UNIVERSIDADE FEDERAL DO MARANHÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Márcio Eduardo Gonçalves Silva

Algoritmos da Família LMS para a Solução Aproximada da HJB em Projetos Online de Controle Ótimo Discreto Multivariável e Aprendizado por Reforço

São Luís-MA
2014

UNIVERSIDADE FEDERAL DO MARANHÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Márcio Eduardo Gonçalves Silva

Algoritmos da Família LMS para a Solução Aproximada da HJB em Projetos Online de Controle Ótimo Discreto Multivariável e Aprendizado por Reforço

Dissertação submetida à Universidade Federal do Maranhão como parte dos requisitos para a obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: João Viana da Fonseca Neto
Coorientador: Francisco das Chagas de Souza

São Luís-MA
2014

Silva, Márcio Eduardo Gonçalves

Algoritmos da família LMS para a solução aproximada da HJB em projetos online de controle ótimo discreto multivariável e aprendizado por reforço / Márcio Eduardo Gonçalves Silva. – São Luís, 2014.

168 f.

Orientador: João Viana da Fonseca Neto

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Maranhão, 2014.

1. Programação Dinâmica 2. Aprendizagem por Reforço 3. Regulador Linear quadrático 4. Crítico adaptativo I. Título

CDU 004.414.23

ALGORITMOS DA FAMÍLIA LMS PARA A SOLUÇÃO APROXIMADA DA HJB E PROJETO ONLINE DE CONTROLE ÓTIMO DISCRETO MULTIVARIÁVEL E APRENDIZADO POR REFORÇO

Marcio Eduardo Gonçalves Silva

Dissertação aprovada em 21 de agosto de 2014.

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

Prof. Francisco das Chagas de Souza, Dr.
(Co-orientador)

Prof. Vandilberto Pereira Pinto, Dr.
(Membro da Banca Examinadora)

Prof. Ewaldo Eder Carvalho Santana, Dr.
(Membro da Banca Examinadora)

Agradecimentos

Ao Prof. Dr. João Viana da Fonseca Neto, meu orientador, pela paciência, conhecimentos repassados, confiança e amizade durante todo o desenvolvimento do trabalho.

Ao Prof. Dr. Francisco das Chagas de Souza, meu coorientador, pelas revisões, sugestões, conhecimentos repassados e amizade formada.

A Universidade Federal do Maranhão, que me proporcionou a oportunidade de cursar Pós-graduação em curso de excelência.

Aos professores e funcionários do Departamento de Pós-graduação em Engenharia Elétrica, pelo apoio e colaboração.

Aos amigos de estudo e irmãs na amizade que fizeram e fazem parte da minha formação e com certeza vão continuar presentes em minha vida.

Aos meus irmãos Tânia, José Raimundo, Sandra, Vânia, Sebastião, Valmary e Luís Fernando, por serem irmãos e amigos por toda vida.

A Joselene Gomes Silva, minha esposa, pelo grande apoio e incentivo dado durante os momentos mais difíceis.

Aos meus pais José Ribamar da Silva e Raimunda Gonçalves Silva, pela alegria que sentem ao me ver realizar um objetivo que certamente é deles também.

Aos meus filhos Márcio Filho e Márlon, que são as recompensas da minha vida.

A Deus, por ter me dado uma família escolhida a dedo, maravilhosa e feliz.

"Errei mais de 9.000 cestas e perdi quase 300 jogos. Em 26 diferentes finais de partidas fui encarregado de jogar a bola que venceria o jogo... e falhei. Eu tenho uma história repleta de falhas e fracassos em minha vida. E é exatamente por isso que sou um sucesso."

Michael Jordan

Resumo

A técnica de controle linear baseado na minimização de um índices de desempenho quadrático utilizando o segundo método de Liapunov garante a estabilidade do sistema, se este for controlável e observável. Por outro lado, nessa técnica inexoravelmente é necessário encontrar a solução da Equação Hamilton-Jacobi-Bellman (HJB) ou Riccati. Em projeto de sistema de controle *online* que necessita, em tempo real, alterar seus ganhos de retroação para manter uma certa dinâmica, impõe o cálculo da solução da equação de Riccati em cada instante de amostragem gerando uma grande carga computacional que pode inviabilizar sua implementação. Neste trabalho, mostra-se o projeto de um sistema de controle inteligente que encontra a ação de controle ótima ou subótima a partir de dados sensoriais dos estados do processo e do custo instantâneo observados após cada transição de estado. Para encontrar essa ação de controle ou política ótima, a programação dinâmica aproximada ou críticos adaptativos são utilizados, tendo como base as parametrizações dado pelo problema do regulador linear quadrático (LQR), mas sem resolver explicitamente a equação de Riccati associada. Mais especificamente, o problema do LQR é resolvido por quatro métodos distintos que são os algoritmos de Programação Dinâmica Heurística, a Programação Dinâmica Heurística Dual, a Programação Dinâmica Heurística Dependente de Ação e a Programação Dinâmica Heurística Dual Dependente de Ação. Entretanto, esses algoritmos dependem do conhecimento das funções valor para, assim, derivar as ações de controle ótimas. Essas funções valor com estruturas conhecidas tem seus parâmetros estimados utilizando os algoritmos da família dos mínimos quadrados médios e o algoritmo de Mínimos Quadrados Recursivo. Dois processos que obedecem à propriedade de Markov foram empregados na validação computacional dos algoritmos críticos adaptativos, um corresponde à dinâmica longitudinal de uma aeronave e o outro à de um circuito elétrico.

Palavras-Chave: Programação Dinâmica, Aprendizagem por Reforço, Regulador Linear Quadrático, Crítico Adaptativo.

Abstract

The technique of linear control based on the minimization of a quadratic performance index using the second method of Lyapunov to guarantee the stability of the system, if this is controllable and observable. however, this technique is inevitably necessary to find the solution of the HJB or Riccati equation. The control system design online need, real time, to adjust your feedback gain to maintain a certain dynamic, it requires the calculation of the Riccati equation solution in each sampling generating a large computational load that can derail its implementation. This work shows an intelligent control system design that meets the optimal or suboptimal control action from the sensory data of process states and the instantaneous cost observed after each state transition. To find this optimal control action or policy, the approximate dynamic programming and adaptive critics are used, based on the parameterizations given by the problem of linear quadratic regulator (LQR), but without explicitly solving the associated Riccati equation. More specifically, the LQR problem is solved by four different methods which are the Dynamic Programming Heuristic, the Dual Heuristic Dynamic Programming, Action Dependent Dynamic Programming Heuristic and Action Dependent Dual Heuristic Dynamic Programming algorithms. However, these algorithms depend on knowledge of the value functions to derive the optimal control actions. These value functions with known structures have their parameters estimated using the least mean square family and Recursive Least Squares algorithms. Two processes that have the Markov property were used in the computational validation of the algorithms adaptive critics implemented, one corresponds to the longitudinal dynamics of an aircraft and the other to an electrical circuit.

Keywords

Dynamic Programming, Reinforcement Learning, Linear Quadratic Regulator, Adaptive critic.

Lista de Figuras

Lista de Tabelas

Abreviaturas

ADDHP	Action Dependent Dual Heuristic Programming (Programação Dinâmica Heurística Dual Dependente de Ação)
ADGDHP	Action Dependent Global Dual Heuristic Programming (Programação Heurística Dual Global dependente de ação)
ADHDP	Action Dependent Heuristic Dynamic Programming (Programação Dinâmica Heurística Dependente de Ação)
AR	Aprendizagem por Reforço
CA	Críticos Adaptativos
DARE	Discrete Algebraic Riccati Equation (Equação Algébrica de Ricatti Discreta)
DHP	Dual Heuristic Programming (Programação Dinâmica Heurística Dual)
DLQR	Discret Linear Quadratic Regulator (Regulador Linear Quadrático Discreto)
GDHP	Global Dual Heuristic Programming (Programação Heurística Dual Global)
HDP	Heuristic Dynamic Programming (Programação Dinâmica Heurística)
HJB	Equação de Hamilton-Jacobi-Bellman
IAFPNLMS	Individual-Activation-Factor Proportionate Normalized Least-Mean-Square (Mínimo Quadrado Médio Normalizado Proporcional com Fator de Ativação Individual)
IP	Iteração de Política
IPNLMS	Improved Proportional Normalized Least Mean Squares (Mínimo Quadrado Médio Normalizado Proporcional Melhorado)
IV	Iteração de Valor
LMS	Least Mean Squares (Mínimo Quadrado Médio)
LQR	Linear Quadratic Regulator (Regulador Linear Quadrático)
LTI	Linear Time-Invariant (Linear e Invariante no Tempo)
LTV	Linear Time-Varying (Linear e Variante no Tempo)
MIMO	Multiple-Input Multiple-Output (Múltiplas Entradas e Múltiplas Saídas)

NLMS	Normalized Least Mean Squares (Mínimo Quadrado Médio Normalizado)
PD	Propriedade de Markov
PDA	Programação Dinâmica Adaptativa ou Programação Dinâmica Aproximada
PDM	Processo de Decisão Markoviano
PNLMS	Proportional Normalized Least Mean Squares (Mínimo Quadrado Médio Normalizado Proporcional)
RLS	Recursive Least Squares (Mínimos Quadráticos Recursivo)

Sumário

1

INTRODUÇÃO

Sumário

A sociedade moderna está extremamente ligada aos sistemas de controle automático, mas quase sempre estes são transparentes às pessoas. A complexidade dos sistemas atuais e sem sombra de dúvidas ainda mais no futuro, exigência da própria necessidade humana, traz a necessidade imperiosa de recorrer a novas técnicas de controle que sejam capazes de tratar toda essa complexidade. Como o advento dos computadores digitais e algoritmos poderosos foi possível controlar sistemas complexos no domínio do tempo baseados na análise de variáveis de estado (KIRK, 1970).

Inicialmente, o que se desejava de um sistema de controle era que ele satisfizesse determinados requisitos de desempenho, tais como margem de fase, margem de ganho, máxima ultrapassagem, tempo de acomodação, entre outros. Para esses tipos de requisitos de desempenho, as técnicas de resposta em frequência e do lugar das raízes, o chamado controle clássico, foram e são bastante úteis, mas são limitados a projeto de sistema de controle a malhas independentes. No final dos anos cinquenta, os projetos de controle se deslocaram da mera necessidade de satisfazer requisitos de desempenho para projetos de sistemas ótimos em relação a algum critério de desempenho (OGATA, 2002). No entanto, o projeto de controladores ótimos requer a solução da equação de Hamilton-Jacobi-Bellman (HJB) para sistemas não lineares e a particular equação de Riccati para sistemas lineares que são difíceis de serem resolvidas *online* por demandar um custo computacional elevado.

Além disso, projetos de controladores ótimos mantém um comportamento fixo que é excelente para sistemas lineares invariantes no tempo. Contudo, controladores clássico e ótimos podem ter seu desempenho degradando ou perder a otimalidade em processos de dinâmica variante ou não lineares, pois exigem um modelo fiel do sistema.

Assim, foi preciso desenvolver outros métodos que garantissem os requisitos de desempenho em processo com mudança de contexto, os sistemas não-lineares e Linear e Variante no Tempo (*Linear Time-Varying*-LTV), por exemplo. O caminho seguido foi projetar um controlador que tenha capacidade de reconhecimento *online* das características do processo ou do ambiente e que possa se adequar e efetuar um controle de acordo com os requisitos prescrito. Estão dentro dessa abordagem o controle por particionamento, o controle adaptativo e o controle com aprendizado (LENDARIS, 2009).

Controles adaptativos são projetados para controlar processos desconhecidos, em tempo real, e garantir um desempenho satisfatório apenas usando dados medidos ao longo da trajetória, mas geralmente não é exigido desempenho ótimo em relação a uma função prescrita pelo projetista (LEWIS; VRABIE; VAMVOUDAKIS, 2012).

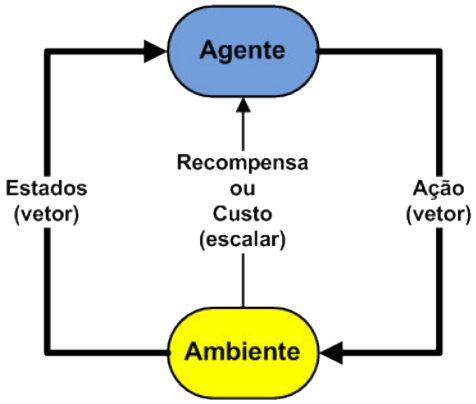
Entre os métodos utilizados para resolver os casos mais geral de controle ótimo está a Programação Dinâmica (PD), proposto Richard Bellman (BELLMAN, 2003) (BERTSEKAS, 1995). O método de programação dinâmica faz pouca suposição sobre o processo que pode ser determinístico, estocástico, linear ou não-linear, diferentemente de técnicas de controle automático clássicas que geralmente impõem limitações ao processo (BUSONIU et al., 2010). O método de Bellman consegue resolver uma classe grande de problema

de otimização não-linear e consequentemente a equação HJB (LENDARIS, 2009). No entanto, a solução de problemas através da programação dinâmica se torna difícil e até mesmo impossível em ambientes com espaço de estado contínuo ou grande, além disso, essa técnica requer um modelo completo do sistema.

Por vezes, um modelo matemático fiel estimado do comportamento de um sistema é difícil de ser obtido, ou por ser caro para se obter conjunto de exemplos de pares de dados de entrada e saída ou por se ter uma fraca compreensão do seu comportamento. Em cenários como esse, pode-se utilizar uma forma de aprendizagem não-supervisionada conhecida como Aprendizagem por Reforço (AR). Esta é uma abordagem que permite ao agente aprender através de suas interações com o ambiente para alcançar certos objetivos (SUTTON; BARTO, 1998).

O que a aprendizagem por reforço faz é apreender uma aplicação ou mapeamento dos estados em ações que maximizar ou minimizar uma função de recompensa ou de custo. Esse mapeamento é encontrado sem o conhecimento antecipado das ações a tomar. A AR consegue encontrar esse mapeamento (política) interagindo com o ambiente e observando as mudanças ocorrida nele (estados) e recebendo uma informação da qualidade dessas mudanças (recompensa ou custo), conforme mostra a figura (1.1).

Figura 1.1: Aprendizagem por Reforço - Ação, Percepção e Recompensa



A aprendizagem por Reforço e a Programação Dinâmica são fundida para gerar o conceito de Crítico Adaptativo (CA). A diferença entre PD e CA reside essencialmente no fato de que esta utiliza uma aproximação da função valor ótima para realizar seu projeto de controlador e age para frente no tempo, enquanto aquela, utiliza a função valor ótima e age para trás no tempo (LENDARIS, 2009). Werbos em (LEWIS; LIU, 2013) utiliza o termo RLADP para unir as várias vertente de pesquisa e tecnologia que se sobrepõem, como críticos adaptativos, programação dinâmica adaptativa (PDA), programação dinâmica aproximada (PDA) e aprendizado por reforço.

Em 1990 e 1994, Werbos (WERBOS, 1990)(WERBOS, 1994) propôs quatro estrutura de PDA que têm sido muito utilizadas pela comunidade cientifica e tecnológica. Tais estruturas de PDA diferem quanto ao tratamento que cada uma dá à função valor e ao

tipo de função valor usadas para encontrar a política ótima. São dois tipos de função valor, as dependente de estado, conhecidas como função valor estado ou função-V, e as dependentes do par estado-ação, conhecidas como função valor ação ou função-Q. As estruturas de PDA propostas por Werbos são: a Programação Dinâmica Heurística (Heuristic Dynamic Programming-HDP), a Programação Dinâmica Heurística Dual (Dual Heuristic Programming-DHP), a Programação Dinâmica Heurística Dependente de Ação (Action Dependent Heuristic Dynamic Programming-ADHDP) e a Programação Dinâmica Heurística Dual Dependente de Ação (Action Dependent Dual Heuristic Programming-ADDHP). Prokhorov e Wunsch (PROKHOROV; WUNSCH, 1997) incluíram mais duas estruturas chamadas de Programação Heurística Dual Global (Global Dual Heuristic Programming-GDHP) e Programação Heurística Dual Global dependente de ação (Action Dependent Global Dual Heuristic Programming-ADGDHP).

Neste trabalho, mostra-se o projeto de um sistema de controle que encontra a ação de controle ótima a partir de dados sensoriais dos estados do processo e do custo instantâneo de transição de estado. Para isso, utiliza-se a programação dinâmica e aprendizagem por reforço justamente com o algoritmo de iteração de política para encontrar uma política ótima para o problema do regulador linear quadrático discreto sem resolver explicitamente a equação de Riccati. Para tanto, utilizar-se-á os quatros métodos proposto por Werbos e os algoritmos dos mínimos quadrados recursivo e os baseados no mínimo quadrado médio (baseados-LMS) para estimar os parâmetros da função valor.

1.1 Objetivo Gerais

Pesquisar as teorias da Programação Dinâmica (PD) e Aprendizagem por Reforço (AR) aplicada a sistemas de controle dinâmico multivariáveis. Desenvolver algoritmos que combinam PD, AR, algoritmos de estimação paramétrica como o algoritmo dos mínimos quadrados recursivo (*Recursive Least Squares*-RLS) e algoritmos baseados no mínimo quadrado médio (*Least Mean Squares*-LMS) em problemas Regulador Linear Quadrático (*Linear Quadratic Regulator*-LQR), encontrado assim a solução da equação HJB-DARE sem conhecimento do modelo do sistema dinâmico.

1.1.1 Objetivo Específicos

- Pesquisar os algoritmos de Programação Dinâmica Heurística, Programação Dinâmica Heurística Dual, Programação Dinâmica Heurística Dependente de Ação e Dinâmica Heurística Dual Dependente de Ação aplicada a teoria de controle;
- Pesquisar os algoritmos RLS e os Algoritmos da família LMS;
- Mostrar a viabilidade no uso de algoritmos de baixo custo computacional baseados

na família LMS na estimação de parâmetros em sistema em PD e AR;

- Encontrar a solução da HJB sem conhecimento do modelo do processo, mas recebendo informações dos estados e custos das ações tomadas;

- Aplicar em ambiente computacional os algoritmos HDP, DHP, ADHDP e ADDHP particularizados para problema do Regulador Linear Quadrático em ambientes Linear e Invariante no Tempo (*Linear Time-Invariant*-LTI) e LTV e usando o RLS, LMS, Mínimo Quadrado Médio Normalizado (*Normalized Least Mean Squares*-NLMS), Mínimo Quadrado Médio Normalizado Proporcional (*Proportional Normalized Least Mean Squares*-PNLMS), Mínimo Quadrado Médio Normalizado Proporcional Melhorado (*Improved Proportional Normalized Least Mean Squares*-IPNLMS) e Mínimo Quadrado Médio Normalizado Proporcional com Fator de Ativação Individual (*Individual-Activation-Factor Proportionate Normalized Least-Mean-Square*-IAFPNLMS) como estimadores paramétricos;

1.2 Justificativa

Considerando a existência de sistema dinâmico de difícil modelagem ou operando em vários pontos diferente sendo necessária sua linearização e projeto de controle em cada um desses pontos e, além disso, ainda se deseja que o controle seja ótimo em relação a uma função de custo, é que escolheu-se utilizar técnicas de inteligência artificial que tire do projetista a árdua tarefa de prever em tempo de projeto cada possibilidade de controle de um processo.

Sistemas modernos estão cada vez mais complexos exigindo métodos mais rigorosos quanto a garantia de desempenho e satisfação dos objetivos prescritos. E, também, o avanço da tecnologia de fabricação de microprocessadores e de memória ocasionou um grande aumento da capacidade de cálculo e de memória juntamente com miniaturização da partilha e custo relativamente baixo. Esses fatos impulsionaram o estudo e implementação de algoritmos e técnicas de controle altamente sofisticados que hoje podem ser utilizados em sistemas embarcados.

Mesmo com o aumento da capacidade computacional e diminuição de custo dos processadores ainda se pode ter problemas em garantir os requisitos temporais de um ambiente. Sistemas de controle que não atuam em prazo específico são inúteis e perigosos. Geralmente sistemas de controle inteligente impõe altas cargas computacionais para sua execução, no caso específico da PD e AR em ambiente com grandes espaços de estado e ações é comum ver algoritmos altamente sofisticados e caros computacionalmente no processo de avaliação da política ou aproximação da função valor. Esses algoritmos utilizados em sistemas embarcados provavelmente tornará o produto final com custo-benefício elevado devido a utilização de processadores mais caros podendo tornar o produto um fracasso.

Para certos ambientes, pode-se diminuir o custo computacional utilizando algoritmos de baixa carga computacional no processo de estimação dos parâmetros da função valor. Isso nos motivou a investigar o usar dos algoritmos da família LMS para estimar os parâmetros da função valor e verificar suas viabilidades em problemas em que o valor desejado ou alvo não é dado por um professor.

Dentre centenas de algoritmos derivados do LMS, optou-se por utilizar os algoritmos indicados acima por serem os mais comuns e que trouxeram inovações significativas. Por exemplo, o PNLMS foi o primeiro a incluir um fator proporcional aos parâmetros ativos, o IPNLMS traz características do PNLMS e NLMS e o IAF-PNLMS explorar eficientemente a característica esparsa do sistema com a introdução de um fator individual na atualização dos parâmetros.

1.3 Contribuição

Pode-se destacar as seguintes contribuições que este trabalho pretende dar às aplicações e implementações da teoria da aprendizagem por reforço e programação dinâmica.

- Projeto e implementação de algoritmos que utilizam os métodos HDP, DHP, ADHDP e ADDHP para obtenção *online* da ação de controle ótima ou sub-ótima de sistemas com Múltiplas Entradas e Múltiplas Saídas (Multiple-Input Multiple-Output-MIMO);
- Avaliação do desempenho dos algoritmo RLS e baseado-LMS como estimadores paramétricos em sistemas em que o valor desejado ou alvo não é dado por um professor;
- Mostrar a viabilidade de utilização de algoritmo de baixo custo computacional e complexidade de hardware em problemas de aprendizagem por reforço;

1.4 Organização

Este trabalho está organizado da seguinte forma:

O capítulo 2 introduz os conceitos básicos da aprendizagem por reforço e programação dinâmica e uma visão geral da programação dinâmica aproximada ou crítico adaptativo com as estruturas HDP, DHP, ADHDP e ADDHP. No capítulo 3, faz-se um estudo dos algoritmos utilizados para estimação dos parâmetros das funções valor. O capítulo 4 apresenta uma análise dos algoritmos HDP, DHP, ADHDP e ADDHP particularizados para o problema do Regulador Linear Quadrático Discreto (*Discrete Linear Quadratic Regulator*-DLQR) com função valor aproximada por uma estrutura paramétrica e utilizando os algoritmos RLS, LMS, PNLMS, IPNLMS e IAFPNLMS como estimadores paramétricos. No capítulo 5, a validação computacional das estruturas proposta é mostrada, além de

uma breve descrição dos ambientes de controle: um modelo linear de terceira ordem que corresponde a dinâmica longitudinal de uma aeronave e um modelo de quarta ordem linear que descreve um circuito elétrico. Por fim, o capítulo 6 mostra a conclusão do trabalho que inclui comentários sobre os resultados obtidos e melhorias que podem ser aplicadas em trabalhos futuros.

2

Programação Dinâmica e Aprendizagem por Reforço

Sumário

Neste capítulo, mostra-se os métodos de aprendizagem por reforço e programação dinâmica para solução ótima de Processo de Decisão Markoviano utilizando os algoritmos de iteração de política e iteração de valor. Inicialmente, faz-se um resumo dos pontos mais importante da teoria de Aprendizagem por Reforço. Em seguida, tratar-se da Programação Dinâmica e sua duas características principais: o Processo de Decisão Markoviano e função valor incluído a formulação da equação de otimização de Bellman. Por último, discorre-se sobre dois métodos utilizados para resolver problemas de Programação dinâmica na busca de políticas ótimas que são algoritmo de iteração de política e algoritmo de iteração de valor.

2.1 Aprendizagem por Reforço (AR)

Todos os sistemas de controle são projetados para interferir no comportamento da dinâmica de um processo e muitas vez é desejável minimizar ou maximizar alguma função de desempenho. Solução para esse tipo de problema é encontrada na área de otimização matemática.

A programação dinâmica introduzida por Bellman na década de 40 é uma ferramenta importante para resolver problema de controle ótima quando o modelo do sistema está disponível. Porém, muitas vezes o modelo do sistema é desconhecido, complexo ou caro demais para ser identificado, nesses casos uma abordagem alternativa foi introduzida a partir da ideias da aprendizagem por reforço. A AR resolve esse problema interagindo com o ambiente sem que este lhe forneça dados rotulados, mas apenas recebendo amostra de transição de estado coletada offline ou online por interação com o sistema.

A aprendizagem por reforço é definida por caracterizar um problema de aprendizagem e não um métodos de aprendizagem (SUTTON; BARTO, 1998). O problema de aprendizagem por reforço é geralmente formulado em termo de controle ótimo de Processos de Decisão de Markov (PDM) de tempo discreto.

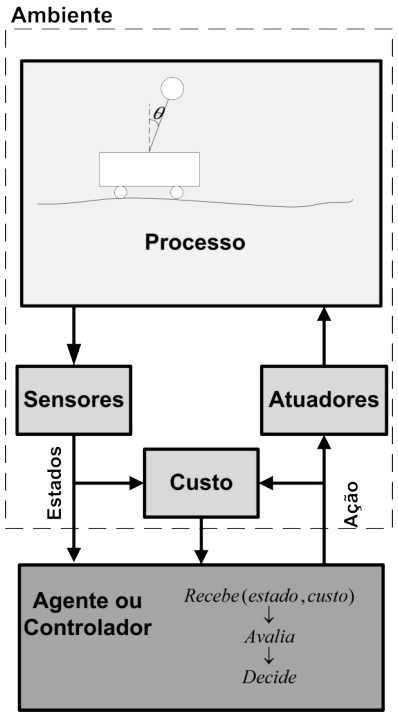
Aprendizagem por Reforço é um tipo de aprendizagem não supervisionada que é realizada pela interação entre agente (controlador) e seu ambiente (processo) para alcança um objetivo específico. O agente ou controlador é o que gera a política de controle, é matematicamente descrito pela política, e o ambiente ou planta é o que sofre as ações do agente. Tudo que está fora do agente é ambiente (LEWIS; VRABIE, 2009)(SUTTON; BARTO, 1998)(BUSONIU et al., 2010).

Todos os agentes de aprendizagem por reforço são baseados em objetivos explícitos, pode captar aspectos de seus ambientes e pode escolher ações para influenciar seu ambiente (SZEPESVARI, 2010).

O processo de interação entre agente e ambiente está ilustrado na figura (2.1) e é descrito da seguinte forma: considerando que o sistema encontra-se inicialmente no estado x_k, o agente seleciona uma ação possível para esse estado μ_k e atua no processo,

consequentemente, ocorre uma mudança de estado. O novo estado x_{k+1} e um custo associado à transição de estado r_{k+1} são retornados ao agente. Este avalia os dados recebidos e decide por nova ação, esse ciclo é repetido até atingir um certo objetivo. O custo é uma medida da qualidade das ações tomadas num determinado estado e pode ser percebido pelo agente em frequências variadas dependendo do problema.

Figura 2.1: Fluxo de interação Ambiente-Agente



Diferentemente da maioria das formas de aprendizagem de máquina, na aprendizagem por reforço não é informado ao agente quais ações ele deve executar, ele deve descobrir por si mesmo quais ações produzem as melhores recompensa ou custos através de tentativa e erro. Experimento por tentativa e erro e recompensa atrasada são as características distintivas mais importantes da aprendizagem por reforço (SUTTON; BARTO, 1998).

A característica mais importante que distingui a aprendizagem por reforço de outros tipos de aprendizagem é que ela usa informações de treinamento que avalia as ações tomadas ao invés de indica a ação correta a ser tomada como ocorre no aprendizado supervisionado.

Há a necessidade na aprendizagem supervisionada que um professor informe ao sistema de aprendizagem o que fazer através de um conjunto de dados de exemplo de entrada-saída representativo do ambiente. Segundo (SUTTON; BARTO, 1998), não se pode dizer que um sistema de aprendizagem supervisionado aprende a controlar o seu ambiente porque ele segue a informação instrutiva que recebe. Em vez de tentar fazer seu ambiente se comportar de uma certa maneira, ele tenta se comportar como instruído pelo seu ambiente. Por outro lado, a AR aprende a realizar uma tarefa predeterminada com base em tentativa

e erro resultante da interação com o ambiente, recebendo uma informação da qualidade ou custo de cada interação e não de seu comportamento correto ou incorreto.

A interação por tentativa e erro cria a necessidade de explorar todo o ambiente ativamente. A informação puramente avaliativa recebido pelo agente de AR indica o quão boa a ação tomada é, mas não se é a melhor ou a pior ação possível. O agente deve tentar uma variedade de ações e, progressivamente, favorecer aqueles que parecem ser melhores (SUTTON; BARTO, 1998).

Selecionar as melhores ações em cada interação significa que o agente segue a recomendação de selecionar as ações ótimas para o modelo aprendido até o momento, não necessariamente para o modelo real (RUSSELL; NORVIG, 2004). Esse tipo de agente recebe o nome de agente guloso e a política selecionada de política gulosa. Então, para o agente garantir que a política encontrada é ótima para o ambiente real ele tem que experimentar todo o espaço de estado do ambiente e escolher as políticas gulosas em cada etapa. Na verdade o agente tem o compromisso em aproveita o modelo aprendido para minimizar seus custos e explorar novos estados para melhorar sua percepção do ambiente e assim ficar mais próximo da otimalidade.

O problema da aprendizagem por reforço é aprender uma sequencia de ação que minimize a soma de valores dos custos observados nas transições de estado do ambiente, ou seja, é encontrar uma política ótima que representa um mapeamento de estados em ações.

Qualquer problema de aprendizagem por reforço está ligado a três elementos principais que são: uma política, uma função de custo e uma função valor.

O objetivo de um problema de aprendizagem por reforço está intrínseco na função de custo. Enquanto a função de custo indica o que é bom em um sentido imediato a função valor especifica o que é bom a longo prazo. O sistema de aprendizagem deve buscar ações que levem a estados de menor valor, não mais baixo custo, porque essas ações levam a uma soma de custos menor a longo prazo. Segundo (SUTTON; BARTO, 1998), o componente mais importante de quase todos os algoritmos de aprendizagem por reforço é o método para estimar os valores dos estados de forma eficiente.

A politica ótima encontrada pelo sistema de aprendizagem está relacionada à função de custo. Diferentes funções de custo produzem diferentes políticas ótimas.

Um desafio que pode surgir é satisfazer certos requisitos de desempenho como máxima ultrapassagem, tempo de acomodação e/ou outros e ainda minimizar um índice de desempenho. Uma forma de satisfazer tais requisitos é codificá-los na função de custo (BUSONIU et al., 2010).

A maioria das abordagens desenvolvidas para resolver o problema de aprendizagem por reforço esta intimamente relacionada a algoritmos de programação dinâmica aplicados a processos que satisfazem a propriedade de Markov.

A tarefa de aprendizagem por reforço que satisfaz a propriedade de Markov é chamada

de um processo de decisão Markoviano (PDM) e se os espaços de estado e ação são finitos é conhecido como PDM finito. Segundo (SUTTON; BARTO, 1998), 90% das aplicações de aprendizagem por reforço moderna são baseadas em Processos de Decisão Markoviano finitos.

2.2 Programação Dinâmica (PD)

Os fundamentos da teoria da programação dinâmica são tratados agora, tendo em vista que esta é a base matemática de toda a metodologia a ser desenvolvida neste trabalho. Nesse sentido, foca-se em problemas de horizonte infinito e nos algoritmos de iteração de valor e iteração de política que são essenciais na resolução de problema de Programação Dinâmica e Aprendizagem por Reforço.

Programação Dinâmica (PD) é um dos métodos utilizados para resolver problemas de aprendizagem por reforço e é usada para encontrar a solução ótima em problemas de decisão sequencial, minimizando ou maximizando um índice de desempenho, dado um modelo completo do ambiente como um PDM. É um procedimento de resolução que divide o problema original em estágios, iniciando sua resolução do último estágio e retroagindo sequencialmente até o início. Nesse processo, solução ou soluções ótimas são encontradas a cada retroação, no final da sequencia que corresponde ao problema original, as políticas ótimas são obtidas.

É possível tratar qualquer problema de maximização como um problema de minimização, dentro do conjunto viável, fazendo a transformação $\max\{H(.)\} = \min\{-H(.)\}$ (IZMAILOV; SOLODOV, 2009). Neste trabalho, considera-se problemas de minimização, sendo esse o motivo de se utilizar mais o termo custo ao termo recompensa.

A PD clássica é muito utilizada para encontrar a solução ótima de pequenos problemas, ou seja, problemas em que o espaço de estado tem poucos elementos e para cada estado se tem um limitado conjunto de ações possíveis. Em razão do custo computacional crescer exponencialmente com o aumento da dimensão do espaço de estados, essa metodologia se torna muito difícil ou mesmo impossível quando o espaço de estado é muito grande ou contínuo. Para resolução de um problema por programação dinâmica é necessário se conhecer o modelo completo do processo, ou seja, um conjunto de probabilidade de transição de estado, um conjunto de ações, um conjunto de estados finitos e uma função de recompensa ou custo.

A técnica de programação dinâmica não trata as decisões de forma isolada, mas o conjuntos de todas as decisões, resolvendo o problema de compromisso entre decisões iniciais de baixo custo com decisões indesejadas de alto custo no futuro (BERTSEKAS, 1995).

A soma dos custo instantâneo ao longo do curso de interações é chamado de retorno ou custo total. A forma de calcular o retorno leva aos problemas de horizonte finito e infinito.

2.2.1 Propriedade de Markov e Processo de Decisão Markoviano

Um ambiente que é sintetizado pelo estado atual é um ambiente que tem a propriedade de Markov, ou seja, toda a informação relevante está no sinal de estado atual. Então, ambiente com essa dinâmica é possível prever o estado seguinte e o custo associado, dado o estado atual e a decisão tomada nesse estado. Neste caso, a dinâmica do ambiente é especificada completamente pela distribuição de probabilidade condicional

$$P_r\{x_{t+1} = x', r_{t+1} = r|x_t, \mu_t), \tag{2.1}$$

sendo x_t o estado atual, x_{t+1} o estado no instante seguinte, r_{t+1} custo incorrido após a transição de estado e μ_t a ação atual.

A equação (2.1) diz que para se conhecer o estado e custo seguintes de um ambiente com a propriedade de Markov não é necessário o conhecimento completo das estatística passadas do ambiente, mas apenas do estado e ação atuais (HAYKIN, 2008).

Um sistema cuja a dinâmica evolua probabilisticamente em um espaço finito de estado, sendo que, para cada estado há um conjunto finito de ações possíveis e, além disso, cada vez que o sistema sofre uma ação ele retorna um indicador da qualidade dessa ação, então, esse sistema opera de acordo com um Processo de Decisão Markoviano (PDM).

Um PDM é caracterizado pela tupla $(X, A(x), P(x, \mu(x), x'), r(x, \mu))$ sendo:

- X é o espaço de estado do processo, ou seja, é o conjunto de todos os estados possível do ambiente;
- A é o espaço de ações, $A(x)$ é o conjunto de todas as ações possíveis para o estado x;
- $P(x, \mu(x), x')$ é a probabilidade de transição do estado x para o estado x' dada a ação $\mu(x)$ no estado x;
- $r(.)$ é uma função que mapeia cada estado ou par estado-ação do ambiente em um valor escala que informa a qualidade de tomar uma determinada ação quando o processo está em um determinado estado. Esse valor escalar é chamado de custo ou recompensa;

Para um PDM, dados o estado atual x e a ação atual μ, a probabilidade do estado x' ser o estado seguinte é dado por

$$P^{\mu}_{xx'} = P_r\left\{x_{t+1} = x'|x_t = x, \mu_t = \mu\right\}, \tag{2.2}$$

e o custo é dado por

$$r^{\mu}_{xx'} = E\left\{r_{t+1}|x_t = x, \mu_t = \mu, x_{t+1} = x'\right\}. \tag{2.3}$$

As equações (2.2) e (2.3) determinam os mais importantes aspectos da dinâmica de um PDM finito (SUTTON; BARTO, 1998).

Os processos de Markov são governados por equação do tipo $x_{t+1} = f(x_t, \mu, u)$, sendo x_t e x_{t+1} estados do processo antes e após transição, μ é a ação de controle e u é uma componente aleatória.

2.2.2 Problema de Horizonte Finito e Horizonte Infinito

Em problemas de decisão sequencial, como é o caso dos PDM, em que o curso das transições de estado é finita, geralmente, o retorno é dado pela soma dos custos instantânea de cada transição, neste caso, o retorno é sempre limitado e dado por

$$R(x_t) = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_{t+n} = \sum_{k=0}^{n-1} r_{t+k+1}. \tag{2.4}$$

No senário estocástico a equação (2.4) é tomada como o valor esperado de diversas realizações iniciando no estado x_t. O retorno é uma medida de desempenho do agente ou controlador (RUSSELL; NORVIG, 2004).

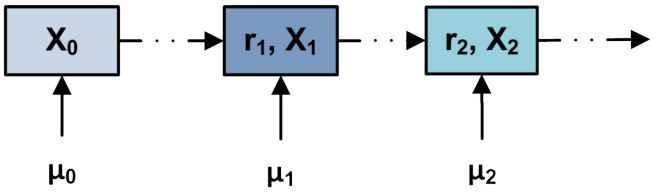
Quando os custos são acumulado ao longo de infinitas transições de estado, utiliza-se o problema de horizonte infinito descontado para garantir um retorno finito, sendo dado por

$$R(x_t) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.5}$$

onde γ é o fator de desconto com valor no intervalo $[0,\ \ 1]$. Também é possível utilizar problema de horizonte infinito para aproximar problema que envolve um grande número de estágios porém finito.

A figura (2.2) ilustra como uma ação (μ) efetuada em um estado (x) produz um custo (r), sendo este retornado juntamente com o estado seguinte ao agente. Cada quadrado representa um momento do ambiente ou estágio. A soma de todas os custos dá o valor de retorno do estado inicial (x_0) para uma dada política.

Figura 2.2: Ações(μ), estados (X) e custo instantâneo(r) no PDM



objetivo: $\min (r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \cdots)$ *onde* $0 \le \gamma \le 1$

O objetivo da Programação Dinâmica é encontrar uma política (π) ou conjunto de decisões que minimize o retorno total. O valor ótimo do retorno (R^*) é o mínimo de todos

os retornos calculados para todas as política possíveis e é representado matematicamente por

$$R^*(x_t) = \min_{\pi} R^{\pi}(x_t). \tag{2.6}$$

A política ou políticas (π^*) que minimizam o retorno são chamadas ótimas.

2.2.3 Funções Valor e as Equações de Bellman

Funções valor estão no centro do aprendizado por reforço e praticamente todos os métodos se concentram em aproximações de funções valor para calcular políticas ótimas ou sub-ótimas (WIERING; OTTERLO, 2011). A ideia chave da PD é o uso de função valor para organizar e estruturar a busca de boas políticas (SUTTON; BARTO, 1998), ou seja, cada estado ou par estado-ação tem um valor dado pela função valor, considerando uma dada política. Então, qualificar uma política (π) é encontrar a função valor para ela.

As funções valor podem ser de dois tipos:

- Função valor estado conhecida também como Função-V, representada pela letra V, é uma função que relaciona cada estados admissível do espaço de estado X a uma valor escalar, $V^{\pi} : X \to \Re$;
- Função valor ação ou Função-Q, representada pela letra Q, é uma função que relaciona cada par estado-ação possível do produto cartesiano do espaço de estado X com o espaço de ações A a uma valor escalar, $Q^{\pi} : X \times A \to \Re$

O valor de um estado x_t sob a política π é o retorno esperado quando se inicia no estado x_t e segue-se a política π (SUTTON; BARTO, 1998). Para a função-V, o valor de um estado é dado por

$$V^{\pi}(x) = E_{\pi}\left\{R(x_t)|x_t = x\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r(x_{t+k}, \mu_{t+k})|x_t = x\right\}, \tag{2.7}$$

e, para a função-Q, é dado por

$$\begin{aligned} Q^{\pi}(x,\mu) &= E_{\pi}\left\{R(x_t,\mu_t)|x_t = x, \mu_t = \mu\right\} && (2.8)\\ &= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r(x_{t+k}, \mu_{t+k})|x_t = x, \mu_t = \mu\right\}, && (2.9) \end{aligned}$$

que é parecida com a função (2.7), mas difere desta pelo fato do custo médio ser calculado para cada par (x, μ). O termo E_{π} é o valor esperado quando o agente segue a política π.

Calcular os valores dos estados utilizando as definições acima traz um inconveniente, pois antes de calcular o valor de um estado é necessário percorrer toda a trajetória imposta pela política. Uma forma de contornar essa dificuldade é utilizar a equação de Bellman que relaciona valores de estados vizinhos ou sucessores.

Primeiramente, foca-se na caracterização da equação de Bellman para a função-V e após, na função-Q. Estas são fundamentais nos algoritmos de iteração de política e iteração de valor (BUSONIU et al., 2010).

Para estruturar (2.7) na forma da equação de Bellman, deve-se separar o primeiro termo (k=0) e pôr em evidencia o γ da soma restante da função-V, chegando a

$$V^{\pi}(x) = E_{\pi}\left\{r\left(x_t,\mu\right)+\gamma\sum_{k=1}^{\infty}\gamma^{k-1}r\left(x_{t+k},\mu'\right)|x_t=x\right\} \tag{2.10}$$

$$= E_{\pi}\left\{r\left(x_t,\mu\right)+\gamma\sum_{k=0}^{\infty}\gamma^{k}r\left(x_{t+k+1},\mu'\right)|x_t=x\right\}, \tag{2.11}$$

as ações, μ, são tomadas do conjunto de ações possíveis para o estado x.

Por fim, utiliza-se novamente a equação (2.7), agora aplicada ao estado x_{t+k+1}, substituindo em (2.11) para obter a equação de Bellman

$$V^{\pi}(x) = E_{\pi}\left\{r\left(x_t,\mu\right)+\gamma V^{\pi}(x_{t+k+1})\right\}, \tag{2.12}$$

que relaciona o valor de um estado ao valor do seu sucessor, ou seja, para se conhecer o valor de um estado basta conhecer o valor do estado sucessor e o custo da transição.

De forma semelhante acima, a função valor-Q na forma de Bellman fica

$$Q^{\pi}(x,\mu) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k}r(x_t,\mu_t)|x_t=x,\mu_t=\mu\right\} \tag{2.13}$$

$$= E_{\pi}\left\{r(x_t,\mu_t)+\gamma\sum_{k=0}^{\infty}\gamma^{k}r(x_{t+k+1},\mu_{t+k+1})|x_t=x,\mu_t=\mu\right\} \tag{2.14}$$

$$= E_{\pi}\left\{r(x_t,\mu_t)+\gamma Q^{\pi}(x_{t+k+1},\mu_{t+k+1})|x_t=x,\mu_t=\mu\right\}. \tag{2.15}$$

Pode-se ver que o valor da função-Q para um determinado par estado-ação (x,μ) para uma política π é dado pelo custo imediato de se tomar a ação μ no estado x mais o valor descontado da função-Q do estado sucessor seguindo a política π.

As equações de Bellman (2.12) e (2.15) são o centro dos algoritmos de iteração de política e iteração de valor.

2.2.4 Funções valor Ótimas e Políticas Ótimas

Uma função valor para uma política π_1 é dita ser melhor que a função valor para uma política π_2 se e somente se $V^{\pi_1} \leq V^{\pi_2}$ para todo o espaço de estado (BUSONIU et al., 2010). Então, a função-V ótima e a função-Q ótima são as melhores funções que se pode

obter para qualquer política e são dadas, respectivamente, por (SUTTON; BARTO, 1998)

$$V^*(x) = \min_{\pi} \left(V^{\pi}(x) \right), \tag{2.16}$$

$$Q^*(x, \mu) = \min_{\pi} \left(Q^{\pi}(x, \mu) \right). \tag{2.17}$$

Em geral, qualquer política π^* que satisfaça a equação

$$\pi^*(x) \in \arg\min_{\pi} \left(V^*(x) \right) \tag{2.18}$$

ou a equação

$$\pi^*(x) \in \arg\min_{\mu} \left(Q^*(x, \mu) \right) \tag{2.19}$$

são ditas ótimas e as políticas π que satisfazem

$$\pi(x) \in \arg\min_{\pi} \left(V^{\pi}(x) \right), \tag{2.20}$$

ou

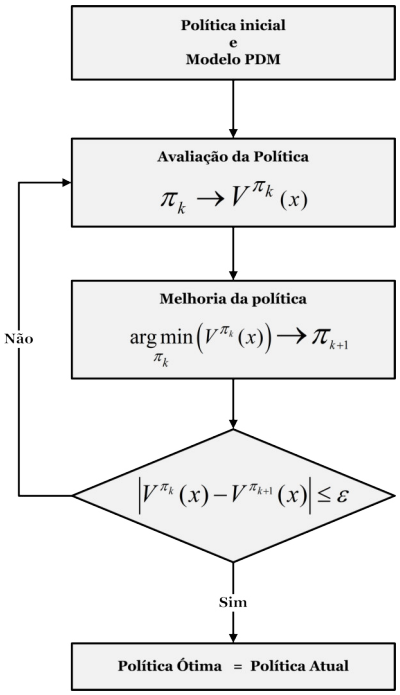
$$\pi(x) \in \arg\min_{\mu} \left(Q^{\pi}(x, \mu) \right). \tag{2.21}$$

são ditas gulosa em V e Q, respectivamente.

2.2.5 Algoritmo de Iteração de Política (IP)

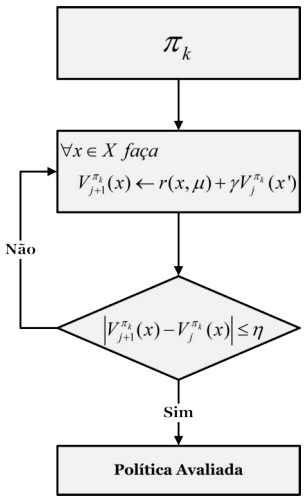
A proposta do algoritmo de Iteração de Política (IP) é, partindo de uma política arbitraria inicial π_0, encontrar a política ótima π^*. Isso é realizado em duas etapas, uma de avaliação da política corrente e outra de melhoria da politica. Tomando como base a função-V, avalia-se a política inicial π_0, ou seja, determina-se V^{π_0}. Esta por sua vez servirá de base para se encontrar uma nova política melhorada π_1 que é gulosa em relação a V^{π_0}. avalia-se agora a política π_1, encontrando uma nova função valor V^{π_1}. Esse processo segue nesse ciclo até que algum critério de parada seja atingido ou uma política ótima seja retornada. A prova de convergência desse algoritmo é baseada no teorema do ponto fixo para contrações ou de Banach (BUSONIU et al., 2010) (LIMA, 2009). O Diagrama de Bloco da figura (2.3) ilustra o algoritmo IP utilizando a função-V.

Figura 2.3: Diagrama de Bloco do Algoritmo de Iteração de Política



O processo de avaliação de política consiste em determinar a função valor para cada estado ou par estado-ação utilizado a política escolhida. Na avaliação de política é gerada uma equação de Bellman (2.12) para cada estado. Para n estados se tem n equações lineares com n incógnitas que podem ser resolvidas de forma exata por métodos da álgebra linear para sistemas com pouco estado possíveis. Pode-se usar, para sistema com espaço de estado grande, um método iterativo que é repetido p vezes para gerar a estimativa seguinte do retorno (RUSSELL; NORVIG, 2004). O fluxograma da figura (2.4) ilustra o processo iterativo de avaliação da política em que $V_j^{\pi_k}(x)$ é o valor da função-V na j-ésima iteração no estado x.

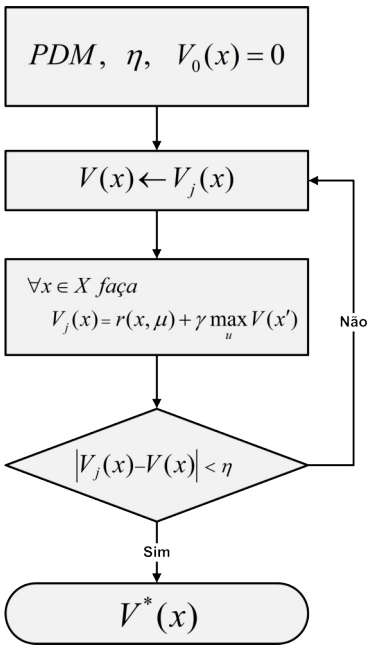
Figura 2.4: Diagrama de Bloco de Avaliação de Política utilizando a função-V



2.2.6 Algoritmo de Iteração de Valor (IV)

O objetivo do algoritmo de iteração de valor (IV) é encontrar uma função valor ótima e daí derivar a política ótima. A base para o algoritmo IV são as equações de Bellman de otimalidade (2.16) e (2.17) que as transforma em uma regra de atualização iterativa. O algoritmo IV é semelhante ao processo de atualização de política vista acima, mas difere por requerer o máximo sobre todas as ações. Inicialmente, uma função valor arbitrária é selecionada e a cada iteração do algoritmo a função valor é atualizada utilizando a função valor armazenada no ciclo anterior da sequencia de iterações. Esses ciclos continuam até que a condição de término seja satisfeita, conforme ilustra a Figura (2.5).

Figura 2.5: Diagrama de Bloco do Algoritmo de Iteração de Valor



A aplicação com frequência infinita desse algoritmo garante que os valores finais da função valor são solução das equações de otimalidade de Bellman (RUSSELL; NORVIG, 2004).

2.3 Programação Dinâmica Aproximada (PDA)

Nesta seção, A programação Dinâmica Aproximada (PDA) é tratada. PDA é uma forma de resolução de Processo de Markov de forma aproximada que evita com sucesso a "maldição da dimensionalidade", óbice da PD. Como foi visto, a programação dinâmica clássica é uma método de difícil utilização prática quando o processo é composto por grandes espaço de estado e ações, podendo tornar o problema intratável.

Werbos (LEWIS; LIU, 2013) foi o primeiro a propor construir sistemas de aprendizagem por reforço utilizando a aproximação adaptativa da equação de Bellman, criando assim a

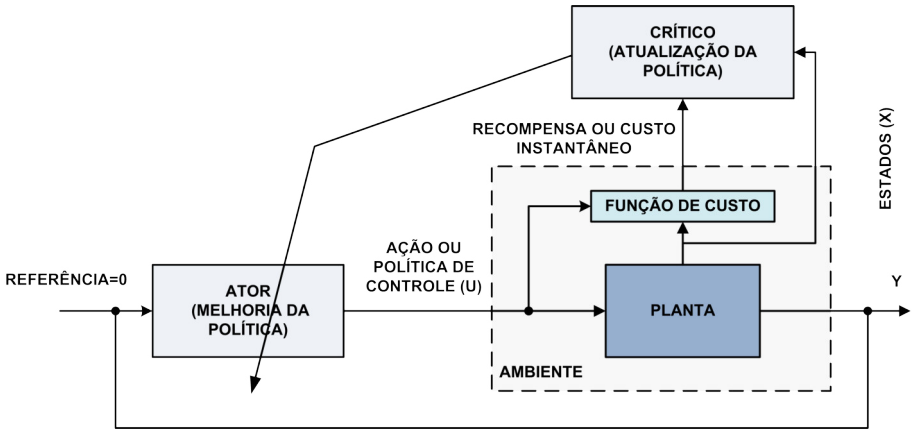
PDA moderna. Antes, a AR e PD eram vistas de forma completamente separadas.

O uso da programação dinâmica clássica para encontrar a política ótima de um PDM com pequenos espaços de estado é bastante viável, caso se conheça o modelo completo do ambiente, pois nessa condição é possível determinar os valores exatos de cada estado ou par de estado-ação e armazená-los em uma tabela em memória. No entanto, para PDM com elevada dimensão e com grande quantidade de ações possíveis por estado pode ser impossível percorrer todos esses estados e calcular seus valores. Por outro lado, pode ser possível encontrar uma função que se adeque à função que gerou todos os valores dos estado e armazenar em memoria apenas alguns parâmetros. Essa representação é dita aproximada porque não se conhece a estrutura da função verdadeira, sendo possível escolher uma que talvez não corresponda a verdadeira. O que é mais significativo é a generalização que ocorre para estados não visitados, ou seja, visitando poucos estados é possível encontrar um função valor bem adequada para todo o espaço de estado.

Essas limitações da PD, espaço pequeno e necessidade de modelo, podem ser contornadas através da programa dinâmica aproximada (PDA). O agente de PDA através de interação com o ambiente estima uma função que se ajusta aos valores ótimos de cada estado ou par estado-ação. De posse dessa função aproximada, é possível determinar a política ótima ou subótima. Em cada passo, o agente de PDA deve estimar um modelo, calcular a função valor ótima e a política ótima para o modelo aprendido.

Como mostra a figura (2.6), existem dois blocos básicos no projeto de controle por PDA: O Crítico e o Ator. A avaliação do desempenho do controlador é realizado no crítico que envia ao ator tal avaliação. O ator de posse dessa avaliação, gera nova ação de forma a melhorar o desempenho. A aplicação da PDA em controle é bastante ampla, baseado em modelo ou não, *online* ou *offline* e tempo real.

Figura 2.6: Esquema de PDA ou Crítico Adaptativo na estrutura Ator-Crítico



Dentre os esquema de PDA existentes, limitou-se aqui às estruturas propostas por Werbos (WERBOS, 1990)(WERBOS, 1994): a Programação Dinâmica Heurística (HDP), a Programação Dinâmica Heurística Dual (DHP), a Programação Dinâmica Heurística

Dependente de Ação (ADHDP) e a Programação Dinâmica Heurística Dual Dependente de Ação (ADDHP).

Neste trabalho, o processo Markoviano é representado por

$$x_{k+1} = f\left(x_k, \mu_k\right), \tag{2.22}$$

e a ação de controle por

$$\mu_k = \mu(x_k). \tag{2.23}$$

2.3.1 Programação Dinâmica Heurística (HDP)

O método de programação dinâmica heurística é utilizado para estimar uma função valor estado a partir de amostras do ambiente. HDP utiliza o algoritmo de iteração de política para estimar uma função valor estado de uma dada política, apenas recebendo amostras instantâneas dos custos. Esta etapa é a avaliação da política que é realizada pelo crítico. Após avaliar a política, a etapa seguinte é melhorar a política que é responsabilidade do ator, sendo necessário o modelo do ambiente.

A função-V para uma dada política é dada por

$$V(x_k) = r(x_k, \mu_k) + \gamma V(f(x_k, \mu_k)), \tag{2.24}$$

como se está interessado em função valor estado parametrizada, tem-se

$$V(x_k, \theta) = r(x_k, \mu_k) + \gamma V(f(x_k, \mu_k), \theta), \tag{2.25}$$

sendo θ os parâmetros da função valor aproximada.

A equação (2.24) pode ser vista com um alvo ($V^{Alvo} = V(x)$) para a equação (2.25) no processo de aproximação paramétrica

$$h\left(\ V^{Alvo}, \ \ V(x_k, \theta') \ \right), \tag{2.26}$$

que representa uma função objetivo de um algoritmo de aprendizagem supervisionado como, por exemplo, $h\left(\ V^{Alvo}, \ \ V(x_k, \theta) \ \right) = \left(V^{Alvo} - V(x_k, \theta)\right)^2$ para o LMS.

Para atualizar o parâmetro θ da função-V parametrizada é necessário encontrar o argumento que minimizar h, ou seja,

$$\theta = \arg\min_{\theta'} \left(h\left(\ V^{Alvo}, \ \ V(x_k, \theta') \ \right)\right). \tag{2.27}$$

A parametrização da função valor estado induz a parametrização da política μ

$$\mu(x_k) = \mu(x_k, w), \tag{2.28}$$

sendo w o vetor de parâmetros da política. Portanto, deve-se encontrar w de forma a minimizar a função valor estado, ou seja,

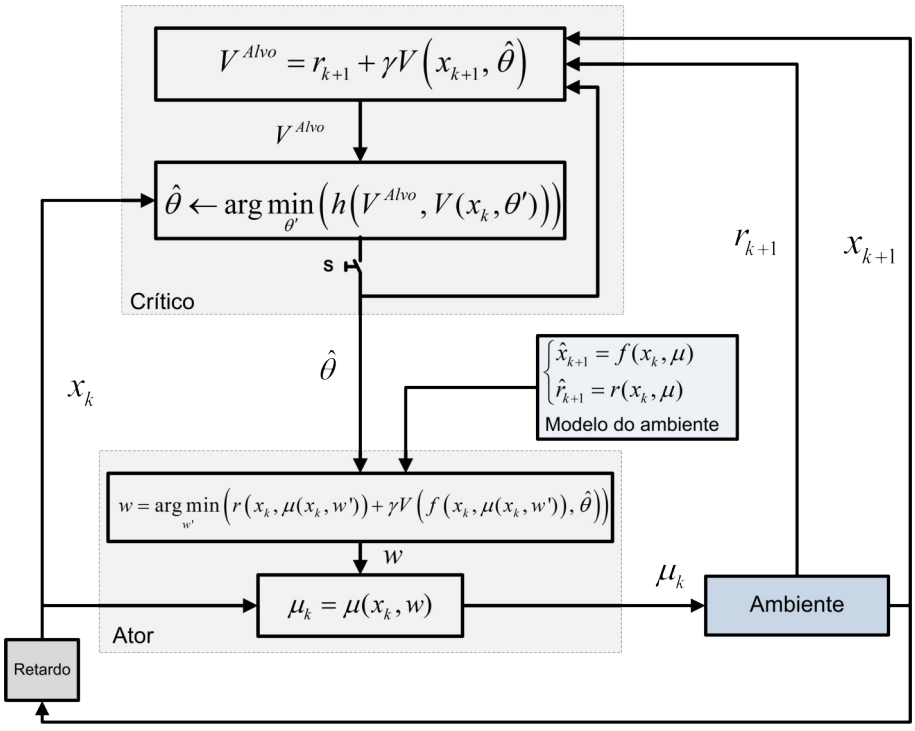
$$w = \arg\min_{w'} \left(r\left(x_k, \mu_k(x_k, w')\right) + V\left(f\left(x_k, \mu_k(x_k, w')\right), \theta\right)\right), \tag{2.29}$$

Para calcular w em (2.29), faz-se

$$\frac{\partial V}{\partial w} = \frac{\partial r}{\partial \mu}\frac{\partial \mu}{\partial w} + \gamma \frac{\partial V}{\partial f}\frac{\partial f}{\partial \mu}\frac{\partial \mu}{\partial w} = 0. \tag{2.30}$$

A figura (2.7) mostra o diagrama esquemático da estrutura HDP para aprendizagem *online*. Neste esquema, há dois blocos, o crítico e o ator que usam aproximações da função-V e da política. Tais aproximações podem ser implementadas por qualquer estrutura de aproximação universal como rede neural, fuzzy e regressor.

