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

Melhorias de Estabilidade Numérica e Custo Computacional de Aproximadores de Funções Valor de Estado Baseados em Estimadores RLS para Projeto Online de Sistemas de Controle HDP-DLQR

Ernesto Franklin Marçal Ferreira

São Luís
2016

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

Melhorias de Estabilidade Numérica e Custo Computacional de Aproximadores de Funções Valor de Estado Baseados em Estimadores RLS para Projeto Online de Sistemas de Controle HDP-DLQR

Ernesto Franklin Marçal Ferreira

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

São Luís
2016

Ferreira, Ernesto Franklin Marçal

Melhorias de Estabilidade Numérica e Custo Computacional de Aproximadores de Funções Valor de Estado baseados em Estimadores RLS para Projeto Online de sistemas de controle HDP-DLQR / Ernesto Franklin Marçal Ferreira. - São Luís, 2016.

178f.

Impresso por computador (fotocópia).

Orientador: João Viana da Fonseca Neto.

Co-Orientadora: Patrícia Helena Moraes Rêgo.

Dissertação (Mestrado) - Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade, 2016.

1. Programação Dinâmica 2. Aprendizagem por Reforço 3. Programação Dinâmica Heurística 4. Controle Multivariável 5. Controle Ótimo 6. Regulador Linear Quadrático Discreto 7. Mínimos Quadrados Recursivos.

CDU 681.5

Melhorias de Estabilidade Numérica e Custo Computacional de Aproximadores de Funções Valor de Estado Baseados em Estimadores RLS para Projeto Online de Sistemas de Controle HDP-DLQR

Ernesto Franklin Marçal Ferreira

Aprovada em 08/03/2016

BANCA EXAMINADORA

Prof. João Viana da Fonseca Neto

Dr. em Engenharia Elétrica

Orientador

Prof[a]. Patrícia Helena Moraes Rêgo

Dr. em Engenharia Elétrica

Co-Orientadora

Prof. Francisco das Chagas Souza

Dr. em Engenharia Elétrica

Examinador Interno

Prof[a]. Laurinda Lúcia Nogueira dos Reis

Dr. em Engenharia Elétrica

Examinador Externo

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

À minha família que ajudou a tornar isso possível e aos meus orientadores que me instruíram na conclusão desse trabalho.

Agradecimentos

Primeiramente a Deus.

À minha minha família que me incentivou nessa conquista.

Ao Prof. Dr. João Viana da Fonseca Neto e a Profª. Drª. Patrícia Helena Moraes Rêgo pelo grande empenho como orientadores.

Aos companheiros e companheiras do Labseci.

Resumo

Neste trabalho, apresenta-se o desenvolvimento e a análise da estabilidade numérica de um novo algoritmo crítico adaptativo para aproximar a função valor de estado para o projeto do sistema de controle ótimo online, utilizando o regulador linear quadrático discreto (DLQR), com base em programação dinâmica heurística (HDP). O algoritmo proposto faz uso de transformações unitárias e métodos de decomposição $\mathcal{QR}$ para melhorar a eficiência da aprendizagem online na rede crítica por meio da abordagem dos mínimos quadrados recursivos (RLS). A estratégia de aprendizagem desenvolvida fornece melhorias no desempenho computacional em termos de estabilidade numérica e custo computacional, que visam tornar possíveis as implementações em tempo real da metodologia do projeto de controle ótimo com base em paradigmas de aprendizado por reforço ator-crítico. O comportamento de convergência e estabilidade numérica do algoritmo online proposto, denominado RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR, são avaliados por meio de simulações computacionais em três modelos Múltiplas-Entradas e Múltiplas-Saídas (MIMO), que representam o piloto automático de uma aeronave F-16 de terceira ordem, um circuito de quarta ordem RLC com duas tensões de entrada e dois níveis de tensão controláveis, e um gerador de indução duplamente alimentados com seis entradas e seis saídas para sistemas de conversão de energia eólica.

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

Abstract

The development and the numerical stability analysis of a new adaptive critic algorithm to approximate the state-value function for online discrete linear quadratic regulator (DLQR) optimal control system design based on heuristic dynamic programming (HDP) are presented in this work. The proposed algorithm makes use of unitary transformations and $\mathcal{QR}$ decomposition methods to improve the online learning efficiency in the critic network through the recursive least-squares (RLS) approach. The developed learning strategy provides computational performance improvements in terms of numerical stability and computational cost which aim at making possible the implementations in real time of optimal control design methodology based upon actor-critic reinforcement learning paradigms. The convergence behavior and numerical stability of the proposed online algorithm, called RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR, are evaluated by computational simulations in three Multiple-Input and Multiple-Output (MIMO) models, that represent the automatic pilot of an F-16 aircraft of third order, a fourth order RLC circuit with two input voltages and two controllable voltage levels, and a doubly-fed induction generator with six inputs and six outputs for wind energy conversion systems.

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

1.1 Introdução

Na atualidade, diversos problemas são encontrados nos mais diversos tipos de setores ligados a indústria, ciência, tecnologia, economia e muitos outros que necessitam de algum tipo de controle que garanta que o processo funcione apropriadamente e satisfaça os objetivos de desempenho prescritos. Com a evolução de processos industriais e com novas aplicações para controle surgindo a cada momento (Franklin *et al.* 2014), técnicas de controle moderno devem ser aplicadas para controle em tempo real dessas plantas. A teoria de controle adaptativo (Lewis and Vrabie 2009*c*) apresenta uma abordagem eficiente e com bons resultados para esses problemas.

Em uma outra perspectiva de projeto de controle, tem-se a teoria de controle ótimo que lida com a operação de um sistema dinâmico com um custo mínimo (Lewis *et al.* 2012)(Kirk 2004)(Friedland 2005)(Falb 2006). Não obstante, no final, após o controlador ser projetado, ele é colocado em serviço sem mecanismos associados para adaptar seu projeto em resposta às mudanças no contexto (planta, ambiente, medidas de desempenho associadas). Os projetos de controle são normalmente realizados via solução offline da equação de Hamilton-Jacobi-Bellman (por exemplo, a equação de Riccati), tendo por base um modelo detalhado da dinâmica da planta.

Controladores adaptativos, por sua vez, aprendem online para controlar sistemas com dinâmicas desconhecidas usando dados medidos em tempo real ao longo

da trajetória do sistema. Neste quadro, incertezas de modelagem podem existir, incluindo os parâmetros imprecisos, dinâmicas não-modeladas de alta frequência, e perturbações. Além disso, tanto a dinâmica do sistema quanto os objetivos de desempenho podem mudar com o tempo. Entretanto, controladores adaptativos não são geralmente projetados com a qualidade de serem ótimos no sentido de minimizarem funções de custo como definido no quadro de controle ótimo.

Em uma classe de métodos denominada controle adaptativo indireto (Astrom and Wittenmark 1994), emprega-se técnicas de estimação para se determinar os parâmetros do modelo a partir das medidas de entrada e saída do sistema e, em seguida, usa-se o modelo obtido para resolver a equação de projeto ótimo (equação de Hamilton-Jacobi-Bellman). Sabe-se bem que procedimentos de modelagem e identificação para a dinâmica de um dado sistema não linear, muitas vezes, é um procedimento iterativo demorado o qual requer projeto de modelo, identificação de parâmetro e validação de modelo em cada iteração. Esta técnica torna-se ainda menos viável quando o sistema apresenta dinâmicas não lineares desconhecidas as quais se manifestam em certas regiões operacionais.

Tornou-se então necessário desenvolver um método online que forneça soluções de controle ótimo para um sistema com dinâmica não-linear sem fazer referência aos parâmetros do modelo de tal sistema. Uma alternativa para suprir essa necessidade são os métodos de programação dinâmica aproximada (*Adaptive Dynamic Programming* - ADP). Tal abordagem emprega um mecanismo "para frente no tempo"que busca por uma política de controle ótima adaptando sucessivamente duas estruturas paramétricas, nominalmente uma rede de ação e uma rede crítica, para aproximar a solução da equação HJB. A rede de ação calcula as ações de controle, enquanto a rede crítica aprende a aproximar a função valor. Esta função avalia o efeito do controle sobre o seu desempenho futuro, e fornece diretrizes sobre como melhorar a lei de controle. Essas estruturas combinadas com métodos de aprendizagem por reforço (*Reinforcement Learning* - RL) têm sido usados como um quadro de "aprendizagem independente de modelo"que resolve problemas de decisão ótima em tempo real.

A programação dinâmica aproximada, também conhecida como programação neuro-dinâmica 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. Outros pesquisadores que contribuíram na formulação de ADP destacam-se Barto (Barto *et al.* 1983), Widrow (Widrow *et al.* 1973), Howard (Howard 1960), Watkins (Watkins 1989), Bertsekas e Tsitsiklis (Bertsekas and Tsitsiklis 1996). Segundo Lewis e Derong (Lewis and Liu 2012), ADP refere-se a uma família de métodos ator-crítico para encontrar as melhores soluções em tempo real. Estas técnicas utilizam melhorias computacionais, como aproximação de função, para desenvolver algoritmos para sistemas complexos com distúrbios e incertezas na sua dinâmica.

A aprendizagem por reforço descreve uma família de sistemas de aprendizado de máquina que operam com base em princípios utilizados em animais, grupos sociais e sistemas naturais. Métodos RL foram usados por Ivan Pavlov na década de 1860 para treinar seus cães (Pavlov 1927). RL refere-se a um ator ou agente que interage com seu ambiente e modifica suas ações ou políticas de controle baseadas em estímulos recebidos em resposta a suas ações. Os algoritmos de RL são construídos baseados na ideia de que as decisões de controle de sucesso devem ser lembradas, por meio de um sinal de reforço, de tal forma que elas se tornam mais propensas a serem usadas outra vez. Desta forma, o foco do processo de aprendizagem é voltado para um índice de desempenho que quantifica o quão próximo da otimalidade está o sistema de controle de malha fechada em operação, já não tendo como objeto de interesse a dinâmica do sistema.

É importante ressaltar que para implementações em tempo real de controladores ótimos adaptativos em sistemas computacionais (FPGA, ASIC e PSoC), algoritmos de ADP devem contar primeiramente com aproximadores de função que encontrem "boas"soluções aproximadas da equação HJB no sentido de atender alguns parâmetros de desempenho, tais como: Convergência Paramétrica, Redução do *Overshoot*, Estabilidade Numérica, Tempo de Convergência e Complexidade Numérica.

Assim, focando-se em melhorar o desempenho de algoritmos de ADP para controle em tempo real, com respeito aos parâmetros mencionados acima, apresenta-se, neste trabalho, o desenvolvimento de um método crítico adaptativo baseado em transformações unitárias e decomposição $\mathcal{QR}$, as quais são incorporadas na rede

crítica para melhorar a eficiência da aprendizagem online por meio da abordagem dos mínimos quadrados recursivos (*Recursive Least Square* - RLS). Especificamente, esses esquemas são orientados para resolver online a equação HJB do tipo Riccati associada ao problema do regulador linear quadrático discreto (*Discrete Linear Quadratic Regulator* - DLQR). Como citado em (Golub and Loan 1996), (Haykin 1995), (Horn and Johson 1987), (de Campos and Strang 2009), métodos de decomposição $\mathcal{QR}$ constituem uma excelente ferramenta matemática que promovem maior robustez, redução da complexidade e a capacidade de uma possível implementação em microcontroladores por meio da implementação sistólica.

1.2 Objetivos

1.2.1 Objetivo Geral

Portanto os objetivos principais desse projeto são:

- Desenvolver métodos de solução online de Controle Ótimo do tipo Regulador Linear Quadrático Discreto (DLQR) por meio de Programação Dinâmica Aproximada (ADP)
- Desenvolver um algoritmo de Controle Ótimo Adaptativo que possua total capacidade de implementação em ambiente industrial.

1.2.2 Objetivos Específicos

- **Desenvolver Algoritmo de ADP com melhor desempenho computacional a nível de custo computacional**

  Com a evolução exponencial da computação, a preocupação com a otimização da execução dos algoritmos de controle foi ignorada. Porém, para algumas soluções em tempo real, essa problemática não pode ser descartada, necessitando assim de um algoritmo de alto desempenho de execução. Assim, pela teoria da complexidade computacional, um problema é considerado como inerentemente difícil se a sua solução requer recursos significativos, qualquer que seja o algoritmo usado. A teoria formaliza esta intuição

por meio da introdução de modelos matemáticos de computação para estudar e quantificar os recursos necessários para resolvê-los, tais como tempo, armazenamento, quantidade de comunicação, número de portas no circuito e número de processadores (Papadimitriou 1994) (Arora and Barak 2006).

- **Melhorar a estabilidade numérica na execução de algoritmos complexos como o RLS$_\lambda$-HDP-DLQR**

  Quando se combina várias técnicas de controle em um único algoritmo é normal que ocorra problemas com estabilidade numérica na execução desse, logo será necessária uma análise da estabilidade do algoritmo. No contexto do presente trabalho isso ocorre devido ao fenômeno de instabilidade numérica que 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 aos problemas de mal condicionamento dessa matriz, a solução para uma política de decisão ótima para um dado ponto de operação pode não convergir.

1.3 Motivação

Com o avanço de tecnologias de informação e a necessidade por soluções de controle de alto desempenho tem havido um interesse crescente no desenvolvimento de métodos de controle ótimo que tenham especificações necessárias para sua implementação no mundo real. Dentre esses requisitos estão a quantidade de cálculos e o uso de memória, com custo computacional relativamente baixo, de modo que algoritmos de controle ótimo possam ser executados por microprocessadores industrializados, os quais são responsáveis pelo controle de automóveis, aeronaves, máquinas automáticas e processos industriais. Outro requisito é a capacidade de tais algoritmos operarem satisfatoriamente mesmo na presença de mal-condicionamento do sinal de entrada e erro de quantização (erro de arredondamento). Esses são os principais tópicos que impulsionaram o desenvolvimento dessa pesquisa tendo em vista o projeto de algoritmos de ADP para controle ótimo com maior capacidade para operarem em sistemas embarcados.

1.4 Estado da Arte

Em um mundo onde os processos devem ser aprimorados em busca de maior produção e menor consumo a automação ganha papel de destaque. Para isso várias técnicas de controle tem sido propostas buscando alcançar índices de satisfação, uma metodologia bem estudada é a do Controle Ótimo que busca em sua solução minimizar ou maximizar um índice de desempenho associado ao comportamento da planta (Dierks and Jagannathan 2011) (Quirion *et al.* 2004) (Anderson and Moore 1990). A principal desvantagem das abordagens clássicas de controle ótimo é que elas resolvem a equação de projeto HJB (Hamilton-Jacobi-Bellman) (Lewis and Vrabie 2009*c*) de modo off-line, o controlador é ajustado antes de entrar em operação, tornando o sistema pouco robusto à possíveis variações paramétricas.

Em busca de uma abordagem que forneça soluções mais abrangentes para problemas de controle ótimo, surge a Programação Dinâmica (*Dynamic Programming* - DP) que ganhou espaço como um método que busca encontrar a solução ótima em problemas de decisão sequenciais. A DP foi desenvolvida no final dos anos 1950 com o trabalho de Bellman e Pontryagin (Bellman and Kalaba 1965) (Bellman 2003)(Bertsekas 1987) (Bertsekas 1995). O problema da programação dinâmica é ela ser um procedimento "para trás no tempo"que não oferece métodos para resolver problemas de controle ótimo de uma forma "para frente no tempo", característica necessária para controle em tempo real.

Na tentativa de desenvolver um método que tivesse características de aprendizado parecida com a dos animais surge a Aprendizagem por Reforço (RL) que, segundo Sutton e Barto (Sutton and Barto 1998) (Sutton *et al.* 1992), é um formalismo da inteligência artificial que permite aos sistemas aprenderem a tomar "boas"decisões (políticas de controle) observando o seu próprio comportamento, e usarem estratégias embutidas para melhorar suas ações através de um mecanismo por reforço. Isso garante que o sistema aprenda qual será a melhor decisão baseada em experiências passadas, possibilitando o uso de estratégias para projetar sistemas para os quais modelos explícitos não estão disponíveis.

Assim buscando uma técnica que venha a combinar as características dos três métodos anteriores, chega-se a Programação Dinâmica Aproximada (Murray *et al.* 2002) (Stingu and Lewis 2010) que é uma classe de algoritmos que fornece uma solução online para problemas de controle ótimo usando representações aproxima-

das da função valor a ser minimizada tendo por base o princípio da otimalidade de Bellman. O uso da técnica do crítico adaptativo permite que a aplicação seja online, porque ela efetivamente resolve as equações de otimização HJB em uma maneira "para frente no tempo", enquanto que a programação dinâmica tradicional, como já mencionado anteriormente, requer um procedimento "para trás no tempo", (Werbos 1990).

ADP tem mostrado inúmeras aplicações em projeto de controle ótimo online, tais como aeronaves de alto desempenho ou sistemas de mísseis, (Bertsekas *et al.* 2000), piloto automático, (Ferrari and Stengel 2004), (Lin 2005), e robótica, (Khan and Lewis 2012). Em (Viana da Fonseca Neto *et al.* 2013), os autores desenvolveram um novo método para o projeto de sistemas de controle ótimo online para geração de energia eólica e solar com base em paradigmas ADP. Pode-se também citar como importantes pesquisas na área de ADP (Lewis and Vrabie 2009*a*), (Wang *et al.* 2009*b*), (Wang *et al.* 2009*a*) e (Buşoniu *et al.* 2011). Além de outras aplicações bem sucedidas que são apresentadas em (Weber and Mayer 2008), e seus trabalhos incluem aprender a controlar: robôs móveis autônomos, helicópteros, PH industrial, plantas de oxidação, e gestão do tráfego aéreo.

Uma outra preocupação para a realização de controle ótimo em tempo real trata-se da estabilidade e desempenho dos algoritmos de controle. A estabilidade numérica é uma característica que garante que o algoritmo apresentará o mesmo comportamento ao passar do tempo de execução (Robert and George 2001). Para isso, propõe-se o desenvolvimento de uma metodologia baseada em decomposições ortogonais para melhorar o condicionamento numérico das matrizes internas desses algoritmos, (Decomposição $\mathcal{QR}$) (Shoaib 2006) (Qingyan 2008), permitindo assim que o sistema opere satisfatoriamente mesmo na execução de cálculos matriciais mais complexos, tais como: multiplicação/divisão, radiciação, inversão e outros.

Ainda, deve-se analisar o custo e o tempo computacional do algoritmo (Papadimitriou 1994). O custo computacional de um algoritmo está relacionado a quantos *flops* (numero de operações matemáticas realizadas por um microprocessador) são requeridos para execução do mesmo, (Golub and Van Loan 1996). O tempo computacional (tempo que o algoritmo leva para executar uma iteração do algoritmo) pode ser reduzido por meio da simplificação de operações matemáticas

envolvidas no algoritmo (Arora and Barak 2006).

1.5 Organização da Dissertação

Este trabalho contêm 7 capítulos e mais 3 apêndices, que serão necessários para apresentar a base teórica sobre os assuntos abordados no trabalho, bem como as contribuições e realizações desenvolvidas no mesmo. No final do documento, estão expostos conhecimentos matemáticos que ajudarão na formulação da metodologia para projeto e avaliação de algoritmos de controle ótimo online baseado em paradigmas de ADP, proposta neste trabalho.

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, **Decomposição** $\mathcal{QR}$ aplicada para o desenvolvimento de algoritmos de controle ótimo online, **Complexidade Computacional** na avaliação de parâmetros da execução dos algoritmos desenvolvidos no trabalho, **Simulações** com os resultados e gráficos dos sistemas avaliados.

Assim, no Capítulo 2 é apresentada a teoria relacionada 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. É discutido ainda as principais abordagens para a solução de problemas de controle ótimo, a saber: cálculo variacional, programação dinâmica para trás e programação dinâmica para frente.

No Capítulo 3, apresenta-se o desenvolvimento de esquemas HDP 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 4 é onde se explicita as principais contribuições do trabalho, ou seja, a proposta de uma metodologia de projeto para sistemas de controle ótimo online baseada em ADP e o desenvolvimento matemático necessário para o projeto dos dois algoritmos resultantes dessa metodologia, nominalmente RLS_λ-HDP-DLQR e RLS_λ-$\mathcal{QR}$-HDP-DLQR.

No Capítulo 5, apresenta-se a Complexidade Computacional bem como alguns métodos para avaliação da mesma, como: Custo Computacional, Estabilidade

Numérica e Tempo de Convergência. Na sequência, testes computacionais são apresentados, Capítulo 6, verificando a melhoria dos algoritmos RLS_λ-$\mathcal{QR}$-HDP-DLQR desenvolvidos ao usar decomposição $\mathcal{QR}$. O desempenho computacional é avaliado a nível de estabilidade numérica desses algoritmos.

No Capítulo 7, apresentam-se as conclusões sobre as metodologias abordadas para aproximação da solução da equação HJB-Riccati e sugestões de trabalhos futuros. Por fim, os Apêndices são voltados para complementar a fundamentação das abordagens utilizadas. Nas Referências Bibliográficas, lista-se o acervo utilizado na construção das bases sólidas necessárias ao desenvolvimento desta dissertação.

Capítulo 2

Aprendizagem por Reforço e Controle Ótimo

2.1 Introdução

No universo do aprendizado (inteligencia artificial) existem dois paradigmas principais: *aprendizagem com um professor* (supervisionada) e *aprendizagem sem professor*. O paradigma de aprendizagem sem professor é subdividido em *aprendizagem auto-organizada* (não supervisionada) (Haykin 2008) e *aprendizagem por reforço*, que é o foco desse capítulo.

Todo organismo vivo interage com o ambiente e usa estas interações para melhorar suas ações para sobreviver. Estas modificações das ações baseadas nas interações, é chamada de Aprendizagem por Reforço (AR).

Aprendizagem por Reforço se refere a um ator ou agente que interage com o ambiente e modifica suas ações, ou políticas de controle, baseados nos estímulos recebidos em resposta a essas ações. Os algoritmos de AR são construídos com a ideia de que decisões sucessivas de controle são tomadas com a intenção de maximizar o reforço ao longo do tempo.

Esse reforço pode ter duas formas básicas: custo ou recompensa, que estão intrinsecamente relacionados, a necessidade por obter o mínimo de custo ou o máximo em recompensa resulta em um problema de otimização, que deve ser resolvido para o projeto do controlador.

A otimalidade dos controladores é garantida por meio da equação de Hamilton-

Jacob-Bellman, tendo-se a possibilidade de estabelecer um caminho de trajetórias ótimas através da Programação Dinâmica.

Este capítulo tem enfoque na caracterização do controlador ótimo online conduzida por princípios de Programação Dinâmica e Processo de Decisão Markoviano, por fim, uma abordagem introdutória sobre RL e os aspectos essenciais como Função Valor, e uma classe de AR chamada Ator-Crítico.

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 1997).

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:

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.

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.

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 g. 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 c), a qual é uma predição de recompensas descontadas total esperadas associadas com pares estado-ação (x, u) iniciais, normalmente representada por $c(x, u)$. Que pode ser encontrados em (Sutton 1988), 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 Programação Dinâmica

A Programação Dinâmica, conhecida também como otimização recursiva, é uma técnica que trata de situações nas quais as decisões são tomadas em etapas, com o resultado de cada decisão sendo previsível até certo ponto, antes que a próxima decisão seja tomada. Um aspecto chave destas situações, é que nenhuma decisão pode ser tomada isoladamente, em vez disso, deve-se ponderar o desejo de um baixo custo no presente em relação a altos custos indesejáveis no futuro (Haykin 2008). Esta se baseia no princípio da otimalidade desenvolvido por Bellman

em 1953, que é descrito como:

"Uma política ótima tem a propriedade que, quaisquer que sejam o estado inicial e a decisão inicial, as decisões restantes devem constituir uma política ótima em relação ao estado resultante da primeira decisão"

A programação dinâmica aborda a questão: Como um sistema pode aprender a melhorar seu desempenho a longo prazo quando isto pode requerer o sacrifício do desempenho a curto prazo

Frequentemente as soluções são obtidas por um processo regressivo, trabalhando o problema do final para o início. Com isso a dificuldade será reduzida, ao se decompor o problema em uma sequência de problemas inter-relacionados mais simples. A programação dinâmica tem como questão fundamental, o interesse em buscar a resposta de como um sistema pode aprender a melhorar o seu desempenho a longo prazo, quando isto pode exigir o sacrifício do desempenho atual.

Assim pode se caracterizar o problema da programação dinâmica, como um **Processo de Decisão Markoviano** (PDM) que é o arcabouço padrão para problemas de planejamento probabilístico

Esse Processo modela a interação de um agente em um ambiente, que executa ações com efeitos probabilísticos que podem levar o agente a diferentes estados

O PDM é caracterizado como:

- O ambiente evolui probabilisticamente ocupando um conjunto finito de estados discretos
- Para cada estado do ambiente há um conjunto finito de ações possíveis que podem ser realizadas pelo sistema de aprendizagem
- Toda vez que o sistema de aprendizagem realiza uma ação, ele incorre em certo custo.
- A observação dos estados, a realização de ações e a incidência de custos ocorrem em tempo discreto

O processo de decisão markoviano pode ser representado como um problema de detecção de rota que pode ser visto na Figura 2.1.

