 e α_2 dois escalares quaisquer em $\Re$. As notações $op_\alpha^{(+)}$, $op_\alpha^{(\cdot)}$ e $op_\alpha^{(\div)}$ serão usadas para denotar uma operação de adição $\alpha_1 + \alpha_2$, multiplicação $\alpha_1\alpha_2$ e divisão α_1/α_2, respectivamente. Cada uma destas operações equivale à um *flop* (medida de complexidade aritmética adotada).

As Tabelas E.1 e E.2 apresentam algumas operações básicas de vetores e matrizes que aparecem nas etapas de avaliação e melhoria de política dos algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR. Estas tabelas também fornecem o número de *flops* para cada uma das operações básicas.

Observações:

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

2) Dada uma matriz M com $M \in \Re^{m\times m}$ inversível, a obtenção da inversa de M, M^{-1}, por meio de transformações de Gauss (processo de 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 $op_{M_{DP}}^{(-1)}$.

Tabela E.1: Operações básicas com vetores em $\Re^{n_\theta}$

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
$op_{v,\alpha}^{(\cdot)}$	Multiplicação de um escalar α por um vetor: αv	n_θ

Tabela E.2: Operações básicas com matrizes em $\Re^{n_\theta \times n_\theta}$

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 por um vetor	$2n_\theta^2 - n_\theta$
$op_{M_{Triang},v}^{(\cdot)}$	Multiplicação de uma matriz triangular por um vetor	n_θ^2
$op_{M_{Diag},v}^{(\cdot)}$	Multiplicação de uma matriz diagonal por um vetor	n_θ

Etapa de Avaliação de Política

Na descrição do custo computacional em uma iteração i de avaliação de política (Algoritmo 5) realizada pelo RLS convencional (sem fatoração UDU^T), tem-se:

1) Construção do vetor de funções de base $\phi_k = \overline{x}_k - \gamma \overline{x}_{k+1}$ via produto de *Kronecker* dos estados.

Considerando que a formação de cada vetor $\overline{x}_k$ envolve $n_\theta[op_\alpha^{(\cdot)}]$, o que equivale à n_θ flops, tem-se

$2n_\theta[op_\alpha^{(\cdot)}] + op_{v,\alpha}^{(\cdot)} + op_v^{(+)} \Rightarrow$

$2n_\theta + n_\theta + n_\theta = 4n_\theta \; flops$

2) Cálculo do ganho $L_j(i)$ de Eq.(5.64):

$op_v^{dot} + 2op_{M,v}^{(\cdot)} + op_\alpha^{(+)} + op_{v,\alpha}^{(\cdot)} + op_\alpha^{(\div)} \Rightarrow$

$(2n_\theta - 1) + (4n_\theta^2 - 2n_\theta) + 1 + n_\theta + 1 = 4n_\theta^2 + n_\theta + 1 \; flops$

3) Atualização do vetor de parâmetros $\widehat{\theta}_j(i)$ de Eq.(5.65):

$op_v^{dot} + op_\alpha^{(+)} + op_{v,\alpha}^{(\cdot)} + op_v^{(+)} \Rightarrow$

$(2n_\theta - 1) + 1 + n_\theta + n_\theta = 4n_\theta \; flops$

4) Atualização da matriz de covariância $\Gamma_j(i)$ de Eq.(5.66):

$op_v^{out} + op_M^{(\cdot)} + op_M^{(+)} + op_{M,\alpha}^{(\cdot)} \Rightarrow$

$n_\theta^2 + (2n_\theta^3 - n_\theta^2) + n_\theta^2 + n_\theta^2 = 2n_\theta^3 + 2n_\theta^2 \; flops$

O custo computacional associado com a recuperação da matriz P a partir do vetor $\widehat{\theta}_j$ gerado pelo RLS convencional é dado por:

$(n^2 - n)op_\alpha^{(\div)} \Rightarrow n^2 - n \; flops$

Agora, será investigado o custo computacional de uma iteração i de avaliação de política realizada pelo RLS-UDU^T (Algoritmo 6). Considere os seguintes passos:

1) Construção do vetor de funções de base $\phi_k = \overline{x}_k - \gamma \overline{x}_{k+1}$

Segue-se o mesmo procedimento do algoritmo RLS$_\mu$-HDP-DLQR, o qual requer $4n_\theta \; flops$ para formação do vetor ϕ_k.

2) Cálculo do erro de estimação de Eq.(5.69)

$op_v^{dot} + op_\alpha^{(+)} \Rightarrow$

$(2n_\theta - 1) + 1 = 2n_\theta \; flops$

3) Cálculo do vetor e de Eq.(5.70)

$op^{(\cdot)}_{M_{Triang},v} \Rightarrow n_\theta^2 \; flops$

4) Cálculo do vetor g de Eq.(5.71)

$op^{(\cdot)}_{M_{Diag},v} \Rightarrow n_\theta \; flops$

5) A Eq.(5.73) requer $n_\theta[op^{(+)}_\alpha + op^{(\cdot)}_\alpha]$, o que implica $2n_\theta$ *flops*. A Eq.(5.74) envolve $n_\theta[2op^{(\cdot)}_\alpha + op^{(\div)}_\alpha]$, que é equivalente à $3n_\theta$ *flops*. A Eq.(5.76) envolve $n_\theta[op^{(\div)}_\alpha]$, o que corresponde à n_θ *flops*. Neste caso, deve-se observar que estas equações são executadas n_θ vezes a cada iteração i, conforme indica a contabilidade do laço $l = 1 : n_\theta$. Desta forma, tem-se um total de $6n_\theta$ *flops* para a atualização dos escalares β, d e τ.

6) O custo total para 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

$$\sum_{l=2}^{n_\theta} 4(l-1) = 2n_\theta^2 - 2n_\theta,$$

considerando que no laço $m = 1 : l-1$, com $l \neq 1$, cada equação requer $(l-1)[op^{(+)}_\alpha + op^{(\cdot)}_\alpha]$, o que corresponde à $2(l-1)$ *flops*.

7) Atualização do vetor de ganho $L_j(i)$ de Eq.(5.79)

$op^{(\div)}_\alpha + op^{(\cdot)}_{v,\alpha} \Rightarrow 1 + n_\theta \; flops$

8) Atualização do vetor de parâmetros $\widehat{\theta}_j(i)$ de Eq.(5.80)

$op^{(+)}_v + op^{(\cdot)}_{v,\alpha} \Rightarrow 2n_\theta \; flops$

Etapa de Melhoria de Política

Na descrição do custo computacional em uma iteração j da etapa de melhoria de política (Algoritmos 5 e 6), tem-se:

$3op^{(*)}_{M_{nennn}} + op^{(*)}_{M_{nennne}} + op^{(*)}_{M_{nenen}} + 2op^{(\cdot)}_{M,\alpha} + op^{(+)}_M + op^{(-1)}_{M_{DP}} \Rightarrow$

$$3(2n_e n^2 - n_e n) + (2n_e^2 n - n_e^2) + (2n_e^2 n - n_e n) + 2n_e^2 + n_e^2 + \tfrac{1}{3}n_e^3 =$$

$$\tfrac{1}{3}n_e^3 + 6n_e n^2 + 4n_e^2 n + 2n_e^2 - 4n_e n \; flops$$

Apêndice F

Modelo do Sistema Dinâmico de Quarta Ordem

A descrição em espaço de estados correspondente ao modelo de Eqs.(6.1)-(6.4) é dada por

$$\dot{x} = Ax + Bu, \qquad t \geq t_0. \tag{F.1}$$

sendo A a matriz de estado,

$$A = \begin{bmatrix} -1/C_{cap}R_1 & 0 & 1/C_{cap} & 0 \\ 0 & -1/C_{cap}R_1 & 0 & -1/C_{cap} \\ -1/L_{ind} & 0 & -R_2/L_{ind} & -R_2/L_{ind} \\ 0 & 1/L_{ind} & -R_2/L_{ind} & -R_2/L_{ind} \end{bmatrix}. \tag{F.2}$$

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

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1/L_{ind} & 0 \\ 0 & 1/L_{ind} \end{bmatrix} \tag{F.3}$$

e

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{F.4}$$

As matrizes do sistema dinâmico para os valores dos parâmetros $L = 10$, $C = 10^{-1}$, $R_1 = 10^4$ e $R_2 = 10^2$ são dadas por

$$A_c = \begin{bmatrix} -0.001 & 0 & 10 & 0 \\ 0 & -0.001 & 0 & -10 \\ -0.1 & 0 & -10 & -10 \\ 0 & 0.1 & -10 & -10 \end{bmatrix} \tag{F.5}$$

e

$$B_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}. \tag{F.6}$$

As matrizes do sistema discretizado para o intervalo de amostragem $T_{amost} = 0.1$ são obtidas pelo método de discretização *Impulse-invariant*.

F.1 Modelo com Variações Paramétricas na Planta

As matrizes do sistema com variações paramétricas para o modelo (6.9) são dadas por

$$A = \begin{bmatrix} (-1/C_{cap}R_1)(1+p(t))^{-2} & 0 & (1/C_{cap})(1+p(t))^{-1} & 0 \\ 0 & (-1/C_{cap}R_1)(1+p(t))^{-2} & 0 & (-1/C_{cap})(1+p(t))^{-1} \\ (-1/L_{ind})(1+p(t))^{-1} & 0 & -R_2/L_{ind} & -R_2/L_{ind} \\ 0 & (1/L_{ind})(1+p(t))^{-1} & -R_2/L_{ind} & -R_2/L_{ind} \end{bmatrix} \tag{F.7}$$

e

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ (1/L_{ind})(1+p(t))^{-1} & 0 \\ 0 & (1/L_{ind})(1+p(t))^{-1} \end{bmatrix}. \tag{F.8}$$

Apêndice G

Modelo do Sistema Dinâmico de Terceira Ordem e Setup

As matrizes do sistema contínuo são dadas por

$$A_c = \begin{bmatrix} -1.102 & 0.905 & -0.002 \\ 4.064 & -0.770 & -0.169 \\ 0.000 & 0.000 & -10.000 \end{bmatrix} \tag{G.1}$$

e

$$B_c = \begin{bmatrix} 0.000 & 0.000 \\ 0.000 & 10.000 \\ 10.000 & 0.000 \end{bmatrix}. \tag{G.2}$$

O intervalo de amostragem é $T_{amost} = 0.1$ para o sistema de terceira ordem. As matrizes do sistema amostrado são dadas por

$$A_d = \begin{bmatrix} 0.912 & 0.083 & -0.001 \\ 0.372 & 0.943 & -0.010 \\ 0.000 & 0.000 & 0.368 \end{bmatrix} \tag{G.3}$$

e

$$B_d = \begin{bmatrix} -0.000 & 0.043 \\ -0.006 & 0.968 \\ 0.632 & 0.000 \end{bmatrix}. \tag{G.4}$$

G.1 Setup do processo iterativo

As formulações gerais do setup do processo iterativo para RLS-TD(λ) são dadas por: ordem do sistema n ($n = 3$), intervalo de amostragem T_{amost} ($T_{amost} = 0.1$s), intervalo de revitalização n_{revit} ($n_{revit} = 12$).

As condições iniciais e revitalização para o sistema de terceira ordem são $x_0 = [4 \quad 2 \quad 5]^T$ e $x_{revit} = [7 \quad 2 \quad -5]^T$, respectivamente. As matrizes de ponderação Q e R da função de utilidade são selecionadas como matrizes de identidade.

Conforme o processo de vetorização de Eq.(5.55), o vetor de parâmetros θ é dado por $[\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6]^T$. A ordem deste vetor é estabelecida a partir da dimensão da matriz P. Para a sistema de ordem 3, tem-se que $\theta \in R^6$ e para os resultados apresentados, tem-se que $\theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta_5 = \theta_6 = 0$.

A matriz inicial da equação de *Riccati*/*Lyapunov* está associada com os parâmetros da aproximação para o sistema de ordem 3, segundo a lei de formação da vetorização de matrizes simétricas de Eq.(5.55). A matriz é dada por

$$P_0^\theta = \begin{bmatrix} \theta_1 & \theta_2/2 & \theta_3/2 \\ & \theta_4 & \theta_5/2 \\ & & \theta_6 \end{bmatrix}. \tag{G.5}$$

G.2 Solução DLQR-Schur

A solução computacional da HJB-*Riccati* pelo método de *Schur* é dada por

$$P_{schur} = \begin{bmatrix} 5.3947 & 0.7427 & -0.0078 \\ 0.7427 & 1.6203 & -0.0067 \\ -0.0078 & -0.0067 & 1.1037 \end{bmatrix}. \tag{G.6}$$

Consequentemente, a política ótima é dada por

$$K_{schur} = \begin{bmatrix} 0.0050 & 0.0039 & -0.1782 \\ -0.5632 & -0.6129 & 0.0066 \end{bmatrix}. \tag{G.7}$$

Apêndice H

Experimentos Complementares 1

Para efeito de análise do desempenho dos algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-UDU^T-HDP-DLQR do ponto de vista de sistemas de controle, mostra-se a seguir a trajetória dos estados x_k e o sinal de controle u_k que foi aplicado ao sistema durante a sintonia da HDP para o modelo de quarta ordem (circuito elétrico), como também são ilustrados os comportamentos dos traços da matriz de malha fechada e os autovalores associados. Os comportamentos são referentes as simulações apresentadas na Seção 6.2.7 que levam em consideração variações paramétricas na planta. Em termos do objetivo de controle, os traços da matriz de malha fechada e os autovalores associados mostram oscilações que representam o comportamento de estabilidade do sistema dinâmico durante o processo de busca da política de controle ótima. Os traços podem ser vistos como uma boa medida para avaliar a convergência do algoritmo sem calcular os autovalores do sistema de malha fechada, evitando, desta forma, o cálculo dos autovalores em aplicações em tempo real devido ao seu alto custo.