Figura 2.7: Esquema HDP



A saída do crítico é uma estimativa dos parâmetros da função valor com estrutura definida antecipadamente. O ator fornece a lei de controle de retroação de estado.

Como mostra a figura (2.7), o crítico necessita do conhecimento do alvo para a adaptação de $V(x_k)$. O alvo é encontrado esperando o custo r_{k+1} e o estado seguinte x_{k+1} estarem disponíveis para assim calcular $V(x_{k+1})$ utilizando a função-V estimada em k. Uma outra forma é utilizar um modelo do ambiente e calcular $V(x_{k+1})$ em k a partir de x_{k+1} estimado do modelo.

2.3.2 Programação Dinâmica Heurística Dual (DHP)

Diferentemente da HDP que estima a função valor estado diretamente, a programação dinâmica heurística dual estima a derivada parcial da função-V em relação aos estados, o que a torna dependente do modelo do processo nas etapas de avaliação e melhoria da política. A partir da equação (2.24), tem-se

$$\frac{\partial V(x)}{\partial x} = \frac{\partial r}{\partial x} + \frac{\partial r}{\partial \mu}\frac{\partial \mu}{\partial x} + \gamma \frac{\partial V}{\partial f}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial \mu}\frac{\partial \mu}{\partial x}\right) = r_x + r_\mu \mu_x + \gamma V_f \left(f_x + f_\mu \mu_x\right), \quad (2.31)$$

onde o subíndice no lado direito da equação (2.31) se refere a derivada parcial em relação a variável indicada por esse subíndice.

Pode-se utilizar a equação (2.31) como alvo para a função parametrizada $V_x(x, \theta)$, sendo θ seu vetor de parâmetros.

Para atualizar o vetor de parâmetros θ da derivada parcial da função-V é necessário encontrar o argumento que minimizar a função objetivo, h, de um algoritmo de aprendizagem supervisionado

$$\theta = \arg\min_{\theta'} \left(h \left(\begin{array}{cc} V_x^{Alvo}, & V_x(x_k, \theta') \end{array} \right) \right). \quad (2.32)$$

A parametrização da função-V induz a parametrização da política μ

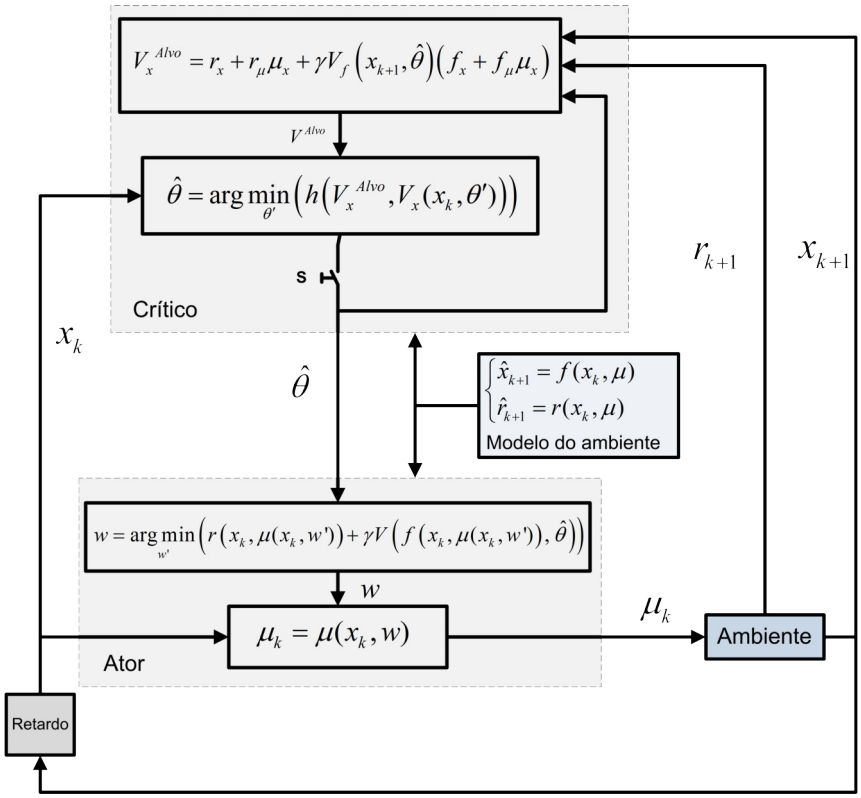
$$\mu(x) = \mu(x, w), \quad (2.33)$$

sendo w o conjunto de parâmetros da política. Portanto, deve-se encontrar w de forma a minimizar a função-V, ou seja,

$$w = \arg\min_{w'} \left(r(x, \mu(x, w')) + V\left(f\left(x, \mu(x, w'), \theta\right)\right)\right). \quad (2.34)$$

A estrutura do método DHP pode ser vista na figura (2.8).

Figura 2.8: Esquema DHP



2.3.3 Programação Dinâmica Heurística Dependente de Ação (ADHDP)

Programação dinâmica heurística dependente de ação ou aprendizagem Q é um método bastante utilizado quando o processo de decisão de Markov que se deseja solucionar não é conhecido antecipadamente. O método utiliza a função-Q que é uma função depende de estado e ação de controle.

A equação de Bellman para função-Q ao longo de uma trajetória é dada pela equação (2.15) que em sua forma determinística fica

$$Q(x_k, \mu(x_k)) = r(x_k, \mu(x_k)) + Q\left(x_{k+1}, \mu(x_{k+1})\right), \tag{2.35}$$

na versão parametrizada

$$Q(x_k, \mu(x_k)) = r(x_k, \mu(x_k)) + \gamma Q\left(x_{k+1}, \mu(x_{k+1}), \theta\right), \tag{2.36}$$

que pode ser usada como alvo para algum algoritmo de aprendizagem supervisionada no processo de aproximação paramétrica da função valor ação ou função-Q.

Para atualizar o parâmetro θ da função-Q parametrizada é necessário encontrar o

argumento que minimizar o mapeamento h, ou seja,

$$\theta = \arg\min_{\theta'} \left(h \left(\begin{array}{cc} Q^{Alvo}, & Q\left(x_k, \mu_k, \theta'\right) \end{array} \right) \right). \tag{2.37}$$

A melhoria da política é feita encontrado o valor mínimo de Q para todas as ações possíveis para cada estado, ou seja,

$$\mu = \arg\min_{\mu'} \left(Q\left(x, \mu'\right) \right). \tag{2.38}$$

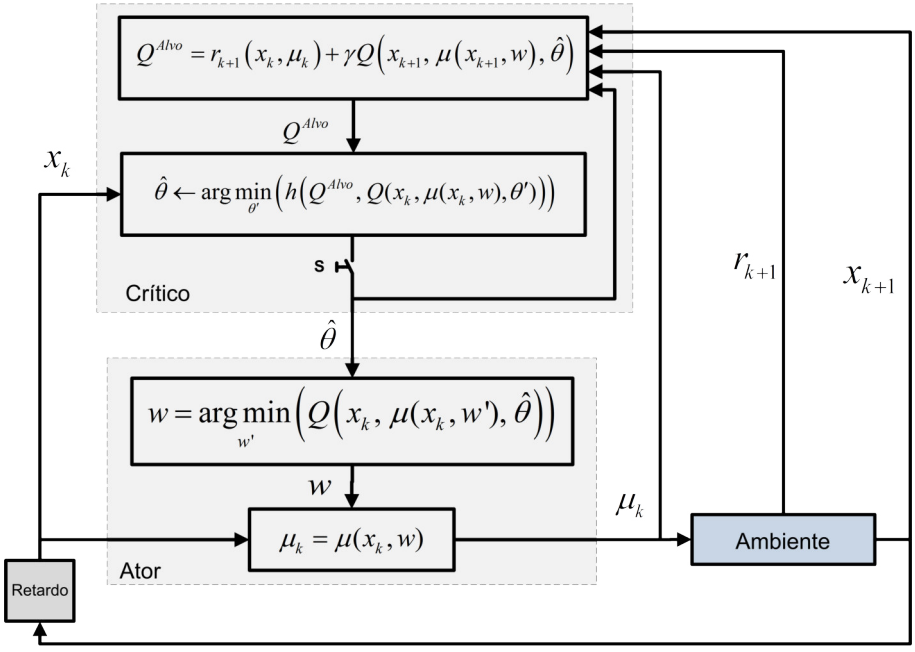
A equação (2.38) pode ser resolvida tomando

$$\frac{\delta Q}{\delta \mu} = 0, \tag{2.39}$$

que é a derivada parcial da função-Q em relação a ação de controle.

A estrutura do método ADHDP pode ser vista na figura (2.9).

Figura 2.9: Esquema ADHDP



2.3.4 Programação Dinâmica Heurística Dual Dependente de Ação (ADDHP)

Semelhantemente à DHP em relação a HDP, a ADDHP pretende encontrar a política ótima a partir da derivada parcial da função-Q em relação aos estados e ações.

A partir da equação (2.35), o alvo para o ADDHP é dado por

$$\left[\begin{array}{cc} \frac{\partial Q}{\partial x} & \frac{\partial Q}{\partial \mu} \end{array} \right] = \left[\begin{array}{cc} \frac{\partial r}{\partial x} + \frac{\partial Q}{\partial f}\frac{\partial f}{\partial x} + \frac{\partial Q}{\partial \mu}\frac{\partial \mu}{\partial f}\frac{\partial f}{\partial x} & \frac{\partial r}{\partial \mu} + \frac{\partial Q}{\partial f}\frac{\partial f}{\partial \mu} + \frac{\partial Q}{\partial \mu}\frac{\partial \mu}{\partial f}\frac{\partial f}{\partial x} \end{array} \right] \tag{2.40}$$

ou

$$\left[\begin{array}{cc} Q_x^{alvo} & Q_\mu^{alvo} \end{array}\right] = \left[\begin{array}{cc} r_x + Q_f f_x + Q_\mu \mu_f f_x & r_\mu + Q_f f_\mu + Q_\mu \mu_f f_x \end{array}\right], \tag{2.41}$$

utilizando a notação Q_x para $\frac{\partial Q}{\partial x}$ e de forma semelhante para as demais derivadas.

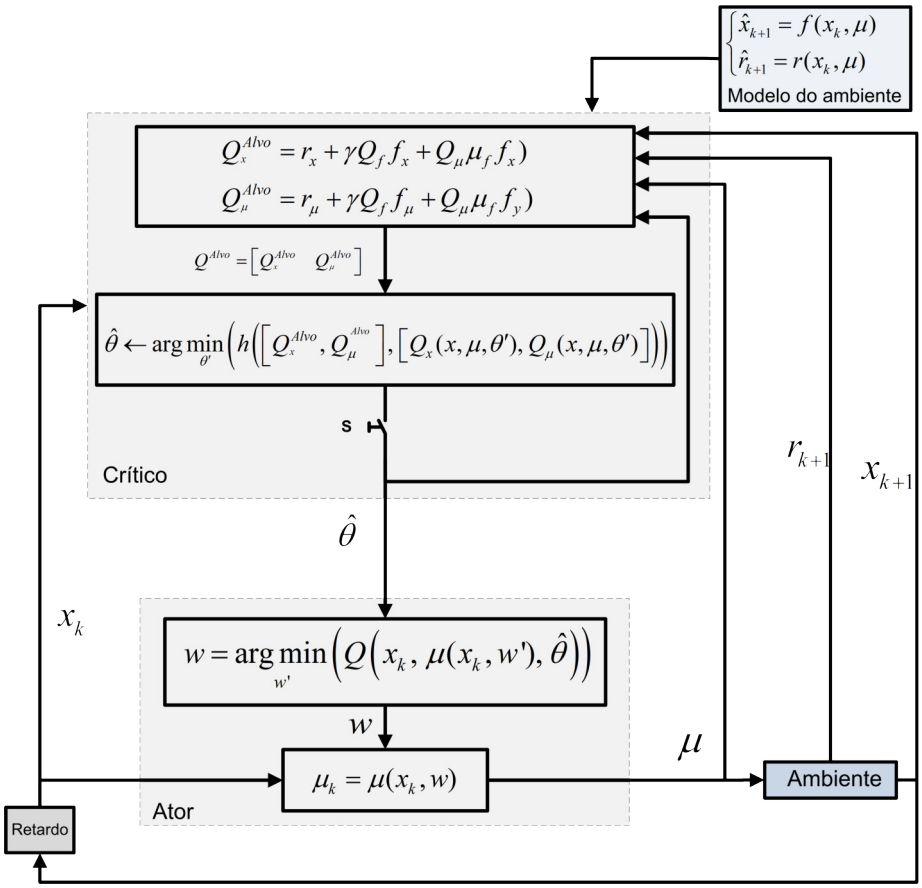
Para atualizar o parâmetro θ da derivada da função-Q parametrizada é necessário encontrar o argumento que minimizar o mapeamento h, ou seja,

$$\theta = \arg\min_{\theta'} \left(h \left(\left[\begin{array}{cc} Q_x^{alvo} & Q_\mu^{alvo} \end{array}\right], \left[\begin{array}{cc} Q_x\left(x_k, \mu_k, \theta'\right) & Q_\mu\left(x_k, \mu_k, \theta'\right) \end{array}\right] \right) \right). \tag{2.42}$$

A atualização da política é feita da mesma forma do caso ADHDP.

A figura(2.10) mostrada o diagrama esquemático do método ADDHP.

Figura 2.10: Esquema ADDHP



3

Algoritmos para Estimação de Parâmetros

Sumário

3.1 Introdução

Como foi visto, os algoritmos baseados em programação Dinâmica requer, para resolução de PDM com grandes espaços de estado ou ação, uma aproximação da função valor para evitar a chamada "maldição da dimensionalidade".

Os aproximadores de função estão incluídos em duas grandes classe: Aproximadores paramétricos e não paramétricos. Aproximadores paramétricos tem sua forma e o número de parâmetros definidos antecipadamente em tempo de projeto, ele não tem dependência direta dos dados. Enquanto, aproximadores não paramétricos tem uma dependência direta dos dados disponíveis, é a partir deles que se vai determinar sua forma e quantidade de parâmetros.

Aproximadores paramétricos podem ser lineares e não lineares nos parâmetros, mas aproximadores paramétricos lineares são muitas vezes preferidos em PD e AR por facilitarem a analise teórica dos resultados de algoritmos ARPD (BUSONIU et al., 2010). Foca-se neste trabalho apenas nos aproximadores lineares nos parâmetros.

Um aproximador paramétrico é um mapeamento do espaço de parâmetros ($\Re^n$) no espaço de funções (Ψ), ou seja, $F : \Re^n \rightarrow \Psi$. Cada vetor de parâmetros está relacionado a uma função no espaço de funções.

Uma forma de aproximar uma função valor, função-V por exemplo, por uma função F que é linear nos parâmetros é empregar um conjunto de N funções de base dependente de estado

$$\phi = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_N(x) \end{bmatrix}, \tag{3.1}$$

e um vetor de parâmetros

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_N \end{bmatrix}, \tag{3.2}$$

de N componentes e utilizar a estrutura

$$V(x) = F(\theta, x) = \phi^T \theta. \tag{3.3}$$

Aproximação de função é um processo de aprendizagem supervisionado em que se tem um conjunto de dados de entrada e saída. No entanto, no caso específico da aproximação de função valor, não se tem um valor alvo fixo desejado, ou melhor, o valor alvo é móvel, mas tende a um valor fixo durante a aprendizagem.

Existem dois aspectos a serem definidos para o problema de aproximar uma função por

uma outra: uma é a forma da função com suas funções de base e a outra os parâmetros. Então, escolhida a estrutura da função, resta determinar o conjunto de parâmetros que melhor aproxima esta da função alvo. Por último, definindo-se por uma função de aproximação linear nos parâmetros, esse conjunto de parâmetros pode ser determinado, por exemplo, por um método de regressão linear.

3.2 Estrutura do Modelo

A estrutura considera aqui para a funções de aproximação é uma estrutura lineares nos parâmetros dada por

$$y = \phi_i^T \theta_i, \tag{3.4}$$

onde y é a saída do modelo da função de aproximação, ϕ representa o vetor ou matriz de regressores e θ o vetor de parâmetros estimado do processo.

Adota-se, ainda, d como o alvo ao alcançar e $e = d - y$ como o erro de estimação.

Como se está interessado em processos de Markov com os dados disponibilizados *online*, é interessante utilizado algoritmos recursivos, tais como o RLS ou baseados-LMS, para determinação dos parâmetros da função valor aproximada. A seguir, os algoritmos utilizados para estimar os parâmetros da função que aproxima a função valor real de um PDA são apresentados, resumidamente e sem demonstração.

3.3 Mínimos Quadrados Recursivos com Fator de Esquecimento (RLS-λ)

Os estimadores RLS-λ são bastante utilizados em controladores que se ajustam em tempo real a mudanças no ambiente. O RLS-λ tem o objetivo de minimizar a soma dos quadrados dos erros (e) entre a observação (d) e a saída do modelo, ponderados por uma valor chamado fator de esquecimento (λ). Considerou-se aqui o fator de esquecimento constante e limitado ao intervalo $]0,\ 1]$. Ele tem a função de diminuir a influência de erros passados na estimação presente.

O que se deseja é minimizar

$$J(\theta) = \sum_{i=1}^{s} \lambda^{i-1} e(i)^2, \tag{3.5}$$

sendo $e(i) = d(i) - \phi(i)^T \theta(i-1)$ e θ os parâmetros a estimar.

A equação de atualização dos parâmetros é dado por

$$\theta(i) = \theta(i-1) + P(i)\phi(i)e(i), \tag{3.6}$$

sendo P a matriz de covariância dos parâmetros estimados e é determinada por

$$P(i) = \frac{1}{\lambda}\left(P(i-1) - \frac{P(i-1)\phi(i)\phi^T(i)P(i-1)}{\lambda - \phi^T(i)P(i-1)\phi(i)}\right), \tag{3.7}$$

onde ϕ é o vetor de regressores.

A demonstração desse algoritmo pode ser encontrado em (AGUIRRE, 2007)(JOHNSON, 1988).

Para utilização prática desse algoritmo, algumas precauções devem ser tomadas para evitar instabilidade, entre elas podem ser citadas:

- O valor do fator de esquecimento deve está geralmente entre 0,95 e 0,999 indicando que as observações corrente terão importância aproximadamente até 20ª e 1000ª iterações a frente, respectivamente;
- Quanto menor o fator de esquecimento mais rápida é a convergência dos parâmetros. No entanto, o algoritmo fica mais propenso a instabilidade numérica;
- No inicio do processo de estimação paramétrica o valor de $P(0)$ deve ser definida positiva e geralmente é escolhido com um múltiplo da matriz identidade. Esse múltiplo deve ser menor, quanto maior é a convicção de se tem uma boa escolha do parâmetros inicial ($\theta(0)$).

3.4 Algoritmo do Mínimo quadrado Médio (LMS)

O algoritmo LMS surge da aplicação do método da descida mais íngreme à função de custo dada pelo quadrado do valor instantâneo do erro de estimação, equação (3.8), e restrição dada pela equação (3.4).

$$J(\theta) = e(i)^2, \tag{3.8}$$

sendo

$$e(i) = d(i) - \theta(i)^T\phi(i), \tag{3.9}$$

e $d(i)$ é a resposta desejada, $\theta(i)$ é o vetor de parâmetros e $\Phi(i)$ é o regressor.

Baseado no método da decida mais íngreme, encontra-se a equação de atualização dos pesos que é dada por (HAYKIN, 2001)

$$\theta(i+1) = \theta(i) + \mu e(i)\phi(i), \tag{3.10}$$

sendo μ a taxa de aprendizagem.

Características do algoritmo:

- A taxa de aprendizagem (μ) é uma relação inversa da constante de tempo do processo adaptativo, ou seja, quanto menor μ maior é o tempo que o LMS leva para convergir;
- Os parâmetros de estimação do algoritmo LMS, durante a aprendizagem, seguem uma trajetória aleatória por serem uma estimativa dos parâmetros quando se utilizasse o método da descida mais íngreme (HAYKIN, 2008);
- O ganho de atualização dos parâmetros é constante;
- O LMS não requer o conhecimento das estatísticas do ambiente (HAYKIN; WIDROW, 2003);
- A escolha de μ para convergência do LMS depende da estatística do vetor de entrada;
- Simplicidade computacional elevada.

3.5 Algoritmo LMS Normalizado (NLMS)

O algoritmo NLMS (LMS normalizado) também conhecido com algoritmo de projeção modificado ou normalizado é uma boa alternativa ao LMS. A diferença dele em relação ao LMS está na função de custo adotada. O processo de otimização é baseado na diferença quadrática de duas estimativas consecutivas dos parâmetros de adaptação. Ou seja, a função de custo é dado por

$$J(\theta) = \frac{1}{2} \left\| \theta(i) - \theta(i-1) \right\|, \tag{3.11}$$

sujeita a restrição imposta pela equação (3.4). Neste caso, utilizando o método dos multiplicadores de Lagrange, tem-se

$$L = \frac{1}{2} \left(\theta(i) - \theta(i-1) \right)^T \left(\theta(i) - \theta(i-1) \right) + \alpha \left(d(i) - \phi(i)^T \theta(i) \right), \tag{3.12}$$

sendo L a Lagrangiana e $e(i) = d(i) - \phi(i)^T \theta(i)$.

A solução dessa Lagrangiana (ÅSTRÖM; WITTENMARK, 2008) resulta em

$$\theta(i+1) = \theta(i) + \frac{\mu}{\phi(i)^T \phi(i) + \xi} e(i) \phi(i), \tag{3.13}$$

que é a equação de atualização dos parâmetros, sendo $0 < \mu < 2$ a taxa de aprendizagem e ξ uma constante escalar para evitar divisão por zero. O termo $\frac{\mu}{\phi(i)^T \phi(i) + \xi}$ determina o ganho de atualização dos parâmetros.

Características do algoritmo:

- Diferentemente do LMS, este algoritmo não é sensível à magnitude das entradas;

- A taxa de aprendizagem é diretamente proporcional à velocidade de convergência;
- O ganho de atualização dos parâmetros varia durante a aprendizagem, mas é comum a todos eles.
- Tem uma boa velocidade de convergência;
- Complexidade computacional baixa.

3.6 Algoritmos Adaptativos Proporcionais

Algoritmos Adaptativos Proporcionais estão sendo muito empregados em aplicações de identificação de sistema, processamento de sinais e predição (HAYKIN; WIDROW, 2003).

Algoritmos adaptativos geralmente aplicam o mesmo ganho a todos os parâmetros como é caso dos LMS e NLMS, apesar deste último ter um ganho variável. Esses algoritmos tem uma velocidade de convergência baixa em ambientes de alto grau de esparsidade. Para melhorar o desempenho desses algoritmos foi desenvolvido no Bell Laboratories, no ano 2000, uma abordagem baseada no NLMS que distribuía ganhos diferentes e individuais aos parâmetros do estimador a partir de algum critério. Tal abordagem foi denominada de NLMS Proporcional (PNLMS) e a partir dela surgiram diversas outras alternativas como o IPNLMS e IAFPNLMS que serão também visto a seguir.

Nesta seção, é convencionado que a letra i representa o ciclo de iteração, n a posição de um elemento no vetor e ξ uma constante escalar positiva utilizada para evitar divisão por zero.

3.6.1 Esparsidade

Esparsidade é uma característica que certos processos físicos tem e que pode ser explorada para facilita as manipulações algébrica e redução de memória. Processos lineares representados por matrizes cujos os elementos são quase todos nulos é um exemplo de processo esparso.

Os algoritmos adaptativos proporcionais mostrados nessa seção explora a característica esparsa dos processos. Uma forma de quantificar a esparsidade é utilizando a equação (HUANG; BENESTY; CHEN, 2006) que relaciona a norma da soma e a norma euclidiana da resposta ao impulso do ambiente, tal quantidade é chamada de grau de esparsidade.

$$S_D(\mathbf{w}^o) \triangleq \frac{N}{N-\sqrt{N}}\left(1-\frac{\|\mathbf{w}^o\|_1}{\sqrt{N}\,\|\mathbf{w}^o\|_2}\right), \tag{3.14}$$

onde $\mathbf{w^o}$ é um vetor de parâmetros de dimensão N and $\|\mathbf{w}^o\|_1$ e $\|\mathbf{w}^o\|_2$ são as normas l_1 e l_2.

Da equação (3.14), pode-se verificar que S_D varia entre zero e um. Assim, quanto mais próximo de um estiver o grau de esparsidade mais o ambiente é esparso e, de modo contrário, quanto mais próximo de zero menos esparso é o ambiente. Pode-se ver que um processo que é representado por uma matriz em que todos os elementos são iguais entre si é um processo que tem grau de esparsidade zero e que processo em que o número de parâmetros tende a infinito com as normas finitas tem seu grau de esparsidade tendendo a um.

3.6.2 Algoritmo NLMS Proporcional (PNLMS)

Nos algoritmos LMS e NMLS a mesma taxa de aprendizagem é aplicada a todos os parâmetros, diferentemente do PNLMS que aplica uma taxa de aprendizagem diferente a cada parâmetros, privilegiando os considerados mais importantes. Ele tem uma convergência bem mais rápida que o LMS e NLMS. No entanto, esse algoritmo tem sua maior aplicação em ambiente esparso.

O PNLMS é descrito pelo conjunto de equações de (3.15) a (3.18) (DUTTWEILER, 2000)(HAYKIN; WIDROW, 2003)(SOUZA, 2012).

A equação de atualização dos parâmetros é dada por

$$\theta(i+1) = \theta(i) + \frac{\mu G(i)}{\phi(i)^T G(i)\phi(i) + \xi} e(i)\phi(i), \tag{3.15}$$

onde μ é a taxa de aprendizagem e ξ é um termo de regulação que evita divisão por zero e $e(i) = d(i) - \theta(i)^T\phi(i)$ é erro de estimação. Pode-se ver que a equação (3.15) tem um termo proporcional dado pela matriz G e um termo normalizante $\phi(i)^T G(i)\phi(i) + \xi$, a razão entre esses termos é que determina o ganho de atualização de cada parâmetros. A matriz diagonal

$$G(i) = diag[\ g_1(i) \quad g_2(i) \quad ... \quad g_N(i)]\ , \tag{3.16}$$

determina o ganho de cada parâmetros individual do estimadora. Os elementos dessa matriz são dados por

$$g_n(i) = \frac{q_n(i)}{\frac{1}{N}\sum_{i=1}^{N} q_n(i)}. \tag{3.17}$$

$$q_n(i) = \max\left\{\rho \times \left\{\max\left[\ \delta, \quad \|\theta\|_\infty\ \right]\right\}, |\theta_n(i)|\right\}, \tag{3.18}$$

sendo δ e ρ parâmetros positivos pequenos que desempenham o papel de regularização. O δ é um parâmetro de arranque do algoritmo quando ele é iniciado com θ igual a zero e ρ impede os parâmetros de pequena amplitude estagnarem.

Característica e virtudes do PNLMS:

- É utilizado principalmente para planta com grau de esparsidade alto;

- Os parâmetros são atualizados com ganhos ajustados de forma adaptativa e atribuídos aos parâmetros individualmente;
- Os ganhos são proporcional à magnitude dos parâmetros, exceto para os com $|\theta_n(i)| < \rho \times \left\{\max\left[\ \delta, \quad \|\theta\|_\infty\ \right]\right\}$.
- Velocidade de convergência maior comparado aos LMS e NLMS;
- Aumento de custo computacional em relação ao LMS e NLMS;
- Moderada complexidade computacional.

3.6.3 Algoritmo PNLMS Melhorado (IPNLMS)

O IPNLMS é um algorítimo derivado do PNLMS, mas difere deste na forma de calcular o ganho individual na atualização dos parâmetros. Detalhes de sua implementação pode ser encontrada em (BENESTY; GAY, 2002).

Neste caso, o novo ganho individual fica

$$g_n(i) = \frac{1-\alpha}{2} + \frac{N(1+\alpha)\,|\theta_n(i)|}{\|\theta(i)\|_1 + \kappa}, \tag{3.19}$$

e é encontrado considerando $\rho = 1/N$ e substituindo a norma infinita dos parâmetros $\|\theta\|_\infty$ pela norma da soma $\|\theta(n)\|_1$ na equação (3.17) que dá

$$q_n(i) = \max\left\{\frac{\|\theta\|_1}{N}, |\theta_n(i)|\right\}, \tag{3.20}$$

em seguida, a função max é substituída pela média ponderada de seus elementos, tomando a forma

$$q_n(i) = (1-\alpha)\,\frac{\|\theta(i)\|_1}{N} + (1+\alpha)\,|\theta_n(i)|\,, \tag{3.21}$$

sendo α um termo de ponderação que atribui crédito mais à soma dos parâmetros ou ao parâmetro individual. Por fim, a equação (3.21) é substituída na equação (3.17) para produzir a equação (3.19).

Resumindo, a equação de atualização fica

$$\theta(i+1) = \theta(i) + \frac{\mu G(i) e(i) \phi(i)}{\phi(i)^T G(i) \phi(i) + \xi}, \tag{3.22}$$

sendo

$$e(i) = d(i) - \theta(i)^T \phi(i) \tag{3.23}$$

e

$$G(i) = diag[\ g_1(i) \quad g_2(i) \quad ... \quad g_N(i)]\ , \tag{3.24}$$

com g_n dado pela equação (3.17).

Características do algoritmo:

- Engloba ambientes de média e alta esparsidade;
- Velocidade de convergência inicial é menor que o PNLMS para ambiente com alto grau de esparsidade;
- Tem um comportamento semelhante ao NLMS quando α está próximo de -1 e semelhante ao PNLMS quando α está próximo de 1.

3.6.4 Algoritmo PNLMS com Fator de Ativação Individual (IAF-PNLMS)

O algoritmo IAF-PNLMS é uma outra abordagem baseada no PNLMS ou na filosofia adaptativo proporcional. Surgiu no trabalho (SOUZA et al., 2010) (SOUZA, 2012) e é o primeiro a adotar um ganho diferenciado a cada parâmetros. Tem como melhorias, em relação ao PNLMS, uma maior velocidade de convergência para ambientes com alto grau de esparsidade e uma fácil sintonia, pois agora há apenas um parâmetros livre para ajuste contra três do PNLMS.

Como foi visto, o PNLMS viola o principio de ganho proporcional aos parâmetros quando a desigualdade $|\theta_n(i)| < \rho \times \left\{\max\left[\ \delta,\quad \|\theta\|_\infty\ \right]\right\}$ é satisfeita. O IAF-PNLMS, por outro lado, tende a não violação do princípio da proporcionalidade.

A equação de atualização dos parâmetros do IAF-PNLMS é igual ao PNLMS

$$\theta(i+1) = \theta(i) + \frac{\mu G(i)}{\phi(i)^T G(i)\phi(i) + \xi} e(i)\phi(i), \tag{3.25}$$

sendo

$$e(i) = d(i) - \theta(i)^T \phi(i). \tag{3.26}$$

A matriz de ganho

$$G(i) = diag[\ g_1(i) \quad g_2(i) \quad ... \quad g_N(i)]\ , \tag{3.27}$$

continua sendo uma matriz diagonal, mas difere em relação ao PNLMS quanto ao calculo dos seus elementos. O ganho individual, a função de proporcionalidade e o fator de ativação são dados, respectivamente, por (SOUZA et al., 2010)

$$g_n(i) = \frac{q_n(i)}{\frac{1}{N}\|\phi(i)\|_1}. \tag{3.28}$$

$$q_n(i) = \max\left(a_n,\ |\theta_n(i)|\right). \tag{3.29}$$

$$a_n = \begin{cases} \frac{|\theta_n|+q_n(i-1)}{2}, \ i = mN, \ m = 1, 2, 3, ... \\ a_n(i-1), \qquad\qquad outros\, casos \end{cases} . \tag{3.30}$$

Características do algoritmo:

- Tende a manter o princípio da proporcionalidade tanto para parâmetros de baixa magnitude quanto de alta;
- Tem uma taxa de convergência alta para ambiente muito esparso;
- Tem apenas um parâmetro livre que é a taxa de aprendizagem;
- Tem moderada complexidade computacional, mas inferior ao PNLMS;
- A velocidade de convergência dos parâmetros de baixa magnitude degrada quando o algoritmo está próximo a fase estacionária.

3.7 Complexidade Computacional

Esta seção, faz-se uma comparação da complexidade computacional dos algoritmos tratados neste capítulo. A complexidade computacional é abordada em termos de números de operações de adição, multiplicação, divisão e comparação efetuadas durante um iteração completa dos algoritmos. Essas operações estão em função da quantidade de parâmetros em estimação (N).

Como exemplo, o calculo da complexidade computacional do LMS é mostrado abaixo.

- Quantidade de adições:
  - Para atualização de um parâmetro, equação (3.10), é necessária uma adição, logo para N parâmetros são N adições;
  - Para calcular o erro são necessárias N adições: $N-1$ para o calculo de $\theta(i)^T\phi(i)$ e 1 para $d(i) - \phi(i)^T\theta(i)$;
  - Total de Adições $2N$;
- Quantidade de multiplicações:
  - Uma multiplicação ($b = \mu e(n)$), mais N multiplicações ($b\ \Phi$) totalizando $N+1$ multiplicações;
  - Para calcular o erro são necessárias N multiplicações $\theta(i)^T\phi(i)$;
  - Total de multiplicações $2N + 1$.
- Não há divisões e nem comparações;

- Os parâmetros a estimar são armazenados apenas uma vez em cada iteração, logo é preciso de uma memória de tamanho N.

Tabela 3.1: Complexidade Computacional em uma iteração completa de cada algoritmo. N é a dimensão do regressor

Comparação de Complexidade Computacional e Memória					
	Adição	Multiplicação	Divisão	Comparação	Memória
RLS	$2N^2 + 2N$	$2N^2 + 3N$	$N^2 + N$	0	$N^2 + 2N$
LMS	$2N$	$2N + 1$	0	0	N
NLMS	$3N$	$3N + 1$	1	0	N
PNLMS	$4N - 1$	$4N + 3$	1	$2N$	N
IPNLMS	$5N + 3$	$5N + 3$	2	0	N
IAFPNLMS	$4N$	$4N + 3$	1	N	$3N$

Comparando a complexidade computacional dos algoritmos, é possível concluir que o RLS tem a maior complexidade computacional O(N^2), seguindo pelos IPNLMS, PNLMS e IAFPNLMS de complexidade média O(N) e dos LMS e NLMS de baixa complexidade. Nota-se, assim, que os proporcionais adaptativos tem uma boa relação de compromisso entre complexidade computacional e desempenho.

4

Programação Dinâmica Adaptativa Aplicada ao DLQR

Sumário

A teoria da Programação Dinâmica Adaptativa é aplicada, neste trabalho, ao projeto do controle ótimo baseado no DLQR cujas parametrizações são empregadas para a função de custo e para a ação de controle, para projetar um sistema que soluciona, com dados sensoriais dos estado e custo instantâneo, a Equação Algébrica de Riccati Discreta (DARE), ou seja, a equação HJB-DARE é resolvido implicitamente ao contrário do projeto do DLQR que impõe a sua resolução explicitamente e, obviamente, com o conhecimento do sistema.

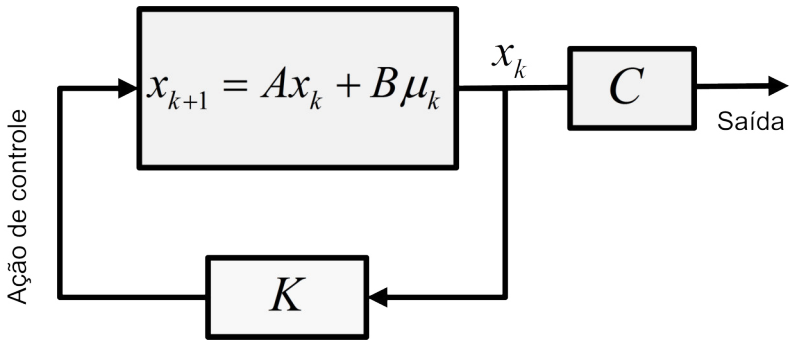
4.1 Caracterização do Problema

Nesta seção, um resumo de controle ótimo baseado no Regulador Linear Quadrático Discreto (DLQR) é apresentado, considerando que a solução proposta é fundamentada nesse tipo de regulador. A seguir, trata-se da descrição do problema e de sua formulação.

4.1.1 Regulador Linear Quadrático Discreto

O projeto do regulador linear quadrático discreto é baseados no método de estabilidade de Lyaponov, dessa forma, a sua estabilidade é garantida. Ele tem a estrutura, em diagrama de blocos, mostrada na figura (4.1).

Figura 4.1: Regulador Linear Quadrático Discreto



onde K é o ganho de retroação de estado e o seu valor, que é ótimo para qualquer condição inicial do sistema, é o objetivo final do projeto do DLQR.

O projeto do DLQR *offline* requer o conhecimento prévio do processo e objetiva encontrar a ação de controle μ que minimiza um índice de desempenho quadrático ou função de custo dado por

$$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + \mu_k^T R \mu_k \right), \tag{4.1}$$

ou seja, é um problema de otimização dado pela seguinte estrutura:

$$\min_u J, \tag{4.2}$$

sujeito a

$$x_{k+1} = Ax_k + B\mu_k, \tag{4.3}$$

onde $A \in \Re^{n\times n}$ e $B \in \Re^{n\times m}$ são matrizes do sistema controlável, ou seja, sistema cujo posto da matriz

$$\left[\begin{array}{cccccc} A^{n-1}B & A^{n-2}B & ,..., & AB & B \end{array}\right] \tag{4.4}$$

é igual a n, x_k é um vetor de estado, μ é a ação de controle não restrita e Q é uma matriz simétrica definida positiva ou semidefinida positiva e R é uma matriz definida positiva simétrica. As matrizes Q e R determinam a importância relativa dos erros e do gasto de energia.

Pode-se ver que a função de custo instantânea

$$r(x_k, \mu_k) = x_k^T Q x_k + \mu_k^T R \mu_k, \tag{4.5}$$

informa o custo de tomar a ação μ_k no estado x_k.

A solução da estrutura de otimização definida pelas Equações (4.2) e (4.3) conduz ao valor de custo ótimo do Regulador Linear Quadrático que é

$$J^* \quad = \quad x_k^T P x_k. \tag{4.6}$$

O valor ótimo da Equação (4.6) conduz a uma ação de controle linear nos estados

$$\mu_k = K(P)x_k, \tag{4.7}$$

que de acordo com (LEWIS; VRABIE, 2009) é dado por

$$K(P_k) = -\left(R + B^T P_k B\right)^{-1} B^T P_k A. \tag{4.8}$$

Projeto de sistema de controle ótimo *online* necessita alterar os ganhos da ação de controle inicial adotada, provavelmente não ótima, de forma a conduzir a uma ação de controle ótima. Neste caso é necessário calcular a solução da equação de Riccati em cada instante de amostragem, gerando uma alta carga computacional que pode inviabilizar a implementação prática da solução obtida.

4.1.2 Descrição do Problema

O problema a ser resolvido requer encontrar uma sequencia de solução da equação de Riccati. Esta equação para o projeto do DLQR é dada por

$$P_{k+1} \quad = \quad A^T P_k A - (A^T P_k B) W_k^{-1} (B^T P_k A) + Q, \tag{4.9}$$

onde W_k é uma função de ponderação da matriz de entrada dada por

$$W_k = (R + B^T P_k B), \tag{4.10}$$

B é a matriz de entrada de controle e P_k, com $k \longrightarrow \infty$, é a solução da DARE.

O que se objetiva é encontrar a solução da equação HJB-DARE sem o conhecimento das matrizes A e B da equação (4.3), ou seja, a equação HJB-DARE deve ser resolvido sem conhecimento do modelo do sistema dinâmico.

No contexto da programação dinâmica adaptativa e projeto de sistema de controle DLQR, o problema é formulado no sentido de estimar a solução da equação (4.9) com base nas medições das interações agente-ambiente e mecanismo de decisão, conforme mostra a Figura (1.1).

4.2 Solução Proposta

A solução que foi desenvolvida para obter a política ótima em cada instante de amostragem é baseada na programação dinâmica adaptativa ou críticos adaptativos (HDP, DHP, ADHDP e ADDHP) juntamente com as parametrizações induzidas pelo DLQR, usando os algoritmos baseados no LMS (NLMS, PNLMS, IPNLMS e AIFPNLMS) e RLS para estimar os parâmetros da função de custo (4.1).

4.2.1 Críticos Adaptativos para o DLQR

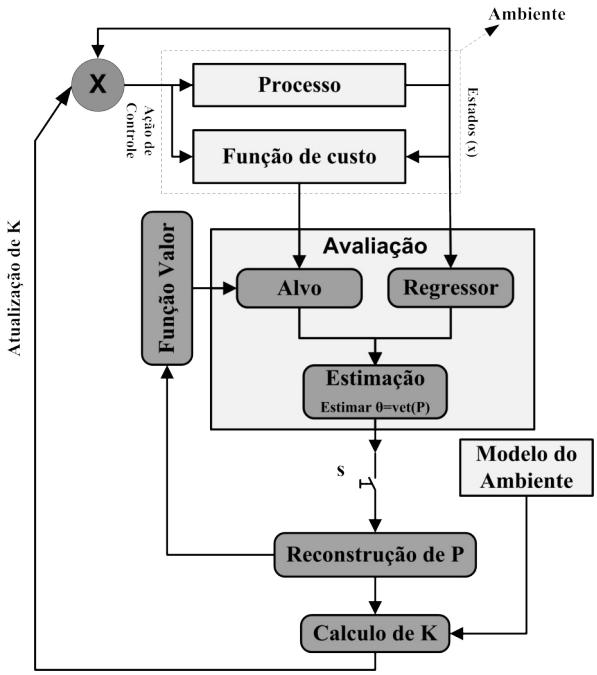
Para encontrar uma política ótima de um problema de decisão sequencial na forma de um PDM utilizando os algoritmos HDP, DHP, ADHDP e ADDHP é necessário conhecer uma função valor, um alvo e como melhorar a política de forma que a função valor tenda para a função valor ótima e o alvo tenda para o valor verdadeiro. A forma geral de calcular a função valor, o alvo e a ação para cada algoritmo critico adaptativo foi visto no capítulo 2.

Em geral, a política parametrizada utilizada nos algoritmos críticos adaptativos apenas representa a política ótima aproximadamente por não se conhecer a sua estrutura real, mas para o caso especial do DLQR, a política parametrizada representa exatamente a política ótima que é linear na variável de estado. Neste capítulo, os algoritmos críticos adaptativos são particularizados utilizando as parametrizações do DLQR. Nesta seção, as estruturas esquemáticas em diagrama de blocos de cada um dos algoritmos críticos adaptativos são mostradas.

4.2.1.1 Estrutura Esquemática do HDP

O diagrama de blocos da figura (4.2) exibe a representação esquemática do algoritmo de controle ótimo adaptativo para o caso HDP-DLQR.

Figura 4.2: Diagrama de blocos do algoritmo HDP-DLQR

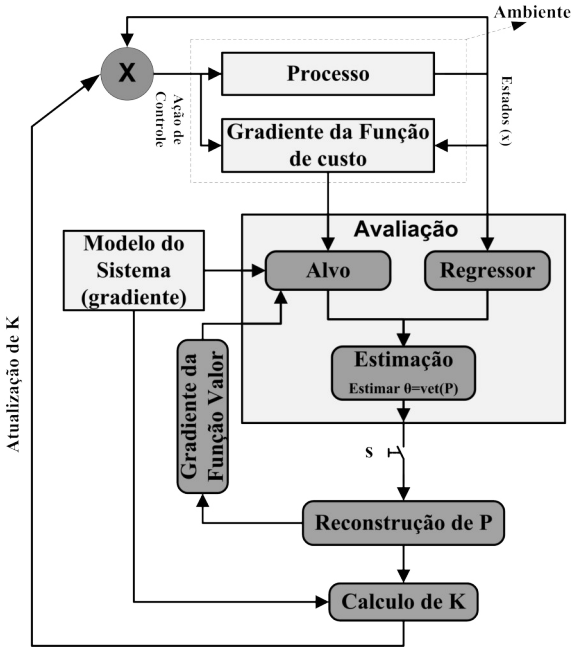


Pode-se ver no diagrama de blocos que o algoritmo de iteração de política foi utilizado para encontrar uma política ótima. Inicialmente, é fornecida uma política que é avaliada no bloco chamado "Avaliação". É nesse bloco que empregamos os algoritmos baseado no LMS para estimar o valor dos parâmetros θ ou P da função-V parametrizada. Após a avaliação a chave "**S**" é fechada para efetuar a melhoria da política gerando nova política. Essas processo iterativo de avaliar e melhorar a política continua até que a política atual convirja para a política ótima. Como foi visto, na etapa de melhoria da política o HDP é altamente dependente do modelo do ambiente e que o alvo não é dependente do modelo ambiente.

4.2.1.2 Estrutura Esquemática do DHP

A figura (4.3) exibe a representação esquemática em diagrama de blocos do algoritmo DHP-DLQR.

Figura 4.3: Diagrama de blocos do algoritmo DHP-DLQR

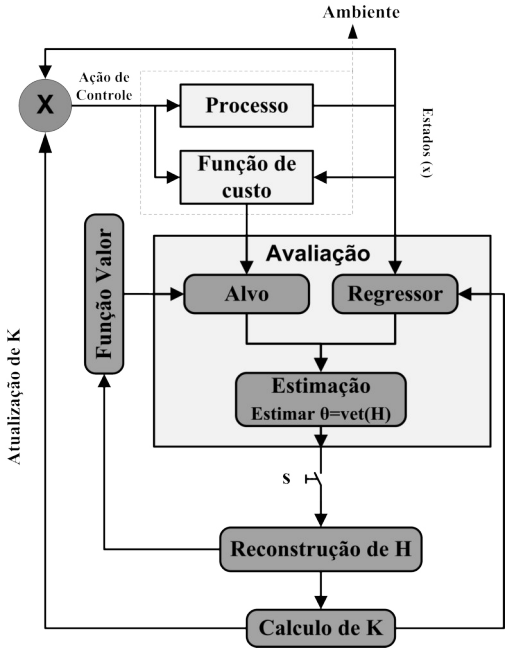


A sua interpretação é semelhante ao caso HDP. Nota-se que agora se estima o gradiente da função-V e não a própria função-V, além disso, as etapas de avaliação e melhoria da política de controle são dependente do modelo do sistema, diferentemente do caso HDP que somente a melhoria da política é dependente do modelo.

4.2.1.3 Estrutura Esquemática do ADHDP

O diagrama de blocos da figura (4.4) exibe a representação esquemática do algoritmo de aprendizagem adaptativa para o caso do ADHDP-DLQR.

Figura 4.4: Diagrama de blocos do algoritmo ADHDP-DLQR

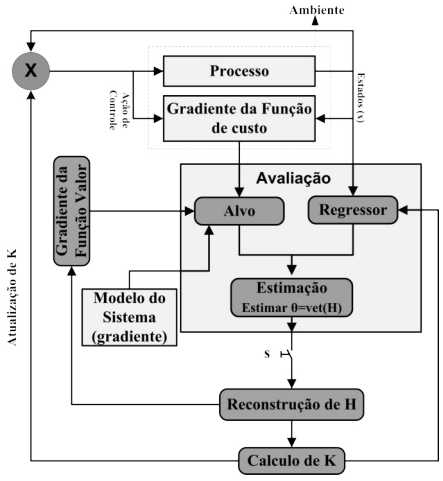


O diagrama mostra que o regressor agora depende tanto dos estados quanto das ações correntes e a construção do alvo é independente do modelo do processo. Além disso, a forma de calcular K no bloco "Calculo de K"depende apenas da matriz H estimada no bloco "Avaliação". Nota-se que o algoritmo ADHDP não depende do ambiente nem na fase de atualização da política e nem na fase de avaliação.

4.2.1.4 Estrutura Esquemática do ADDHP

O diagrama de blocos (4.5) exibe a representação esquemática do algoritmo ADDHP-DLQR.

Figura 4.5: Diagrama de blocos do algoritmo ADDHP-DLQR



Percebe-se, como no caso do DHP-DLQR, que este algoritmo depende da estimação

do gradiente da função valor que neste caso é a função-Q e que, por sua vez, depende do modelo do ambiente. Por outro lado, semelhantemente ao algoritmo ADHDP, ele é independente do modelo do processo na fase de melhoria da política.

4.3 Elementos dos Críticos Adaptativos

Mostra-se nesta seção como construir os elementos principais dos CA, ou seja, com gerar as funções valores (crítico), as ações de controle (ator) e os alvos necessários no processo de aproximação paramétrica particularizados para o problema do DLQR.

4.3.1 Funções Valor

Mostra-se abaixo como construir as função valor estado ou função-V e a função valor ação ou função-Q utilizando a parametrização dada pelo problema do DLQR.

4.3.1.1 Função-V para DLQR

Substituindo-se na função-V dada pela equação (2.7) a função de custo instantânea dada pela equação (4.5) e considerando o sistema determinístico, tem-se

$$
\begin{aligned}
V(x_k) &= \sum_{t=k}^{\infty} \gamma^{t-k} \left(x_t^T Q x_t + \mu_t^T R \mu_t \right) && (4.11)\\
&= \sum_{t=k}^{\infty} \gamma^{t-k} \left(x_t^T Q x_t + \left[K x_t \right]^T R K x_t \right) && (4.12)\\
&= \sum_{j=0}^{\infty} \gamma^{j} \left(x_{j+k}^T Q x_{j+k} + x_{j+k}^T K^T R K x_{j+k} \right) && (4.13)\\
&= \sum_{j=0}^{\infty} \gamma^{j} x_{j+k}^T \left(Q + K^T R K \right) x_{j+k}, && (4.14)
\end{aligned}
$$

sendo

$$
\begin{aligned}
x_{j+k} &= A x_{j+k-1} + B \mu_{j+k-1} && (4.15)\\
&= \left(A + BK \right) x_{j+k-1} && (4.16)\\
&= \left(A + BK \right)^2 x_{j+k-2} && (4.17)\\
&= \left(A + BK \right)^j x_k, && (4.18)
\end{aligned}
$$

substituindo-se este último resultado em (4.14) fica

$$V(x_k) = \sum_{j=0}^{\infty} \gamma^j \left((A+BK)^j x_k\right)^T \left(Q + K^T R K\right) \left((A+BK)^j x_k\right) \tag{4.19}$$

$$= \sum_{j=0}^{\infty} \gamma^j x_k^T \left((A+BK)^j\right)^T \left(Q + K^T R K\right) \left((A+BK)^j\right) x_k \tag{4.20}$$

$$= x_k^T \left[\sum_{j=0}^{\infty} \gamma^j \left((A+BK)^j\right)^T \left(Q + K^T R K\right) (A+BK)^j\right] x_k, \tag{4.21}$$

chamando

$$P = \sum_{j=0}^{\infty} \gamma^j \left((A+BK)^j\right)^T \left(Q + K^T R K\right) (A+BK)^j, \tag{4.22}$$

uma matriz $n \times n$, a equação (4.21) fica

$$V(x_k) = x_k^T P x_k \tag{4.23}$$

$$= x_k^T \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & \dots & \vdots \\ p_{n1} & p_{n2} & \dots & p_{nn} \end{bmatrix} \begin{bmatrix} x_1^k \\ x_2^k \\ \vdots \\ x_n^k \end{bmatrix} \tag{4.24}$$

$$= x_k^T \begin{bmatrix} p_1 & p_2 & \dots & p_n \end{bmatrix} \begin{bmatrix} x_1^k \\ x_2^k \\ \vdots \\ x_n^k \end{bmatrix} \tag{4.25}$$

$$= x_k^T p_1 x_1^k + x_k^T p_2 x_2^k + ... + x_k^T p_n x_n^k \tag{4.26}$$

$$= \begin{bmatrix} x_k^T x_1^k & x_k^T x_2^k & \dots & x_k^T x_n^k \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \tag{4.27}$$

$$= vet\left(x_k x_k^T\right)^T vet\left(P\right) = \left(x_k^T \otimes x_k^T\right) vet\left(P\right) = \phi^T \theta, \tag{4.28}$$

sendo

$$x_k = \begin{bmatrix} x_1^k & x_2^k & ... & x_n^k \end{bmatrix}^T \tag{4.29}$$

$$p_i = \begin{bmatrix} p_{1i} & p_{2i} & ... & p_{ni} \end{bmatrix}^T \tag{4.30}$$

$$\theta = vet\left(P\right) = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} & p_{22} & \dots & p_{2n} & \dots & p_{nn} \end{bmatrix} \tag{4.31}$$

$$\phi^T = \begin{bmatrix} x_1^k x_1^k & x_1^k x_2^k & ... & x_1^k x_n^k & x_2^k x_2^k & x_2^k x_3^k & ... & x_{n-1}^k x_{n-1}^k & x_{n-1}^k x_n^k & x_n^k x_n^k. \end{bmatrix} \tag{4.32}$$

A equação (4.28) é adotada como a função-V quando se está usando a parametrização

induzida pelo problema do DLQR e HDP. A matriz P é definida positiva e simétrica por esse motivo na sua vetorização os elementos repetidos são descartados e tornando o vetor ϕ com apenas $n(n+1)/2$ elementos que é igual a quantidade de parâmetros a estimar.

No caso do DHP, os parâmetros da derivada parcial da função-V em relação aos estados é que são estimados, ou seja,

$$\begin{aligned}\frac{\partial V(x_k)}{\partial x_k} &= 2x_k^T P && (4.33)\\ &= 2(x_k^T PI) && (4.34)\\ &= 2\left(I \otimes x_k^T\right) vet(P) && (4.35)\\ &= \phi^T \theta. && (4.36)\end{aligned}$$

Uma forma de construir a matriz de regressores para o DHP é através da seguinte regra:

- Cada linha da matriz de regressores é formada pelo vetor linha

$$\begin{aligned}\phi_l^T = \; 2 \; \Big[\; 0_{1\times(l-1)} \quad x_1 \quad ... \quad 0_{1\times\left[(n-s)\left\lceil\frac{l-(s-1)}{l}\right\rceil\right]} \quad x_s \quad ... \quad 0_{1\times\left[\left\lceil\frac{l-(n-2)}{l}\right\rceil\right]} \\ x_{n-1} \quad x_n \quad ... \; \Big], \end{aligned} \tag{4.37}$$

  sendo s a posição do estado no vetor de estados, l o número da linha, $0_{1\times q}$ vetor linha de q zeros e o símbolo $\lceil y \rceil$ indica o menor inteiro não negativo maior que y, por exemplo, $\lceil -4 \rceil = 0$ e $\lceil 0,2 \rceil = 1$;

- Caso q em $0_{1\times q}$ for menor ou igual a zero significa que esse vetor não existe e o seu lugar é ocupado pelo elemento imediatamente posterior;

- A reticencia final indica que se deve completar com zeros os últimos elementos do vetor linha de forma que ele tenha $n(n+1)/2$ elementos.

