UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Leandro Rocha Lopes

APRENDIZAGEM POR REFORÇO E PROGRAMAÇÃO DINÂMICA ADAPTATIVA PARA PROJETO E AVALIAÇÃO DO DESEMPENHO DE ALGORITMOS DLQR EM SISTEMAS MIMO

São Luís
2011

Leandro Rocha Lopes

APRENDIZAGEM POR REFORÇO E PROGRAMAÇÃO DINÂMICA ADAPTATIVA PARA PROJETO E AVALIAÇÃO DO DESEMPENHO DE ALGORITMOS DLQR EM SISTEMAS MIMO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

São Luís
2011

Lopes, Leandro Rocha

Aprendizagem por reforço e programação dinâmica adaptativa para síintese de controladores ótimos do tipo DLQR em sistemas MIMO/ Leandro Rocha Lopes - São Luís, 2011.

130f.

Impresso por computador (fotocópia).
Orientador: João Viana da Fonseca Neto.

Dissertação (Mestrado) - Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade, 2011.

1. Programação dinâmica. 2. Controle ótimo. 3. HDP. 4. Convergência.I.Título.

CDU 519.857

APRENDIZAGEM POR REFORÇO E PROGRAMAÇÃO DINÂMICA ADAPTATIVA PARA PROJETO E AVALIAÇÃO DO DESEMPENHO DE ALGORITMOS DLQR EM SISTEMAS MIMO

Leandro Rocha Lopes

Dissertação aprovada em 04 de abril de 2011.

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

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

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

"[...] não importa tanto o tema da tese quanto a experiência de trabalho que ela comporta"

Umberto Eco

A Deus, aos meus pais Conceição e Juareis e a minha irmã Paula

Agradecimentos

Em primeiro lugar a Deus, a quem sempre recorri nos momentos difíceis.

Ao meu orientador, Profº. Dr. João Viana da Fonseca Neto, pela orientação segura, amizade e companheirismo durante a realização deste trabalho.

Aos grandes amigos do LAC e LCP que me acompanharam e me ajudaram nessa jornada, Dennis Fabrício, Gustavo de Andrade, João Inácio, Aline Coelho, Ernesto, Ana Dulce e Sarah Mesquita. A todos que contribuíram de forma direta ou indireta nesse grandioso trabalho.

A UFMA, PPGEE, DEE e a CAPES pelos recursos materiais e financeiros destinados a esse projeto.

Resumo

Em decorrência do crescente desenvolvimento tecnológico e das consequentes aplicações industriais, técnicas de controle de alto desempenho e aprendizado por reforço estão sendo desenvolvidas não só para solucionar novos problemas, mas também para melhorar o desempenho de controladores já implementados em sistemas do mundo real. As abordagens do aprendizado por reforço e do regulador linear quadrático discreto (DLQR) são conectadas pelos métodos de programação dinâmica adaptativa. Esta união é orientada para o projeto de controladores ótimos em sistemas multivariáveis (MIMO). O método proposto para sintonia de controladores DLQR fornece diretrizes para construção de heurísticas polarizadas que são aplicadas na seleção das matrizes de ponderação da recompensa instantânea. Investiga-se o desempenho das heurísticas associadas com a sintonia de controladores lineares discretos e aspectos de convergência que estão relacionados com as variações QR nos algoritmos de programação dinâmica heurística (HDP) e Ação Dependente (ADHDP). Os algoritmos e a sintonia são avaliados pela capacidade em estabelecer a política de controle ótimo que mapeia o plano-Z em um sistema dinâmico multivariável de terceira ordem.

Palavras-Chave: Programação Dinâmica, Controle Ótimo, HDP, *Q-Function*, ADHDP, Sistemas Multivariáveis, Convergência, DLQR.

Abstract

Due to the increasing of technological development and its associated industrial applications, control design methods to attend high performance requests and reinforcement learning are been developed, not only, to solve new problems, as well as, to improve the performance of implemented controllers in the real systems. The reinforcement learning (RL) and discrete linear quadratic regulator (DLQR) approaches are connected by adaptive dynamic programming (ADP). This connection is oriented to the design of optimal controller for multivariable systems (MIMO). The proposed method for DLQR controllers tuning can been heuristic guidance for biased variations in weighting matrices of instantenous reward. The heuristics performance are evaluated in terms of convergence of heuristic dynamic programming (HDP) and action dependent (AD-HDP) algorithms. The algorithms and tuning are evaluated by the capability to map the plane-Z in MIMO dynamic system of third order.