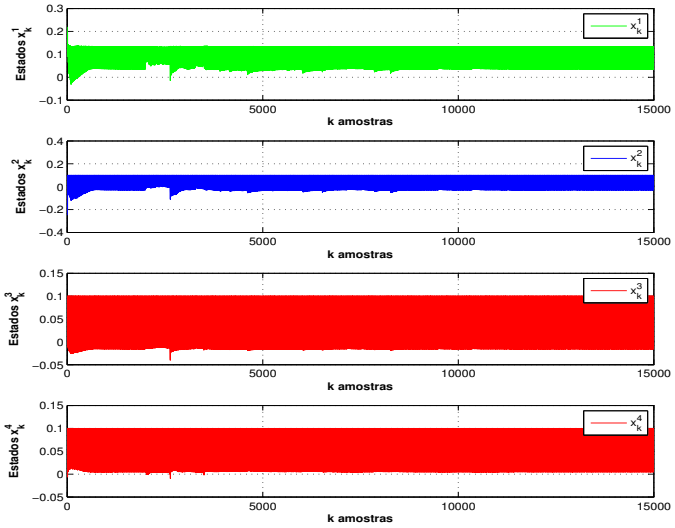

Figura H.1: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.001$ - Comportamento dos estados com revitalização para o modelo de quarta ordem para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$. Os estados são inicializados com $x_0 = [0.22 \quad -0.25 \quad -0.005 \quad 0.005]^T$ e a revitalização com $x_{revit} = [0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

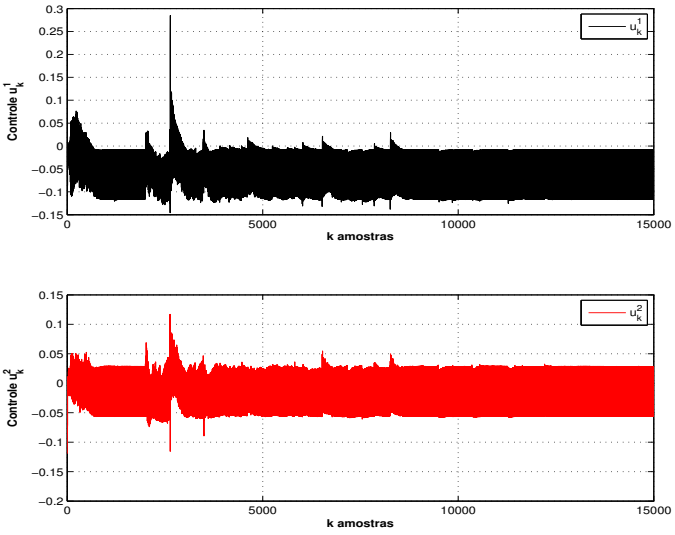

Figura H.2: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.001$ - Comportamento do controlador com revitalização de estado para o modelo de quarta ordem para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

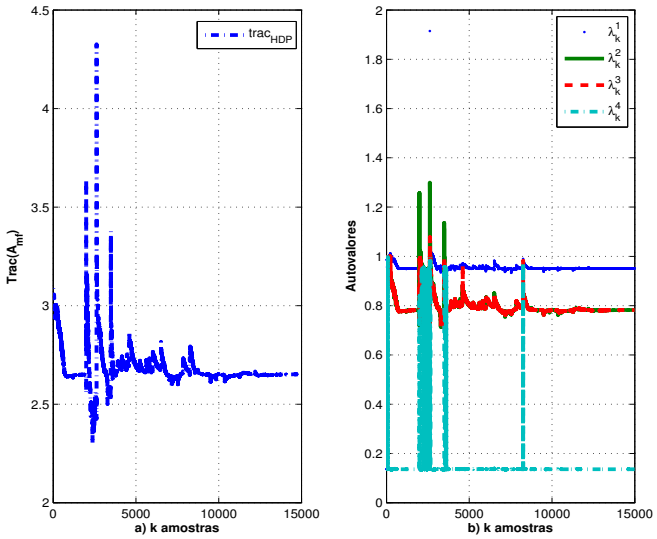

Figura H.3: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.001$ - O comportamento dos traços e os autovalores associados do modelo de malha fechada para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

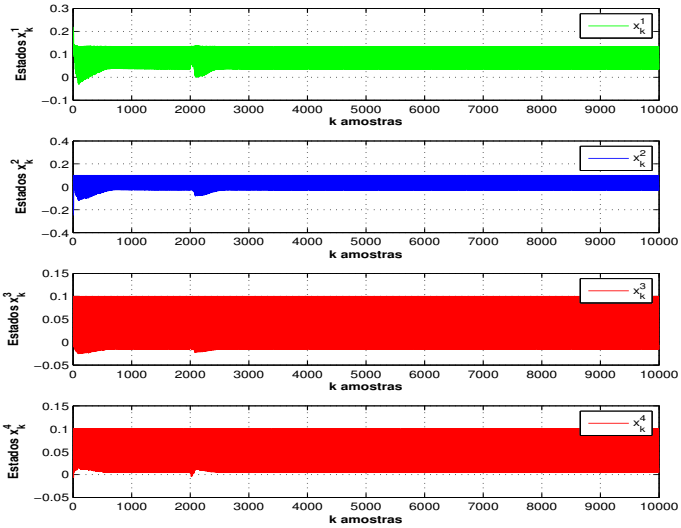

Figura H.4: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10^{13}$ - Comportamento dos estados com revitalização para o modelo de quarta ordem para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$. Os estados são inicializados com $x_0 = [0.22 \quad -0.25 \quad -0.005 \quad 0.005]^T$ e a revitalização com $x_{revit} = [0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

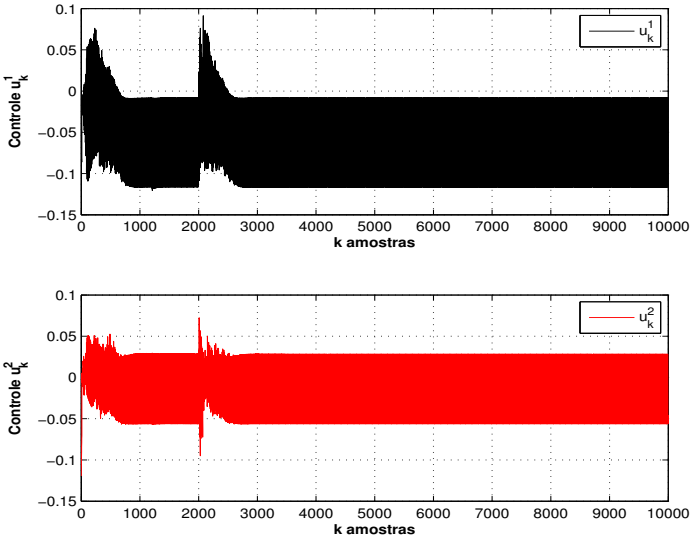

Figura H.5: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10^{13}$ - Comportamento do controlador com revitalização de estado para o modelo de quarta ordem para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

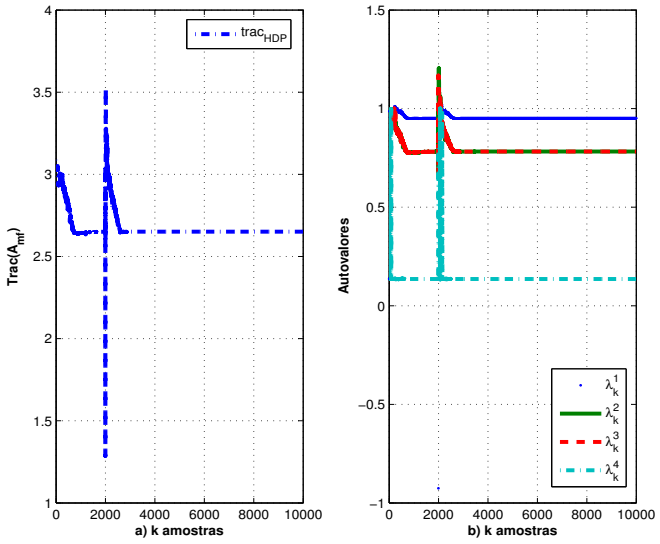

Figura H.6: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10^{13}$ - O comportamento dos traços e os autovalores associados do modelo de malha fechada para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

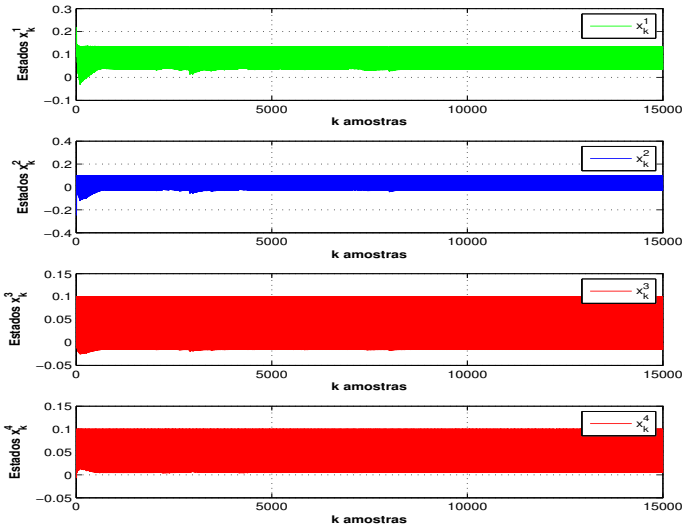

Figura H.7: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Comportamento dos estados com revitalização para o modelo de quarta ordem para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$. Os estados são inicializados com $x_0 = [0.22 \quad -0.25 \quad -0.005 \quad 0.005]^T$ e a revitalização com $x_{revit} = [0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

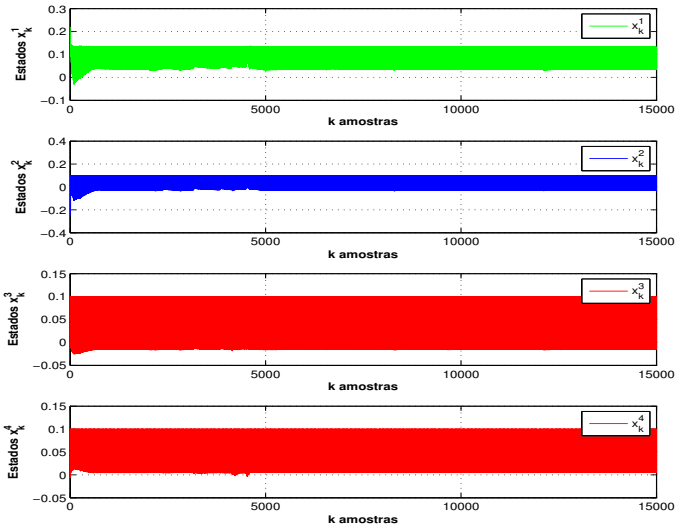

Figura H.8: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Comportamento dos estados com revitalização para o modelo de quarta ordem para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$. Os estados são inicializados com $x_0 = [0.22 \quad -0.25 \quad -0.005 \quad 0.005]^T$ e a revitalização com $x_{revit} = [0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

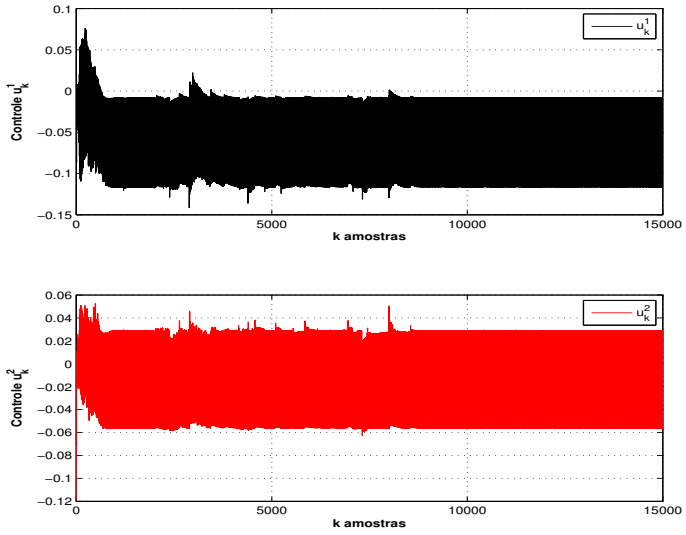

Figura H.9: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Comportamento do controlador com revitalização de estado para o modelo de quarta ordem para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

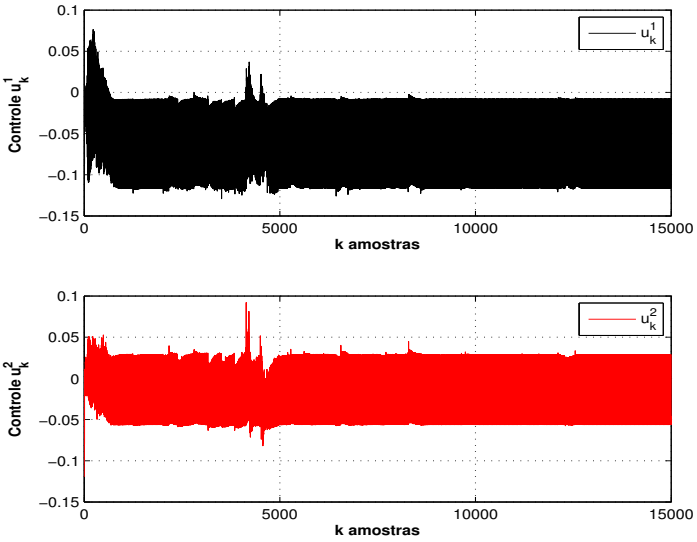

Figura H.10: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - Comportamento do controlador com revitalização de estado para o modelo de quarta ordem para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