Exemplo:
Seja o vetor x formado por quatro estado, ou seja, $x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$, tem-se:

- Dimensão do regressor ϕ^T: 4 Linhas e $4(4+1)/2 = 10$ colunas;

- A primeira linha é formada substituindo-se $l = 1$ e $n = 4$ em (4.37), ou seja,

$$\begin{aligned}\phi_1^T &= 2 \; \Big[\; 0_{1\times(1-1)} \quad x_1 \quad 0_{1\times\left[(4-2)\left\lceil\frac{1-1}{1}\right\rceil\right]} \quad x_2 \quad 0_{1\times\left[(4-3)\left\lceil\frac{1-2}{1}\right\rceil\right]} \quad x_3 \quad x_4 \quad ... \; \Big] \\ &= 2 \; \Big[\; 0_{1\times 0} \quad x_1 \quad 0_{1\times 0} \quad x_2 \quad 0_{1\times 0} \quad x_3 \quad 0_{1\times 0} \quad x_4 \quad ... \; \Big] \\ &= 2 \; \Big[\; x_1 \quad x_2 \quad x_3 \quad x_4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \; \Big]\end{aligned}$$

- A segunda linha é formada substituindo-se $l = 2$ e $n = 4$ em (4.37):

$$\begin{aligned}
\phi_2^T = \; & 2 \; \left[\; 0_{1\times(2-1)} \quad x_1 \quad 0_{1\times\left[(4-2)\left\lceil\frac{2-1}{2}\right\rceil\right]} \quad x_2 \quad 0_{1\times\left[(4-3)\left\lceil\frac{2-2}{2}\right\rceil\right]} \quad x_3 \quad x_4 \quad ... \; \right] \\
= \; & 2 \; \left[\; 0_{1\times1} \quad x_1 \quad 0_{1\times2} \quad x_2 \quad 0_{1\times0} \quad x_3 \quad 0_{1\times0} \quad x_4 \quad ... \; \right] \\
= \; & 2 \; \left[\; 0 \quad x_1 \quad 0 \quad 0 \quad x_2 \quad x_3 \quad x_4 \quad 0 \quad 0 \quad 0 \; \right]
\end{aligned}$$

- A terceira linha é formada substituindo-se $l = 3$ e $n = 4$ em (4.37):

$$\begin{aligned}
\phi_3^T = \; & 2 \; \left[\; 0_{1\times(3-1)} \quad x_1 \quad 0_{1\times\left[(4-2)\left\lceil\frac{3-1}{3}\right\rceil\right]} \quad x_2 \quad 0_{1\times\left[(4-3)\left\lceil\frac{3-2}{3}\right\rceil\right]} \quad x_3 \quad x_4 \quad ... \; \right] \\
= \; & 2 \; \left[\; 0_{1\times2} \quad x_1 \quad 0_{1\times2} \quad x_2 \quad 0_{1\times1} \quad x_3 \quad 0_{1\times0} \quad x_4 \quad ... \; \right] \\
= \; & 2 \; \left[\; 0 \quad 0 \quad x_1 \quad 0 \quad 0 \quad x_2 \quad 0 \quad x_3 \quad x_4 \quad 0 \; \right]
\end{aligned}$$

- A quarta e última linha é formada substituindo-se $l = 4$ e $n = 4$ em (4.37):

$$\begin{aligned}
\phi_4^T = \; & 2 \; \left[\; 0_{1\times(4-1)} \quad x_1 \quad 0_{1\times\left[(4-2)\left\lceil\frac{4-1}{4}\right\rceil\right]} \quad x_2 \quad 0_{1\times\left[(4-3)\left\lceil\frac{4-2}{4}\right\rceil\right]} \quad x_3 \quad x_4 \quad ... \; \right] \\
= \; & 2 \; \left[\; 0_{1\times3} \quad x_1 \quad 0_{1\times2} \quad x_2 \quad 0_{1\times1} \quad x_3 \quad 0_{1\times0} \quad x_4 \quad ... \; \right] \\
= \; & 2 \; \left[\; 0 \quad 0 \quad 0 \quad x_1 \quad 0 \quad 0 \quad x_2 \quad 0 \quad x_3 \quad x_4 \; \right]
\end{aligned}$$

- a matriz de regressores fica:

$$\phi^T = 2 \begin{bmatrix} \phi_1^T \\ \phi_2^T \\ \phi_3^T \\ \phi_4^T \end{bmatrix} = 2 \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_1 & 0 & 0 & x_2 & x_3 & x_4 & 0 & 0 & 0 \\ 0 & 0 & x_1 & 0 & 0 & x_2 & 0 & x_3 & x_4 & 0 \\ 0 & 0 & 0 & x_1 & 0 & 0 & x_2 & 0 & x_3 & x_4 \end{bmatrix} \tag{4.38}$$

4.3.1.2 Função-Q para DLQR

Utilizando a parametrização do DLQR, a função-Q fica:

$$
\begin{aligned}
Q\left(x_k, \mu_k\right) &= r\left(x_k, \mu_k\right) + V(x_{k+1}) && (4.39)\\
&= x_k^T Q x_k + \mu_k^T R \mu_k + x_{k+1}^T P x_{k+1} && (4.40)\\
&= \begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix} + \left(Ax_k + B\mu_k\right)^T P \left(Ax_k + B\mu_k\right) && (4.41)\\
&= \begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix} + \begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \begin{bmatrix} A^T \\ B^T \end{bmatrix} P \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix} && (4.42)\\
&= \begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \left(\begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} + \begin{bmatrix} A^T \\ B^T \end{bmatrix} P \begin{bmatrix} A & B \end{bmatrix} \right) \begin{bmatrix} x_k \\ \mu_k \end{bmatrix} && (4.43)\\
&= \begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \begin{bmatrix} Q + A^T P A & B^T P A \\ A^T P B & R + B^T P B \end{bmatrix} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix} && (4.44)\\
&= \begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \begin{bmatrix} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{bmatrix} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}, && (4.45)
\end{aligned}
$$

fazendo as substituições

$$
\begin{aligned}
H &= \begin{bmatrix} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{bmatrix} && (4.46)\\
z_k &= \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}, && (4.47)
\end{aligned}
$$

na última equação, tem-se

$$
\begin{aligned}
Q\left(x_k, \mu_k\right) &= z_k^T H z_k && (4.48)\\
&= vet\left(z_k z_k^T\right)^T vet\left(H\right) && (4.49)\\
&= \left(z_k^T \otimes z_k^T\right) vet\left(H\right) && (4.50)\\
&= \phi^T \theta. && (4.51)
\end{aligned}
$$

A forma de construir a matriz de regressores ϕ e de vetorizar H são iguais às vistas para função-V, mas utilizando z no lugar de x. A matriz H é simétrica e $H_{\mu x} = H_{x\mu}^T$.

A equação (4.50) é adotada como a função-Q quando se está usando a parametrização induzida pelo problema do DLQR e ADHDP.

Para o caso do ADDHP, deve-se estimar a derivada parcial da função-Q em relação

aos estados e ações, ou seja,

$$\begin{aligned}
\frac{\partial Q\left(x_k,\mu_k\right)}{\partial z_k} &= \left[\begin{array}{cc} \frac{\partial Q(x_k,\mu_k)}{\partial x_k} & \frac{\partial Q(x_k,\mu_k)}{\partial \mu_k} \end{array}\right] && (4.52)\\
&= \left[\begin{array}{cc} 2\left(x_k^T H_{xx} + \mu_k^T H_{\mu x}\right) & 2\left(x_k^T H_{x\mu} + \mu_k^T H_{\mu\mu}\right) \end{array}\right] && (4.53)\\
&= 2\left[\begin{array}{cc} x_k^T & \mu_k^T \end{array}\right]\left[\begin{array}{cc} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{array}\right] && (4.54)\\
&= 2z_k^T H && (4.55)\\
&= 2(I\otimes z_k^T)vet(H) && (4.56)\\
&= \phi^T\theta. && (4.57)
\end{aligned}$$

A forma de construir ϕ e θ é igual ao caso DHP, mas substituindo-se x por z.

4.3.2 Política de controle

Após cada atualização da função valor precisa-se melhorar a política. Para isso, é necessário encontrar um novo K de forma a minimizar

$$K = \arg\min_{K'}\left(r(x_k,\mu_k) + V\left(f\left(x_k,\mu_k(x_k,K'),P\right)\right)\right), \tag{4.58}$$

que é a mesma equação (2.29) com K no lugar de w e P no lugar de θ, esse procedimento é adotado para o HDP e DHP. Para o ADHDP e ADDHP deve-se minimizar

$$K = \arg\min_{K'}\left(Q\left(f\left(x_k,\mu_k(x_k,K'),H\right)\right)\right). \tag{4.59}$$

4.3.2.1 Ação de controle para HDP e DHP

No caso do HDP e DHP a ação de controle é encontrada resolvendo a equação (2.30) com as parametrizações dadas pelo problema DLQR.

$$\begin{aligned}
\frac{\partial \mu_k}{\partial K_k} &= x_k && (4.60)\\
\frac{\partial r}{\partial \mu_k} &= 2\mu_k^T R && (4.61)\\
\frac{\partial V}{\partial f} &= 2fP = 2x_{k+1}^T P = 2(Ax_k + B\mu_k)^T P = 2(x_k^T A^T P + \mu_k^T B^T P) && (4.62)\\
\frac{\partial f}{\partial \mu_k} &= B, && (4.63)
\end{aligned}$$

substituindo essas equações na equação (2.30) fica

$$\begin{aligned} 2\mu_k^T R + 2(x_k^T A^T P + \mu_k^T B^T P)B &= 0 && (4.64)\\ \mu_k^T R + x_k^T A^T PB + \mu_k^T B^T PB &= 0 && (4.65)\\ \mu_k^T \left(R + B^T PB\right) &= -x_k^T A^T PB && (4.66)\\ \left(R + B^T PB\right)\mu_k &= -B^T PAx_k && (4.67)\\ \mu_k &= -\left(R + B^T PB\right)^{-1} B^T PAx_k, && (4.68) \end{aligned}$$

portanto,

$$K = -\left(R + B^T PB\right)^{-1} B^T PA, \tag{4.69}$$

lembrando que R e P são matrizes simétricas por isso $R^T = R$ e $(B^T PB)^T = B^T P^T B = B^T PB$.

Pode-se ver pelo desenvolvimento das equações que é necessário ter o modelo da planta para encontrar a equações de melhoria da política, ou seja, o HDP e o DHP são dependentes do modelo do sistema na fase de melhoria da política.

4.3.2.2 Ação de controle para ADHDP e ADDHP

Neste caso, deve-se resolver a equação

$$\frac{\partial Q}{\partial \mu_k} = 0, \tag{4.70}$$

dessa forma tem-se

$$\begin{aligned} \frac{\partial Q}{\partial \mu_k} &= \frac{\partial\left(\begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix}\begin{bmatrix} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{bmatrix}\begin{bmatrix} x_k \\ \mu_k \end{bmatrix}\right)}{\partial \mu_k} = 2\left(x_k^T H_{x\mu} + \mu_k^T H_{\mu\mu}\right) && (4.71)\\ 0 &= x_k^T H_{\mu x} + \mu_k^T H_{\mu\mu} && (4.72)\\ \mu_k^T &= -x_k^T H_{x\mu} H_{\mu\mu}^{-1} && (4.73)\\ \mu_k &= -H_{\mu\mu}^{-1} H_{\mu x} x_k, && (4.74) \end{aligned}$$

sendo que $H_{\mu\mu}$ é simétrica e $H_{\mu x}^T = H_{\mu x}$. Portanto,

$$K = -H_{\mu\mu}^{-1} H_{\mu x}. \tag{4.75}$$

Pode-se observar que a ação de controle não depende do modelo do sistema e sim da estimativa de H.

4.3.3 Alvos para Avaliação da Política

4.3.3.1 Alvo para HDP

O alvo para o HDP é uma estimativa da função-V para o estado atual tomando como base a estimativa da função-V na última avaliação da política aplicado no estado seguinte, ou seja,

$$\begin{aligned} Y^{alvo} &= r(x_k, \mu_k) + V(f(x_k, \mu(x_k))) && (4.76)\\ &= x_k^T Q x_k + \mu_k^T R \mu_k + \phi_i^T \left(f(x_k, \mu(x_k))\right) \theta_{i-1} && (4.77)\\ &= r_{k+1} + \phi_i^T \left(x_{k+1}\right) \theta_{i-1}. && (4.78) \end{aligned}$$

Na aplicação *online* do HDP, deve-se esperar o custo $r(x_k, \mu_k)$ e o estado x_{k+1} provenientes da ação μ_k tomada no estado x_k para calcular o valor do alvo e atualizar θ por algum algoritmo de estimação de parâmetros. Vê-se que esse alvo não depende do modelo do sistema.

4.3.3.2 Alvo para DHP

O alvo para o DHP é uma estimativa da derivada da função-V para o estado atual tomando como base a estimativa da derivada da função-V na última avaliação da política aplicado no estado seguinte, ou seja,

$$Y^{alvo} = \frac{\partial V(x)}{\partial x_k} = \frac{\partial r}{\partial x_k} + \frac{\partial r}{\partial \mu_k}\frac{\partial \mu_k}{\partial x_k} + \frac{\partial V}{\partial f}\left(\frac{\partial f}{\partial x_k} + \frac{\partial f}{\partial \mu_k}\frac{\partial \mu_k}{\partial x_k}\right), \tag{4.79}$$

sendo

$$\begin{aligned} \frac{\partial r}{\partial x} &= 2x_k^T Q && (4.80)\\ \frac{\partial r}{\partial \mu_k} &= 2\mu_k^T R && (4.81)\\ \frac{\partial \mu_k}{\partial x_k} &= K && (4.82)\\ \frac{\partial V}{\partial f} &= 2(x_k^T A^T P + \mu_k^T B^T P) && (4.83)\\ \frac{\partial f}{\partial x_k} &= A && (4.84)\\ \frac{\partial f}{\partial \mu_k} &= B, && (4.85) \end{aligned}$$

substituindo essas equações na equação (4.79), tem-se

$$Y^{alvo} = 2x_k^T Q + 2\mu_k^T RK + 2\left(x_k^T A^T P + \mu_k^T B^T P\right)(A + BK) \tag{4.86}$$
$$= 2\left(x_k^T Q + \mu_k^T RK + \left(x_k^T A^T + \mu_k^T B^T\right) P\,(A + BK)\right) \tag{4.87}$$
$$= 2\left(x_k^T Q + \mu_k^T RK + (Ax_k + B\mu_k)^T P\,(A + BK)\right), \tag{4.88}$$

neste caso o alvo é um vetor de ordem $1 \times n$ onde n é o número de estados do sistema.

Fica evidente a partir da equação (4.88) que o alvo é dependente do modelo do ambiente.

4.3.3.3 Alvo para ADHDP

O alvo para o ADHDP é uma estimativa da função-Q para o estado atual tomando como base a estimativa da função-Q da última avaliação da política aplicado no estado seguinte, ou seja,

$$Y^{alvo} = r(x_k, \mu_k) + Q(x_{k+1}, \mu_{k+1}(x_{k+1})) \tag{4.89}$$
$$= \; r_{k+1} + \phi_i^T\,(x_{k+1}, \mu_{k+1})\,\theta_{i-1}, \tag{4.90}$$
$$\tag{4.91}$$

Esse alvo não depende do modelo do ambiente, mas apenas das amostras dos estados e custos instantâneo.

4.3.3.4 Alvo para ADDHP

O alvo para o ADDHP é uma estimativa da derivada da função-Q para o estado e ação atuais tomando como base a estimativa da derivada da função-Q na última avaliação da política aplicado no estado e ação seguinte, ou seja,

$$Y^{alvo} = \begin{bmatrix} \frac{\partial Q(x_k,\mu_k)}{\partial x_k} & \frac{\partial Q(x_k,\mu_k)}{\partial \mu_k} \end{bmatrix} \tag{4.92}$$
$$= \begin{bmatrix} \frac{\partial r(x_k,\mu_k)}{\partial x_k} \\ \frac{\partial r(x_k,\mu_k)}{\partial \mu_k} \end{bmatrix}^T + \begin{bmatrix} \frac{\partial Q(x_{k+1},\mu_{k+1})}{\partial x_k} \\ \frac{\partial Q(x_{k+1},\mu_{k+1})}{\partial \mu_k} \end{bmatrix}^T \tag{4.93}$$
$$= \begin{bmatrix} \frac{\partial r(x_k,\mu_k)}{\partial x_k} \\ \frac{\partial r(x_k,\mu_k)}{\partial \mu_k} \end{bmatrix}^T +$$
$$+ \begin{bmatrix} \frac{\partial Q(x_{k+1},\mu_{k+1})}{\partial x_{k+1}}\frac{\partial x_{k+1}}{\partial x_k} + \frac{\partial Q(x_{k+1},\mu_{k+1})}{\partial \mu_{k+1}}\frac{\partial \mu_{k+1}}{\partial x_{k+1}}\frac{\partial x_{k+1}}{\partial x_k} \\ \frac{\partial Q(x_{k+1},\mu_{k+1})}{\partial x_{k+1}}\frac{\partial x_{k+1}}{\partial \mu_k} + \frac{\partial Q(x_{k+1},\mu_{k+1})}{\partial \mu_{k+1}}\frac{\partial \mu_{k+1}}{\partial x_{k+1}}\frac{\partial x_{k+1}}{\partial \mu_k} \end{bmatrix}^T, \tag{4.94}$$

sendo

$$\frac{\partial r\left(x_k,\mu_k\right)}{\partial x_k} = 2x_k^T Q \tag{4.95}$$

$$\frac{\partial r\left(x_k,\mu_k\right)}{\partial \mu_k} = 2\mu_k^T R \tag{4.96}$$

$$\frac{\partial Q\left(x_{k+1},\mu_{k+1}\right)}{\partial x_{k+1}} = x_{k+1}^T H_{xx} + \mu_{k+1}^T H_{\mu x} \tag{4.97}$$

$$\frac{\partial x_{k+1}}{\partial x_k} = A \tag{4.98}$$

$$\frac{\partial Q\left(x_{k+1},\mu_{k+1}\right)}{\partial \mu_{k+1}} = x_{k+1}^T H_{x\mu} + \mu_{k+1}^T H_{\mu\mu} \tag{4.99}$$

$$\frac{\partial \mu_{k+1}}{\partial x_{k+1}} = K \tag{4.100}$$

$$\frac{\partial x_{k+1}}{\partial \mu_k} = B, \tag{4.101}$$

$$\tag{4.102}$$

substituindo essas equações na equação (4.94), tem-se

$$\begin{aligned}
Y^{alvo} &= 2\begin{bmatrix} x_k^T Q \\ \mu_k^T R \end{bmatrix}^T + \\
&+ 2\begin{bmatrix} \left(x_{k+1}^T H_{xx} + \mu_{k+1}^T H_{\mu x}\right) A + \left(x_{k+1}^T H_{x\mu} + \mu_{k+1}^T H_{\mu\mu}\right) KA \\ \left(x_{k+1}^T H_{xx} + \mu_{k+1}^T H_{\mu x}\right) B + \left(x_{k+1}^T H_{x\mu} + \mu_{k+1}^T H_{\mu\mu}\right) KB \end{bmatrix}^T \qquad (4.103)\\
&= 2\begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix}\begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} + \\
&+ 2\begin{bmatrix} x_{k+1}^T H_{xx} + \mu_{k+1}^T H_{\mu x} & x_{k+1}^T H_{x\mu} + \mu_{k+1}^T H_{\mu\mu} \end{bmatrix}\begin{bmatrix} A & B \\ KA & KB \end{bmatrix} \qquad (4.104)\\
&= 2\begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix}\begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} + \\
&+ 2\begin{bmatrix} x_{k+1}^T & \mu_{k+1}^T \end{bmatrix}\begin{bmatrix} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{bmatrix}\begin{bmatrix} A & B \\ KA & KB \end{bmatrix} \qquad (4.105)\\
&= 2\begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix}\begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} + 2\begin{bmatrix} x_{k+1}^T & \mu_{k+1}^T \end{bmatrix} H \begin{bmatrix} A & B \\ KA & KB, \end{bmatrix} \qquad (4.106)
\end{aligned}$$

neste caso o alvo é um vetor de ordem $1 \times (n+m)$ onde n é o número de estados do sistema e m é o número de entradas.

Fica evidente a partir da equação (4.106) que o alvo é dependente do modelo do ambiente.

4.3.4 Resumo

As tabelas abaixo mostram um resumo dos elementos mais significativos dos algoritmos crítico adaptativos, HDP, DHP, ADHDP e ADDHP, aplicados ao problema do DLQR.

Tabela 4.1: Funções valor e ações de controle para os CA induzidos pelo DLQR

Algoritmo	Função Valor	Ação de Controle
HDP	$V(x_k) = \left(x_k^T \otimes x_k^T\right) vet\,(P)$	$\mu_k = -\left(R + B^T P B\right)^{-1} B^T P A x_k$
DHP	$\frac{\partial V(x_k)}{\partial x_k} = 2\left(I \otimes x_k^T\right) vet(P)$	$\mu_k = -\left(R + B^T P B\right)^{-1} B^T P A x_k$
ADHDP	$Q\left(x_k, \mu_k\right) = \left(z_k^T \otimes z_k^T\right) vet\left(H\right)$	$\mu_k = -H_{\mu\mu}^{-1} H_{\mu x} x_k$
ADDHP	$\frac{\partial Q(x_k,\mu_k)}{\partial z_k} = 2(I \otimes z_k^T) vet(H)$	$\mu_k = -H_{\mu\mu}^{-1} H_{\mu x} x_k$

Tabela 4.2: Alvos para os CA induzidos pelo DLQR

Algoritmo	Alvo
HDP	$Y^{alvo} = x_k^T Q x_k + \mu_k^T R \mu_k + \phi_i^T \left(f(x_k, \mu(x_k))\right) \theta_{i-1}$
DHP	$Y^{alvo} = 2\left(x_k^T Q + \mu_k^T R K + \left(A x_k + B \mu_k\right)^T P \left(A + B K\right)\right)$
ADHDP	$Y^{alvo} = r_{k+1} + \phi_i^T \left(x_{k+1}, \mu_{k+1}\right) \theta_{i-1}$
ADDHP	$Y^{alvo} = 2\begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} + 2\begin{bmatrix} x_{k+1}^T & \mu_{k+1}^T \end{bmatrix} H \begin{bmatrix} A & B \\ KA & KB, \end{bmatrix}$

Tabela 4.3: Dependência do modelo para os CA

Algoritmos	Dependência de Modelo	
	Função Valor	Ação de Controle
HDP	Não	Sim
DHP	Sim	Sim
ADHDP	Não	Não
ADDHP	Sim	Não

5

Validação Computacionais dos Algoritmos

Sumário

Neste capítulo, realiza-se alguns experimentos computacionais utilizando os algoritmos HDP, DHP, ADHDP e ADDHP, com parametrizações da função valor e políticas induzidas pelo problema do LQR, para encontrar a política ótima ou subótima que minimiza uma função de custo quadrática. Emprega-se os algoritmos RLS, NLMS, PNLMS, IPNLMS e IAF-PNLMS para estimar a função valor. Adota-se ainda o modelo de um sistema dinâmico longitudinal de uma aeronave encontrado em (STEVENS; LEWIS, 1992) e um modelo de circuito elétrico como ambientes para a simulação computacional. Tais modelos não têm a pretensão de cobrir todas as minucias do ambiente real, mas essa hipótese é considerada verdadeira aqui, pois o objetivo é a avaliação dos algoritmos de aprendizagem por reforço.

5.1 Notação

Como foi visto anteriormente, os algoritmos de aprendizagem têm basicamente dois ciclos de execução, um mais externo em que ocorre a avaliação da política e um interno em que se dá a melhora da política. Levando em consideração esses dois ciclos, adotou-se a letra k para indicar a iteração mais externa dos algoritmos de aprendizagem e a letra k' para indicar as iterações de atualização da política. A iteração k' se torna igual a k quando a política é melhorada no mesmo ciclo de avaliação, abordagem conhecida como totalmente otimista (BERTSEKAS; TSITSIKLIS, 1996). A letra k também é utilizada, mas com algum número como subíndice, para indicar um elemento da matriz de ganho das ações de controle ou política.

A notação sem a indicação do índice da iteração é utilizada quando não trouxer prejuízo para o entendimento. Por exemplo, usa-se $x_k = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}$ e $\mu_k = \begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix}$ no lugar de $x_k = \begin{bmatrix} x_1^k & x_2^k & x_3^k & x_4^k \end{bmatrix}$ e $\mu_k = \begin{bmatrix} \mu_1^k & \mu_2^k \end{bmatrix}$, respectivamente.

5.2 Especificando o Ambiente

O ambiente é composto pelo processo a controlar e pela função de custo. O processo é o vínculo ou restrição imposta a função objetivo ou de custo que se deseja minimizar, sendo esta, a soma ponderada dos custo instantâneo.

Utilizou-se, na validação computacional dos algoritmos, dois processos diferentes ambos MIMO, um de terceira ordem com duas entradas e três saídas e o outro de quarta ordem também com duas entradas, mas com duas saídas.

5.2.1 Modelo Longitudinal de uma aeronave

Este modelo encontrado em (STEVENS; LEWIS, 1992) corresponde a dinâmica longitudinal de uma aeronave. No modelo apresentado, as equações da dinâmicas foram

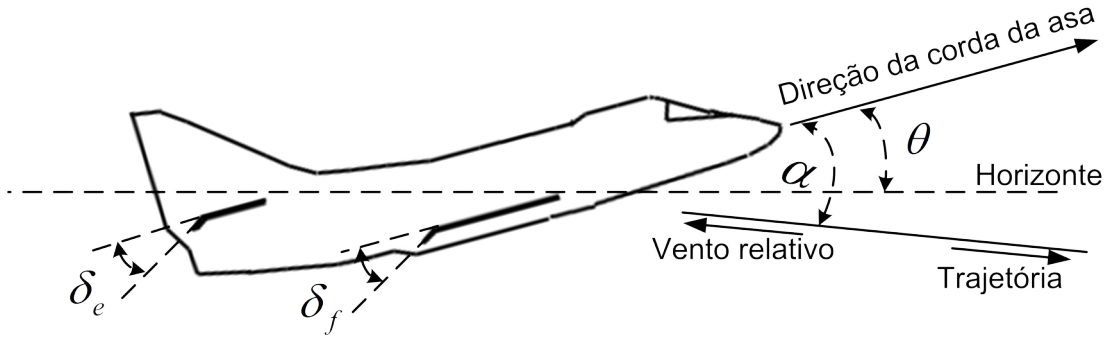
modificadas para incluir uma segunda ação de controle. Transformado, assim, a dinâmica longitudinal da aeronave em um sistema MIMO.

A figura (5.1) mostra as variáveis de estado que descrevem a dinâmica longitudinal da aeronave e são definidas como

- a taxa de *pitch* q, sendo $q = \dot{\theta}$ a derivada temporal do ângulo de *pitch* (θ);
- o ângulo de ataque α, que é o ângulo formada entre a trajetória da aeronave e a direção da corda da asa;
- o angulo de deflexão do profundor ou *elevador* (δ_e)

As ações de controle são sinais ($\mu = [\mu_1 \quad \mu_2]^T$) enviados ao sistema de atuação do profundor e ao sistema de atuação extra, este é um sistema conhecido como flaperon.

Figura 5.1: Movimento Longitudinal da Aeronave, levantar e abaixar nariz



A dinâmica longitudinal da aeronave é governado pelo sistema de equações

$$\begin{cases} x = Ax + B\mu \\ y = Cx + D\mu \end{cases}, \tag{5.1}$$

onde:

$$A = \begin{bmatrix} -1.10188 & 0.90528 & -0.00212 \\ 4.0639 & -0.77013 & -0.16919 \\ 0 & 0 & -10 \end{bmatrix}, \tag{5.2}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 10 \\ 10 & 0 \end{bmatrix}, \tag{5.3}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \; e \tag{5.4}$$

$$D = 0, \tag{5.5}$$

sendo um sistema MIMO com três saídas e duas entradas. As saídas são os próprios estados

$$x = \begin{bmatrix} \alpha \\ q \\ \delta_e \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \tag{5.6}$$

Esse modelo é instável como pode ser visto pelos valores dos autovalores da matriz A que são -2.8612, 0.9892 e -10. Além disso, é um modelo relativamente esparso como pode ser observado pelo grau de esparsidade indicados na tabela (5.1)

Tabela 5.1: Grau de esparsidade da aeronave

	Entrada 1	Entrada 2
Saída 1	0,86108	0,86108
Saída 2	0,86108	0,86108
Saída 3	0,98925	1

5.2.2 Circuito Elétrico

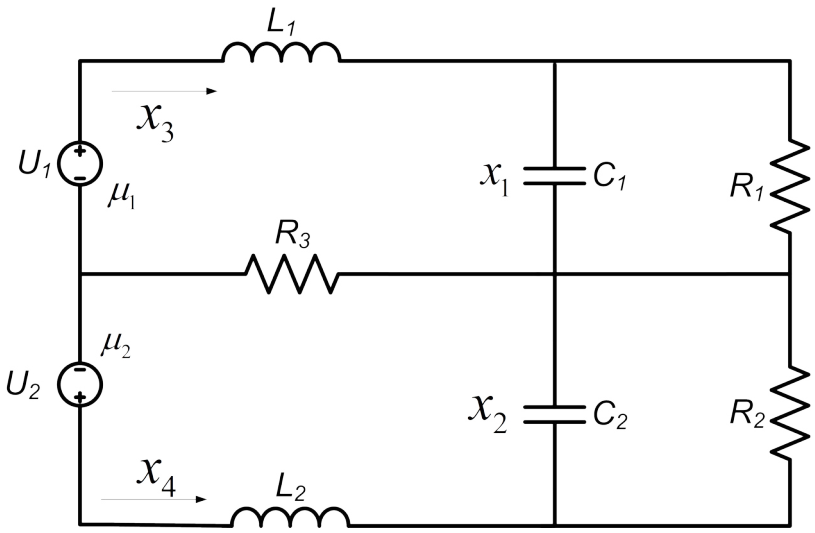
O segundo processo corresponde a um circuito elétrico de elementos passivos encontrados em (MACIEL, 2012). A figura (5.2) mostra o diagrama esquemático do circuito elétrico e também as variáveis de estado que descrevem a dinâmica desse circuito, as quais são definidas como

- x_1 é a tensão no capacitor C_1;
- x_2 é a tensão no capacitor C_2;
- x_3 é a corrente no indutor L_1;

- x_4 é a corrente no indutor L_2;

As saídas correspondem às tensões nos capacitores C_1 e C_2 e as ações de controle ($\mu = [\mu_1 \;\; \mu_2]^T$) são as tensões que as fontes U_1 e U_2 injetam no circuito;

Figura 5.2: Diagrama Esquemático do Circuito Elétrico



O modelo matemático do circuito é encontrado aplicando as equações de *Kirchoff* no circuito ilustrado na figura (5.2). Após manipulações algébricas, encontra-se o sistema de equações no espaço de estado dado por (MACIEL, 2012)

$$\begin{cases} x = Ax + B\mu \\ y = Cx + D\mu \end{cases}, \tag{5.7}$$

onde:

$$A = \begin{bmatrix} -\frac{1}{C_1 R_1} & 0 & \frac{1}{C_1} & 0 \\ 0 & -\frac{1}{C_2 R_2} & 0 & -\frac{1}{C_2} \\ \frac{1}{L_1} & 0 & -\frac{R_3}{L_1} & -\frac{R_3}{L_1} \\ 0 & -\frac{1}{L_2} & -\frac{R_3}{L_2} & -\frac{R_3}{L_2} \end{bmatrix}, \tag{5.8}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{L_1} & 0 \\ 0 & \frac{1}{L_2} \end{bmatrix}, \tag{5.9}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} e \tag{5.10}$$

$$D = 0, \tag{5.11}$$

$$\tag{5.12}$$

sendo um sistema MIMO com duas saídas e duas entradas.

Esse circuito tem um grau de esparsidade inferior a $0,5$ o que indica que ele é pouco esparso.

5.2.3 Função de Custo Instantânea ou de Recompensa

Finaliza-se a descrição do ambiente com a função de custo que é uma função quadrática dos estados e entradas. O objetivo do controlador ou agente é encontrar uma política que minimiza a soma dos valores fornecidos pela função de custo nas transições de estado. A função de custo tem a forma dada por

$$r(x_k, \mu_k) = x^t Q x + \mu^t R \mu \tag{5.13}$$

sendo Q e R matrizes simétricas definida positiva e semidefinida positiva, respectivamente, x um vetor de estados e μ é vetor de ações.

5.3 Aplicação dos Algoritmos à Aeronave

5.3.1 HDP

Nesta seção, o algoritmo HDP é empregado para encontrar uma política ótima utilizando a parametrização da função de custo e política dada pelo problema DLQR. Associado ao

HDP são utilizados os algoritmos RLS, PNLMS e LMS como estimadores dos parâmetros da função-V.

O objetivo do sistema de aprendizagem é estabilizar a aeronave, levando para zeros ou para os valores de referência os a taxa de *pitch*, o ângulo de ataque e a deflexão do profundo. Além disso, deve encontrar uma política que minimize uma função de custo.

5.3.1.1 Experimento com Ambiente Invariante no Tempo

Neste seção, são realizadas duas simulações as quais se diferenciam apenas quanto ao momento de melhoria da política.

Parâmetros de Configuração das Simulações: Os seguintes parâmetros são definidos para o algoritmo de aprendizagem HDP e para o ambiente:

- O fator de desconto igual a um que conduz a uma política ótima que estabiliza a aeronave;
- O retorno para cada estado inicial é estimado a cada cinco passos de simulação da trajetória, não obstante o retorno ser uma soma infinita de custos descontadas, na prática, tem-se que estimar o retorno no tempo finito.
- A trajetória dos estados é reiniciada em um estado selecionado aleatoriamente no espaço de estado após cinco iterações;
- Na primeira simulação, nomeada de experimento-1, a política é melhorada a cada transição de estado, uma abordagem conhecida como totalmente otimista (BERTSEKAS; TSITSIKLIS, 1996) e na segunda, denominada experimento-2, a política é melhorada a cada cinquenta iterações da avaliação da política;
- As matrizes Q e R da função de custo são matrizes identidade de ordem 3 e 2, respectivamente;
- A política inicial adotada

$$\mu_1 = \mu(x_1) = \begin{bmatrix} 0,0009 & 0,0021 & -0,1661 \\ -0,2059 & -0,4725 & 0,0046 \end{bmatrix} x_1,$$

  mas qualquer política do espaço de política pode ser adotada.
- O tempo de amostragem do ambiente é de 0,1 s;
- O ângulo de ataque (x_1) pertence ao intervalo $[-\pi/4\ \pi/4)rad$;
- A taxa de *pitch* (x_2) está restrita a $[-5\pi\ 5\pi]rad/s$;
- A deflexão do profundor(x_3) está restrito a $[-\pi/3\ \pi/3)rad$;

Forma Regressiva da HJB: Como se está usando a abordagem do problema DLQR, isso conduz às seguintes parametrizações, considerando o ambiente em mãos,

$$\begin{aligned} V(x_k) &= x_k^T P x_k \\ &= (x_k^T \otimes x_k^T) vet(P) \\ &= \left[\begin{array}{cccccc} x_1^2 & x_1x_2 & x_1x_3 & x_2^2 & x_2x_3 & x_3^2 \end{array} \right] \left[\begin{array}{cccccc} \theta_1 & 2\theta_2 & 2\theta_3 & \theta_4 & 2\theta_5 & \theta_6 \end{array} \right]^T \quad (5.14) \\ &= \phi^T \theta, \quad (5.15) \end{aligned}$$

sendo

$$P = \left[\begin{array}{ccc} \theta_1 & \theta_2 & \theta_3 \\ \theta_2 & \theta_4 & \theta_5 \\ \theta_3 & \theta_5 & \theta_6 \end{array} \right], \quad (5.16)$$

e

$$\mu_k = \mu(x_k) = Kx_k = (R + B^T P B)^{-1} B^T P A x_k, \quad (5.17)$$

para a função-V e para a política, respectivamente (LEWIS; VRABIE, 2009). Sendo $x^T \otimes x^T$ o produto de Kronecker entre os vetores de estado e θ é o vetor de parâmetros a determinar.

Para usar os algoritmos RLS, LMS e PNLMS ou qualquer algoritmo de aprendizagem supervisionado é preciso de um alvo e, tal alvo, é dado pela própria função valor estado estimada anteriormente, mas aplicada ao estado atual, mais o custo pela transição de estado.

$$\begin{aligned} V(x_k) &= r(x_k, \mu_k) + V(f(x_k, \mu_k)) \quad (5.18) \\ &= x_k^T Q x_k + \mu_k^T R \mu_k + x_{k+1}^T P x_{k+1}, \quad (5.19) \end{aligned}$$

sendo P a matriz dos parâmetros da função-V estimada na iteração k' da etapa de atualização da política. Pode-se ver que o alvo depende da estimativa atual da matriz P e, em consequência, ele muda a cada melhoria da política, alguns autores chamam esse tipo de alvo de alvo móvel.

A partir das parametrizações da política, da função-V e do alvo percebe-se que o algoritmo HDP independe do modelo do ambiente na fase de avaliação da política, mas é altamente dependente na etapa de melhoria da política.

Estimação da Função Valor: A tabela (5.2) mostra os valores dos parâmetros livres dos algoritmos RLS, PNLMS e LMS utilizados na simulação para estimar os parâmetros da função de custo ou função-V. Dentre vários parâmetros testados, esses foram os melhores do ponto de vista da convergência e estabilidade das estimativas paramétricas.

Tabela 5.2: Ajustes dos Parâmetros Livres para RLS, LMS e PNLMS.

Algoritmos	Parâmetros Livres			
	ρ	μ	λ	P
RLS			0,97	I_n
LMS		0,0001		
PNLMS	0,575	1,563		

Para constatar a convergência dos parâmetros, a solução de Riccati para o DLQR é tomada como referência que dá o valor exato dos parâmetros da função de custo calculados em tempo de projeto. Então, o valor de P para a aeronave com tempo de amostragem de $0,1$ s, Q e R definidos acima e utilizando o método de Schur é dado por

$$P = \begin{bmatrix} 5,3947 & 0,7427 & -0,0078 \\ 0,7427 & 1,6203 & -0,0067 \\ -0,0078 & -0,0067 & 1,1037 \end{bmatrix}, \tag{5.20}$$

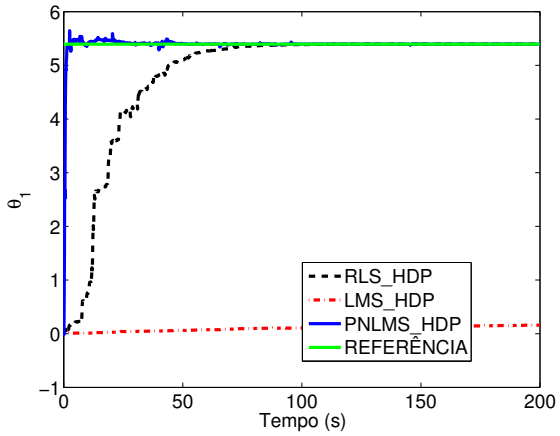
é para essa solução que o sistema de aprendizagem deve fazer os parâmetros da função de custo convergir.

As simulações partiram do estado inicial $\begin{bmatrix} 0,5 & 5 & 0,5 \end{bmatrix}^T$ para o ângulo de ataque, taxa de *pitch* e deflexão do profundor do avião, respectivamente. Após um determinado tempo de simulação, os estados iniciais eram revitalizados, não necessariamente nos mesmos valores originais, para impedir que eles se tornassem nulos e evitar a dependência linear do conjunto de regressores de Kronecker. Os valores dos estados e do custo são obtidos durante a execução do algoritmo de aprendizagem, ou seja, *online*.

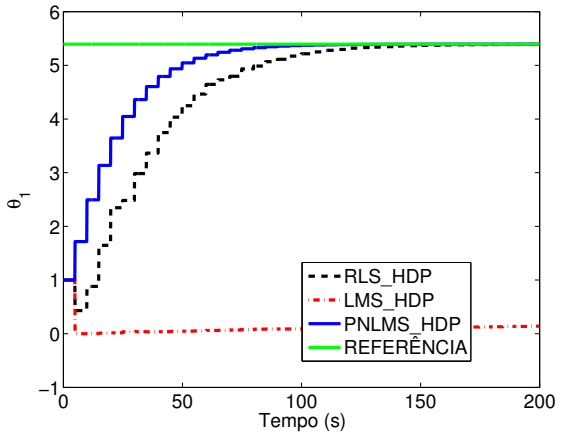
As figuras (5.3) e (5.4) mostram as trajetórias dos parâmetros da função-V para os três estimadores adotados nas simulações. Pode-se ver, exceto para o LMS, que a convergência dos parâmetros da função-V ocorreu com maior velocidade para o experimento-1. O experimento-2, porém, tem menor custo computacional já que é realizada menos melhoria da política que é um processo relativamente caro envolvendo uma inversão de matriz e um produto de matrizes.

Figura 5.3: Trajetórias seguidas pelos Parâmetros θ_1, θ_2 e θ_3 no processo de aprendizagem para 200s de simulação ou 2000 iterações - As figuras à esquerda geradas pelo experimento-1 e à direita pelo experimento-2.

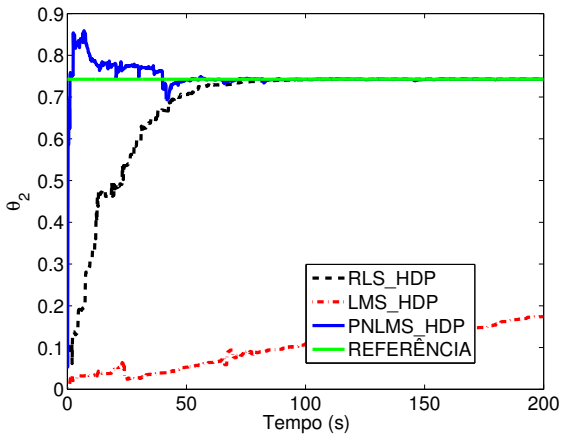
(a) Trajetória de θ_1 para experimento-1.



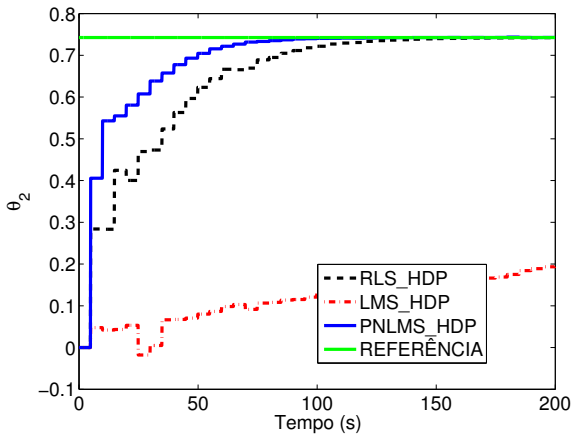
(b) Trajetória de θ_1 para experimento-2.



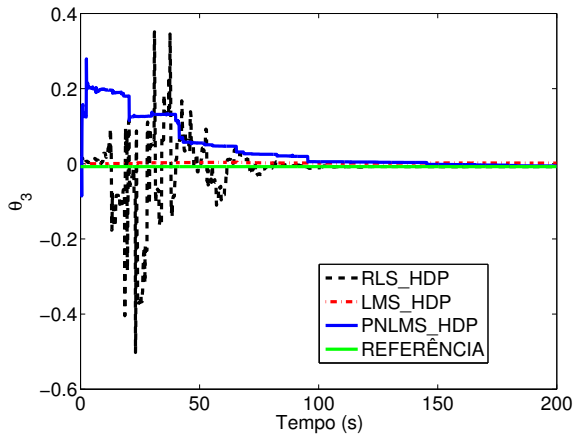
(c) Trajetória de θ_2 para experimento-1.



(d) Trajetória de θ_2 para experimento-2.



(e) Trajetória de θ_3 para experimento-1.



(f) Trajetória de θ_3 para experimento-2.

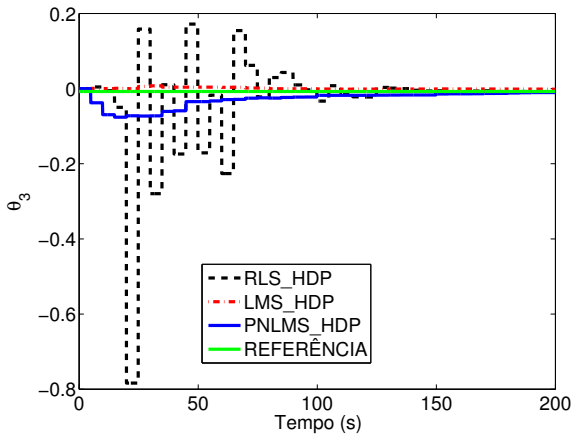
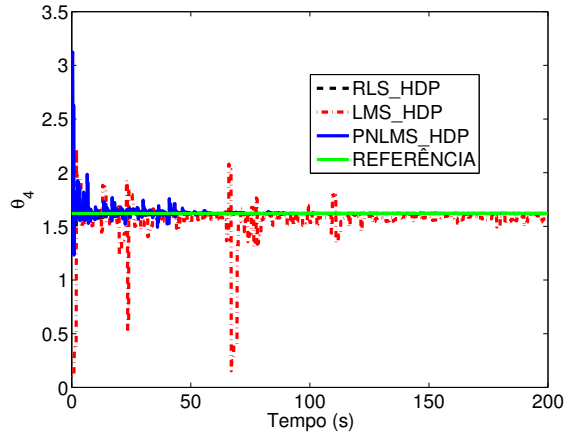
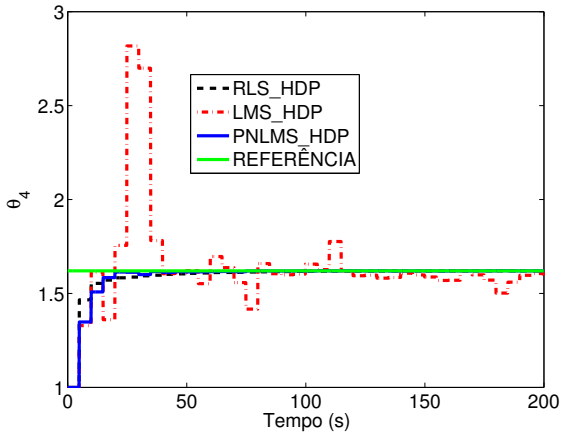


Figura 5.4: Trajetórias seguidas pelos Parâmetros θ_4, θ_5 e θ_6 no processo de aprendizagem para 200s de simulação ou 2000 iterações - As figuras à esquerda geradas pelo experimento-1 e à direita pelo experimento-2.

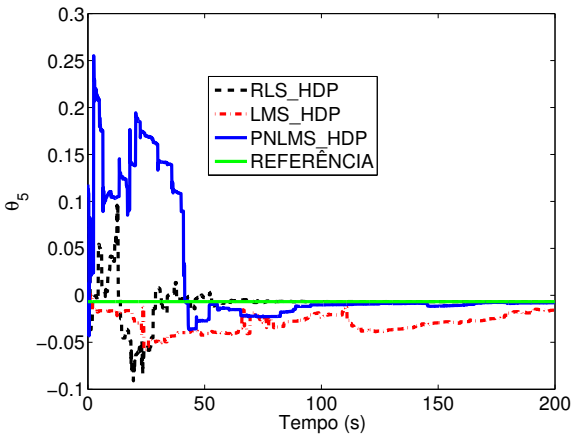
(a) Trajetória de θ_4 para experimento-1.



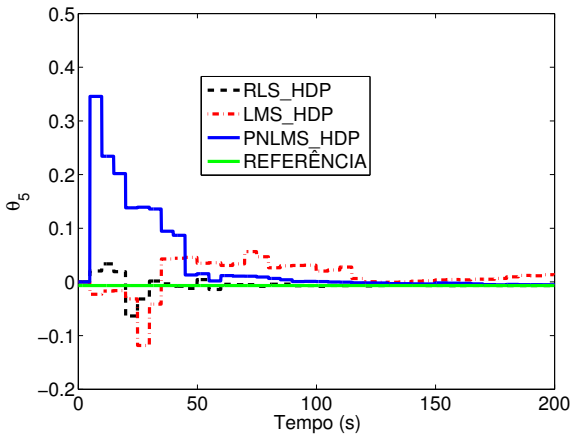
(b) Trajetória de θ_4 para experimento-2.



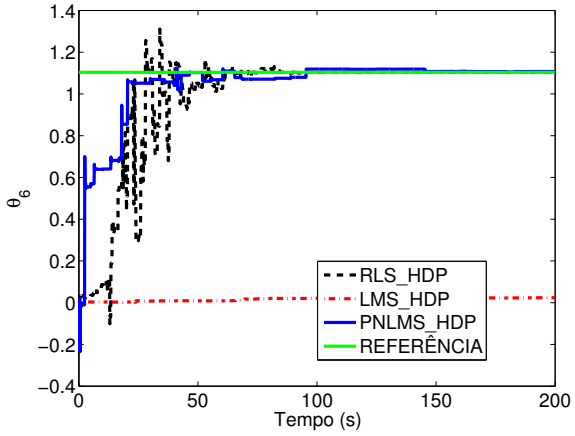
(c) Trajetória de θ_5 para experimento-1.



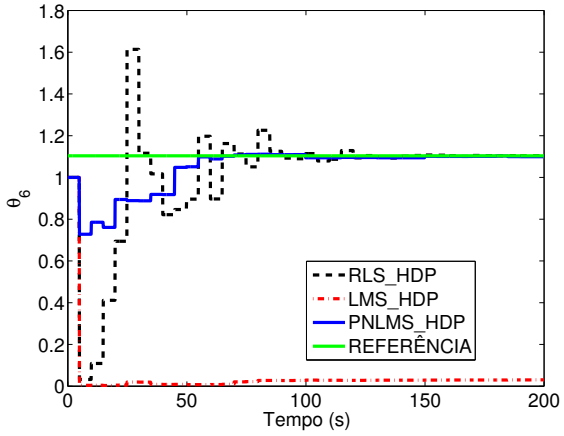
(d) Trajetória de θ_5 para experimento-2.



(e) Trajetória de θ_6 para experimento-1.



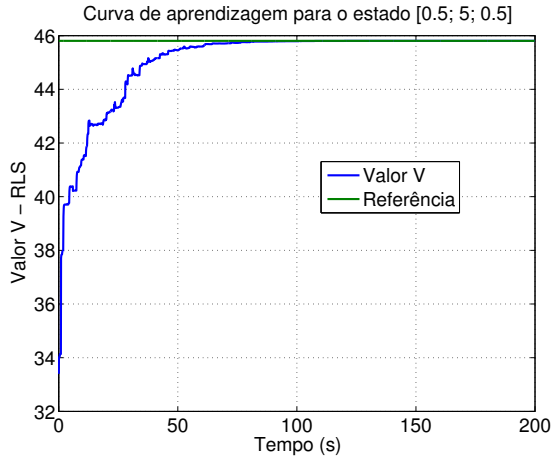
(f) Trajetória de θ_6 para experimento-2.



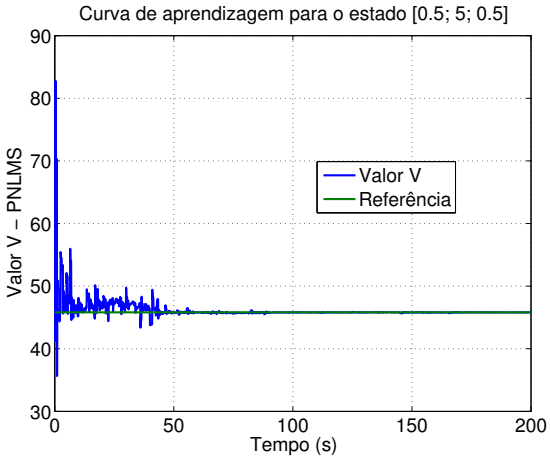
A figura (5.5) mostra as curvas de aprendizagem da função-V para o estado $\begin{bmatrix} 0,5 & 5 & 0,5 \end{bmatrix}^T$ para os experimento-1 e experimento-2.

Figura 5.5: Curva de aprendizagem para os experimentos 1 (gráficos (a) e (b)) e 2 (gráficos (c) e (d)) com o RLS e o PNLMS para o estado $[0,5 \;\; 5 \;\; 0,5]^T$.

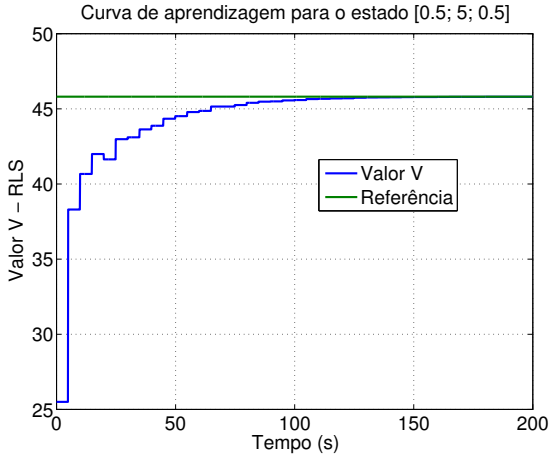
(a) Curva de aprendizagem com o RLS - Experimento-1.



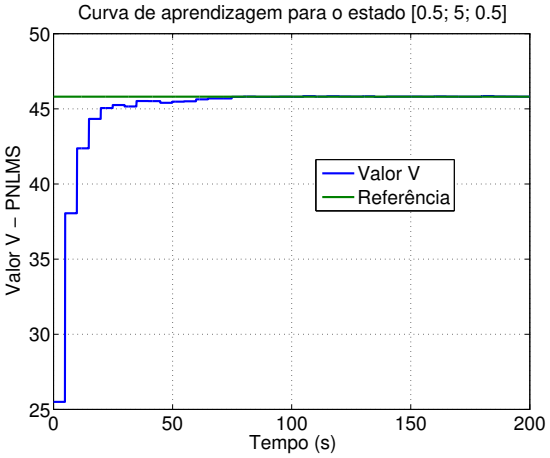
(b) Curva de aprendizagem com o PNLMS - Experimento-1.



(c) Curva de aprendizagem com o RLS - Experimento-2.

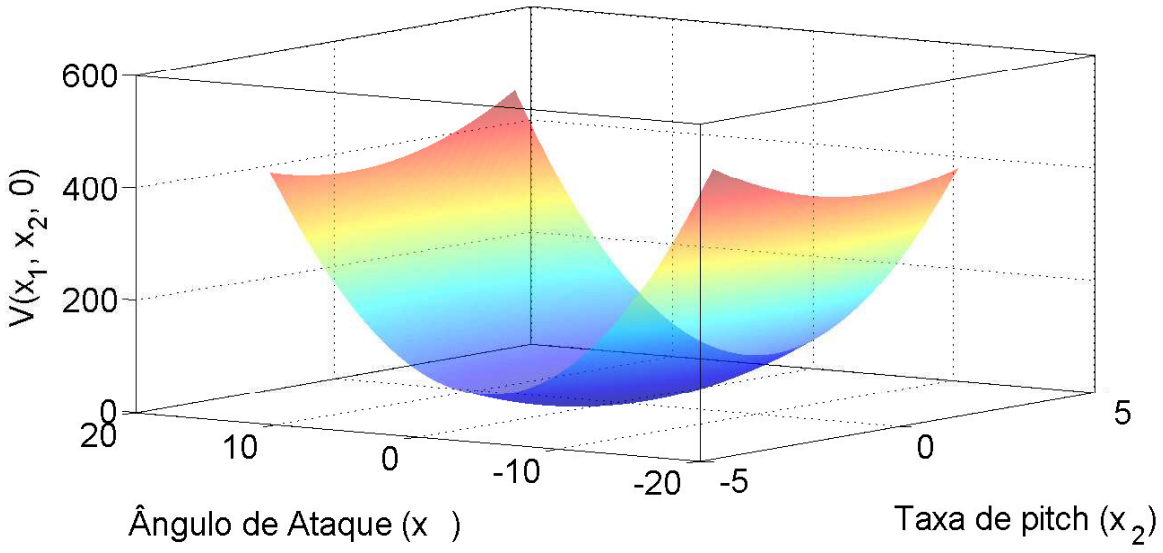


(d) Curva de aprendizagem com o PNLMS - Experimento-2.



A figura (5.6) mostra a superfície correspondente a secção transversão da função valor estado final quando a deflexão do profundo do avião é zero ($x_3 = 0$). Ampliou-se o intervalo do eixo da taxa de *pitch* para se ter uma visão melhor do gráfico.

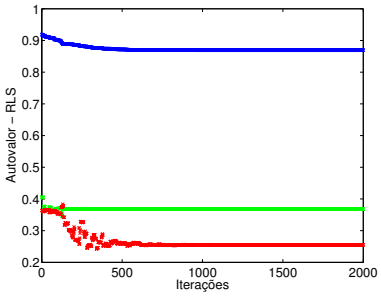
Figura 5.6: Função-V quando $x_3 = 0$.



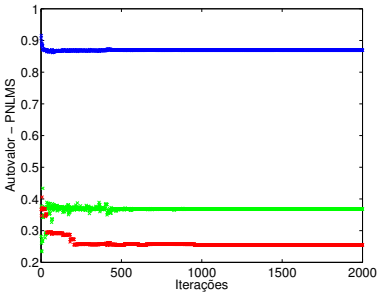
Os gráficos mostrados na figura (5.7) são referentes aos módulos dos autovalores encontrados durante o processo de aprendizagem para o experimento-1 e experimento-2. Constatar-se que em nenhum momento o sistema ficou instável com o RLS e PNLMS, mas o LMS encontrou autovalores instáveis.

Figura 5.7: Autovalores do sistema em malha fechada obtidos durante o processo de aprendizagem para o experimento-1 e experimento-2.

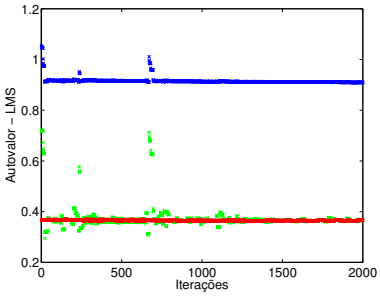
(a) Autovalores com o RLS - experimento-1.



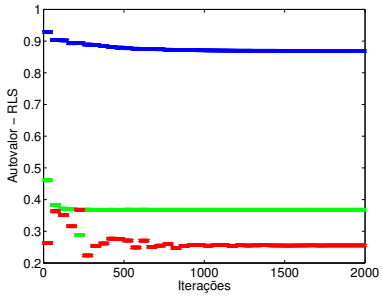
(b) Autovalores com o PNLMS - experimento-1.



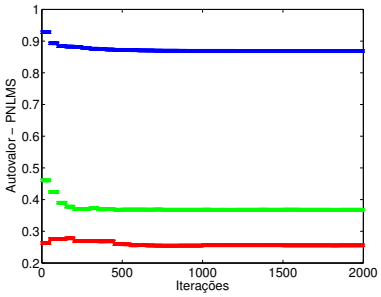
(c) Autovalores com o LMS - experimento-1.



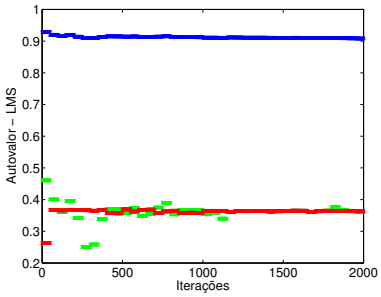
(d) Autovalores com o RLS - experimento-2.