Keywords: Dynamic Programming, Optimal Control, HDP, Q-Function, ADHDP, Multivariable Systems, Convergence, DLQR.

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas e Siglas

AD	*Action Dependent*
AD-HDP	*Action Dependent Heuristic Dynamic Programming*
ADP	*Adaptive* (*Approximate*) *Dynamic Programming*
AR	Aprendizagem por Reforço
DARE	*Discrete Algebraic Riccati Equation*
DLQR	*Discrete Linear Quadratic Regulator*
GPI	*Generalized Policy Iteration*
HDP	*Heuristic Dynamic Programming*
IA	Inteligência Artificial
LQR	*Linear Quadratic Regulator*
LS	*Least Squares*
MIMO	*Multiple Input - Multiple Output*
MSH - QR	Método de Sintonia Heurística QR
PD	Programação Dinâmica
PDM	Processo de Decisão *Markoviano*
PI	Política de Iteração
TD	Temporal Diferencial
VFA	*Value Function Approximation*
VI	Valor de Iteração

Sumário

Capítulo 1

INTRODUÇÃO

O controle automático desempenha um papel importante na engenharia e na ciência, sendo de vital relevância em ambientes industriais. Os sistemas dinâmicos, tem adquirido um grau elevado de complexidade, que acaba por exigir estratégias de controle cada vez mais robustas. Projetos clássicos de sistema de controle são geralmente processos de tentativa e erro em que diversos métodos de análise são utilizados de forma interativa para determinar os parâmetros aceitáveis de projeto.

Um desempenho aceitável, está geralmente definido em termos de margem de fase e ganho, frequência de banda, maxima elevação, tempo de subida e tempo de acomodação, porem não são otimizados (BERTSEKAS, 1995) (KIRK, 1970) (KUO, 1980). Entretanto em sistemas de múltiplas entradas e saídas (DOYLE e STEIN, 1981), em que a complexidade se torna maior, necessita-se conhecer técnicas modernas de controle com novas tecnologias. Uma abordagem para estes sistemas complexos pode-se denominar de controle ótimo, que tornou-se altamente viabilizado pelo desenvolvimento dos computadores digitais (KIRK, 1970).

Métodos de controle ótimo tem alcançado grande sucesso para sistemas lineares (através das equações de *Riccati*), mas o sucesso é restrito a algumas aplicações, visto que os métodos exigem o conhecimento da dinâmica da planta, e não é de fácil aplicabilidade *online*. Em algumas aplicações, em que a dinâmica da planta varia bastante, o projeto de controlador fixo, resultante dos métodos clássicos, não são suficientes (LENDARIS, 2009).

Uma série de questões difíceis relacionadas a esta tarefa foram tratados e resolvidos, o que implica o desenvolvimento bem sucedido de conceitos matemáticos e algoritmos. Um deles foi o método de *Bellman* de Programação Dinâmica (PD) (BERTSEKAS, 1995) (BELLMAN, 1958)(BELLMAN, 2003), e a subsequente

equação *Hamilton-Jacobi-Bellman* (HJB) para resolver uma ampla classe de problemas de otimização não linear. Porém, o alto custo computacional associado com a realização dessas formulações, trouxe dificuldade na aplicabilidade desse procedimento de forma *online* (por exemplo o aspecto da resolução por recorrência da Programação Dinâmica) (LENDARIS, 2009).

Durante a década atual, o trabalho significativo ocorreu no campo da inteligência computacional, como em Lógicas *Fuzzy*, Redes Neurais e computação evolutiva (GUPTA e SINHA, 1995), relativas à aplicação na tarefa de otimização de controle através de um método que emprega Críticos Adaptativos (Ator-Crítico) (LENDARIS, 2009). Nos últimos anos, esta abordagem tem sido chamada de *Adaptive Dynamic Programming ou Approximate Dynamic Programming* - ADP (Programação Dinâmica Adaptativa) (LENDARIS, 2009). Esta, evoluiu a partir da abordagem denominada Aprendizagem por Reforço - AR (*Reinforcement Learning* - RL) (WERBOS, 2008).

O Controle ótimo é, geralmente, uma técnica de projeto *offline* que exige pleno conhecimento da dinâmica do sistema, por exemplo, no caso do sistema linear, deve-se resolver a equação de *Riccati*. Por outro lado, o controle adaptativo é um corpo de técnicas de projeto *online*, que usa os dados medidos ao longo da trajetória do sistema para proporcionar um desempenho garantido por meio da compensação da dinâmica desconhecida, perturbações e erros de modelagem.