Figura 2.1: Problema de detecção de rota

Para o *estado* i, por exemplo, o conjunto disponíveis de ações, entradas aplicadas ao ambiente pelo sistema de aprendizagem, é representado por $\mathcal{U}_i = \{u_{ik}\}$, onde o segundo indicie k na ação u_{ik} realizada pelo sistema de aprendizagem meramente indica a disponibilidade de mais que uma ação possível quando o ambiente está no *estado* i. A transição do ambiente do *estado* i para o novo *estado* j, por exemplo, devido a ação u_{ik} é de natureza probabilística. Entretanto a probabilidade de transição do *estado* i para o *estadoj* depende inteiramente do estado corrente i e da ação correspondente u_{ik}, Esta é a propriedade de Markov,

Considere que $p_{ij}(u)$ representa a probabilidade de transição do estado i para o estado j devido à ação realizada no passo de tempo k, onde $u_t = u$. Assim,

$$p_{ij}(u) = \mathcal{P}\left(x_{k+1} = j | \, x_k = i, u_k = u\right)$$

A probabilidade de transição $p_{ij}(u)$ satisfaz as duas condições que são impostas pela teoria da probabilidade.

$$p_{ij}(u) \geq 0 \;\; para \;\; todo \;\; i \;\; e \;\; j$$

e

$$\sum_j p_{ij}(u) = 1 \; para \; todo \; i$$

Para um dado número de estados e probabilidades de transição, a sequencia de estados do ambiente resultante das ações realizadas pelo sistema de aprendizagem sobre o tempo forma uma cadeia de Markov.

A cada transição de um estado para outro, o sistema de aprendizagem incorre em um custo, Assim, na k-ésima transição do estado i para o estado j sobre a ação

u_{ik}, o sistema de aprendizado incorre em um custo representado por $\gamma^k c(i, u_{ik}, j)$, onde $c(.,.,.)$ é uma função predeterminada, e γ é um escalar com $0 \leq \gamma < 1$ chamado de **fator de desconto**. Ajustando-se γ é capaz de controlar o grau com que o sistema de aprendizagem está preocupado com as consequências a longo prazo de suas próprias ações em relação às consequências a curto prazo destas ações.

O interesse está na formulação de uma política, definida como um mapeamento de estados para ações. Em outras palavras, uma política é uma regra usada pelo sistema de aprendizagem para decidir o que fazer, dado o conhecimento do estado atual do ambiente. A política é representada por

$$G = \{g_0, g_1, g_2, ...\}, \tag{2.1}$$

onde g_k é uma função que mapeia o estado $x_k = i$ em uma ação $u_k = u$ no passo de tempo $k = 0, 1, 2, ...$ Este mapeamento é tal que

$$g_k(i) \in U_i \text{ para todos os estados } i \in \mathcal{X},$$

onde U_i representa o conjunto de todas as ações possíveis realizadas pelo sistema de aprendizagem no estado i. Tais políticas são denominadas admissíveis.

Uma política pode ser não-estacionaria ou estacionaria. Uma política não-estacionária é variável no tempo, como indicado na Equação (2.1). Entretanto, quando a política é independente do tempo, ou seja,

$$G = \{g, g, g, ...\}, \tag{2.2}$$

diz-se que a política é estacionária. Em outras palavras, uma política estacionária especifica exatamente a mesma ação cada vez que um estado particular é visitado. Para uma política estacionária, a cadeia de Markov relacionada pode ser estacionária ou não-estacionária; é possível utilizar uma política estacionária sobre uma cadeia de Markov não-estacionária, mas não é recomendável se fazer isso. Se uma política estacionária g for empregada, então a sequencia de estados $\{x_k, k = 0, 1, 2, ...\}$ forma uma cadeia de Markov com probabilidades de transição $p_{il}(g(i))$, onde $g(i)$ significa uma ação. É por essa razão o processo é referido como um processo de decisão de Markov (Haykin 2008).

Um problema de programação dinâmica pode ser do tipo horizonte finito ou de horizonte infinito. Em um problema de horizonte finito, o custo é acumulado em um número finito de estágios. Em um problema de horizonte infinito, o custo é acumulado em um número infinito de estágios. Os problemas de horizonte infinito fornecem uma aproximação razoável a problemas envolvendo um numero finito mas muito grande de estágios. Eles também são particularmente interessantes porque o desconto assegura que os custos para todos os estados são finitos para qualquer política.

O custo total de um problema de horizonte infinito, iniciando de um estado $x_0 = 1$ e usando um política $G = \{g_0, g_1, g_2, ...\}$, é dado por:

$$V^G(i) = E\left[\sum_{k=0}^{\infty} \gamma^k g(x_k, g_k(x_k), x_{k+1} | \, x_0 = i\right], \tag{2.3}$$

e o seu valor ótimo é dado por:

$$V^*(i) = \min_G V^G(i).$$

Pode-se agora resumir o problema básico de programação dinâmica como:

Dado um processo de decisão markoviano estacionário que descreve a interação entre um sistema se aprendizagem e seu ambiente, encontre uma política estacionaria $G = \{g, g, g, ...\}$ que minimize a função custo para avançar $V^g(i)$ para todos os estados iniciais i

2.2.1 Equação de Bellman e Política Ótima

A técnica de programação dinâmica se fundamenta em uma ideia muito simples conhecida como o princípio de otimalidade de Bellman (Bellman 2003). Expresso de uma forma simples, esse princípio declara:

"Uma política ótima tem a propriedade de que, quaisquer que sejam o estado e as decisões iniciais (isto é, o controle), as decisões restantes devem constituir uma política ótima com respeito ao estado resultante da primeira decisão "

Uma "decisão"é estabelecida como uma escolha de controle em um tempo particular, e uma "política"é a sequencia de controle inteira ou a função de controle.

Para formular o principio da otimização em termos matemáticos, considere um problema de horizonte finito para o qual a função de custo para avançar e definida por

$$V_0(x_0) = E\left[c_N(x_N) + \sum_{k=0}^{N-1} c_k(x_k, g_k(x_k), x_{k+1})\right], \tag{2.4}$$

onde N é o horizonte de tempo e $c_N(x_N)$ é o custo final. Dado x_0, o valor esperado na Equação (2.4) é em relação aos estados restantes $x_1, ..., x_{N-1}$.

Pode-se assim formalmente formular o algoritmo de programação dinâmica como (Bertsekas 1987)

Para todo estado inicial x_0, o custo ótimo $V^*(x_0)$ do problema básico de horizonte finito é igual a $V_0(x_0)$, onde a função V_0 é obtida do último passo do seguinte algoritmo:

$$V_k(x_k) = \min_{g_k} \underset{x_{k+1}}{E} \left[c_k(x_k, g_k(x_k), x_{k+1}) + V_{k+1}(x_{k+1})\right], \tag{2.5}$$

que age "para trás no tempo", com

$$V_N(x_N) = c_N(x_N),$$

Além disso, se g_k^* minimiza o lado direito da Equação (2.5) para cada x_k e k, então a política $G^* = \{g_0^*, g_1^*, ..., g_{N-1}^*\}$ é ótima.

Na sua forma básica, o algoritmo de programação dinâmica trata de um problema de horizonte finito. Estamos interessados em estender o uso deste algoritmo para tratar do problema descontado de horizonte infinito descrito pela função de custo para avançar da Equação (2.3) sob uma política estacionária $G = \{g, g, g, ...\}$. Tendo este objetivo em mente, pode-se proceder da seguinte forma:

- Inverter o índice de tempo do algoritmo de modo que corresponda ao problema descontado.
- Definir o custo $c_k(x_k, g(x_k), x_{k+1})$ como:

$$c_k(x_k, g(x_k), x_{k+1}) = \gamma^k c(x_k, g(x_k), x_{k+1}), \tag{2.6}$$

Pode-se agora reformular o algoritmo de programação dinâmica como segue:

$$V_{k+1}(x_0) = \min_g \underset{x_1}{E} \left[c(x_0, g(x_0), x_1) + \gamma V_k(x_1) \right], \tag{2.7}$$

que começa a partir das condições iniciais

$$V_0(x) = 0 \qquad \text{para todo } x \in \mathcal{X},$$

O estado x_0 é o estado inicial, x_1 é o novo estado que resulta da ação da política g e γ é o fator de desconto.

Considere que $V^*(i)$ represente o custo ótimo de horizonte infinito para o estado inicial $x_0 = i$. Pode-se então ver $V^*(i)$ como o limite do custo ótimo de N estágios correspondente $V_N(i)$ quando o horizonte N se aproxima do infinito; isto é,

$$V^*(i) = \lim_{N \to \infty} V_N(i) \quad \textit{para todo } i \tag{2.8}$$

Esta relação é o elo de conexão entre os problemas descontados de horizonte finito e de horizonte infinito. Fazendo $k + 1 = N$ e $x_0 = i$ na Equação (2.7) e então aplicando a Equação (2.8), obtêm-se

$$V^*(i) = \min_g \underset{x_1}{E} \left[c(i, g(i), x_1) + \gamma V^*(x_1) \right] \tag{2.9}$$

Para estimar o custo ótimo de horizonte infinito $V^*(i)$, procede-se em dois estágios:

1. Estima-se o valor esperado do custo $c(i, g(i), x_i)$ em relação a x_1 escrevendo

$$E\left[c(i), g(i), x_1\right] = \sum_{j=1}^{s} p_{ij} c(i, g(i), j) \tag{2.10}$$

onde s é o número de estados do ambiente e p_{ij} é a probabilidade de transição do estado inicial $x_0 = i$ para o novo estado $x_1 = j$. A quantidade definida na Equação (2.10) é o **custo esperado imediato** incorrido no estado i por seguir a ação governada pela política g. Representando este custo por $\tilde{c}(i, g(i))$, que pode ser escrito como

$$\tilde{c}(i, g(i)) = \sum_{j=1}^{s} p_{ij} c(i, g(i), j) \tag{2.11}$$

2. Estima-se o valor esperado de $V^*(x_1)$ em relação a x_1. Aqui nota-se que se há conhecimento do custo $V^*(x_1)$ para cada estado x_1 de um sistema de estados finitos, pode-se determinar facilmente o valor esperado de $V^*(x_1)$ em termos das probabilidades de transição da cadeia de Markov subjacente escrevendo

$$E[\beta\alpha] = \sum_{j=1}^{s} p_{ij} V^*(j) \tag{2.12}$$

Assim, utilizando as Equações (2.10) a (2.12) na Equação (2.9), obtêm-se o resultado desejado

$$V^*(i) = \min_g \left\{ \tilde{c}(i, g(i)) + \gamma \sum_{j=1}^{s} p_{ij} V^*(j) \right\} \quad para\ i = 1, 2, ..., s \tag{2.13}$$

A Equação (2.13) é chamada a equação de otimalidade de Bellman. Ela não deve ser vista como um algoritmo. Em vez disso, representa um sistema de s equações, com uma equação por estado. A solução deste sistema de equações define as funções de custo para avançar ótimas para os s estados do ambiente.

2.2.2 Métodos de Busca de Política Ótima

Como bem observado, o sistema necessita de grande poder computacional para que os custos das transições estejam disponíveis para o agente, que é o subsistema responsável por implementar as políticas ótimas. Tipicamente, existem três métodos para se determinar uma política ótima (Khan *et al.* 2012) **iteração de política**, **iteração de valor** e **iteração gulosa**, detalhadas a seguir.

Iteração de Política: Neste método é providenciado um laço que alterna entre a avaliação e a melhoria da política. Primeiramente, o algoritmo avalia a política g e obtêm a função valor correspondente que venha a satisfazer:

$$V^{g_k}(i) = \tilde{c}(i, g_k(i)) + \gamma \sum_{j=1}^{s} p_{ij}(g_k(i)) V^{g_k}(j), \quad i = 1, 2, \ldots, s. \tag{2.14}$$

A função V^{g_k} mostrada na Equação (2.14) é estimada de maneira a atualizar a política, através da melhoria da mesma:

$$g_{k+1}(i) = \arg\min_{u \in U_i} \left\{ \tilde{c}(i,u) + \gamma \sum_{j=1}^{s} p_{ij}(u) V^{g_k}(j) \right\}, \quad i = 1, 2, \ldots, s. \tag{2.15}$$

Estas duas medidas são aplicadas ao sistema até haver a convergência. Em (Bradtke *et al.* 1994) é mostrado que a convergência é verdadeira se as Equações (2.14) e (2.15) forem satisfeitas.

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 a seguinte Equação (2.16)

$$V_{k+1}(i) = \min_{u \in U_i} \left\{ \tilde{c}(i,u) + \gamma \sum_{j=1}^{s} p_{ij}(u) V_k(j) \right\}, \tag{2.16}$$

para todos os estados i. A iteração de valor é garantida convergir para uma função que satisfaz a equação da otimalidade de Bellman (2.13) baseada em uma representação tabular da função valor, (Bertsekas 1995).

Iteração Gulosa: É um método que tanto a avaliação da política quanto a melhoria são executadas ao mesmo tempo. Desta maneira é necessário que exista uma avaliação de política inicial e uma melhoria inicial, pois há interdependência nas Equações (2.14) e (2.15). Desta forma, a política atual é considerada, para cada ciclo, a ser sempre a política ótima. A busca é feita utilizando-se técnicas iterativas e em cada iteração a parametrização da função valor é atualizada afim de satisfazer a equação da otimalidade de Bellman o que resultara na atualização de $\hat{V}^*$ tendendo ao valor ótimo de V^*, logo

$$\hat{V}^*(i) = \tilde{c}(i, \hat{g}^*(i)) + \gamma \sum_{j=1}^{s} p_{ij}(\hat{g}^*) \hat{V}^*(j), \tag{2.17}$$

$$\hat{g}^*(i) = \arg\min_{u \in U_i} \left\{ \tilde{c}(i,u) + \gamma \sum_{j=1}^{s} p_{ij}(u) \hat{V}^*(j) \right\}, \quad i = 1, 2, \ldots, s. \tag{2.18}$$

No método de iteração gulosa (Landelius and Knutsson 1996), a função valor estimada é tratada como ótima em relação a nova política estimada de controle.

Desta forma, as funções das Equações (2.17) e (2.18) são avaliadas na ordem inversa pois o foco do algoritmo é a política parametrizada $\hat{g}^*(x, w)$, sendo w o vetor de parâmetros da representação parametrizada da função valor e seus resultados induzirão a versão atualizada de $\hat{V}^*$.

2.3 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 $\mathcal{P}$ da função de probabilidade de transição de estados e c da função de custo instantâneo. 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, é impossível fornecer estimativas das probabilidades de transição de estados. Neste caso, métodos de diferenças temporais (*Temporal Differences* - TD) (Sutton and 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).

O erro de Diferenças Temporais (DT) pode ser considerado como um erro de predição entre o desempenho previsto e o desempenho observado em resposta a uma ação aplicado ao sistema (Lewis and Vrabie 2009*b*). como pode ser visto na Figura 2.2 .

Figura 2.2: Erro de predição (Diferença Temporal)

2.3.1 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^g para uma dada política g. A abordagem envolve a correção das predições anteriores e constrói estimativas da função valor V^g baseadas em tuplas de dados da forma $(x_k, x_{k+1}, g(x_k, g(x_k), x_{k+1}))$ observadas ao longo da realização de trajetória do estado, (Sutton and Barto 1998). O método, em sua versão à um passo, atualiza V^g após cada transição de estado de acordo com a equação que é dada por

$$V_{k+1}(x_k) = V_k(x_k) + \alpha_k \Delta_k, \tag{2.19}$$

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

$$\Delta_k = c(x_k, g(x_k), x_{k+1}) + \gamma V_k(x_{k+1}) - V_k(x_k), \tag{2.20}$$

expressando a diferença entre a estimativa atualizada $c(x_k, u_k, x_{k+1}) + \gamma V_k(x_{k+1})$ da função valor V^g 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 Equação de Bellman se as estimativas de valor geradas pelo algoritmo de Equação (2.19) 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.

2.4 Críticos Adaptativos

Os Projetos Críticos Adaptativos fornecem procedimentos "para frente no tempo"para determinação de politicas ótimas que podem ser implementadas em tempo real. Esta abordagem utiliza duas estruturas paramétricas chamadas o **ator** e o **crítico** .

O ator consiste de uma política parametrizada. O crítico aproxima uma função relacionada a valor e capta o efeito que a política exercerá sobre o custo futuro.

Em um determinado momento o crítico fornece orientações sobre a forma de melhorar a política. O ator, por sua vez, pode ser usado para atualizar o crítico. Um esquema que sucessivamente itera entre estas duas operações levará a uma política ótima, ao longo do tempo.

Mais importante ainda, o crítico adaptativo pode ser usado para projetar sistemas de controle que melhoram seu desempenho online, sujeitos a dinâmica real da planta. Na prática, as estruturas paramétricas que representam a lei de controle (política de controle) e a função de custo total (função valor ótimo) se adaptam aos parâmetros variantes do sistema, incluindo falhas do sistema, sem identificação explicita dos parâmetros.

Um esquema de funcionamento da estrutura ator-crítico pode ser observado na Figura 2.3, representando o sistema de controle com realimentação no contexto de controle adaptativo.

Figura 2.3: Estrutura de Aprendizado Ator-Crítico

2.5 Controle Ótimo

Nesta seção, os fundamentos da programação dinâmica aproximada, denominada crítica adaptativa, são apresentados no quadro de controle ótimo. A solução online dos problemas de horizonte infinito, com a informação que se torna disponível de forma incremental ao longo do tempo, é enfatizada. Em problemas de controle ótimo, o objetivo é elaborar uma estratégia de ação, ou lei de controle, que otimiza uma métrica de desempenho desejado (por exemplo, o custo). Duas abordagens de solução bem conhecidas são: o cálculo variacional, envolvendo as equações de Euler-Lagrange, e a programação dinâmica para trás. Elas são apre-

sentadas, nesta seção, devido ambas terem fortes relações com a abordagem crítica adaptativa.

2.5.1 Cálculo Variacional

O objetivo é determinar uma lei de controle ótimo estacionária que minimiza uma medida de desempenho J expressa por uma função integral escalar invariante no tempo do estado e controle, e por um custo final escalar (Bryson and Ho 1975) (Kirk 2004),

$$J = \varphi(x(t_f)) + \sum_{t_k=t_0}^{t_f-1} \mathcal{L}[x(t_k), u(t_k)], \tag{2.21}$$

sujeita à restrições impostas pela dinâmica da planta. Esta função de custo representa o custo da operação, quando ele se acumula do tempo inicial t_0 até o tempo final t_f. O lagrangiano $\mathcal{L}[.]$ é o custo associado com um incremento de tempo. Ele também é conhecido como utilidade ou recompensa, em que o objetivo é tipicamente maximizar o desempenho global, Equação (2.21). A dinâmica da planta é discreta, invariante no tempo, e determinística, e ela pode ser modelada por uma equação a diferença da forma:

$$x(t_{k+1}) = f(x(t_k), u(t_k)), \tag{2.22}$$

com incrementos de tempo igualmente divididos no intervalo $t_0 \leq t_k \leq t_f$, e condição inicial $x(t_0)$. x é o vetor de estado $n \times 1$ da planta e u é o vetor de controle $m \times 1$. Para simplificar, também admite-se que o estado é totalmente observável e que as medições de saída estão perfeitamente disponíveis.

A lei de controle g é suposta como sendo apenas uma função do estado x, ou seja,

$$u = g(x), \tag{2.23}$$

a qual pode conter funções de seus argumentos, tais como integrais e derivadas, e a forma ótima é representada por $g^*(x)$. Em qualquer momento no tempo t_k, o custo que vai se acumulando a partir desse momento em diante pode ser expressa por uma **função valor**,

$$V = \varphi[x(t_f)] + \sum_{t_k}^{t_f-1} \mathcal{L}[x(t_k), u(t_k)] = V[x(t_k), g(x)], \tag{2.24}$$

que depende do valor atual do estado, $x(t_k)$, e da política de controle escolhida $g(x)$. Portanto, $V[x(t_k), g(x)]$ também pode ser escrita na forma abreviada como $V[x_k, g]$, onde $x_k \equiv x(t_k)$.

Uma abordagem para a otimização da Equação (2.21) sujeito a Equação (2.22) é aumentar a função de custo por meio da equação dinâmica, reformulando o problema em termos do Hamiltoniano

$$\mathcal{H}(x_k, u_k, \lambda_k) = \mathcal{L}[x(t_k), u(t_k)] + \lambda^T(t_k) f[x(t_k), u(t_k)], \tag{2.25}$$

em que λ é um vetor de co-estado ou vetor adjunto que contém os multiplicadores de Lagrange e representa a sensibilidade do custo a perturbações de estado sobre a trajetória ótima; pode-se provar que $\lambda(t_f) = \partial\varphi/\partial x|_{t_f}$ (com o gradiente definido como um vetor coluna)(Ferrari *et al.* 2008). Quando o tempo final for fixado, condições necessárias para a otimalidade, as equações de Euler-Lagrange, podem ser obtidas por diferenciação do Hamiltoniano em relação ao estado e ao controle.

Assim, o problema de otimização dinâmica é reduzido a um problema de valor de fronteira com dois-pontos, em que o estado e o vetor adjunto são especificados no tempo inicial e final, respectivamente. Por fim, condições necessárias e suficientes para otimalidade são fornecidas pelo princípio do mínimo de Pontryagin, declarando que sobre a trajetória ótima, $\mathcal{H}$ deve ser estacionário e convexa

$$\mathcal{H}^* = \mathcal{H}(x_k^*, u_k^*, \lambda_k^*) \leq \mathcal{H}(x_k^*, u_k, \lambda_k^*). \tag{2.26}$$

2.5.2 Programação Dinâmica para trás

Outra abordagem para resolver o problema de controle ótimo consiste em incorporar a minimização da função custo, Equação (2.21), na minimização da função valor, Equação (2.24). Quando o sistema está em um estado admissível x_k, a função valor, isto é, o custo da operação do instante t_k para o tempo final t_f, pode ser escrita como

$$V(x_k, u_k, ..., u_{f-1}) = \mathcal{L}\left(x_k, u_k\right) + V(x_{k+1}, u_{k+1}, ..., u_{f-1}). \tag{2.27}$$

x_{k+1} depende de x_k e u_k por meio da Equação (2.22). Todos os valores subsequentes do estado podem ser determinados a partir de x_k e da história de controle escolhida, $u_k, ..., u_{f-1}$. Portanto, o custo da operação de t_k em diante pode ser minimizada em relação a todos os valores futuros do controle

$$V^*(x_k^*) = \min_{u_k,...,u_{f-1}} \left\{ \mathcal{L}\left(x_k^*, u_k\right) + V(x_{k+1}^*, u_{k+1}, ..., u_{f-1}) \right\}. \tag{2.28}$$

Daí, resulta que a função valor ótimo depende do estado atual do sistema e é independente de qualquer historia anterior. Suponha que uma política seja ótima ao longo de um intervalo de tempo $(t_f - t_k)$. Então, pelo Princípio da Otimalidade, quaisquer que sejam o estado inicial (x_k) e a decisão (u_k), as decisões restantes também devem constituir uma política ótima com respeito ao estado x_{k+1}, ou seja,

$$V^*(x_k^*) = \min_{u_k} \{ \mathcal{L}(x_k^*, u_k) + V^*(x_{k+1}^*) \}. \tag{2.29}$$

A função valor pode ser minimizada somente em relação ao valor atual do controle, u_k, desde que, o custo futuro da operação, $V(x_{k+1}, u_{k+1}, ..., u_{f-1})$, seja ótimo (neste caso, ele depende somente do próximo valor de estado sobre a trajetória ótima, isto é, x_{k+1}^*). A Equação (2.29) constitui a relação de recorrência da programação dinâmica isto é para o caso determinístico. A função valor ótimo pode ser interpretada como o custo mínimo para o período de tempo que resta depois de um instante t_k em um processo que começou no momento t_0, isto é, $(t_f - t_k)$. Alternativamente, ela pode ser vista como o custo mínimo para um processo que se inicia no instante t_k e termina no instante t_f.

A relação de recorrência pode ser usada "para trás no tempo", iniciando-se a partir de t_f, para se obter uma solução aproximada para a historia de controle ótimo exata. Esta abordagem é conhecida como Programação Dinâmica para Trás. Ela discretiza o espaço de estado e faz uma comparação direta entre o custo associado a todas as trajetórias possíveis, garantindo uma solução para o problema de controle ótimo global. O espaço de soluções admissíveis é reduzido pela análise de um processo de decisão multi-estágio como uma sequência de processos a um só estágio. Esta abordagem é geralmente muito dispendiosa computacionalmente para os sistemas de dimensões mais elevadas, com um grande número de estágios (ou incrementos de tempo). A geração múltipla necessária e expansão do estado,

e o armazenamento de todos os custos ótimos levam a um número de cálculos que cresce exponencialmente com o número de variáveis de estado. Este fenômeno é comumente referido como a "maldição da dimensionalidade".