(e) Autovalores com o PNLMS - experimento-2.



(f) Autovalores com o LMS - experimento-2.



Variável de Estado e Esforço de Controle: Obtém-se, durante o processo de aprendizagem, uma política diferente a cada melhoria da política do algoritmo HDP. Dessa forma, As políticas encontradas nas iterações 2, 50, 250 e 2000 são selecionadas para verificar o andamento do processo de aprendizagem e traçar suas trajetórias representativas tomando como estado inicial o ponto $\begin{bmatrix} 0,5 & 5 & 0,5 \end{bmatrix}^T$. Tais políticas correspondem aos ganhos de retroação de estado mostrados na tabela (5.3) para o experimento-1 e, para comparação, os ganhos obtidos para o experimento-2 nas mesmas iterações são mostrados na tabela (5.4).

Tabela 5.3: Ganhos ou parâmetros de políticas obtidas durante o processo de aprendizagem para o experimento-1.

Iteração	Ganhos da Política para experimento-1		
	RLS	PNLMS	LMS
2	0,0078 0,0134 0,0022 0,2462 0,5427 0,0040	0,0118 0,0215 0,0038 0,2498 0,5879 0,0085	0,0011 0,0022 0,0002 0,0377 0,0845 0,0007
50	0,0048 0,0094 0,0067 0,2999 0,5846 0,0014	0,0685 0,0411 0,1023 0,5915 0,6085 0,0188	-0,0029 -0,0076 0,0008 0,2386 0,5752 -0,0087
250	-0,1858 -0,0270 0,0623 0,4742 0,6028 -0,0155	0,0334 0,0267 0,1705 0,5730 0,6148 0,0109	-0,0048 -0,0164 0,0021 0,2501 0,6122 -0,0142
2000	-0,0050 -0,0039 0,1782 0,5644 0,6129 -0,0066	-0,0042 -0,0040 0,1786 0,5643 0,6130 -0,0067	-0,0018 -0,0073 0,0056 0,2929 0,5844 -0,0086

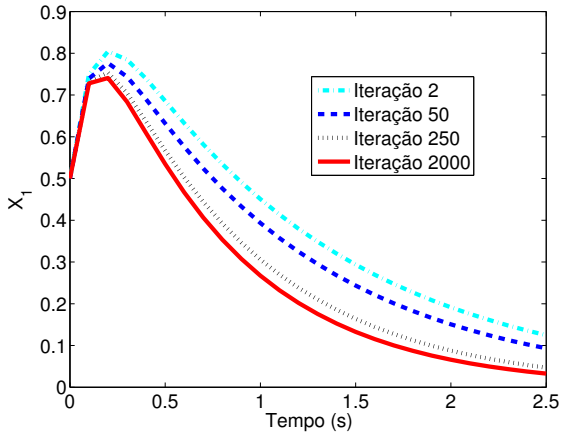
Tabela 5.4: Ganhos ou parâmetros de políticas obtidas durante o processo de aprendizagem para o experimento-2.

Iteração	Ganhos da Política Para experimento-2		
	RLS	PNLMS	LMS
2	-0,0009 -0,0021 0,1662 0,2060 0,4725 -0,0047	-0,0009 -0,0021 0,1662 0,2060 0,4725 -0,0047	-0,0009 -0,0021 0,1662 0,2060 0,4725 -0,0047
50	0,0010 0,0016 0,0073 0,3338 0,5729 -0,0032	-0,0212 0,0652 0,1234 0,4008 0,5531 0,0363	-0,0032 -0,0094 0,0010 0,2320 0,5417 -0,0095
250	0,0534 -0,0032 0,2282 0,4316 0,5982 -0,0078	-0,0400 0,0183 0,1514 0,5014 0,6065 0,0080	-0,0063 -0,0235 0,0041 0,2749 0,7054 -0,0192
2000	-0,0050 -0,0039 0,1781 0,5643 0,6129 -0,0066	-0,0061 -0,0038 0,1776 0,5645 0,6129 -0,0065	-0,0021 -0,0007 0,0070 0,3002 0,5864 -0,0046

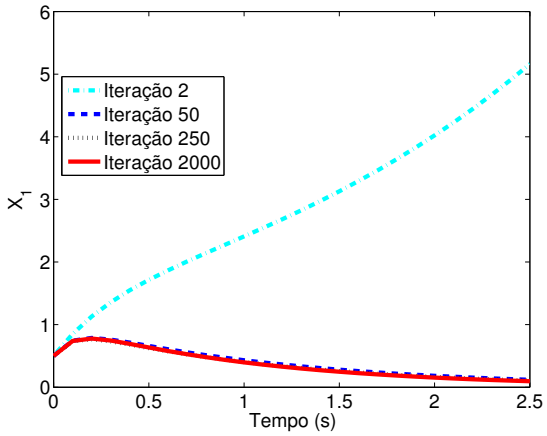
A figura (5.8) mostra as trajetórias representativas seguidas pelos estados do sistema com os ganhos das políticas dadas na tabela (5.3) e estado inicial $\begin{bmatrix} 0,5 & 5 & 0,5 \end{bmatrix}^T$. Observa-se que o LMS encontra políticas não estabilizante durante o processo de aprendizagem. Nota-se que as políticas das iterações 50, 250 e 2000 são tão próximas uma da outra que não se consegue diferenciá-las nos gráficos.

Figura 5.8: Trajetórias dos estados do sistema utilizando RLS (esquerda) e LMS (Direita) para políticas encontradas durante estimação dos parâmetros da função-V no experimento-1.

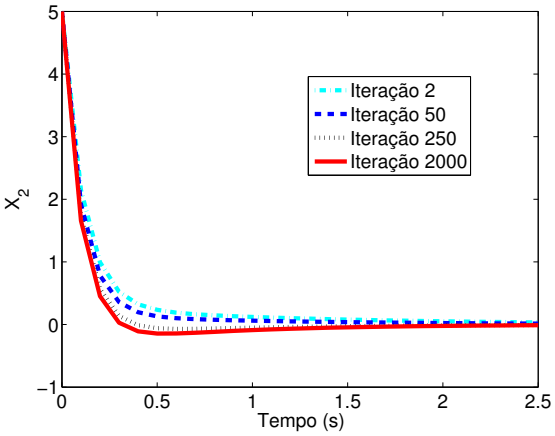
(a) Trajetória de x_1 com o RLS.



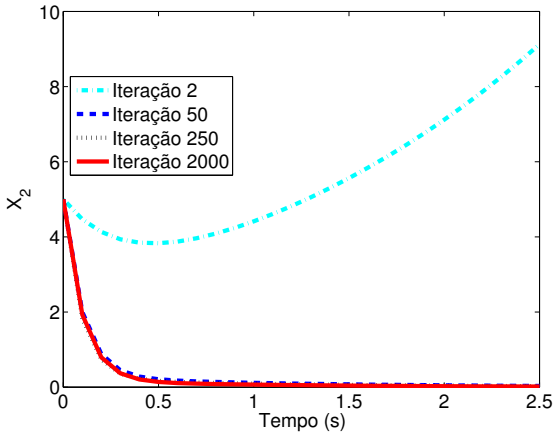
(b) Trajetória de x_1 com o LMS.



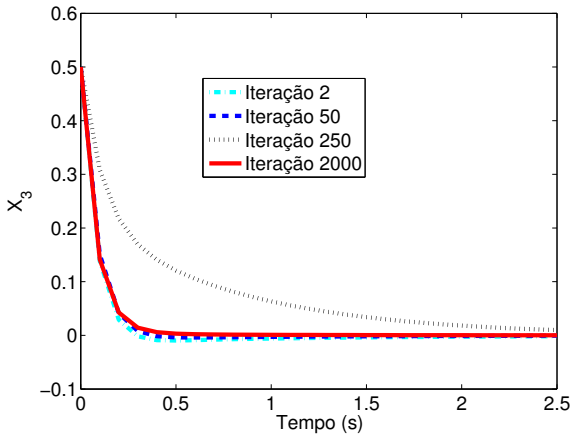
(c) Trajetória de x_2 com o RLS.



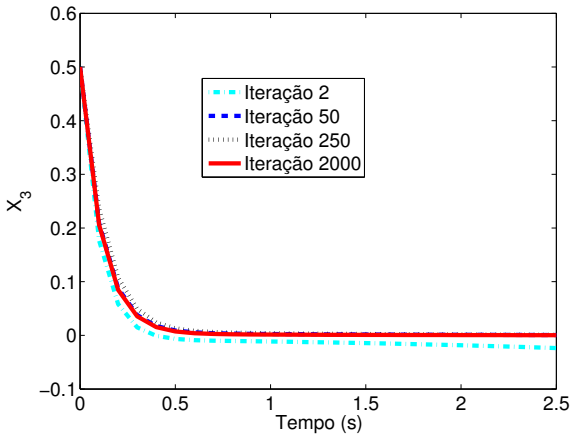
(d) Trajetória de x_2 com o LMS.



(e) Trajetória de x_3 com o RLS.



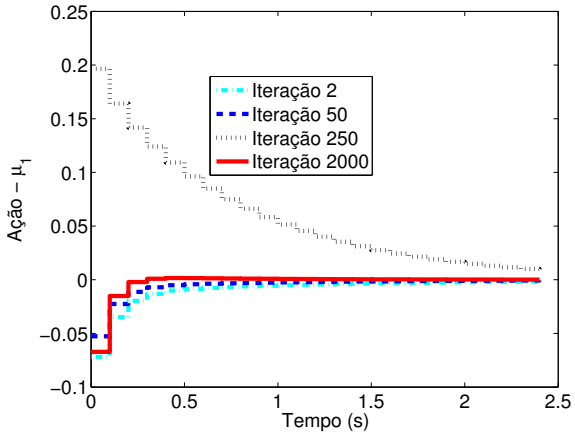
(f) Trajetória de x_3 com o LMS.



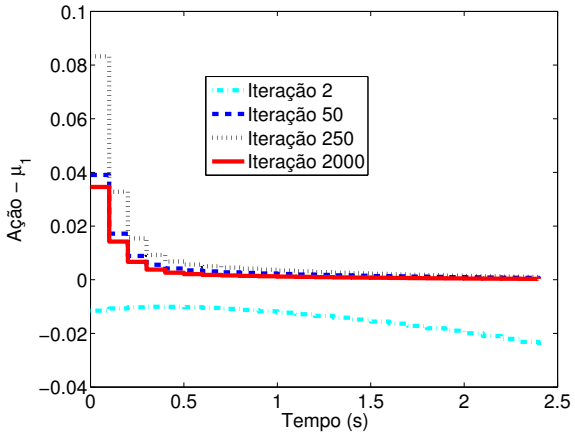
As figuras (5.9) e (5.10) mostram as trajetórias representativas das ações de controle e dos custos quando o sistema inicia no estado $\begin{bmatrix} 0,5 & 5 & 0,5 \end{bmatrix}^T$, respectivamente. As funções de custo a partir da quinquagésima iteração são bastante próximas.

Figura 5.9: Políticas encontradas com o RLS (esquerda) e o LMS (Direita) para o experimento-1.

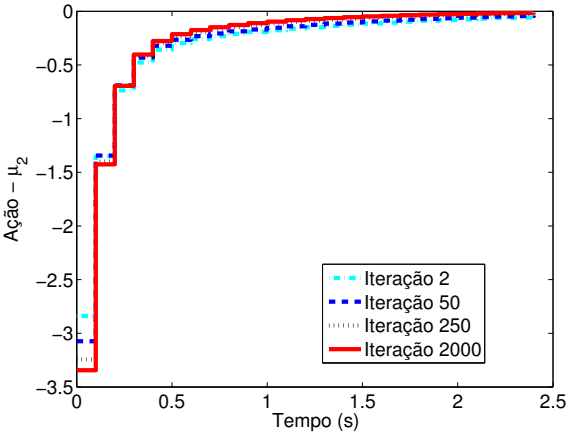
(a) Ações de controle μ_1 com o RLS.



(b) Ações de controle μ_1 com o LMS.



(c) Ações de controle μ_2 com o RLS.



(d) Ações de controle μ_2 com o LMS.

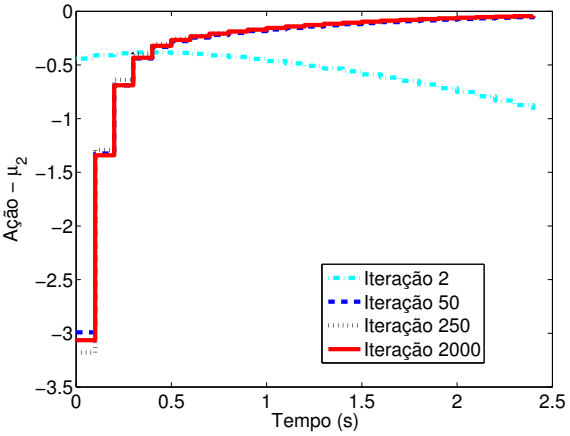
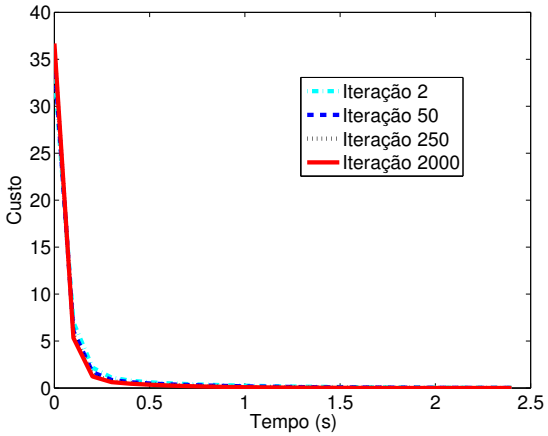
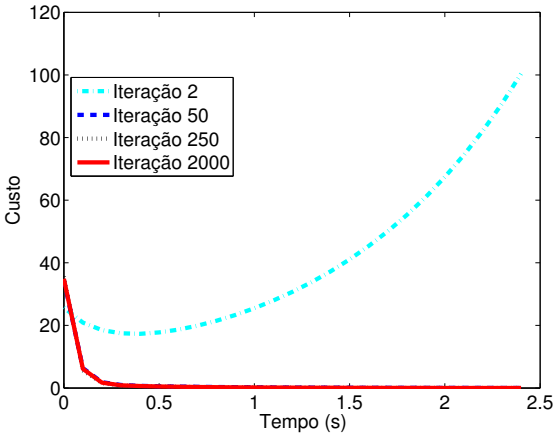


Figura 5.10: Custos encontradas com o RLS (esquerda) e o LMS (Direita) para o experimento-1.

(a) Custos encontrados com o RLS.



(b) Custos encontrados com o LMS.



5.3.1.2 Conclusão

Nesta abordagem do HDP, os algoritmos RLS, PNLMS e LMS forma empregados para estimação dos parâmetros da função-V. Os parâmetros convergiram para os valores esperados, solução pelo método de Schur, quando o RLS e PNLMS são utilizados como estimadores paramétricos, embora apresentem velocidades de convergência diferentes. Considera-se que o sistema atingiu a convergência quando ele alcança 5% do valor desejado e não sai mais desse limite. Com o LMS não foi possível obter convergência em 200 segundos de simulação. O PNLMS obteve uma velocidade de convergência tão boa quanto o RLS, no entanto, devido a quantidade de parâmetros de ajuste desse algoritmo, houve maior dificuldade em sua sintonia. O RLS estimou os parâmetros com melhor precisão e exatidão além de precisar de menos excitação que os dois algoritmo anteriores, mas com um custo computacional bem maior.

Neste ambiente, verifica-se que a simulação com atualização da política a cada transição de estado (experimento-1) teve uma convergência mais rápida para os parâmetros da função valor estado que a com atualização da política a cada cinquenta iterações da avaliação da política (experimento-2). No entanto, o experimento-1 teve um custo computacional maior devido a sua melhoria da política ser mais frequente que a do experimento-2. Um ponto importante é que o sistema ficou estável durante o processo de aprendizagem tanto para o experimento-1 quanto para o experimento-2 como pode ser visto pelo autovalores do sistema em malha fechada mostrados na figura (5.7), exceto para o LMS que não se conseguiu convergência no período considerado.

5.3.1.3 Experimento com Ambiente Variante no Tempo

Realizou-se, ainda, uma análise do comportamento do sistema de aprendizagem face uma mudança brusca no ambiente para verificar o comportamentos das estimativas dos parâmetros feitas pelos algoritmos RLS, LMS e PNLMS da função de custo. Além disso, Ampliou-se o espaço de estados para se ter maior variação do estado inicial nas revitalizações de estado.

A mudança no ambiente é realizada na função de custo, as matrizes Q e R são alteradas durante a simulação o que ocasiona mudança na política de controle ótima. Após a mudança da função de custo a matriz P dada em (5.20) passa a ser

$$\begin{bmatrix} 15,2905 & 3,1909 & -0,0395 \\ 3,1909 & 5,0172 & -0,0318 \\ -0,0395 & -0,0318 & 2,7044 \end{bmatrix}. \tag{5.21}$$

Parâmetros de Configuração das Simulações: Os parâmetros para o algoritmo HDP para esta simulação são exatamente iguais ao experimento-1 realizado anteriormente,

exceto pela ampliação do espaço de estado.

O novo espaço de estado está definido da seguinte forma:

- O ângulo de ataque (x_1) pertence ao intervalo $[-2\pi\ 2\pi)rad$;
- A taxa de *pitch* (x_2) está restrita a $[-15\pi\ 15\pi]rad/s$;
- A deflexão do profundor(x_3) está restrito a $[-2\pi\ 2\pi)rad$;

ampliou-se o espaço de estado para se ter uma maior excursão dos estados inicias nas revitalização e consequências quanto a convergência dos parâmetros.

As simulações foram iniciadas com a função de custo dada pela equação (5.13) e no instante 80 s ela foi alterada. A nova função de custo tem a mesmo estrutura da função anterior, mas com R e Q dados por

$$Q = \begin{bmatrix} 5,3 & 0 & 0 \\ 0 & 5,3 & 0 \\ 0 & 0 & 5,3 \end{bmatrix} e \tag{5.22}$$

$$R = \begin{bmatrix} 2,4 & 0 \\ 0 & 2,4 \end{bmatrix} \tag{5.23}$$

Estimação da Função Valor: Foram realizadas simulações com os algoritmos LMS, PNLMS e RLS como estimadores da função-V e os valores dos seus parâmetros livres foram ajustados segundo a tabela

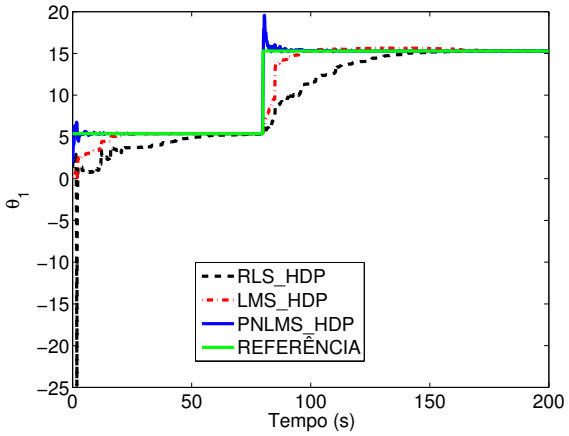
Tabela 5.5: Ajuste dos parâmetros livres para RLS, LMS e PNLMS.

Algoritmos	Parâmetros Livres			
	ρ	μ	λ	P
RLS			0,97	$100I_n$
LMS		0,0001		
PNLMS	0,575	1,563		

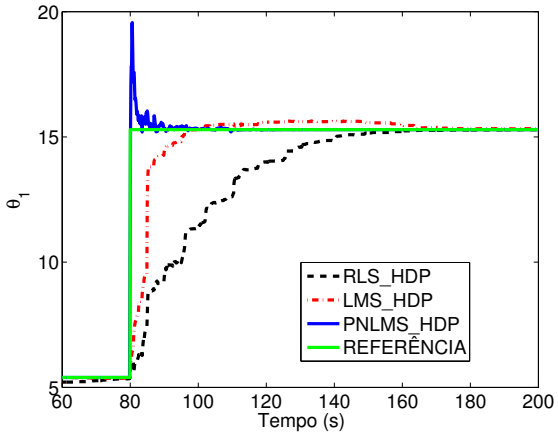
As figuras (5.11) e (5.12) mostram as trajetórias seguidas pelos parâmetros da função-V antes e após a mudança no ambiente e as soluções pelo método de Schur (em verde) correspondentes. Pode-se verificar que os valores desejados foram alcançados relativamente rápido pelo algoritmo PNLMS, enquanto os algoritmos LMS e RLS tiveram uma velocidade de convergência menor.

Figura 5.11: Trajetória seguida pelo Parâmetro θ_1, θ_2 e θ_3 no processo de aprendizagem antes e após mudança no ambiente - As figuras à direita são uma ampliação das figuras à esquerda entre os instante 60s e 200s. A mudança no ambiente ocorre no instante 80s

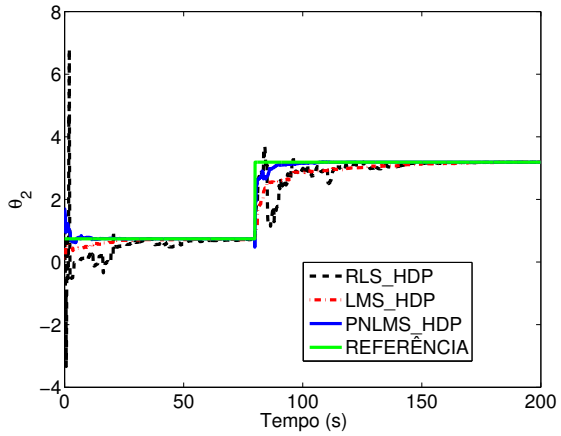
(a) Trajetória do parâmetro θ_1 .



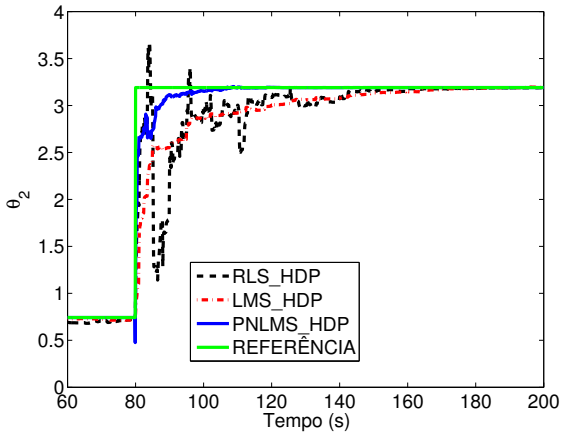
(b) Ampliação da simulação entre 60 a 200s.



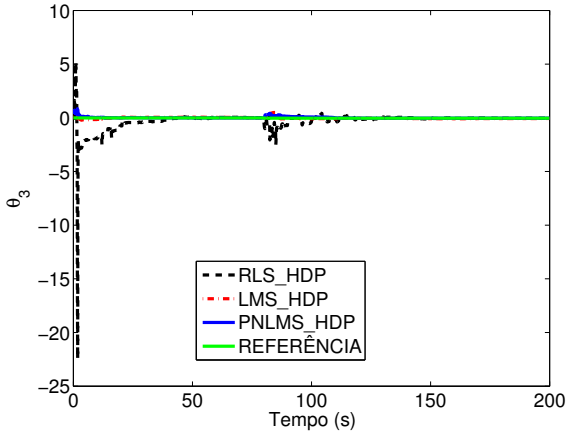
(c) Trajetória do parâmetro θ_2.



(d) Ampliação da simulação entre 60 a 200s.



(e) Trajetória do parâmetro θ_3.



(f) Ampliação da simulação entre 60 a 200s.

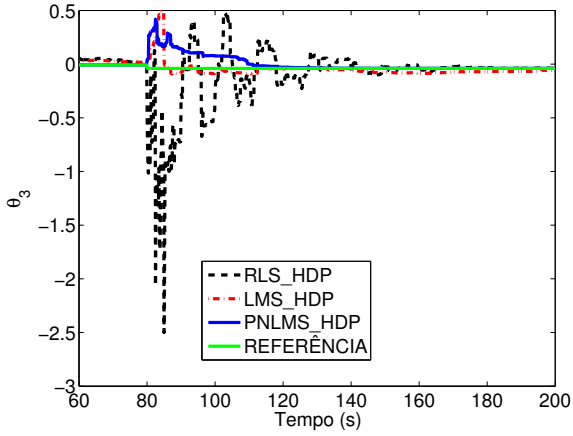
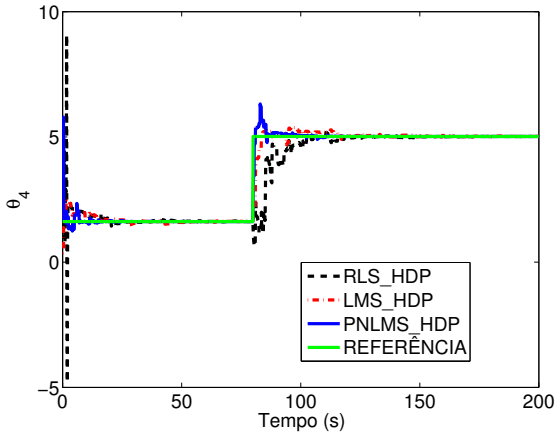
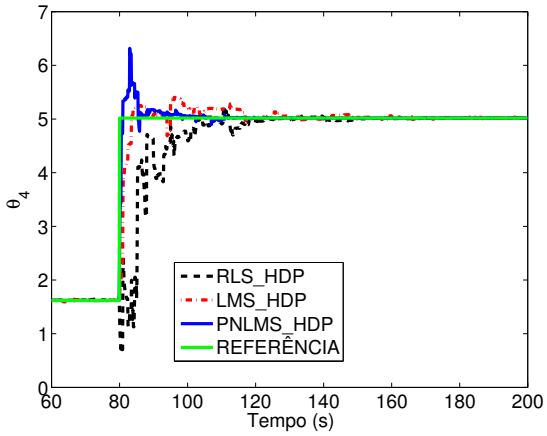


Figura 5.12: Trajetória seguida pelo Parâmetro θ_4, θ_5 e θ_6 no processo de aprendizagem antes e após mudança no ambiente - As figuras à direita são uma ampliação das figuras à esquerda entre os instante 60s e 200s. A mudança no ambiente ocorre no instante 80s

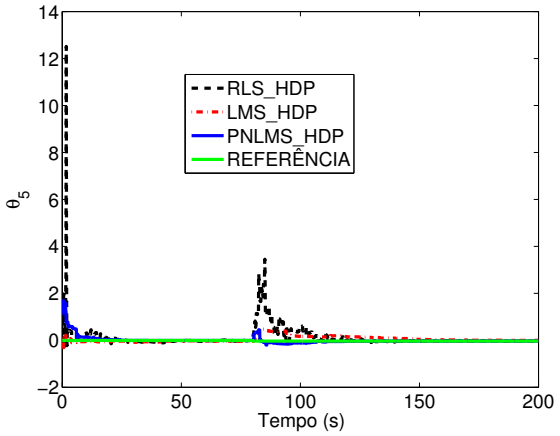
(a) Trajetória do parâmetro θ_4.



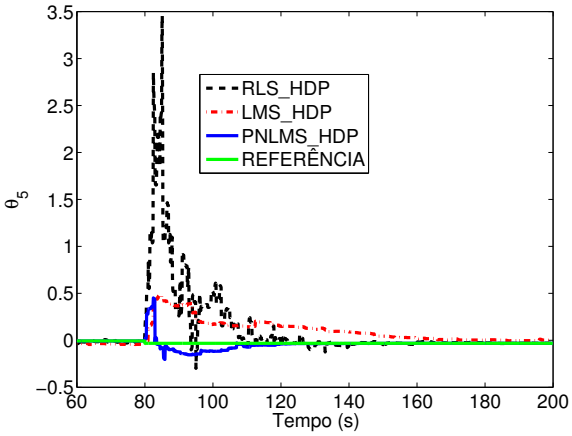
(b) Ampliação da simulação entre 60 a 200s.



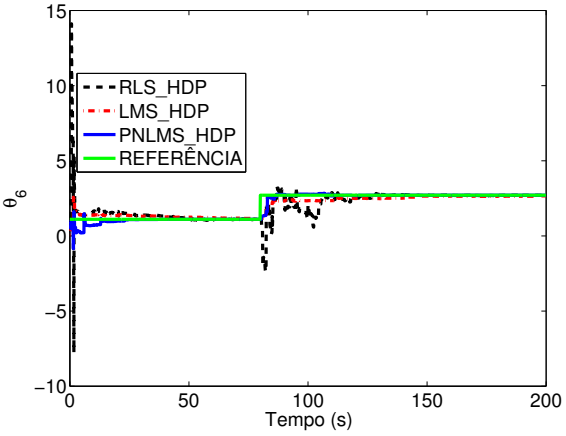
(c) Trajetória do parâmetro θ_5.



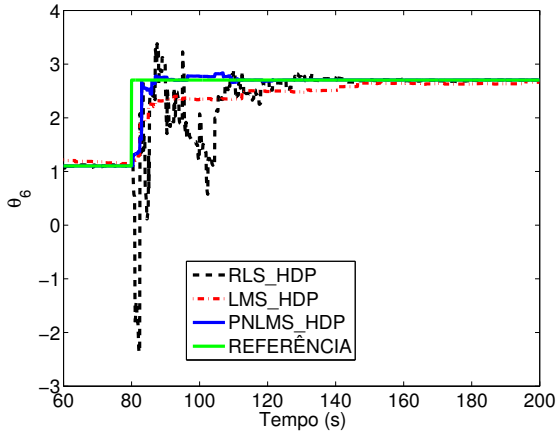
(d) Ampliação da simulação entre 60 a 200s.



(e) Trajetória do parâmetro θ_6.



(f) Ampliação da simulação entre 60 a 200s.



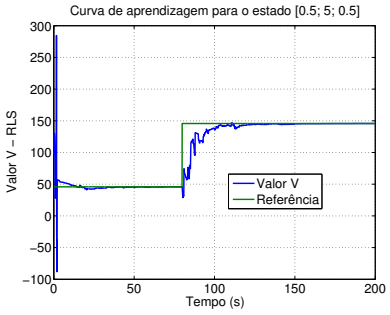
Comparando os experimentos anteriores com esta simulação, constatou-se que uma maior excursão dos estados iniciais para a abordagem de exploração por revitalização de estado provocou uma grande melhora na velocidade de convergência. Até mesmo o LMS

convergiu neste experimento diferentemente dos anteriores.

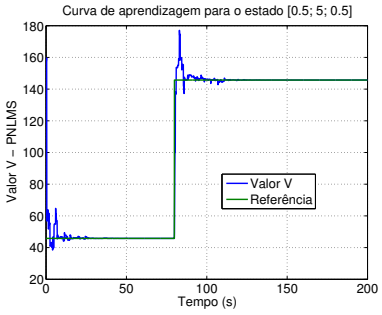
A figura (5.13) mostra as curvas de aprendizagem da função-V para o estado $\begin{bmatrix} 0,5 & 5 & 0,5 \end{bmatrix}^T$ para antes e após mudança do ambiente.

Figura 5.13: Curva de aprendizagem com o RLS (esquerda) e o PNLMS (centro) e o LMS (esquerda) para o estado $[0,5 \;\; 5 \;\; 0,5]^T$ antes e após mudança do ambiente.

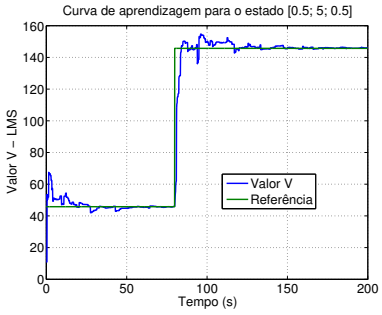
(a) Curva de aprendizagem com o RLS.



(b) Curva de aprendizagem com o PNLMS.

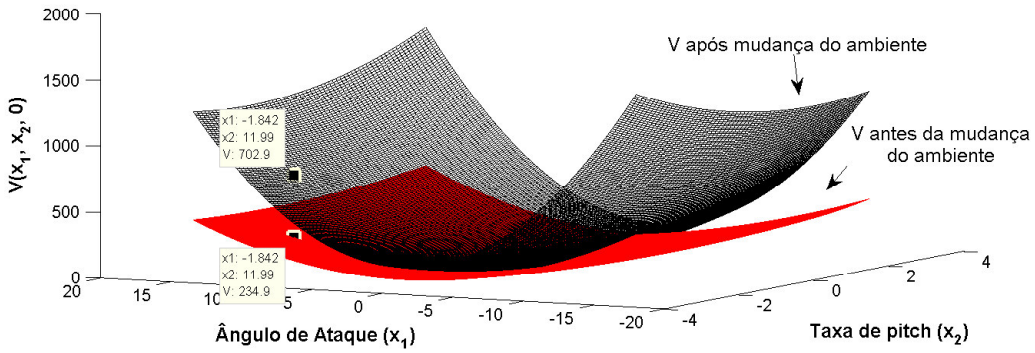


(c) Curva de aprendizagem com o LMS.



A figura (5.14) mostra a secção transversão da função valor estado nas iterações 799 (convergência já ocorrida, mas antes da mudança do ambiente) e 2000 (após mudança do ambiente) quando a deflexão do profundo do avião é zero ($x_3 = 0$).

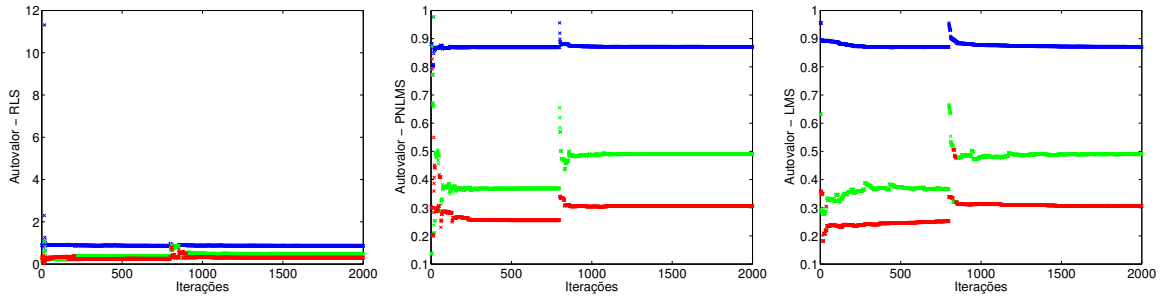
Figura 5.14: Funções valor nas iterações 799 e 2000 quando $x_3 = 0$.



Traçou-se os gráficos mostrados na figura (5.15) referentes aos módulos dos valores dos autovalores encontrados durante o processo de aprendizagem com mudança paramétrica do ambiente na iteração 800 do algoritmo HDP. Pode-se constatar que em nenhum momento o sistema ficou instável após mudança do ambiente.

Figura 5.15: Autovalores do sistema em malha fechada obtidos durante o processo de aprendizagem com mudança do ambiente na iteração 800.

(a) Autovalores com o RLS. **(b)** Autovalores com o PNLMS. **(c)** Autovalores com o LMS.



Variável de Estado e Esforço de Controle: A tabela (5.6) mostra os ganhos das ações de controle obtidos nas iterações 799 e 2000. Tais políticas são usadas para traçar os gráficos dos estados e custo para um estado específicos.

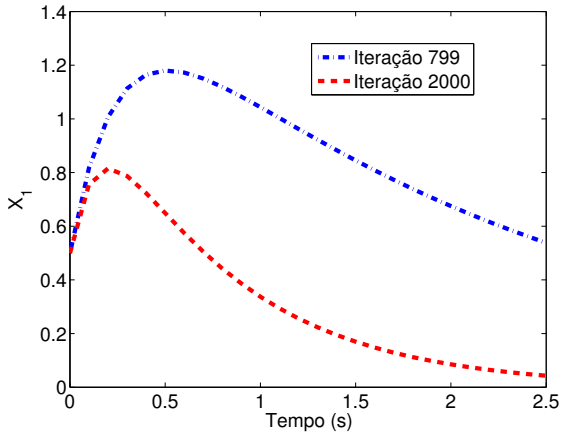
Tabela 5.6: Ganhos ou parâmetros de políticas obtidas durante o processo de aprendizagem.

Iteração	Ganhos da Política Para experimento-1		
	RLS	PNLMS	LMS
799	-0,0055 -0,0037 0,1771 0,5607 0,6128 -0,0065	-0,0050 -0,0039 0,1782 0,5644 0,6129 -0,0066	0,0078 -0,0084 0,1831 0,5563 0,6146 -0,0097
2000	-0,0057 -0,0043 0,0986 0,5121 0,4874 -0,0059	-0,0058 -0,0043 0,0986 0,5122 0,4875 -0,0059	-0,0080 -0,0029 0,0973 0,5125 0,4878 -0,0050

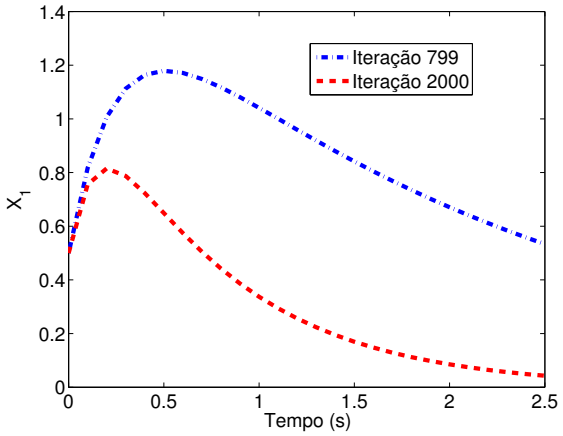
A figura (5.16) mostra uma representação das trajetórias seguidas pelos estados do sistema do estado inicial $\begin{bmatrix} 0,5 & 5 & 0,5 \end{bmatrix}$ ao estado $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ para as políticas ótimas encontradas antes e após mudança no ambiente.

Figura 5.16: Trajetórias dos estados do sistema para políticas ótimas encontradas antes e após mudança do ambiente com o RLS (esquerda) e o PNLMS (direita).

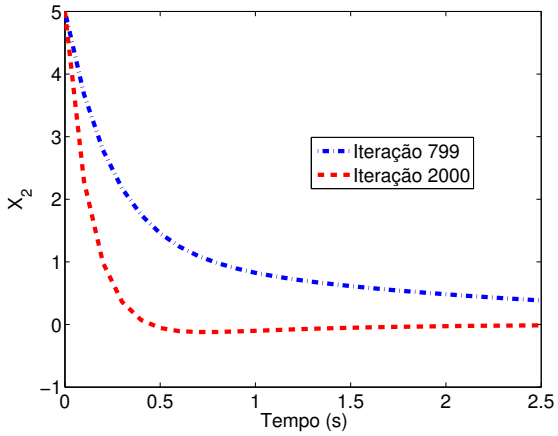
(a) Trajetórias de x_1.



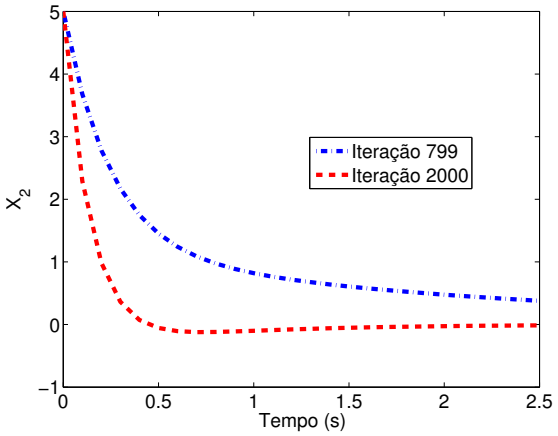
(b) Trajetórias de x_1.



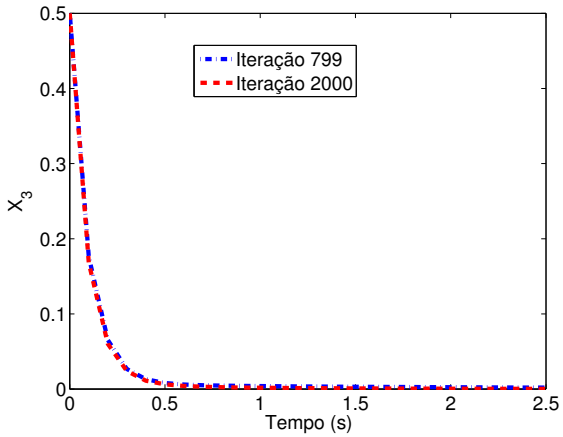
(c) Trajetórias de x_2.



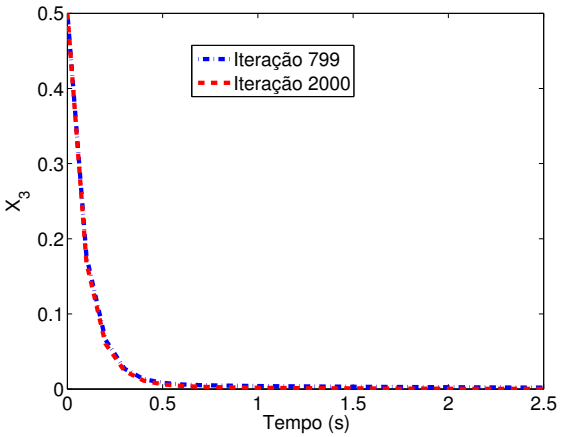
(d) Trajetórias de x_2.



(e) Trajetórias de x_3.



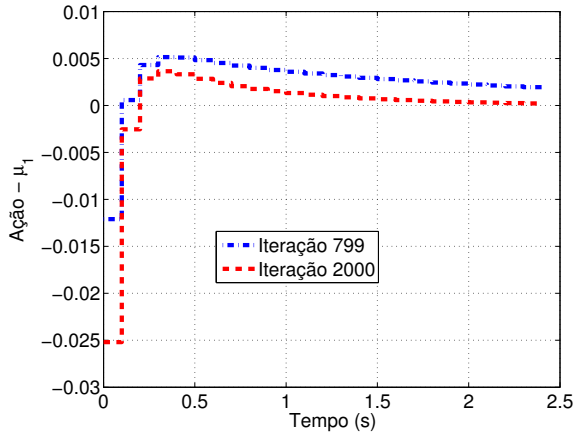
(f) Trajetórias de x_3.



A figura (5.18) mostra os gráficos do custo mínimo encontrados antes, iteração 799, e após, iteração 2000, a mudança no ambiente e a figura (5.17) mostra os gráficos das políticas ótimas encontradas antes e após a mudança no ambiente. Esses custos e políticas foram gerados tomando como base o estado inicial $\begin{bmatrix} 0,5 & 5 & 0,5 \end{bmatrix}$.

Figura 5.17: Política ótima para o estado inicial $[0,5 \quad 5 \quad 0,5]$ encontradas no processo de aprendizagem antes (azul) e após (vermelho) mudança no ambiente.

(a) Ações de controle ótimas μ_1.



(b) Ações de controle ótimas μ_2.

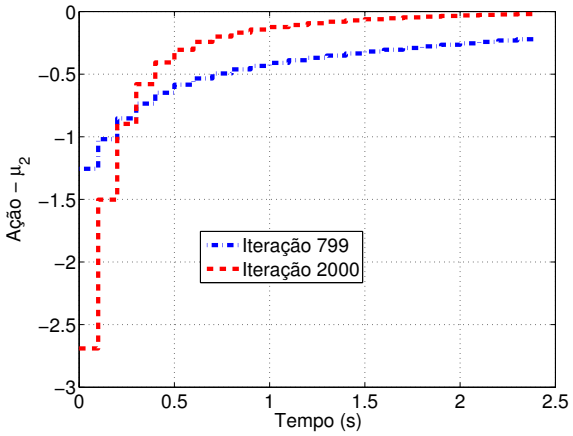
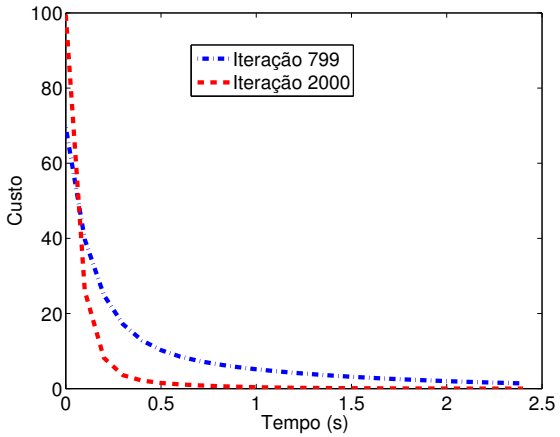


Figura 5.18: Custos mínimos para o estado inicial $[0,5 \quad 5 \quad 0,5]$ antes (azul) e após (vermelho) mudança de ambiente



5.3.1.4 Conclusão

Neste experimento, empregou-se novamente os algoritmos RLS, PNLMS e LMS para estimação dos parâmetros da função valor estado, verificou-se a convergência desses parâmetros para os valores esperados antes e após mudança no ambiente sempre permanecendo a revitalização de estado como excitador do sistema de aprendizagem. O LMS teve um comportamento bem inferior aos demais, enquanto o PNLMS teve um desempenho tão bom quanto o RLS.

O intervalo de validação do processo foi ampliado, em consequência, obteve-se uma maior variação do estado inicial que ocasionou maior velocidade de convergência. Como o processo é linear e não mudou antes dos 80s, a maior excursão dos estados iniciais nas revitalizações provocou uma maior excitação do processo ocasionando melhores estimativas dos parâmetros da função valor principalmente para o LMS.

5.3.2 ADHDP

Nesta parte, a programação dinâmica heurística dependente de ação (ADHDP) baseada nos algoritmos RLS, IPNLMS e PNLMS é aplicada para encontrar a política ótima para o problema do DLQR a partir de amostras *online* dos estado e do custo para o ambiente representado pelas equações (5.1) e (5.13).

5.3.2.1 Experimento

Neste experimento, a convergência da função-Q é verificada não pela trajetória dos seus parâmetros, mas pelos parâmetros da política, os quais estão relacionados aos parâmetros da função-Q e à solução de Riccati.

Parâmetros de Configuração das Simulações: Os seguintes parâmetros para o algoritmo de aprendizagem ADHDP e para o ambiente são definidos:

- O fator de desconto igual a um;
- A técnica de exploração adotada foi injetar um pequeno sinal de ruido na ação de controle;
- A política é melhorada a cada transição de estado;
- A política inicial é dada por

$$\mu_k = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & -1 \end{bmatrix} x_k;$$

- As matrizes Q e R da função de custo são matrizes identidade de ordem 3 e 2, respectivamente;
- O tempo de amostragem do ambiente é de 0,1 s;
- As ações de controle são limitadas ao intervalo $[-5,\ 5]$.

Forma Regressiva da HJB: A parametrização da função de custo e da política para ADHDP podem ser encontradas em (LEWIS; VRABIE, 2009) e são dadas por

$$Q(x_k, \mu_k) = z_k^T H z_k = \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}^T \begin{bmatrix} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{bmatrix} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix} = \phi^T \theta \tag{5.24}$$

e

$$\mu_k = -(H_{\mu\mu})^{-1} H_{\mu x} x_k, \tag{5.25}$$

respectivamente, sendo ϕ a matriz de regressores e θ o vetor de parâmetros. Para o caso específico do ambiente escolhido, H é uma matriz simétrica de ordem 5×5, H_{xx} é matriz

simétrica de ordem 3×3, $H_{x\mu}$ é matriz simétrica de ordem 3×2, $H_{\mu x}$ é matriz simétrica de ordem 2×3, $H_{\mu\mu}$ é matriz simétrica de ordem 2×2 e o vetor $z_k = \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}$ é a concatenação dos vetores de estado e ação.

Então, representado a matriz H por

$$H = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \\ \theta_2 & \theta_6 & \theta_7 & \theta_8 & \theta_9 \\ \theta_3 & \theta_7 & \theta_{10} & \theta_{11} & \theta_{12} \\ \theta_4 & \theta_8 & \theta_{11} & \theta_{13} & \theta_{14} \\ \theta_5 & \theta_9 & \theta_{12} & \theta_{14} & \theta_{15} \end{bmatrix}, \tag{5.26}$$

sua vetorização fica

$$\begin{aligned} vec(H) \quad = \quad \theta \quad = \quad & \Big[\ \theta_1 \quad 2\theta_2 \quad 2\theta_3 \quad 2\theta_4 \quad 2\theta_5 \quad \theta_6 \quad 2\theta_7 \quad 2\theta_8 \quad 2\theta_9 \quad \theta_{10} \\ & \quad 2\theta_{11} \quad 2\theta_{12} \quad \theta_{13} \quad 2\theta_{14} \quad \theta_{15} \ \Big]^T \end{aligned} \tag{5.27}$$

e o vetor de regressores e a ação tomam, respectivamente, as formas

$$\begin{aligned} \phi^T \quad = \quad & \Big[\ x_1^2 \quad x_1x_2 \quad x_1x_3 \quad x_1\mu_1 \quad x_1\mu_2 \quad x_2^2 \quad x_2x_3 \quad x_2\mu_1 \quad x_2\mu_2 \quad x_3^2 \\ & \quad x_3\mu_1 \quad x_3\mu_2 \quad \mu_1^2 \quad \mu_1\mu_2 \quad \mu_2^2 \ \Big] \end{aligned} \tag{5.28}$$

e

$$\mu_k = -(H_{\mu\mu})^{-1} H_{\mu x} x_k = - \begin{bmatrix} \theta_4 & \theta_8 \\ \theta_5 & \theta_9 \end{bmatrix}^{-1} \begin{bmatrix} \theta_{11} & \theta_{13} & \theta_{14} \\ \theta_{12} & \theta_{14} & \theta_{15} \end{bmatrix} x_k = - \begin{bmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \end{bmatrix} x_k. \tag{5.29}$$

O alvo para ADHDP é dado pela própria função-Q corrente mais o custo pela transição de estado.

$$Q(x_k, \mu_k) = r(x_k, \mu_k) + Q\left(f(x_k, \mu_k), \mu(f(x_k, \mu_k))\right) \tag{5.30}$$

$$= x_k^T Q x_k + \mu_k^T R \mu_k + \begin{bmatrix} x_{k+1}^T & \mu_{k+1}^T \end{bmatrix} H \begin{bmatrix} x_{k+1} \\ \mu_{k+1} \end{bmatrix}, \tag{5.31}$$

sendo H a estimativa da matriz H na iteração k' da etapa de melhoramento da política. Pode-se ver que o alvo depende da estimativa atual da matriz H.

A partir das parametrizações da política, da função-Q e do alvo, percebe-se que o algoritmo ADHDP é quase totalmente independe do modelo do ambiente tanto na fase de avaliação da política quanto na etapa de melhoria da política. O algoritmo ADHDP exige unicamente o conhecimento da ordem do sistema.

Estimação da Função Valor: Neste caso, a função-Q tem como parâmetros os elementos da matriz simétrica H de ordem 5×5, logo 15 parâmetros são estimados ao invés de 6 como no caso do HDP, mas por uma questão de fácil referência, mostra-se a convergência dos ganhos da política que são concebidos a partir da matriz H. Os algoritmos utilizados para estimar os parâmetros da função-Q foram os RLS, PNLMS e IPNLMS ajustados segundo a tabela (5.7).

Tabela 5.7: Ajuste dos Parâmetros Livres para RLS, IPNLMS, PNLMS

Algoritmos	Parâmetros Livres				
	ρ	μ	λ	P	α
RLS			0,97	I_n	
PNLMS	1	1,2			
IPNLMS		0,8			-0,6

A função de custo e o fator de desconto foram os mesmo utilizados para o caso HDP, mas diferentemente deste caso, a estratégia de revitalização de estados não foi utilizada, mas sim um ruido de exploração na entrada para manter uma excitação persistente (HAGEN; KRöSE, 1998) (BRADTKE; YDSTIE; BARTO, 1994). Foi mantida, também, a melhoria da política a cada transição de estado.

As figuras abaixo mostram a evolução dos ganhos de retração de estado obtidos durante a simulação e os ganhos de referência calculados em tempo de projeto (em verde) correspondentes. A partir da ampliação das figuras, nota-se um erro da ordem de 10^{-2} para as estimativas dos ganhos da política com os algoritmos PNLMS e IPNLMS. Foi possível diminuir esse erro para ordem de 10^{-3} apenas aumentando o tempo de simulação para 400s ou 4000 iterações.

Figura 5.19: Trajetória seguida pelo Parâmetro k_2 no processo de aprendizagem para 200s de simulação ou 2000 iterações - A figura (b) é uma ampliação da (a) entre os instante 150 a 200s.

(a) Trajetória do Parâmetro k_1. **(b)** Ampliação da figura (a) entre 150 a 200s.

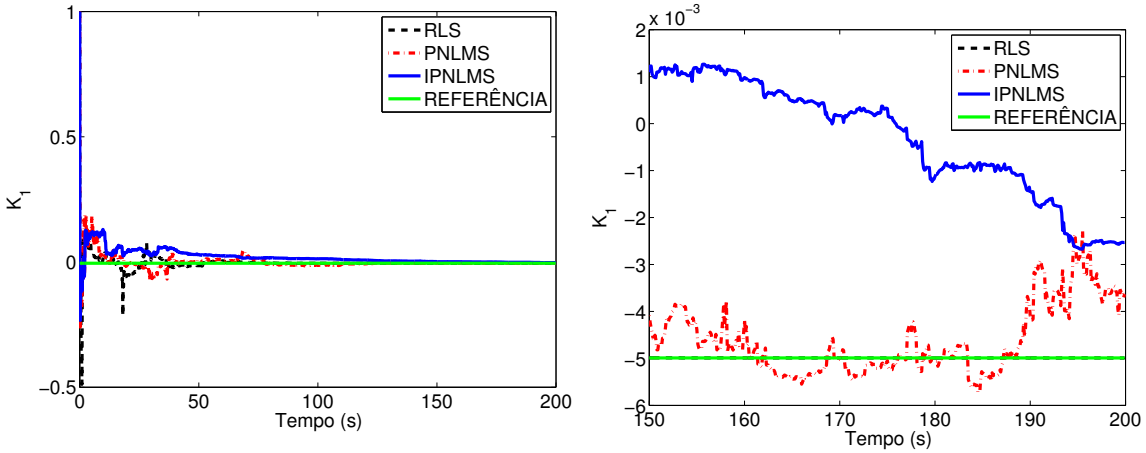
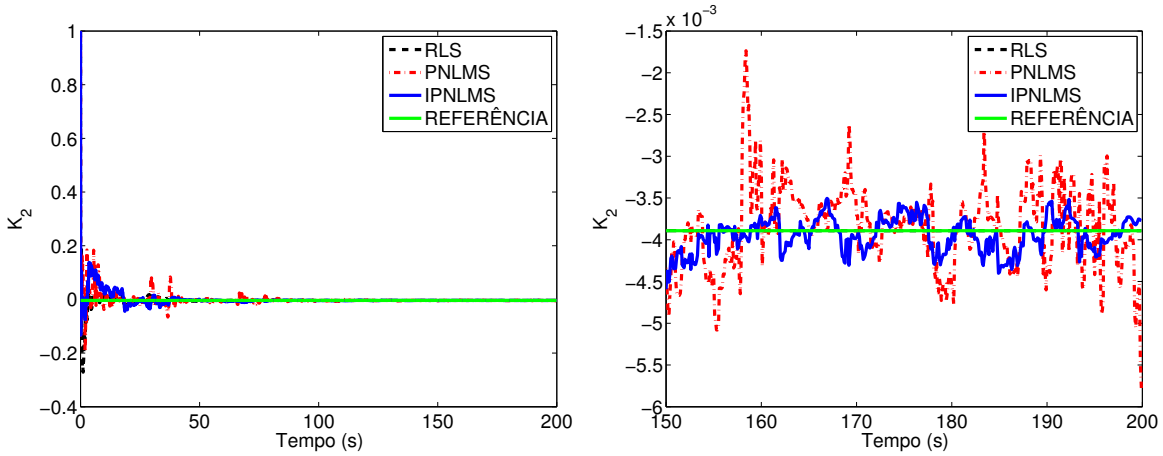


Figura 5.20: Trajetória seguida pelo Parâmetro k_1 no processo de aprendizagem para 200s de simulação ou 2000 iterações - A figura (b) é uma ampliação da (a) entre os instante 150 a 200s.



(a) Trajetória do Parâmetro k_2. **(b)** Ampliação da simulação entre 150 a 200s.

Figura 5.21: Trajetória seguida pelo Parâmetro k_3 no processo de aprendizagem para 200s de simulação ou 2000 iterações - A figura (b) é uma ampliação da (a) entre os instante 150 a 200s.

(a) Trajetória do Parâmetro k_3. **(b)** Ampliação da figura (a) entre 150 a 200s.

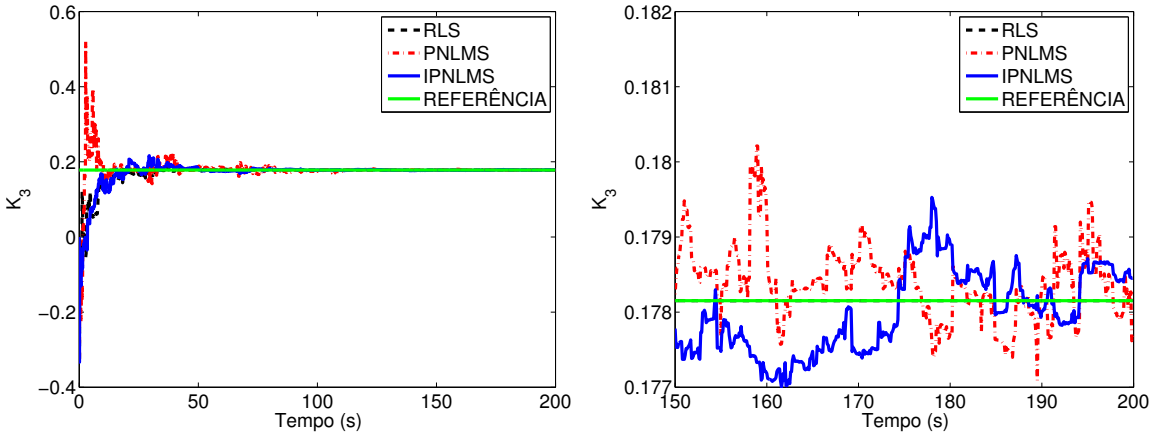


Figura 5.22: Trajetória seguida pelo Parâmetro k_4 no processo de aprendizagem para 200s de simulação ou 2000 iterações - A figura (b) é uma ampliação da (a) entre os instante 150 a 200s.