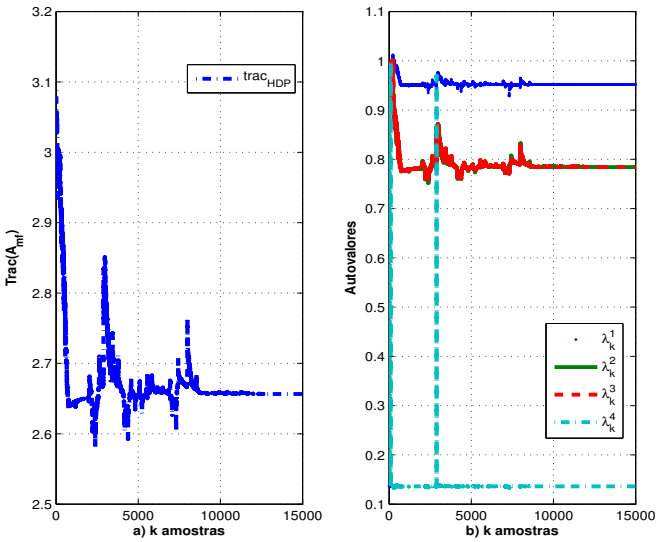

Figura H.11: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - O comportamento dos traços e os autovalores associados do modelo de malha fechada para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

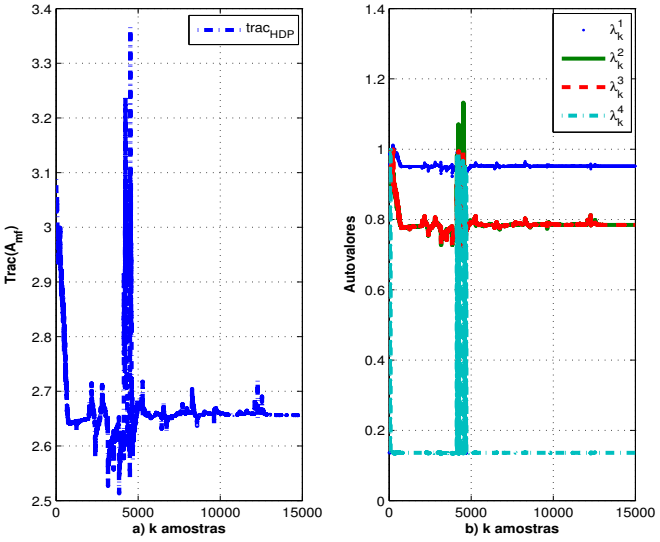

Figura H.12: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 0.001$ - O comportamento dos traços e os autovalores associados do modelo de malha fechada para um ciclo de 15000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

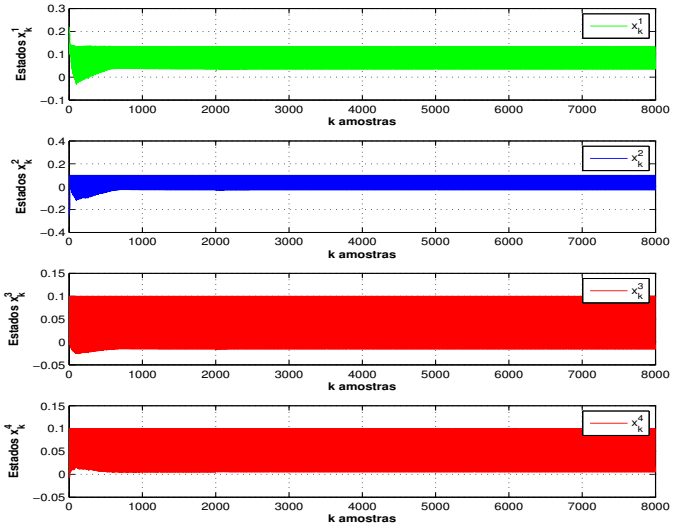

Figura H.13: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Comportamento dos estados com revitalização para o modelo de quarta ordem para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$. Os estados são inicializados com $x_0 = [0.22 \quad -0.25 \quad -0.005 \quad 0.005]^T$ e a revitalização com $x_{revit} = [0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

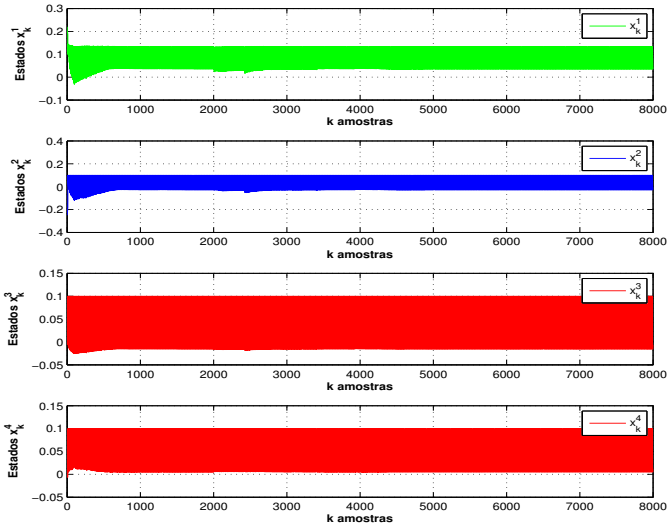

Figura H.14: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Comportamento dos estados com revitalização para o modelo de quarta ordem para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$. Os estados são inicializados com $x_0 = [0.22 \quad -0.25 \quad -0.005 \quad 0.005]^T$ e a revitalização com $x_{revit} = [0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

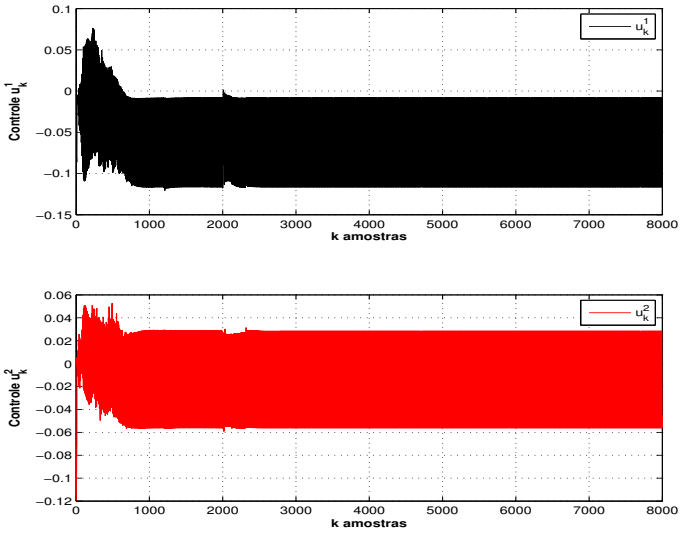

Figura H.15: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Comportamento do controlador com revitalização de estado para o modelo de quarta ordem para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

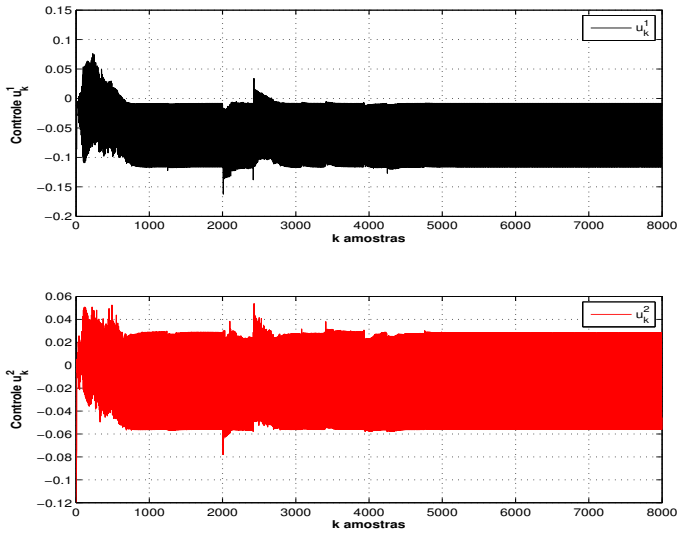

Figura H.16: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - Comportamento do controlador com revitalização de estado para o modelo de quarta ordem para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

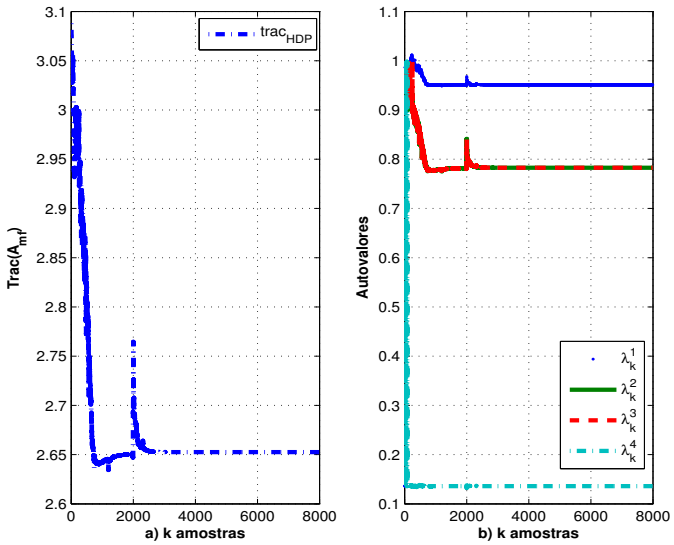

Figura H.17: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - O comportamento dos traços e os autovalores associados do modelo de malha fechada para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

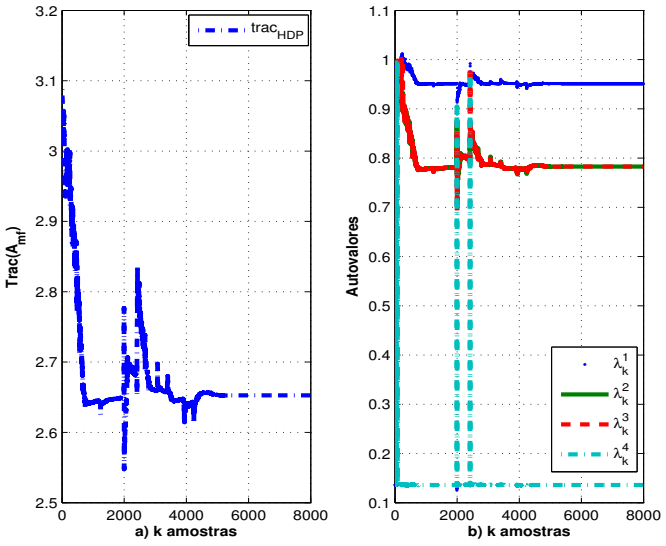

Figura H.18: Variação paramétrica do tipo p_2 com $\alpha = 0.25$ e $\beta = 10^{13}$ - O comportamento dos traços e os autovalores associados do modelo de malha fechada para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

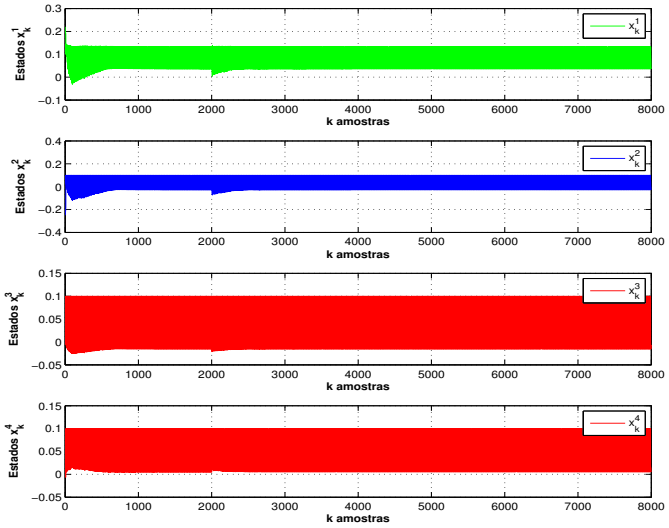

Figura H.19: Variação paramétrica do tipo p_2 com $\alpha = 1,5$ e $\beta = \infty$ - Comportamento dos estados com revitalização para o modelo de quarta ordem para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$. Os estados são inicializados com $x_0 = [0.22 \quad -0.25 \quad -0.005 \quad 0.005]^T$ e a revitalização com $x_{revit} = [0.1 \quad 0.1 \quad 0.1 \quad 0.1]^T$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

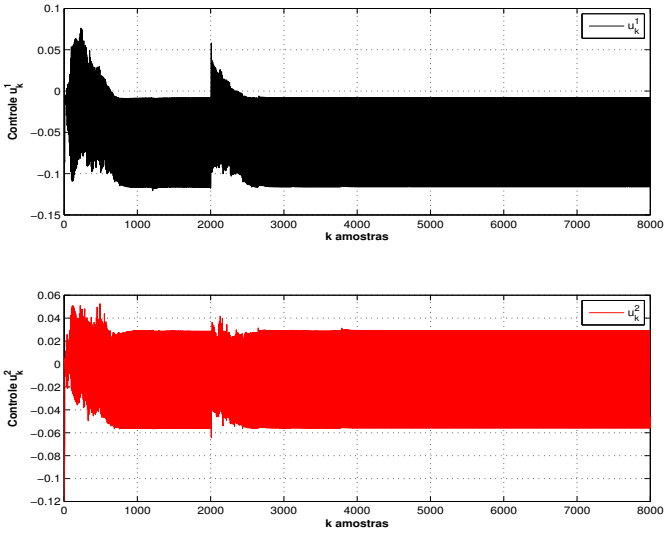

Figura H.20: Variação paramétrica do tipo p_2 com $\alpha = 1,5$ e $\beta = \infty$ - Comportamento do controlador com revitalização de estado para o modelo de quarta ordem para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