Figura 2.4: Abordagem de Programação Dinâmica para Trás

2.5.3 Programação Dinâmica para Frente

A Programação Dinâmica para Frente e a diferença temporal são métodos que usam otimização incremental combinada com uma estrutura paramétrica para reduzir a complexidade computacional associada com a avaliação do custo. Uma estrutura paramétrica consiste em uma relação funcional cujos parâmetros ajustáveis permitem aproximar diferentes mapeamentos. Projetos crítico adaptativos (ACD) são derivados da abordagem de Programação Dinâmica para Frente, também chamada programação dinâmica aproximada.

Figura 2.5: Abordagem de Programação Dinâmica para Frente

Para efeito de comparação, as abordagens DP para trás e para frente são ilustradas para os dois últimos estágios de um processo hipotético nas Figuras 2.4 e

2.5, respectivamente. A abordagem "para trás no tempo"começa considerando-se o último estágio. Uma vez que o estado final ótimo ainda não é conhecido, o seguinte procedimento deve ser realizado para todos os valores admissíveis do estado final. As trajetórias ótimas são computadas para todos os estados-intermediários admissíveis , c, d, e e, produzindo, assim, os custos ótimos V^*_{cf}, V^*_{df} e V^*_{ef}, respectivamente. Pelo princípio da otimalidade essas trajetórias também são ótimas para o último estagio das trajetórias ótimas que vão de b para f através dos respectivos valores de estado, por exemplo, $V^*_{b(c)f} = V_{bc} + V^*_{cf}$. Desta forma, se os custos do último estagio, V^*_{cf}, V^*_{df} e V^*_{ef}, são armazenados, em seguida, os custos totais $V^*_{b(c)f}$, $V^*_{b(d)f}$, e $V^*_{b(e)f}$ podem ser comparados para encontrar a trajetória ótima a partir de b para f. Por um processo com mais de dois estágios, o estado b seria desconhecido. Daí, o mesmo cálculo seria realizado para todos os valores possíveis de estado, tal como g na Figura 2.4, a fim de determinar V^*_{bf}, V^*_{gf}, e assim por diante. O algoritmo termina quando o estágio inicial é alcançado, em que o estado é conhecido a partir das condições iniciais.

Ao contrário da DP para trás, os algoritmos DP para frente progridem para frente no tempo, eles fazem a aproximação do controle ótimo e do custo futuro considerando apenas o valor atual do estado. Supondo que a seja o estado inicial de um processo de dois estágios ou, de forma equivalente, o estado de um processo em andamento do segundo ao último estágio, como esboçado na Figura 2.5. Então, a é conhecido e o custo V_{ab} pode ser calculado a partir do Lagrangiano para uma predita política de controle selecionada. O custo ótimo sobre todos os estágios futuros,V^*_{bf}, é previsto por um aproximador função ou estrutura paramétrica. Daí, a soma $(V_{ab} + \tilde{V}^*_{bf})$ pode ser minimizada em relação ao valor atual do controle, de acordo com a relação de recorrência da programação dinâmica na Equação (2.29). No estágio seguinte $b-c$ este procedimento é repetido. A aproximação do custo, denotada por $\tilde{V}$, foi melhorada com base na informação coletada durante o primeiro estágio. Portanto, a próxima trajetória, a partir de c para f, está mais próximo da trajetória ótima.

Projetos críticos adaptativos reproduzem a solução mais geral do FDP por derivação das relações de recorrência para a política ótima, do custo, e, possivelmente, as suas derivadas. O objetivo é superar a maldição da dimensionalidade, assegurando ao mesmo tempo convergência para uma solução quase-ótima, ao

longo do tempo.

2.6 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 Equação (2.14) a da avaliação de política e a Equação (2.13) 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 online 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

Projeto de Controle Ótimo DLQR via Programação Dinâmica Heurística

3.1 Introdução

A otimalidade dos controladores é garantida por meio da solução da equação de Riccati. Através de uma discretização do sistema dinâmico, o DLQR pode ser caracterizado como um problema de possíveis estágios e estados, tendo-se a possibilidade em estabelecer um caminho das trajetórias através da Programação Dinâmica.

A Programação Dinâmica como solução do problema de controle ótimo, é um procedimento "para trás no tempo", sendo utilizado para planejamento offline. A Equação de Bellman conduz a vários métodos iterativos para aprender a solução do controle ótimo sem ter que resolver diretamente a equação de Hamilton-Jacobi-Bellman (HJB).

Diversas contribuições para Aprendizado por Reforço são feitas por meio da Aproximação da Função Valor. Na aprendizagem da Função Valor por métodos de RL, é necessário armazenar o valor ótimo e o controle ótimo em função do vetor de estado $x \in R^n$. No Processo de Decisão Markoviano, o estado deve assumir apenas um número finito de valores discretos definidos (para uma dada quantização). A proporção que a quantidade de valores cresce, o que pode ocorrer devido

ao refinamento da quantização e/ou aumento da quantidade de variáveis, mais informações devem ser armazenadas, normalmente em forma de tabelas, o que torna o problema computacionalmente intratável. Bellman denominou esse problema de "Maldição da Dimensionalidade"(Bellman 2003). No entanto, utilizando a aproximação da Função Valor, onde o crítico e, se desejar, o ator são parametrizados usando aproximadores de função, este problema é contornado (Lewis and Vrabie 2009*b*).

Neste capítulo, será visto como formular estes procedimentos em um método de Aprendizado por Reforço usando-se dados do sistema ao longo de sua trajetória, utilizando a Programação Dinâmica Heurística, onde o crítico estima a função valor baseado diretamente do estado da planta, ou seja, o crítico não necessita do modelo da planta para o cálculo da função valor. Já o treinamento do controlador necessita realizar a transição de estado a cada iteração. Assim, o algoritmo HDP utiliza o modelo da planta somente para a atualização do controlador.

3.2 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

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

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 $u_{(\cdot)}$ para controlar o sistema de Equação (3.1) é estabelecida para tentar fazê-lo tão próximo quanto possível de uma trajetória desejada de estado, a qual é dada por uma função prescrita $\widetilde{x}_{(\cdot)}$ 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, $k \in [i, N]$, que minimiza o índice de desempenho (função de custo) dado por

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

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

Um dos resultados mais marcantes em teoria de controle ótimo linear quadrático é que se a otimização é sobre um horizonte de tempo infinito, a lei de controle ótima resultante fornece boas propriedades, incluindo estabilidade de malha fechada. Esses resultados estão intrinsecamente relacionados à propriedades de estabilidade e detectabilidade do sistema. Para mais discussão sobre este tópico, veja (Bryson and Ho 1975) e (Anderson and Moore 1990).

3.2.1 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

$$V_i = \frac{1}{2} x_N^T P_N x_N + \frac{1}{2} \sum_{k=i}^{N-1} (x_k^T Q_k x_k + u_k^T R_k u_k), \tag{3.3}$$

e tem como restrição a Equação de Estado (3.1), 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 (3.3) 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

$$V_N^* = \frac{1}{2} x_N^T P_N x_N, \tag{3.4}$$

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

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

$$V_{N-1} = \frac{1}{2} x_{N-1}^T Q_{N-1} x_{N-1} + \frac{1}{2} u_{N-1}^T R_{N-1} u_{N-1} + \frac{1}{2} x_N^T P_N x_N. \tag{3.5}$$

De acordo com a Equação (2.29), é preciso se determinar u_{N-1}^* por minimização da Equação (3.5). Dessa forma, usando-se a equação de estado, tem-se

$$\begin{aligned} V_{N-1} = {} & \tfrac{1}{2} x_{N-1}^T Q_{N-1} x_{N-1} + \tfrac{1}{2} u_{N-1}^T R_{N-1} u_{N-1} \\ & + \tfrac{1}{2} (A_{N-1} x_{N-1} + B_{N-1} u_{N-1})^T P_N \left(A_{N-1} x_{N-1} + B_{N-1} u_{N-1}\right). \end{aligned} \tag{3.6}$$

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

$$0 = \frac{\partial V_{N-1}}{\partial u_{N-1}} = R_{N-1} u_{N-1} + B_{N-1}^T P_N (A_{N-1} x_{N-1} + B_{N-1} u_{N-1}). \tag{3.7}$$

Resolvendo-se para o controle ótimo, tem-se

$$u_{N-1}^* = -\left(B_{N-1}^T P_N B_{N-1} + R_{N-1}\right)^{-1} B_{N-1}^T P_N A_{N-1} x_{N-1}. \tag{3.8}$$

Definindo

$$K_{N-1} = \left(B_{N-1}^T P_N B_{N-1} + R_{N-1}\right)^{-1} B_{N-1}^T P_N A_{N-1}, \tag{3.9}$$

pode-se escrever

$$u_{N-1}^* = -K_{N-1} x_{N-1}. \tag{3.10}$$

O custo ótimo no instante $k = N - 1$ é encontrado substituindo-se a Equação (3.10) na Equação (3.6) que após simplificações, resulta

$$V^*_{N-1} = \tfrac{1}{2} x^T_{N-1} \Big[(A_{N-1} - B_{N-1}K_{N-1})^T P_N (A_{N-1} - B_{N-1}K_{N-1}) + K^T_{N-1} R_{N-1} K_{N-1} + Q_{N-1} \Big] x_{N-1}. \tag{3.11}$$

Definindo

$$P_{N-1} = (A_{N-1} - B_{N-1}K_{N-1})^T P_N (A_{N-1} - B_{N-1}K_{N-1}) + K^T_{N-1} R_{N-1} K_{N-1} + Q_{N-1}, \tag{3.12}$$

a Equação (3.11) pode ser escrita como

$$V^*_{N-1} = \frac{1}{2} x^T_{N-1} P_{N-1} x_{N-1}. \tag{3.13}$$

Agora reduz-se para $k = N - 2$. Então

$$V_{N-2} = \frac{1}{2} x^T_{N-2} Q_{N-2} x_{N-2} + \frac{1}{2} u^T_{N-2} R_{N-2} u_{N-2} + \frac{1}{2} x^T_{N-1} P_{N-1} x_{N-1} \tag{3.14}$$

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}, minimiza-se (3.14).

Observe que (3.14) é da mesma forma que (3.5). O controle ótimo e o custo ótimo no instante $k = N - 2$ são portanto dados por (3.9), (3.10), (3.12) e (3.13) 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

$$K_k = \left(B^T_k P_{k+1} B_k + R_k \right)^{-1} B^T_k P_{k+1} A_k, \tag{3.15}$$

$$u^*_k = -K_k x_k, \tag{3.16}$$

$$P_k = (A_k - B_k K_k)^T P_{k+1} (A_k - B_k K_k) + K^T_k R_k K_k + Q_k, \tag{3.17}$$

$$V_k^* = \frac{1}{2} x_k^T P_k x_k, \tag{3.18}$$

sendo a condição final P_N para (3.17) é dada em (3.3). 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 Equação (3.17). Se o ganho K_k é dado por uma sequência arbitrária de ganhos estabilizantes, então (3.17) é simplesmente uma equação de *Lyapunov*, a qual é linear em P_k. Por outro lado, se a sequência de ganhos ótimos de Equação (3.15) for aplicada na Equação (3.17), esta torna-se a equação matricial de *Riccati* em sua versão estabilizada de *Joseph*. Não é difícil verificar que, substituindo-se (3.15) em (3.17) e após transformações, obtém-se a forma

$$P_k = Q_k + A_k^T P_{k+1} A_k - A_k^T P_{k+1} B_k (R_k + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k. \tag{3.19}$$

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

$$P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A, \tag{3.20}$$

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

$$x_{k+1} = (A - BK_\infty)x_k \tag{3.21}$$

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

$$K_\infty = (B^T PB + R)^{-1} B^T PA. \tag{3.22}$$

3.3 Programação Dinâmica Heurística (HDP)

A Programação Dinâmica Heurística, que inclui os métodos TD, é um método para estimar a função valor V. Se a política não é fixa, mas atualizada com a iteração de política ou com a iteração gulosa, a função valor correspondente à política ótima pode ser estimada. Estimar a função valor para uma determinada política exige apenas amostras da função de custo instantâneo c, enquanto os modelos do ambiente e o custo instantâneo são necessários para encontrar a função valor correspondente à política ótima. Assim, a função valor para uma política g pode ser definida recursivamente como

$$V(x) = c(x, g(x)) + V(f(x, g(x))). \tag{3.23}$$

Pode-se usar o lado direito desta equação como alvo d para um esquema iterativo baseado em qualquer algoritmo de aprendizado supervisionado que tenta aproximar V por um modelo parametrizado $V(x, w)$ (Landelius and Knutsson 1996)

$$d(x, c, f, w) = c(x, g(x)) + V(f(x, g(x)), w). \tag{3.24}$$

Dado uma estimativa w_i, do vetor de parâmetro ótimo, se inicia a busca para um novo vetor de parâmetro w_{i+1} que minimiza o erro quadrático esperado entre o modelo e o alvo:

$$w_{i+1} = arg \min_w E_x \left\{ |V(x, w) - d(x, c, f, w_i)|^2 \right\}. \tag{3.25}$$

Uma vez que um novo vetor de parâmetros define novos alvos, ocasiona-se um problema de alvo móvel que dá origem a um esquema iterativo para determinar o parâmetro correspondente à política ótima.

No entanto, começa-se por determinar como parametrizar a política $g(x) = g(x, \kappa)$ e determinar os parâmetros de resposta apropriados de acordo com a equação da Otimalidade de Bellman usando a estimativa de V:

$$\kappa^* = \arg\min_{\kappa} \left\{ c(x, g(x, \kappa)) + V(f(x, g(x, \kappa)), w) \right\}. \tag{3.26}$$

Uma maneira de buscar os parâmetros ótimos é empregar um algoritmo de otimização baseado no gradiente $\partial V_{g(\kappa)}(x)/\partial\kappa$. Nestes cálculos, o requisito de um número de modelos torna-se evidente:

$$\frac{\partial V}{\partial \kappa} = \frac{\partial c}{\partial g}\frac{\partial g}{\partial \kappa} + \frac{\partial V}{\partial f}\frac{\partial f}{\partial g}\frac{\partial g}{\partial \kappa}. \tag{3.27}$$

Precisa-se de modelos de três derivadas, além do modelo conhecido da derivada da política g atual em relação aos seus parâmetros κ. Os três modelos são as derivadas do custo instantâneo c em relação à resposta g, aquela da função valor V com respeito ao próximo vetor de estado f, e também a derivada do ambiente f em relação à resposta g. Estas derivadas podem ser obtidos, por exemplo por diferenciação de modelos parametrizados das funções $c(x, u)$, $V(x)$, e $f(x, u)$.

3.4 Solução do DLQR via HDP

Considere o problema do DLQR e estabeleça a parametrização das diferentes funções envolvidas, de acordo com,

$$V(x) = x^T P x \tag{3.28}$$

$$c(x, u) = x^T Q x + u^T R u \tag{3.29}$$

$$f(x, u) = Ax + Bu \tag{3.30}$$

$$u = g(x) = Kx \tag{3.31}$$

Os parâmetros, Q e R, do custo instantâneo c e os parâmetros, A e B, do ambiente f podem ser obtidos utilizando qualquer algoritmo de identificação padrão. Os

parâmetros da função valor podem ser estimados pela minimização iterativa do problema de alvo móvel na Equação (3.25) através da iteração gulosa (Landelius 1997).

Agora, usa-se as expressões para as funções parametrizadas nas Equações (3.28), (3.29), (3.30) e (3.31) para se obter as derivadas necessárias na Equação (3.27). No caso do DLQR, obtém-se o mínimo da Equação (3.26), resolvendo $\partial V/\partial \kappa = 0$ para κ. Uma vez que tem-se $\partial g/\partial \kappa \neq 0$, pode-se simplificar esta equação para produzir:

$$\begin{aligned} 0 &= \left(\frac{\partial c}{\partial g} + \frac{\partial V}{\partial f}\frac{\partial f}{\partial g}\right)\frac{\partial g}{\partial \kappa} = \frac{\partial r}{\partial g} + \frac{\partial V}{\partial f}\frac{\partial f}{\partial g} \\ &= 2u^T R + 2(Ax + Bu)^T PB \\ &= u^T(R + B^T PB) + x^T A^T PB \end{aligned} \tag{3.32}$$

A segunda linha vem do fato que $g = u$ e que ao estado seguinte, $f = Ax + Bu$, é atribuído o valor $V(f) = f^T Pf$ o que significa $\partial V/\partial f = 2f^T P$. A equação acima fornece a solução para a resposta apropriada por:

$$u = g(x, \kappa) = -(R + B^T KB)^{-1} B^T PAx = Kx, \tag{3.33}$$

Observa-se que o vetor de parâmetros κ pode ser visto como a versão vetorizado da matriz K, ou seja, $k = vec(K)$, sendo vec a função de vetorização (Brewer 1978). Note-se também que a parametrização de $V(P)$; $c(Q, R)$, e $f(A, B)$ induz uma parametrização da política $g(P, R, A, B)$.

Se todos os parâmetros são conhecidos, a matriz P^*, correspondente à política ótima, pode ser determinada como a única solução semidefinida positiva de uma equação não linear de Riccati. Esta equação de Riccati é obtida através da inserção da expressão para a política ótima na Equação (3.33) dentro da Equação (3.24). Após reordenamento dos termos, o resultado é:

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

Observa-se da Equação (3.32) que, dado o modelo de V, apenas duas matrizes adicionais precisam ser estimadas, $\partial c/\partial u$ e $\partial f/\partial u$. Usando-se uma abordagem crítica adaptativa gulosa, os parâmetros da função valor ótimo podem ser estimados.

Voltando para o esquema iterativo baseado na minimização da Equação (3.25) com relação a P_{i+1}, a matriz correspondente ao novo vetor parâmetro w_{i+1} é dado por:

$$P_{i+1} = \arg\min_{P} E_x \left\{ \left| x^T P x - d(x, c, f, P_i) \right|^2 \right\}.$$

Antes de prosseguir deve-se introduzir dois conceitos necessários para o desenvolvimento algébrico:

1. Para um vetor $x \in \mathbb{R}^n$, defina o vetor $\bar{x} = (x_1^2, ..., x_1 x_n, x_2^2, ..., x_{n-1} x_n, x_n^2)^T$, ou seja, as componentes do vetor $\bar{x}$ correspondem a funções quadráticas sobre os elementos de x. A dimensão do vetor $\bar{x}$ torna-se então $n(n+1)/2$.

2. Uma função $v(M)$ para matrizes quadradas $M \in \mathbb{R}^n$. A função $v(M)$ é um vetor que contém $n(n+1)/2$ somas $M_{ij} + M_{ji}$. Os elementos nos vetores $\bar{x}$, $x \in \mathbb{R}^{n(n+1)/2}$, definido anteriormente, e $v(M)$ é ordenado de modo que sua forma quadrática $x^T H x$ pode ser escrito como $\bar{x}^T v(M)$. Note que se M é simétrica, ela pode ser recuperada a partir dos componentes de $v(M)$

Com as notações $\bar{x}$ e $\theta = v(P)$, esta regra de atualização torna-se:

$$\theta_{i+1} = \arg\min_{\theta} E_x \left\{ \left| \bar{x}^T \theta - d(x, c, f, P_i) \right|^2 \right\}. \tag{3.35}$$

Isto define uma função de erro quadrático em termos do operador linear θ, o que significa que existe um único mínimo dado pela solução de mínimos quadrados padrão:

$$\theta_{i+1} = \left(E_x \left\{ \bar{x} \bar{x}^T \right\} \right)^1 E_x \left\{ \bar{x} d\left(x, c, f, P_i\right) \right\}. \tag{3.36}$$

O alvo para o algoritmo de aprendizagem HDP foi definido na Equação (3.24). Usando a parametrização do sistema dado pelas Equações (3.28), (3.29), (3.30) e (3.31), o alvo pode ser expresso como:

$$\begin{aligned} d\left(x, c, f, P_i\right) &= c(x, u) + V(f(x, g(x))) \\ &= x^T Q x + u^T R u + f^T P_i f \\ &= x^T Q x + u^T R u + (Ax + Bu)^T P_i \left(Ax + Bu\right). \end{aligned} \tag{3.37}$$

Note que toda a informação necessária para produzir alvos para a HDP é o custo atual e a função valor avaliados no estado que o sistema entra após a resposta atual ter sido aplicada.

A inserção da expressão acima para o alvo d dentro da Equação (3.36) juntamente com a resposta gulosa $u = K(P_i)x$ fornece:

$$\begin{aligned}\theta_{i+1} &= (E_x\{\bar{x}\bar{x}^T\})^{-1}E_x\{\bar{x}(x^T(Q + K^T(P_i)RK(K_i))x \\ &\quad + x^T(A + BK(P_i))^T P_i(A + BK(P_i))x)\} \\ &= (E_x\{\bar{x}\bar{x}^T\})^{-1}E_x\{\bar{x}(x^T(Q + K^T(P_i)RK(P_i)) \\ &\quad + (A + BK(P_i))^T K_i(A + BK(P_i)))\} \\ &= (E_x\{\bar{x}\bar{x}^T\})^{-1}E_x\{\bar{x}\bar{x}^T\}v(f(K_i, K(P_i))) \\ &= v(f(P_i, K(P_i))).\end{aligned} \tag{3.38}$$

Uma vez que a matriz correspondente a θ_{i+1} é simétrica, este esquema iterativo pode ser redeclarado na forma matricial:

$$\begin{aligned}P_{i+1} &= f(P_i, K(P_i)) \\ &= Q + K^T(P_i)RK(P_i) + (A + BK(P_i))^T P_i(A + BK(P_i)).\end{aligned} \tag{3.39}$$

Este esquema pode-se ser provado convergir para o parâmetro ótimo P^*, o qual define a única solução positiva semi-definida para a Equação de Riccati (3.34), sob a seguintes suposições: $P_0 \geq 0$, $R > 0$, $Q \geq 0$, (A, B) é um par controlável, e que a resposta é gerada de acordo com a Equação (3.33) (Landelius 1997).

Os valores esperados que aparecem na regra de atualização (3.36) não são obtíveis em uma implementação prática. No entanto, uma vez que os valores esperados anulam-se uns aos outros na Equação (3.38), isto é suficiente para fazer uso das duas seguintes somas:

$$\Phi_i = \sum_{j=j_{i-1}}^{j_i} \bar{x}_j \bar{x}_j^T, \tag{3.40}$$

$$z_i = \sum_{j=j_{i-1}}^{j_i} \bar{x}_j d(x_j, c, f, P_i) \tag{3.41}$$

Uma vez que há liberdade para a escolha das amostras x_j, não deve ser um problema obter uma matriz não singular $C_i = \sum \bar{x}\bar{x}^T$ a partir de $j_i - j_{i-1} = \dim(\bar{x})$ amostras linearmente independentes.

Isto significa que pode-se implementar o esquema de iteração como a multiplicação entre o inverso da matriz de covariância não singular calculada C_i e vetor q_i de acordo com:

$$\begin{aligned}\theta_{i+1} &= \Phi_i^{-1} z_i \\ &= \left(\sum \bar{x}_j \bar{x}_j^T\right)^{-1} \sum \bar{x}_j d(x_j, c, f, P_i \\ &= \left(\sum \bar{x}_j \bar{x}_j^T\right)^{-1} \sum \bar{x}_j [x_j^T (Q + K^T(P_i) R K(P_i)) x_j \\ &\quad + x_j^T (A + BK(P_i))^T P_i (A + BK(P_i)) x_j] \\ &= \left(\sum \bar{x}_j \bar{x}_j^T\right)^{-1} \left(\sum \bar{x}_j \bar{x}_j^T\right) v(Q + K^T(P_i) R K(P_i) \\ &\quad + (A + BK(P_i))^T P_i (A + BK(P_i))) \\ &= v(f(P_i, K(P_i))).\end{aligned} \tag{3.42}$$

O resultado é exatamente o mesmo esquema iterativo como obtido em (3.38), a partir da minimização da função de erro na Equação (3.25).

Uma vez que o esquema na Equação (3.38) converge, o mesmo acontece com o esquema iterativo dada pela Equação (3.42). O limite é determinado pela matriz, P^*, que constitui a função valor ótimo para o sistema linear quadrático descrito nas Equações (3.28), (3.29), (3.30) e (3.31).

Para melhor apresentar a solução DLQR apresenta-se o fluxograma do algoritmo DLQR-HDP na Figura 3.1.

3.5 Considerações Finais

Neste capítulo, o método de programação dinâmica heurística, que esta inserido na abordagem crítica adaptativa, foi apresentado para a determinação de soluções aproximadas da equação de Hamilton-Jacobi-Bellman e políticas de decisão ótimas para viabilizar o desenvolvimento de um projeto online de sistema de controle ótimo.

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 valor e sistema dinâmico montam um quadro para a solução do problema DLQR.