(a) Trajetória do Parâmetro k_4. **(b)** Ampliação da figura (a) entre 150 a 200s.

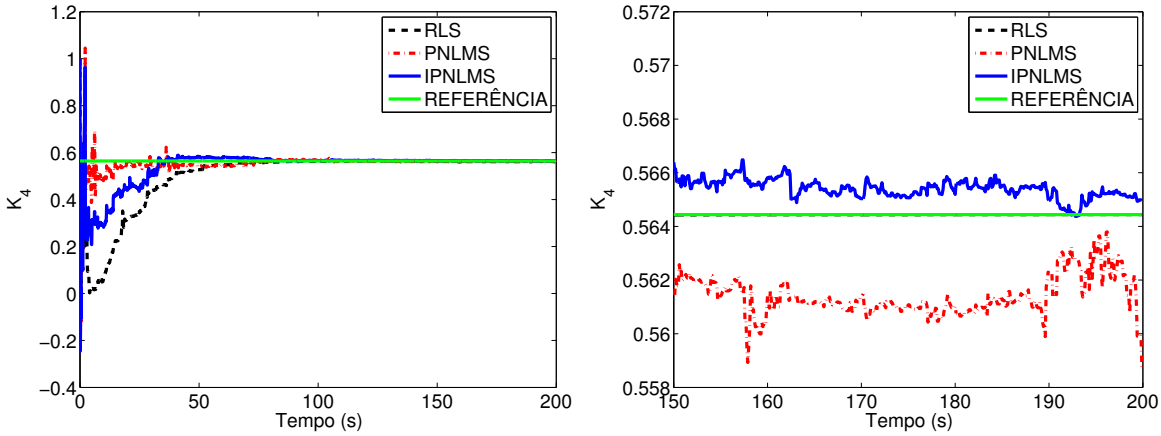


Figura 5.23: Trajetória seguida pelo Parâmetro k_5 no processo de aprendizagem para 200s de simulação ou 2000 iterações - A figura (b) é uma ampliação da (a) entre os instante 150 a 200s.

(a) Trajetória do Parâmetro k_5. **(b)** Ampliação da figura (a) entre 150 e 200s.

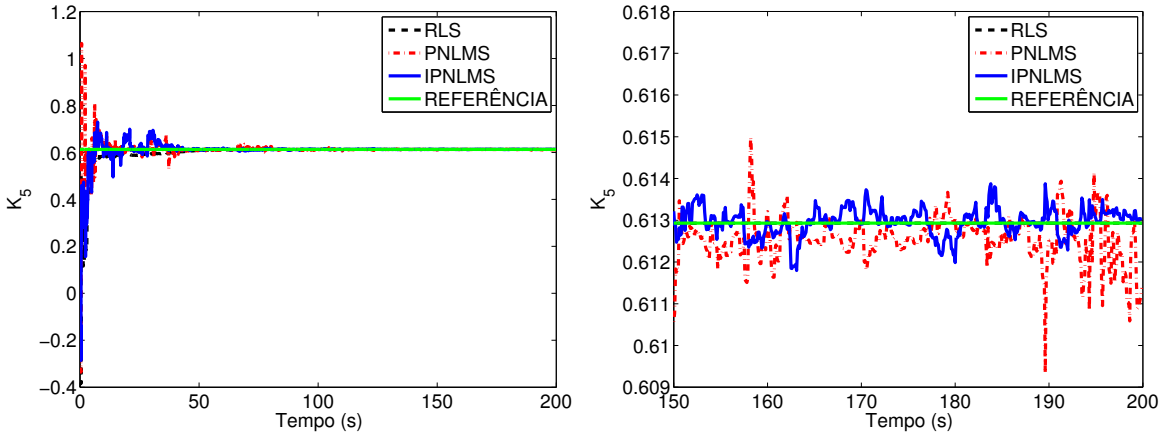
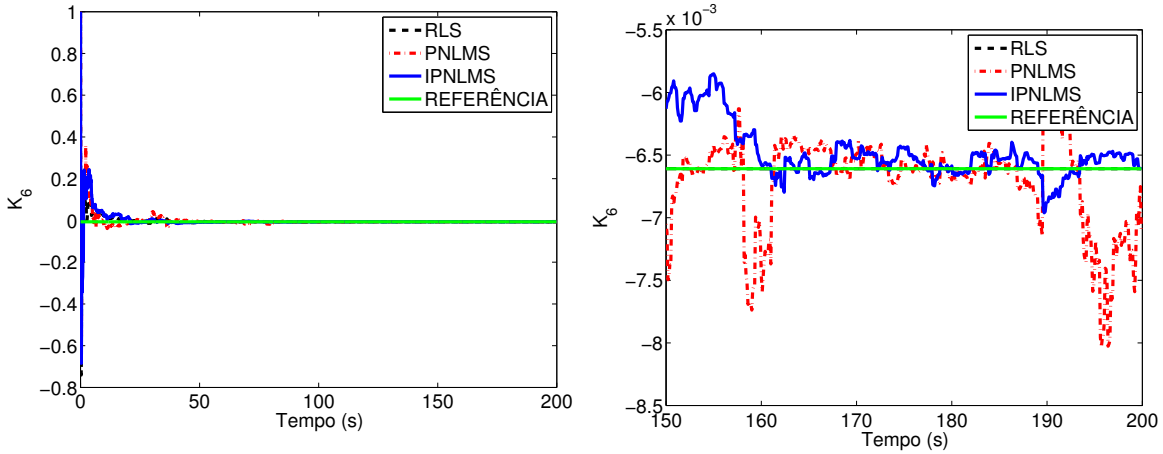


Figura 5.24: Trajetória seguida pelo Parâmetro k_6 no processo de aprendizagem para 200s de simulação ou 2000 iterações - A figura (b) é uma ampliação da (a) entre os instante 150 a 200s.

(a) Trajetória do Parâmetro k_6. **(b)** Ampliação da figura (a) entre 150 a 200s.



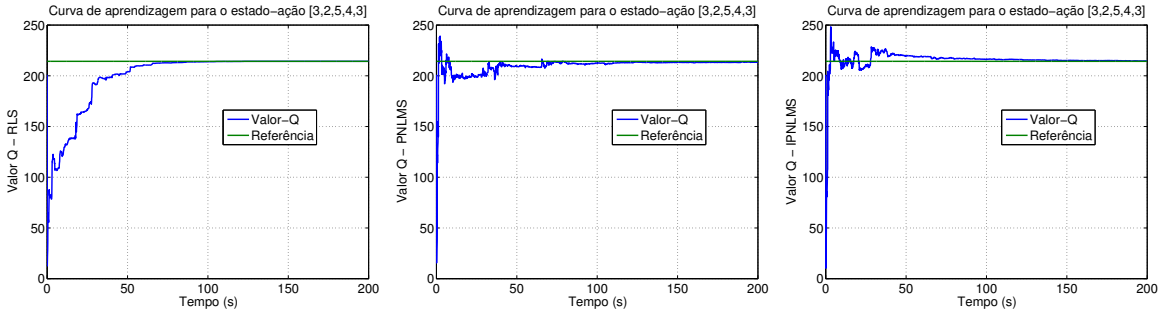
A figura (5.25) mostra as curvas de aprendizagem da função-Q para o par estado-ação $\begin{bmatrix} 3 & 2 & 5 & 4 & 3 \end{bmatrix}^T$.

Figura 5.25: Curva de aprendizagem com o RLS (esquerda), o PNLMS (centro) e o IPNLMS (direita) para o estado-ação $[3 \;\; 2 \;\; 5 \;\; 4 \;\; 3]^T$.

(a) Curva de aprendizagem com o RLS.

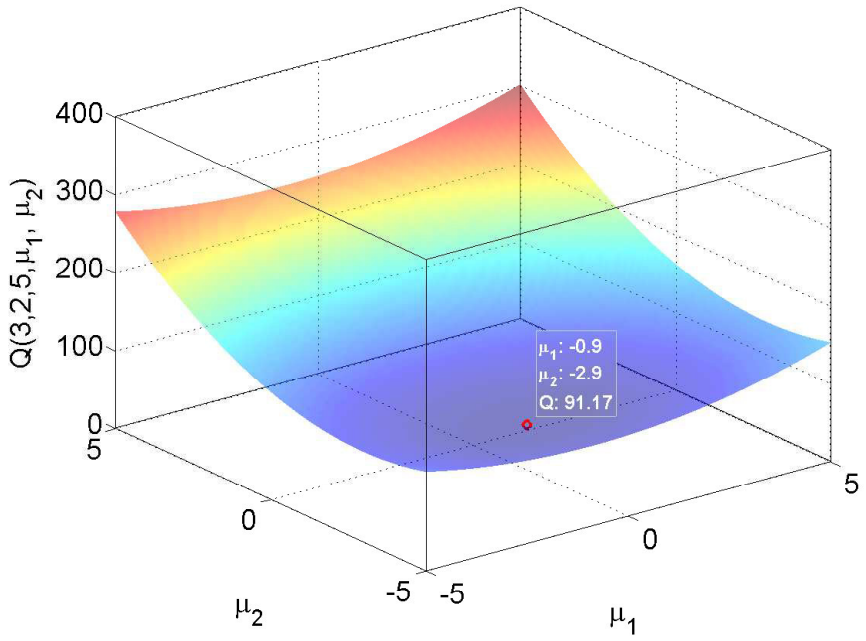
(b) Curva de aprendizagem com o PNLMS.

(c) Curva de aprendizagem com o IPNLMS.



A figura (5.26) mostra a secção transversal da função-Q final feita pelos planos $x_1 = 3$, $x_2 = 2$ e $x_3 = 5$.

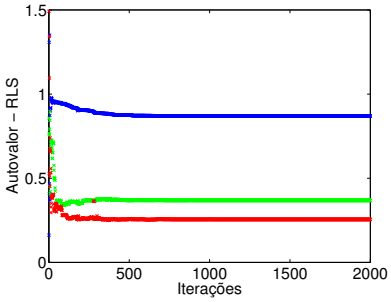
Figura 5.26: Função $Q(3, 2, 5, \mu_1, \mu_2)$.



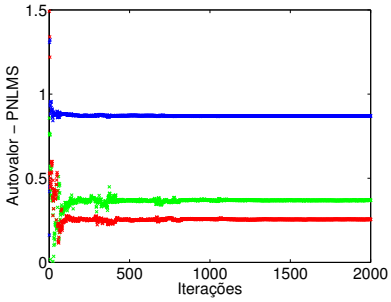
Os gráficos mostrados na figura (5.27) referem-se aos autovalores encontrados durante o processo de aprendizagem com os algoritmos RLS, PNLMS e IPNLMS como estimadores paramétricos. A aprendizagem partiu de uma política que deixa o sistema instável, mas logo ele encontrou uma política estável e não perdeu mais a estabilidade.

Figura 5.27: Autovalores do sistema em malha fechada obtidos durante o processo de aprendizagem.

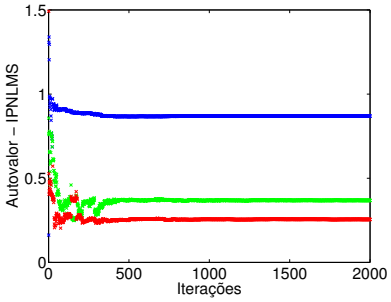
(a) Autovalores com o RLS.



(b) Autovalores com o PNLMS.



(c) Autovalores com o IPNLMS.



Variável de Estado e Esforço de Controle: As políticas encontradas nas iterações 20 e 2000 foram selecionadas e utilizadas para analisar a evolução das trajetórias dos estados e custos para o estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T$. Tais políticas correspondem às matrizes de ganhos de retroação de estado mostradas na tabela (5.8).

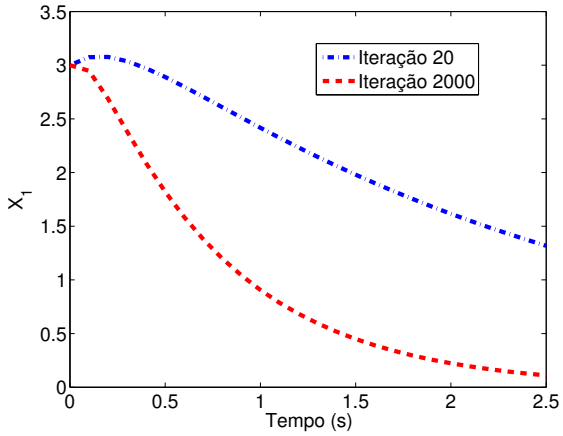
Tabela 5.8: Ganhos das políticas obtidas durante o processo de aprendizagem

Iteração	Ganhos da Política		
	RLS	PNLMS	LMS
20	0,0747 -0,1528 0,0037 0,2285 0,1793 0,0984	-0,0634 -0,1888 0,0755 0,5044 0,4430 0,1707	-0,0591 -0,0793 -0,0153 0,7191 0,3199 0,1257
2000	-0,0050 -0,0039 0,1782 0,5644 0,6129 -0,0066	-0,0036 -0,0058 0,1777 0,5587 0,6114 -0,0071	-0,0025 -0,0038 0,1785 0,5650 0,6130 -0,0066

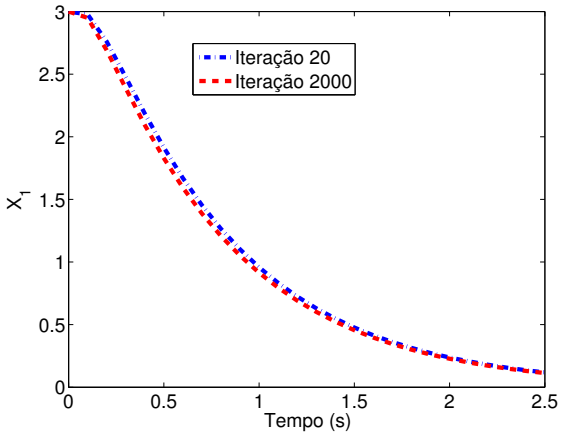
Observa-se na figura (5.28) as trajetórias seguidas pelos estados do sistema do estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T$ ao estado $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$.

Figura 5.28: Trajetórias dos estados do sistema utilizando o RLS (esquerda) e o PNLMS (direita) para ações de controle encontradas nas iterações 20 e 2000.

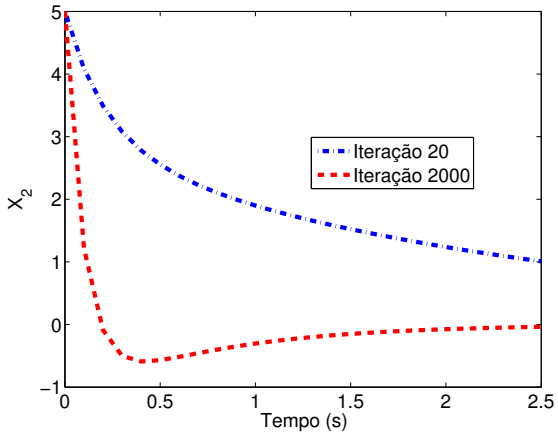
(a) Trajetória de x_1 com o RLS.



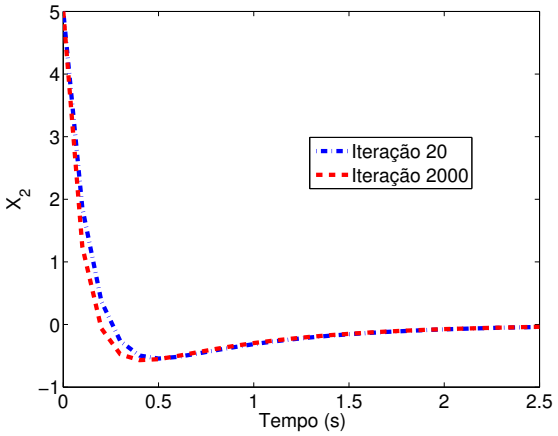
(b) Trajetória de x_1 com o PNLMS.



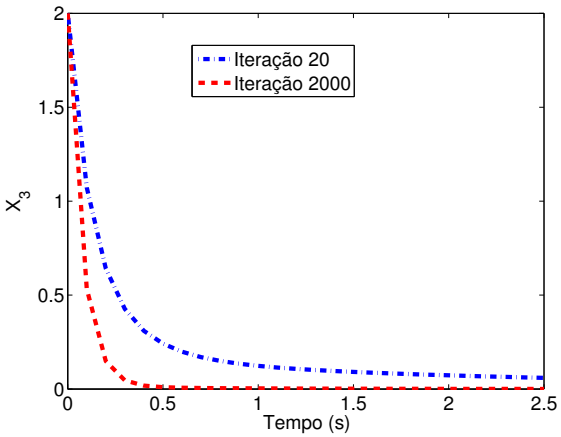
(c) Trajetória de x_2 com o RLS.



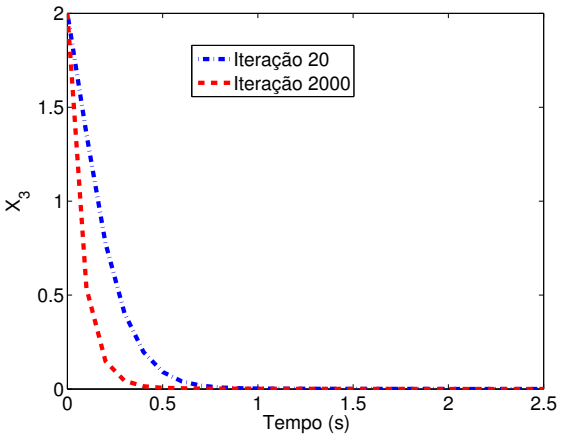
(d) Trajetória de x_2 como PNLMS.



(e) Trajetória de x_3 com o RLS.



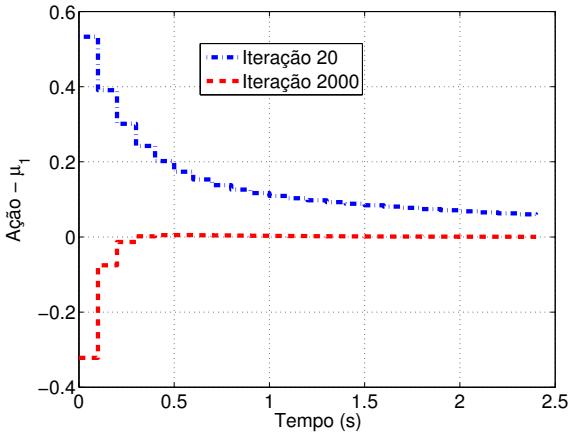
(f) Trajetória de x_3 com o PNLMS.



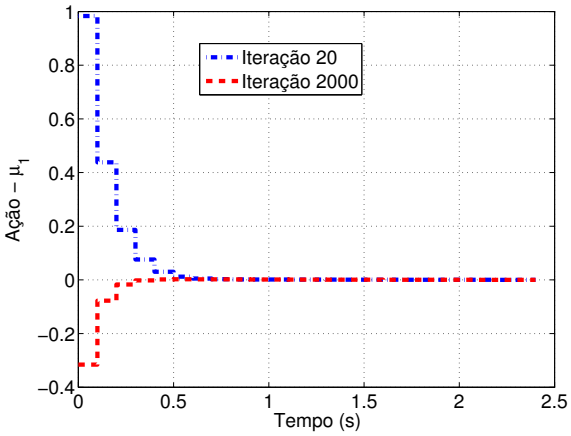
A figura (5.29) mostra as ações de controle usadas para gerar as trajetórias dos estados reproduzidas na figura (5.28).

Figura 5.29: Política encontradas com o RLS (esquerda) e o PNLMS (Direita) na iterações 20 e 2000.

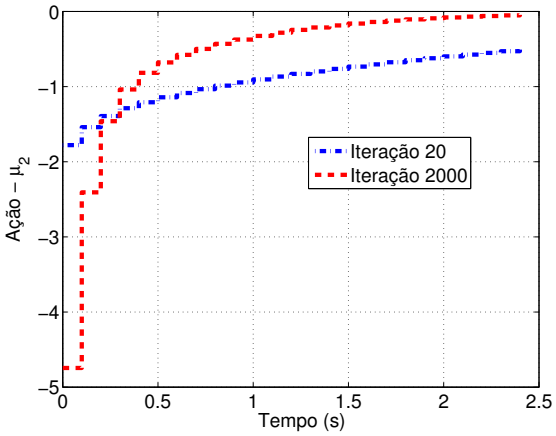
(a) Ações de controle μ_1 com o RLS..



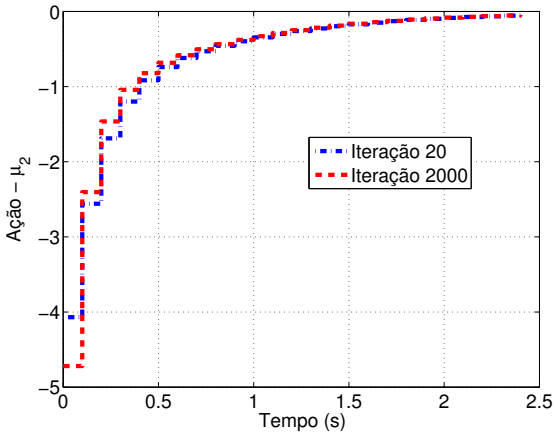
(b) Ações de controle μ_1 com o PNLMS.



(c) Ações de controle μ_2 com o RLS.



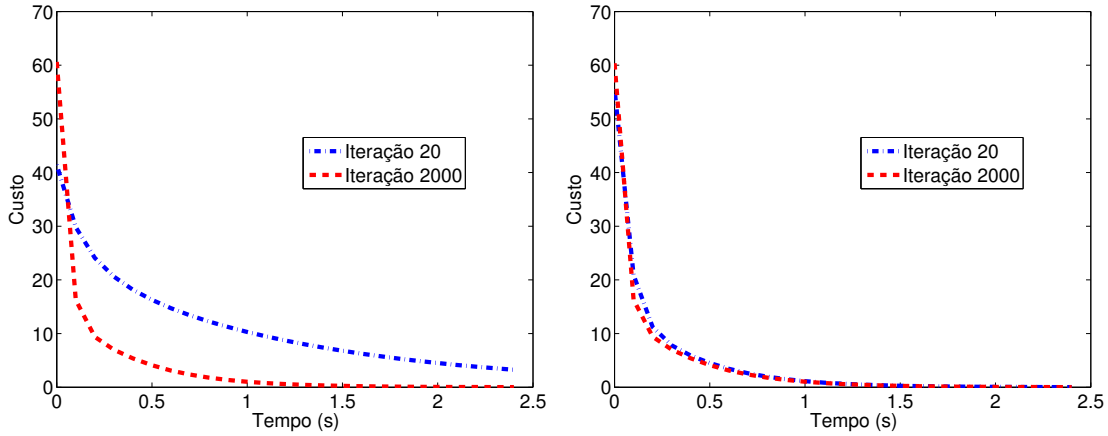
(d) Ações de controle μ_1 com o PNLMS.



Os gráficos representativos das funções de custo para o estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T$ são mostrados na figura (5.30).

Figura 5.30: Custo encontrado com o RLS (esquerda) e com o PNLMS (Direita) para o estado inicial $[3 \quad 2 \quad 5]^T$.

(a) Custos encontrados com o RLS. **(b)** Custos encontrados com o PNLMS.



5.3.2.2 Conclusão

Empregou-se os algoritmos RLS, PNLMS e IPNLMS para estimação da função-Q. Os parâmetros da política convergiram para os valores esperados, solução de Schur. Constatou-se, a partir dos gráficos das trajetórias dos ganhos de retroação, que o algoritmo ADHDP teve um desempenho com PNLMS e IPNLMS inferior ao RLS principalmente em regime. Mesmo assim, as políticas finais encontradas são muitos próximas. Uma forma encontrada para melhorar o desempenho do PNLMS e do IPNLMS foi realizar uma maior exploração nos espaços de ação e de estado e aumentar o tempo de simulação.

Percebe-se que em nenhum momento foi preciso da dinâmica do ambiente para encontrar uma política ótima, porém é necessário saber a ordem do sistema para construir a matriz H.

5.3.3 DHP

Agora, utiliza-se a programação dinâmica heurística dual (DHP) para encontrar uma política ótima que minimiza uma determinada função de custo quadrática através das observações dos gradientes da função de custo e do ambiente. Nesse algoritmo é dependente do modelo do ambiente tanto na fase de avaliação quanto na melhoria da política.

5.3.3.1 Experimento

Faz-se nesse experimento de simulação uma análise geral dos algoritmos DHP baseados em RLS, PNLMS e IAF-PNLMS. Mostra-se o comportamento das estimativas das soluções da equação de Ricatti, avaliação dos esforços de controle e trajetória dos estados para um estado inicial específico.

Parâmetros de Configuração das Simulações: Foram definidos os seguintes parâmetros para o algoritmo de aprendizagem DHP e para o ambiente:

- O fator de desconto igual a um;
- A técnica de exploração adotada foi injetar um sinal de ruido na ação de controle;
- A política é melhorada a cada dez transição de estado;
- As matrizes Q e R da função de custo são matrizes identidade de ordem 3 e 2, respectivamente;
- A política inicial é dada por

$$\mu(x_1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_1;$$

- O tempo de amostragem do ambiente é de 0,1 s;
- As ações de controle são limitadas ao intervalo $[-5,\ 5]$.

A função de custo e o fator de desconto são os mesmo utilizados para o caso do HDP, mas a excitação persistente ou exploração foi feita injetando um ruido na entrada.

Forma Regressiva da HJB: Como foi visto, o DHP é totalmente dependente do modelo do ambiente, pois é necessário, para sua implementação, as derivadas do processo tanto quanto da função de custo.

A parametrização do gradiente da função-V e da política para o DHP em relação ao problema do DLQR são dadas pelas equações

$$\begin{aligned} \frac{\partial V(x_k)}{\partial x} &= \frac{\partial (x^T \otimes x^T)}{\partial x} vet(P) = \phi^T \theta \\ &= 2 \begin{bmatrix} x_1 & x_2 & x_3 & 0 & 0 & 0 \\ 0 & x_1 & 0 & x_2 & x_3 & 0 \\ 0 & 0 & x_1 & 0 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} \end{aligned} \tag{5.32}$$

e

$$\mu(x_k) = Kx_k = (R + B^T P B)^{-1} B^T P A x_k. \tag{5.33}$$

O alvo utilizado pelos algoritmos RLS, PNLMS e IAF-PNLMS para estimar os parâ-

metros da função-V é dado por

$$\frac{\partial V(x_k)}{\partial x} = \frac{\partial r}{\partial x} + \frac{\partial r}{\partial \mu}\frac{\partial \mu}{\partial x} + \frac{\partial V}{\partial f}(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial \mu}\frac{\partial \mu}{\partial x}) \tag{5.34}$$

$$= 2[x_k^T Q + \mu_k^T RK + x_{k+1}^T P(A + BK)], \tag{5.35}$$

sendo K e P o ganho de retroação e matriz de parâmetros da função-V que são atualizados, neste caso, a cada dez iterações da avaliação da política. Pode-se ver que o alvo depende da estimativa atual da matriz P e também do ambiente.

A partir das parametrizações da política, da função valor e do alvo se percebe que o algoritmo DHP depende do modelo do ambiente tanto na etapa de avaliação da política quanto na sua melhoria.

Estimação da Função Valor: Os algoritmos utilizados para estimar os parâmetros da função valor foram ajustados conforme tabela (5.9).

Tabela 5.9: Parâmetros Livres para RLS, IAF-PNLMS, PNLMS

Algoritmos	Parâmetros Livres			
	ρ	μ	λ	P
RLS			0,97	$100I_n$
PNLMS	0,575	1,563		
IAF-PNLMS		1,4		

Foram realizadas 2000 iterações equivalente a 200 segundos de simulação que partiu do estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}$ para as variáveis de estado da aeronave. As figuras (5.31), (5.32), (5.33), (5.34), (5.35) e (5.36) mostram as trajetórias dos parâmetros da função-V e convergência para a solução da DARE. Pode-se ver nas figuras ampliadas que o algoritmo IAF-PNLMS manteve uma pequena oscilação em torno da resposta desejada.

Figura 5.31: Trajetória seguida pelo Parâmetro θ_1 no processo de aprendizagem. A figura (b) é uma ampliação da (a) entre os instantes 150 e 200 segundos.

(a) Trajetória do parâmetro θ_1. **(b)** Ampliação da simulação entre 150 a 200s.

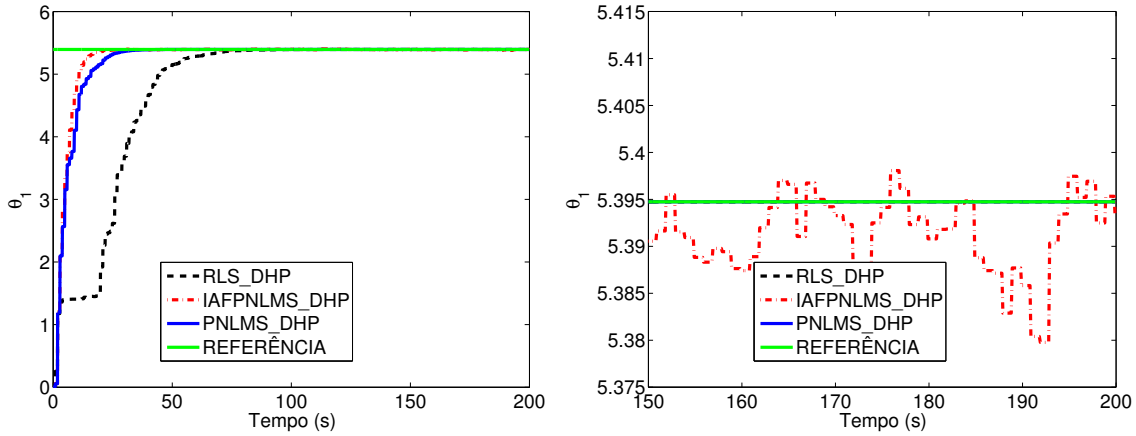


Figura 5.32: Trajetória seguida pelo Parâmetro θ_2 no processo de aprendizagem. A figura (b) é uma ampliação da (a) entre os instantes 150 e 200 segundos.

(a) Trajetória do parâmetro θ_2. **(b)** Ampliação da simulação entre 150 a 200s.

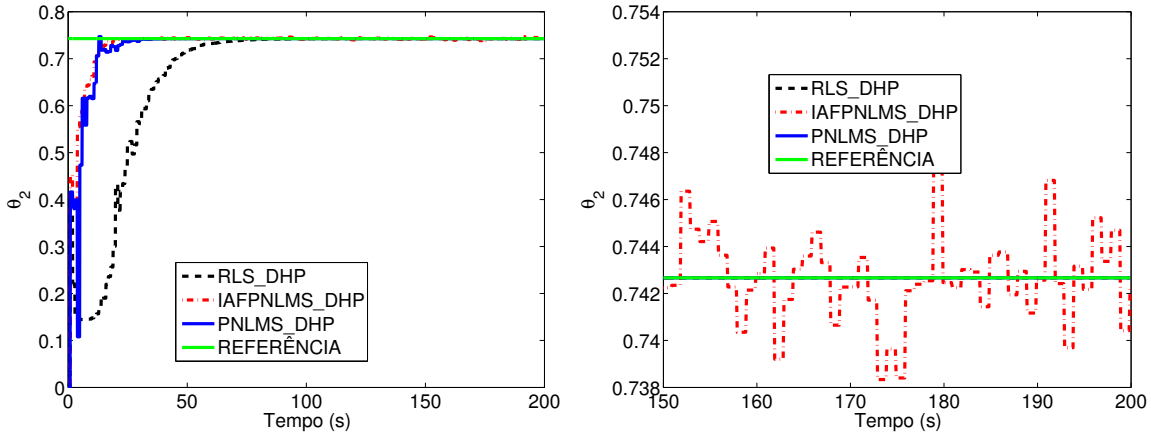


Figura 5.33: Trajetória seguida pelo Parâmetro θ_3 no processo de aprendizagem. A figura (b) é uma ampliação da (a) entre os instantes 150 e 200 segundos.

(a) Trajetória do parâmetro θ_3. **(b)** Ampliação da simulação entre 150 a 200s.

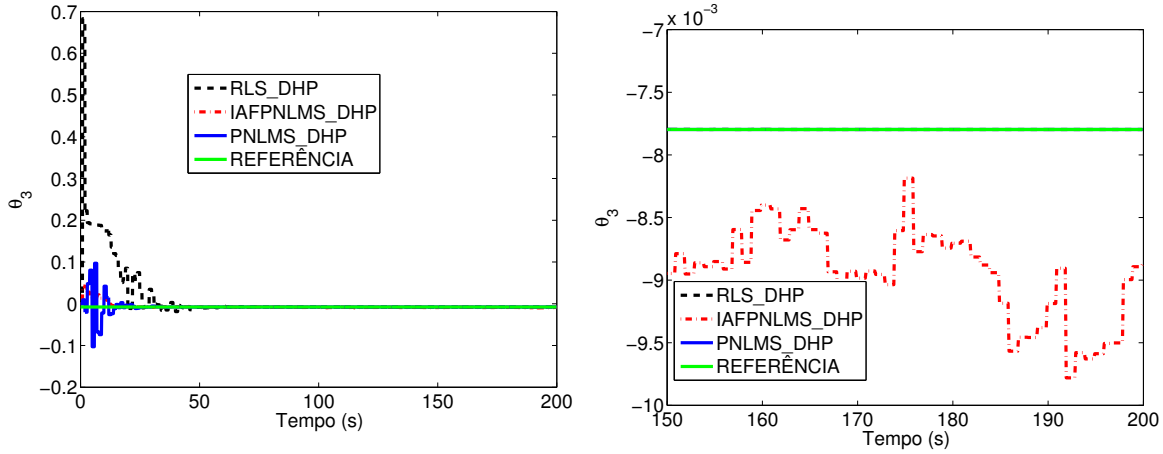


Figura 5.34: Trajetória seguida pelo Parâmetro θ_4 no processo de aprendizagem. A figura (b) é uma ampliação da (a) entre os instantes 150 e 200 segundos.

(a) Trajetória do parâmetro θ_4. **(b)** Ampliação da simulação entre 150 a 200s.

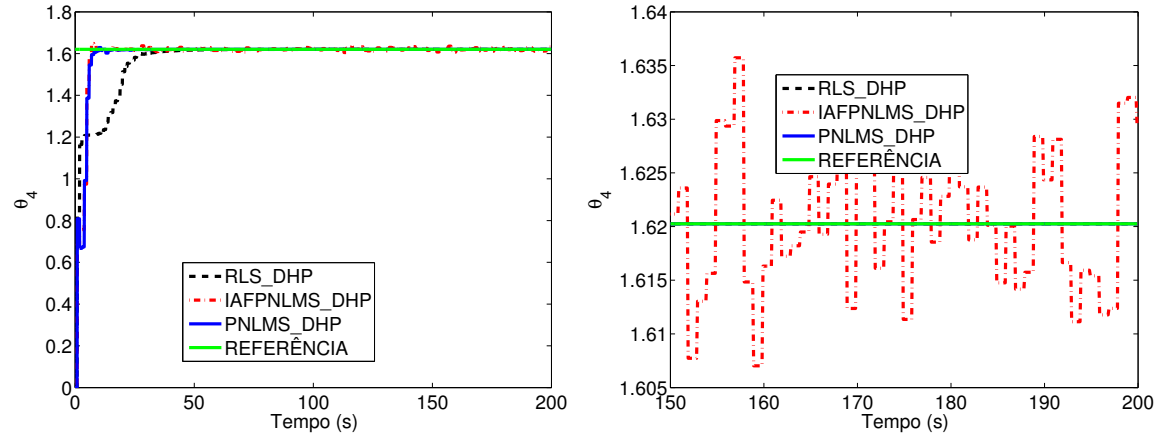


Figura 5.35: Trajetória seguida pelo Parâmetro θ_5 no processo de aprendizagem. A figura (b) é uma ampliação da (a) entre os instantes 150 e 200 segundos.

(a) Trajetória do parâmetro θ_5. **(b)** Ampliação da simulação entre 60 a 200s.

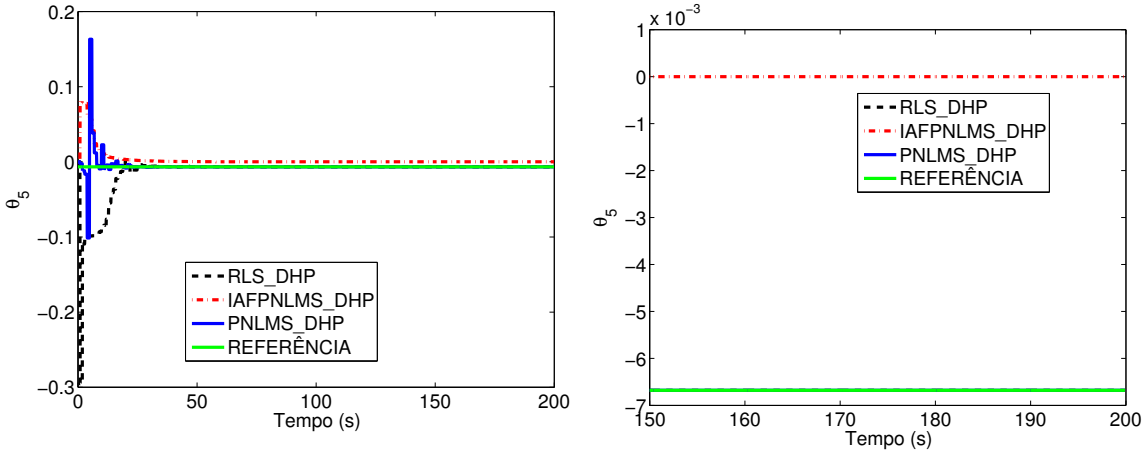
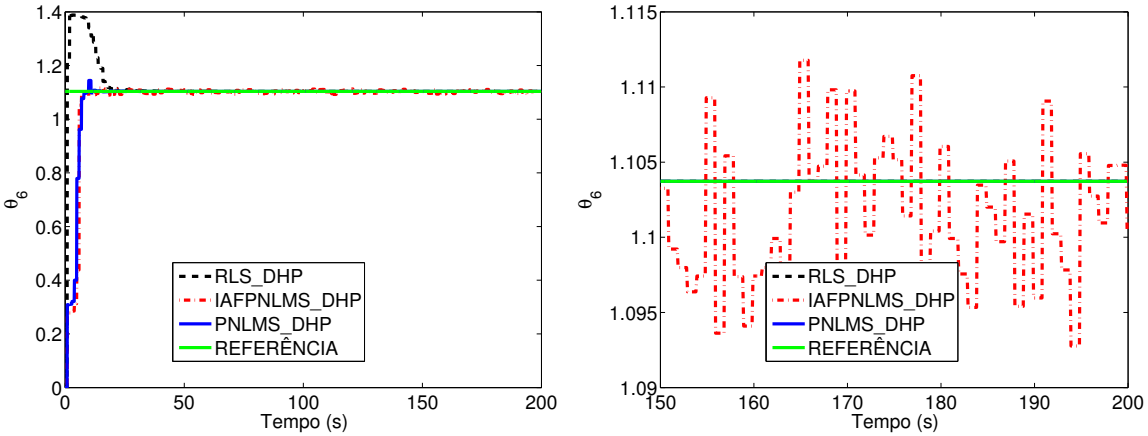


Figura 5.36: Trajetória seguida pelo Parâmetro θ_6 no processo de aprendizagem. A figura (b) é uma ampliação da (a) entre os instantes 150 e 200 segundos.

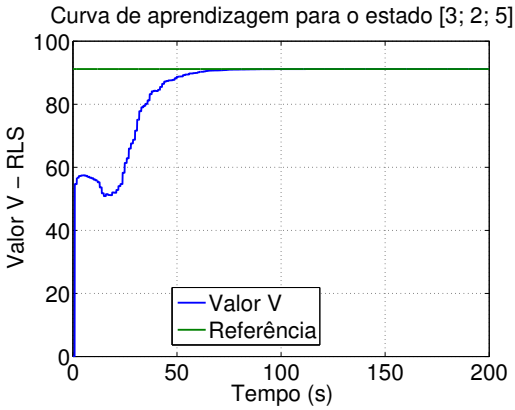
(a) Trajetória do parâmetro θ_6. **(b)** Ampliação da simulação entre 60 a 200s.



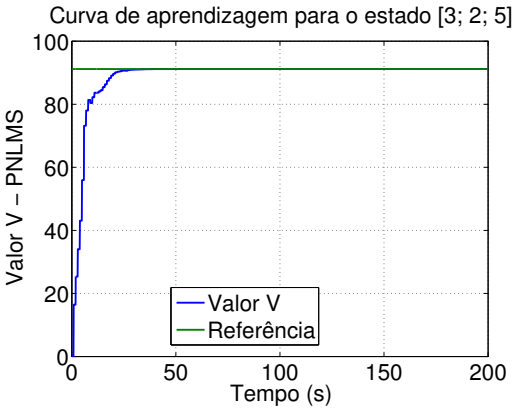
A figura (5.37) mostra as curvas de aprendizagem da função valor para o estado $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T$.

Figura 5.37: Curva de aprendizagem com o RLS (esquerda), o PNLMS (centro) e o IAF-PNLMS (direita) para o estado $[3 \quad 2 \quad 5]^T$.

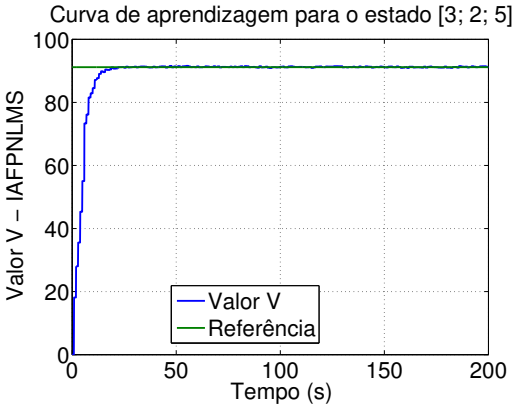
(a) Curva de aprendizagem com o RLS.



(b) Curva de aprendizagem com o PNLMS.



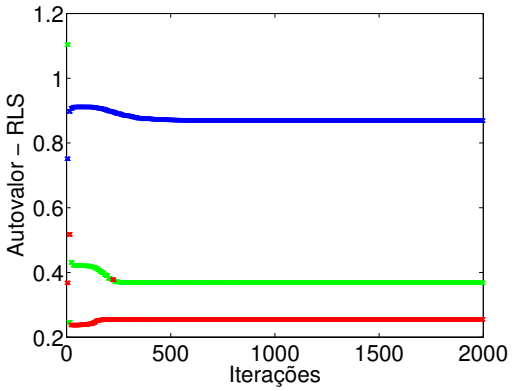
(c) Curva de aprendizagem com o IAF-PNLMS.



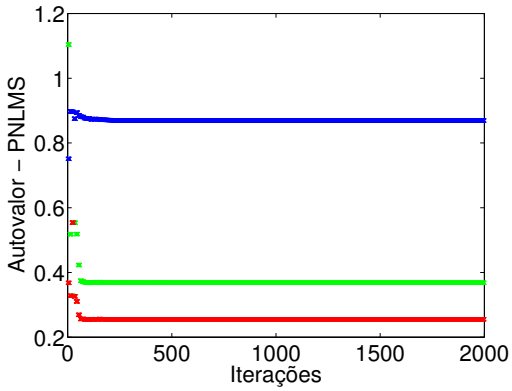
Os gráficos mostrados na figura (5.38) são referentes aos módulos dos autovalores encontrados durante o processo de aprendizagem com os algoritmos RLS, PNLMS e IAF-PNLMS. Constata-se que após a primeira melhoria da política, iteração 11, em nenhum momento o sistema ficou instável.

Figura 5.38: Autovalores do sistema em malha fechada obtidos durante o processo de aprendizagem.

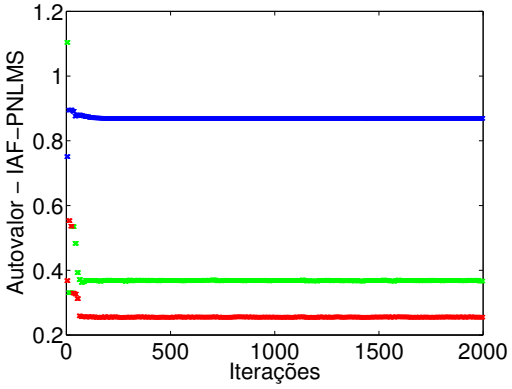
(a) Autovalores com o RLS.



(b) Autovalores com o PNLMS.



(c) Autovalores com o IAF-PNLMS.



Variável de Estado e Esforço de Controle: Selecionou-se dois momentos distintos da fase de aprendizagem para verificar seu andamento e traçou-se as trajetórias representativas das políticas, dos estados e custos tomando como estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T$. Os momentos selecionados foram a iteração onze que corresponde a primeira atualização da política e a iteração 2000 que é a última atualização da política para o algoritmo DHP. A tabela (5.10) mostra os ganhos de retroação de estado encontrados nessas iterações.

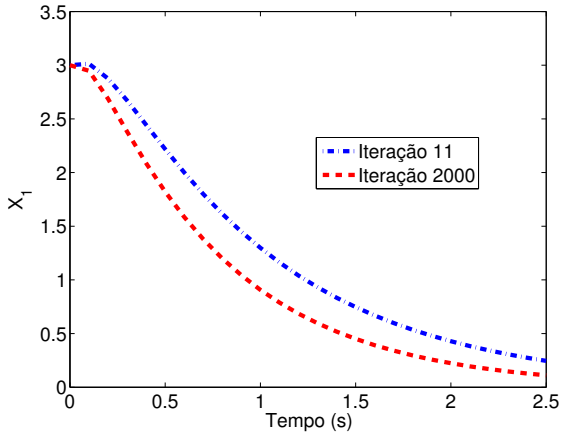
Tabela 5.10: Ganhos ou parâmetros de políticas obtidas nas iterações 11 e 2000 do processo de aprendizagem

Iteração	Ganhos da Política		
	RLS	PNLMS	IAF-PNLMS
11	$\begin{bmatrix} 0.2603 & -0.0477 & 0.1857 \\ 0.3714 & 0.4429 & -0.0396 \end{bmatrix}$	$\begin{bmatrix} 0.0004 & -0.0045 & 0.0633 \\ 0.3778 & 0.4374 & -0.0059 \end{bmatrix}$	$\begin{bmatrix} 0.0068 & 0.0264 & 0.0588 \\ 0.4038 & 0.3941 & 0.0131 \end{bmatrix}$
2000	$\begin{bmatrix} -0.0050 & -0.0039 & 0.1782 \\ 0.5644 & 0.6129 & -0.0066 \end{bmatrix}$	$\begin{bmatrix} -0.0050 & -0.0039 & 0.1782 \\ 0.5644 & 0.6129 & -0.0066 \end{bmatrix}$	$\begin{bmatrix} -0.0058 & -0.0029 & 0.1786 \\ 0.5643 & 0.6135 & -0.0060 \end{bmatrix}$

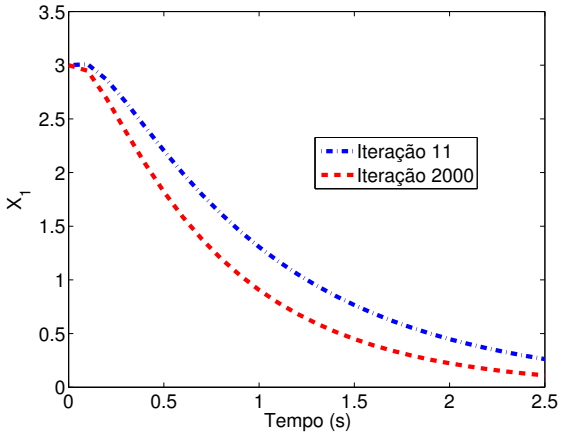
A figura (5.39) mostra as trajetórias representativas seguidas pelos estados do sistema adotando as políticas dadas na tabela (5.10) e estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T$.

Figura 5.39: Trajetórias dos estados do sistema utilizando o IAF-PNLMS (esquerda) e o PNLMS (Direita) para estimar os parâmetros da função de custo.

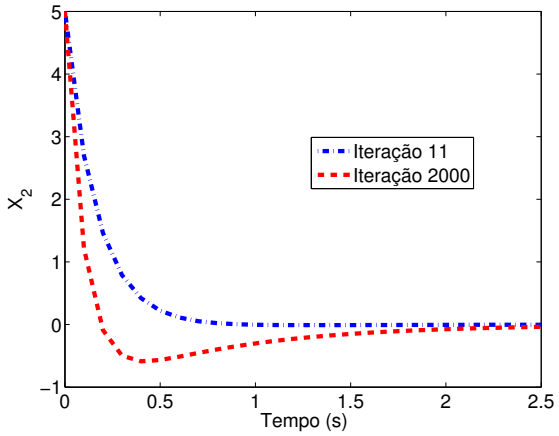
(a) Trajetória de x_1 com o IAF-PNLMS.



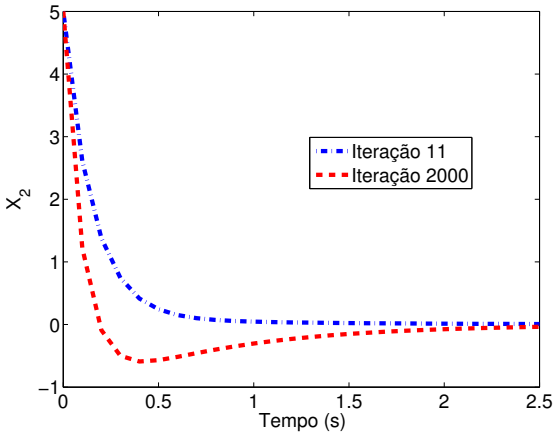
(b) Trajetória de x_1 com o PNLMS.



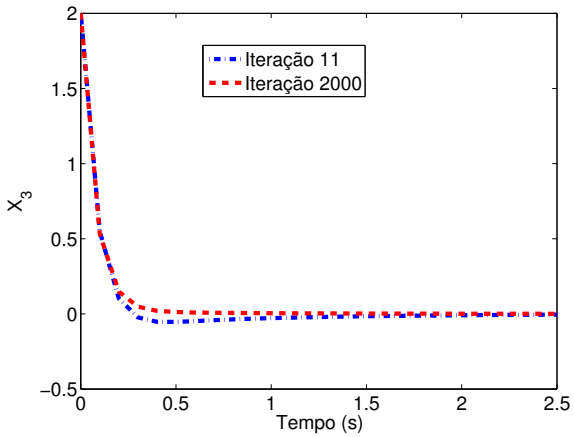
(c) Trajetória de x_2 com o IAF-PNLMS.



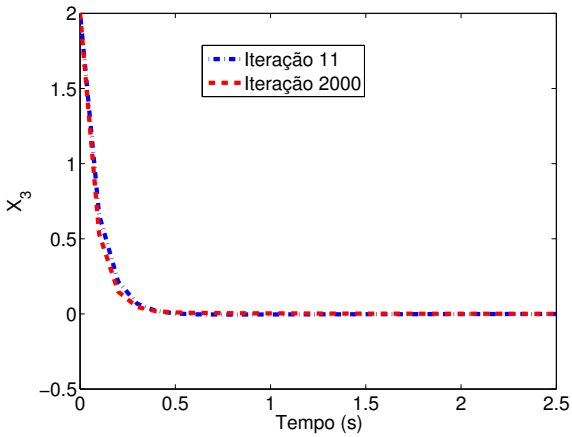
(d) Trajetória de x_2 com o PNLMS.



(e) Trajetória de x_3 com o IAF-PNLMS.



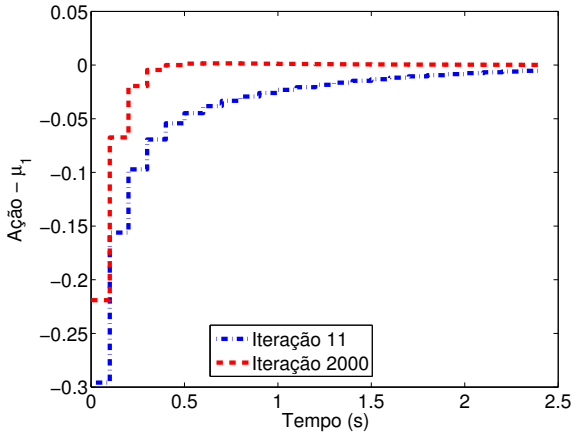
(f) Trajetória de x_3 com o PNLMS.



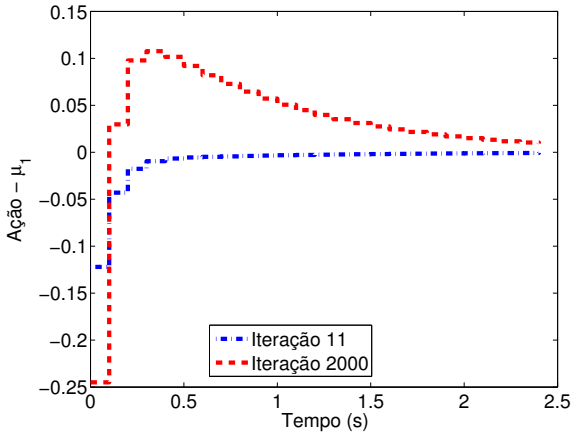
A figura (5.40) mostra as ações de controle para o estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T$ e que foram usadas para gerar as trajetórias de estado reproduzidas na figura (5.39).

Figura 5.40: Política encontradas para o IAF-PNLMS (esquerda) e o PNLMS (Direita).

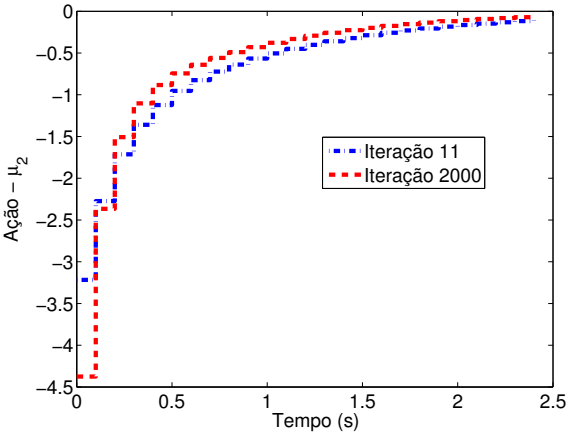
(a) Ações de controle μ_1 com o IAF-PNLMS.



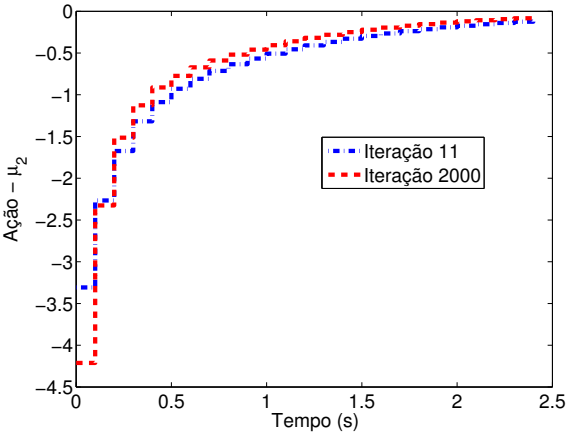
(b) Ações de controle μ_1 com o PNLMS.



(c) Ações de controle μ_2 com o IAF-PNLMS.



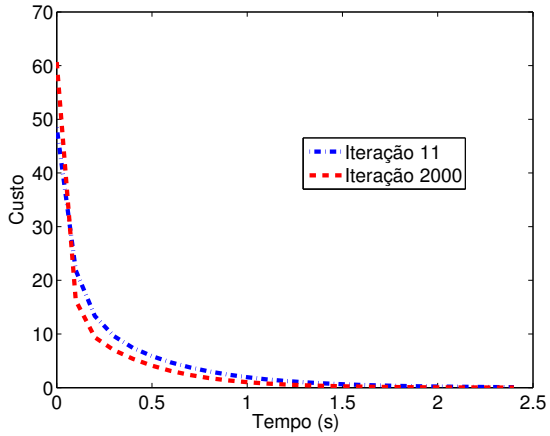
(d) Ações de controle μ_2 com o PNLMS.



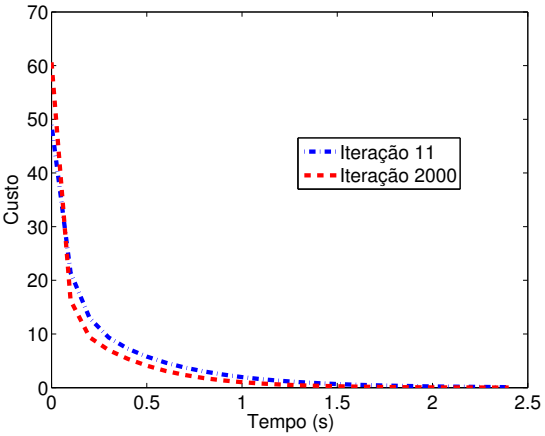
Observa-se na figura (5.41) os gráficos das funções de custo para as políticas indicadas na figura (5.40) e estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T$.

Figura 5.41: Custo encontrado com o IAF-PNLMS (esquerda) e o PNLMS (Direita).

(a) Custos com o IAF-PNLMS.



(b) Custos com o PNLMS.