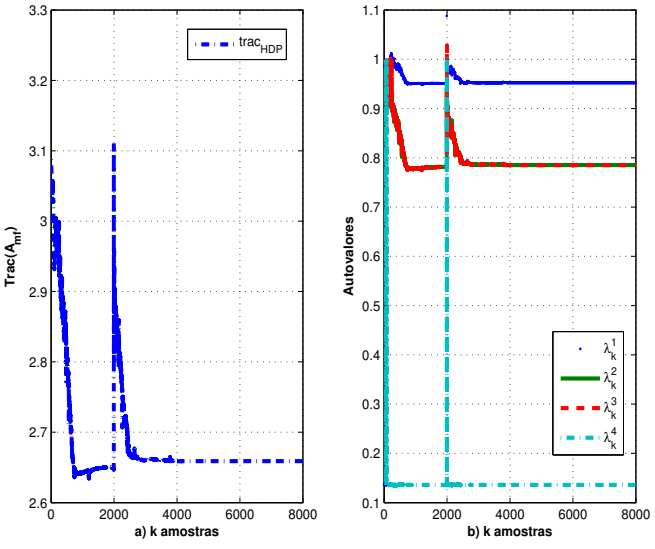

Figura H.21: Variação paramétrica do tipo p_2 com $\alpha = 1,5$ e $\beta = \infty$ - O comportamento dos traços e os autovalores associados do modelo de malha fechada para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Apêndice I

Experimentos Complementares 2

Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.1$

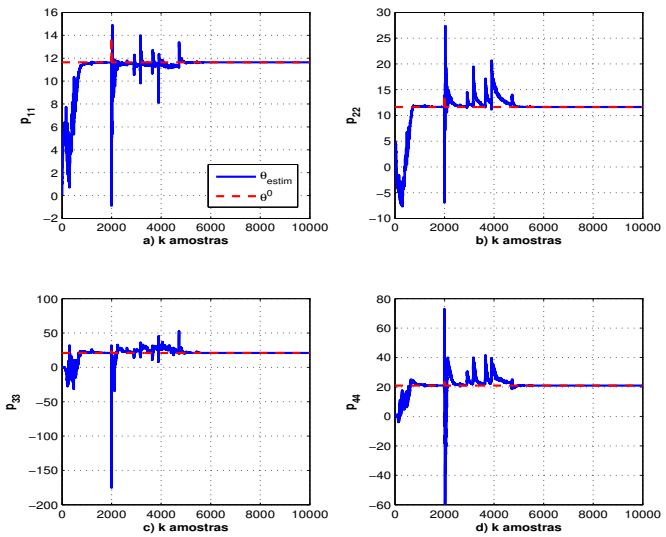

Figura I.1: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

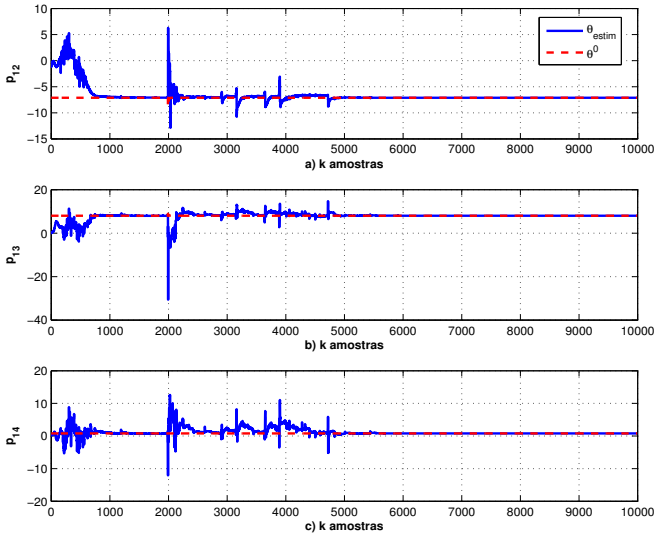

Figura I.2: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

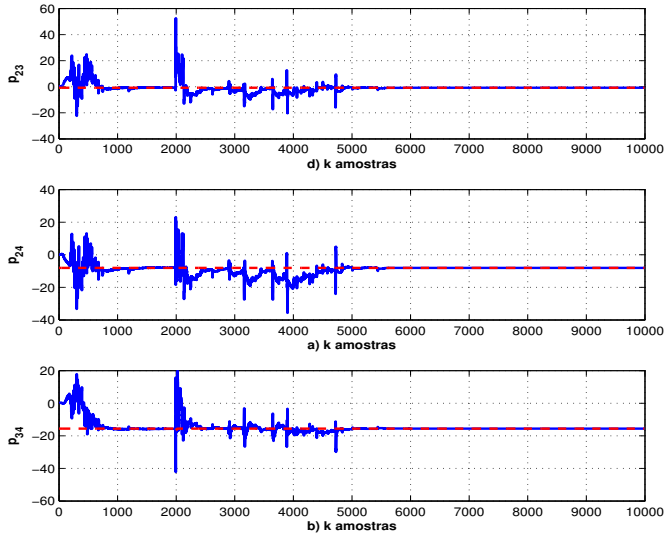

Figura I.3: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

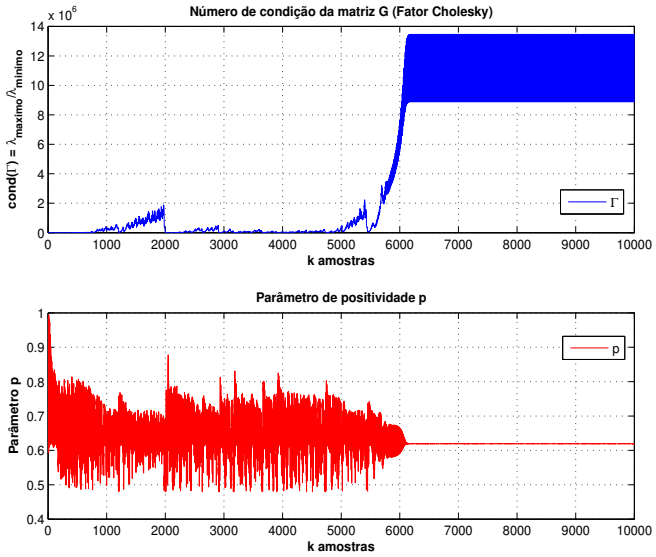

Figura I.4: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 0.1$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$

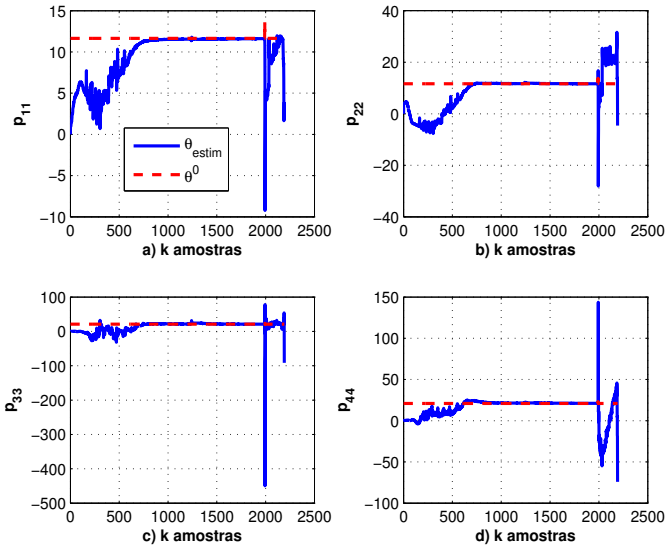

Figura I.5: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 2200 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

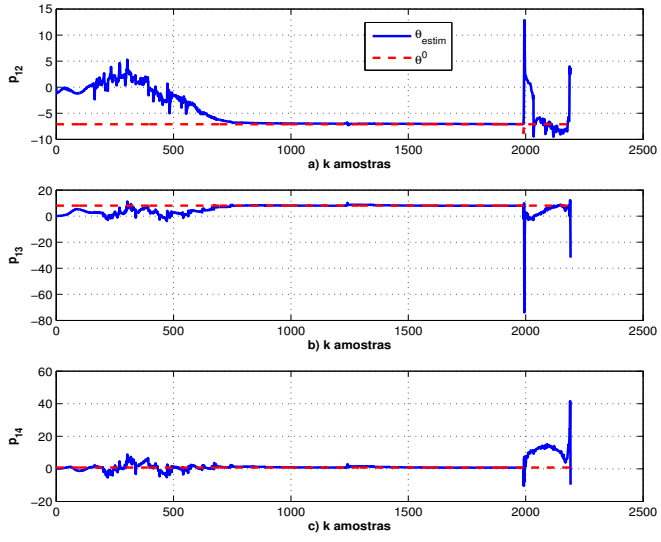

Figura I.6: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 2200 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

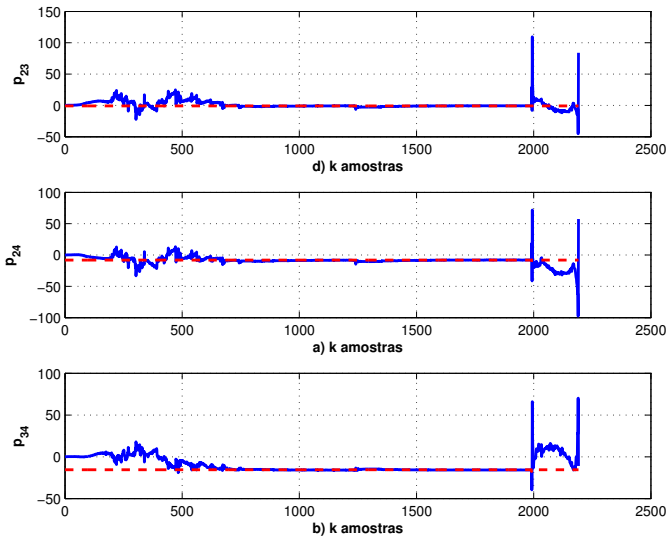

Figura I.7: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 2200 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

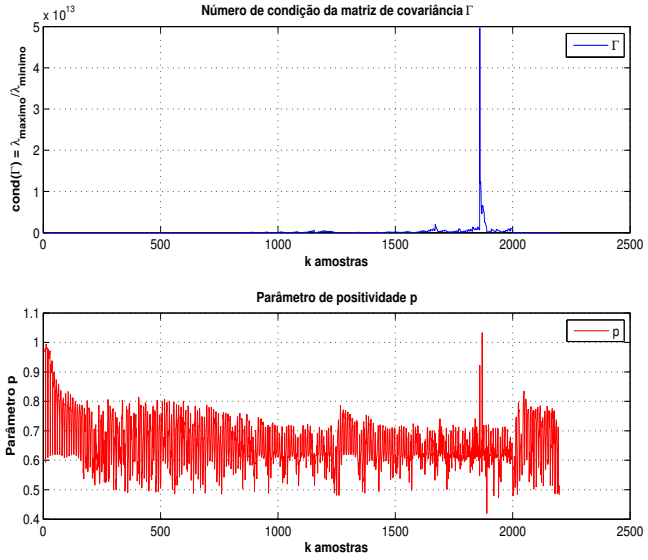

Figura I.8: Variação paramétrica do tipo p_1 com $\alpha = 10$ e $\beta = 10$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 2200 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 1$

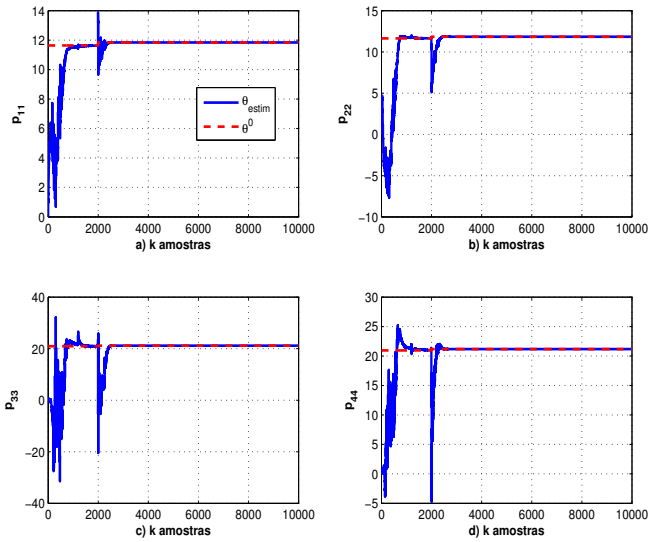

Figura I.9: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 1$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

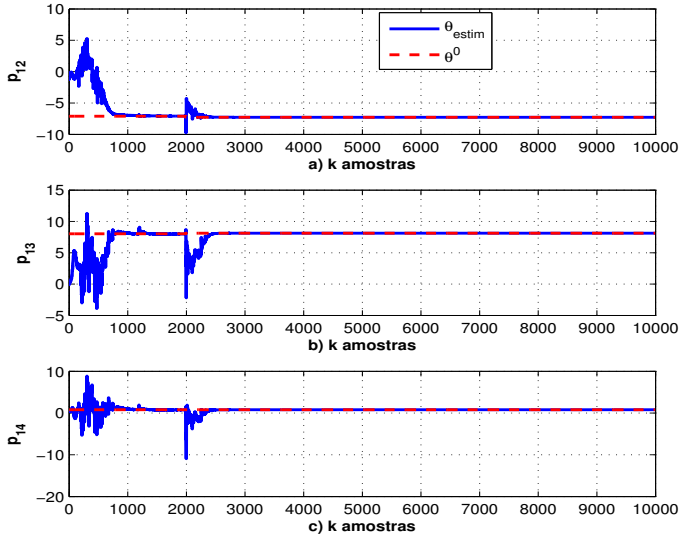

Figura I.10: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 1$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

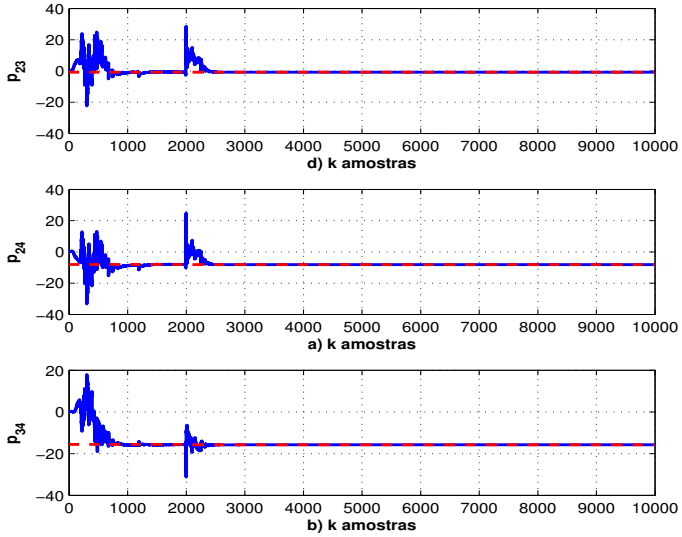

Figura I.11: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 1$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