Alguns métodos de projetos de controladores, que levam em consideração as variações dinâmicas *online* dos parâmetros, e foram desenvolvidas, podem ser citadas: métodos de particionamento, métodos de controle adaptativo e controle de aprendizagem (LENDARIS, 2009). No método de particionamento, regiões não lineares são divididas em regiões lineares e se desenvolve controladores para cada uma. No caso do controle adaptativo, especifica-se um conjunto de controladores disponíveis (via modelos parametrizados) e um algoritmo para selecionar o modelo baseado em observações. Já no controle de aprendizagem, especifica-se uma estrutura de controlador parametrizado (por exemplo de uma Rede Neural) e um algoritmo correspondente para incrementalmente ajustar os parâmetros para convergir sobre um projeto apropriado sempre que novas situações são encontradas (LENDARIS, 2009). Tais métodos de projetos de controladores, tornaram-se viáveis graças aos avanços tecnológicos, o que possibilitou o uso de inteligência computacional com um nível cada vez maior no processamento de dados.

A Inteligência Computacional é um ramo da Inteligência Artificial (IA) que se

baseia em algoritmos heurísticos (WERBOS, 2008) , tais como Lógicas *Fuzzy*, Redes Neurais e computação evolutiva (GUPTA e SINHA, 1995) (WERBOS, 2008) (MEYER *et al.*, 2009). Combina elementos de aprendizagem e adaptação pra criar programas inteligentes sendo bastante difundido em aplicações industriais.

Para o propósito de obter um controlador ótimo que minimize uma dada função de custo, sem o uso do modelo do sistema a ser controlado, uma classe de técnicas de AR, chamada de Críticos Adaptativos, foi desenvolvida pela comunidade de inteligência computacional. O conceito de Críticos Adaptativos é essencialmente a combinação das idéias de AR e PD. Enquanto a Programação Dinâmica calcula o controle por meio da Função Valor (*Value Function*) Ótimo, o Crítico Adaptativo utiliza uma aproximação dessa Função Valor Ótimo para realizar o projeto de controle. O principal atributo do Crítico Adaptativo que permite a aplicação *online* é que ele efetivamente resolve a equação de otimização de *Hamilton-Jacobi* em avanço no tempo (*forward*) enquanto a Programação Dinâmica faz a aproximação em retrocesso (*backward*) no tempo (LENDARIS, 2009) (BALAKRISHNAN *et al.*, 2008) (SI *et al.*, 2004) (LEWIS e VRABIE, 2009*a*).

Dentro deste contexto, propõe-se neste trabalho, métodos de soluções de sistemas com controle discreto ótimo, utilizando-se de técnicas de otimização através de ADP (MURRAY *et al.*, 2002) (WERBOS, 2008). O objetivo é aplicação do controle ótimo discreto, que tenha seus parâmetros ajustados de forma *online* a um sistema dinâmico MIMO (DOYLE e STEIN, 1981).

1.1 Objetivos

1. Gerais

   - Pesquisar, desenvolver algoritmos e sistematizar o conhecimento para realização do projeto de um Controlador Ótimo Discreto via Aprendizado por Reforço e Programação Dinâmica Adaptativa;

2. Específicos

   - Investigar o estado da arte para o projeto do controlador ótimo discreto, abrangendo teoria e aplicações;
   - Desenvolver algoritmos de AR para o projeto DLQR;

- Avaliar o desempenho dos Algoritmos AD e AD-HDP;
- Desenvolver um método baseado em heurística para sintonia das matrizes de ponderação do controlador ótimo.
- Analisar a convergência da ADP face a variações heurísticas das matrizes de ponderação Q e R.

1.2 Justificativa

O objetivo do controle ótimo é determinar o sinal de controle que levará o processo a satisfazer as restrições físicas e, ao mesmo tempo, minimizar (ou maximizar) algum índice de desempenho (KIRK, 1970). Tal controle pode ser obtida através da utilização do princípio do máximo de *Pontryagin* (condição necessária), ou por problemas da equação de *Hamilton-Jacobi-Bellman* (condição suficiente) (KIRK, 1970).