Figura 3.1: O algoritmo DLQR-HDP

Capítulo 4

Algoritmos RLS-HDP-DLQR e Decomposição $\mathcal{QR}$

4.1 Introdução

Em metodologias de projeto de controle baseadas em ADP, dentre os algoritmos iterativos propostos para estimar os parâmetros da função valor, o aprendizado dos mínimos quadrados recursivos (RLS) é um dos mais bem-sucedidos (Xu *et al.* 2002). Tal resultado favorável é principalmente devido à sua robustez para lidar com variações no tempo dos parâmetros de regressão e a rápida convergência quando comparado com os métodos de gradiente estocástico. Porém, problemas de convergência relacionados a instabilidade numérica podem ocorrer devido ao mal condicionamento da matriz de covariância da abordagem RLS para aproximações de função valor (Liu and Balakrishnan 2000).

Neste capítulo serão apresentados a caracterização e a formulação do problema de estimação RLS para a solução da equação HJB-Riccati associada com o problema DLQR via HDP. Resultando, assim, no algoritmo de Controle Ótimo Adaptativo RLS$_\lambda$-HDP-DLQR. Entretanto, devido à problemas de estabilidade numérica, a solução para a política de decisão ótima para um dado ponto de operação pode não convergir mesmo existindo uma solução.

Para contornar esse problema, modifica-se o algoritmo padrão com a substituição do estimador RLS convencional pelo estimador ortogonal $\mathcal{QR}$-RLS, formulando-se assim o algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR, que é o algoritmo proposto neste tra-

balho. 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 decomposição $\mathcal{QR}$.

Golub (Golub 1965) sugeriu a decomposição $\mathcal{QR}$ para resolver problemas dos mínimos quadrados (LS) empregando a transformação de Householder. O uso de rotações de Givens juntamente com a decomposição $\mathcal{QR}$ para resolver problemas LS foi abordado por Gentleman (Gentleman 1973), que foi um dos pioneiros no uso da decomposição $\mathcal{QR}$ para resolver problemas (RLS) (Gentleman and Hung 1981). McWhirter (McWhirter 1983*a*) (McWhirter 1983*b*) e mais tarde Ward (Ward 1986) propuseram implementações de filtragem adaptativa do algoritmo RLS para resolver a equação de minimização do erro quadrático. Eles aproveitaram as rotações de Givens para a triangularização da matriz de dados e sugeriram uma implementação sistólica, que é interessante por se tratar de um método de paralelismo das rotações de Givens.

4.2 Algoritmo RLS$_\lambda$-HDP-DLQR

Considere o problema dos mínimos quadrados (Least-Squares - LS) associado com a aproximação da função valor V do problema DLQR, expressa na forma da Equação (3.28), com a seguinte função de perda:

$$\mathcal{J}(i) = \sum_{j=j_{i-1}}^{j_i} \lambda^{j_i - j} \left| e(j)^2 \right|, \tag{4.1}$$

sendo $e(j) = d(j) - \theta^H(j_i)\bar{x}(j)$, λ é uma contante positiva, $0 < \lambda < 1$, e o alvo desejado $d(j)$ definido na Equação (3.37) e $\bar{x} \in \Re^{n(n+1)/2}$ definido de acordo com o produto de Kronecker que é dado por. $\bar{x}^T = [x_1^2 \ldots x_1 x_n \;\; x_2^2 \ldots x_{n-1} x_n \;\; x_n^2]$.

O valor ótimo do vetor de parâmetro $\hat{\theta}(i)$, para o qual a função de perda $\mathcal{J}(i)$ da Equação (4.1) atinge o seu valor mínimo, é definido pela equação normal, escrita em forma matricial por:

$$\Phi(i)\hat{\theta}(i) = z(i), \tag{4.2}$$

em que $\Phi(i)$ é a matriz de correlação, $n_\theta \times n_\theta$, a qual é definida por:

$$\Phi(i) = \sum_{j=j_{i-1}}^{j_i} \lambda^{j_i - j} \bar{x}(j) \bar{x}^H(j), \tag{4.3}$$

e $z(i)$ é o vetor de correlação cruzada, $n_\theta \times 1$, entre as entradas do estimador LS e a resposta desejada é definida por:

$$z(i) = \sum_{j=j_{i-1}}^{j_i} \lambda^{j_i - j} \bar{x}(j) d^*(j), \tag{4.4}$$

onde o asterisco representa o conjugado complexo.

Isolando o termo correspondente a $j = j_i$ do restante do somatório no lado direito da Equação (4.3), pode-se escrever:

$$\Phi(i) = \lambda \left[\sum_{j=j_{i-1}}^{j_i - 1} \lambda^{j_i - 1 - j} \bar{x}(j) \bar{x}^H(j) \right] + \bar{x}(i) \bar{x}^H(i). \tag{4.5}$$

No entanto, por definição, a expressão dentro dos colchetes no lado direito da Equação (4.5) é igual a matriz de correlação $\Phi(i-1)$. Assim, tem-se a seguinte recursão para atualização do valor da matriz de correlação das entradas:

$$\Phi(i) = \lambda \Phi(i-1) + \bar{x}(i) \bar{x}^H(i), \tag{4.6}$$

onde $\Phi(i-1)$ é o valor anterior da matriz de correlação, e o produto matricial $\bar{x}(i)\bar{x}^H(i)$ desempenha o papel de um termo de "correção"na operação de atualização.

Da mesma forma, pode-se usar a Equação (4.4) para obter a seguinte recursão para a atualização do vetor de correlação cruzado entre as entradas e a resposta desejada

$$z(i) = \lambda z(i-1) + \bar{x}(i) d^*(i). \tag{4.7}$$

Para calcular a estimativa dos mínimos quadrados $\hat{\theta}(i)$ para o vetor de parâmetros de acordo com a Equação (4.2), deve-se determinar a inversa da matriz de correlação $\Phi(i)$. Na prática, no entanto, geralmente tenta-se evitar a execução de tal operação, pois ela pode ser muito demorada, se o número de parâmetros n_θ for alto.

Com a matriz de correlação $\Phi(i)$ sendo definida positiva e, portanto, não singular, pode-se aplicar o Lema da Inversão de Matriz (A.2) para a Equação Recursiva (4.6). Mas, primeiro faz-se as seguintes identificações:

$$A = \Phi(i)$$
$$B^{-1} = \lambda\Phi(i-1)$$
$$C = \bar{x}(i)$$
$$D = 1$$

Então, substituindo essas definições no Lema da Inversão de Matriz, Equação (A.11), obtêm-se a seguinte equação recursiva para a inversa da matriz de correlação

$$\Phi^{-1}(i) = \lambda^{-1}\Phi^{-1}(i-1) - \frac{\lambda^{-2}\Phi^{-1}(i-1)\bar{x}(i)\bar{x}^H(i)\Phi^{-1}(i-1)}{1+\lambda^{-1}\bar{x}^H(i)\Phi^{-1}(i-1)\bar{x}(i)} \tag{4.8}$$

Por questões de conveniência, defina a seguinte relação

$$\Gamma(i) = \Phi^{-1}(i), \tag{4.9}$$

e

$$L(i) = \frac{\lambda^{-1}\Gamma(i-1)\bar{x}(i)}{1+\lambda^{-1}\bar{x}^H(i)\Gamma(i-1)\bar{x}(i)}. \tag{4.10}$$

Usando essas definições, pode-se reescrever a Equação (4.8) como segue:

$$\Gamma(i) = \lambda^{-1}\Gamma(i-1) - \lambda^{-1}L(i)\bar{x}^H\Gamma(i-1), \tag{4.11}$$

A matriz $\Gamma(i)$, $n_\theta \times n_\theta$, é denominada **matriz correlação inversa/matriz de covariância**. O vetor $L(i)$, $n_\theta \times 1$, é chamado vetor de ganho.

Rearranjando-se a Equação (4.10), tem-se

$$\begin{aligned} L(i) &= \lambda^{-1}\Gamma(i-1)\bar{x}(i) - \lambda^{-1}L(i)\bar{x}^H(i)\Gamma(i-1)\bar{x}(i) \\ &= [\lambda^{-1}\Gamma(i-1) - \lambda^{-1}L(i)\bar{x}^H(i)\Gamma(i-1)]\bar{x}(i) \end{aligned}, \tag{4.12}$$

Pode-se observar da Equação (4.11) que a expressão dentro dos colchetes no lado direito da Equação (4.12) é igual a $\Gamma(i)$. Assim, a Equação (4.12) pode ser simplificada para:

$$L(i) = \Gamma(i)\bar{x}(i). \tag{4.13}$$

Este resultado, juntamente com $\Gamma(i) = \Phi^{-1}(i)$, pode ser usado como a definição para o vetor de ganho:

$$L(i) = \Phi^{-1}(i)\bar{x}(i), \tag{4.14}$$

Em outras palavras, o vetor de ganho $L(i)$ é definido como vetor de entrada $\bar{x}(i)$ transformado pela inversa da matriz de correlação $\Phi(i)$.

Em seguida, se desenvolverá uma equação recursiva para atualizar a estimativa dos mínimos quadrados $\hat{\theta}(i)$ para o vetor de parâmetro. Para fazer isso, usa-se as Equações (4.2), (4.7) e (4.9) para expressar a estimativa dos mínimos quadrados $\hat{\theta}(i)$ para o vetor de parâmetros na iteração i da seguinte forma:

$$\begin{aligned} \hat{\theta}(i) &= \Phi^{-1}(i)z(i) \\ &= \Gamma(i)z(i) \\ &= \lambda\Gamma(i)z(i-1) + \Gamma(i)\bar{x}(i)d^*(i), \end{aligned} \tag{4.15}$$

Substituindo a Equação (4.11) para $\Gamma(i)$ somente no primeiro termo no lado direito da Equação (4.15), obtêm-se:

$$\begin{aligned} \hat{\theta}(i) &= \Gamma(i-1)z(i-1) - L(i)\bar{x}^H(i)\Gamma(i-1)z(i-1) \\ &\quad +\Gamma(i)\bar{x}(i)d^*(i) \\ &= \Phi^{-1}(i-1)z(i-1) - L(i)\bar{x}^H(i)\Phi^{-1}(i-1)z(i-1) \\ &\quad +\Gamma(i)\bar{x}(i)d^*(i) \\ &= \hat{\theta}(i-1) - L(i)\bar{x}^H(i)\hat{\theta}(i-1) + \Gamma(i)\bar{x}(i)d^*(i), \end{aligned} \tag{4.16}$$

Finalmente, usando o fato de que $\Gamma(i)\bar{x}(i)$ é igual ao vetor de ganho $L(i)$ como na Equação (4.13), obtêm-se a equação recursiva desejada para a atualização do vetor de parâmetro θ :

$$\begin{aligned} \hat{\theta}(i) &= \hat{\theta}(i-1) + L(i)[d^*(i) - \bar{x}^H(i)\hat{\theta}(i-1)] \\ &= \hat{\theta}(i-1) + L(i)\xi^*(i), \end{aligned} \tag{4.17}$$

em que $\xi(i)$ é o erro de estimativa a priori definido por

$$\begin{aligned}\xi(i) &= d(i) - \bar{x}^T(i)\hat{\theta}^*(i-1) \\ &= d(i) - \hat{\theta}^H(i-1)\bar{x}(i),\end{aligned} \tag{4.18}$$

O produto interno $\hat{\theta}^H(i-1)\bar{x}(i)$ representa uma estimativa da resposta desejada $d(i)$, com base na estimativa anterior de mínimos quadrados do vetor de parâmetros $\hat{\theta}(\cdot)$ que foi calculado no passo de tempo $i-1$.

O erro de estimativa a priori $\xi(i)$ é, em geral, diferente do erro de estimativa a posteriori

$$e(i) = d(i) - \hat{\theta}^H(i)\bar{x}(i) \tag{4.19}$$

Um resumo do Algoritmo RLS$_\lambda$-HDP-DLQR é apresentado da seguinte forma:

Algorítimo 1 - Algorítimo HDP - Recursive least-square and Recurrence Step.

RLS$_\lambda$-HDP-DLQR

---

1 ▷ Configuração - Condições Iniciais
2 ▷ Ponderações e Sistema Dinâmico - $[Q, R, A_d, B_d, N] \leftarrow [\quad]$
3 ▷ Fator de Desconto - $0 < \gamma \leq 1$.
4 ▷ Parâmetro RLS: $\hat{\theta_0}$, Γ_0
5 ▷ Fator de Esquecimento - $0 < \lambda \leq 1$.
6 ▷ Estado Inicial - x_0.
7 ▷ Ponderações e Sistema Dinâmico - $[N, n_{revit}] \leftarrow [\quad]$
8 ▷ Valores inicias de P e K - $[P_0, K_0] \leftarrow [\quad]$

---

9 ▷ Processo Iterativo
10 **for** $i \leftarrow 0 : N$
11 **do**
12 ▷ Politica Ótima
13 $u_i \leftarrow K_i x_i$
14 ▷ Estado
15 $x_{i+1} \leftarrow A_d x_i + B_d u_i$
16 ▷ Definição da Base - Produto de Kronecker
17 $\overline{x}_i \leftarrow [x_{1i}^2; \ldots x_{1i}x_{ni}; x_{2i}^2; \ldots; x_{n-1i}x_{ni}; x_{ni}^2]$
18 ▷ Montagem do Alvo
19 $d(x, r, f, P) \leftarrow x_i^T Q x_i + u_i^T R u_i + x_{i+1}^T P x_{i+1}$
20 ▷ RLS - Recursive least-square
21 $L_{i+1} = \frac{\Gamma_i \overline{x}_i}{\lambda + \overline{x}_i^T \Gamma(i) \overline{x}_i}$
22 $\hat{\theta}_{i+1} = \hat{\theta}_i + L_{i+1}(d(\cdot) - \overline{x}_i^T \hat{\theta}_i)$
23 $\Gamma_{i+1} = \lambda^{-1}\left(\Gamma_i - L_{i+1}\overline{x}_i^T \Gamma_i\right)$
24 ▷ recuperação de matriz P a partir dos vetores θ
25 $m = \frac{n(n+1)}{2}$
26 $P^{\hat{\theta}_{i+1}} \leftarrow \begin{bmatrix} \hat{\theta}_1 & \hat{\theta}_2/2 & \cdots & \hat{\theta}_n/2 \\ \hat{\theta}_2/2 & \hat{\theta}_{1+n} & \cdots & \hat{\theta}_{2n-1}/2 \\ \vdots & \vdots & \ddots & \hat{\theta}_{2n}/2 \\ \hat{\theta}_n/2 & \hat{\theta}_{n+1}/2 & \hat{\theta}_{2n}/2 & \hat{\theta}_m \end{bmatrix}$
27 ▷ Atualização da Política (Ganho Ótimo de Realimentação K) K
28 $K_{i+1} \leftarrow -\gamma(R + \gamma B_d^T P^{\hat{\theta}_{i+1}} B_d)^{-1} B_d^T P^{\hat{\theta}_{i+1}} A_d$
29 **if** $i \% n_{revit} = 0$
30 **then**
31 $x_{i+1} \leftarrow x_{revit}$

---

32 Fim do Processo Iterativo

4.3 Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR

Nesta seção se mostrará a adequação do Algoritmo RLS$_\lambda$-HDP-DLQR em RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR, mais precisamente, o algorítimo RLS$_\lambda$-HDP-DLQR baseado em decomposição $\mathcal{QR}$.

Do algoritmo RLS$_\lambda$-HDP-DLQR, tem-se que a equação de atualização recur-

siva do valor da matriz de correlação é dada por:

$$\Phi(i) = \lambda\Phi(i-1) + \bar{x}(i)\bar{x}^H(i), \tag{4.20}$$

onde $\Phi(i-1)$ é o valor "anterior"da matriz de correlação, e o produto matricial $\bar{x}(i)\bar{x}^H(i)$ é o termo de correção na operação de atualização.

Antes de proceder com a formulação do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR, é conveniente introduzir algumas notações. De acordo com as equações normais do estimador dos mínimos quadrados, o vetor de parâmetros $\hat{\theta}(i)$ é definido por:

$$\Phi(i)\hat{\theta}(i) = z(i), \tag{4.21}$$

onde $z(i)$ é o vetor de correlação cruzada entre a resposta desejada $d(i)$ e o vetor de entrada $\bar{x}(i)$.

Seja a matriz $\Phi(i)$ expressa na forma:

$$\Phi(i) = \Phi^{1/2}(i).\Phi^{H/2}(i), \tag{4.22}$$

sendo $\Phi^{1/2}$ uma matriz triangular inferior definida como a raiz quadrada de $\Phi(i)$, e $\Phi^{H/2}$ é a transposta Hermitiana de $\Phi^{1/2}$ (Decomposição de Cholesky).

Em seguida, pré-multiplicando os dois lados da Equação (4.21) pela matriz $\Phi^{-1/2}(i)$, produz-se um novo vetor de variáveis definido por:

$$\omega(i) = \Phi^{H/2}(i)\hat{\theta}(i) = \Phi^{-1/2}(i)z(i), \tag{4.23}$$

Assim, pode-se chegar ao vetor de correlação cruzada entre a entrada $\bar{x}(i)$ e a saída desejada $d(i)$:

$$z(i) = \lambda z(i-1) + \bar{x}(i)d^*(i), \tag{4.24}$$

ou equivalentemente,

$$\Phi(i)\hat{\theta}(i) = \lambda\Phi(i-1)\hat{\theta}(i-1) + \bar{x}(i)d^*(i) \tag{4.25}$$

Devido a questões que serão explicadas mais adiante, é mais conveniente expressar as Equações (4.20) e (4.25) na forma transposta Hermitiana, em tal caso pode-se expressar cada um dos termos que aparece no segundo membro de cada equação em suas formas fatoradas individuais da seguinte maneira:

$$\lambda\Phi^H(i-1) = [\lambda^{1/2}\Phi^{1/2}(i-1)].[\lambda^{1/2}\Phi^{H/2}(i-1)] \tag{4.26}$$

$$\begin{aligned}\lambda\hat{\theta}^H(i-1)\Phi^H(i-1) &= [\lambda^{1/2}\hat{\theta}^H(i-1)\Phi^{1/2}(i-1)][\lambda^{1/2}\Phi^{H/2}(i-1)] \\ &= [\lambda^{1/2}\omega^H(i-1)][\lambda^{1/2}\Phi^{H/2}(i-1)]\end{aligned} \tag{4.27}$$

Pode-se identificar agora quatro fatores distintos como os elementos de bloco da pré-matriz, que são arranjados da seguinte forma:

- $\lambda^{1/2}\Phi^{1/2}(i-1)$ e $\bar{x}(i)$ que são de dimensão $n_\theta \times n_\theta$ e $n_\theta \times 1$ respectivamente.
- $[\lambda^{1/2}\hat{\omega}^H(i-1)]$ e $d(i)$, que são de dimensão $1 \times n_\theta$ e 1×1 respectivamente

O primeiro par de fatores é naturalmente compatível em virtude de ser composto por uma matriz e um vetor. A compatibilidade do último par de fatores como vetores linha é a razão para trabalhar com as formas transpostas hermitianas das Equações (4.20) e (4.25) que podem, assim, construir a seguinte pre-matriz

$$\begin{bmatrix} \lambda^{1/2}\Phi^{1/2}(i-1) & \bar{x}(i) \\ \lambda^{1/2}\theta^H(i-1) & d(i) \\ 0^T & 1 \end{bmatrix}$$

A última linha, constituída por um bloco de zeros seguido por um termo unitário, foi adicionado a fim de abrir espaço para a geração de outras variáveis do RLS na pós-matriz.

Suponha-se que em seguida se escolhe uma rotação unitária $\Theta(i)$ que transforma esta pré-matriz de modo a produzir um bloco nulo na segunda entrada da linha do bloco superior da pós matriz, como mostrado pela forma:

$$B = A\Theta, \tag{4.28}$$

sendo

$$A = \begin{bmatrix} \lambda^{1/2}\Phi^{1/2}(i-1) & \bar{x}(i) \\ \lambda^{1/2}\omega^H(i-1) & d(i) \\ 0^T & 1 \end{bmatrix}, \quad B = \begin{bmatrix} B_{11}^H(i) & 0 \\ b_{21}^H(i) & b_{22}^*(i) \\ b_{31}^H(i) & b_{32}^*(i) \end{bmatrix}. \tag{4.29}$$

Para avaliar os elementos dos blocos desconhecidos $B_{11}^H(i)$, $b_{21}^H(i)$, $b_{22}^*(i)$, $b_{31}^H(i)$ e $b_{32}^*(i)$ da pós-matriz, procede-se elevando ambos os membros da Equação (4.30) ao quadrado. Em seguida, reconhecendo que $\Theta(i)$ é uma matriz unitária, e, portanto, $\Theta(i)\Theta^H(i)$ é igual à matriz identidade para todo i, pode-se escrever:

$$AA^H = BB^H, \tag{4.30}$$

que expandindo, obtêm-se:

$$\begin{bmatrix} \lambda^{1/2}\Phi^{1/2}(i-1) & \bar{x}(i) \\ \lambda^{1/2}\omega^H(i-1) & d(i) \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} \lambda^{1/2}\Phi^{H/2}(i-1) & \lambda^{1/2}\omega(i-1) & 0 \\ \bar{x}^H(i) & d^*(i) & 1 \end{bmatrix} = \begin{bmatrix} B_{11}^H(i) & 0 \\ b_{21}^H(i) & b_{22}^*(i) \\ b_{31}^H(i) & b_{32}^*(i) \end{bmatrix} \begin{bmatrix} B_{11}(i) & b_{21}(i) & b_{31}(i) \\ 0^T & b_{22}(i) & b_{32}(i) \end{bmatrix} \tag{4.31}$$