5.3.3.2 Conclusão

os algoritmos RLS, PNLMS e IAF-PNLMS foram adotados para estimação da função valor estado. Verificou-se que os valores estimados para os parâmetros do gradiente dessa função valor convergiram para os valores reais, solução de Schur. Com o RLS foi obtida uma velocidade de convergência bem menor que com os outros dois estimadores, mas após convergir ele tende ao valor real sem oscilação. Com PNLMS e o IAF-PNLMS foram obtidas alta velocidade de convergência dos parâmetros, mas oscilam em torno do valor real que é uma característica dos algoritmos baseados no LMS. O algoritmo IAF-PNLMS tem uma maior dificuldade na estimação de parâmetros de baixa amplitude chegando a manter um erro em regime, além disso, ele tem uma oscilação maior que o PNLMS.

O algoritmo DHP consegue uma maior velocidade de convergência dos parâmetros da função-V em comparação ao algoritmo HDP, mas a custo de maior conhecimento prévio do ambiente e computacional.

5.3.4 ADDHP

Agora, mostra-se um experimento computacional que utiliza a programação dinâmica heurística dual dependente de ação (ADDHP) para encontrar a ação de controle ótima que estabiliza uma aeronave no seu movimento longitudinal.

5.3.4.1 Experimento

Os parâmetros da ação de controle ótima calculada em tempo de projeto pelo método de Schur foram utilizados como referência para analisar a convergência dos parâmetros das ações de controle encontradas durante a aprendizagem pelo algoritmo ADDHP baseado em NLMS, PNLMS e IAF-PNLMS, os quais são utilizados para estimar os parâmetros da função-Q na etapa de avalização da política.

Parâmetros de Configuração das Simulações: Neste experimento computacional foram definidos os seguintes parâmetros para o algoritmo de aprendizagem ADDHP e para o ambiente:

- O fator de desconto igual a um;
- A técnica de exploração adotada foi injetar um pequeno sinal de ruido na ação de controle;
- As matrizes Q e R da função de custo são matrizes identidade de ordem 3 e 2, respectivamente;
- A política é melhorada a cada transição de estado;

- A política inicial é dada por

$$\mu_1 = \mu(x_1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_1;$$

- O tempo de amostragem do ambiente é de 0,1 segundos;
- As ações de controle são limitadas ao intervalo $[-5,\ 5]$.

Forma Regressiva da HJB: As parametrizações do gradiente da função-Q e da política para ADDHP são dadas por

$$\begin{pmatrix} \frac{\partial Q}{\partial x} & \frac{\partial Q}{\partial \mu} \end{pmatrix} = z_k^T H = \begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \begin{bmatrix} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{bmatrix} = \phi^T \theta \tag{5.36}$$

e

$$\mu_k = -(H_{\mu\mu})^{-1} H_{\mu x} x_k, \tag{5.37}$$

respectivamente, sendo ϕ a matriz de regressores e θ o vetor de parâmetros. Para o caso específico do ambiente adotado, H é uma matriz simétrica de ordem 5×5, H_{xx} é matriz simétrica de ordem 3×3, $H_{x\mu}$ é matriz simétrica de ordem 3×2, $H_{\mu x}$ é matriz simétrica de ordem 2×3, $H_{\mu\mu}$ é matriz simétrica de ordem 2×2 e o vetor $z_k = \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}$ é a concatenação dos vetores de estado e ação.

Então, representado a matriz H por

$$H = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \\ \theta_2 & \theta_6 & \theta_7 & \theta_8 & \theta_9 \\ \theta_3 & \theta_7 & \theta_{10} & \theta_{11} & \theta_{12} \\ \theta_4 & \theta_8 & \theta_{11} & \theta_{13} & \theta_{14} \\ \theta_5 & \theta_9 & \theta_{12} & \theta_{14} & \theta_{15} \end{bmatrix}, \tag{5.38}$$

sua vetorização fica

$$vec(H) = \theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 & \theta_7 & \theta_8 & \theta_9 & \theta_{10} & \theta_{11} & \theta_{12} & \theta_{13} & \theta_{14} & \theta_{15} \end{bmatrix}^T \tag{5.39}$$

e a matriz de regressores e a ação tomam, respectivamente, as formas

$$\phi = 2 \begin{bmatrix} x_1 & x_2 & x_3 & \mu_1 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_1 & 0 & 0 & 0 & x_2 & x_3 & \mu_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_1 & 0 & 0 & 0 & x_2 & 0 & 0 & x_3 & \mu_1 & \mu_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & 0 & 0 & 0 & x_2 & 0 & 0 & x_3 & 0 & \mu_1 & \mu_2 & 0 \\ 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & x_2 & 0 & 0 & x_3 & 0 & \mu_1 & \mu_2 \end{bmatrix} \tag{5.40}$$

e

$$\mu_k = -(H_{\mu\mu})^{-1} H_{\mu x} x_k = -\begin{bmatrix} \theta_4 & \theta_8 \\ \theta_5 & \theta_9 \end{bmatrix}^{-1} \begin{bmatrix} \theta_{11} & \theta_{13} & \theta_{14} \\ \theta_{12} & \theta_{14} & \theta_{15} \end{bmatrix} x_k = -\begin{bmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \end{bmatrix} x_k. \tag{5.41}$$

O alvo utilizado pelos algoritmos PNLMS, NLMS e IAF-PNLMS para estimar os parâmetros da função valor é dado por

$$\begin{aligned} \begin{pmatrix} \frac{\partial Q}{\partial x} & \frac{\partial Q}{\partial \mu} \end{pmatrix} &= \begin{bmatrix} \frac{\partial r}{\partial x} + \frac{\partial Q}{\partial f}\frac{\partial f}{\partial x} + \frac{\partial Q}{\partial \mu}\frac{\partial \mu}{\partial f}\frac{\partial f}{\partial x} & \frac{\partial r}{\partial \mu} + \frac{\partial Q}{\partial f}\frac{\partial f}{\partial \mu} + \frac{\partial Q}{\partial \mu}\frac{\partial \mu}{\partial f}\frac{\partial f}{\partial x} \end{bmatrix} \\ &= 2\left\{ \begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix} \begin{bmatrix} Q_{p\times p} & 0_{p\times q} \\ 0_{q\times p} & R_{q\times q} \end{bmatrix} + \begin{bmatrix} x_{k+1}^T & \mu_{k+1}^T \end{bmatrix} H \begin{bmatrix} A & B \\ KA & KB \end{bmatrix} \right\} \end{aligned} \tag{5.42}$$

sendo H a estimativa dos parâmetros da função-Q no fim do ciclo da avaliação de política. O alvo depende das derivadas parciais do sistema em relação ao estado e ação, consequentemente, o alvo depende do modelo do sistema.

Levando em consideração a equação (5.42), Constata-se que o algoritmo ADDHP é dependente do modelo do ambiente na fase de avaliação da política, mas é independente na etapa de melhoria da política.

Estimação da Função Valor: Neste caso, Deve-se estimar 15 parâmetros da função-Q, mas, por uma questão de fácil referência, mostra-se a convergência dos ganhos das ações de controle que são concebidos a partir da estimativa da matriz H.

Os algoritmos utilizados para estimar os parâmetros da função-Q foram os NLMS, PNLMS e IAF-PNLMS com parâmetros livres ajustados segundo tabela (5.11).

Tabela 5.11: Ajuste dos Parâmetros Livres para NLMS, IAF-PNLMS, PNLMS

Algoritmos	Parâmetros de Ajuste	
	ρ	μ
NLMS		1,6
PNLMS	0,575	1,21
IAF-PNLMS		0,8

As figuras abaixo mostram as trajetórias seguidas pelos ganhos de retração de estados estimados pelos algoritmos NLMS , PNLMS e IAF-PNLMS e os ganhos ótimos de referência (em verde) correspondentes.

Figura 5.42: Trajetória seguida pelo Parâmetro k_1 no processo de aprendizagem

(a) Trajetória seguida por k_1. **(b)** Ampliação da figura (a) entre 0 e 10 s.

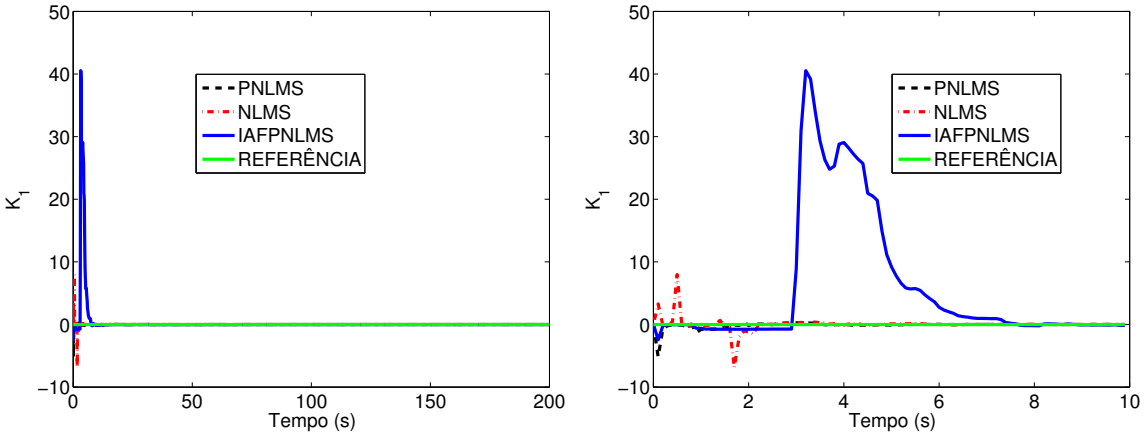


Figura 5.43: Trajetória seguida pelo Parâmetro k_2 no processo de aprendizagem

(a) Trajetória seguida por k_2. **(b)** Ampliação da figura (a) entre 0 e 10 s

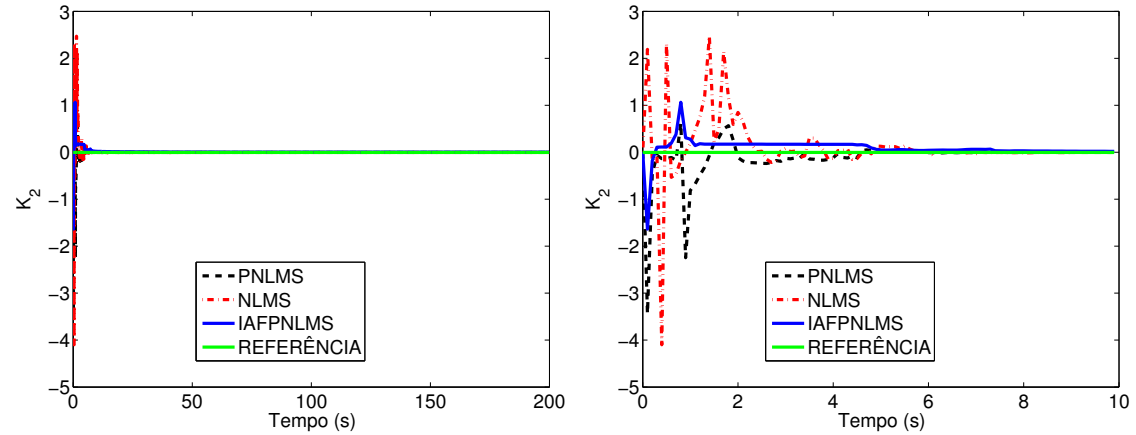


Figura 5.44: Trajetória seguida pelo Parâmetro k_3 no processo de aprendizagem

(a) Trajetória seguida por k_3. **(b)** Ampliação da figura (a) entre 0 e 10 s

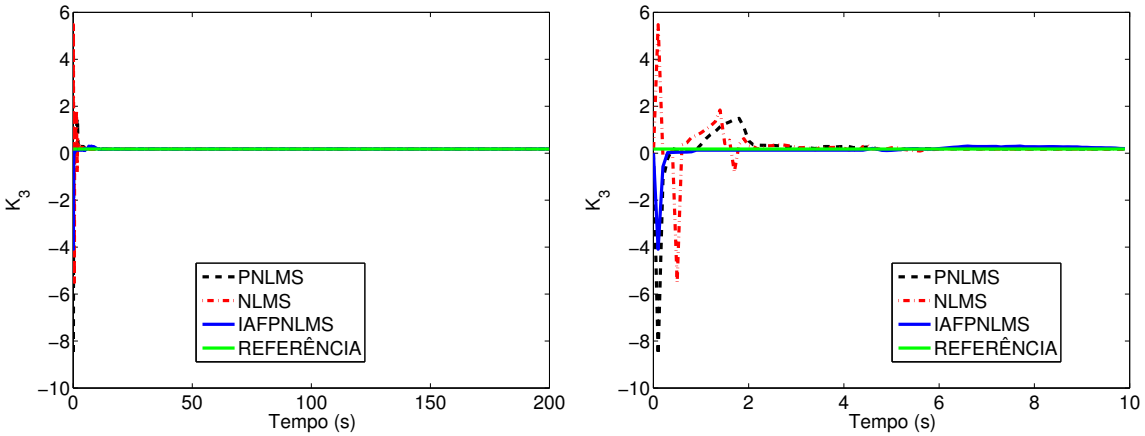


Figura 5.45: Trajetória seguida pelo Parâmetro k_4 no processo de aprendizagem

(a) Trajetória seguida por k_4. **(b)** Ampliação da figura (a) entre 0 e 10 s

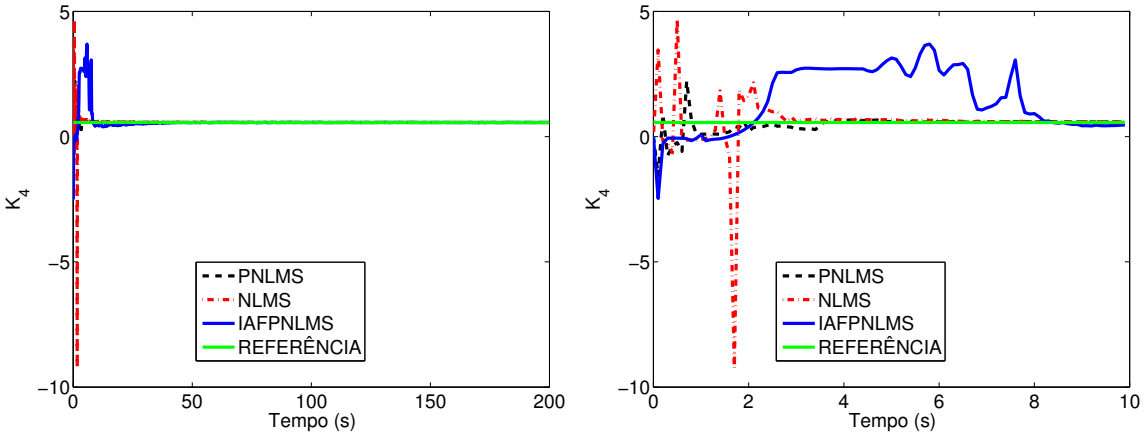


Figura 5.46: Trajetória seguida pelo Parâmetro k_5 no processo de aprendizagem

(a) Trajetória seguida por k_5. **(b)** Ampliação da figura (a) entre 0 e 10 s

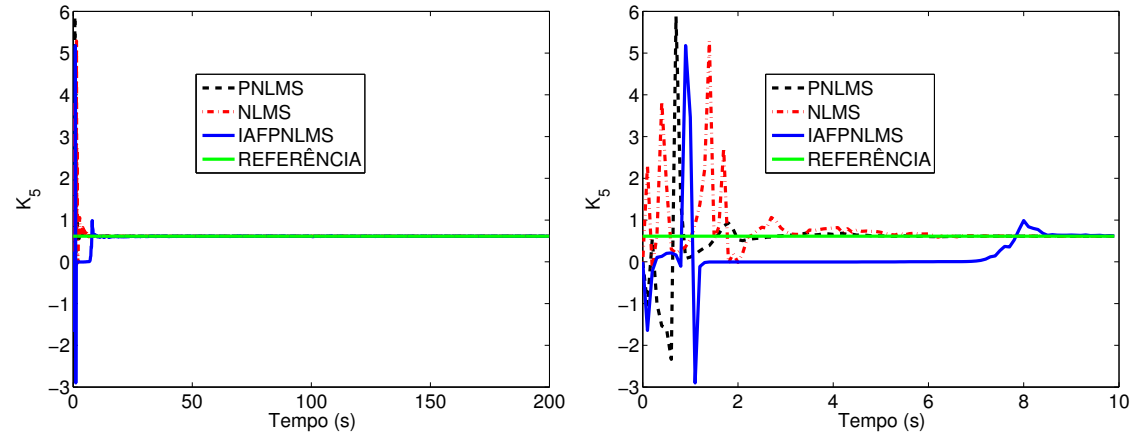
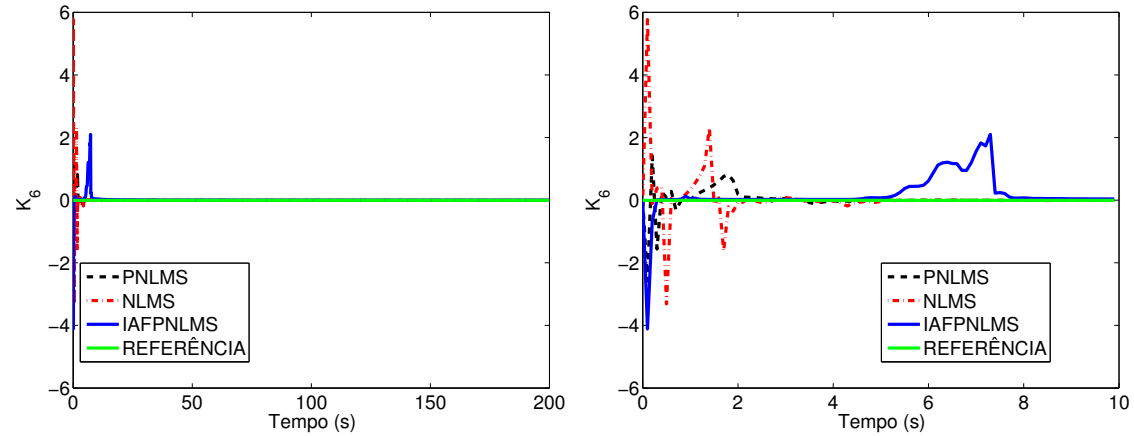


Figura 5.47: Trajetória seguida pelo Parâmetro k_6 no processo de aprendizagem

(a) Trajetória seguida por k_6. **(b)** Ampliação da figura (a) entre 0 e 10 s



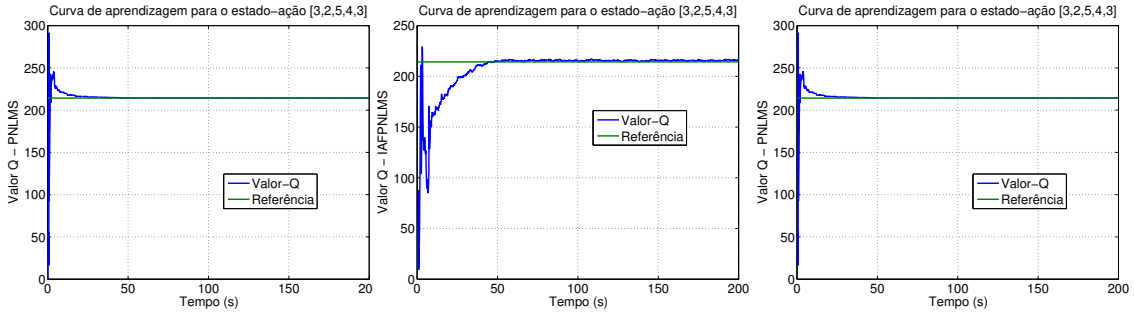
A figura (5.48) mostra as curvas de aprendizagem da função-Q para o par estado-ação $\begin{bmatrix} 3 & 2 & 5 & 4 & 3 \end{bmatrix}^T$.

Figura 5.48: Curva de aprendizagem com o NLMS (esquerda), o IAF-PNLMS (centro) e o PNLMS (direita) para o estado-ação $[3 \;\; 2 \;\; 5 \;\; 4 \;\; 3]^T$.

(a) Curva de aprendizagem como o NLMS.

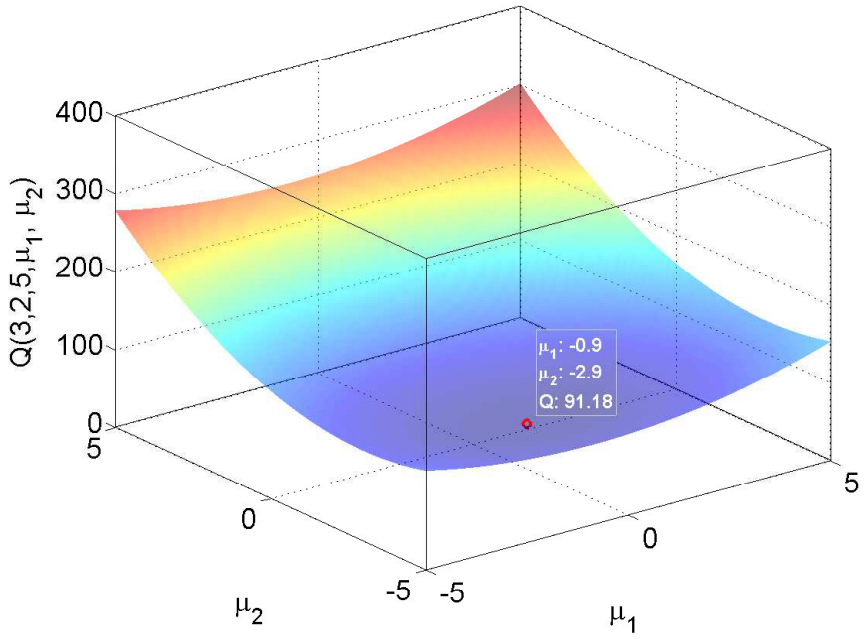
(b) Curva de aprendizagem como o IAF-PNLMS.

(c) Curva de aprendizagem como o PNLMS.



A figura (5.49) mostra a secção transversal da função-Q final para os planos $x_1 = 3$, $x_2 = 2$ e $x_3 = 5$.

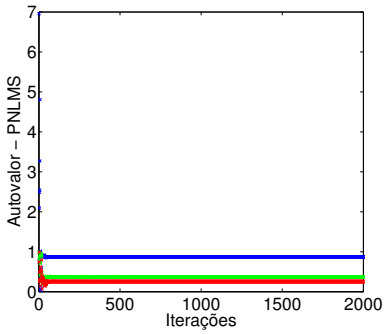
Figura 5.49: Função $Q(3, 2, 5, \mu_1, \mu_2)$.



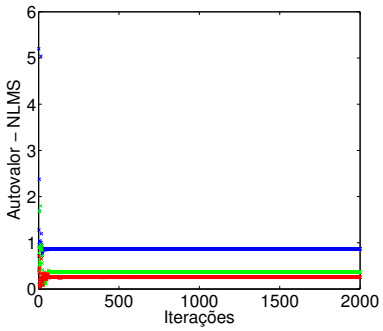
Os gráficos mostrados na figura (5.50) são referentes aos módulos dos autovalores encontrados durante o processo de aprendizagem com os estimadores paramétricos PNLMS, NLMS e IAF-PNLMS. Apenas no inicio da aprendizagem é encontrada políticas instabilizantes.

Figura 5.50: Autovalores do sistema em malha fechada obtidos durante o processo de aprendizagem.

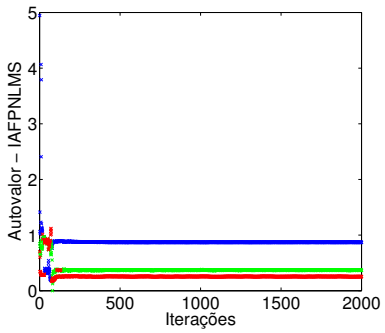
(a) Autovalores com o PNLMS.



(b) Autovalores com o NLMS.



(c) Autovalores com o IAF-PNLMS.



Variável de Estado e Esforço de Controle: A tabela (5.12) mostra os ganhos de retroação de estados obtidos na fase de aprendizagem correspondentes às iterações 30 e 2000 do algoritmo ADDHP.

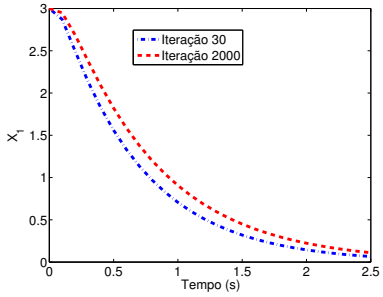
Tabela 5.12: Ganhos ou parâmetros de políticas obtidas durante o processo de aprendizagem

Iteração	Ganhos da Política		
	PNLMS	NLMS	IAF-PNLMS
30	$\begin{bmatrix} 0{,}2067 & -0{,}1932 & 0{,}2558 \\ 0{,}3761 & 0{,}6220 & 0{,}0455 \end{bmatrix}$	$\begin{bmatrix} 0{,}1413 & 0{,}0249 & 0{,}2997 \\ 0{,}7834 & 0{,}8151 & 0{,}0635 \end{bmatrix}$	$\begin{bmatrix} -0{,}7286 & 0{,}1707 & 0{,}1292 \\ 2{,}5630 & -0{,}0036 & 0{,}0238 \end{bmatrix}$
2000	$\begin{bmatrix} -0{,}0050 & -0{,}0039 & 0{,}1782 \\ 0{,}5644 & 0{,}6129 & -0{,}0066 \end{bmatrix}$	$\begin{bmatrix} -0{,}0050 & -0{,}0039 & 0{,}1782 \\ 0{,}5645 & 0{,}6129 & -0{,}0066 \end{bmatrix}$	$\begin{bmatrix} -0{,}0128 & 0{,}0020 & 0{,}1798 \\ 0{,}5640 & 0{,}6129 & 0{,}0003 \end{bmatrix}$

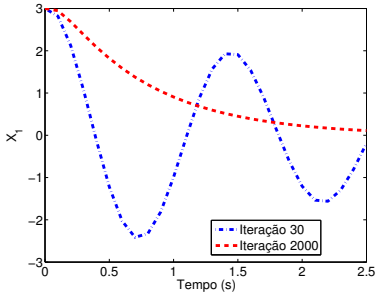
A figura (5.51) exibe as trajetórias seguidas pelos estados do sistema quando estes iniciam em $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}$.

Figura 5.51: Trajetórias dos estados do sistema utilizando o NLMS (esquerda), o IAF-PNLMS (centro) e o PNLMS (direita) para o estado inicial $[3, \quad 2, \quad 5]^T$.

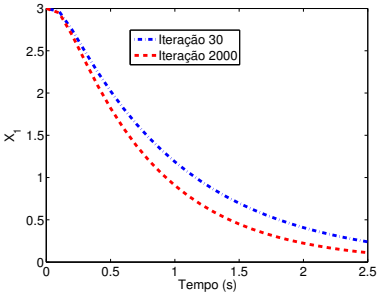
(a) Trajetória de x_1 com o NLMS.



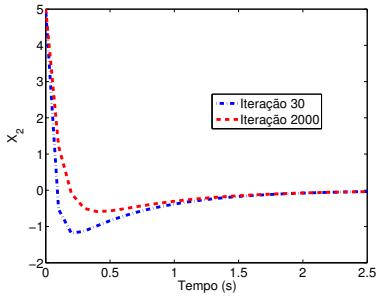
(b) Trajetória de x_1 com o IAF-PNLMS.



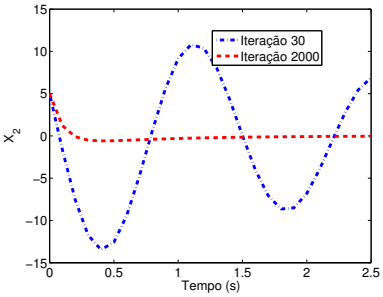
(c) Trajetória de x_1 com o PNLMS.



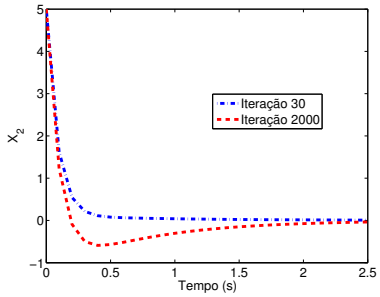
(d) Trajetória de x_2 com o NLMS.



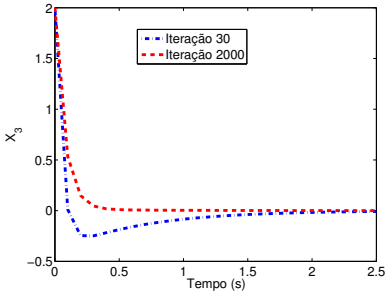
(e) Trajetória de x_2 com o IAF-PNLMS.



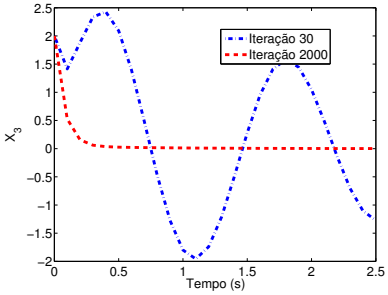
(f) Trajetória de x_2 com o PNLMS.



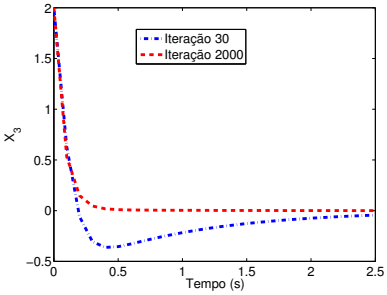
(g) Trajetória de x_3 com o NLMS.



(h) Trajetória de x_3 com o IAF-PNLMS.



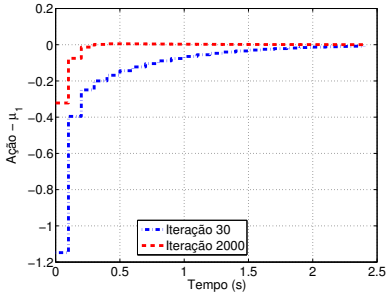
(i) Trajetória de x_3 com o PNLMS.



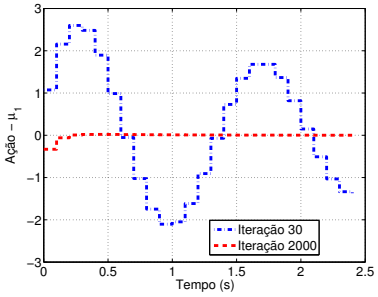
A figura (5.52) mostra as duas ações de controle para o estado inicial $\begin{bmatrix} 3 & 2 & 5 \end{bmatrix}$ e que foram usadas para gerar as trajetórias reproduzidas na figura (5.51).

Figura 5.52: Política encontradas com o NLMS (esquerda) e o IAF-PNLMS (centro) e o PNLMS (Direita) para o estado inicial $[3, \quad 2, \quad 5]^T$.

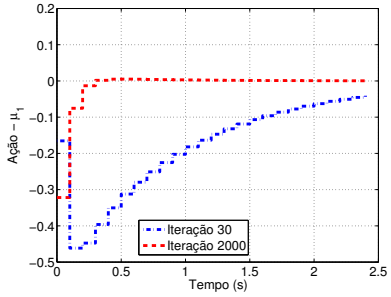
(a) Ação μ_1 encontradas com NLMS.



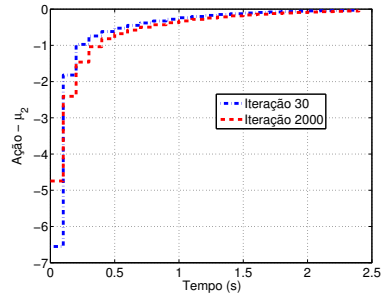
(b) Ação μ_1 encontradas com IAF-PNLMS.



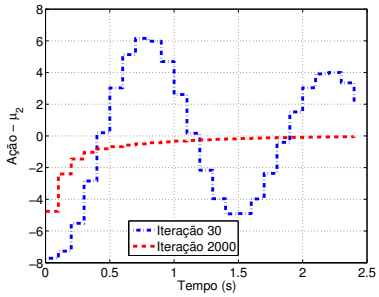
(c) Ação μ_1 encontradas com PNLMS.



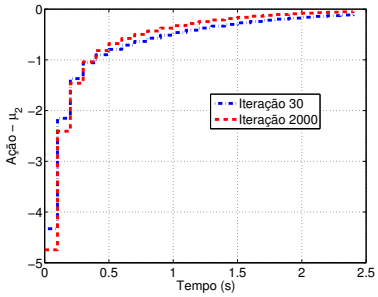
(d) Ação μ_2 encontradas com NLMS.



(e) Ação μ_2 encontradas com IAF-PNLMS.



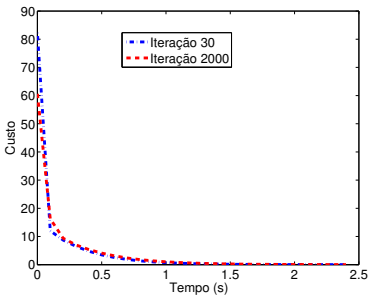
(f) Ação μ_2 encontradas com PNLMS.



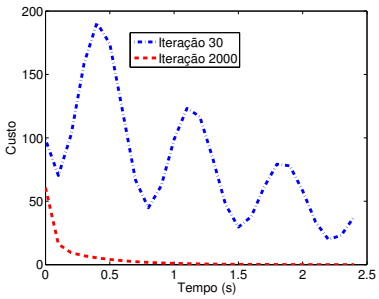
É exibido na figura (5.53) os gráficos das funções de custo para as ações de controle indicadas na figura (5.52).

Figura 5.53: Custo encontrado com o NLMS (esquerda), o IAF-PNLMS (centro) e o PNLMS (direita) para o estado inicial $[3, \quad 2, \quad 5]^T$.

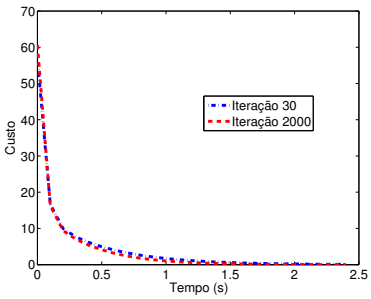
(a) Custos encontrados com o NLMS.



(b) Custos encontrados com o IAF-PNLMS.



(c) Custos encontrados com o PNLMS.



5.3.4.2 Conclusão

Na estimativa dos parâmetros da função valor utilizando os algoritmos PNLMS, NLMS e AIFPNLMS foram obtidos bons resultados, sendo que o desempenho do IAF-PNLMS foi um pouco inferior aos demais.

Dos algoritmos dependente de ação, o ADDHP foi o que teve melhor desempenho em encontrar uma política ótima para o problema do DLQR. No entanto, foi necessário maior custo computacional e conhecimento do ambiente.

5.4 Aplicação dos Algoritmos ao Circuito Elétrico

Nesta seção, os algoritmos HDP, DHP, ADHDP e ADDHP são aplicados novamente com parametrizações dada pelo DLQR para obter uma ação de controle ótima para o circuito elétrico representado pela figura (5.2), lembrando que o circuito tem baixo grau de esparsidade e é de quarta ordem.

Uma planta que sofre mudança em seus parâmetros é utilizada com os algoritmos HDP e ADHDP. Com o DHP e o ADDHP, a ordem do processo é aumentada com integradores em sua entrada para eliminar erro em regime em seguir referência em degrau.

Para o HDP e ADHDP, os valores dos componentes do circuito são inicialmente $C_1 = C_2 = 50\mu F, L_1 = L_2 = 30\mu H,\ R_1 = R_2 = 1K\Omega$ e $R_3 = 2K\Omega$ e são alterados em algum momento para $C_1 = C_2 = 59\mu F,\ L_1 = L_2 = 39\mu H,\ R_1 = 1K\Omega,\ R_2 = 1,1K\Omega$ e $R_3 = 2K\Omega$.

A solução da Equação de Riccati do sistema em malha fechada, com tempo de amostragem de $0,01$ s, utilizando Schur, é dado por

$$P = \begin{bmatrix} 3,0396 & -0,5172 & 0,1194 & -0,1192 \\ -0,5172 & 3,0396 & -0,1192 & 0,1194 \\ 0,1194 & -0,1192 & 1,4234 & -0,4234 \\ -0,1192 & 0,1194 & -0,4234 & 1,4234 \end{bmatrix}, \tag{5.43}$$

antes da mudança de parâmetros, e por

$$P = \begin{bmatrix} 2,6594 & 0,2845 & 0,1881 & -0,1878 \\ 0,2845 & 2,6862 & -0,1841 & 0,1844 \\ 0,1881 & -0,1841 & 1,2877 & -0,2877 \\ -0,1878 & 0,1844 & -0,2877 & 1,2877 \end{bmatrix}, \tag{5.44}$$

após mudanças de parâmetros, que são as referências usadas para analisar o comportamento do sistema de aprendizagem.

O objetivo dos sistemas de aprendizagem é levar a zero ou para os valores de referência as tensões e correntes que representam os estados do sistema de forma a minimizar uma dada função de custo.

5.4.1 HDP

Neste experimento, os algoritmos de estimação da função-V empregados na simulação do HDP para o avião também são utilizados aqui, ou seja, os algoritmos RLS, PNLMS e LMS.

5.4.1.1 Experimento

Considera-se aqui um modelo que sofre uma mudança nos seu parâmetros, conforme exporto acima. Uma política inicial escolhida aleatoriamente é adotada, sendo atualizada a cada iteração do HDP. O processo de aprendizagem para quando alguma política satisfaz uma determinada condição e é reiniciando quando uma nova mudança no comportamento da planta ocorre.

Parâmetros de Configuração das Simulações: Define-se os seguintes parâmetros do processo e do algoritmo de aprendizagem HDP:

- O fator de desconto da função valor foi definido igual a um;
- O retorno para cada estado inicial é estimado para cada cinco passos de simulação da trajetória;
- Após cinco iterações, a trajetória dos estados é reiniciada em um estado selecionado aleatoriamente no espaço de estado
- A política é melhorada a cada transição de estado;
- Foram realizadas 6000 iterações equivalente a 60 segundos de simulação.
- A política inicial adota é dada por

$$\mu_1 = \begin{bmatrix} 0,0619 & -0,0963 & -0,1972 & 0,1972 \\ -0,0963 & 0,06192 & 0,1972 & -0,1972 \end{bmatrix} x_1;$$

- O tempo de amostragem de $0,01$ s.
- Função de custo definida pela equação (5.13);
- São realizadas 6000 iterações equivalente a 60 segundos de simulação;
- Na iteração 3000 a planta sofre uma variação brusca em seus parâmetros;

Forma Regressiva da HJB: Agora como o circuito tem quatro estados é necessário estimar 10 parâmetros $((4+1)\times 4/2)$. A função valor é parametrizada da seguinte forma

$$V(x_k) = \left[\begin{array}{cccccccccc} x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 & x_2^2 & x_2x_3 & x_2x_4 & x_3^2 & x_3x_4 & x_4^2 \end{array}\right] \left[\begin{array}{c} \theta_1 \\ 2\theta_2 \\ 2\theta_3 \\ 2\theta_4 \\ \theta_5 \\ 2\theta_6 \\ 2\theta_7 \\ \theta_8 \\ 2\theta_9 \\ \theta_{10} \end{array}\right], \quad (5.45)$$

sendo

$$P = \left[\begin{array}{cccc} \theta_1 & \theta_2 & \theta_3 & \theta_4 \\ \theta_2 & \theta_5 & \theta_6 & \theta_7 \\ \theta_3 & \theta_6 & \theta_8 & \theta_9 \\ \theta_4 & \theta_7 & \theta_9 & \theta_{10} \end{array}\right], \quad (5.46)$$

e

$$x_k = \left[\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array}\right]. \quad (5.47)$$

A política é dada por

$$\mu(x_k) = Kx_k = (R + B^T PB)^{-1} B^T PAx_k \quad (5.48)$$

O alvo para os algoritmos de regressão é dado pela própria função valor estado estimada anteriormente mais o custo pela transição de estado.

$$\begin{aligned} V(x_k) &= r(x_k, \mu(x_k)) + V(f(x_k, \mu(x_k))) & (5.49)\\ &= x_k^T Q x_k + \mu(x_k)^T R \mu(x_k) + x_{k+1}^T P x_{k+1} & (5.50) \end{aligned}$$

sendo P a matriz de parâmetros da função valor estado estimada na iteração k' da etapa de melhoria da política, ou seja, após a avaliação da política.

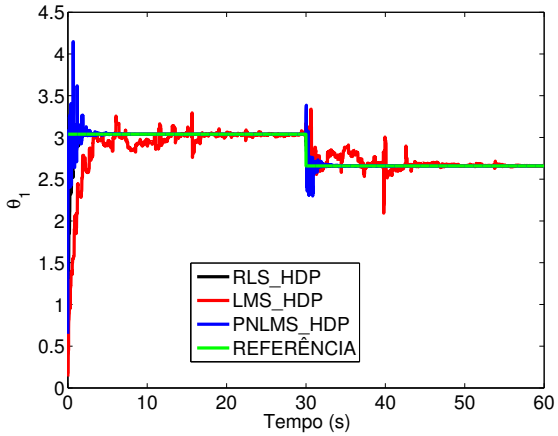
Estimação da Função Valor: Na tabela(5.13) estão indicados os parâmetros dos algoritmos RLS, PNLMS e LMS utilizados para estimar os parâmetros da função valor.

Tabela 5.13: Ajustes dos Parâmetros Livres para RLS, LMS e PNLMS.

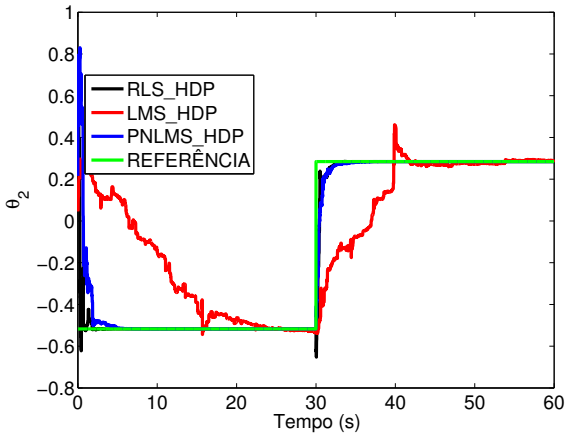
Algoritmos	Parâmetros Livres			
	ρ	μ	λ	P
RLS			0,95	$10I_n$
LMS		0,0003		
PNLMS	0,71	1,6		

Observa-se novamente que os parâmetros da função valor convergiram rapidamente para o valor esperado tanto com o RLS quanto com o PNLMS, enquanto que o LMS teve um desempenho relativamente ruim.

As figuras (5.54d), (5.54e), (5.55a), (5.55b), (5.56a), (5.56b), (5.57a), (5.57b), (5.58a) e (5.58b) mostram as trajetórias dos parâmetros da função-V. Pode-se ver que a convergência dos parâmetros ocorre antes dos 10 segundos de simulação com os algoritmos RLS e PNLMS.



(d) Parâmetro θ_1.



(e) Parâmetro θ_2.

Figura 5.54: Trajetórias seguidas pelos parâmetros θ_1 e θ_2 no processo de aprendizagem.

Figura 5.55: Trajetórias seguidas pelos parâmetros θ_3 e θ_4 no processo de aprendizagem.

(a) Parâmetro θ_3. **(b)** Parâmetro θ_4.

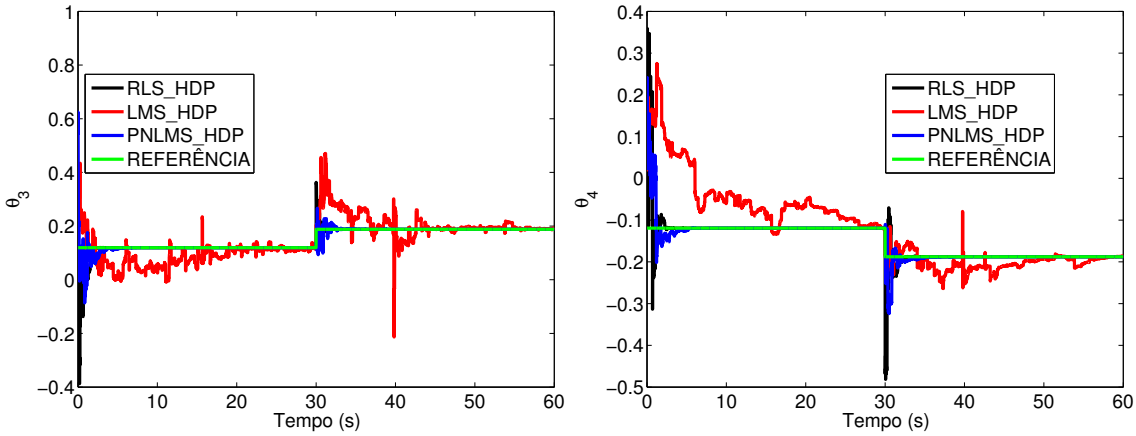


Figura 5.56: Trajetórias seguidas pelos parâmetros θ_5 e θ_6 no processo de aprendizagem.

(a) Parâmetro θ_5. **(b)** Parâmetro θ_6.

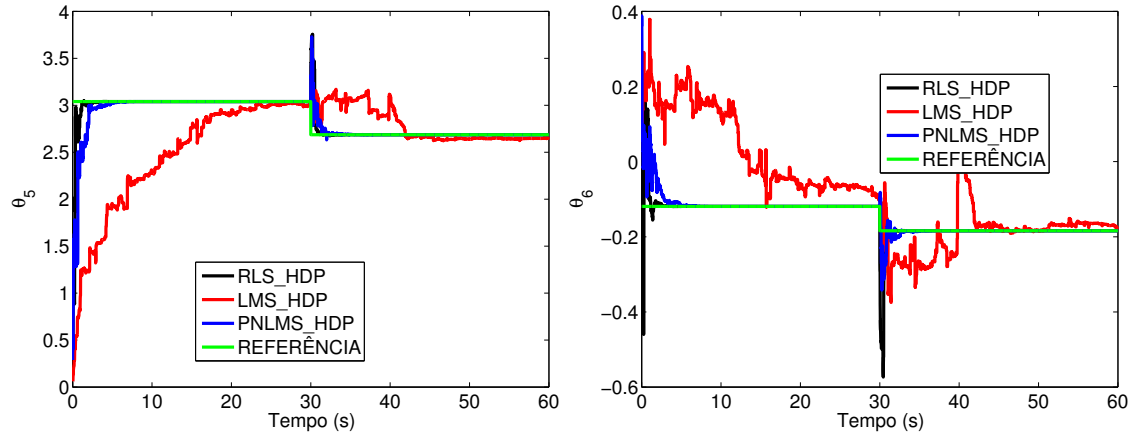


Figura 5.57: Trajetórias seguidas pelos parâmetros θ_7 e θ_8 no processo de aprendizagem.

(a) Parâmetro θ_7. **(b)** Parâmetro θ_8.

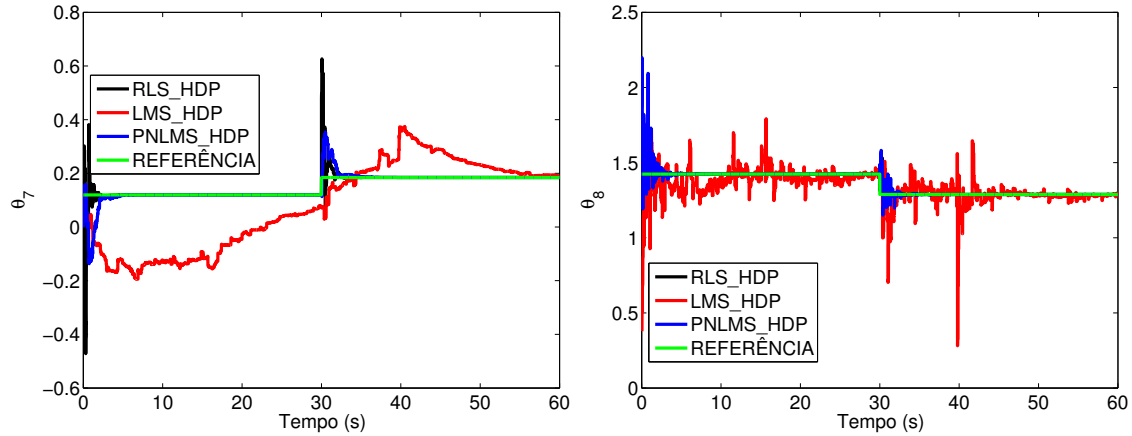
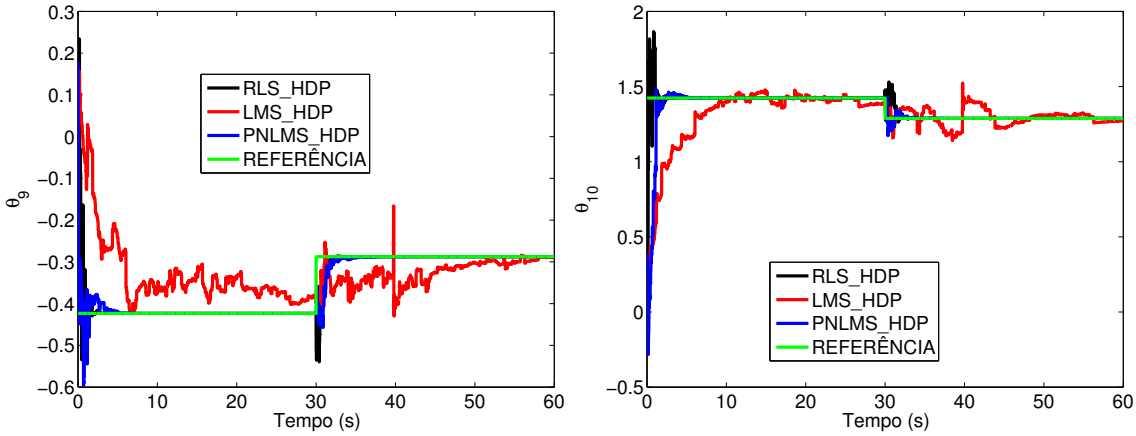


Figura 5.58: Trajetórias seguidas pelos parâmetros θ_9 e θ_{10} no processo de aprendizagem.

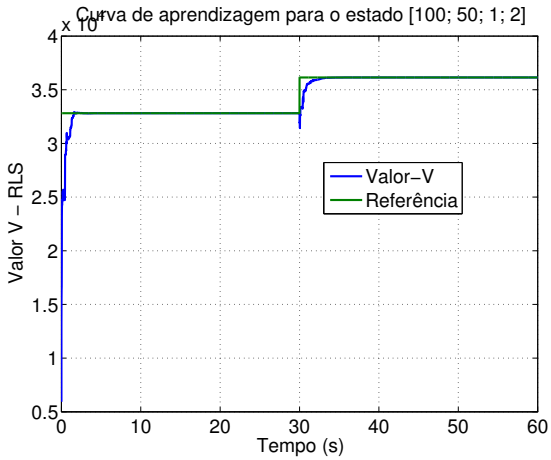
(a) Parâmetro θ_9. **(b)** Parâmetro θ_{10}.



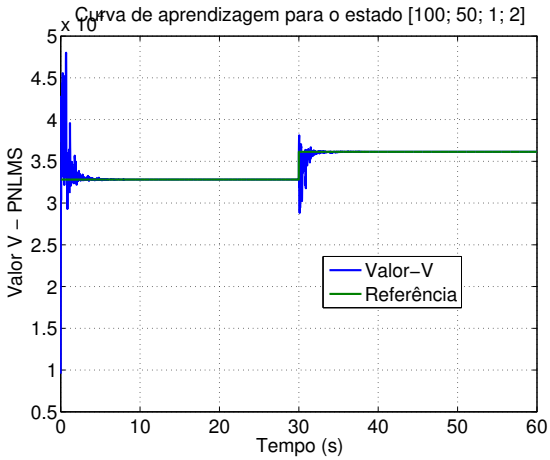
A figura (5.59) mostra as curvas de aprendizagem da função valor para o estado $[100 \quad 50 \quad 1 \quad 2]^T$.

Figura 5.59: Curva de aprendizagem com o RLS (esquerda), o PNLMS (Direita) e o LMS (abaixo) para o estado $[100 \;\; 50 \;\; 1 \;\; 2]^T$.

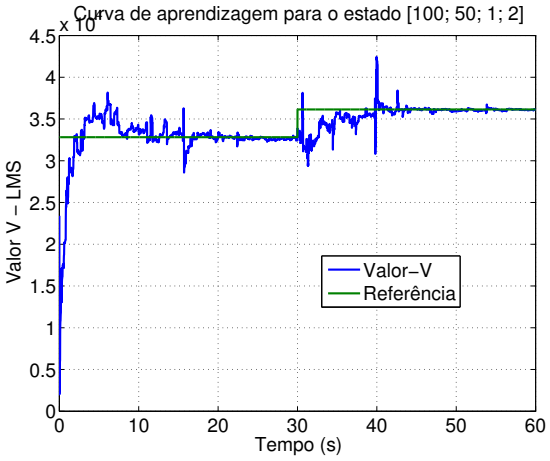
(a) Curva de aprendizagem com o RLS.



(b) Curva de aprendizagem com o PNLMS.



(c) Curva de aprendizagem com o LMS.



Variável de Estado e Esforço de Controle: As tabelas (5.14) e (5.15) mostram os ganhos de retroação de estados ótimos obtidos pelo algoritmo HDP antes e após mudança da planta, respectivamente.

Tabela 5.14: Ganhos da políticas ótima obtidos antes da mudança da planta

Algoritmo	Iteração 3000			
RLS	0,0344	0,0521	0,3256	-0,3256
	0,0521	0,0344	-0,3256	0,3256
PNLMS	0,0344	0,0521	0,3256	-0,3256
	0,0521	0,0344	-0,3256	0,3256
LMS	0,0411	0,0521	0,3208	-0,3207
	0,0436	0,0336	-0,3202	0,3202

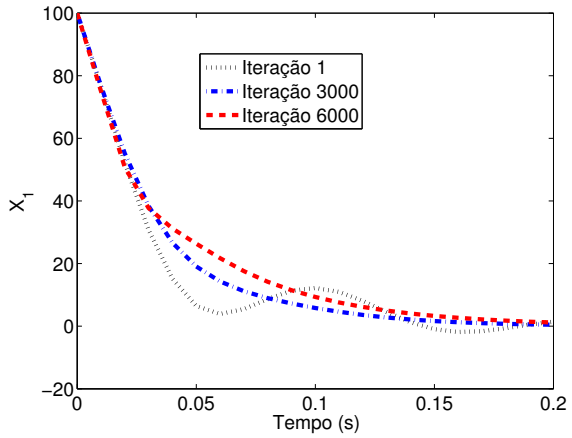
Tabela 5.15: Ganhos da política ótima obtidos após alteração dos parâmetros da planta

Algoritmo	Iteração 6000
RLS	-0,1022 0,1892 0,2469 -0,2469 0,1936 -0,0966 -0,2468 0,2468
PNLMS	-0,1022 0,1892 0,2469 -0,2469 0,1936 -0,0966 -0,2468 0,2468
LMS	-0,1011 0,1907 0,2466 -0,2466 0,1925 -0,0988 -0,2461 0,2461

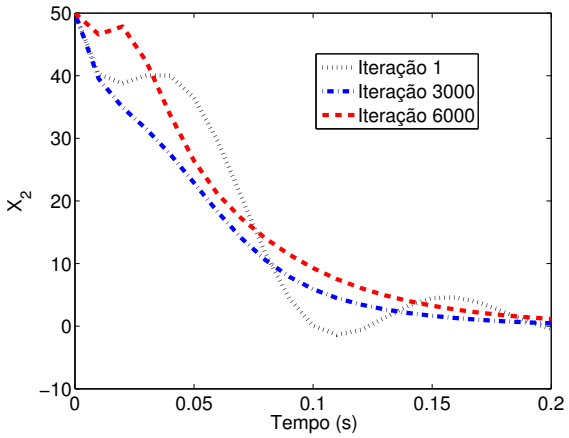
As figuras (5.60) e (5.61) mostram as trajetórias seguidas pelos estados do sistema do estado inicial $\begin{bmatrix} 100 & 50 & 1 & 2 \end{bmatrix}$ ao estado $\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$ quando se utiliza as políticas ótimas encontradas pelo algoritmo HDP com o LMS e o PNLMS, respectivamente. A trajetória para a política inicial adotada é também mostrada para comparação.

Figura 5.60: Trajetórias dos estados do sistema para a política inicial adotada e para as políticas ótimas encontradas com o LMS, antes e após mudança da planta (estado inicial $[100 \;\; 50 \;\; 1 \;\; 2]^T$).

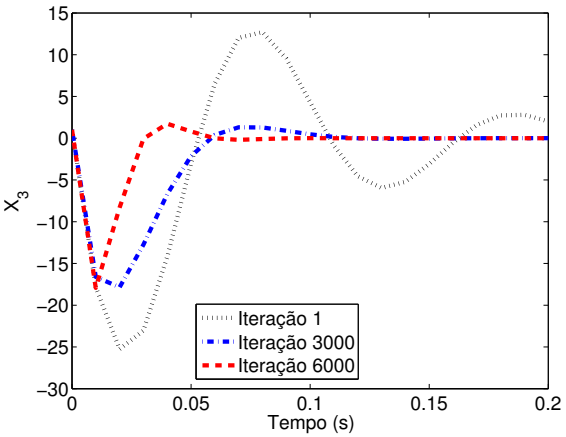
(a) Trajetória de x_1.



(b) Trajetória de x_2.



(c) Trajetória de x_3.



(d) Trajetória de x_4.