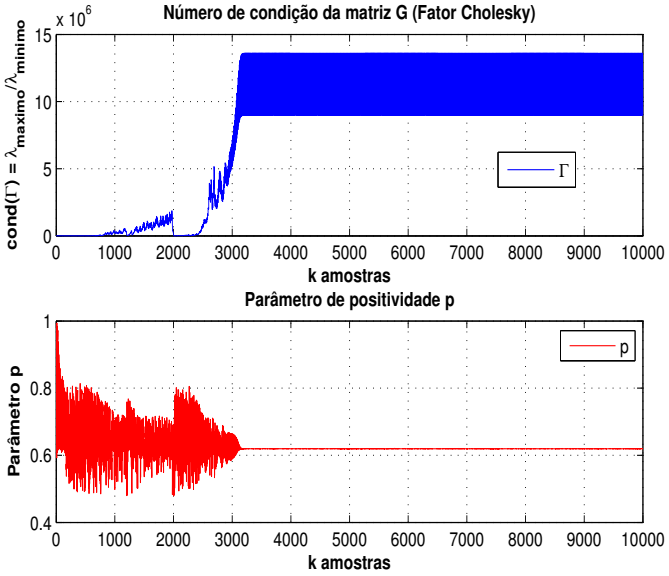

Figura I.12: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 1$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 10$

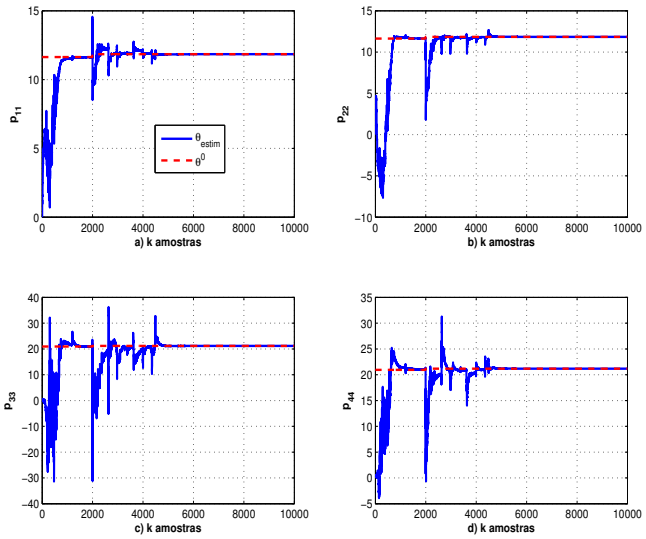

Figura I.13: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

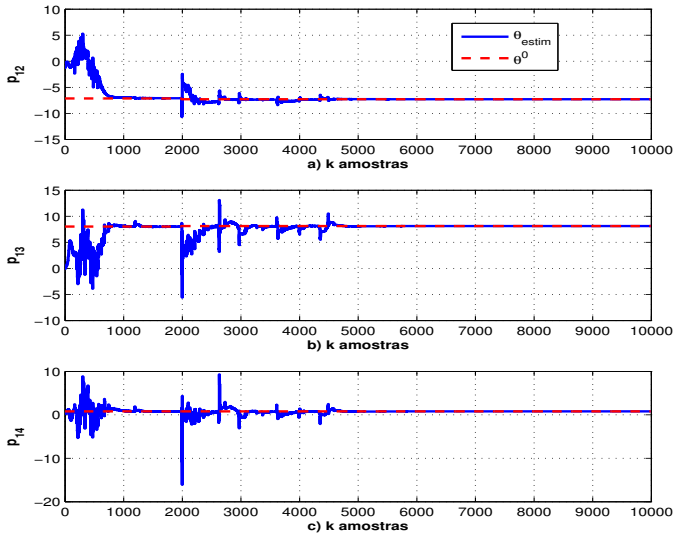

Figura I.14: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

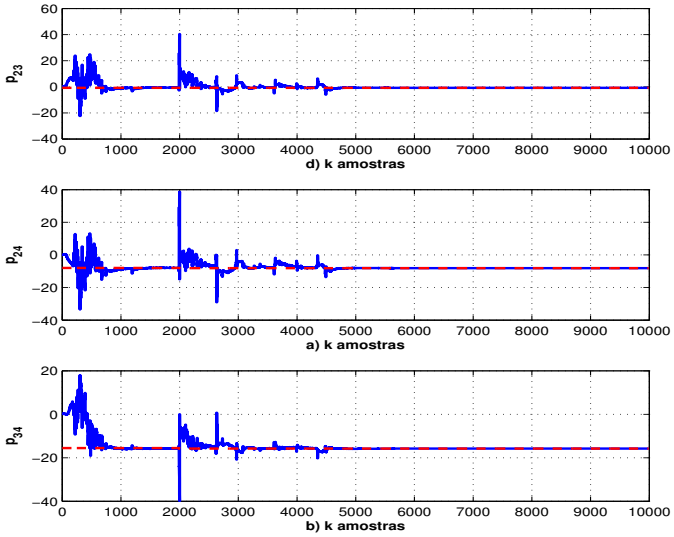

Figura I.15: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 10$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

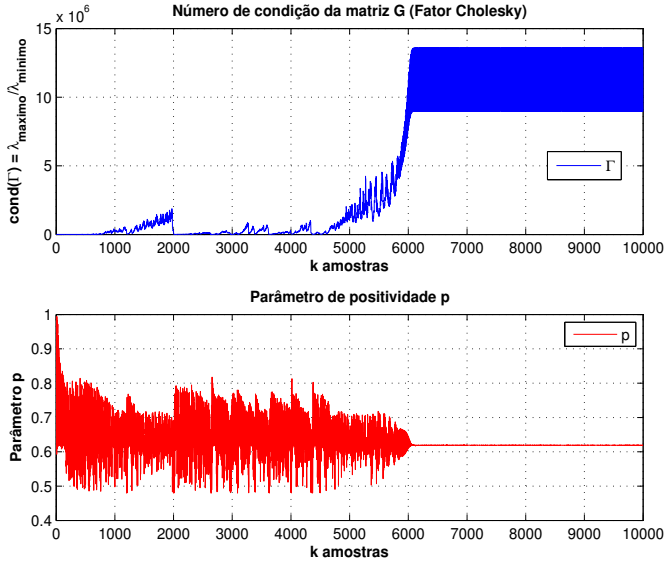

Figura I.16: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 10$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 100$

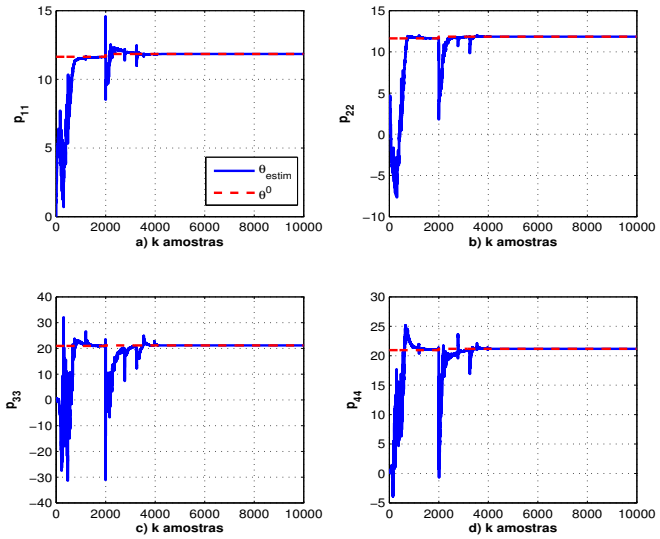

Figura I.17: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 100$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

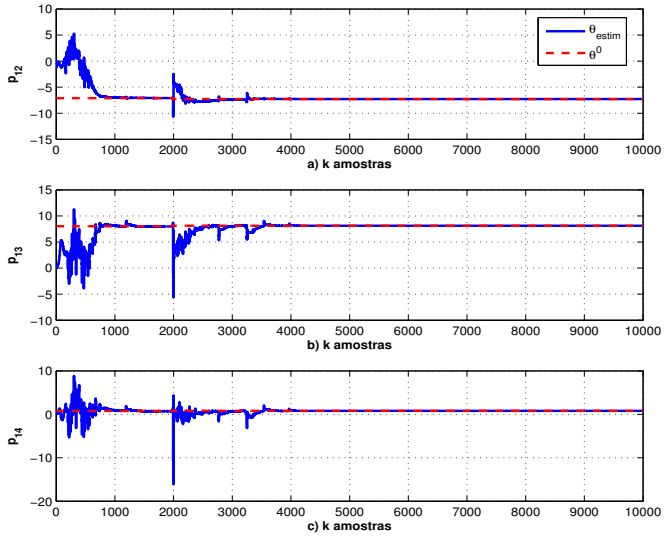

Figura I.18: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 100$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

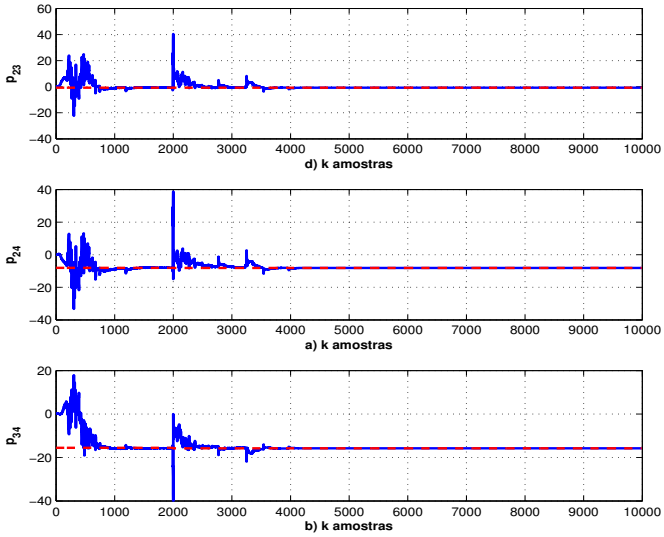

Figura I.19: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 100$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

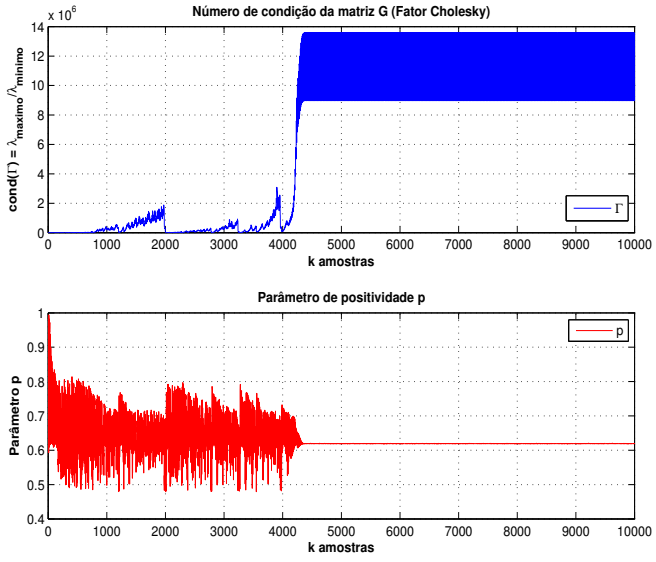

Figura I.20: Variação paramétrica do tipo p_2 com $\alpha = 1$ e $\beta = 100$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.01$

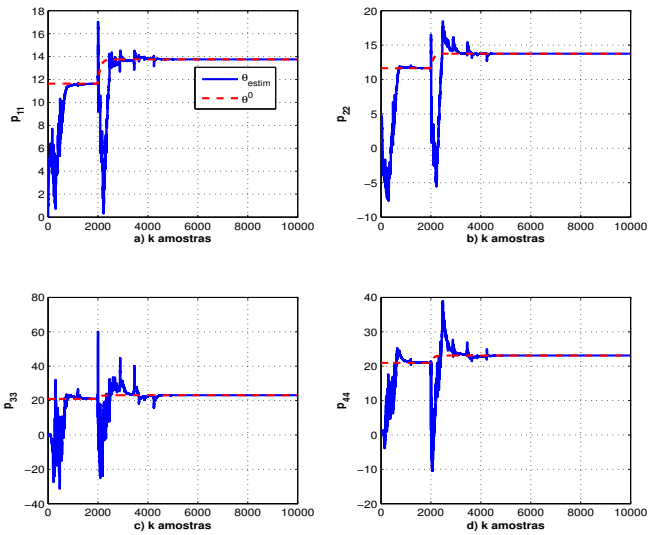

Figura I.21: Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

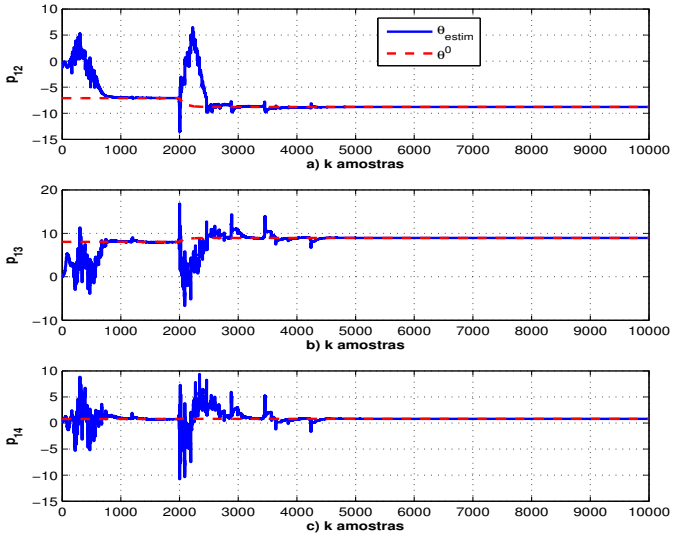

Figura I.22: Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

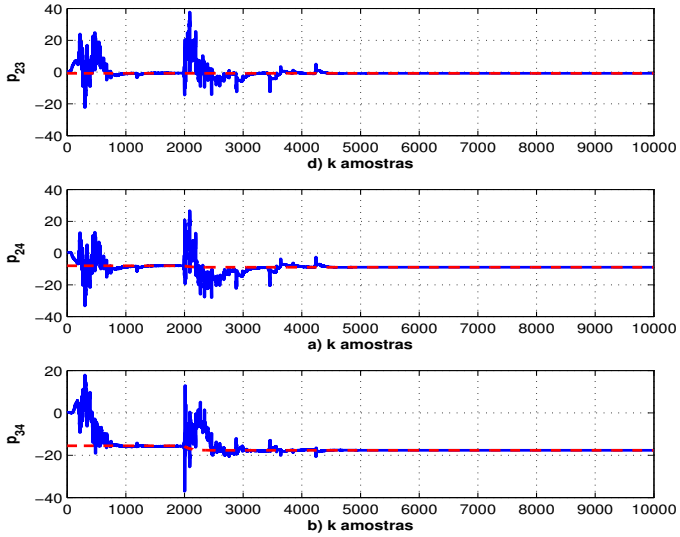

Figura I.23: Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.01$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