Assim, comparando-se os respectivos termos de ambos os lados da igualdade (4.31), obtêm-se as seguintes identidades:

$$B_{11}^H(i)B_{11}(i) = \lambda\Phi(i-1) + \bar{x}(i)\bar{x}^H(i) = \Phi(i) \tag{4.32}$$

$$b_{21}^H(i)B_{11}(i) = \lambda\omega^H(i-1)\Phi^{H/2}(i-1) + d(i)\bar{x}^H(i) \tag{4.33}$$

$$b_{31}^H(i)B_{11}(i) = \bar{x}^H(i) \tag{4.34}$$

$$B_{11}^H(i)b_{21}(i) = \lambda\Phi^{1/2}(i-1)\omega(i-1) + \bar{x}(i)d^*(i) \tag{4.35}$$

$$b_{21}^H(i)b_{21}(i) + b_{22}^*(i)b_{22}(i) = \lambda\omega^H(i-1)\omega(i-1) + d(i)d^*(i) \tag{4.36}$$

$$b_{31}^H(i)b_{21}(i) + b_{32}^*(i)b_{22}(i) = d^*(i) \tag{4.37}$$

$$B_{11}^H(i)b_{31}(i) = \bar{x}(i) \tag{4.38}$$

$$b_{21}^H(i)b_{31}(i) + b_{22}^*(i)b_{32}(i) = d(i) \tag{4.39}$$

$$b_{31}^H(i)b_{31}(i) + b_{32}^*(i)b_{32}(i) = 1 \tag{4.40}$$

Desenvolvendo a Equação (4.32), encontra-se o valor de B_{11}

$$B_{11}^H(i)B_{11}(i) = \lambda\Phi(i-1) + \bar{x}(i)\bar{x}^H(i) = \Phi(i)$$

$$B_{11}^H(i)B_{11}(i) = \Phi(i)$$

$$B_{11}^H(i) = \Phi^{1/2}(i) \tag{4.41}$$

Desenvolvendo a Equação (4.33), encontra-se o valor de b_{21}

$$b_{21}^H(i)B_{11}(i) = \lambda\omega^H(i-1)\Phi^{H/2}(i-1) + d(i)\bar{x}^H(i)$$

$$b_{21}^H(i)\Phi^{H/2}(i) = \lambda\omega^H(i-1)\Phi^{H/2}(i-1) + d(i)\bar{x}^H(i)$$

$$b_{21}^H(i)\Phi^{H/2}(i) = \lambda\hat{\theta}^H(i-1)\Phi^{1/2}(i-1)\Phi^{H/2}(i-1) + d(i)\bar{x}^H(i)$$

$$b_{21}^H(i)\Phi^{H/2}(i) = \lambda\hat{\theta}^H(i-1)\Phi(i-1) + d(i)\bar{x}^H(i)$$

$$b_{21}^H(i)\Phi^{H/2}(i) = \hat{\theta}^H(i)\Phi^H(i)$$

$$b_{21}^H(i) = \hat{\theta}^H(i)\Phi^{1/2}(i)$$

$$b_{21}^H(i) = \omega^H(i) \tag{4.42}$$

Desenvolvendo a Equação (4.34), encontra-se o valor de b_{31}

$$b_{31}^H(i)B_{11}(i) = \bar{x}^H(i)$$

$$b_{31}^{H}(i)\Phi^{H/2}(i) = \bar{x}^{H}(i)$$

$$b_{31}^{H}(i) = \bar{x}^{H}(i)\Phi^{-H/2}(i) \tag{4.43}$$

Desenvolvendo a Equação (4.40), encontra-se o valor de b_{32}

$$b_{31}^{H}(i)b_{31}(i) + b_{32}^{*}(i)b_{32}(i) = 1$$

$$b_{32}^{*}(i)b_{32}(i) = 1 - b_{31}^{H}(i)b_{31}(i)$$

$$b_{32}^{*}(i)b_{32}(i) = 1 - (\bar{x}^{H}(i)\Phi^{-H/2}(i)(\Phi^{-1/2}(i)\bar{x}(i))$$

$$b_{32}^{*}(i)b_{32}(i) = 1 - \bar{x}^{H}(i)\Phi^{-1}(i)\bar{x}(i)$$

$$b_{32}^{*}(i)b_{32}(i) = 1 - \bar{x}^{H}(i)L(i)$$

$$b_{32}^{*}(i) = (1 - L^{H}(i))^{1/2}\bar{x}(i)$$

Definindo $\rho(i) = 1 - L^{H}(i)\bar{x}(i)$

$$b_{32}^{*}(i) = \rho^{1/2}(i) \tag{4.44}$$

Desenvolvendo a Equação (4.39), encontra-se o valor de b_{22}

$$b_{21}^{H}(i)b_{31}(i) + b_{22}^{*}(i)b_{32}(i) = d(i)$$

$$\omega^{H}(i)\Phi^{-1/2}(i)\bar{x}(i) + b_{22}^{*}(i)\left(\rho^{1/2}\right)^{*}(i) = d(i)$$

$$b_{22}^{*}(i)\left(\rho^{1/2}\right)^{*}(i) = d(i) - \omega^{H}(i)\bar{x}(i)\Phi^{1/2}(i)$$

$$b_{22}^{*}(i) = (d(i) - \omega^{H}(i)\bar{x}(i)\Phi^{1/2}(i))\left(\rho^{-1/2}\right)^{*}(i)$$

$$b_{22}^{*}(i) = (d(i) - \bar{x}(i)\hat{\theta}^{H}(i))\left(\rho^{-1/2}\right)^{*}(i)$$

$$b_{22}^{*}(i) = (d(i) - \bar{x}(i)(\hat{\theta}^{H}(i-1) + L^{H}(i)\xi(i)))\left(\rho^{-1/2}\right)^{*}(i)$$

$$b_{22}^{*}(i) = (d(i) - \bar{x}(i)\hat{\theta}^{H}(i-1) - \bar{x}(i)L^{H}(i)\xi(i))\left(\rho^{-1/2}\right)^{*}(i)$$

$$b_{22}^{*}(i) = (\xi(i) - \bar{x}(i)L^{H}(i)\xi(i))\left(\rho^{-1/2}\right)^{*}(i)$$

$$b_{22}^{*}(i) = \xi(i)(1 - \bar{x}(i)L^{H}(i))\left(\rho^{-1/2}\right)^{*}(i)$$

$$b_{22}^{*}(i) = \xi(i)\rho(i)\left(\rho^{-1/2}\right)^{*}(i)$$

$$b_{22}^{*}(i) = \xi(i)\rho^{1/2}(i) \tag{4.45}$$

Assim, reorganizando a matriz da Equação (4.31) com os valores de $B_{11}^{H}(i)$, $b_{21}^{H}(i)$, $b_{22}^{*}(i)$, $b_{31}^{H}(i)$ e $b_{32}^{*}(i)$, retirados das Equações (4.41), (4.42), (4.45), (4.43) e (4.44), respectivamente, tem-se:

$$\begin{bmatrix} \lambda^{1/2}\Phi^{1/2}(i-1) & \bar{x}(i) \\ \lambda^{1/2}\omega^{H}(i-1) & d(i) \\ 0^{T} & 1 \end{bmatrix} \Theta(i) = \begin{bmatrix} \Phi^{1/2}(i) & 0 \\ \omega^{H}(i) & \xi(i)\rho^{1/2}(i) \\ \bar{x}^{H}(i)\Phi^{-H/2}(i) & \rho^{1/2}(i) \end{bmatrix}, \tag{4.46}$$

e a atualização do vetor de parâmetro $\hat{\theta}(i)$, tendo em vista a Equação (4.23), é dado por:

$$\hat{\theta}^{H}(i) = \omega^{H}(i)\Phi^{-1/2}(i) \tag{4.47}$$

Algorítimo 2 - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR - Decomposição $\mathcal{QR}$ por Rotações de Givens.

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (ROTAÇÃO DE GIVENS)

▷ Configuração - Condições Iniciais
▷ Ponderações e Sistema Dinâmico - $[Q, R, A_d, B_d, N] \leftarrow [\quad]$
▷ Fator de Desconto - $0 < \gamma \leq 1$.
▷ Parâmetro RLS - $\mathcal{QR}$: $\Phi_0^{1/2}$, ω_0
▷ Fator de Esquecimento - $0 < \lambda \leq 1$.
▷ Seleção - Estado Inicial - x_0.
▷ Ponderações e Sistema Dinâmico - $[N, n_{revit}] \leftarrow [\quad]$
▷ Valores inicias de P e K - $[P_0, K_0] \leftarrow [\quad]$

▷ Processo Iterativo
for $i \leftarrow 0 : N$
do
▷ Politica Ótima
$u_i \leftarrow K_i x_i$
▷ Estados
$x_{i+1} \leftarrow A_d x_i + B_d u_i$
▷ Definição da Base - Produto de Kronecker
$\overline{x}_i \leftarrow [x_{1i}^2; x_{1i}x_{2i}; \ldots; x_{4i}^2]$
▷ Montagem do Alvo
$d(x, r, f, P) \leftarrow x_i^T Q x_i + u_i^T R u_i + x_{i+1}^T P_i x_{i+1}$
▷ RLS baseado em Decomposição $\mathcal{QR}$ (Rotação de Givens)

$$A_{\mathcal{QR}} \leftarrow \begin{bmatrix} \lambda^{1/2}\Phi_i^{1/2} & \overline{x}_i \\ \lambda^{1/2}\omega_i^H & d(\cdot) \\ 0^T & 1 \end{bmatrix}$$

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

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

$$Q_{\mathcal{QR}} \leftarrow eye(i)$$

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

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

$$\mathcal{R} \rightarrow \begin{bmatrix} \Phi_{i+1}^{1/2} & 0 \\ \omega_{i+1}^H & \xi_i \rho_{i+1}^{1/2} \\ \overline{x}_i^H \Phi_{i+1}^{-H/2} & \rho_{i+1}^{1/2} \end{bmatrix}$$

▷ Atualização do vetor $\hat{\theta}$ por meio da "substituição para trás"
$\hat{\theta}_{i+1}^H = \omega_{i+1}^H \Phi_{i+1}^{-1/2}$
▷ Recuperação de matriz P a partir dos vetores θ
$P^{\hat{\theta}_{i+1}} \leftarrow [\hat{\theta}_1, \quad \hat{\theta}_{2/2}, ..., \quad \hat{\theta}_{9/2}; ..., \quad \hat{\theta}_{10}]$
▷ Atualização da Política (Ganho Ótimo de Realimentação K)
$K_{i+1} \leftarrow -\gamma(R + \gamma B_d^T P^{\hat{\theta}_{i+1}} B_d)^{-1} B_d^T P^{\hat{\theta}_{i+1}} A_d$
if $i \% n_{revit} = 0$
then
$x_{i+1} \leftarrow x_{revit}$

Fim do Processo Iterativo

Algorítimo 3 - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR - Decomposição $\mathcal{QR}$ por Reflexão de Householder.

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (REFLEXÃO DE HOUSEHOLDER)

▷ Configuração - Condições Iniciais

▷ Ponderações e Sistema Dinâmico - $[Q, R, A_d, B_d, N] \leftarrow [\quad]$

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

▷ Parâmetro RLS - $\mathcal{QR}$: $\Phi_0^{1/2}$, ω_0

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

▷ Seleção - Estado Inicial - x_0.

▷ Ponderações e Sistema Dinâmico - $[N, n_{revit}] \leftarrow [\quad]$

▷ Valores inicias de P e K - $[P_0, K_0] \leftarrow [\quad]$

▷ Processo Iterativo

for $i \leftarrow 0 : N$

do

▷ Politica Ótima

$u_i \leftarrow K_i x_i$

▷ Estados

$x_{i+1} \leftarrow A_d x_i + B_d u_i$

▷ Definição da Base - Produto de Kronecker

$\overline{x}_i \leftarrow [x_{1i}^2; x_{1i}x_{2i}; \ldots; x_{4i}^2]$

▷ Montagem do Vetor Alvo

$d(x, r, f, P) \leftarrow x_i^T Q x_i + u_i^T R u_i + x_{i+1}^T P x_{i+1}$

▷ RLS baseado em Decomposição $\mathcal{QR}$ (Reflexão de Householder)

$$A_{\mathcal{QR}} \leftarrow \begin{bmatrix} \lambda^{1/2}\Phi^{1/2}(i-1) & \bar{x}(i) \\ \lambda^{1/2}p^H(i-1) & d(i) \\ 0^T & 1 \end{bmatrix}$$

$$s = \sqrt{a_k^2 + a_{k+1}^2 + \ldots + a_{n_{\theta+2}}^2}$$

$$w = \frac{1}{\sqrt{2s(s+|a_k|)}} \begin{pmatrix} 0 & \ldots & a_k + \text{sinal}(a_k)s & a_{k+1} & \ldots & a_{n_{\theta+2}} \end{pmatrix}^T$$

$$H = (I - 2ww^t)$$

$$\mathcal{R} = HA_{\mathcal{QR}}$$

$$\mathcal{R} \rightarrow \begin{bmatrix} \Phi_{i+1}^{1/2} & 0 \\ \omega_{i+1}^H & \xi_i \rho_{i+1}^{1/2} \\ \bar{x}_i^H \Phi_{i+1}^{-H/2} & \rho_{i+1}^{1/2} \end{bmatrix}$$

▷ Atualização do vetor $\hat{\theta}$ por meio da "substituição para trás"

$\hat{\theta}_i^H = \omega_i^H \Phi_i^{-1/2}$

▷ Recuperação de matriz P a partir dos vetores θ

$P^{\hat{\theta}_{i+1}} \leftarrow [\hat{\theta}_1, \quad \hat{\theta}_{2/2}, ..., \quad \hat{\theta}_{9/2}; ..., \quad \hat{\theta}_{10}]$

▷ Atualização da Política (Ganho Ótimo de Realimentação K)

$K_{i+1} \leftarrow -\gamma(R + \gamma B_d^T P^{\hat{\theta}_{i+1}} B_d)^{-1} B_d^T P^{\hat{\theta}_{i+1}} A_d$

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

then

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

Fim do Processo Iterativo

4.4 Considerações Finais

A formulação do problema foi apresentada como a fusão de três metodologias para o projeto de controle ótimo online 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 desta técnica está relacionado com o mal condicionamento da matriz de covariância da abordagem RLS para estimar a solução da equação HJB-Riccati.

Para diminuir esse problema propôs-se a união da decomposição $\mathcal{QR}$ no algoritmo padrão visando melhorar o comportamento do número de condição e parâmetro de positividade da matriz de covariância resultando assim no algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (variação do algoritmo RLS$_\lambda$-HDP-DLQR).

Capítulo 5

Complexidade Computacional

5.1 Introdução

A teoria da complexidade computacional é um ramo da teoria da computação em ciência da computação teórica e matemática que se concentra em classificar problemas computacionais de acordo com sua dificuldade inerente, e relacionar essas classes entre si (Downey and Fellows 1999). Neste contexto, um problema computacional é entendido como uma tarefa que é, em princípio, possível de ser resolvida por um computador (o que basicamente significa que o problema pode ser descrito por um conjunto de instruções matemáticas). Informalmente, um problema computacional consiste de instâncias do problema e soluções para essas instâncias do problema.

Um problema é dificílimo no sentido de que nenhum algoritmo pode resolvê-lo

Por exemplo, o teste de primalidade é o problema de determinar se um dado número é primo ou não. As instâncias deste problema são números naturais, e a solução para uma instância é sim ou não, dependendo se o número é primo ou não. Um problema é considerado como inerentemente difícil se a sua solução requer recursos significativos, qualquer que seja o algoritmo usado. A teoria formaliza esta intuição através da introdução de modelos matemáticos de computação para estudar estes problemas e quantificar os recursos necessários para resolvêlos, tais como tempo e armazenamento. Outras medidas de complexidade também são utilizadas, tais como a quantidade de comunicação (usada em complexidade de comunicação), o número de portas em um circuito (usado na complexidade de

circuito) e o número de processadores (usados em computação paralela) (Du and Ko 2000). Um dos papéis da teoria da complexidade computacional é determinar os limites práticos do que os computadores podem e não podem fazer.

Campos intimamente relacionados com a ciência da computação teórica são a análise de algoritmos e a teoria da computabilidade. Uma distinção chave entre a análise de algoritmos e teoria da complexidade computacional é que a primeira é dedicada a analisar a quantidade de recursos necessários para um determinado algoritmo resolver um problema, enquanto o segundo faz uma pergunta mais geral sobre todos os possíveis algoritmos que podem ser usados para resolver o mesmo problema (Goldreich 2008). Mais precisamente, ele tenta classificar os problemas que podem ou não podem ser resolvidos com os recursos devidamente restritos. Por sua vez, impondo restrições sobre os recursos disponíveis é o que distingue a complexidade computacional da teoria da computabilidade: a segunda pergunta que tipos de problemas podem, em princípio, ser resolvidos através de algoritmos.

5.2 Parâmetros de Avaliação de Algoritmos

Um algoritmo é uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais pode ser executada mecanicamente em um período de tempo finito e com uma quantidade de esforço finita (Goldreich 2008).

O conceito de algoritmo é frequentemente ilustrado pelo exemplo de uma receita culinária, embora muitos algoritmos sejam mais complexos. Eles podem repetir passos (fazer iterações) ou necessitar de decisões (tais como comparações ou lógica) até que a tarefa seja completada. Um algoritmo corretamente executado não irá resolver um problema se estiver implementado incorretamente ou se não for apropriado ao problema.

Um algoritmo não representa, necessariamente, um programa de computador , e sim os passos necessários para realizar uma tarefa. Sua implementação pode ser feita por um computador, por outro tipo de autômato ou mesmo por um ser humano. Diferentes algoritmos podem realizar a mesma tarefa usando um conjunto diferenciado de instruções em mais ou menos tempo, espaço ou esforço do que outros. Tal diferença pode ser reflexo da complexidade computacional aplicada, que depende de estruturas de dados adequadas ao algoritmo (Sipser

2006). Por exemplo, um algoritmo para se vestir pode especificar que você vista primeiro as meias e os sapatos antes de vestir a calça enquanto outro algoritmo especifica que você deve primeiro vestir a calça e depois as meias e os sapatos. Fica claro que o primeiro algoritmo é mais difícil de executar que o segundo apesar de ambos levarem ao mesmo resultado.

O conceito de um algoritmo foi formalizado em 1936 pela Máquina de Turing de Alan Turing e pelo cálculo *lambda* de Alonzo Church (Church 1996), que formaram as primeiras fundações da Ciência da computação.

Logo, pode-se avaliar um algoritmo em diversos parâmetros tais como:

1. Velocidade de Execução, que talvez seja o ponto mais importante de algoritmos para processamento em tempo real.

2. Utilização de Memória, o avanço da informática elevou a ampla disponibilidade de memória diminuindo a preocupação com esse parâmetro.

3. Estabilidade, garantir que o funcionamento do algoritmo seja o mesmo ao passar do tempo não sofrendo com mau funcionamento pelo tempo de execução, entre outros.

Os algoritmos desenvolvidos no presente trabalho são avaliados tendo como referencia três importantes parâmetros: Custo Computacional, Estabilidade Numérica e Tempo de Convergência, os quais serão discutidos a seguir.

5.2.1 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.

Assim, a analise do custo computacional deve ser calculada em uma unidade computacional chamada *flops* que é um acrônimo na computação que significa *Floating-point Operations Per Second*, que, em português, quer dizer operações de ponto flutuante por segundo. Isto é usado para determinar o desempenho de um computador ou de um algoritmo, especificamente no campo de cálculos científicos, que fazem grande uso de cálculos com ponto flutuante; similar a instruções por segundo (Arora and Barak 2006).

Já que dispositivos de computação têm enorme capacidade de processamento, convém utilizar unidades maiores que *flops*, seus múltiplos. Os múltiplos mais utilizados são: *megaflops* (Mflop/s), *gigaflops* (Gflop/s), *teraflops* (Tflop/s), *petaflops* (Pflop/s) e *exaflops* (Eflop/s).

A evolução da informática esta elevando cada vez a capacidade de *flops* de um computador porém ainda existe cálculos que necessitam de uma quantidade muito grande de *flops* para ser executados, o maior exemplo são os cálculos necessários para previsão de tempo que sempre são realizados por supercomputadores que tem capacidade de processamento muito maior que computadores pessoais PC's. Por isso, ainda se deve ter a preocupação em redução do custo computacional dos algoritmos.

Nessa seção serão descritos os custos computacionais dos algoritmos propostos neste estudo. Antes de prosseguir com tais desenvolvimentos, serão necessários introduzir algumas notações.

Sejam α_1 e α_2 dois escalares quaisquer em $\Re$. As notações $op_{\alpha}^{(+)}$, $op_{\alpha}^{(.)}$ 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 5.1 e 5.2 apresentam algumas operações básicas de vetores e matrizes que aparecem nas etapas de avaliação e melhoria de política dos algoritmos RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR. Estas tabelas também fornecem o número de *flops* (operações de ponto flutuante por segundo) 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 $M_1 \in \Re^{p\times m}$ 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 $op^{(-1)}_{M_{DP}}$.

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

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

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

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

RLS$_\lambda$-HDP-DLQR

ETAPA AVALIAÇÃO DE POLÍTICA

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

1- Construção do vetor de funções de base $\bar{x}_k$ via produto de Kronecker dos estados. Considerando que a formação de cada vetor $\bar{x}_k$ envolve $n_\theta[op^{(.)}_\alpha]$, o que equivale à n_θ *flops*, tem-se

$$op^{(.)}_\alpha = n_\theta = n_\theta flops$$

2- Cálculo do ganho $L(i)$ da Equação (4.10):

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

3- Atualização do vetor de parâmetros $\hat{\theta}(i)$ de Equação (4.17):

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

4- Atualização da matriz de covariância $\Gamma(i)$ de Equação (4.08):

$$op_v^{out} + op_M^{(\cdot)} + op_M^{(+)} + op_{M,\alpha}^{(\cdot)} + op_{\alpha}^{(\div)} \Rightarrow n_\theta^2 + (2n_\theta^3 - n_\theta^2) + n_\theta^2 + n_\theta^2 + 1 = 2n_\theta^3 + 2n_\theta^2 + 1 \quad flops$$

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

$$(n^2 - n) op_{\alpha}^{(\div)} \Rightarrow n^2 - n \quad flops$$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

ETAPA AVALIAÇÃO DE POLÍTICA

Na descrição do custo computacional em uma iteração i de avaliação de política realizada pelo RLS $\mathcal{QR}$ com rotação de Givens, tem-se:

1- Segue-se o mesmo procedimento do algoritmo RLS$_\mu$-HDP-DLQR o qual requer n_θ *flops*

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

$$2(2op_{\alpha}^{(\cdot)} + op_{\alpha}^{(+)} + op_{\alpha}^{(sqrt)} + op_{\alpha}^{(\div)}) \Rightarrow 2(2 + 1 + 1 + 1) = 10 \quad flops$$

3- Cálculo da Matriz $\mathcal{R}$:

$$op_M^{(\cdot)} \Rightarrow 2(2)^3 - (2)^2 \quad flops$$

4- Atualização do vetor de parâmetros $\hat{\theta}(i)$ (Back Substitution)

$$op_{backsub} \Rightarrow n_\theta^2 = \; n_\theta^2 \, flops$$

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

$$(n^2 - n)op_{\alpha}^{(\div)} \Rightarrow n^2 - n \quad flops$$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

ETAPA AVALIAÇÃO DE POLÍTICA

Na descrição do custo computacional em uma iteração i de avaliação de política realizada pelo RLS $\mathcal{QR}$, tem-se:

1- Segue-se o mesmo procedimento do algoritmo RLS$_\mu$-HDP-DLQR o qual requer n_θ $flops$

2- Cálculo de s (Householder):

$$op_{v,v}^{(\cdot)} + op_{\alpha}^{(sqrt)} \Rightarrow n_\theta + n_\theta = 2n_\theta \ \ flops$$

3- Montagem da Matriz z (Householder):

$$op_{\alpha}^{(\cdot)} + op_{\alpha}^{(+)} \Rightarrow 1 + 1 = \ 2flops$$

4- Calculo da Matriz w (Householder):

$$op_{\alpha}^{(+)} + 2op_{\alpha}^{(\cdot)} + op_{\alpha}^{(\div)} + op_{M,\alpha}^{(\cdot)} \Rightarrow 1 + 2 + 1 + n_\theta^2 = \ n_\theta^2 + 4 \ flops$$

5- Calculo da Matriz H (Householder):

$$op_{M}^{(-)} + op_{M,\alpha}^{(\cdot)} + op_{v,v}^{(\cdot)} \Rightarrow n_\theta^2 + n_\theta^2 + 2n_\theta = \ 2n_\theta^2 + 2n_\theta \ flops$$

6- Calculo da matriz R

$$op_{M}^{(\cdot)} \Rightarrow 2n_\theta^3 - n_\theta^2 \, flops$$

7- Atualização do vetor de parâmetros $\hat{\theta}(i)$ (Back Substitution):

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

8- Recuperação da matriz P a partir do vetor $\hat{\theta}$ gerado pelo $\mathcal{QR}$-RLS é dado por:

$$(n^2 - n)op_{\alpha}^{(\div)} \Rightarrow n^2 - n \quad flops$$

Agora pode-se apresentar um comparativo entre o Custo Computacional dos algoritmos RLS$_\lambda$-HDP-DLQR, RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (Givens) e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (Householder), essa avaliação será feita apenas na etapa de avaliação de política, por ser a única etapa a possuir diferença entre os algoritmos.

Uma forma simples de comparar os algoritmos e apresentar quantos *flops* cada algoritmo gasta para realizar as operações de $op^{(+/-)}$, $op^{(\cdot)}$ e $op^{(\div)}$. Um resumo da quantidade de cada tipo de operação necessária para execução dos algoritmos e apresentado abaixo.

$$\text{RLS}\mu \text{ - HDP - DLQR}$$
$$op^{(+/-)} \rightarrow \frac{n^4}{4} + \frac{n^3}{2} + \frac{3n^2}{4} + \frac{n}{2} + 2$$
$$op^{(\cdot)} \rightarrow \frac{n^6}{4} + \frac{3n^5}{4} + 2n^4 + \frac{11n^3}{4} + \frac{15n^2}{4} + \frac{5n}{2} - 2$$
$$op^{(\div)} \rightarrow n^2 - n + 1$$
$$\text{RLS}_\mu - \mathcal{QR} - \text{HDP - DLQR}_{Givens}$$
$$op^{(+/-)} \rightarrow \text{n}^2 - n$$
$$op^{(\cdot)} \rightarrow \frac{n^6}{8} + \frac{n^5}{8} + \frac{n^4}{8} - \frac{n^3}{8} + \frac{7n^2}{4} - 2n$$
$$op^{(\div)} \rightarrow \frac{n^4}{2} - n^3 + \frac{5n^2}{2} - 2n$$
$$\text{RLS}_\mu - \mathcal{QR} - \text{HDP - DLQR}_{Householder}$$
$$\frac{n^5}{4} + \frac{n^4}{4} - \frac{n^3}{4} - \frac{n^2}{4} + 2n - 2$$
$$\frac{n^7}{4} + \frac{n^6}{2} + \frac{n^5}{2} + \frac{5n^3}{4} - \frac{n^2}{4} + n - 3$$
$$\frac{3n^3}{2} - 2n^2 + \frac{3n}{2} - 1$$

Fazendo a substituição do valor de n por 4, sistema de 4ª ordem, encontra-se o valor real de *flops* usados na execução de cada algoritmo. O resultado é apresentado na Tabela 5.3.

Tabela 5.3: Resultado da avaliação do custo computacional

Algoritmo	*flops*
RLS$_\mu$-HDP-DLQR	2674
RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR$_{Givens}$	792
RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR$_{Householder}$	7108

Com base nos resultados apresentados anteriormente, observa-se que o algoritmo que requer menos *flops* para a sua execução é o RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR$_{Givens}$. isso é explicado pela falta de operações matriciais de grande ordem nesse algoritmo,as únicas matrizes que passam por multiplicação possuem ordem 2, independentemente da ordem do sistema,diminuindo em muito o custo computacional. Já o algoritmo RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR$_{Householder}$ apresentou um resultado bem pior comparado ao algoritmo RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR$_{Givens}$ e ao RLS$_\mu$-HDP-DLQR também, sendo assim o algoritmo com o maior custo computacional.