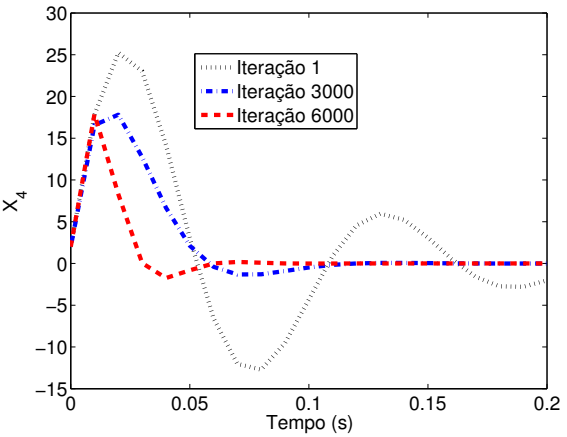
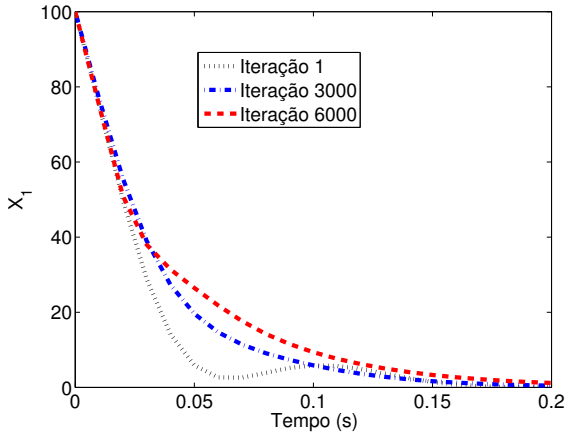
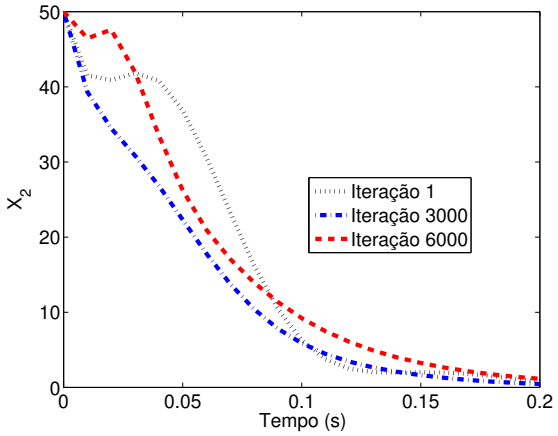


Figura 5.61: Trajetórias dos estados do sistema para a política inicial adotada e para as políticas ótimas encontradas com o PNLMS, antes e após mudança da planta (estado inicial $[100 \quad 50 \quad 1 \quad 2]^T$).

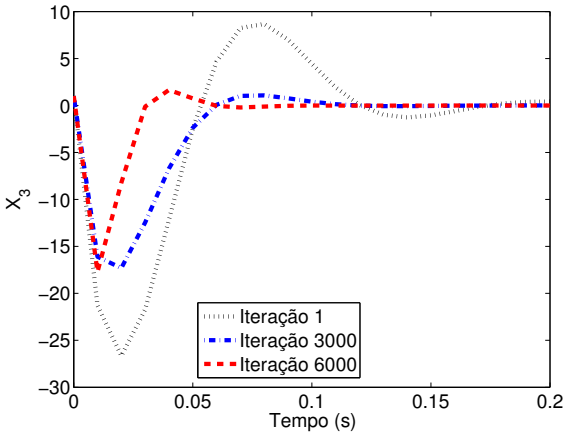
(a) Trajetória de x_1.



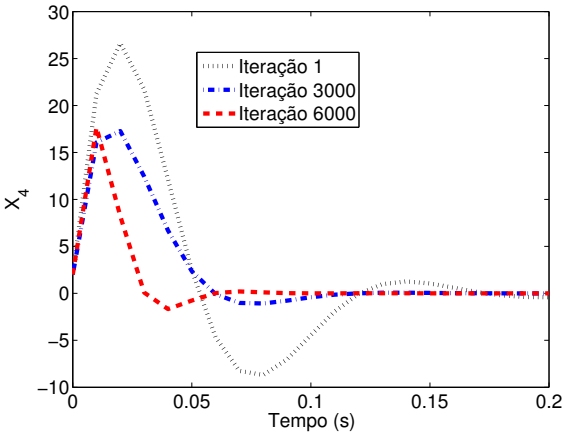
(b) Trajetória de x_2.



(c) Trajetória de x_3.



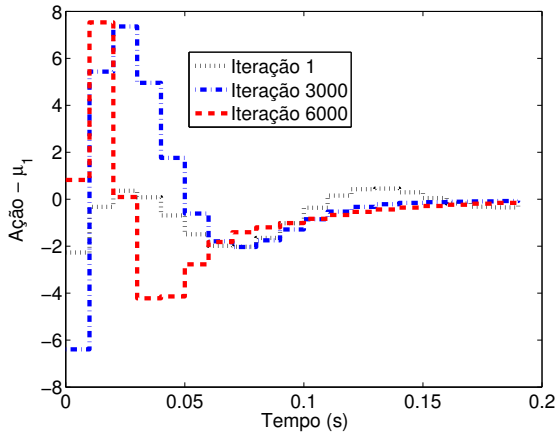
(d) Trajetória de x_4.



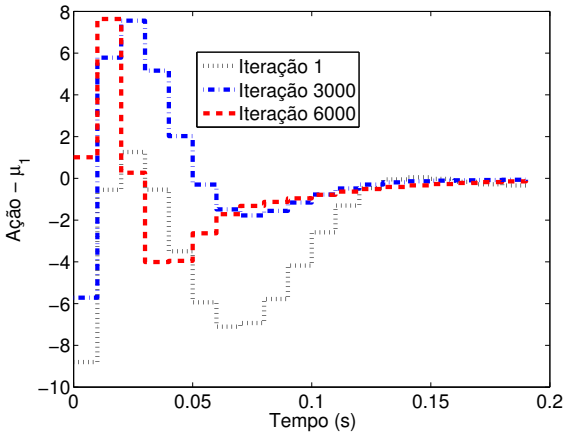
A figura (5.62) mostra as políticas ou ações de controle encontras nas iterações 3000 e 6000 que são consideradas ótimas e a politica inicialmente adotada.

Figura 5.62: Política encontradas com o LMS (esquerda) e o PNLMS (Direita).

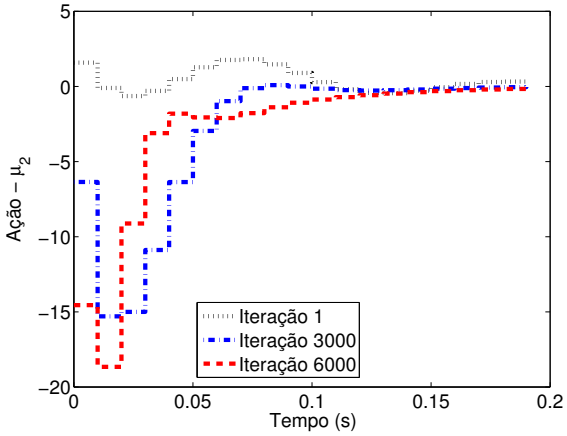
(a) Ação μ_1 encontradas com o LMS.



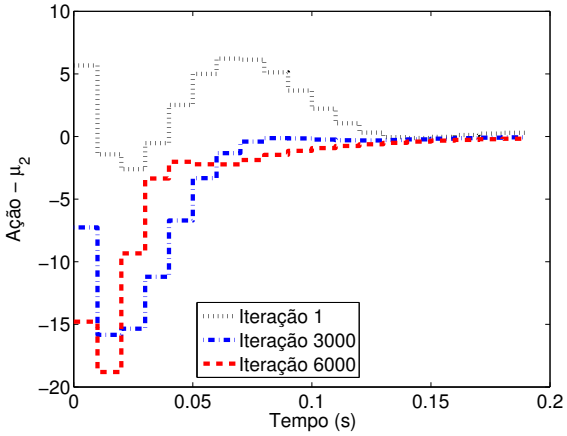
(b) Ação μ_1 encontradas com o PNLMS.



(c) Ação μ_2 encontradas com o LMS.



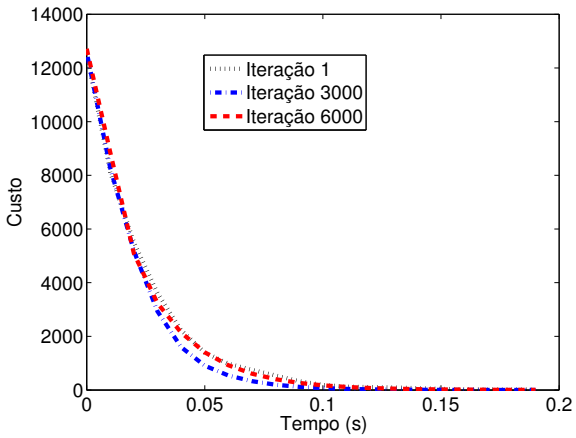
(d) Ação μ_2 encontradas com o PNLMS.



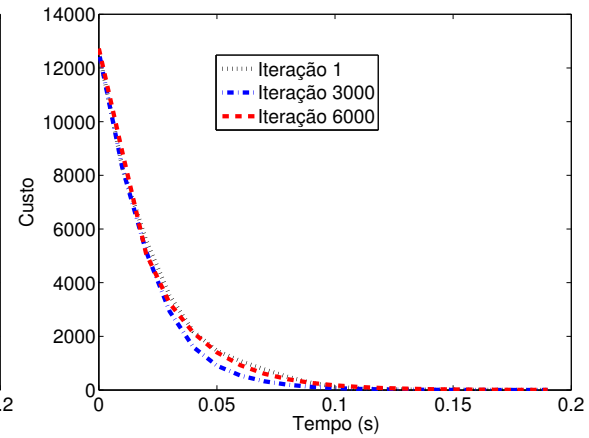
A figura (5.63) mostra os gráficos das funções de custo ótimas para o estado inicial $[100 \;\; 50 \;\; 1 \;\; 2]^T$ antes e após mudança da planta e também o gráfico do custo para as ações iniciais adotadas.

Figura 5.63: Custo encontradas com o LMS (esquerda) e o PNLMS (Direita).

(a) Custos encontrados com o LMS.



(b) Custos encontrados com o PNLMS.



5.4.1.2 Conclusão

A mesma estratégia adotada para o HDP do experimento com a aeronave foi assumida. Devido o maior número de parâmetros a estimar, foi necessário mais iterações para encontrar a política ótima neste experimento que com o da aeronave correspondente.

Mais uma vez o algoritmo LMS teve um comportamento relativamente ruim na estimação dos parâmetros da função-V comparado aos algoritmos RLS e PNLMS. Estes últimos tiveram um desempenho bastante semelhante na tarefa de estimação paramétrica, ambos precisaram de cerca de 1000 iterações para convergir, antes da mudança de parâmetros da planta, enquanto o LMS não convergiu plenamente. Esses mesmos comportamentos ocorreram após a mudança dos parâmetros da planta.

5.4.2 ADHDP

Para validação computacional do ADHDP também foi utilizado o modelo do circuito elétrico e a função de custo dados por (5.7) e (5.13), respectivamente. Diferentemente do caso HDP, na abordagem ADHDP foi constatada sua convergência pelo ganhos de retroação de estado obtidos a cada iteração do algoritmo e tendo como referência os ganhos obtidos através da resolução da equação de Riccati pelo método de Schur.

5.4.2.1 Experimento

Diferentemente do experimento utilizando o ADHDP aplicado ao modelo do avião, considera-se aqui um modelo que sofre uma mudança nos seu parâmetros. Inicialmente, o sistema parte de uma política escolhida aleatoriamente, mas dentro do espaço de políticas com estrutura dada pela equação (5.52), a qual é atualizada a cada iteração do ADHDP. O processo de aprendizagem para quando as cinquenta últimas atualizações da política produz uma variação nos seus parâmetros inferior a 10^{-5}, ou seja, quando a norma do máximo em relação ao vetor de parâmetros da política for inferior a 10^{-5}. Além disso, toda vez que há mudança no comportamento da planta um novo ciclo de aprendizagem é executado.

Parâmetros de Configuração das Simulações: Neste experimento computacional definiu-se os seguintes parâmetros para o algoritmo de aprendizagem ADHDP e para o ambiente:

- O fator de desconto da função valor foi definido igual a um;
- O retorno para cada estado inicial é estimado para cada seis passos de simulação da trajetória;
- Após seis iterações, a trajetória dos estados é reiniciada em um estado selecionado aleatoriamente no espaço de estado;

- Um ruido branco é adicionado na ação de controle;
- A política é melhorada a cada transição de estado;
- A política inicial tem dado por

$$\mu_1 = \begin{bmatrix} -0,1 & -0,1 & -0,1 & -0,1 \\ -0,1 & -0,1 & -0,1 & -0,1 \end{bmatrix} x_1;$$

- O tempo de amostragem de $0,01$ s.
- Função de custo definida pela equação (5.13);
- São realizadas 10000 iterações equivalente a 100 segundos de simulação;
- Na iteração 6000 a planta sofre uma variação brusca em seus parâmetros;

Forma Regressiva da HJB: A parametrização da função de custo e da política para ADHDP são dadas por

$$Q_k(x_k, \mu_k) = z_k^T H z_k = \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}^T \begin{bmatrix} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{bmatrix} \begin{bmatrix} x_k \\ \mu_k \end{bmatrix} = \phi^T \theta \tag{5.51}$$

e

$$\mu_k = -(H_{\mu\mu})^{-1} H_{\mu x} x_k, \tag{5.52}$$

respectivamente, sendo ϕ a matriz de regressores e θ o vetor de parâmetros. Para o caso específico do ambiente escolhido, H é uma matriz simétrica de ordem 6×6, H_{xx} é matriz simétrica de ordem 4×4, $H_{x\mu}$ é matriz simétrica de ordem 4×2, $H_{\mu x}$ é matriz simétrica de ordem 2×4, $H_{\mu\mu}$ é matriz simétrica de ordem 2×2 e o vetor $z_k = \begin{bmatrix} x_k \\ \mu_k \end{bmatrix}$ é a concatenação dos vetores de estado e ação.

Então, representado a matriz H por

$$H = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 \\ \theta_2 & \theta_7 & \theta_8 & \theta_9 & \theta_{10} & \theta_{11} \\ \theta_3 & \theta_8 & \theta_{12} & \theta_{13} & \theta_{14} & \theta_{15} \\ \theta_4 & \theta_9 & \theta_{13} & \theta_{16} & \theta_{17} & \theta_{18} \\ \theta_5 & \theta_{10} & \theta_{14} & \theta_{17} & \theta_{19} & \theta_{20} \\ \theta_6 & \theta_{11} & \theta_{15} & \theta_{18} & \theta_{20} & \theta_{21} \end{bmatrix}, \tag{5.53}$$

sua vetorização fica

$$\begin{aligned} vec(H) = \theta = \Big[& \theta_1 \quad 2\theta_2 \quad 2\theta_3 \quad 2\theta_4 \quad 2\theta_5 \quad 2\theta_6 \quad \theta_7 \quad 2\theta_8 \quad 2\theta_9 \quad 2\theta_{10} \quad 2\theta_{11} \\ & \theta_{12} \quad 2\theta_{13} \quad 2\theta_{14} \quad 2\theta_{15} \quad \theta_{16} \quad 2\theta_{17} \quad 2\theta_{18} \quad \theta_{19} \quad 2\theta_{20} \quad \theta_{21} \Big]^T \end{aligned} \tag{5.54}$$

e o regressor e a ação tomam as formas

$$\begin{aligned}\phi &= \left[\begin{array}{cccccccccc} x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 & x_1\mu_1 & x_1\mu_2 & x_2^2 & x_2x_3 & x_2x_4 & x_2\mu_1 & x_2\mu_2 \\ x_3^2 & x_3x_4 & x_3\mu_1 & x_3\mu_2 & x_4^2 & x_4\mu_1 & x_4\mu_2 & \mu_1^2 & \mu_1\mu_2 & \mu_2^2 \end{array}\right] \end{aligned} \tag{5.55}$$

e

$$\mu_k = -\left[\begin{array}{cc} \theta_{19} & \theta_{20} \\ \theta_{20} & \theta_{21} \end{array}\right]^{-1} \left[\begin{array}{cccc} \theta_5 & \theta_{10} & \theta_{14} & \theta_{17} \\ \theta_6 & \theta_{11} & \theta_{15} & \theta_{18} \end{array}\right] x_k = -\left[\begin{array}{cccc} k_1 & k_2 & k_3 & k_4 \\ k_5 & k_6 & k_7 & k_8 \end{array}\right] x_k, \tag{5.56}$$

respectivamente.

O alvo para ADHDP é dado pela própria função-Q corrente mais o custo pela transição de estado.

$$\begin{aligned} Q(x_k, \mu_k) &= r(x_k, \mu_k) + Q\left(f(x_k, \mu_k), \mu_k(f(x_k, \mu_k))\right) && (5.57) \\ &= x_k^T Q x_k + \mu_k^T R \mu_k + \left[\begin{array}{cc} x_{k+1}^T & \mu_{k+1}^T \end{array}\right] H \left[\begin{array}{c} x_{k+1} \\ \mu_{k+1} \end{array}\right], && (5.58) \end{aligned}$$

sendo H a estimativa dos parâmetros da função-Q na iteração k' da melhoria da política.

Estimação da Função Valor: Neste caso a função-Q tem como parâmetros os elementos da matriz simétrica H de ordem 6×6, portanto deve ser estimados 21 parâmetros ao invés de 10 como no caso do HDP. Os algoritmos NLMS, PNLMS e IAF-PNLMS foram utilizados para estimar os parâmetros da função-Q com ajustes indicados na tabela (5.16).

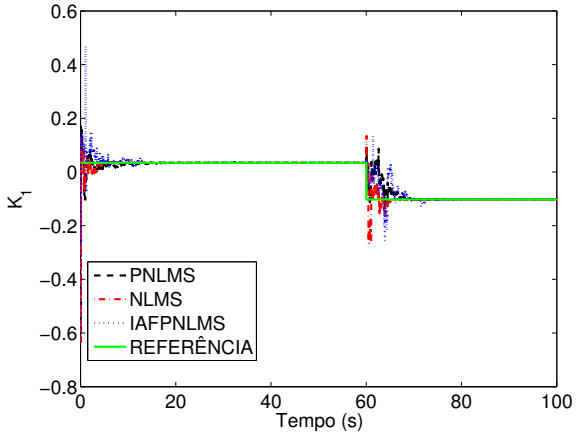
Tabela 5.16: Ajuste dos Parâmetros Livres para NLMS, IAF-PNLMS, PNLMS

Algoritmos	Parâmetros Livres	
	ρ	μ
NLMS		1,6
PNLMS	0,575	1,663
IAF-PNLMS		1,2

As figuras abaixo mostram as trajetórias seguidas pelos ganhos de retração de estado e os ganhos ótimos (em verde) correspondentes.

Figura 5.64: Trajetória seguida pelo Parâmetro k_1 e k_2 no processo de aprendizagem.

(a) Valor k_1 da matriz de ganho de retroação de estado.



(b) Valor k_2 da matriz de ganho de retroação de estado.

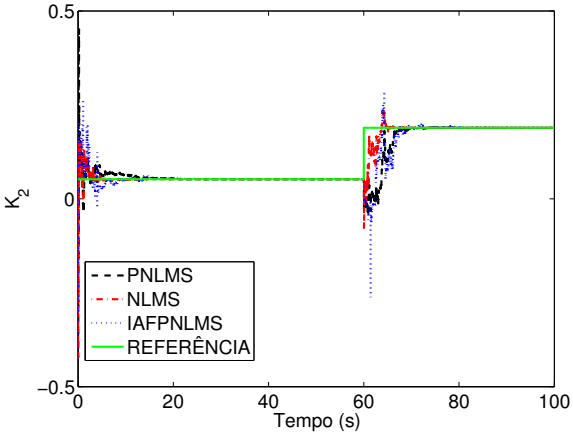
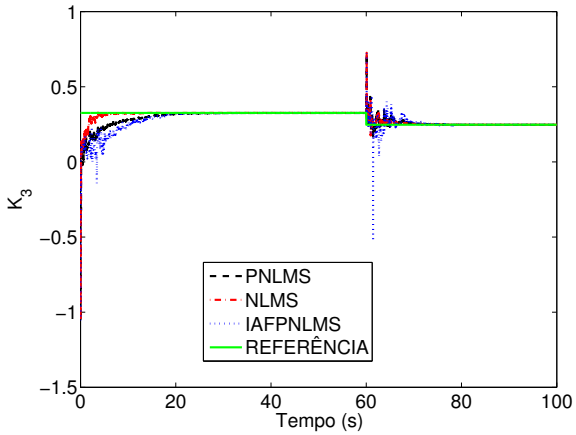


Figura 5.65: Trajetória seguida pelo Parâmetro k_3 e k_4 no processo de aprendizagem.

(a) Valor k_3 da matriz de ganho de retroação de estado.



(b) Valor k_4 da matriz de ganho de retroação de estado.

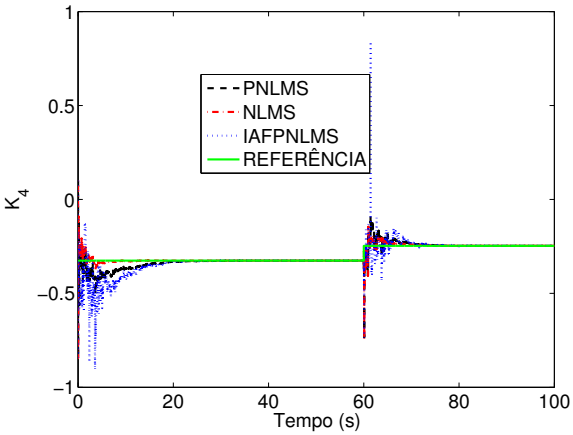
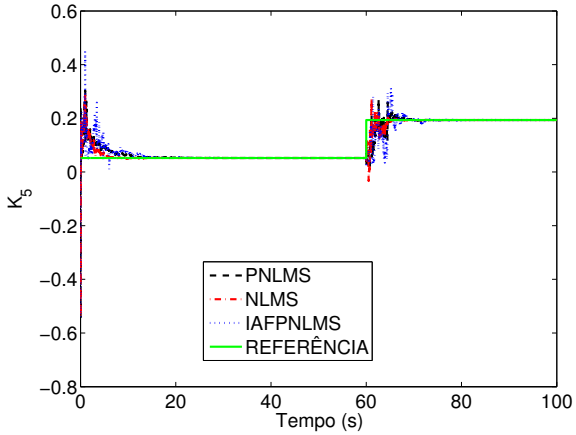


Figura 5.66: Trajetória seguida pelo Parâmetro k_5 e k_6 no processo de aprendizagem.

(a) Valor k_5 da matriz de ganho de retroação de estado.



(b) Valor k_6 da matriz de ganho de retroação de estado.

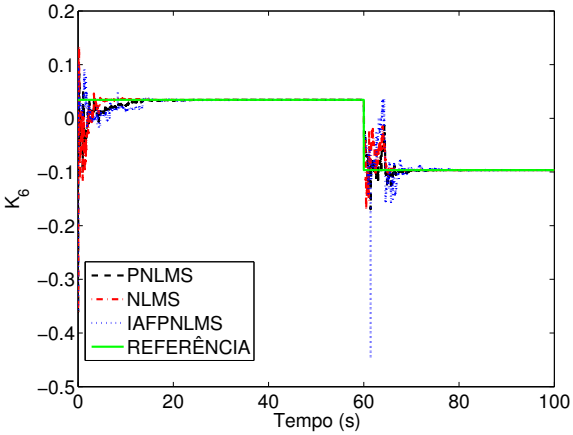
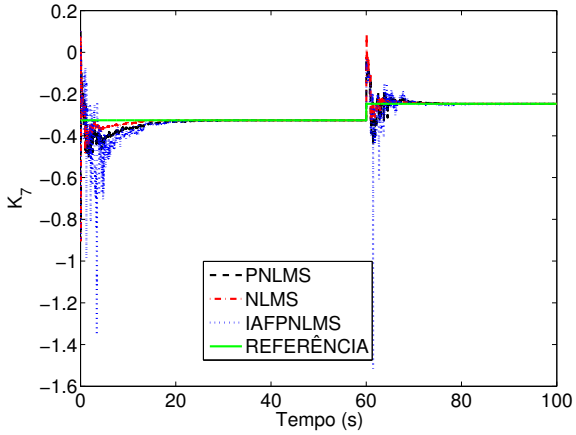
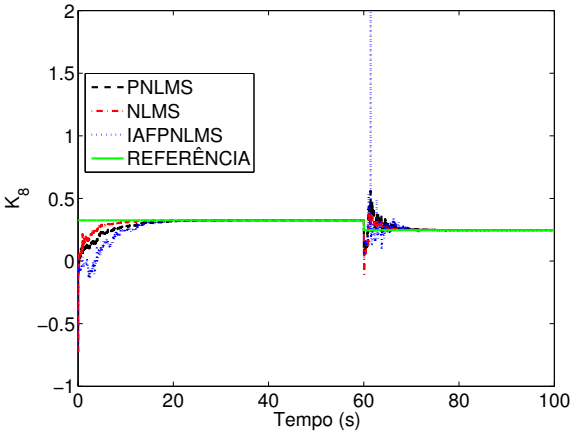


Figura 5.67: Trajetória seguida pelo Parâmetro k_7 e k_8 no processo de aprendizagem.

(a) Valor k_7 da matriz de ganho de retroação de estado.

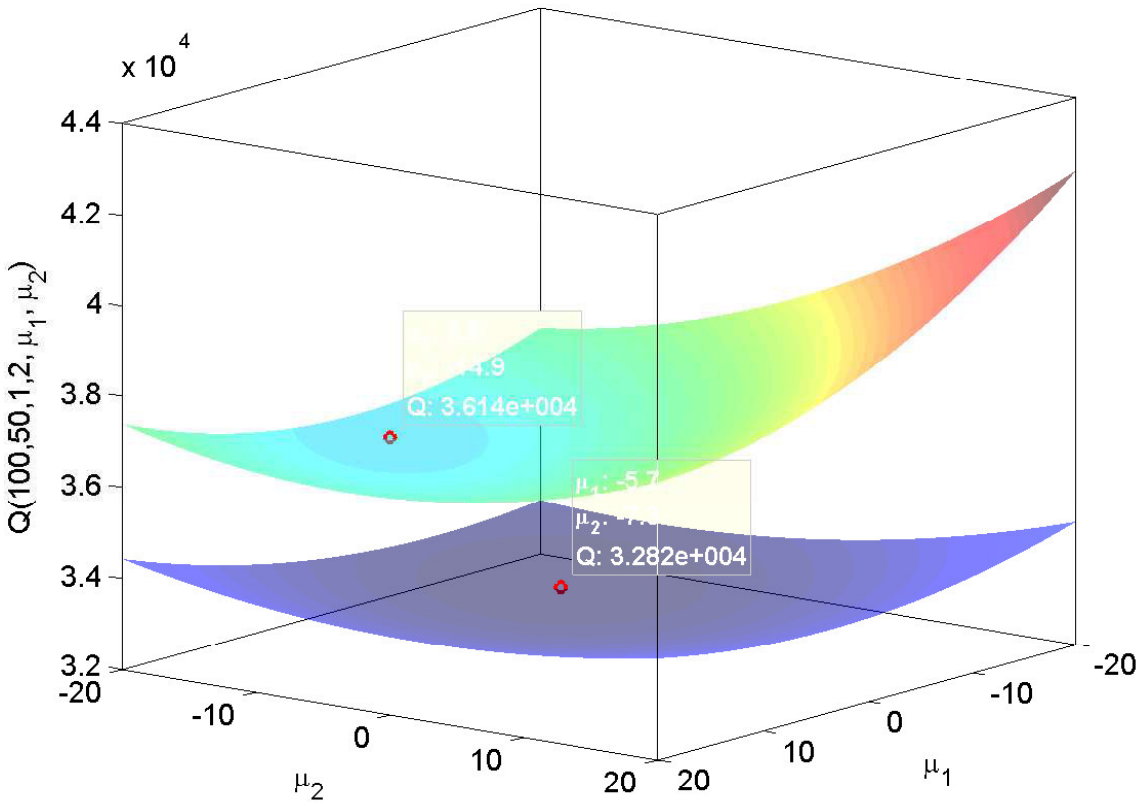


(b) Valor k_8 da matriz de ganho de retroação de estado.



A figura (5.68) mostra a superfície gerada para o estado $\begin{bmatrix} 100 & 50 & 1 & 2 \end{bmatrix}^T$ antes e após mudança nos parâmetros da planta. Observa-se que o valor escolhido para ação de controle no estado $\begin{bmatrix} 100 & 50 & 1 & 2 \end{bmatrix}^T$ é o minimizador da função que gerou a superfície e também é mostrado na figura.

Figura 5.68: Superfícies da função-Q no estado $[100\ 50\ 1\ 2]^T$ com indicação do valor mínimo antes (gráfico inferior) e após (gráfico superior) mudança da planta.



Variável de Estado e Esforço de Controle: Os ganhos de retroação de estado ótimos antes e após mudança na planta (iterações 6000 e 10000) são mostrados nas tabelas (5.17) e (5.18.

Tabela 5.17: Ganhos da políticas ótima obtidos antes da mudança da planta

Algoritmo	Iteração 6000			
PNLMS	0,0344	0,0521	0,3254	-0,3258
	0,0521	0,0344	-0,3258	0,3254
NLMS	0,0344	0,0521	0,3255	-0,3257
	0,0521	0,0343	-0,3261	0,3251
IAF-PNLMS	0,0343	0,0522	0,3256	-0,3256
	0,0521	0,0344	-0,3256	0,3256

Tabela 5.18: Ganhos da política ótima obtidos após alteração dos parâmetros da planta

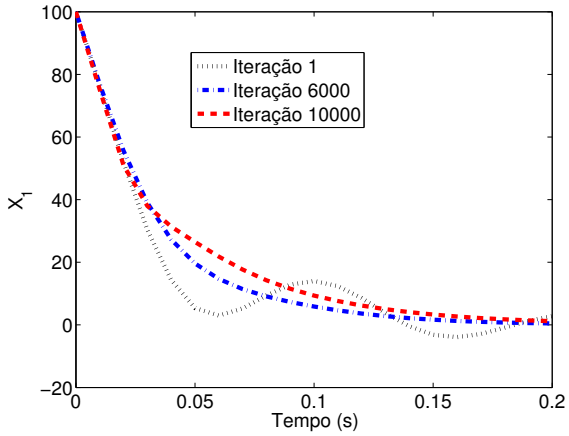
Algoritmo	Iteração 10000
PNLMS	-0,1022 0,1892 0,2470 -0,2469 0,1936 -0,0966 -0,2468 0,2467
NLMS	-0,1023 0,1892 0,2472 -0,2467 0,1936 -0,0966 -0,2467 0,2468
IAF-PNLMS	-0,1023 0,1892 0,2470 -0,2468 0,1936 -0,0965 -0,2467 0,2468

As trajetórias ótimas e inicial representativas dos estados, política e custo para o estado inicial $\begin{bmatrix} 100 & 50 & 1 & 2 \end{bmatrix}^T$ são mostradas abaixo.

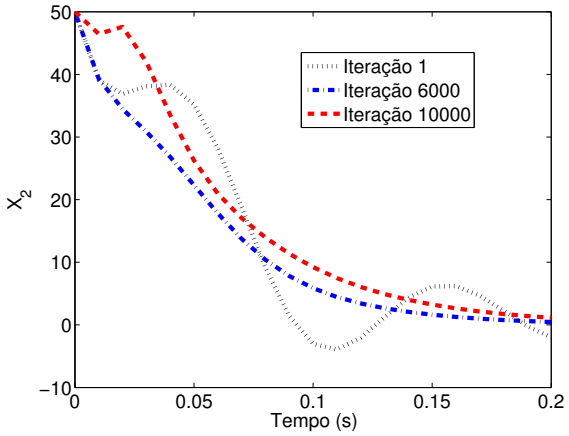
Observa-se nas figuras (5.69) e (5.70) as trajetórias seguidas pelos estados do sistema quando são tomadas a ação inicial e as ações ótimas encontradas com os algoritmos IAF-PNLMS e PNLMS, respectivamente.

Figura 5.69: Trajetórias dos estados do sistema utilizando o IAF-PNLMS.

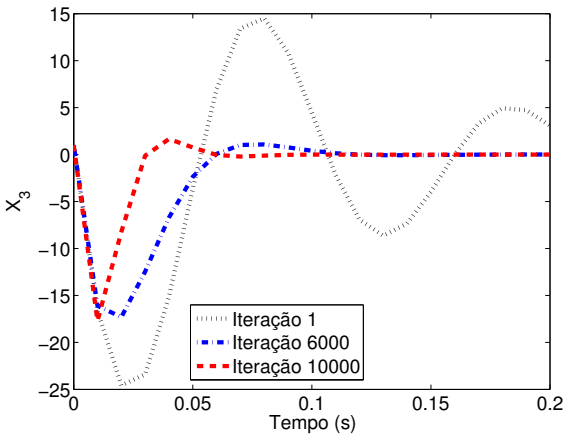
(a) Trajetória de x_1.



(b) Trajetória de x_2.



(c) Trajetória de x_3.



(d) Trajetória de x_4.

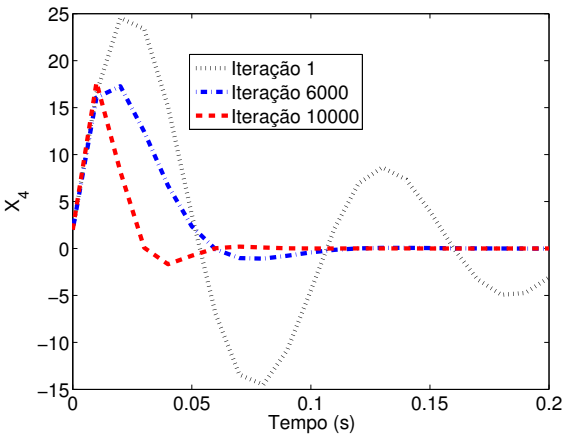
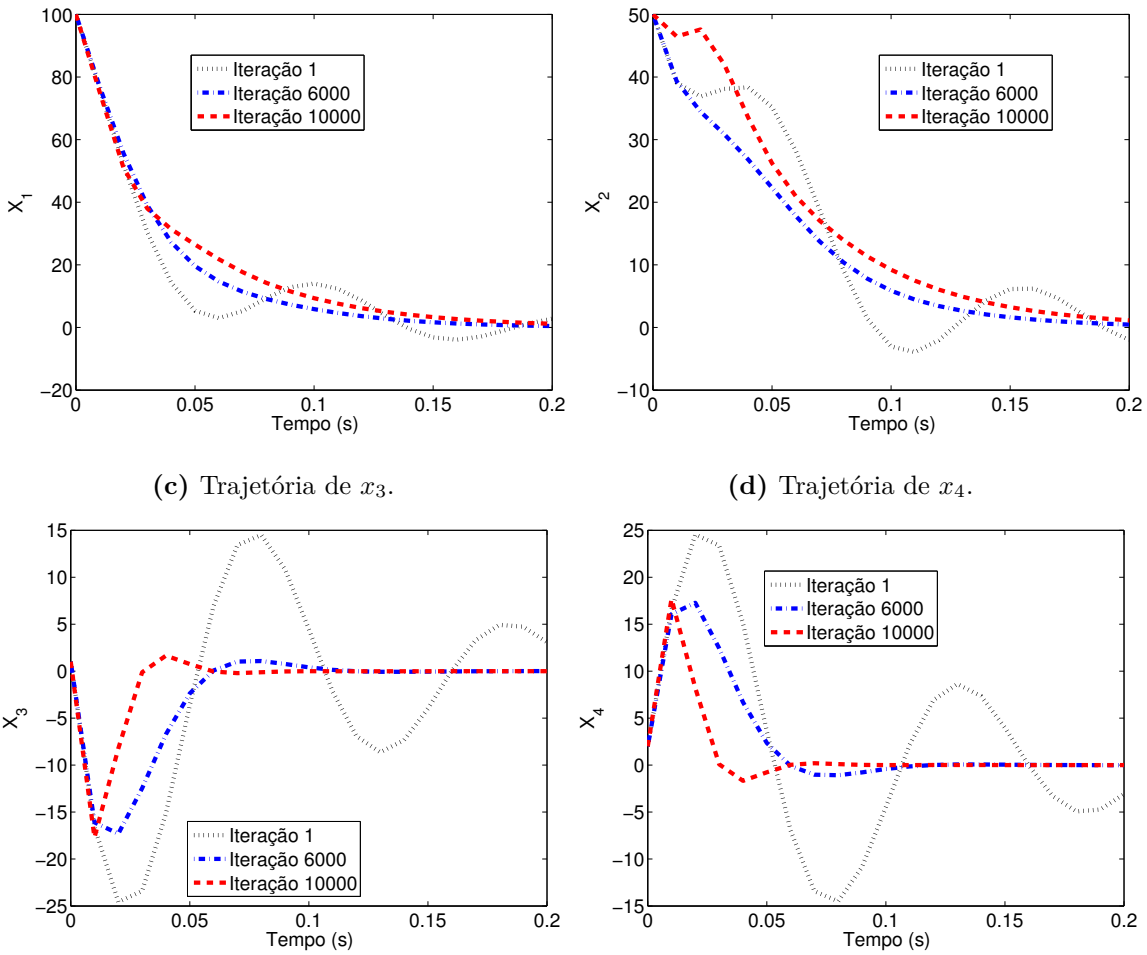


Figura 5.70: Trajetórias dos estados do sistema utilizando 0 PNLMS

(a) Trajetória de x_1.

(b) Trajetória de x_2.

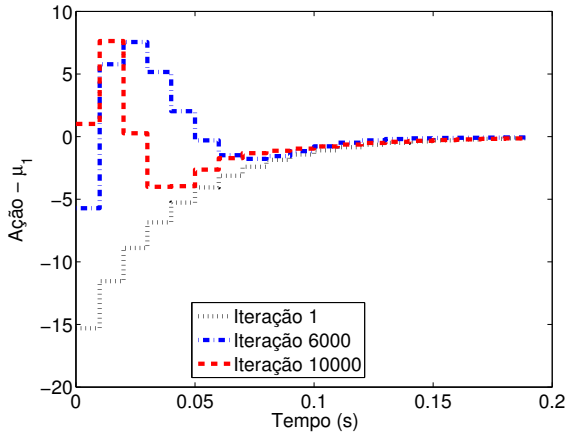


(c) Trajetória de x_3.

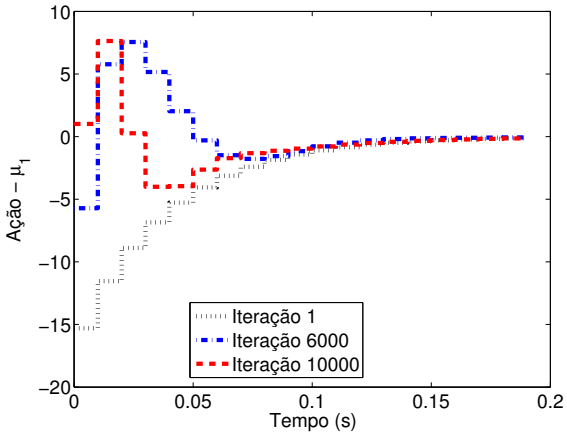
(d) Trajetória de x_4.

Figura 5.71: Política encontradas com o IAF-PNLMS (esquerda) e o PNLMS (Direita).

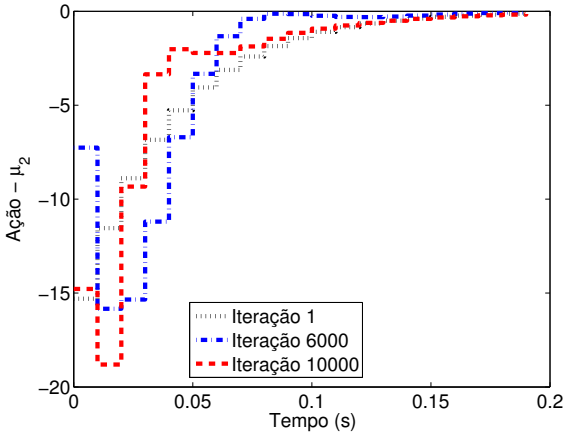
(a) Ação de controle μ_1 com o IAF-PNLMS.



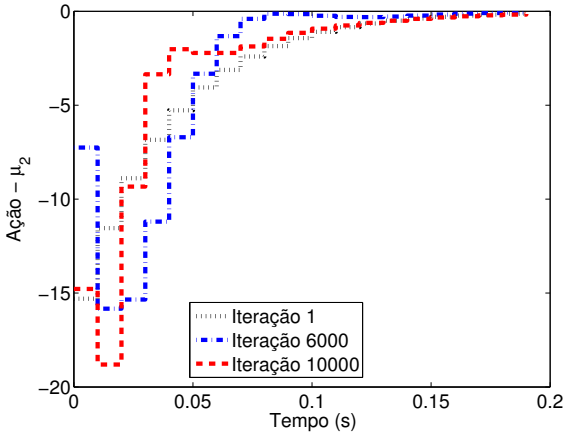
(b) Ação de controle μ_2 com o PNLMS.



(c) Ação de controle μ_1 com o IAF-PNLMS.



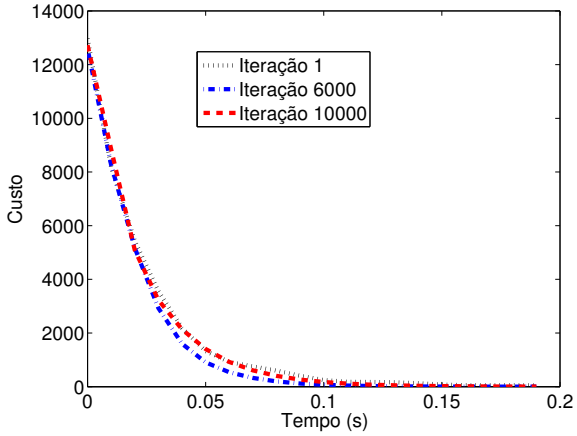
(d) Ação de controle μ_2 com o PNLMS.



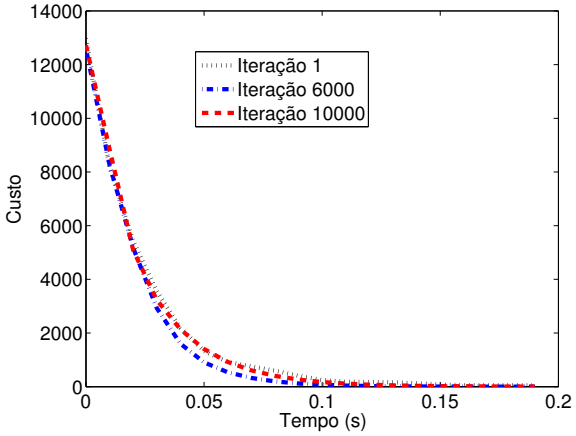
Observa-se na figura (5.72) os gráficos das funções de custo para as políticas indicadas na figura (5.71).

Figura 5.72: Custo encontrado com o IAF-PNLMS (esquerda) e o PNLMS (Direita).

(a) Custos encontrados com o IAF-PNLMS.



(b) Custos encontrados com o PNLMS.



5.4.2.2 Conclusão

Este experimento mostrou a dificuldade em encontrar uma política ótima para o problema do DLQR quando se utiliza o ADHDP. O que era de se esperar devido a falta de conhecimento prévio do ambiente. Foi necessário uma grande exploração dos espaços de estado e de ações para conseguir convergência dos parâmetros da função valor. Devido o maior número de parâmetros para estimar, foi necessário mais iterações neste experimento que o correspondente para a aeronave para se obter a política ótima.

Verificou-se, ainda, que a estratégia de incluir ruido branco na ação de controle combinada com revitalização de estado como forma de exploração levou a uma convergência bem mais rápida que utilizando só revitalização de estado ou só ruido.

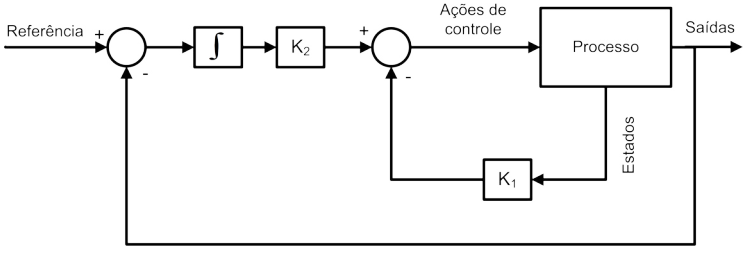
O algoritmo IAF-PNLMS teve um comportamento pior na estimação dos parâmetros da função valor comparado aos algoritmos NLMS e PNLMS. O NLMS e o PNLMS tiveram um desempenho bastante semelhante na tarefa de estimação paramétrica, ambos precisaram de cerca de 2000 iterações para obter convergência dos parâmetros da função valor.

5.4.3 DHP

Nesta simulação do algoritmo DHP, a estrutura do sistema de controle foi modificada para prevenir erro em estado estacionário a resposta em degrau. Este tipo de erro ocorre em sistema do tipo zero, ou seja, processo sem nenhum integrador puro na sua função de transferência.

A nova estrutura é mostrada na figura (5.73). Nota-se que foram introduzidos integradores entre o comparador de erro e o processo a controlar aumentando a ordem do sistema de quatro estados para seis. Os dois novos estados são as saídas dos integradores e são vinculados aos erros entre as saídas do processo e às referências.

Figura 5.73: Estrutura do Controlador.



5.4.3.1 Experimento

Parâmetros de Configuração das Simulações: Os seguintes parâmetros para o algoritmo de aprendizagem DHP e para o ambiente são adotados:

- O fator de desconto da função valor foi definido igual a um;

- A política é melhorada a cada transição de estado
- O tempo de amostragem de $0,01$ s.
- Função de custo definida pela equação (5.13) com
$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$ e $R = 0,1I_2$;
- Foram realizadas 6000 iterações equivalente a 60 segundos de simulação;
- A matriz de ganho é formada pela concatenação das matrizes K_1 e K_2 vista na figura (5.73).

Forma Regressiva da HJB: A parametrização do gradiente da função valor é dada por

$$\frac{\partial V(x)}{\partial x} = \frac{\partial (x^T \otimes x^T)}{\partial x} vet(P) = \phi^T \theta \tag{5.59}$$

$$\tag{5.60}$$

sendo

$$\phi^T = 2 \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_1 & 0 & 0 & 0 & 0 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 0 & 0 & x_1 & 0 & 0 & 0 & 0 & x_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & x_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & x_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & x_2 \end{bmatrix} \tag{5.61}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_3 & x_4 & x_5 & x_6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_3 & 0 & 0 & x_4 & x_5 & x_6 & 0 & 0 & 0 \\ 0 & 0 & x_3 & 0 & 0 & x_4 & 0 & x_5 & x_6 & 0 \\ 0 & 0 & 0 & x_3 & 0 & 0 & x_4 & 0 & x_5 & x_6 \end{bmatrix}, \tag{5.62}$$

$$vec(P) = \theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 & \theta_7 & \theta_8 & \theta_9 & \theta_{10} & \theta_{11} & \theta_{12} \\ \theta_{13} & \theta_{14} & \theta_{15} & \theta_{16} & \theta_{17} & \theta_{18} & \theta_{19} & \theta_{20} & \theta_{21} \end{bmatrix}^T \tag{5.63}$$

e

$$P = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 \\ \theta_2 & \theta_7 & \theta_8 & \theta_9 & \theta_{10} & \theta_{11} \\ \theta_3 & \theta_8 & \theta_{12} & \theta_{13} & \theta_{14} & \theta_{15} \\ \theta_4 & \theta_9 & \theta_{13} & \theta_{16} & \theta_{17} & \theta_{18} \\ \theta_5 & \theta_{10} & \theta_{14} & \theta_{17} & \theta_{19} & \theta_{20} \\ \theta_6 & \theta_{11} & \theta_{15} & \theta_{18} & \theta_{20} & \theta_{21} \end{bmatrix}, \tag{5.64}$$

A parametrização da política fica

$$\mu(x) = Kx = (R + B^T P B)^{-1} B^T P A x. \tag{5.65}$$

O alvo utilizado pelos algoritmos RLS, NLMS e PNLMS para estimar os parâmetros da função valor é dado por

$$\frac{\partial V(x)}{\partial x} = \frac{\partial r}{\partial x} + \frac{\partial r}{\partial \mu}\frac{\partial \mu}{\partial x} + \frac{\partial V}{\partial f}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial \mu}\frac{\partial \mu}{\partial x}\right) \tag{5.66}$$

$$= 2[x_k^T Q + \mu_k^T R K + x_{k+1}^T P(A + BK)], \tag{5.67}$$

sendo K e P o ganho de retroação e matriz de parâmetros da função valor estimados após a ultima avaliação da política que no nosso caso é na própria iteração k (avaliação completamente otimista).

Estimação da Função Valor: Os algoritmos utilizados para estimar os parâmetros da função valor foram ajustados conforme tabela (5.19).

Tabela 5.19: Ajuste dos parâmetros livres para RLS, PNLMS e NLMS

Algoritmos	Parâmetros Livres			
	ρ	μ	λ	P
RLS			0,97	I_n
PNLMS	0,04	1,0		
NLMS		0,8		

As figuras de (5.74) a (5.84) mostram as trajetórias dos parâmetros da função-Q. Pode-se ver que a convergência ocorre antes dos 60 segundos para todos os algoritmos de estimação da função-Q.

Figura 5.74: Trajetória seguida pelo Parâmetro θ_1 (esquerda) e θ_2 (direita) no processo de aprendizagem.

(a) Parâmetro θ_1. **(b)** Parâmetro θ_2.

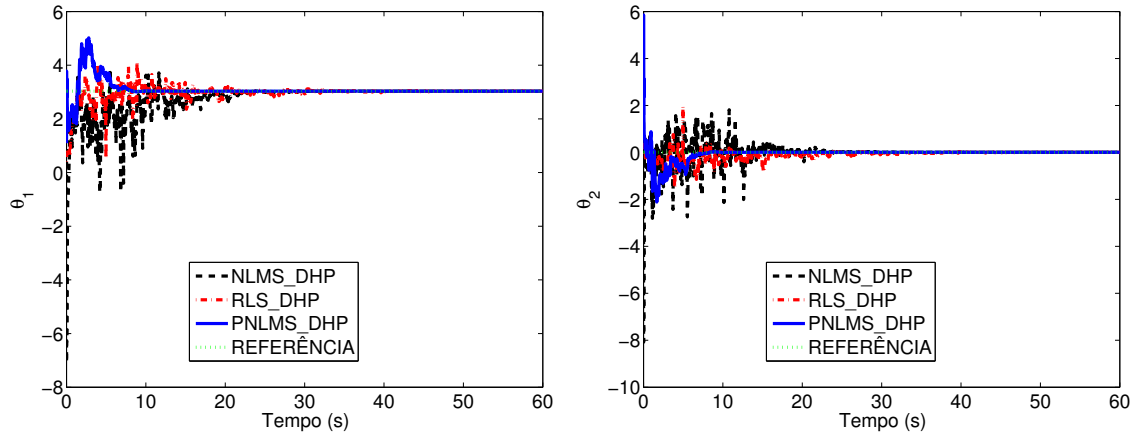


Figura 5.75: Trajetória seguida pelo Parâmetro θ_3 (esquerda) e θ_4 (direita) no processo de aprendizagem.

(a) Parâmetro θ_3. **(b)** Parâmetro θ_4.

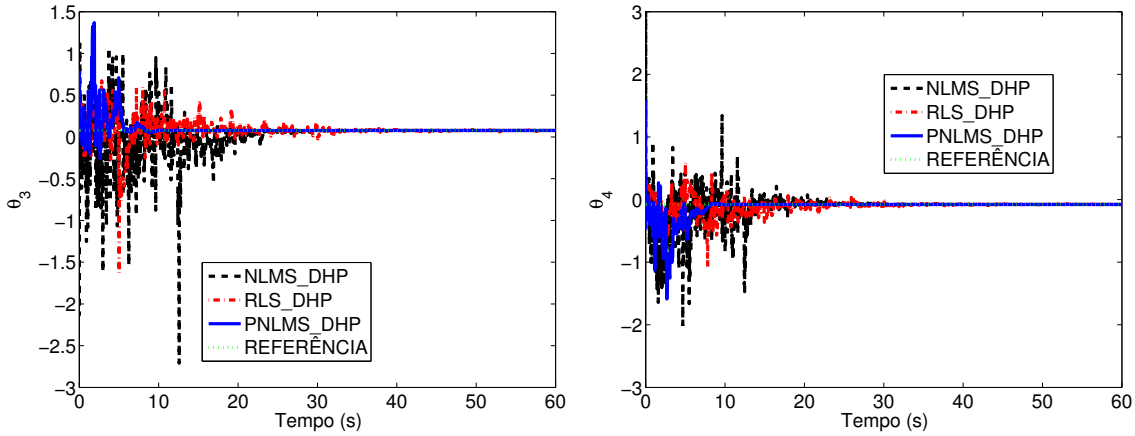


Figura 5.76: Trajetória seguida pelo Parâmetro θ_5 (esquerda) e θ_6 (direita) no processo de aprendizagem.

(a) Parâmetro θ_5. **(b)** Parâmetro θ_6.

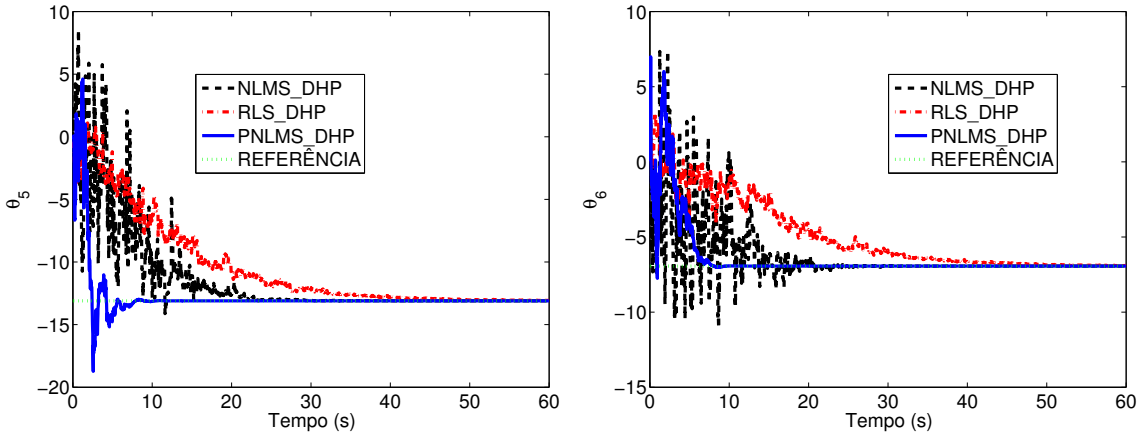


Figura 5.77: Trajetória seguida pelo Parâmetro θ_7 (esquerda) e θ_8 (direita) no processo de aprendizagem.

(a) Parâmetro θ_7. **(b)** Parâmetro θ_8.

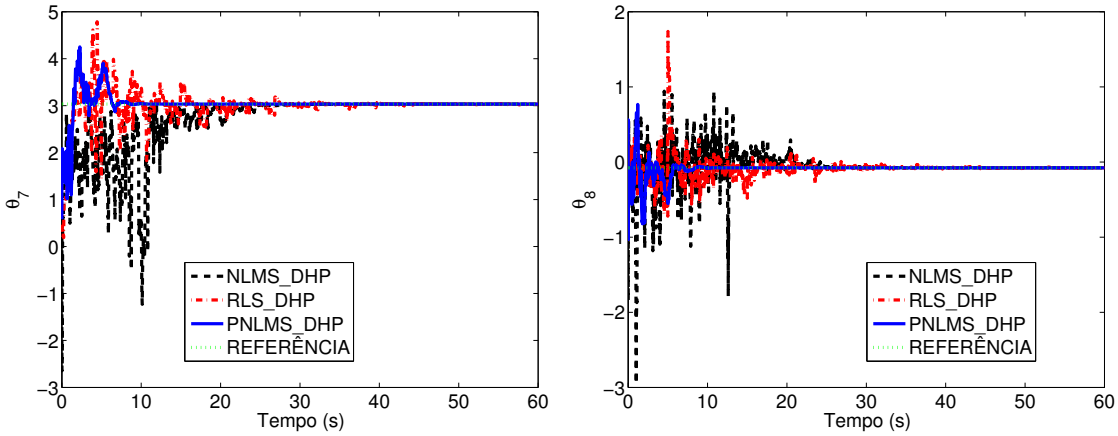


Figura 5.78: Trajetória seguida pelo Parâmetro θ_9 (esquerda) e θ_{10} (direita) no processo de aprendizagem.

(a) Parâmetro θ_9. **(b)** Parâmetro θ_{10}.

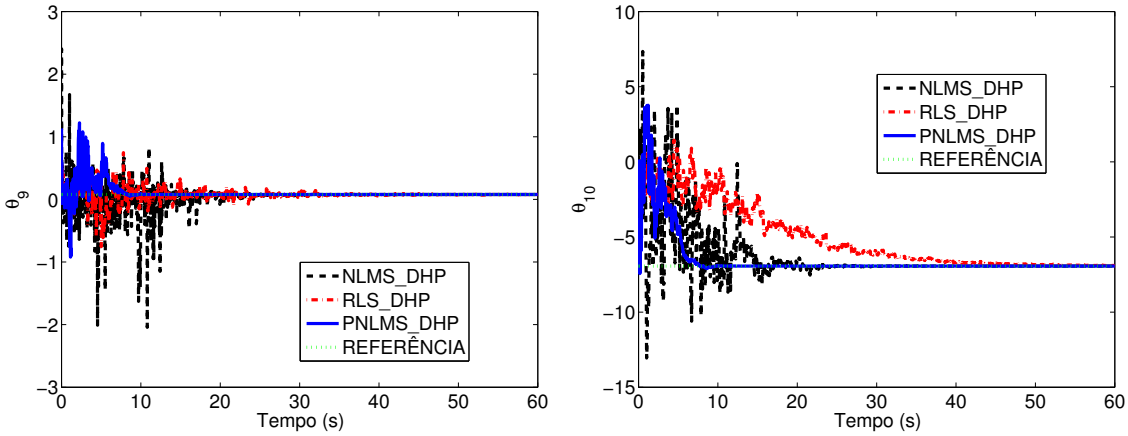


Figura 5.79: Trajetória seguida pelo Parâmetro θ_{11} (esquerda) e θ_{12} (direita) no processo de aprendizagem.