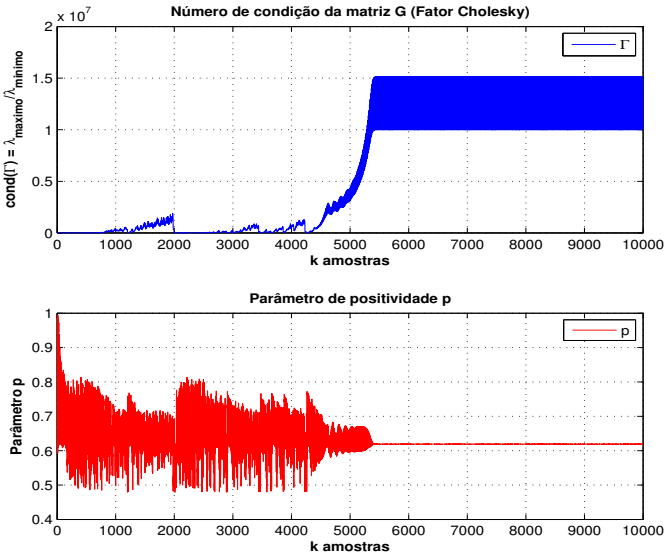

Figura I.24: Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.01$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.1$

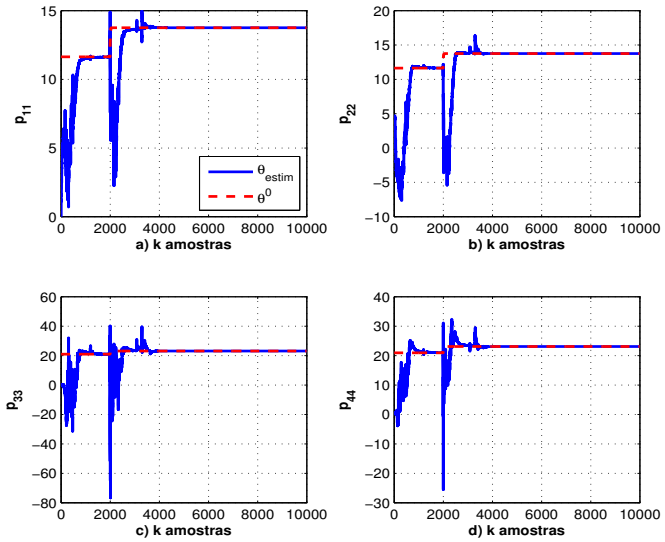

Figura I.25: Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

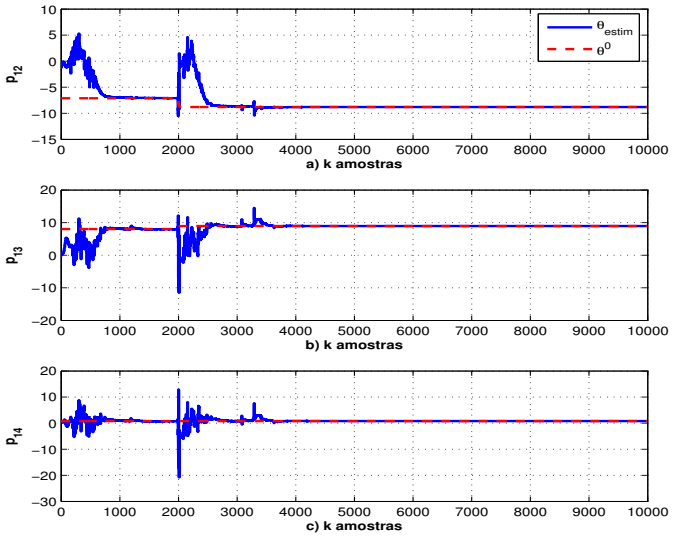

Figura I.26: Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

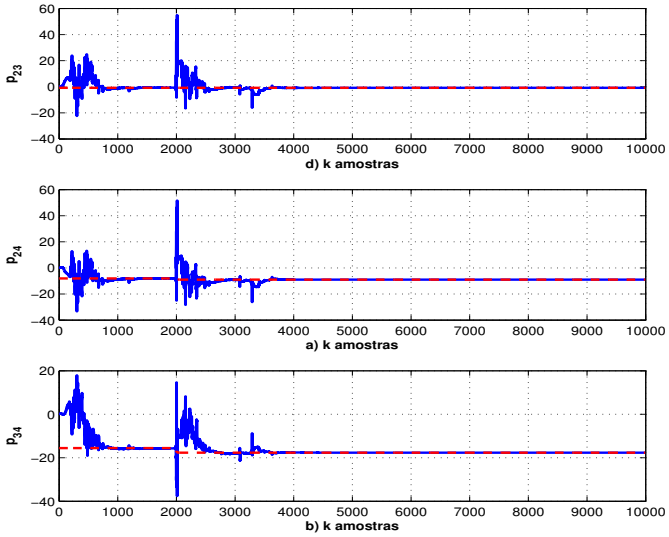

Figura I.27: Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.1$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

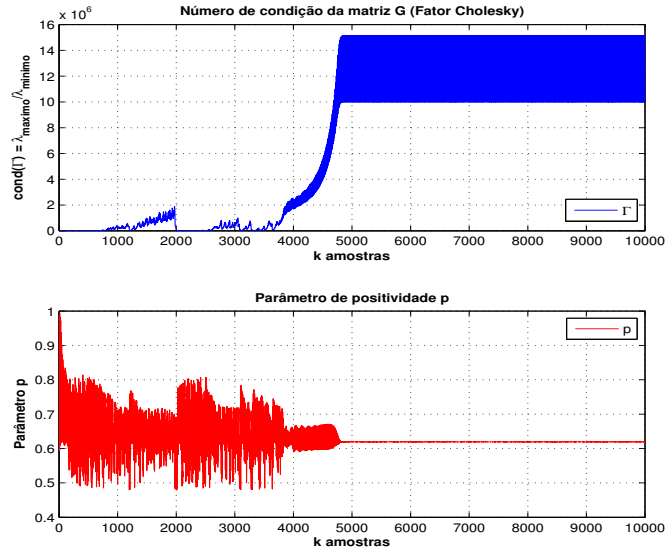

Figura I.28: Variação paramétrica do tipo p_2 com $\alpha = 10$ e $\beta = 0.1$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 10000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Variação paramétrica do tipo p_2 com $\alpha = 1,5$ e $\beta = \infty$

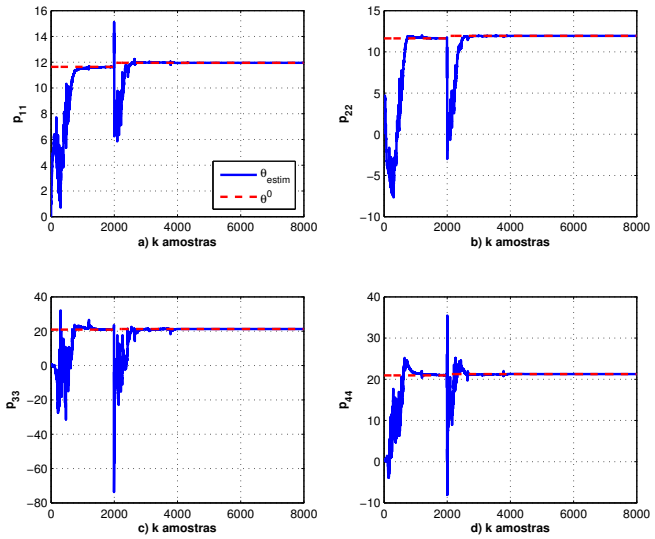

Figura I.29: Variação paramétrica do tipo p_2 com $\alpha = 1,5$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{ii} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

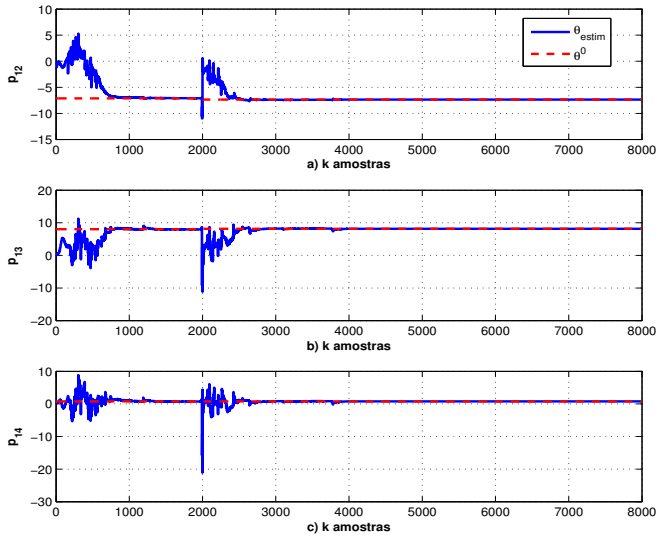

Figura I.30: Variação paramétrica do tipo p_2 com $\alpha = 1,5$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{12}, p_{13} e p_{14} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

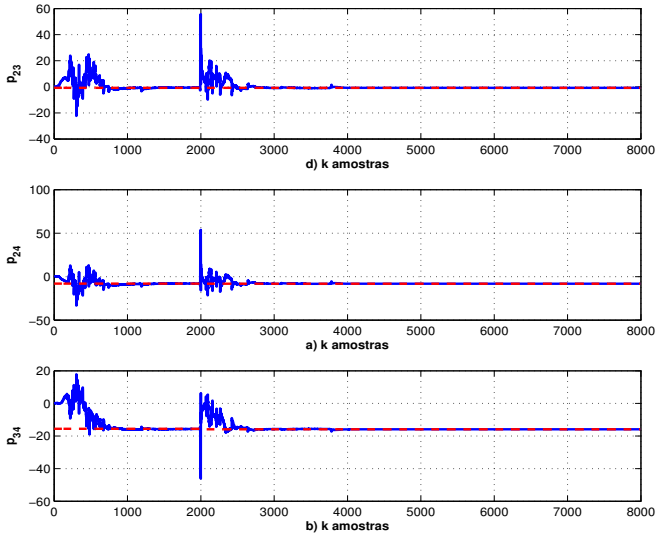

Figura I.31: Variação paramétrica do tipo p_2 com $\alpha = 1,5$ e $\beta = \infty$ - Evolução do processo iterativo para os parâmetros p_{23}, p_{24} e p_{34} para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

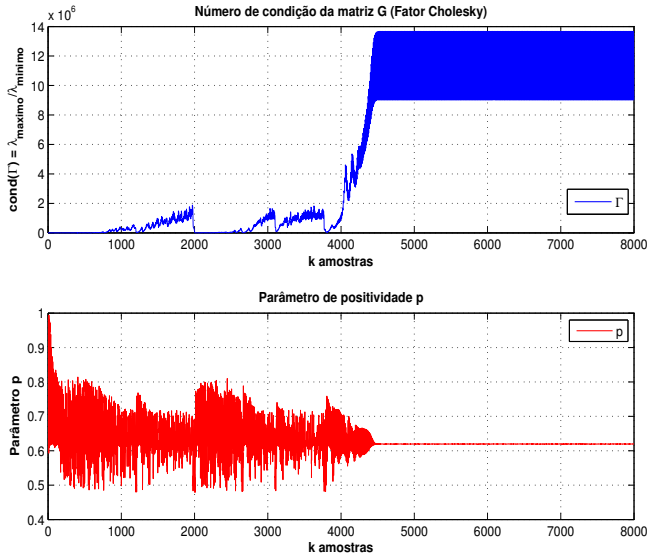

Figura I.32: Variação paramétrica do tipo p_2 com $\alpha = 1,5$ e $\beta = \infty$ - Número de condição do fator Cholesky $G(k)$ da matriz de covariância $\Gamma(k)$ e parâmetro de positividade para um ciclo de 8000 iterações, com fator de esquecimento $\mu = 0.92$ - Algoritmo RLS$_\mu$-UDU^T-HDP-DLQR Otimístico.

Referências Bibliográficas

Abu-Khalaf, M. e F.L. Lewis (2005). Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* **41**(5), 779–791.

Al-Tamimi, A. e F.L. Lewis (2007). Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof. *in Proc. IEEE Int. Symp. Approximate Dynamic Programming and Reinforcement Learning* pp. 38–43.