Com esse resultado pode se concluir que os algoritmo RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR$_{Givens}$ propostos nesse trabalho tem custo computacional 70% menor que o do algoritmo RLS$_\mu$-HDP-DLQR padrão.

5.2.2 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$_\lambda$-HDP-DLQR quando implementados de acordo com as Equações (4.10), (4.11) e (4.17), 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 Equação (4.11) perder sua positividade. Tal comportamento instável do RLS, o qual resulta de imprecisões numéricas devido ao uso do ambientes de precisão finito, é chamado "instabilidade numérica".

O problema da perda da positividade da matriz $\Gamma(\cdot)$ na implementação do algoritmo RLS$_\lambda$-HDP-DLQR tem sido contornado empregando-se métodos do tipo raiz quadrada (Fatoração UDU^T) (Rêgo et al., 2013) e como proposto nesse trabalho Decomposição $\mathcal{QR}$ que no caso a matriz avaliada é a $\Phi(\cdot)$. Estas abordagens tem custos computacionais menores com respeito ao número de operações arit-

méticas, requeridas na implementação do algoritmo, que se reflete na redução dos erros de arredondamento, tornando os algoritmos RLS$_\lambda$-HDP-DLQR menos sensíveis à perda da positividade da matriz $\Gamma(\cdot)$ e da matriz $\Phi(\cdot)$.

O problema da instabilidade numérica para o algoritmo RLS$_\lambda$-HDP-DLQR é abordado nessa subseção em termos das avaliações dos números de condição da matriz de covariância $\Gamma(\cdot)$ da estimativa RLS e do fator Cholesky da matriz $\Phi(\cdot)$, bem como do parâmetro de positividade.

Este problema pode ser superado usando transformações unitárias numericamente estáveis a cada iteração do algoritmo RLS. Em particular, a matriz $\Gamma(i)$ é propagada numa forma de raiz quadrada usando a fatoração de Cholesky

$$\Gamma(i) = \Gamma^{1/2}(i)\Gamma^{H/2}(i),$$

em que $\Gamma^{1/2}(i)$ é uma matriz triangular inferior, e $\Gamma^{H/2}(i)$ é a sua transposta Hermitiana.

Em álgebra linear, o fator de Cholesky $\Gamma^{1/2}(i)$ é comumente referido como a raiz quadrada da matriz $\Gamma(i)$.

O ponto importante a observar aqui é que o produto matricial $\Gamma^{1/2}(i)\Gamma^{H/2}(i)$ é muito menos provável de se tornar indefinido, porque o produto de qualquer matriz quadrada por sua transposta Hermitiana é sempre definido positivo. De fato, mesmo na presença de erros de arredondamento, o condicionamento numérico do fator de Cholesky $\Gamma^{1/2}(i)$ é geralmente bem melhor do que aquela da própria $\Gamma(i)$

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

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

Com efeito, mostra-se a relação entre os números de condição da matriz de covariância $\Gamma(i)$ com o do seu fator Cholesky $\Phi^{1/2}(i)$, a qual é dada por

$$\begin{aligned} cond\left[\Gamma(i)\right] &= cond\left[\Gamma^{1/2}(i)\Gamma^{H/2}(i)\right] \\ &= cond\left[\Gamma^{1/2}(i)\right]^2 \end{aligned} \tag{5.2}$$

da relação (5.2), observa-se que à medida que $cond(\Gamma(i))$ aumenta a robustez numérica é melhorada para os métodos raiz quadrada, a diferença entre os numeros

de condição das matrizes $\Gamma^{1/2}(i)$ e $\Gamma(i)$ torna-se significativa.

O algoritmo de estimação RLS também pode ser implementado na forma de raiz quadrada pela propagação da raiz quadrada $\Gamma^{-1/2}(i)$ ao invés da própria matriz inversa $\Gamma^{-1}(i)$. Nesta variante do RLS, a fatoração de Cholesky é usada para expressar a matriz inversa $\Gamma^{-1}(i)$ como segue:

$$\Gamma^{-1}(i) = \Gamma^{-H/2}(i)\Gamma^{-1/2}(i)$$

onde $\Gamma^{-1/2}(i)$ é uma matriz triangular inferior, e $\Gamma^{-H/2}(i)$ é a sua transposta Hermitiana

Uma outra medida para avaliar o problema da estabilidade numérica para o algoritmo RLS$_\lambda$-HDP-DLQR é a analise do parâmetro de positividade ρ da matriz $\Gamma(i)$, o qual é definido por.

$$\rho(i) = 1 - L^H(i)\bar{x}(i). \tag{5.3}$$

Usando a definição do vetor de ganho $L(i)$ de Equação (4.10) e o fato de que $\Gamma(i)$ é simétrica, isto é $\Gamma^T(i) = \Gamma(i)$, após simplificações a Equação (5.3) pode ser reescrita por

$$\rho(i) = \frac{\lambda}{\lambda + \bar{x}^H(i)\Gamma(i-1)\bar{x}(i)}. \tag{5.4}$$

Observando que a forma simétrica em (5.4) possui um valor real não-negativo, tem-se $0 < \rho \leqslant 1$. Portanto, uma condição necessária para que a matriz $\Gamma(i)$ seja definida positiva é dada por $0 < \rho < 1$. A ocorrência de um valor fora desta faixa indica que a matriz $\Gamma(i)$ perdeu a propriedade definida positiva.

O parâmetro de positividade $\rho(i)$ 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 ocorrência da divergência provocada por erros de quantização (erros de arredondamento).

Para estabelecer a relação entre esses dois erros de estimativa, começa-se com a definição do erro a posteriori da Equação (4.19) substituindo o valor de atualização pela Equação de atualização em (4.17), obtêm-se

$$\begin{aligned} e(i) &= d(i) - [\hat{\theta}(i-1) + L(i)\xi^*(i)]^H \bar{x}(i) \\ &= d(i) - \hat{\theta}^H(i-1)\bar{x}(i) - L^H(i)\bar{x}(i)\xi(i) \\ &= (1 - L^H(i)\bar{x}(i))\xi(i), \end{aligned} \tag{5.5}$$

onde, na última linha, faz-se o uso da definição dada na Equação (4.18). A razão entre a estimativa a posteriori com a estimativa a priori, é chamado o fator de conversão, denotado por ρ. que pode, assim, escrever

$$\begin{aligned} \rho(i) &= \frac{e(i)}{\xi(i)} \\ &= 1 - L^H(i)\bar{x}(i) \end{aligned} \tag{5.6}$$

cujo valor é determinado exclusivamente pelo vetor de ganho $L(i)$ e o vetor de derivação de entrada $\bar{x}(i)$

5.2.3 Tempo de Convergência

É o tempo decorrido necessário para um algoritmo encontrar uma solução aceitável, ou seja, onde o erro de regime tem valor menor que o erro de projeto. O tempo de convergência está diretamente relacionado a capacidade computacional da máquina que está executando esse algoritmo, por isso em casos que não se pode controlar essa velocidade de processamento esse parâmetro deve ser medido por interações necessárias para encontrar a solução.

Assim, quanto menos iterações o algoritmo precisar para encontrar a solução, menor será seu tempo de convergência e mais rápido será sua resposta para o sistema. Contando com sistemas que encontram rapidamente a solução para problemas de controle pode-se, então, almejar realizar controle em tempo real.

Na ciência da computação, a expressão **em tempo real**, se refere a sistemas em que o tempo de execução de uma determinada tarefa é rígido independente da carga do sistema. O tempo de execução de uma operação pode ser muito curto ou não. O que importa para este tipo de sistema é que a tarefa seja executada.

5.3 Considerações Finais

Neste Capítulo foi apresentado conceitos de complexidade computacional com enfoque no estudo dos parâmetros de avaliação: Custo Computacional, Estabilidade numérica e Tempo de Convergência, que serviram de métrica para a analise dos algoritmos apresentados no Capítulo 4, visando assim, que esses algoritmos tenham características necessárias para aplicações em tempo real em sistemas microprocessados como: FPGA's (Field Programmable Gate Array), DSP's (Digital Signal Processor) e PSoC's (Programmable System-on-Chip).

Capítulo 6

Teste Computacional

6.1 Introdução

Os procedimentos para avaliar o desempenho dos algoritmos RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para o projeto de controladores via HDP são estabelecidas levando em consideração a inicialização do processo iterativo bem como a política de referência, que é estabelecida pela solução offline de Schur da equação HJB-Riccati. As avaliações verificam o comportamento da convergência dos estimadores RLS-HDP, tendo como referência a complexidade computacional e a estabilidade numérica, bem como as características de polarização e a consistência dos estimadores acima mencionados.

Assim, os testes computacionais foram realizados via $MATLAB^{®}$ para os algoritmos RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR e, em seguida, foram realizadas comparações para avaliar a precisão das soluções fornecidas por esses algoritmos. Para várias simulações realizadas, verificou-se a influência do fator de esquecimento λ no processo de convergência e na estabilidade numérica dos algoritmos RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para a solução da equação HJB-Riccati para diferentes valores de λ.

A fim de assegurar que a comparação do desempenho dos algoritmos RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR seja "justa"no que diz respeito a variações no valor do fator de esquecimento, os outros parâmetros foram mantidos constantes e iguais para ambos os algoritmos. Por exemplo, os valores iniciais das matrizes de correlação e de correlação inversa (covariância) são mantidos equivalentes de

acordo com a seguinte relação:

$$\Gamma(i) \rightarrow \Phi^{1/2}(i).$$

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

Os experimentos de simulação são apresentados para três sistemas dinâmicos MIMO que avaliam a funcionalidade dos estimadores RLS baseados em decomposição $\mathcal{QR}$ de funções valor de estado para o projeto do sistema de controle online HDP-DLQR, e verifica-se a estabilidade e convergência dos algoritmos RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR. No primeiro sistema dinâmico, os resultados serão explorados no sistema de terceira ordem para um projeto de piloto automático de uma aeronave F-16 dado por (Lewis and Vrabie 2009*c*). No segundo sistema, os resultados do controle das tensões através dos capacitores e das correntes que passam através dos indutores pela variação das tensões nas fontes de alimentação de um circuito eléctrico de quarta ordem são mostrados, ver (Fonseca Neto *et al.* 2010). Finalmente, um modelo de sexta ordem para o controle de um DFIG (Gerador de Indução Duplamente Alimentado) em plantas eólicas é apresentado, (Viana da Fonseca Neto *et al.* 2013).

6.2 Modelo da Aeronave F-16

O modelo da aeronave pode ser visto em (Lewis and Liu 1992), este modelo linearizado corresponde à dinâmica longitudinal. Neste modelo, as equações da dinâmica foram modificadas para incluir uma segunda ação de controle. Assim, a dinâmica longitudinal da aeronave foram transformadas em um sistema MIMO.

A Figura (6.1) mostra as variáveis de estado que descrevem a dinâmica longitudinal da aeronave e que são definidas como

- A taxa de inclinação q, sendo $q = \dot{\beta}$ a derivada em relação ao tempo do ângulo de inclinação (β);

- O ângulo de ataque α, que é o ângulo formado entre a trajetória da aeronave e o sentido da linha da asa;
- O ângulo de deflexão de elevação (δ_e).

As saídas são os próprios estados. De acordo com a Figura (6.1), a representação das variáveis do espaço de estado do modelo da aeronave, é dada por

$$x_k = \begin{bmatrix} \alpha & q & \delta_e \end{bmatrix}^T = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T \tag{6.1}$$

As ações de controle são dadas pelo sinal $u_k = [\ u_1 \quad u_2\ ]^T$ que é enviado para o sistema de elevação e o sistema de atuação adicional (flaperon).

Figura 6.1: Movimento longitudinal da aeronave, levantar e baixar o nariz

O modelo em espaço de estado de tempo discreto da dinâmica da aeronave é uma versão discretizada do modelo de tempo contínuo, encontrado em (Lewis and Vrabie 2009*c*), onde as matrizes do sistema dinâmico são

$$A = \begin{bmatrix} -1.10188 & 0.90528 & -0.00212 \\ 4.0639 & -0.77013 & -0.16919 \\ 0 & 0 & -10 \end{bmatrix} \quad e \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 10 \\ 10 & 0 \end{bmatrix}. \tag{6.2}$$

As matrizes do sistema discretizado para o intervalo de amostragem $T_{amost} = 0.1$ são obtidas pelo método de *segurador de ordem zero*.

6.2.1 Simulação com $\lambda = .55$

RLS$_\lambda$-HDP-DLQR

A Figura 6.2 mostra o comportamento de convergência do algoritmo RLS$_\lambda$-HDP-DLQR para a solução da equação HJB-Riccati, com fator de esquecimento

$\lambda = 0.55$. Pode ser observado que o parâmetro P converge, no máximo, antes de 100 iterações, e depois de convergir ele não sofre nenhuma perturbação. No entanto, o estimador causa um grande *overshoot* no início do processo iterativo.

Figura 6.2: Evolução do processo iterativo para os parâmetros p_{11}, p_{33}, p_{13} e p_{23} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = 0.55$ - Algoritmo RLS$_\lambda$-HDP-DLQR.

Nota-se na Figura 6.3 que durante o processo iterativo de ajuste do parâmetro P, que nos dois casos a matriz Γ mostra um bom comportamento em relação à positividade parâmetro ρ, considerando-se que os valores de ρ devem estar no intervalo $0 < \rho < 1$, satisfazendo assim a condição necessária de positividade. Além disso, o número de condição não apresenta mudanças bruscas de seus valores, que são mantidos em um determinado intervalo após o período transitório.

Figura 6.3: Número de condição da matriz de covariância Γ e parâmetro positividade ρ, com fator de esquecimento $\lambda = 0.55$ para um ciclo de 5000 iterações.

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Quanto ao comportamento de convergência do estimador RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para o fator de esquecimento $\lambda = 0.55$, pode se observar que a convergência do processo de estimação do parâmetro P é atingido em torno da iteração 50, tal como mostrado na Figura 6.4, depois de ter um rápido *overshoot* no início do período transitório, porém, esse *overshoot* foi muito menor do que o encontrado no algoritmo RLS$_\lambda$-HDP-DLQR.

Figura 6.4: Evolução do processo iterativo para os parâmetros p_{11}, p_{33}, p_{13} e p_{23} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = 0.55$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR.

Observando o comportamento do parâmetro ρ, Na Figura 6.5, verifica-se que os valores de ρ também satisfazem a condição necessária de positividade durante todo o processo iterativo de parâmetro P, que é, $0 < \rho < 1$, confirmando a boa estabilidade do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR, e trabalhar com o fator de Cholesky da matriz de correlação $\Phi(i)$, $\Phi^{1/2}(i)$, o valor máximo do número de condicionamento foi reduzido a metade, em comparação com a do algoritmo RLS$_\lambda$-HDP-DLQR.

Figura 6.5: Número da condição da matriz Φ (fator de Cholesky) e parâmetro de positividade ρ, com fator de esquecimento $\lambda = 0.55$ para um ciclo de 5000 iterações.

6.2.2 Simulação com $\lambda = .92$

RLS$_\lambda$-HDP-DLQR

A evolução do processo iterativo para a solução da equação HJB-Riccati é mostrado na Figura 6.6, no caso $\lambda = 0.92$, quando o estimador RLS$_\lambda$-HDP-DLQR foi usado. Observa-se que o parâmetro P alcança a convergência antes da iteração 200, depois de passar por um *overshoot* no início do processo iterativo.

Figura 6.6: Evolução do processo iterativo para os parâmetros p_{11}, p_{33}, p_{13} e p_{23} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = 0.92$ - Algoritmo RLS$_\lambda$-HDP-DLQR.

A partir da Figuras 6.7 percebe-se que durante o processo iterativo de ajuste do parâmetro P, que nos dois casos a matriz Γ mostra um bom comportamento em relação à positividade parâmetro ρ, considerando-se que os valores de ρ devem estar no intervalo $0 < \rho < 1$, satisfazendo assim a condição necessária de positividade. Além disso, o número de condição não apresenta mudanças bruscas de seus valores, que são mantidos em um determinado intervalo após o período transitório.

Figura 6.7: Número de condição da matriz de covariância Γ e parâmetro positividade ρ, com fator de esquecimento $\lambda = 0.92$ para um ciclo de 5000 iterações.

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

A Figura 6.8 mostra a convergência do parâmetro P para o fator de esquecimento $\lambda = 0.92$ utilizando o estimador RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR. Nesta simulação, o número de iterações necessárias para a convergência foi de cerca de 400 (maior do que a do algoritmo padrão), com um *overshoot* inicial significativo, mas menor do que o apresentado pelo estimador RLS$_\lambda$-HDP-DLQR.

Figura 6.8: Evolução do processo iterativo para os parâmetros p_{11}, p_{33}, p_{13} e p_{23} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = 0.92$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR.

Pode ser visto a partir da Figura 6.9 que os valores de ρ, novamente, encontram-se dentro de $(0 < \rho < 1)$ para o estimador, mostrando que o algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS) fornece uma boa estabilidade para estimar a função de valor DLQR para o sistema de terceira ordem, e mais uma vez o estimador RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS) teve melhores resultados em comparação com o estimador RLS$_\lambda$-HDP-DLQR.

Figura 6.9: Número da condição da matriz Φ (fator de Cholesky) e parâmetro de positividade ρ, com fator de esquecimento $\lambda = 0.92$ para um ciclo de 5000 iterações.

6.3 Modelo do Circuito Elétrico MIMO

O circuito elétrico de quarta ordem usado para avaliar o desempenho da metodologia proposta é mostrado no diagrama da Figura 6.10. O circuito é um sistema de componentes elétricos passivos constituído por capacitores, indutores e resistores.

Figura 6.10: Circuito RLC de 4ª ordem

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

$$C\frac{dx_1}{dt} + \frac{x_1}{R_1} - x_3 = 0 \tag{6.3}$$

$$C\frac{dx_2}{dt} + \frac{x_2}{R_1} - x_4 = 0 \tag{6.4}$$

$$L\frac{dx_3}{dt} + x_1 + R_2(x_3 + x_4) - u_1 = 0 \tag{6.5}$$

$$L\frac{dx_4}{dt} + x_2 + R_2(x_3 + x_4) - u_2 = 0, \tag{6.6}$$

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.

A descrição em espaço de estados correspondente ao modelo das Equações (6.3) a (6.6) é dada por:

$$\dot{x} = Ax + Bu, \quad t \geqslant t_0. \tag{6.7}$$

sendo A a matriz de estado,

$$A = \begin{bmatrix} -1/CR_1 & 0 & 1/C & 0 \\ 0 & -1/CR_1 & 0 & -1/C \\ -1/L & 0 & -R_2/L & -R_2/L \\ 0 & 1/L & -R_2/L & -R_2/L \end{bmatrix} \tag{6.8}$$

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

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1/L & 0 \\ 0 & 1/L \end{bmatrix} \tag{6.9}$$

e

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.10}$$

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

$$A_c = \begin{bmatrix} -0.001 & 0 & 10 & 0 \\ 0 & -0.001 & 0 & -10 \\ -0.1 & 0 & -10 & -10 \\ 0 & 0.1 & -10 & -10 \end{bmatrix} \tag{6.11}$$

e

$$B_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{6.12}$$

As matrizes do sistema discretizado para o intervalo de amostragem $T_{amost} = 0.1$ são obtidas pelo método de discretização *Impulse-invariant*.

6.3.1 Simulação com $\lambda = .72$

RLS$_\lambda$-HDP-DLQR

Nas Figuras 6.11, 6.12 e 6.13, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .72$. As curvas (a)-(d) da Figura 6.11 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 6.12 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)-(c) da Figura 6.13, respectivamente.

Figura 6.11: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.12: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.13: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Na Figura 6.14, é observado que, mesmo que o parâmetro de positividade assuma valores fora da sua gama de validade ($0 < \rho < 1$) e o número de condição apresentar variações drásticas em seus valores após o estimador ter atingido o regime permanente, estes comportamentos já não tem qualquer efeito sobre a perda de estabilidade do parâmetro P. Primeiramente, é necessário considerar como hipótese que os termos $L_i\xi_i$ assumem valores muito pequenos (próximo a zero), e como o erro de estimação ξ_i tendendo a zero, a norma do vetor L_i pode assumir uma gama muito ampla de valores, sem grandes alterações nos componentes do parâmetro P. Isso faz com que o produto $L_i\xi_i$ permaneça com limites relativamente aceitáveis, mantendo os estimadores indefinidamente estáveis.

Figura 6.14: Número de condições da matriz Γ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-HDP-DLQR para $\lambda = 0.72$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Nas Figuras 6.15, 6.16 e 6.17, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .72$. As curvas (a)-(d) da Figura 6.15 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 6.16 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)-(c) da Figura 6.17, respectivamente.

Figura 6.15: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Figura 6.16: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Figura 6.17: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Mostra-se na Figura 6.18, o número de condição do fator Cholesky da matriz de correlação $\Phi(i)$, $\Phi^{1/2}(i)$, e o parâmetro de positividade ρ do processo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para estimação da solução da equação HJB-Riccati. Observa-se que os valores de ρ satisfazem a condição necessária da propriedade definida positiva durante todo o processo iterativo de parâmetro P, ou seja, $0 < \rho < 1$. Quanto ao número de condição, nota-se valor substancialmente mais baixo comparado ao número de condição da matriz de covariância do estimador padrão RLS$_\lambda$-HDP-DLQR. Além disso, seus valores são mantidos limitados a um determinado intervalo após um período de tempo à transição.

Figura 6.18: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.72$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Nas Figuras 6.19, 6.20 e 6.21, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .72$. As curvas (a)-(d) da Figura 6.19 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 6.20 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)-(c) da Figura 6.21, respectivamente.

Figura 6.19: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.20: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.21: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .72$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Na Figura 6.22 mostra-se o número de condição do fator Cholesky da matriz de correlação $\Phi(i)$, $\Phi^{1/2}(i)$, e o parâmetro de positividade ρ do processo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para estimação da solução da equação HJB-Riccati. Observa-se que os valores de ρ satisfazem a condição necessária da propriedade definida positiva durante todo o processo iterativo de parâmetro P, ou seja, $0 < \rho < 1$.

Figura 6.22: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.72$

Analisando os resultados para o fator de esquecimento $\lambda = .72$, percebe-se que os três algoritmos alcançaram a convergência, porém, os algoritmos RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS) e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER) apresentaram comportamentos idênticos, atingindo a convergência ao mesmo tempo, já o algoritmo RLS$_\lambda$-HDP-DLQR convergiu somente depois de algumas iterações.

6.3.2 Simulação com $\lambda = .82$

RLS$_\lambda$-HDP-DLQR

Nas Figuras 6.23, 6.24 e 6.25, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .82$. As curvas (a)-(d) da Figura 6.23 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 6.24 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)-(c) da Figura 6.25, respectivamente.

Figura 6.23: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.24: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.25: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Observa-se na Figura 6.26, que, embora o parâmetro de positividade assuma valores fora da sua gama de validade ($0 < \rho < 1$) e o número de condição apresentar variações drásticas em seus valores após o estimador ter atingido o regime permanente, estes comportamentos já não tem qualquer efeito sobre a perda de estabilidade do parâmetro P. Primeiramente, é necessário considerar como hipótese que os termos $L_i\xi_i$ assumem valores muito pequenos (próximo a zero), e como

o erro de estimação ξ_i tendendo a zero, a norma do vetor L_i pode assumir uma gama muito ampla de valores, sem grandes alterações nos componentes do parâmetro P. Isso faz com que o produto $L_i\xi_i$ permaneça com limites relativamente aceitáveis, mantendo os estimadores indefinidamente estáveis.

Figura 6.26: Número de condições da matriz Γ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-HDP-DLQR para $\lambda = 0.82$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Nas Figuras 6.27, 6.28 e 6.29, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .82$. As curvas (a)-(d) da Figura 6.27 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 6.28 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)-(c) da Figura 6.29, respectivamente.

Figura 6.27: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Figura 6.28: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Figura 6.29: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

A Figura 6.30 mostra o número de condição do fator Cholesky da matriz de correlação $\Phi(i)$, $\Phi^{1/2}(i)$, e o parâmetro de positividade ρ do processo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para estimação da solução da equação HJB-Riccati. Observa-se que os valores de ρ satisfazem a condição necessária da propriedade definida positiva durante todo o processo iterativo de parâmetro P, ou seja, $0 < \rho < 1$. Quanto ao número de condição, nota-se um valor substancialmente mais baixo comparado ao número de condição da matriz de covariância do estimador padrão RLS$_\lambda$-HDP-DLQR. Além disso, seus valores são mantidos limitados a um determinado intervalo após um período de tempo à transição..

Figura 6.30: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.82$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Nas Figuras 6.31, 6.32 e 6.33, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .82$. As curvas (a)-(d) da Figura 6.31 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 6.32 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)-(c) da Figura 6.33, respectivamente.

Figura 6.31: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.32: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.33: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .82$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

O comportamento do número de condição e parâmetro positividade ρ estão ilustrados na Figura 6.34, observa-se que o comportamento de ρ não mostra irregularidades, estabilizando em 1500 iterações, bem como, o número de condição da matriz $\Phi^{1/2}(i)$ exibe um comportamento sem mudanças abruptas, e com valores parecidos aos da decomposição $\mathcal{QR}$ (GIVENS) com um pequeno atraso.