(a) Parâmetro θ_{11}. **(b)** Parâmetro θ_{12}.

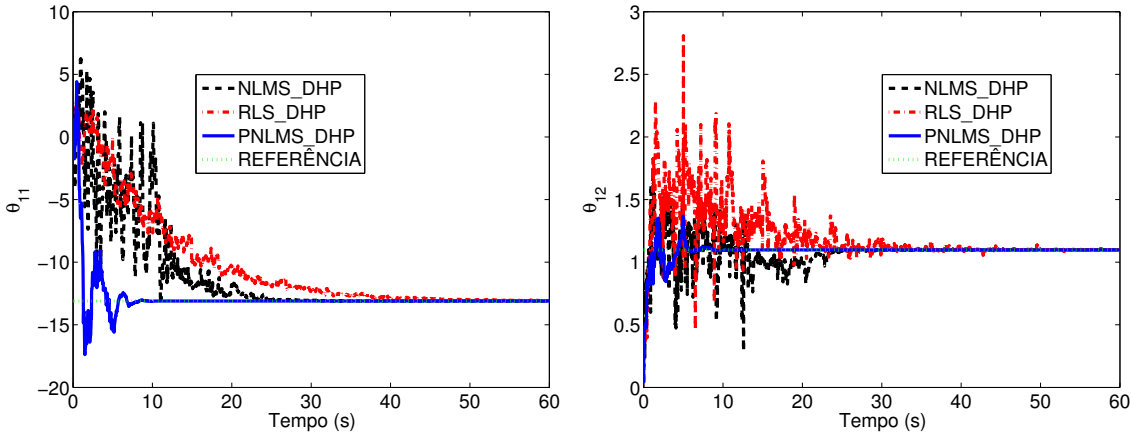


Figura 5.80: Trajetória seguida pelo Parâmetro θ_{13} (esquerda) e θ_{14} (direita) no processo de aprendizagem.

(a) Parâmetro θ_{13}. **(b)** Parâmetro θ_{14}.

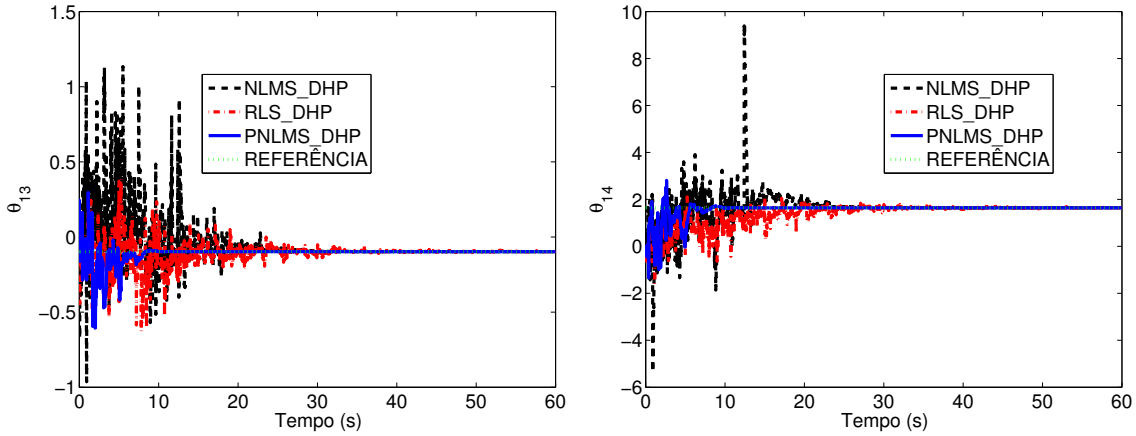


Figura 5.81: Trajetória seguida pelo Parâmetro θ_{15} (esquerda) e θ_{16} (direita) no processo de aprendizagem.

(a) Parâmetro θ_{15}. **(b)** Parâmetro θ_{16}.

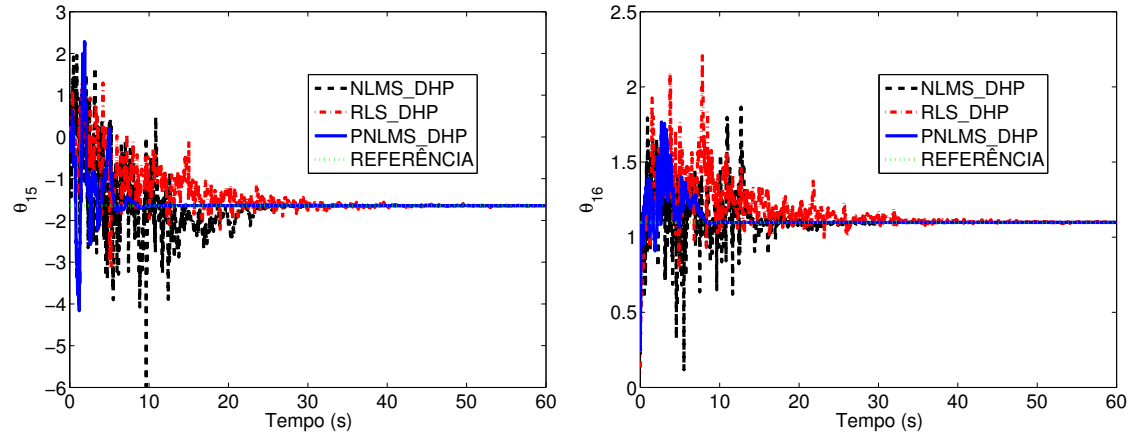


Figura 5.82: Trajetória seguida pelo Parâmetro θ_{17} (esquerda) e θ_{18} (direita) no processo de aprendizagem.

(a) Parâmetro θ_{17}. **(b)** Parâmetro θ_{18}.

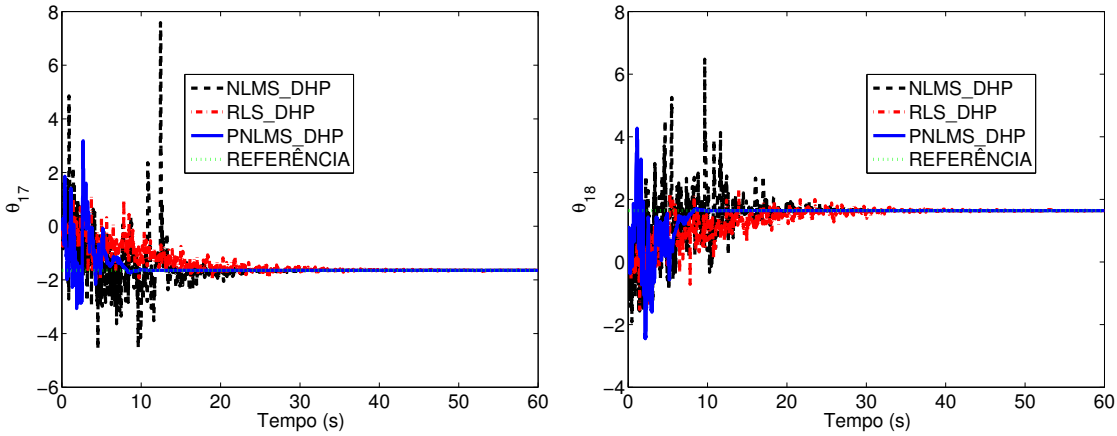


Figura 5.83: Trajetória seguida pelo Parâmetro θ_{19} (esquerda) e θ_{20} (direita) no processo de aprendizagem.

(a) Parâmetro θ_{19}. **(b)** Parâmetro θ_{20}.

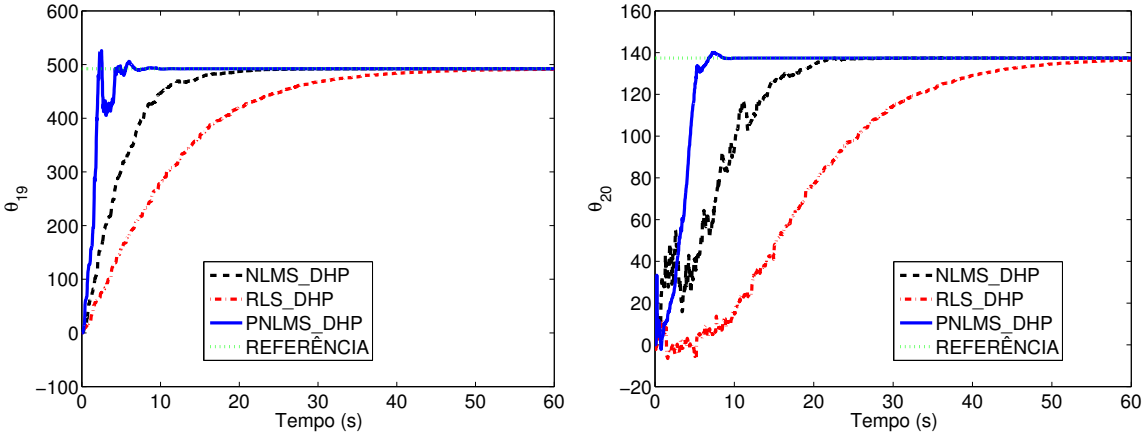
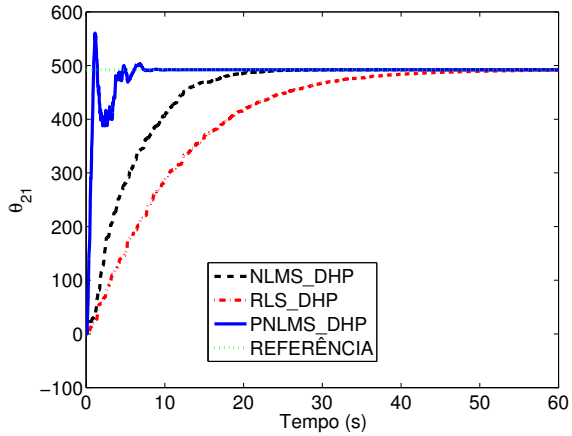


Figura 5.84: Trajetória seguida pelo Parâmetro θ_{21} no processo de aprendizagem.



Variável de Estado e Esforço de Controle: Considerando que os três estimadores, RLS, NLMS e PNLMS, convergiram para os valores esperados, selecionou-se os ganhos obtidos pelo PNLMS para traçar as trajetórias dos estados e ações de controle e compará-los aos obtidos pelo método de Schur. A tabela (5.20) mostra os ganhos encontrados na iteração 6000 do algoritmo DHP associado com os estimadores RLS, PNLMS e NLMS e o ganho obtidos pelo método de Schur. O erro entre a solução ótimo e a obtida pelo método HDP se torna menor com o aumento do tempo de simulação.

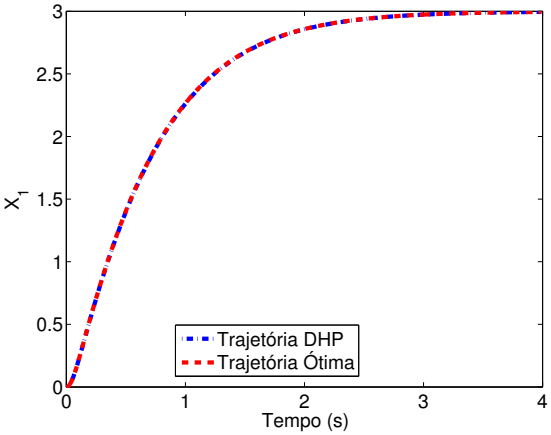
Tabela 5.20: Ganhos da política obtida na iteração 6000 do processo de aprendizagem

Algoritmo	Iteração 6000					
RLS	0,6556	0,4470	0,5474	-0,5473	-6,5358	-3,1266
	0,4474	0,6554	-0,5473	0,5474	-3,1266	-6,5362
PNLMS	0,6560	0,4478	0,5474	-0,5473	-6,5517	-3,1433
	0,4478	0,6560	-0,5473	0,5474	-3,1433	-6,5517
NLMS	0,6560	0,4478	0,5474	-0,5473	-6,5517	-3,1433
	0,4478	0,6560	-0,5473	0,5474	-3,1433	-6,5517
Ótimo	0,6559	0,4478	0,5474	-0,5472	-6,5517	-3,1433
	0,4478	0,6559	-0,5472	0,5474	-3,1432	-6,5517

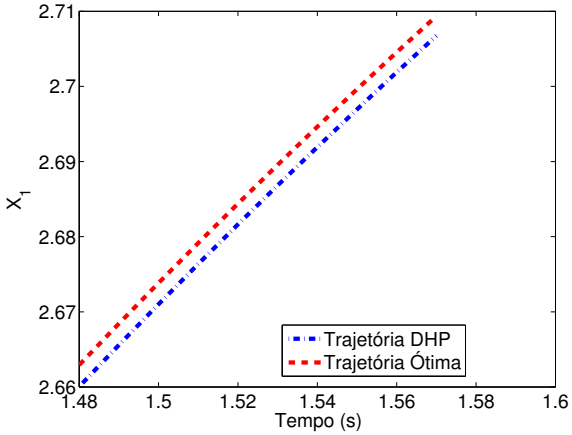
Observa-se nas figuras (5.85),(5.86) e (5.87) as trajetórias seguidas pelos estados do sistema que está inicialmente em $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ para alcançar a referência $\begin{bmatrix} 3 & 5 \end{bmatrix}^T$, sendo as saídas os estados x_1 e x_2. A figura mostra também a trajetória ótima calculada pelo método de Schur.

Figura 5.85: Trajetórias dos estados x_1 e x_2 que correspondem as saídas do sistema. A figura à direita é uma ampliação da figura a esquerda entre os instantes 1,48 e 1,58 s. Observa-se como as saídas tendem para a referência $[3 \;\; 5]^T$.

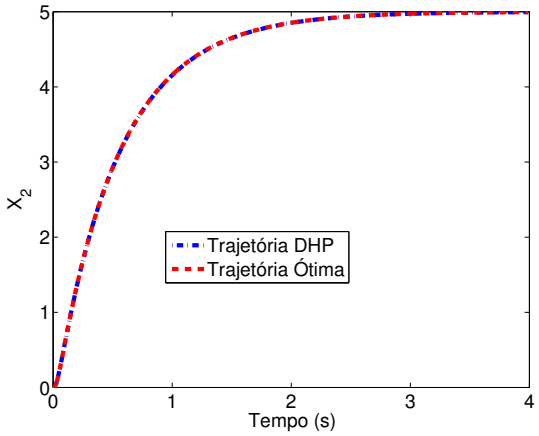
(a) Trajetória de x_1.



(b) Trajetória de x_1 ampliada entre 1,48 e 1,58 s.



(c) Trajetória de x_2.



(d) Trajetória de x_2 ampliada entre 1,48 e 1,58 s.

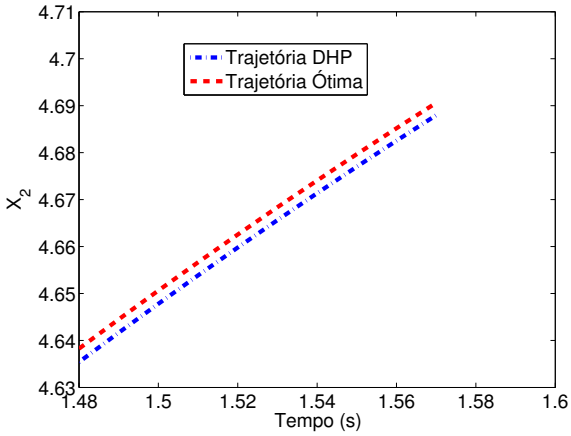
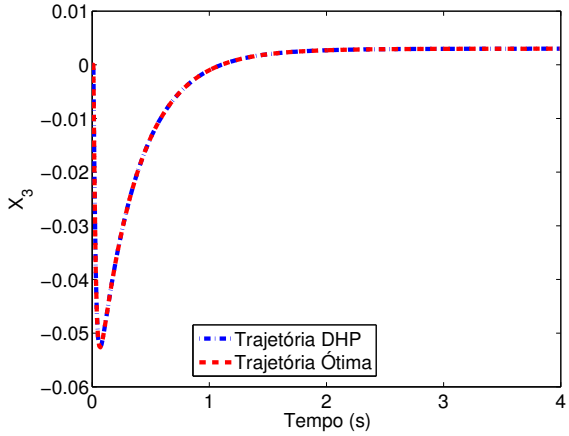
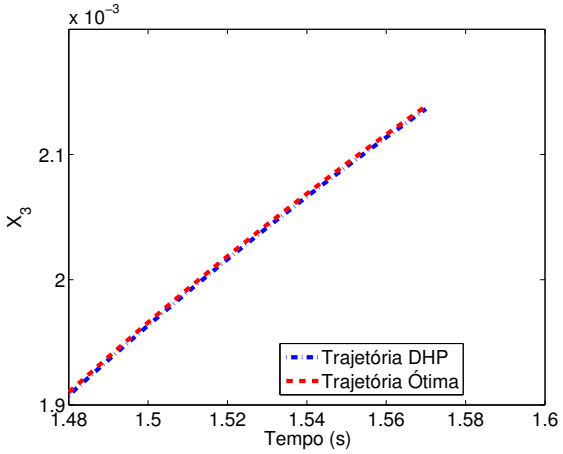


Figura 5.86: Trajetórias dos estados x_3 e x_4. A figura à direita é uma ampliação da figura a esquerda entre os instantes 1,48 e 1,58 s.

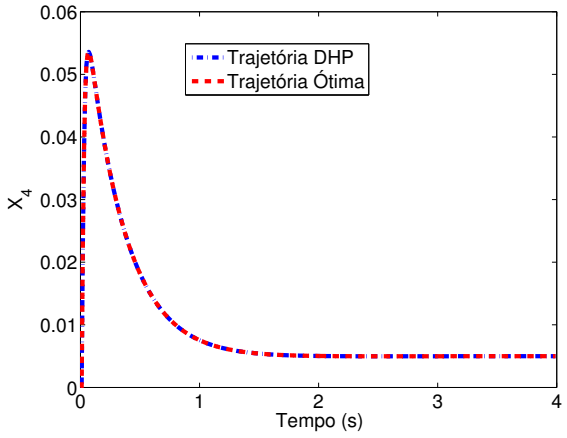
(a) Trajetória de x_3.



(b) Trajetória de x_3 ampliada entre 1,48 e 1,58 s.



(c) Trajetória de x_4.



(d) Trajetória de x_4 ampliada entre 1,48 e 1,58 s.

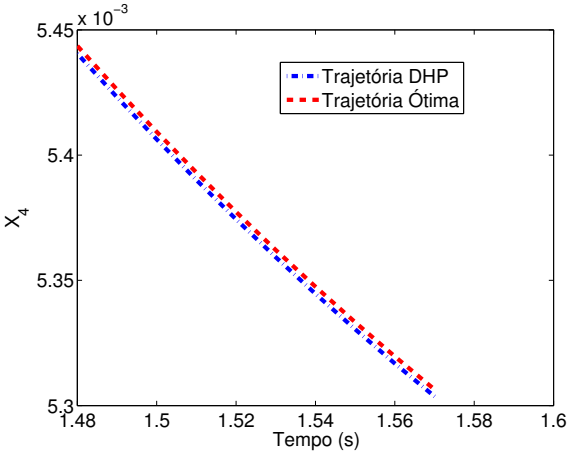
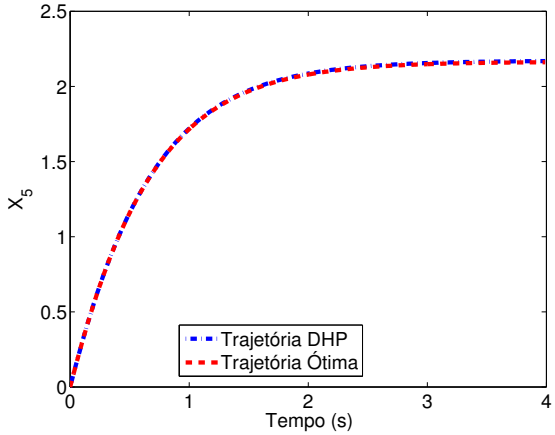
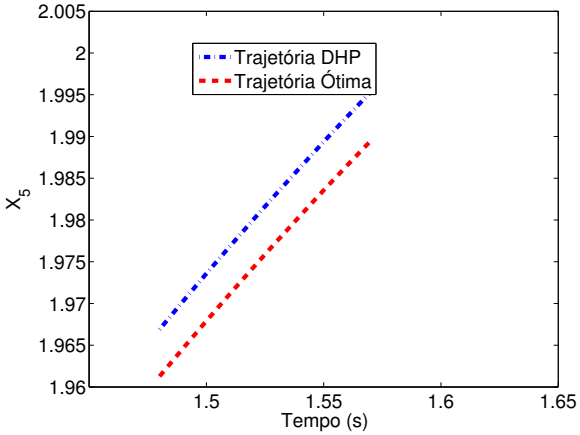


Figura 5.87: Trajetórias dos estados x_5 e x_6. A figura à direita é uma ampliação da figura a esquerda entre os instantes 1,48 e 1,58 s. Estes estados foram criados com a introdução dos integradores.

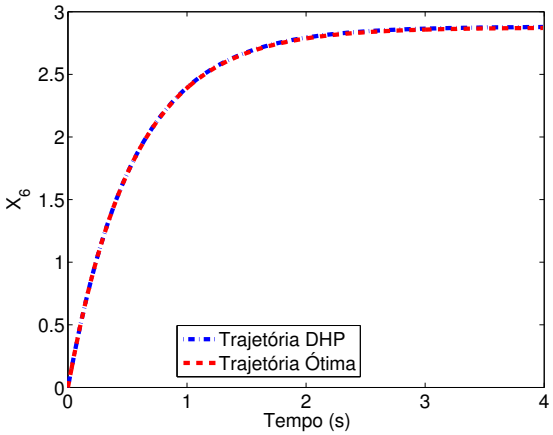
(a) Trajetória de x_5.



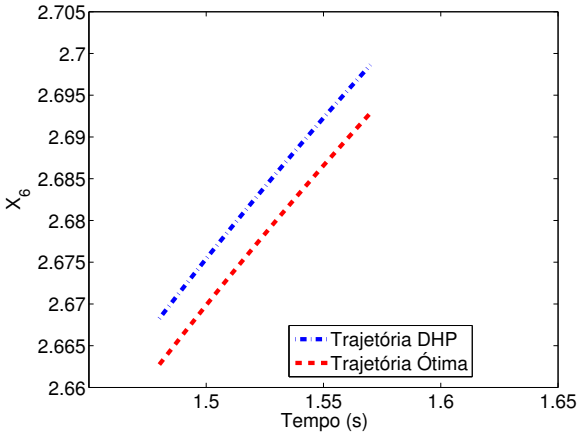
(b) Trajetória de x_5 ampliada entre 1,48 e 1,58 s.



(c) Trajetória de x_6.



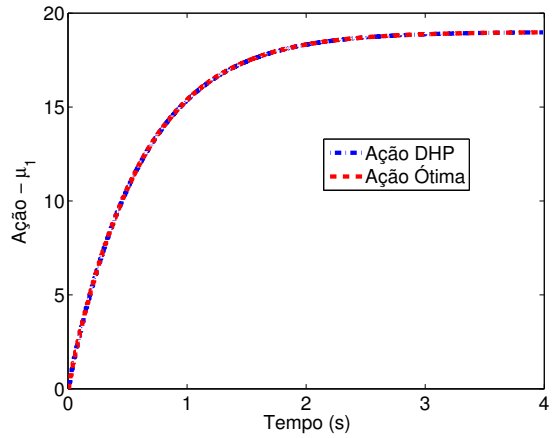
(d) Trajetória de x_6 ampliada entre 1,48 e 1,58 s.



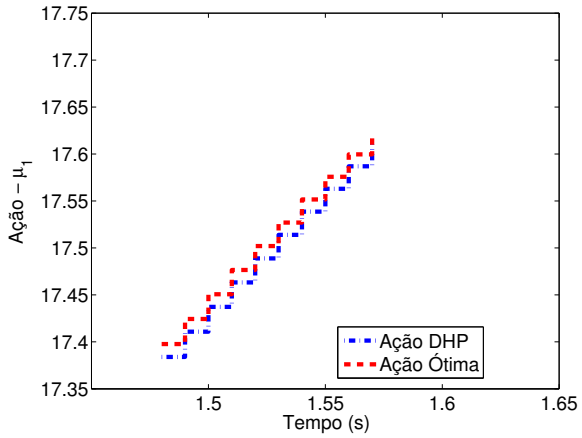
A figura (5.88) mostra as ações de controle ótima calculadas *offline* e as encontradas pelo algoritmo HDP em 60 s de simulação para gerar as trajetórias dos estados reproduzidas acima.

Figura 5.88: Ações de controle μ_1 e μ_2. A figura à direita é uma ampliação da figura a esquerda entre os instantes 1,48 e 1,58 s.

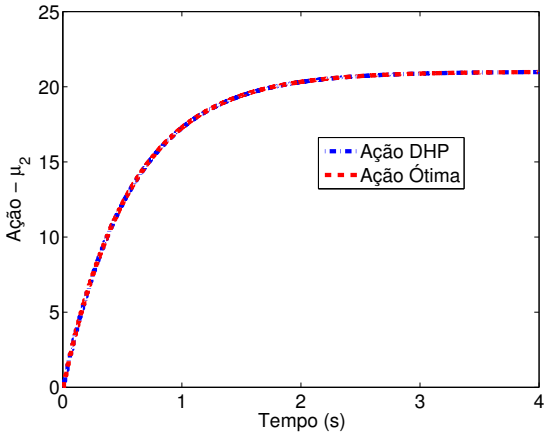
(a) Ação de Controle μ_1.



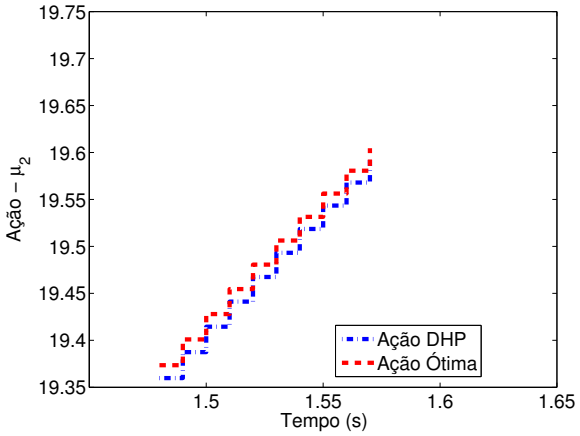
(b) Ação de Controle μ_1 ampliada entre 1,48 e 1,58 s.



(c) Ação de Controle μ_2.



(d) Ação de Controle μ_2 ampliada entre 1,48 e 1,58 s.



5.4.3.2 Conclusão

Verificou-se um melhor desempenho do algoritmo PNLMS na estimação dos parâmetros da função valor em comparação aos demais. Nota-se que com o PNLMS foi obtido um tempo de convergência em torno de 10 s, enquanto que o NLMS demorou 30 s e o RLS, 60 s. Os parâmetros desses algoritmos foram ajustados de forma a se obter os seus melhores desempenhos e estabilidade. O sistema foi excitado de forma semelhante durante a fase de aprendizagem para os três algoritmos.

5.4.4 ADDHP

Utilizando a mesma estrutura do sistema mostrado na Figura (5.73), mas, agora, aplica-se o algoritmo ADDHP associado com os estimadores IPNLMS, PNLMS e RLS

para encontrar as ações de controle ótimas.

5.4.4.1 Experimento

Parâmetros de Configuração das Simulações: Neste experimento computacional são definidos os seguintes parâmetros para o algoritmo de aprendizagem ADDHP e para o ambiente:

- O fator de desconto igual a um;
- A técnica de exploração adotada foi injetar um pequeno sinal de ruido na ação de controle (HAGEN; KRöSE, 1998) (BRADTKE; YDSTIE; BARTO, 1994) e revitalização de estado a cada 30 iterações do ADDHP;
- A política é melhorada a cada 30 avaliação da função-Q;
- Adotou-se uma politica inicial aleatória;
- Função de custo é a mesma definida na simulação para DHP aplicado ao circuito elétrico;
- São realizadas 12000 iterações equivalente a 120 segundos de simulação;
- O tempo de amostragem é de $0,01$ s;
- A planta sofre uma mudança em seus parâmetros na iteração 6000 igual à mudança feita para o DHP;

Forma Regressiva da HJB: As parametrizações do gradiente da função-Q e da política para ADDHP são dadas por

$$\left(\begin{array}{cc} \frac{\partial Q}{\partial x} & \frac{\partial Q}{\partial \mu} \end{array} \right) = \left[\begin{array}{cc} x^T & \mu^T \end{array} \right] \left[\begin{array}{cc} H_{xx} & H_{x\mu} \\ H_{\mu x} & H_{\mu\mu} \end{array} \right] = \phi^T \theta \tag{5.68}$$

e

$$\mu_k = -(H_{\mu\mu})^{-1} H_{\mu x} x_k, \tag{5.69}$$

respectivamente, sendo ϕ a matriz de regressores e θ o vetor de parâmetros. Para o caso específico do circuito elétrico, H é uma matriz simétrica de ordem 8×8, H_{xx} é matriz simétrica de ordem 6×6, $H_{x\mu}$ é matriz simétrica de ordem 6×2, $H_{\mu x}$ é matriz simétrica de ordem 2×6, $H_{\mu\mu}$ é matriz simétrica de ordem 2×2 e o vetor $z_k = \left[\begin{array}{c} x_k \\ \mu_k \end{array} \right]$ é a concatenação dos vetores de estado e ação.

Então, representado a matriz H por

$$H = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 & \theta_7 & \theta_8 \\ \theta_2 & \theta_9 & \theta_{10} & \theta_{11} & \theta_{12} & \theta_{13} & \theta_{14} & \theta_{15} \\ \theta_3 & \theta_{10} & \theta_{16} & \theta_{17} & \theta_{18} & \theta_{19} & \theta_{20} & \theta_{21} \\ \theta_4 & \theta_{11} & \theta_{17} & \theta_{22} & \theta_{23} & \theta_{24} & \theta_{25} & \theta_{26} \\ \theta_5 & \theta_{12} & \theta_{18} & \theta_{23} & \theta_{27} & \theta_{28} & \theta_{29} & \theta_{30} \\ \theta_6 & \theta_{13} & \theta_{19} & \theta_{24} & \theta_{28} & \theta_{31} & \theta_{32} & \theta_{33} \\ \theta_7 & \theta_{14} & \theta_{20} & \theta_{25} & \theta_{29} & \theta_{32} & \theta_{34} & \theta_{35} \\ \theta_8 & \theta_{15} & \theta_{21} & \theta_{26} & \theta_{30} & \theta_{33} & \theta_{35} & \theta_{36} \end{bmatrix}, \tag{5.70}$$

sua forma vetorizada fica

$$\begin{aligned} vec(H) = \theta = \Big[& \theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6 \quad \theta_7 \quad \theta_8 \quad \theta_9 \quad \theta_{10} \quad \theta_{11} \quad \theta_{12} \\ & \theta_{13} \quad \theta_{14} \quad \theta_{15} \quad \theta_{16} \quad \theta_{17} \quad \theta_{18} \quad \theta_{19} \quad \theta_{20} \quad \theta_{21} \quad \theta_{22} \quad \theta_{23} \quad \theta_{24} \\ & \theta_{25} \quad \theta_{26} \quad \theta_{27} \quad \theta_{28} \quad \theta_{29} \quad \theta_{30} \quad \theta_{31} \quad \theta_{32} \quad \theta_{33} \quad \theta_{34} \quad \theta_{35} \quad \theta_{36} \Big]^T . \end{aligned} \tag{5.71}$$

Matriz de regressor e a ação tomam as formas

$$\phi = 2 \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & \mu_1 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & x_3 & x_4 & x_5 & x_6 & \mu_1 & \mu_2 & 0 & 0 & 0 \\ 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & 0 & 0 & 0 & 0 & 0 & x_3 & x_4 & x_5 \\ 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & 0 & 0 & 0 & 0 & 0 & x_3 & 0 \\ 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & 0 & 0 & 0 & 0 & 0 & x_3 \\ 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & x_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_6 & \mu_1 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_4 & x_5 & x_6 & \mu_1 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_4 & 0 & 0 & 0 & x_5 & x_6 & \mu_1 & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_3 & 0 & 0 & 0 & 0 & x_4 & 0 & 0 & 0 & x_5 & 0 & 0 & x_6 & \mu_1 & \mu_2 & 0 & 0 & 0 \\ 0 & x_3 & 0 & 0 & 0 & 0 & x_4 & 0 & 0 & 0 & x_5 & 0 & 0 & x_6 & 0 & \mu_1 & \mu_2 & 0 \\ 0 & 0 & x_3 & 0 & 0 & 0 & 0 & x_4 & 0 & 0 & 0 & x_5 & 0 & 0 & x_6 & 0 & \mu_1 & \mu_2 \end{bmatrix} \tag{5.72}$$

e

$$\mu_k = \begin{bmatrix} \theta_{34} & \theta_{35} \\ \theta_{35} & \theta_{36} \end{bmatrix}^{-1} \begin{bmatrix} \theta_7 & \theta_{14} & \theta_{20} & \theta_{25} & \theta_{29} & \theta_{32} \\ \theta_8 & \theta_{15} & \theta_{21} & \theta_{26} & \theta_{30} & \theta_{33} \end{bmatrix} x_k \tag{5.73}$$

$$= \begin{bmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 \\ k_7 & k_8 & k_9 & k_{10} & k_{11} & k_{12} \end{bmatrix} x_k, \tag{5.74}$$

respectivamente.

O alvo para ADDHP é um vetor dado por

$$\left(\frac{\partial Q}{\partial x} \quad \frac{\partial Q}{\partial \mu}\right)=2\left\{\begin{bmatrix} x_k^T & \mu_k^T \end{bmatrix}\begin{bmatrix} Q & \mathbf{0} \\ \mathbf{0} & R \end{bmatrix}+\begin{bmatrix} x_{k+1}^T & \mu_{k+1}^T \end{bmatrix} H \begin{bmatrix} A & B \\ KA & KB \end{bmatrix}\right\}, \tag{5.75}$$

Pode-se ver que o alvo depende da estimativa atual da matriz H, da planta e das matrizes Q e R da função de custo.

Estimação da Função Valor: Neste caso, o gradiente da função-Q tem como parâmetros os elementos da matriz simétrica H de ordem 8×8. Portanto, 36 parâmetros devem ser estimados, mas é mostrada a convergência dos ganhos de retroação que são concebidos a partir da matriz H.

Os algoritmos IPNLMS, PNLMS e RSL e são ajustados conforme tabela (5.21).

Tabela 5.21: Ajuste dos Parâmetros Livres para RLS, IPNLMS, PNLMS

Algoritmos	Parâmetros Livres				
	ρ	α	μ	λ	P
RLS				0,995	I_n
IPNLMS		0,6	1,2		
PNLMS	0,0001		1,2		

As figuras abaixo mostram as trajetórias seguidas pelos ganhos de retração durante o processo de aprendizagem e os ganhos ótimos (em verde) correspondentes.

Figura 5.89: Trajetória seguida pelo Parâmetro k_1 e k_2 no processo de aprendizagem

(a) Trajetória k_1. **(b)** Trajetória k_2 .

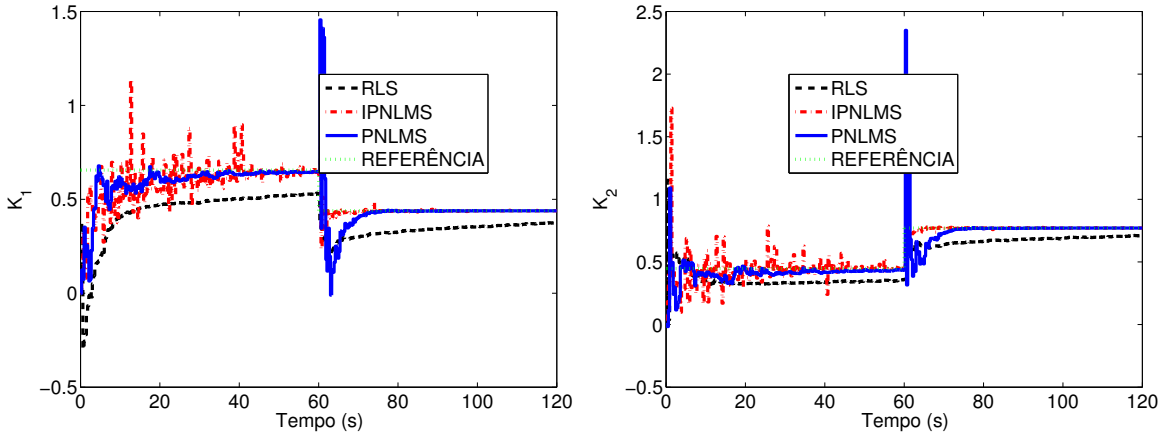


Figura 5.90: Trajetória seguida pelo Parâmetro k_3 e k_4 no processo de aprendizagem

(a) Parâmetro k_3. **(b)** Parâmetro k_4.

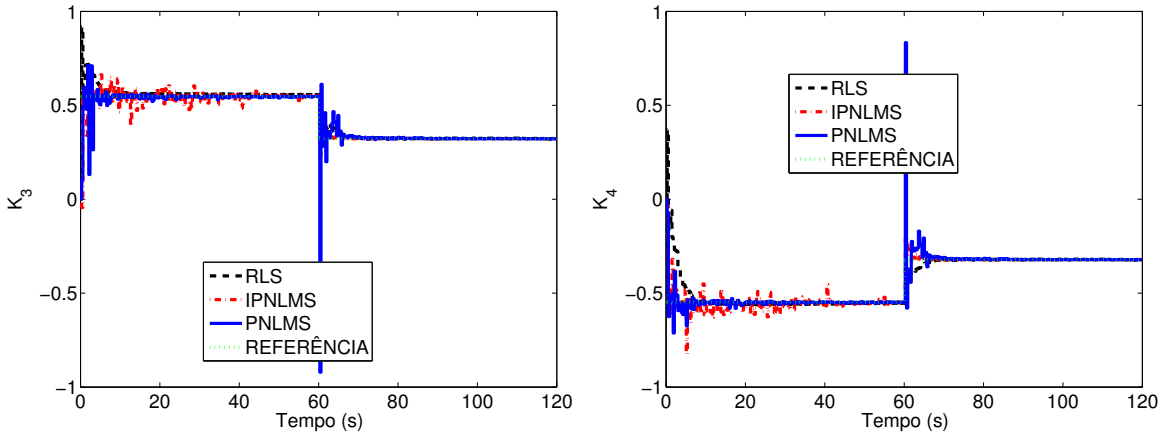


Figura 5.91: Trajetória seguida pelo Parâmetro k_5 e k_6 no processo de aprendizagem

(a) Parâmetro k_5. **(b)** Parâmetro k_6.

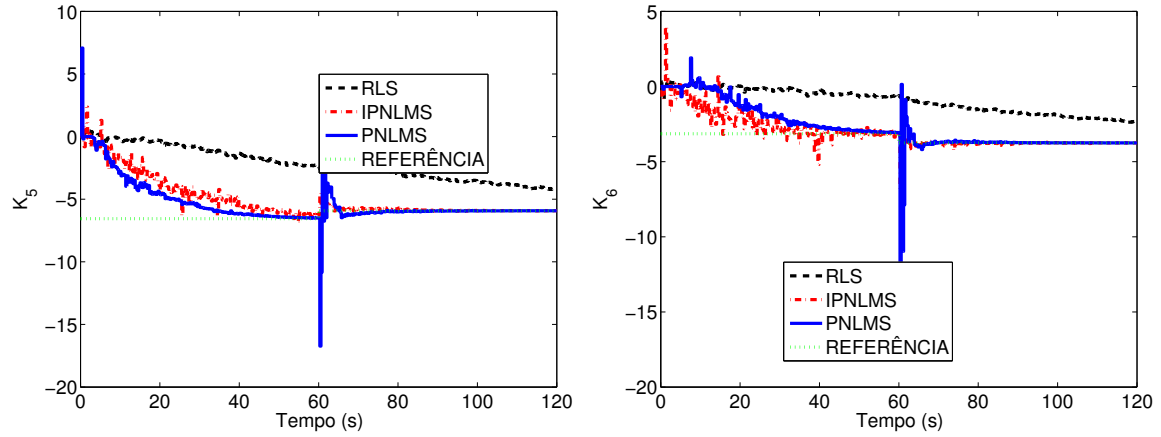


Figura 5.92: Trajetória seguida pelo Parâmetro k_7 e k_8 no processo de aprendizagem

(a) Parâmetro k_7. **(b)** Parâmetro k_8.

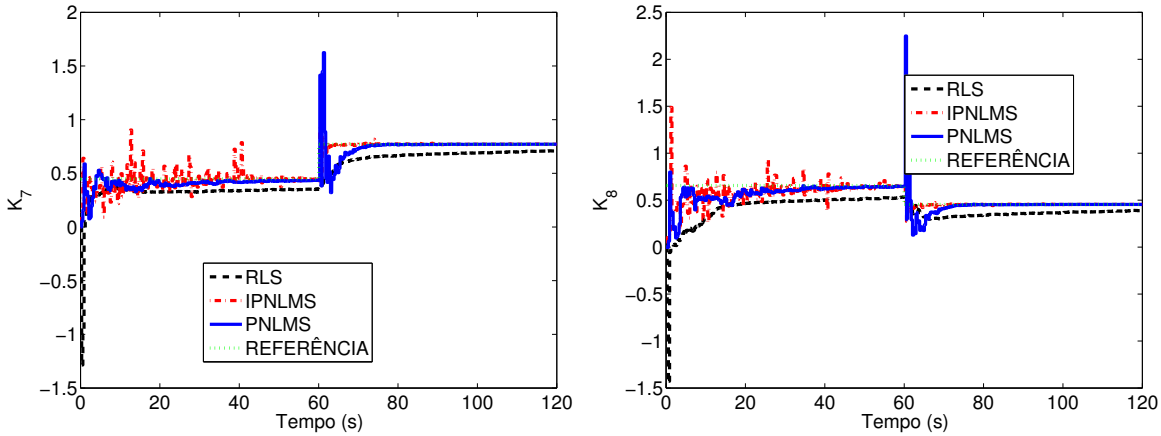


Figura 5.93: Trajetória seguida pelo Parâmetro k_9 e k_{10} no processo de aprendizagem

(a) Parâmetro k_9. **(b)** Parâmetro k_{10}.

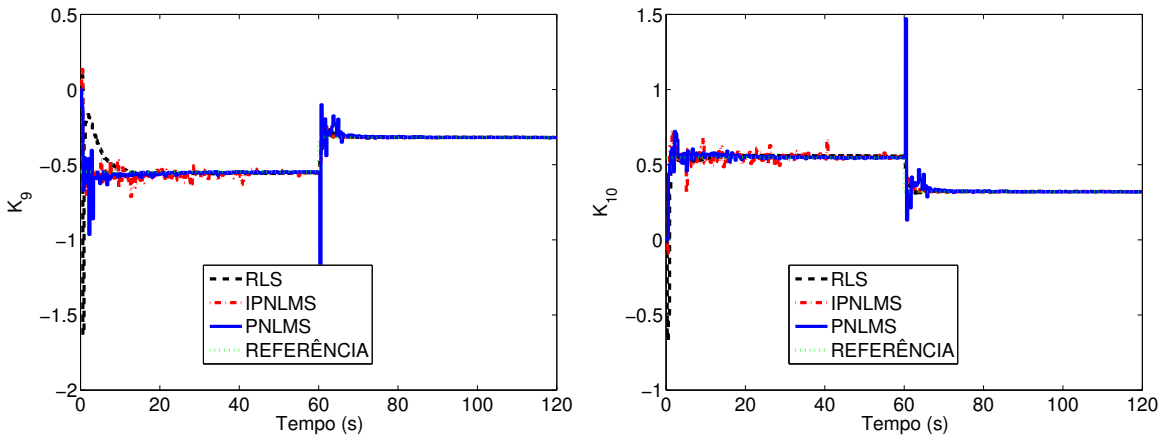
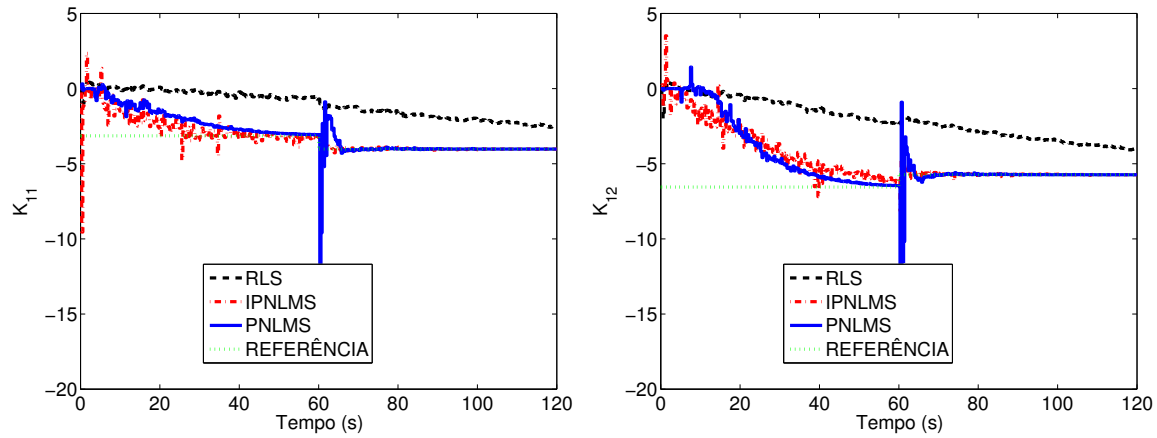


Figura 5.94: Trajetória seguida pelo Parâmetro k_{11} e k_{12} no processo de aprendizagem

(a) Parâmetro k_{11}. **(b)** Parâmetro k_{12} - Mostrando detalhe.



5.4.4.2 Conclusão

Nota-se que, para a configuração de simulação adotada, os estimadores proporcionais foram superiores em relação à velocidade de convergência dos parâmetros da função-Q. A política final encontrada com esses algoritmos está bem próxima da ótima calculada pelo método de Schur.

Para os estimadores proporcionais só foi possível obter convergência com uma boa estimação da função-Q antes da atualização da política.

Resultado excelente é obtido com o RLS quando a abordagem completamente otimista é implementada, ou seja, quando a avaliação e a atualização da política se dá na mesma iteração. O RLS com esta abordagem e o PNLMS e IPNLMS com as configurações acima têm um comportamento semelhante.

6

Conclusão

Sumário

Neste trabalho, realizou-se um estudo de técnicas de inteligencia computacional para resolver problema de controle ótimo e adaptativo de sistema dinâmico. Controle ótimo é projetado *offline* com um completo conhecimento do processo e projeto de controle adaptativo geralmente não está relacionado a uma solução ótima. Nesse sentido, projetos de controladores ótimos e adaptativos foram apresentados. Especificamente, os métodos de aprendizagem por reforço HDP, DHP, ADHDP e ADDHP são utilizados para resolver *online*, com dados medidos ao longo da trajetória, a equação Hamilton-Jacobi-Bellman e com as parametrizações da política de controle e da função valor induzidas pelo problema do DLQR.

Dois processos diferentes ambos MIMO foram empregados na validação computacional dos algoritmos, um de terceira ordem com duas entradas e três saídas correspondendo à dinâmica longitudinal de uma aeronave e o outro de quarta ordem, com duas entradas e duas saídas, correspondendo a um modelo matemático de um circuito elétrico. A função de custo instantânea utilizada é uma forma quadrática dos estados e das ações de controle.

Primeiramente, fez-se a implementação do algoritmo HDP associado aos algoritmos RLS, PNLMS e LMS aplicado à aeronave. Foram realizadas três simulações diferentes todas utilizado a estratégia de revitalização de estado como excitação do sistema, a primeira com limitação do espaço de estado e com atualização da política a cada transição de estado, a segunda com atualização da política a cada cinquenta iterações de estimativa da função-V, continuando com a limitação no espaço de estado e a terceira simulação com variação da função de custo durante a execução do algoritmo e sem limitação do ambiente. Em todos os caso, O HDP convergiu para os valores esperados e uma política ótima foi encontrada. O sistema ficou estável durante o processo de aprendizagem para as três simulações. Para o mesmo processo, o HDP convergiu mais rápido quando foi considerado um maior intervalo de linearidade do processo.

Após, implementou-se o algoritmo ADHDP associados aos RLS, PNLMS e IPNLMS com um ruido branco na entrada do processo como estratégia de excitação persistente ou exploração. Este algoritmo é independente do ambiente, em compensação, a convergência para a política ótima é mais lenta que do HDP. O ADHDP, para o processo em questão, precisou estimar 15 parâmetros contra 6 do HDP.

A seguir, foi implementado o DHP associado aos RLS, PNLMS e IAF-PNLMS e como técnica de excitação a injeção de ruído na ação de controle. Verificou-se uma rápida convergência dos parâmetros estimados superando os algoritmos HDP e ADHDP, mas é necessário um conhecimento completo do processo.

Por fim, o ADDHP associado aos NLMS, PNLMS, IAF-PNLMS foi implementado. Foi necessário, como o ADHDP, estimar 15 parâmetros, no entanto, o ADDHP teve um comportamento melhor a custa de mais conhecimento da dinâmica do ambiente.

Foram implementados, ainda, os algoritmos HDP, ADHDP, DHP e ADDHP aplicados a um circuito elétrico. Nestes casos, para mostrar a questão da adaptabilidade e otimalidade,

a planta sofreu mudança de parâmetros durante as simulações. Também foram incluídos integradores na entrada da planta, aumentado sua ordem, com o objetivo de reduzir a zero o erro da saída quando o sistema segue uma referência em degrau. Neste último caso e para o ADDHP, a quantidade de parâmetros a estimar aumentou de 21 para 36.

O que se buscou mostrar aqui é a viabilidade do uso de algoritmos simples que exigem bem menos esforço computacional e que, mesmo assim, ainda têm um comportamento equivalente a outros mais complexos quando certas restrições são impostas. No caso concreto, os algoritmos proporcionais, como o PNLMS, obtiveram um desempenho tão bom quanto o RLS e com menor custo computacional em plantas com alto grau de esparsidade e com conhecimento exato da estrutura das função valor e da política de controle.

6.1 Trabalhos Futuros

O desafio maior da teoria de controle é encontrar maneiras de controlar sistemas dinâmicos sem conhecimento prévio do seu comportamento, mantendo-o estável e cumprindo determinados desempenho e objetivos. O projetista deve determinar qual algoritmo de aprendizado usar, como definir seus parâmetros, e como representar a solução do agente (WHITESON, 2010). Por causa da natureza dos problemas de controle, poucas são as metodologias de aprendizagem por reforço que podem ser aplicadas *online*.

Algumas sugestões de estudo, investigação e desenvolvimento para trabalhos futuros são listadas abaixo.

- Implementar em hardware as metodologias propostas;
- Utilizar outras formas de aproximação de função como rede neurais, fuzzy e métodos de *kernel*;
- Estender os algoritmos a ambientes não lineares, principalmente o ADHDP que independe de modelo;
- Aplicar os métodos GHDP e GADHDP utilizando os algoritmos da família LMS e RLS em problemas de controle DLQR;
- Aplicar a aprendizagem por reforço *online* e programação dinâmica a sistema de tempo contínuo para o caso particular do LQR.

Referências Bibliográficas

AGUIRRE, L. *Introdução à Identificação de Sistemas – Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais*. 3. ed. [S.l.]: Editora UFMG, 2007. ISBN 9788570415844.

ÅSTRÖM, K.; WITTENMARK, B. *Adaptive Control*. [S.l.]: Dover Publications, 2008. (Dover Books on Electrical Engineering Series). ISBN 9780486462783.

BELLMAN, R. *Dynamic Programming*. [S.l.]: Dover Publications, 2003. (Dover Books on Computer Science Series). ISBN 9780486428093.

BENESTY, J.; GAY, S. L. An improved pnlms algorithm. *IEEE Int. Conf. Acoust*, v. 2, p. 1881–1884, may 2002.

BERTSEKAS, D. P. *Dynamic Programming and Optimal Control*. [S.l.]: Athena Scientific, 1995.

BERTSEKAS, D. P.; TSITSIKLIS, J. N. *Neuro-Dynamic Programming*. 1st. ed. [S.l.]: Athena Scientific, 1996. ISBN 1886529108.

BRADTKE, S. J.; YDSTIE, B. E.; BARTO, A. G. *Adaptive Linear Quadratic Control Using Policy Iteration*. [S.l.], 1994.

BUSONIU, L. et al. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. [S.l.]: CRC Press, 2010.

DUTTWEILER, D. L. Proportionate normalized least-mean-squares adaptation in echo cancellers. *IEEE Transactions on Speech and Audio Processing*, v. 8, p. 508–517, 2000.

HAGEN, S. T.; KRöSE, B. *Linear Quadratic Regulation using Reinforcement Learning*. 1998.

HAYKIN, S. *Neural Networks and Learning Machines*. [S.l.]: Prentice Hall, 2008. ISBN 0131471392.

HAYKIN, S.; WIDROW, B. *Least-Mean-Square Adaptive Filters*. [S.l.]: Wiley, 2003. (Adaptive and Learning Systems for Signal Processing, Communications and Control Series). ISBN 9780471215707.

HAYKIN, S. S. *Redes Neurais Princípios e Práticas*. 2. ed. Porto Alegre: Bookman, 2001.

HUANG, Y.; BENESTY, J.; CHEN, J. *Acoustic MIMO Signal Processing (Signals and Communication Technology)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN 3540376305.

IZMAILOV, A.; SOLODOV, M. *Otimização, volume 1: Condições de Otimalidade, Elementos de Análise e de Dualidade*. 2. ed. Rio de Janeiro: Instituto de Matematica Pura e Aplicada - IMPA, 2009.

JOHNSON, C. *Lectures on Adaptive Parameter Estimation*. [S.l.]: Pearson Education Canada, 1988. (Prentice-Hall Information and System Sciences Series). ISBN 9780135281260.

KIRK, D. E. *Optimal Control Theory: An Introduction*. 1st. ed. New Jersey: Prentice-Hall, 1970.

LENDARIS, G. G. A retrospective on adaptive dynamic programming for control. In: *Proceedings of the 2009 International Joint Conference on Neural Networks*. Piscataway, NJ, USA: IEEE Press, 2009. (IJCNN'09), p. 945–952. ISBN 978-1-4244-3549-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1704175.1704314>.

LEWIS, F. L.; LIU, D. *Reinforcement learning and approximate dynamic programming for feedback control*. [S.l.]: John Wiley and Sons, 2013.

LEWIS, F. L.; VRABIE, D. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 2009.

LEWIS, F. L.; VRABIE, D.; VAMVOUDAKIS, K. G. Reinforcement learning and feedback control. *IEEE Circuits and Systems Magazine*, 2012.

LIMA, E. *Elementos de topologia geral*. 3. ed. Rio de Janeiro: SBM, 2009. (Instituto de Matemática Pura e Aplicada. Coleção elementos de matemática).

MACIEL, A. J. F. *Convergência do Estimador RLS para Algoritmos de Programação Dinâmica Heurística*. Dissertação (Mestrado) — Universidade Federal do Maranhão, 2012.

OGATA, K. *Modern Control Engineering*. 5. ed. [S.l.]: Prentice Hall, 2002.

PROKHOROV, D.; WUNSCH, D. Adaptive critic designs. *IEEE Transactions on Neural Networks*, v. 8, p. 997–1007, 1997.

RUSSELL, S.; NORVIG, P. *Inteligência artificial*. [S.l.]: Elsevier : Campus, 2004. ISBN 9788535211771.

SOUZA, F. C. d. *Algoritmos Adaptativos LMS Normalizados Proporcionais: Proposta de um novo Algoritmo e sua Modelagem Estocástica*. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2012.

SOUZA, F. C. d. et al. A pnlms algorithm with individual activation factors. *IEEE Trans*, v. 58, n. 4, p. 2036–2047, Jun 2010.

STEVENS, B.; LEWIS, F. *Aircraft control and simulation*. Wiley, 1992. (Wiley-interscience publication). ISBN 9780471613978. Disponível em: <http://books.google.com.br/books?id=b6dTAAAAMAAJ>.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. [S.l.]: The MIT Press, 1998.

SZEPESVARI, C. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010. (Synthesis lectures on artificial intelligence and machine learning). ISBN 9781608454921. Disponível em: <http://books.google.com.br/books?id=qwtphfl7U74C>.

WERBOS, P. J. Neural networks for control. In: MILLER III, W. T.; SUTTON, R. S.; WERBOS, P. J. (Ed.). Cambridge, MA, USA: MIT Press, 1990. cap. A Menu of Designs for Reinforcement Learning over Time, p. 67–95. ISBN 0-262-13261-3.

WERBOS, P. J. Approximate dynamic programming for real-time control and neural modeling. In: WHITE, D. A.; SORGE, D. A. (Ed.). *Handbook of Intelligent Control*. Van Nostrand Reinhold, NY, USA: MIT Press, 1994. p. 493–525.

WHITESON, S. *Adaptive Representations for Reinforcement Learning*. 1. ed. [S.l.]: Springer, 2010.

WIERING, M.; OTTERLO, M. van. *Reinforcement Learning: State-of-the-Art*. [S.l.]: Springer, 2011.