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

Al-Tamimi, A., F.L. Lewis e M. Abu-Khalaf (2008). Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **38**(4), 943–949.

Al-Tamimi, A., M. Abu-Khalaf e Lewis F.L. (2007*b*). Adaptive critic designs for discrete-time zero-sum games with application to h^{∞} control. *IEEE Trans. Syst., Man, Cybern. B* **37**(1), 240–247.

Al-Tamimi, A.A. (2007). Discrete-time Control Algorithms and Adaptive Intelligent Systems Designs. PhD thesis. University of Texas, Arlington.

Anderson, B.D.O. e J.B. Moore (1990). *Optimal Control: Linear Quadratic Methods*. Dover Publications, Inc.. Mineola, New York.

Anderson, C.W. (1988). Strategy learning with multilayer connectionist representations. Technical Report 87-509.3, GTE Laboratories Incorporated, Computer and Intelligent Systems Laboratory. 40 Sylvan Road, Waltham, MA.

Astrom, K.J. e B. Wittenmark (1994). *Adaptive Control*. 2nd ed.. Addison-Wesley Longman Publishing Co., Inc.. Boston, MA, USA.

Athans, M. e P.L. Falb (1966). *Optimal Control- An Introduction to the Theory and Its Apllications*. McGraw-Hill Book Company . United States of America.

Balakrishnan, S.N. e V. Biega (1996). Adaptive-critic-based neural networks for aircraft optimal control. *J. Guid. Control Dyn.* **19**, 893–898.

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

Barron, A.R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *Information Theory, IEEE Transactions on* **39**(3), 930–945.

Barron, A.R. (1994). Approximation and estimation bounds for artificial neural networks. *Mach. Learn.* **14**(1), 115–133.

Barto, A.G., R.S. Sutton e C.W. Anderson (1983). Neuron-like elements that can solve difficult learning control problems. In: *IEEE Transactions on Systems, Man, and Cybernetics*. pp. 835–846.

Barto, A.G., S.J. Bradtke e S.P. Singh (1995). Learning to act using real-time dynamic programming. Vol. 72. pp. 81–138.

Baxter, J. e P.L. Bartlett (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*.

Bellman, R.E. (1957). *Dynamic Programming*. Princeton Univ. Press. Princeton, NJ.

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

Bellman, R.E. (2003). *Dynamic Programming*. Dover Books on Computer Science Series. Dover Publications.

Bellman, R.E. e R.E. Kalaba (1965). *Dynamic Programming and Modern Control Theory*. Academic paperbacks. Academic Press.

Bellman, R.E. e S.E. Dreyfus (1962). *Applied Dynamic Programming*. Rand Corporation. Research studies. Princeton University Press.

Bertsekas, D.P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*. Vol. 2. Prentice-Hall.

Bertsekas, D.P. (1995*a*). *Dynamic Programming and Optimal Control*. Vol. 1. Athena Scientific, Belmont, Massachusetts.

Bertsekas, D.P. (1995*b*). *Dynamic Programming and Optimal Control*. Vol. 2. Athena Scientific, Belmont, Massachusetts.

Bertsekas, D.P. (2012). *Dynamic Programming and Optimal Control*. Vol. 2, 4nd ed.. Athena Scientific, Belmont, Massachusetts.

Bertsekas, D.P. e Huizhen Yu (2010). Q-learning and enhanced policy iteration in discounted dynamic programming. In: *Decision and Control (CDC), 2010 49th IEEE Conference on*. pp. 1409–1416.

Bertsekas, D.P. e J.N. Tsitsiklis (1996). *Neuro-Dynamic Programming*. Athena Scientific. Belmont, MA.

Bertsekas, D.P., M.L. Homer, D.A. Logan, S.D. Patek e N.R. Sandell (2000). Missile defense and interceptor allocation by neuro-dynamic programming. *IEEE Trans. Syst., Man, Cybern. A* **30**(1), 42–51.

Bierman, G.J. (1977). *Factorization Methods for Discrete Estimation*. Academic Press. New York.

Bottura, C.P. e J.V. Fonseca Neto (1999). Parallel genetic algorithm fitness function team for eigenstructure assignment via LQR designs. *Congress on Evolutionary Computation - CEC99. Washington, DC, USA. Vol* **2**, 1035–1042.

Boyan, J. (2002). Least-squares temporal difference learning. In: *Technical update: least-squares temporal difference learning. Machine Learning, Special Issue on Reinforcement Learning, to appear.*

Bradtke, S.J. (1994). Incremental Dynamic Programming for On-Line Adaptive Optimal Control. PhD thesis. University of Massachusetts.

Bradtke, S.J. e A. Barto (1996). Linear least-squares algorithms for temporal difference learning. In: *Machine learning.* Vol. 22. pp. 33–57.

Bradtke, S.J., B.E. Ydstie e A.G. Barto (1994). Adaptive linear quadratic control using policy iteration. In: *Computer Science Department University of Massachusetts.* Amherst, MA. pp. 3475–3479.

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

Bryson, A.E. e Yu-Chio Ho (1975). *Applied Optimal Control.* Taylor and Francis. UK.

Buşoniu, L., A. Lazaric, M. Ghavamzadeh, R. Munos, R. Babuška e B. De Schutter (2012). Least-squares methods for policy iteration. In: *Reinforcement Learning: State-Of-The-Art* (M. Wiering and M. van Otterlo, Eds.). Vol. 12 of *Adaptation, Learning, and Optimization.* pp. 75–109. Springer. Heidelberg, Germany.

Buşoniu, L., B. De Schutter e R. Babuška (2010*a*). Approximate dynamic programming and reinforcement learning. In: *Interactive Collaborative Information Systems* (R. Babuška and F.C.A. Groen, Eds.). Vol. 281 of *Studies in Computational Intelligence.* pp. 3–44. Springer. Berlin, Germany.

Buşoniu, L., B. De Schutter, R. Babuška e D. Ernst (2010*b*). Exploiting policy knowledge in online least-squares policy iteration: An empirical study. *Automation, Computers, Applied Mathematics* **19**(4), 521–529.

Buşoniu, L., B. De Schutter, R. Babuška e D. Ernst (2010*c*). Using prior knowledge to accelerate online least-squares policy iteration. In: *Proceedings of*

the 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR 2010). Cluj-Napoca, Romania. Paper A-S2-1/3005.

Buşoniu, L., D. Ernst, B. De Schutter e R. Babuška (2010*d*). Online least-squares policy iteration for reinforcement learning control. In: *Proceedings of the 2010 American Control Conference*. Baltimore, Maryland. pp. 486–491.

Buşoniu, L., D. Ernst, B. De Schutter e R. Babuška (2011). Approximate reinforcement learning: An overview. In: *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2011)*. Paris, France. pp. 1–8.

Buşoniu, L., R. Babuška, B. De Schutter e D. Ernst (2010*e*). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, ISBN: 1439821089. Boca Raton, Florida.

Chen, Z. e S. Jagannathan (2008). Generalized Hamilton-Jacobi-Bellman formulation- based neural network control of affine nonlinear discrete-time systems. *IEEE Trans. Neural Networks* **19**(1), 90–106.

Cheng, T., F.L. Lewis e M. Abu-Khalaf (2007). Fixed-final-time-constrained optimal control of nonlinear systems using neural network HJB approach. *IEEE Trans. Neural Networks* **18**(6), 1725–1736.

Cheng, Y., H. Feng e X. Wang (2012). Efficient data use in incremental actor and critic algorithms. *Neurocomputing*.

Cioffi, J.M. (1987). Limited precision effects in adaptive filtering. *Circuits and Systems, IEEE Transactions on* **34**(7), 821–833.

Dalton, J. e S.N. Balakrishnan (1996). A neighboring optimal adaptive critic for missile guidance. *Math. Comput. Model.* **23**, 175–188.

Dayan, P. (1992). The convergence of td(lambda) for general lambda. *Machine Learning* **8**(3-4), 341–362.

De Farias, D.P. e B.V. Roy (2003). The linear programming approach to approximate dynamic programming. *Oper. Res.* **51**(6), 850–865.

Deng, C. e M. Joo Er (2004). Real-time dynamic fuzzy Q-learning and control of mobile robots. In: *Control Conference, 2004. 5th Asian.* Vol. 3. pp. 1568–1576.

Dierks, T. e S. Jagannathan (2011). Online optimal control of nonlinear discrete-time systems using approximate dynamic programming. *Journal of Control Theory and Applications* **9**(3), 361–369.

Eleftheriou, E. e D.D. Falconer (1986). Tracking properties and steady-state performance of RLS adaptive filter algorithms. *Acoustics, Speech and Signal Processing, IEEE Transactions on* **34**(5), 1097–1110.

Enns, R. e J. Si (2003). Helicopter trimming and tracking control using direct neural dynamic programming. *IEEE Trans. Neural Networks* **14**(4), 929–939.

Ferrari, S. e R.F. Stengel (2004). Online adaptive critic flight control. *J. Guid. Control Dyn.* **27**(5), 777–786.

Ferreira, C.C.T., J.V. Fonseca Neto e F.A. Torrico (2003). Alocação de Autoestrutura via Controle LQG/LTR e Computação Evolutiva. *VI Simpósio Brasileiro de Automação Inteligente.*

Fonseca Neto, J.V. (2000). Alocação Computacional Inteligente de Autoestruturas para Controle Multivariável. PhD thesis. UNICAMP.

Fonseca Neto, J.V. e P.H.M. Rêgo (2014). QR-tuning and approximate-LS solutions of the HJB equation for online DLQR design via state and action-dependent heurirstic dynamic programming. *International Journal of Innovative Computing, Information and Control* **10**(3), 1071–1094.

Fonseca Neto, J.V., I.S. Abreu, P.H.M. Rêgo, M.P.M. Wolff e O.F. Silva (2008). Modelos e Convergência de um Algoritmo Genético para Alocação de Autoestrutura via RLQ. *IEEE Latin America Transactions.*

Fu, J., H. He e X. Zhou (2011). Adaptive learning and control for mimo system based on adaptive dynamic programming. *Neural Networks, IEEE Transactions on* **22**(7), 1133–1148.

Geist, M., O. Pietquin e G. Fricout (2009). Kalman temporal differences: The deterministic case. In: *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL '09. IEEE Symposium on*. pp. 185–192.

Golub, G.H. e F.V.L. Charles (1996). *Matrix Computations*. The Johns Hopkins University Press.

Gosavi, A. (2009). Reinforcement learning: A tutorial survey and recent advances. *INFORMS J. on Computing* **21**(2), 178–192.

Graupe, D. (1972). Derivation of Weighting Matrices towards satisfying Eigenvalue requirements. *Int. J. Control* **16**(5), 881–888.

Gullapalli, V. (1992). Reinforcement Learning And Its Application to Control. PhD thesis. Amherst, MA, USA. UMI Order No. GAX92-19438.

Harvey, C.A. e G. Stein (1978). Quadratic Weights for Asymptotic Regulator. *IEEE Transactions on Automatic Control* **23**(3), 378–387.

Hauskrecht, M. (2000). Value-Function Approximations for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research* **13**, 33–94.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd ed.

He, P. e S. Jagannathan (2007). Reinforcement learning neural-network-based controller for nonlinear discrete-time systems with input constraints. Vol. 37. pp. 425–436.

Howard, R.A. (1960). Dynamic programming and markov processes. John Wiley and Sons, Inc.. New York.

Iosifescu, M. (2007). *Finite Markov Processes and Their Applications*. Dover Publications.

Iyer, M.S. e D.C. Wunsch (2001). Dynamic re-optimization of a fed-batch fermentor using adaptive critic designs. *IEEE Trans. Neural Networks* **12**(6), 1433–1444.

Jacquot, R.G. (1994). *Modern Digital Control Systems*. Marcel Dekker Inc.

Javaherian, H., D. Liu, Y. Zhang e O. Kovalenko (2004). Adaptive critic learning techniques for automotive engine control. *in Proc. American Control Conf.* pp. 4066–4071.

Jiang, Y. e Z.-P. Jiang (2011). Robust approximate dynamic programming and global stabilization with nonlinear dynamic uncertainties. In: *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. pp. 115–120.

Jin, N., D. Liu, T. Huang e Z. Pang (2007). Discrete-time adaptive dynamic programming using wavelet basis function neural networks. *in Proc. IEEE Int. Symp., Approximate Dynamic Programming and Reinforcement Learning* pp. 135–142.

Johnson, M.A. e M.J. Grimble (1987). Recent Trends in Linear Optimal Quadratic Multivariable Control Systems Design. *IEEE-review* **134**, 53–71.

Kael, L., M. Littman e A. Moore (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research, 4(0)*.

Kavukcuoglu, K., P. Sermanet, Y-L. Boureau, K. Gregor, M. Mathieu e Y. LeCun (2010). Learning convolutional feature hierarchies for visual recognition. In: *Advances in Neural Information Processing Systems* (J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel and A. Culotta, Eds.). Curran Associates, Inc.. pp. 1090–1098.

Kawasaki, N. e E. Shimemura (1983). Determining Quadratic Weighting Matrices to Locate Poles in a Specified Region. *Automatica* **19**(5), 555–560.

Khan, S.G., G. Herrmann, F.L. Lewis, T. Pipe e C. Melhuish (2012). Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control* **36**(1), 42–59.

Kirk, D.E. (1970). *Optimal Control Theory: An Introduction*. Prentice-Hall Network Series. Prentice-Hall Inc.. Englewood Cliffs, New Jersey.

Kulkarni, N.V. e K. KrishnaKumar (2003). Intelligent engine control using an adaptive critic. *IEEE Trans. Contr. Syst. Technol.* **11**, 164–173.

Kuo, B.C. (1980). *Digital Control Systems*. Harcourt Brace College Publishers.

Kuzmin, V. (2002). Connectionist Q-learning in robot control task.

Lagoudakis, M.G. e R. Parr (2003). Least-squares policy iteration. *Journal of Machine Learning Research* **4**, 1107–1149.