Figura 6.34: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.82$

Analisando os resultados para o fator de esquecimento $\lambda = .82$, percebe-se que os dois algoritmos RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR chegaram a convergência, no entanto, o algoritmo RLS$_\lambda$-HDP-DLQR chegou a essa mais rápido e de forma mais suave em comparação aos algoritmos RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR.

6.3.3 Simulação com $\lambda = .874$

Utilizando o fator de esquecimento de $\lambda = .874$

RLS$_\lambda$-HDP-DLQR

Nas Figuras 6.35, 6.36 e 6.37, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .874$. As curvas (a)-(d) da Figura 6.35 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 6.36 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)-(c) da Figura 6.37, respectivamente.

Figura 6.35: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.36: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.37: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Anormalidade no comportamento da matriz de covariância foram evidenciados pouco depois da iteração 1000, em torno de iteração 1112, quando o valor do parâmetro de positividade extrapolou a gama de validade, com duas grandes variações em torno das iterações 1550 e 4700, como ilustrado na Figura 6.38. também é visto na mesma figura, grandes valores na matriz de covariância Γ.

Figura 6.38: Número de condições da matriz Γ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-HDP-DLQR para $\lambda = 0.874$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Nas Figuras 6.39, 6.40 e 6.41, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .874$. As curvas (a)-(d) da Figura 6.39 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 6.40 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)-(c) da Figura 6.41, respectivamente.

Figura 6.39: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR

Figura 6.40: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR

Figura 6.41: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR

Observa-se na Figura 6.42 que o número de condição da matriz $\Phi^{1/2}(i)$ apresenta um comportamento sem variações bruscas de seus valores, e que, mais uma vez, são mantidos limitados para um determinado intervalo após um período de convergência, e com valores notavelmente inferiores quando comparados com o número de condição da matriz de covariância do estimador padrão RLS$_\lambda$-HDP-DLQR.

Figura 6.42: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.874$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Nas Figuras 6.43, 6.44 e 6.45, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .874$. As curvas (a)-(d) da Figura 6.43 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 6.44 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)-(c) da Figura 6.45, respectivamente.

Figura 6.43: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.44: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.45: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .874$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

A Figura 6.46 mostra o número de condição do fator Cholesky da matriz de correlação $\Phi(i)$, $\Phi^{1/2}(i)$, e o parâmetro de positividade ρ do processo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para estimação da solução da equação HJB-Riccati. Observa-se que os valores de ρ satisfazem a condição necessária da propriedade definida positiva durante todo o processo iterativo de parâmetro P, ou seja, $0 < \rho < 1$.

Figura 6.46: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.874$

Analisando os resultados para o fator de esquecimento $\lambda = .874$, percebe-se que apenas os algoritmos RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR alcançaram a convergência, antes de 1200 iterações, e de forma consideravelmente suave, com Householder apresentando mais variações durante o treinamento, já o algoritmo RLS$_\lambda$-HDP-DLQR após 3000 iterações perdeu completamente a estabilidade, isso pode ter sido causado pela falta de estabilidade numérica necessária para o cálculo da inversa do RLS.

6.3.4 Simulação com $\lambda = .905$

Utilizando o fator de esquecimento de $\lambda = .905$

RLS$_\lambda$-HDP-DLQR

Nas Figuras 6.47, 6.48 e 6.49, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .905$. As curvas (a)-(d) da Figura 6.47 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 6.48 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)-(c) da Figura 6.49, respectivamente.

Figura 6.47: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.48: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.49: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Nota-se também na Figura 6.50, variações significativas no número de condição da matriz de covariância, bem como irregularidades no parâmetro de positividade ρ. Anormalidade no comportamento deste parâmetro foi evidenciada antes mesmo da iteração 3300, e logo após, os valores de ρ sofreram variações com um grande valor negativo na iteração 4250.

Figura 6.50: Número de condições da matriz Γ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-HDP-DLQR para $\lambda = 0.905$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Nas Figuras 6.51, 6.52 e 6.53, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .905$. As curvas (a)-(d) da Figura 6.51 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 6.52 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)-(c) da Figura 6.53, respectivamente.

Figura 6.51: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Figura 6.52: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Figura 6.53: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

O comportamento do número de condição e parâmetro positividade ρ estão ilustrados na Figura 6.54, observa-se que o comportamento de ρ não mostra irregularidades, bem como o número de condição da matriz $\Phi^{1/2}(i)$ exibe um comportamento sem mudanças abruptas, e com menores valores em relação aos apresentados pelo estimador RLS$_\lambda$-HDP-DLQR.

Figura 6.54: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.905$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Nas Figuras 6.55, 6.56 e 6.57, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .905$. As curvas (a)-(d) da Figura 6.55 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 6.56 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)-(c) da Figura 6.57, respectivamente.

Figura 6.55: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.56: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.57: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .905$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Mais uma vez, o parâmetro de positividade ρ do processo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para estimação da solução da equação HJB-Riccati da Figura 6.58, respeitou a propriedade definida positiva, durante todo o processo iterativo de parâmetro P, ou seja, $0 < \rho < 1$.

Figura 6.58: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.905$

Analisando os resultados para o fator de esquecimento $\lambda = .905$, pode se dizer

que os dois algoritmos chegaram a convergência porém, nesse exemplo mostra-se a vantagem dos algoritmos RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR em relação ao algoritmo RLS$_\lambda$-HDP-DLQR, os algoritmos com decomposição chegaram a convergência antes de 1000 iterações e após a convergência quase não sofreram pertubações, isso é devido a uma matriz bem condicionada pela decomposição $\mathcal{QR}$. Já o algoritmo sem decomposição sofreu com pertubações depois do transitório conseguindo a convergência plena após a iteração 4000.

6.3.5 Simulação com $\lambda = .915$

Utilizando o fator de esquecimento de $\lambda = .915$

RLS$_\lambda$-HDP-DLQR

Nas Figuras 6.59, 6.60 e 6.61, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .915$. As curvas (a)-(d) da Figura 6.59 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 6.60 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)-(c) da Figura 6.61, respectivamente.

Figura 6.59: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.60: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-HDP-DLQR

Figura 6.61: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-HDP-DLQR

O números de condição e o parâmetro positividade do estimador RLS$_\lambda$-HDP-DLQR é apresentada na Figura 6.62. Observa-se que o parâmetros de positividade apresenta irregularidades quando ρ assume valores fora do intervalo $0 < \rho < 1$ para algumas iterações do processo de estimativa do parâmetro P, indicando perda de positividade da matriz de covariância Γ. Nota-se que os números de condição de $\Gamma(i)$ são muito grandes, o que pode levar a erros numéricos mais altos.

Figura 6.62: Número de condições da matriz Γ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-HDP-DLQR para $\lambda = 0.915$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Nas Figuras 6.63, 6.64 e 6.65, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .915$. As curvas (a)-(d) da Figura 6.63 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 6.64 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)-(c) da Figura 6.65, respectivamente.

Figura 6.63: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Figura 6.64: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

Figura 6.65: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (GIVENS)

O número de condição e os parâmetros positividade do processo de estimação RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR é apresentado na Figura 6.66, Com respeito aos parâmetros de positividade, observa-se que os valores de ρ encontram-se no intervalo de validade durante todo o processo iterativo.

Figura 6.66: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.915$

RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Nas Figuras 6.67, 6.68 e 6.69, apresenta-se a evolução do processo iterativo da solução da HJB-Riccati para um ciclo de 5000 iterações com fator de esquecimento $\lambda = .915$. As curvas (a)-(d) da Figura 6.67 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 6.68 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)-(c) da Figura 6.69, respectivamente.

Figura 6.67: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.68: Evolução do processo iterativo para os parâmetros p_{12}, p_{13}, p_{14} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Figura 6.69: Evolução do processo iterativo para os parâmetros p_{23}, p_{24}, p_{34} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = .915$ - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR (HOUSEHOLDER)

Anormalidade no comportamento da matriz covariância foram evidenciados pouco depois da iteração 1000, em torno de iteração 1200, quando o valor do parâmetro de positividade extrapolada a gama de validade, como ilustrado na Figura 6.70. também é visto na mesma figura, grandes valores no numero de condições do fator Cholesky de Φ .

Figura 6.70: Número de condições da raiz quadrada da matriz Φ e parâmetro de positividade ρ do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para $\lambda = 0.915$

Analisando os resultados para o fator de esquecimento $\lambda = .915$, pode-se dizer que os dois algoritmos chegaram a convergência e mais uma vez o algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR teve melhor desempenho em relação ao algoritmo RLS$_\lambda$-HDP-DLQR, ambos chegaram a convergência próximo da iteração 1000, porém, o algoritmo com decomposição chegou ao resultado de forma mais suave, enquanto o algoritmo sem decomposição sofreu uma pertubação antes de alcançar a convergência plena.

Na próxima seção serão avaliados: O numero de condição (no algoritmo RLS$_\lambda$-HDP-DLQR a matriz a ser analisada é Γ, já no algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR é a raiz quadrada de Φ) e o parâmetro de positividade ρ. Esses dois parâmetros serão analisados a fim de estudar a estabilidade numérica dos algoritmos.

6.4 Modelo do Gerador DFIG

O gerador de indução duplamente alimentado DFIG é uma máquina elétrica de indução com rotor bobinado alimentada com tensão alternada tanto no rotor quanto no estator, utilizando dois conversores interligados por um circuito capacitivo, sendo um conversor conectado à rede e outro aos terminais do rotor.

Este tipo de gerador possui uma propriedade particular. Quando o gerador opera abaixo da velocidade síncrona da maquina, no estator é gerada potência enquanto no rotor é consumida potência. Quando o gerador opera acima da velocidade síncrona, é gerada potência tanto pelo estator quanto pelo rotor.

O conversor do lado da máquina, interligado ao rotor, controla a velocidade do rotor e a potência reativa no estator, enquanto o conversor interligado ao lado da rede controla a tensão no barramento **cc** e a potência reativa que é trocada entre o conversor e a rede (Boldea 2006).

Figura 6.71: Esquema Elétrico do Gerador de induçao duplamente alimentado

O modelo de espaço-estado no tempo continuo da dinâmica do DFIG pode ser encontrado em (Pinto 2007), e as matrizes de sistema dinâmicos são:

$$A = \begin{bmatrix} -101.49 & 193.74 & 0 & 0 & 0 & 0 \\ -193.74 & -101.49 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.15 & 0 & 0 & 0 \\ 0 & 0 & 0 & -250 & 377 & 0 \\ 0 & 0 & 0 & -377 & 250 & 0 \\ 0 & 0 & 0 & -767.72 & 0 & 22.73 \end{bmatrix} \tag{6.13}$$

$$B = \begin{bmatrix} 33.97 & 0 & 0 & 0 & 0 & 0 \\ 0 & 33.97 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & -83.33 & 0 & 0 \\ 0 & 0 & 0 & 0 & -83.33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 454.54 \end{bmatrix}. \tag{6.14}$$

As matrizes do sistema discretizado para o intervalo de amostragem $T_{amost} = 0.1$ são obtidas através do método *retentor-de-ordem-zero*.

Para esta planta em particular, a fim de manter o vetor de regressão $\bar{x}$ persistentemente excitado, a revitalização de estado é gerada através de uma sequência pseudo-aleatória uniformemente distribuída. O algoritmo tem a opção de alterar a semente do gerador de números pseudo-aleatórios utilizados ao longo do processo de estimativa do parâmetro P. A mesma semente foi tomada na execução dos dois algoritmos para garantir que a função pseudo-aleatórios forneça a mesma sequência de revitalização de estado para ambos os algoritmos. Os seguintes resultados são apresentados para as duas sementes diferentes para a geração do vetor de regressão $\bar{x}$ com a revitalização de estado.

6.4.1 Sequência de Revitalização de Estado 1

A evolução do processo iterativo para a solução da equação HJB-Riccati do algoritmo RLS$_\lambda$-HDP-DLQR é mostrado na Figura 6.72 para um ciclo de 5000 iterações, com o fator de esquecimento $\lambda = 0,983$. As curvas (a)-(f) representam o comportamento da convergência dos elementos p_{11}, p_{22}, p_{33}, p_{44}, p_{55} e p_{66} da matriz P correspondente aos componentes θ_1, θ_7, θ_{12}, θ_{16}, θ_{19} e θ_{21} do vetor de parâmetros θ, respectivamente. Observa-se que houve a convergência do parâmetro θ com esta sequência, mas com grandes oscilações durante o período transitório, e a necessidade de várias iterações para alcançar a convergência.

Figura 6.72: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44}, p_{55} e p_{66} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = 0.983$ e sequência de revitalização de estado 1 - Algoritmo RLS$_\lambda$-HDP-DLQR.

Tal como no caso anterior, as curvas (a)-(f) da Figura 6.73 representa o comportamento de convergência do estimador RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para um ciclo de 5000 iterações, usando a mesma semente. Verificou-se que os parâmetros θ convergiram para a vizinhança do valor verdadeiro, com uma solução estável e aceitável considerando que o regime de erro foi dado por $\left|\theta_0 - \hat{\theta}_0\right| < 10^{-3}$.

Figura 6.73: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44}, p_{55} e p_{66} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = 0.983$ e sequência de revitalização de estado 1 - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR.

Os números de condição e os parâmetros de positividade dos estimadores RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR são apresentados nas Figuras 6.74 e 6.75, respectivamente. Observa-se que, para ambos os estimadores, os valores do parâmetro de positividade ρ oscilaram indefinidamente na faixa $(0, 1)$, mostrando irregularidade acentuada para o estimador RLS$_\lambda$-HDP-DLQR quando o parâmetro ρ assume periodicamente o valor igual a 1 até aproximadamente a iteração 2000.

Figura 6.74: Número de condição da matriz de covariância Γ e o parâmetro de positividade ρ, com fator de esquecimento $\lambda = 0{,}983$ para um ciclo de 5000 iterações - Sequencia de Revitalização Estado 1.

Figura 6.75: Número de condição da matriz Φ (Fator Cholesk) e o parâmetro de positividade ρ, com fator de esquecimento $\lambda = 0{,}983$ para um ciclo de 5000 iterações - Sequencia de Revitalização Estado 1.

6.4.2 Sequência de Revitalização de Estado 2

A Figura 6.76 mostra a evolução para o processo iterativo da solução da equação HJB-Riccati através do estimador RLS$_\lambda$-HDP-DLQR para os parâme-

tros p_{11}, p_{22}, p_{33}, p_{44}, p_{55} e p_{66} da matriz P, com fator de esquecimento $\lambda = 0.983$. Observou-se que não existe uma convergência dos parâmetro P, o qual exibe um comportamento com perturbações abruptas durante todo o processo iterativo.

Figura 6.76: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44}, p_{55} e p_{66} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = 0.983$ e sequência de revitalização de estado 2 - Algoritmo RLS$_\lambda$-HDP-DLQR.

A Figura 6.77 mostra o comportamento de convergência do estimador RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para um ciclo de 5000 iterações. Observou-se que os parâmetros da matriz P convergiram com menos de 1000 iterações, mas tiveram overshoots significativos no início do período transitório.

Figura 6.77: Evolução do processo iterativo para os parâmetros p_{11}, p_{22}, p_{33}, p_{44}, p_{55} e p_{66} para um ciclo de 5000 iterações, com fator de esquecimento $\lambda = 0.983$ e sequência de revitalização de estado 2 - Algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR.

As Figuras 6.78 e 6.79 apresentam os números de condição e os parâmetros de positividade dos processos de estimação RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR. Mais uma vez, os valores do parâmetro de positividade ρ oscilaram indefinidamente na faixa $0 < \rho < 1$ para os dois estimadores, onde o estimador RLS$_\lambda$-HDP-DLQR mostrou um comportamento mais irregular, quando comparado com o estimador RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR uma vez que o parâmetro de positividade para o primeiro estimador leva periodicamente valor igual a 1 durante todo o processo iterativo, enquanto que a do segundo estimador apresenta tal valor somente nas primeiras iterações do processo de estimação de parâmetro P.

Figura 6.78: Número de condição da matriz de covariância Γ e o parâmetro de positividade ρ, com fator de esquecimento $\lambda = 0{,}983$ para um ciclo de 5000 iterações - Sequencia de Revitalização Estado 2.

Figura 6.79: Número de condição da matriz Φ (Fator Cholesk) e o parâmetro de positividade ρ, com fator de esquecimento $\lambda = 0{,}983$ para um ciclo de 5000 iterações - Sequencia de Revitalização Estado 2.

Os resultados das simulações mostram claramente que as estimativas dadas pelos algoritmos RLS$_\lambda$-HDP-DLQR e RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR para o sistema de terceira ordem apresentam, em geral, menor tempo de convergência (número de

iterações necessárias para chegar a uma dada precisão) e custo computacional reduzido por iteração quando comparados com os sistemas de quarta e sexta ordem. Esses resultados iniciais semelhantes para os dois algoritmos podem ser explicados pelo modelo apresentado inicialmente ser de pequena ordem, tendo apenas seis parâmetros a serem estimados. Assim, é necessário salientar que, para as plantas de ordem mais alta, o custo computacional é maior e o estimador tende a apresentar maiores sensibilidades a perturbações provocadas por problemas numéricos, sendo assim necessário o uso de métodos de decomposição para melhorar o comportamento do sistema.

6.5 Considerações Finais

Neste Capítulo foram apresentados os resultados dos algoritmos propostos no Capítulo 4 em relação a três diferentes plantas, tais como: Aeronave F16, Circuito RLC e Gerador DFIG. Os resultados mostraram que os algoritmos tem sua funcionalidade comprovada, para cada caso pode apresentar vantagens e desvantagens.

Levando em consideração os três parâmetros de avaliação de complexidade computacional têm-se:

Custo Computacional: Conforme mostrado no Capitulo 5, O algoritmo que necessita de menor quantidade de *flops* para sua execução é o RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR, isso garante ao executar uma iteração desse algoritmo leva-se menos tempo do que a do algoritmo padrão.

Estabilidade Numérica: Neste parâmetro o algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR, mais uma vez, levou vantagem, garantindo em quase todos os casos, fator de positividade ρ dentro da faixa de 0 a 1, e o numero de condição quando comparado com o algoritmo RLS$_\lambda$-HDP-DLQR mostraram-se sempre menores.

Tempo de Convergência: Neste requisito os dois algoritmos testados mostraram desempenho parecidos, em relação a quantidade de iterações necessárias para encontrar a solução desejada, contudo, deve se lembrar que o algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR é mais rápido por iteração, o que lhe da vantagem no cálculo real desse parâmetro.

Assim pode-se concluir que as vantagens na execução do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR são maiores do que as do algoritmo RLS$_\lambda$-HDP-DLQR.

Capítulo 7

Conclusões

7.1 Conclusões Gerais

No trabalho foi apresentado um método alternativo para estabelecer políticas de controle ótimas, baseadas na forma HJB da equação algébrica de Riccati, Bem como a teoria de aprendizagem por reforço aplicada a programação dinâmica, levando ao desenvolvimento da programação dinâmica heurística para a solução online de controle ótimo discreto do tipo linear quadrático.

E na tentativa de aproximar esses métodos de controle ótimo moderno ao mundo real, buscou-se otimizar a execução dos algoritmos gerados por essas técnicas de controle a fim de tornar possível uma aplicação em tempo real em microcontroladores, presentes em todo sistema de controle automático.

Na avaliação do desempenho dos algoritmos RLS$_\mu$-HDP-DLQR e RLS$_\mu$-$\mathcal{QR}$-HDP-DLQR, em relação à escolha do fator de esquecimento, observou-se uma grande influência desse fator no processo de convergência e estabilidade numérica na solução da equação HJB-Riccati. Verificou-se também que a escolha apropriada do fator juntamente com a decomposição $\mathcal{QR}$ podem contribuir significativamente para melhoria no processo de convergência da equação HJB-Riccati, assegurando ao mesmo tempo um limite aceitável para a estabilidade numérica dos estimadores RLS-HDP.

Os resultados das simulações mostraram-se promissores para os três modelos de sistemas dinâmicos MIMO, uma vez que a inserção da decomposição $\mathcal{QR}$ no processo de aprendizagem RLS da solução da equação HJB-Riccati via HDP con-

tribuiu para resolver os problemas de convergência e estabilidade numérica por uma escolha plausível do fator de esquecimento. Levando em consideração os resultados da simulação apresentados, futuros experimentos realizados em plantas do mundo real são encorajadores.

Provando que o algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR é uma boa proposta para um algoritmo de Controle Ótimo Adaptativo que possua total capacidade de implementação em ambiente industrial.

7.2 Principais Contribuições

Desenvolvimento de um algoritmo de Controle Ótimo Adaptativo que possui especificações adequadas para implementação no mundo real.

7.3 Trabalhos Futuros

Como trabalhos futuros pode-se citar o desenvolvimento do algoritmo RLS$_\lambda$-$\mathcal{QR}$-HDP-DLQR sistólico através da aplicação de técnicas de computação paralela para melhorar do desempenho computacional do mesmo, bem como a implementação em hardware (Microcontroladores) dos algoritmos de controle aqui propostos.

Capítulo 8

Publicações

8.1 Congressos

1. **Computational Performance of State-Value Function Approximators based on RLS-HDP Estimators for Online DLQR Control System Design**, UKSim-AMSS 18^{th} International Conference on Modelling & Simulation, Cambridge University - UK (Publicado).

8.2 Revistas

1. **Adaptability of HDP-based DLQR Controllers Design for MIMO Dynamic Systems**, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 11, November 2015, ISSN: 2277 128X (Publicado).

2. **Numerical Stability Improvements of State-Value Function Approximations based on RLS Learning for Online HDP-DLQR Control System Design**, Engineering Applications of Artificial Intelligence, ISSN: 0952-1976 (Submetido).

Apêndice A

Lemas

A.1 Lema da Fatoração da Matriz

Dada duas matrizes A e B, $N \times M$, sendo $N \leq M$, o Lema da Fatoração de Matriz estabelecido por (Stewart 1973), (Golub 1989) e (Sayed 1994) declara que

$$AA^H = BB^H \tag{A.1}$$

se, e somente se, existir uma matriz unitária Θ tal que

$$B = A\Theta \tag{A.2}$$

Assumindo que a condição (A.2) é mantida, prontamente se encontra que

$$BB^H = A\Theta\Theta^H A^H \tag{A.3}$$

A partir da definição de uma matriz unitária, tem-se

$$\Theta\Theta^H = I \tag{A.4}$$

onde I é a matriz identidade. Portanto, a Equação (A.3) reduz-se imediatamente a Equação (A.1).

Por outro lado, a igualdade descrita na Equação (A.1) implica que as matrizes A e B devem ser relacionadas. Pode-se provar a implicação da recíproca do Lema da fatoração da matriz invocando o teorema da decomposição do valor singular. De acordo com este teorema, a matriz A, pode ser tida como se segue

$$A = U_A \Sigma_A V_A^H \tag{A.5}$$

onde U_A e V_A representam matrizes unitárias $N \times N$ e $M \times M$, respectivamente, e Σ_A é uma matriz $N \times N$ definida pelos valores singulares da matriz A. Similarmente, a segunda matriz B pode ser obtida como se segue

$$B = U_B \Sigma_B V_B^H \tag{A.6}$$

A identidade $AA^H = BB^H$ implica que

$$U_A = U_B \tag{A.7}$$

e

$$\Sigma_A = \Sigma_B \tag{A.8}$$

Agora tem-se

$$\Sigma = V_A V_B^H \tag{A.9}$$

Em seguida, substituindo as Equações (A.5) e (A.7) na Equação (A.9) no produto matricial $A\Theta$ produz um resultado igual a matriz B em virtude da Equação (A.6), que é precisamente a implicação recíproca do lema da fatoração de matriz.

A.2 Lema da Inversão de Matriz

Sejam A e B duas matrizes $M \times M$ definidas positivas relacionadas por

$$A = B^{-1} + CD^{-1}C^H \tag{A.10}$$

em que D é uma outra matriz $N \times M$ definida positiva, e C é uma matriz $M \times N$. De acordo com o **lema da inversão de matrizes**, pode-se expressar a inversa da matriz A como segue:

$$A^{-1} = B - BC(D + C^H BC)^{-1} C^H B \tag{A.11}$$

A prova deste lema é estabelecida multiplicando-se a Equação (A.10) pela Equação (A.11) e reconhecendo que o produto de uma matriz quadrada pela sua

inversa é igual a matriz identidade. O Lema da inversão da matriz declara que dada uma matriz A como definida na Equação (A.10), a inversa A^{-1} pode ser determinada usando a relação da Equação (A.11).

Apêndice B

Decomposição $\mathcal{QR}$

B.1 Introdução

Nesta seção se discutira um tipo de decomposição de matrizes, que será utilizada na obtenção de soluções de problemas de mínimos quadrados.

A decomposição $\mathcal{QR}$ é uma operação elementar estável, que pode ser aplicada a qualquer tipo de matriz e consiste na fatoração de uma matriz dada em duas matrizes, uma matriz ortogonal e uma matriz triangular superior. Assim, a decomposição $\mathcal{QR}$ de uma matriz é apresentada como o produto de uma matriz ortogonal $\mathcal{Q}$ por uma matriz triangular superior $\mathcal{R}$.

Teorema 1

Seja A uma matriz $m \times n$ cujas colunas são vetores linearmente independentes. Então, A pode ser fatorada como:

$$A = \mathcal{QR},$$

sendo $\mathcal{Q}$ uma matriz $m \times n$ cujas colunas formam um conjunto ortonormal, e $\mathcal{R}$ é uma matriz triangular superior inversível $n \times n$.

Demonstração

Seja

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Denota-se por a_1, a_2, ..., a_n as colunas de A. Como elas são linearmente independentes, elas formam uma base para o espaço coluna de A.

Pode ser utilizado o processo de Gram-Schmidt para se obter um conjunto $\{q_1, q_2, ..., q_n\}$ de vetores ortonormais a partir das colunas de A.

Seja Q a matriz cujas colunas são q_1, q_2, ..., q_n , isto é,

$$Q = [q_1,\, q_2, \dots, q_n]$$

As colunas q_1, q_2, ..., q_n de Q formam um base ortonormal para o espaço coluna de A.

Teorema 2

Seja V um espaço vetorial com produto interno e v um vetor em V. Seja $B = \{v_1, v_2, ..., v_n\}$ uma base ortogonal de V. Então todo vetor $v \in V$ pode ser escrito como combinação linear da base B da seguinte forma:

$$v = \frac{\langle v, v_1\rangle}{\langle v_1, v_1\rangle} v_1 + \frac{\langle v, v_2\rangle}{\langle v_2, v_2\rangle} v_2 + \ldots + \frac{\langle v, v_n\rangle}{\langle v_n, v_n\rangle} v_n.$$

Os coeficientes $a_j = \frac{\langle v, v_j\rangle}{\langle v_j, v_j\rangle}$, $j = 1, 2, ..., n$, são chamados de coeficientes de Fourier de v.

Assim toda coluna a_i $i = 1, 2, ..., n$, é combinação linear dos vetores da base $\{q_1, q_2, ..., q_n\}$:

$$\begin{aligned} a_1 &= \langle a_1, q_1\rangle\, q_1 + \langle a_1, q_2\rangle\, q_2 + \ldots + \langle a_1, q_n\rangle\, q_n \\ a_2 &= \langle a_2, q_1\rangle\, q_1 + \langle a_2, q_2\rangle\, q_2 + \ldots + \langle a_2, q_n\rangle\, q_n \\ &\vdots \\ a_n &= \langle a_n, q_1\rangle\, q_1 + \langle a_n, q_2\rangle\, q_2 + \ldots + \langle a_n, q_n\rangle\, q_n \end{aligned}$$

Para que se possa representar este sistema matricialmente, terá que existir uma terceira matriz $R \in \mathbb{R}^{n\times n}$ para conectar as matrizes A e $\mathcal{Q}$. Assim sendo toma-se:

$$\mathcal{R} = \begin{bmatrix} \langle a_1, q_1 \rangle & \langle a_2, q_1 \rangle & \dots & \langle a_n, q_1 \rangle \\ \langle a_1, q_2 \rangle & \langle a_2, q_2 \rangle & \dots & \langle a_n, q_2 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle a_1, q_n \rangle & \langle a_2, q_n \rangle & \cdots & \langle a_n, q_n \rangle \end{bmatrix}$$

Pela multiplicação de matrizes, tem se:

$$\mathcal{QR} = [\mathcal{Q}r_1 \ \mathcal{Q}r_2 \ \dots \ \mathcal{Q}r_n]$$

sendo r_1, r_2, ..., r_n as colunas da matriz $\mathcal{R}$.

Portanto,

$$\mathcal{Q}r_i = \langle a_i, q_1 \rangle \, q_1 + \langle a_i, q_2 \rangle \, q_2 + \ldots + \langle a_i, q_n \rangle \, q_n$$

sendo

$$i = 1, 2, \ldots, n.$$

Assim,

$$\mathcal{Q}r_i = a_i$$

Logo,

$$\mathcal{QR} = A$$

Porém, $\mathcal{R}$ é uma matriz triangular superior, pois do processo de Gram-Schmidt, temos que q_n é ortogonal aos vetores a_1, a_2, ..., $a_{(n-1)}$, se $n > 2$. Logo:

$$\mathcal{R} = \begin{bmatrix} \langle a_1, q_1 \rangle & \langle a_2, q_1 \rangle & \dots & \langle a_n, q_1 \rangle \\ 0 & \langle a_2, q_2 \rangle & \dots & \langle a_n, q_2 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \langle a_n, q_n \rangle \end{bmatrix}$$

Portanto,

$$A = \mathcal{QR},$$

sendo $\mathcal{Q}$ uma matriz $m \times n$ com colunas ortonormais e $\mathcal{R}$ uma matriz $n \times n$ triangular superior.

Mostra-se que $\mathcal{R}$ é inversível.

Seja

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

uma solução do sistema $\mathcal{R}x = 0$

Segue-se que $\mathcal{QR}x = Q_0$, ou seja, $Ax = 0$. Mas, $Ax = x_1a_1 + x_2a_2 + ... + x_na_n$. Então:

$$x_1a_1 + x_2a_2 + ... + x_na_n = 0 \tag{B.1}$$

Como as colunas a_i, $i = 1, ..., n$, são linearmente independentes então

$$x_1 = x_2 = ... = x_n = 0 \tag{B.2}$$

Logo, $x = 0$

Como $x = 0$ é a única solução do sistema homogêneo $\mathcal{R}x = 0$, então, $\mathcal{R}$ é inversível.

Teorema 3

Sejam uma matriz $A \in \mathbb{R}^{n \times n}$, x, $A \in \mathbb{R}^n$ e $Ax = b$ um sistema linear. Então as seguintes afirmações são equivalentes:

- A é inversível ($det(A) \neq 0$).
- $Col(A) = \mathbb{R}^n$.

As colunas de A são linearmente independentes e geram $\mathbb{R}^n$, e portanto formam uma base de $\mathbb{R}^n$.

- $Ax = 0$ tem apenas a solução trivial.
- $Ax = b$ é compatível para qualquer vetor $b \in \mathbb{R}^n$.
- $Ax = b$ tem exatamente uma solução para qualquer vetor $b \in \mathbb{R}^n$.

Assim pode-se concluir que $\mathcal{R}$ é inversível

B.2 Rotação de Givens

Uma rotação de Givens é uma matriz da forma:

$$G_{ij} = \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & c & \cdots & -s & & \\ & & \vdots & \ddots & \vdots & & \\ & & s & \cdots & c & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{bmatrix}$$

onde a linha com $-s$ é a linha j, a linha com s é a linha i e $c^2 + s^2 = 1$. Note que a intersecção das linhas i e j com as colunas i e j de G_{ij} forma

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

Esta matriz é ortogonal, pois

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} * \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} c^2 + s^2 & 0 \\ 0 & c^2 + s^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Pode-se concluir, assim, que as matrizes G_{ij} são ortogonais. Essas matrizes são chamadas de rotações pois, se x e y satisfazem:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

então y é o vetor x girado do ângulo ϕ.

Figura B.1: Rotação de Givens

Assim, multiplicar um vetor por G_{ij} é equivalente a girar duas componentes do vetor do ângulo $\phi = arctan\frac{s}{c}$, deixando as demais componentes intactas.

Seja x um vetor com duas componentes se defini

$$c = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}$$

e

$$s = -\frac{x_2}{\sqrt{x_1^2 + x_2^2}}$$

Observe que $c^2 + s^2 = 1$ e

$$\begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sqrt{x_1^2 + x_2^2} \\ 0 \end{pmatrix}$$

Resultando em um processo interativo que busca o zeramento de elementos da matriz através de multiplicações como visto na figura

Figura B.2: Rotação de Givens para formação de uma matriz triangular superior

B.3 Reflexão de Householder

O algoritmo de Householder para a decomposição QR consiste em construir uma matriz triangular através de uma sequência de matrizes ortonormais. Isto é, a ideia é encontrar matrizes ortonormais H_1, H_2, ..., H_n tais que o produto $H_n...H_2H_1$ seja triangular superior. No k-ésimo passo do algoritmo, a matriz H_k é escolhida de forma que todos os elementos abaixo da diagonal na k-ésima coluna sejam zerados. Ou seja, o que se procura no k-ésimo passo, é uma transformação que zere, em um vetor $x \in \mathbb{R}^{m-k+1}$, todas as entradas com exceção da primeira

Geometricamente, isso pode ser feito rotacionando ou refletindo o vetor x de forma que ele coincida com o eixo e_1.

Um refletor de Householder reflete o espaço $\in \mathbb{R}^{m-k+1}$ através do hiperplano ortogonal ao vetor de Householder $v = \|x\| e_1 - x$, conforme Figura B.4.

Figura B.3: Aplicação de sucessivas reflexões de Householder a uma Matriz

Assim, quando o refletor é aplicado, a imagem de x é mapeada em $\|x\| e_1$. Algebricamente, o refletor de Householder é dado por

$$H = I - 2\frac{vv^T}{v^T v} \tag{B.3}$$

Para uma interpretação geométrica da transformação de Householder, considere qualquer vetor x $M \times 1$ pre multiplicado pela matriz H, demonstrada por:

$$Hx = (I - 2\frac{vv^T}{v^T v})x \tag{B.4}$$

H é ortogonal, uma vez que

$$\begin{aligned} H^T H &= (I - 2\tfrac{vv^T}{v^T v})^T (I - 2\tfrac{vv^T}{v^T v}) \\ &= I - 2 * 2\tfrac{vv^T}{v^T v} + 2\tfrac{vv^T}{v^T v} 2\tfrac{vv^T}{v^T v} \\ &= I - 4(\tfrac{vv^T}{v^T v} - \tfrac{vv^T}{v^T v}) \\ H^T H &= I \end{aligned}$$

Para que os zeros obtidos nas iterações anteriores sejam preservados, cada transformação H_k é aplicada da coluna k em diante. Em outras palavras, suponha que se tenha calculado as transformações de Householder H_1,..., H_{k-1} tais que

Assim para estabilidade numérica, é importante escolher a reflexão que mova o vetor a maior distância.

$$H_{K-1} ... H_1 A = \begin{bmatrix} R_{11} & r_{1k} & R_{1,k+1} \\ 0 & a_{kk} & A_{k,k+1} \end{bmatrix}$$

onde R_{11} é triangular superior. Sejam $v = \|a_{kk}\| e_1 - a_{kk}$ e $\hat{H}_k = I - 2\frac{vv^T}{v^T v}$. Toma-se a transformação linear

Figura B.4: Reflexão de Householder

$$H_K = \begin{bmatrix} I_{K-1} & 0 \\ 0 & \hat{H}_K \end{bmatrix}$$

e logo

$$H_k H_{k-1} ... H_1 A = \begin{bmatrix} R_{11} & r_{1k} & R_{1,k+1} \\ 0 & \hat{H}_k a_{kk} & \hat{H}_k A_{k,k+1} \end{bmatrix} = \begin{bmatrix} R_{11} & r_{1k} & R_{1,k+1} \\ 0 & ||\, a_{kk}|| & r^T_{k\,,k+1} \\ 0 & 0 & A_{k+1,k+1} \end{bmatrix}$$

É fácil ver que o processo tende à triangulação, convergindo para a matriz R. Como o produto de matrizes ortonormais é também ortonormal, toma-se $Q^T = H_n ... H_2 H_1$ e obtém-se o fator Q.

B.3.1 Propriedades

A transformada de Householder definida pela Equação (B.3), segue as seguintes propriedades:

Propriedade 1.

A transformação de Householder H é Hermitiana, isto é,

$$H^H = H \tag{B.5}$$

Propriedade 2.

A transformação de Householder H é unitária, isto é,

$$H^{-1} = H^H \tag{B.6}$$

Propriedade 3.

A transformação de Householder H preserva a norma, isto é,

$$\|Hx\| = \|x\| \tag{B.7}$$

Propriedade 4.

Se dois vetores se submeterem a mesma transformação de Householder, seu produto interno permanece inalterada.

Considere qualquer três vetores: x, y e u. Seja a matriz de Householder H definida em termos do vetor u, como na Equação (B.3). Deixe os restantes dois vetores x e y ser transformado por H, fornecendo Hx e Hy, respectivamente. O produto interno destes dois vetores é transformadas em:

$$\begin{aligned}(Hx)^H(Hy) &= x^H H^H H y \\ &= x^H y\end{aligned} \tag{B.8}$$

Propriedade 5.

Dada a transformação de Householder H, o vetor transformado Hx é uma reflexão de x em relação ao hiperplano perpendicular do vetor u envolvido na definição de H.

Esta propriedade é apenas uma reafirmação da Equação (B.4). Os dois casos limites seguintes da Propriedade 5 são especialmente notáveis:

- x é o vetor escalar múltiplo de u: neste caso, a Equação (B.4) pode ser simplificada em:
$$Hx = -x$$
- o vetor x é ortogonal a u: isto é, o seu produto interno é zero: neste segundo caso, a equação reduz-se a:
$$Hx = x$$

Propriedade 6.

Seja x um vetor qualquer não nulo $M \times 1$ com a norma Euclidiana $\|x\|$. Suponha que o vetor 1 denote a primeira coluna da matriz identidade; isto é,

$$1 = [1, 0, ..., 0]^T \tag{B.9}$$

Então, existe uma transformação Householder H definida pelo vetor:

$$u = x - \|x\| 1 \tag{B.10}$$

de tal modo que o vetor transformado Hx correspondente a u, é um múltiplo linear do vetor 1.

Com o vetor u atribuído pelo valor da Equação (B.10), tem-se

$$\|u\|^2 = 2 \|x\| (\|x\| - x_1) \tag{B.11}$$

em que x_1 é o primeiro elemento do vetor x. Da mesma forma, pode-se escrever:

$$u^H x = \|x\| (\|x\| - x_1) \tag{B.12}$$

Assim, substituindo as Equações (B.11) e (B.12) na Equação (B.4), descobri-se que o vetor transformado Hx correspondente à definição de vetor u da Equação (B.10) é dada por:

$$\begin{aligned} Hx &= x - u \\ &= x - (x - \|x\| 1) \\ &= \|x\| 1 \end{aligned}$$

que prova a Propriedade **6**.

Apêndice C

Metodos de Otimização do Algoritmo

C.1 Back Substitution

O algoritmo similar para sistemas triangulares superiores $Ux = B$ é chamado de substituição para trás (*back-substitution*). O procedimento para encontrar x é descrito por

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) / u_{ii}$$

onde b_i pode ser substituído por x_i.

Referências Bibliográficas

Anderson, B.D.O. and 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.

Arora, Sanjeev and Boaz Barak (2006). *Computational Complexity: A Modern Approach*. Princeton University.

Astrom, Karl Johan and Bjorn Wittenmark (1994). *Adaptive Control*. 2nd ed.. Addison-Wesley Longman Publishing Co., Inc.. Boston, MA, USA.

Barto, A.G., R.S. Sutton and 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.

Bellman, R.E. (2003). *Dynamic Programming*. Dover Books on Computer Science Series. Dover Publications.

Bellman, R.E. and R.E. Kalaba (1965). *Dynamic Programming and Modern Control Theory*. Academic paperbacks. Academic Press.

Bertsekas, D.P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*. Vol. 2. Prentice-Hall.

Bertsekas, D.P. (1995). *Dynamic Programming and Optimal Control*. Vol. 1. Athena Scientific, Belmont, Massachusetts.

Bertsekas, D.P. and J.N. Tsitsiklis (1996). *Neuro-Dynamic Programming*. Athena Scientific. Belmont, MA.

Bertsekas, D.P., M.L. Homer, D.A. Logan, S.D. Patek and Nils R. Sandell (2000). Missile defense and interceptor allocation by neuro-dynamic programming. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* **30**(1), 42–51.

Boldea, I. (2006). *Variable Speed Generators*. 1 ed.. Crc Press.

Bradtke, S.J., B.E. Ydstie and A.G. Barto (1994). Adaptive linear quadratic control using policy iteration. In: *Computer Science Department University of Massachusetts*. Amherst, MA. pp. 3475–3479.

Bryson, Arthur E. and Yu-Chio Ho (1975). *Applied Optimal Control*. Taylor and Francis. UK.

Buşoniu, L., D. Ernst, B. De Schutter and R. Babuška (2011). Approximate reinforcement learning: An overview. In: *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2011)*. pp. 1–8.

Church, Alonzo (1996). *Introduction to Mathematical Logic*. Princeton Landmarks in Mathematics.

de Campos, Marcello L. R. and Gilbert Strang (2009). *QR Decomposition: An Annotated Bibliography*. J.A. Apolinario Jr. (ed.), ´ QRD-RLS Adaptive Filtering,.

Dierks, T. and 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.

Downey, Rod and Michael Fellows (1999). *Parameterized complexity*. Springer.

Du, Ding-Zhu and Ker-I Ko (2000). *Theory of Computational Complexity*. John Wiley and Sons.

Falb, Peter L. (2006). *Optimal Control: An Introduction to the Theory and Its Applications*. Dover Publications.

Ferrari, S. and R.F. Stengel (2004). Online adaptive critic flight control. *J. Guid. Control Dyn.* **27**(5), 777–786.

Ferrari, S., J. E. Steck and R. Chandramohan (2008). Adaptive feedback control by constrained approximate dynamic programming. *Trans. Sys. Man Cyber. Part B* **38**(4), 982–987.

Fonseca Neto, J.V., I.S. Abreu and F.N. da Silva (2010). Neural-genetic synthesis for state-space controllers based on linear quadratic regulator design for eigenstructure assignment. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **40**(2), 266–285.

Franklin, Gene F., J. Da Powel and Abbas Emami-Naeini (2014). *Feedback Control of Dynamic Systems*. Prentice Hall.

Friedland, Bernard (2005). *Control System Design: An Introduction to State-Space Methods*. Dover Publications.

Gentleman, W. M. (1973). Least squares computations by givens transformations without square roots. *IMA Journal of Applied Mathematics*.

Gentleman, W. M. and H. T. Hung (1981). Matrix triangularization by systolic arrays. *SPIE Real-Time Signal Processing IV*.

Goldreich, Oded (2008). *Computational Complexity: A Conceptual Perspective*. Cambridge University Press.

Golub, G. H. (1965). Numerical methods for solving linear least squares problems.. *Numerische*.

Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations*. 3nd ed.. Johns Hopkins University Press.

Golub, G. H. e Van Loan, C. F. (1989). *Matrix Computations*. 2nd ed.. The Johns Hopkins University Press.

Golub, Gene H. and Charles F. Van Loan (1996). *Matrix Computations*. The Johns Hopkins University Press.

Haykin, Simon (1995). *Adaptive Filter Theory*. 3rd ed.. Prentice-Hall, Inc.. Upper Saddle River, NJ, USA.

Haykin, Simon (1997). *Adaptive Filter Theory*. Prentice Hall.

Haykin, Simon (2008). *Redes Neurais Princípios e prática*. bookman.

Horn, Roger A. and Charles R. Johson (1987). *Matrix Analysis*. Cambridge University Press.

Howard, R.A. (1960). *Dynamic Programming and Markov Processes*. John Wiley and Sons, Inc.. New York.

Khan, Said G., Guido Herrmann, Frank L. Lewis, Tony Pipe and Chris Melhuish (2012). Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control* **36**(1), 42–59.

Khan, S.G. and F.L. Lewis (2012). Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control*.

Kirk, Donald E. (2004). *Optimal Control Theory An Introduction*. Dover Publications.

Landelius, T. and H. Knutsson (1996). Greedy adaptive critics for lqr problems: Convergence proofs. Department of Electrical Engineering, Computer Vision Laboratoty. Linkoping University, 581 83 Linkoping, Sweden.

Landelius, Tomas (1997). Reinforcement Learning and Distributed Local Model Synthesis. PhD thesis. Linkoping University.

Lewis, F. L. and D. Liu (1992). *Aircraft control and simulation*. Wiley.

Lewis, F. L. and D. Vrabie (2009*a*). Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits and Systems Magazine, IEEE* **9**(3), 32–50.

Lewis, F.L. and D. Vrabie (2009*b*). Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits and Systems Magazine, IEEE* **9**(3), 32–50.

Lewis, Frank L. and Derong Liu (2012). *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. John Wiley and Sons, Inc.

Lewis, Frank L. and Draguna Vrabie (2009*c*). Adaptive dynamic programming for feedback control. In: *Asian Control Conference, 2009. ASCC 2009. 7th*. pp. 1402 –1409.

Lewis, Frank L., Draguna L. Vrable and Vassilis L. Syrmos (2012). *Optimal Control*. 3rd ed.. Wiley.

Lin, Chuan-Kai (2005). Adaptive critic autopilot design of bank-to-turn missiles using fuzzy basis function networks. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **35**(2), 197–207.

Liu, Xin and 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.

McWhirter, J. G. (1983*a*). Recursive least-squares minimization using a systolic array.. *Real-Time Signal Processing VI, SPIE Proceedings,*.

McWhirter, J. G. (1983*b*). Systolic array for recursive least-squares minimisation. *Electronics Letters*.

Murray, J.J., C.J. Cox, G.G. Lendaris and R. Saeks (2002). Adaptive dynamic programming. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* **32**(2), 140 – 153.

Papadimitriou, Christos H. (1994). *Computational Complexity*. Addison Wesley Longman.

Pavlov, I. P. (1927). Conditional reflexes: An investigation of the physiological activity of thecerebral cortex. *Oxford University Press*.

Pinto, Vandilberto Pereira (2007). Modelagem e Simulação de uma Planta Elétrica Controlada por um Regulador Linear Quadratico Conectada ao Sistema de Distribuição de Energia Elétrica. PhD thesis. Universidade Federal do Ceará.

Qingyan, W. Shuyan; W. Renbiao; S. (2008). Recursive least squares constant modulus algorithm based on the qr decomposition. *IEEE Int. Conference Neural Networks and Signal Processing*.

Quirion, J.P., E. Gunn and J. Gu (2004). Optimal control of permanent magnet motors using dynamic programming. In: *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*. Vol. 1. pp. 364 – 369 vol.1.

Robert, K. and S. George (2001). Computational aspects of performance-adaptive self-organizing control algorithms. *CDC*.

Sayed, A. H. e Kailath T. (1994). A state-space approach to adaptive rls filtering. *IEEE Signal Process Magazine* **II**, 18–60.

Shoaib, Jose A. Apolinario Jr; Mobien (2006). Solution to the weight extraction problem in fast qr-decomposition rls algorithms. *ICASSP 2006*.

Sipser, Michael (2006). *Introduction to the Theory of Computation*. Thomson Course Technology.

Stewart, G. W. (1973). *Introduction to Matrix Computations*. Academic Press.

Stingu, P. and 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.

Sutton, R.S. (1988). Learning to predict by the method of temporal diferences. In: *Machine Learning*. pp. 9–44.

Sutton, R.S., A.G. Barto and R.J. Williams (1992). Reinforcement learning is direct adaptive optimal control. *Control Systems, IEEE* **12**(2), 19–22.

Sutton, R.S. and A.G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press. Cambridge, MA.

Viana da Fonseca Neto, J., E.F.M. Ferreira and P.H.M. Rêgo (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*. pp. 3174–3179.

Wang, F. Y., H. Zhang and D. Liu (2009*a*). Adaptive dynamic programming: An introduction. *IEEE Computer Intelligence Magazine* **4**(2), 39–47.

Wang, Lin, Hui Peng, Hua-yong Zhu and Lin-Cheng Shen (2009*b*). A survey of approximate dynamic programming. In: *Proceedings of the 2009 International Conference on Intelligent Human-Machine Systems and Cybernetics*. Vol. 2 of *IHMSC '09*. IEEE Computer Society. Washington, DC, USA. pp. 396–399.

Ward, C. R. (1986). A novel algorithm and architecture for adaptive digital beamforming.. *IEEE Transactions on Antennas and Propagation*.

Watkins, C.J.C.H. (1989). learning from delayed rewards. PhD thesis. University of Cambridge. Cambridge, England.

Weber, C., Elshaw M. and N. M Mayer (2008). *Reinforcement Learning, Theory and Applications.*. I-TECH Education and Publishing.

Werbos, P.J. (1968). The elements of intelligence. *Cybernetica*.

Werbos, P.J. (1990). Consistency of hdp applied to a simple reinforcement learning problem. *Neural Netw.* **3**(2), 179–189.

Widrow, B., N. Gupta and S. Miatra (1973). Punish/reward: Learning with a critic in adaptive threshold systems. *IEEE Trans. Syst., Man, Cybern.*

Xu, X., H. He and D. Hu (2002). Efficient reinforcement learning using recursive least-squares methods. In: *Department of Automatic Control. National University of Defense Technology*. Vol. 16. ChangSha, Hunan, 410073, P.R.China. pp. 259–292.