Landelius, T. (1997). Reinforcement Learning and Distributed Local Model Synthesis. PhD thesis. Linköping University, Sweden. SE-581 83 Linköping, Sweden. Dissertation No 469, ISBN 91-7871-892-9.

Landelius, T. e H. Knutsson (1996). Greedy adaptive critics for LQR problems: Convergence proofs. Department of Electrical Engineering, Computer Vision Laboratoty. Linkoping University, 581 83 Linkoping, Sweden.

Laub, A.J. (1979). A schur method for solving algebraic riccati equations. *IEEE Transactions on Automatic Control* **24**(6), 913–921.

Lecun, Y., K. Kavukcuoglu e C. Farabet (2010). Convolutional networks and applications in vision.

Lee, J.Y., J.B. Park e Y.H. Choi (2010). Policy-iteration-based adaptive optimal control for uncertain continuous-time linear systems with excitation signals. In: *Control Automation and Systems (ICCAS), 2010 International Conference on*. pp. 464–651.

Lendaris, G.G. (2009). A retrospective on adaptive dynamic programming for control. In: *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. Vol. 0. pp. 1750–1757.

Lendaris, G.G. e C. Paintz (1997). Training strategies for critic and action neural networks in dual heuristic programming method. *IEEE Int. Conf. Neural Networks* pp. 712–717.

Lewis, F.L. e D. Vrabie (2009*a*). Adaptive dynamic programming for feedback control. In: *Asian Control Conference, 2009. ASCC 2009. 7th*. pp. 1402–1409.

Lewis, F.L. e D. Vrabie (2009*b*). Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits and Systems Magazine, IEEE* **9**(3), 32–50.

Lewis, F.L. e K.G. Vamvoudakis (2011). Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **41**(1), 14–25.

Lewis, F.L., D. Vrabie e V.L. Syrmos (2012). *Optimal Control*. 3nd ed.. John Wiley and Sons, Inc.. USA.

Li, L., M.L. Littman e C.R. Mansley (2009). Online exploration in least-squares policy iteration. In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*. Vol. 2 of *AAMAS '09*. pp. 733–739.

Liavas, A.P. e P.A. Regalia (1999). On the numerical stability and accuracy of the conventional recursive least squares algorithm. *Signal Processing, IEEE Transactions on* **47**(1), 88–96.

Lin, C.K. (2005). Adaptive critic autopilot design of bank-to-turn missiles using fuzzy basis function networks. *IEEE Trans. Syst., Man., Cybern. B* **35**(2), 197–207.

Liu, D., D. Wang e D. Zhao (2011). Adaptive dynamic programming for optimal control of unknown nonlinear discrete-time systems. In: *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*. pp. 242–249.

Liu, D., X. Xiong e Y. Zhang (2001). Action-dependent adaptive critic designs. *in Proc. Int.Joint Conf. Neural Networks* **2**, 990–995.

Liu, D., Y. Zhang e H. Zhang (2005). A self-learning call admission control scheme for CDMA cellular networks. *IEEE Trans. Neural Networks* **16**(5), 1219–1228.

Liu, G.P. e Patton, R.J (1998). *Eigenstructure Assignment for Control System Design*. John Willey & Sons.

Liu, Xin e S.N. Balakrishnan (2000). Convergence analysis of adaptive critic based optimal control. In: *American Control Conference, 2000. Proceedings of the 2000.* Vol. 3. pp. 1929–1933.

Ljung, S. e L. Ljung (1985). Error propagation properties of recursive least-squares adaptation algorithms. *Automatica* **21**(2), 157–167.

Maciejowski, J.M. (1989). *Multivariable Feedback Design.* Addison - Wiley.

Medanic, J., H.S. Tharp e W.R. Perkins (1988). Pole Placement by Performance Criterion Modification. *IEEE Transactions on Automatic Control* **33**(5), 469–472.

Miller, W.T., R. Sutton e P.J. Werbos (1990). *Neural Networks for Control.* MIT Press. Cambridge, MA.

Mohagheghi, S., Y.D. Valle, G.K. Venayagamoorthy e R.G. Harley (2007). A proportional-integrator type adaptive critic design-based neurocontroller for a static compensator in a multimachine power system. *IEEE Trans. Ind. Electron.* **54**(1), 86–96.

Murray, J.J., C.J. Cox e R.E. Saeks (2003). The adaptive dynamic programming theorem, in stability and control of dynamical systems with applications. pp. 379–394.

Murray, J.J., C.J. Cox, G.G. Lendaris e R. Saeks (2002). Adaptive dynamic programming. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* **32**(2), 140–153.

Ng, Andrew Y. e Michael Jordan (2000). Pegasus: a policy search method for large mdps and pomdps. In: *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence.* UAI'00. Morgan Kaufmann Publishers Inc.. San Francisco, CA, USA. pp. 406–415.

Park, J.W., R.G. Harley e G.K. Venayagamoorthy (2003). Adaptive-critic-based optimal neurocontrol for synchronous generators in a power system using MLP/RBF neural networks. *IEEE Trans. Ind. Applicat.* **39**(5), 1529–1540.

Pavlov, I. P. (1927). Conditional reflexes: An investigation of the physiological activity of the cerebral cortex. *Oxford University Press*.

Peng, J. e R.J. Williams (1996). Incremental multi-step Q-learning. In: *Machine Learning*. Morgan Kaufmann. pp. 226–232.

Pietquin, O., M. Geist e S. Chandramohan (2011). 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*. Vol. 3 of *IJCAI'11*. AAAI Press. pp. 1878–1883.

Pipe, A.G. (1998). An architecture for building "potential field"cognitive maps in mobile robot navigation. In: *Systems, Man, and Cybernetics, 1998. IEEE International Conference on*. Vol. 3. pp. 2413–2417.

Porta, J.M., N. Vlassis, M.T.J. Spaan e P. Poupart (2006). Point-based value iteration for continuous pomdps. *Journal of Machine Learning Research* **7**, 2329–2367.

Potter, J.E. (1963). New statistical formulas. In: *Space Guidance Analysis Memo 40*. Instrumentation Laboratory, Massachusetts Institute of Technology. Cambridge, MA.

Powell, W.B. (2007). *Approximate Dynamic Programming Solving the Curses of Dimensionality*. Wiley. Princeton, NJ.

Prokhorov, D.V. e D.C. Wunsch (1997). Adaptive critic designs. *IEEE Trans. Neural Networks* **8**(5), 997–1007.

Prokhorov, D.V., R.A. Santiago e D.C. Wunsch (1995). Adaptive critic designs: A case study for neurocontrol. *Neural Netw.* **8**, 1367–1372.

Qiao, W., R.G. Harley e G.K. Venayagamoorthy (2009). Coordinated reactive power control of a large wind farm and a statcom using heuristic dynamic programming. *IEEE Power and Energy Society General Meeting*.

Rêgo, P.H.M., J.V. Fonseca Neto e E.M. Ferreira (2013). Convergence of the standard RLS method and UD factorisation of covariance matrix for solving the

algebraic riccati equation of the DLQR via heuristic approximate dynamic programming. *International Journal of Systems Science* **0**(0), 1–23.

Romano, J.M.T. (1995). Sobre a Estabilidade Numérica dos Algoritmos de Mínimos Quadrados Rápidos. PhD thesis. UNICAMP.

Rummery, G.A. e M. Niranjan (1994). On-line Q-learning using connectionist systems. Technical report. Cambridge University, UK.

Saeks, R.E., C.J. Cox, K. Mathia e A.J. Maren (1997). Asymptotic dynamic programming: Preliminary concepts and results. *IEEE Int. Conf. Neural Networks.*

Samuel, A.L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* **3**(3), 210–229.

Sauer, T. (2011). *Numerical Analysis*. 2nd ed.. Addison-Wesley.

Shamma, J.S. (1988). Analysis and Design of Gain Scheduled Control Systems. PhD thesis. Massachusetts Institute of Technology, Department of Mechanical Engineering.

Shamma, J.S. (1996). Linearization and gain scheduling. In: *Control Handbook* (W. Levine, Ed.). pp. 388–396. CRC Press: Boca Raton, FL.

Shamma, J.S. (2012). An overview of LPV systems. In: *Control of Linear Parameter Varying Systems with Applications* (J. Mohammadpour and C. Scherer, Eds.). Springer.

Si, J., A. Barto, W. Powell e D. Wunsch (2004). *Handbook of Learning Dynamic Programming*. Wiley. Hoboken, New Jersey.

Si, J. e Y.T. Wang (2001). On-line learning control by association and reinforcement. *IEEE Trans. Neural Networks* **12**(2), 264–276.

Slock, D.T.M. (1992). Backward consistency concept and round-off error propagation dynamics in recursive least-squares algorithms. *Optical Engineering* **31**(6), 1153–1169.

Stein, G. (1979). Generalized Quadratic Weights for Asymptotic Regulator Properties. *IEEE Transactions on Automatic Control* **24**(4), 559–566.

Stingu, P. e F. Lewis (2010). *Adaptive Dynamic Programming Applied to a 6DoF Quadrotor*. Computational modeling and simulation of intellect.

Sutton, R.S. (1984). Temporal Credit Assignment in Reinforcement Learning. PhD thesis. University of Massachusetts at Amherst, Department of Computer and Information Science. Amherst, MA.

Sutton, R.S. (1988). Learning to predict by the method of temporal diferences. *Machine Learning* **3**, 9–44.

Sutton, R.S. (1999). Reinforcement learning: Past, present and future. In: *Simulated Evolution and Learning* (Bob McKay, Xin Yao, CharlesS. Newton, Jong-Hwan Kim and Takeshi Furuhashi, Eds.). Vol. 1585 of *Lecture Notes in Computer Science*. pp. 195–197. Springer Berlin Heidelberg.

Sutton, R.S., A.G. Barto e R.J. Williams (1992). Reinforcement learning is direct adaptive optimal control. *Control Systems, IEEE* **12**(2), 19–22.

Sutton, R.S. e A.G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press. Cambridge, MA.

Tesauro, G. (1989). Neurogammon wins computer olympiad. *Neural Comput.* **1**(3), 321–323.

Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.* **6**(2), 215–219.

Thrun, S. e M. Littman (2000). A review of reinforcement learning. *AI Magazine* **21**, 103–105.

Tsitsiklis, J.N. e B.V. Roy (1997). An analysis of temporal difference learning with function approximation. *IEEE Transactions on Automatic Control* **42**(5), 674–690.

Vamvoudakis, K.G. e F.L. Lewis (2010). Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **46**(5), 878–888.

Venayagamoorthy, G. K, R. G. Harley e D. G. Wunsch (2002). Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator. *IEEE Trans. Neural Networks* **13**, 764–773.

Verhaegen, M. H. (1989). Round-off error propagation in four generally-applicable, least-squares estimation schemes. *Automatica* **25**(3), 437–444.

Watkins, C.J.C.H. (1989). learning from delayed rewards. PhD thesis. University of Cambridge. Cambridge, England.

Watkins, C.J.C.H. e P. Dayan (1992). Q-learning. In: *Machine Learning*. Vol. 8. pp. 279–292.

Werbos, P.J. (1968). The elements of intelligence. *Cybernetica*.

Werbos, P.J. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. Vol. 22. Gen. Syst. Yearbk. pp. 25–38.

Werbos, P.J. (1990*a*). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Netw.* **3**(2), 179–189.

Werbos, P.J. (1990*b*). A menu of designs for reinforcement learning over time. MIT Press. pp. 67–95.

Werbos, P.J. (1992). Approximate dynamic programming for real-time control and neural modelling. In: *Handbook of Intelligent Control Neural* (D.A. White and D.A. Sofge, Eds.). pp. 493–525. Van Nostrand Reinhold, New York.

Werbos, P.J. (1995). Elastic fuzzy logic: a better fit to neurocontrol and true intelligence. *Journal of Intelligence and Fuzzy Sytems*.

Werbos, P.J. (2004). *ADP: Goals, Opportunities and Principles*. John Wiley and Sons, Inc.

Werbos, P.J. (2007). Using ADP to understand and replicate brain intelligence: the next level design. In: *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*. pp. 209–216.

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

Werbos, P.J. (2009). Intelligence in the brain: A theory of how it works and how to build it. *Neural Networks* pp. 200–212.

Werbos, P.J. (2011). *Mathematical foundations of prediction under complexity bibtex*. Erdos Lecture series. http://www.werbos.com/Neural/Erdos-talk-Werbos-final.pdf.

Werbos, P.J. (2012). *Reinforcement Learning and Approximate Dynamic Programming (RLADP): Foundations, Common Misconceptions, and the Challenges Ahead*. John Wiley and Sons, Inc.

White, D.A. e Sofge, D.A., Eds.) (1992). *Handbook of intelligent control: fuzzy, neural, and adaptive approaches*. Van Nostrand Reinhold. New York.

Wilkinson, J.H. (1968). À priori error analysis of algebraic processes. In: *International Congress Math*. Moscow.

Williams, J.K. (2009). Reinforcement learning of optimal controls. In: *Artificial Intelligence Methods in the Environmental Sciences* (SueEllen Haupt, Antonello Pasini and Caren Marzban, Eds.). pp. 297–327. Springer Netherlands.

Xu, X., D. Hu e X. Lu (2007). Kernel-based least squares policy iteration for reinforcement learning. *Neural Networks, IEEE Transactions on* **18**(4), 973–992.

Xu, X., H. He e D. Hu (2002). Efficient reinforcement learning using recursive least-squares methods. In: *Department of Automatic Control. National University of Defense Technology*. ChangSha, Hunan, 410073, P.R.China.

Yang, L., R. Enns, Y.T. Wang e J. Si (2003). Direct neural dynamic programming,. In: *in Stability and Control of Dynamical Systems with Applications* (D. Liu and P.J. Antsaklis, Eds.). Birkhaüser. Boston, MA.

Zhao, D. e Z. Hu (2011). Supervised adaptive dynamic programming based adaptive cruise control. In: *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*. pp. 318–323.