A teoria de otimização contribui de forma significativa para a sistematização e resolução de problemas de controle ótimo (KIRK, 1970) (KUO, 1980). A Programação Dinâmica é um método de otimização matemática de processos utilizada para decisão de multiestágios (BERTSEKAS, 1995) (BELLMAN, 1958). A condição do processo dentro de cada estágio é denominada de estado. Cada estágio inclui uma tomada de decisão que pode alterar o estado do processo representando uma transição do estado corrente e o estado futuro (BERTSEKAS, 1995) (BELLMAN, 1958). Dentro do processo multiestágios, o objetivo do tomador de decisão é encontrar uma política ótima proveniente das decisões. A determinação de uma política ótima para um processo de decisão multiestágios está embasada em uma abordagem alternativa que surgiu na década 50, chamada de princípio da otimalidade de *Bellman* (devido ao seu criador *Richard Bellman*) (BELLMAN, 1958).

O projeto do controlador ótimo discreto pode ser realizado pelo uso do princípio da otimalidade (KIRK, 1970) (BELLMAN, 1958) (BERTSEKAS, 1995) (KUO, 1980). Independente do estado inicial e do controle nos estágios iniciais, o controle deve fornecer um controle ótimo com relação ao estado resultante, a partir do controle dos estágios iniciais. De outra forma, qualquer estratégia de controle que é ótima no intervalo de $[i;N]$ é necessariamente ótima $[i+1;N]$ para $i = 0;\ 1;\ 2,...,N-1$ (KUO, 1980).

Se o sistema é modelado por uma dinâmica linear e a função de custo a ser minimizada é quadrática no estado e controle, então o controle ótimo é uma realimentação linear de estados, em que os ganhos são obtidos pela resolução da equação de *Riccati*. Por outro lado, se o sistema for modelado por dinâmicas não lineares ou a função de custo é não-linear quadrática, o controle ótimo de realimentação de estado dependerá da solução para a equação *Hamilton-Jacobi-Bellman* (HJB), que geralmente é uma equação diferencial parcial não linear ou equação a diferença (LEWIS e SYRMOS, 1995).

Entretanto, muitas vezes é computacionalmente insustentável a execução da Programação Dinâmica devido ao processo numérico da recursividade necessário para sua solução, conhecido como "maldição da dimensionalidade"(BELLMAN, 2003) (DREYFUS e LAW, 1977). Ao longo dos anos, progressos têm sido feitos para contornar a este problema (POWELL, 2007) através da construção de um sistema, chamado "crítico", para aproximação da função de custo. A idéia é aproximar a solução da Programação Dinâmica utilizando uma estrutura de função de aproximação , tal como Redes Neurais, Lógica *Fuzzy*, *Least Squares* - LS (Mínimos Quadrados), para obter-se a função de custo (SI *et al.*, 2004) (LEWIS e VRABIE, 2009*a*) (LENDARIS, 2009) (MURRAY *et al.*, 2002).

Como será discutido na dissertação, alguns dos projetos apresentados como Críticos Adaptativos não exigem o modelo de planta para ajustar a ação de controle, o Crítico, ou de ambos.

Vários esquemas de ADP aparecem na literatura. Em (BRADTKE *et al.*, 1994), foi implementado o método de Política de Iteração *Q-Learning* para LQR discreto. Werbos, (WERBOS, 1974) (WERBOS, 1989), (WERBOS, 1990) classificou ADP em quatro esquemas principais: *Heuristic Dynamic Programming*- HDP (Programação Dinâmica Heurística), *Dual* HDP - DHP (Programação Heurística Dual), *Action Dependent* - HDP (ADHDP ou *Q-Learning*) (Programação Dinâmica Heurística Dependente da Ação), *Action Dependent Dual* - HDP (ADDHP) (Programação Heurística Dual Dependente da Ação). Landelius, (LANDELIUS e KNUTSSON, 1996) aplicou as técnicas HDP,DHP, ADHDP e ADDHP para o DLQR e discutiu suas convergências mostrando que eles eram iguais a iteração pela recorrência de *Riccati*.

Em um número de aplicações industriais, a dinâmica da planta muda tanto durante a operação, que os projetos de controladores pelos métodos clássicos de controle não apresentam um desempenho satisfatório pela dificuldade de se fazer

os ajustes dos parâmetros em virtude de suas variações. O método de solução de sistemas com controle ótimo discreto, com auxílio de inteligências computacionais, através de ADP (MURRAY *et al.*, 2002) (WERBOS, 2008) vem no intuito de saciar a necessidade de ajustes dessas variações de forma mais eficiente através somente da coleta de dados do sistema dinâmico.

1.3 Motivação

A avanço no campo da inteligência computacional e das técnicas modernas de controle de sistemas no campo industrial, foi um grande estímulo para o desenvolvimento dessa pesquisa que envolve teoria de controle ótimo discreto. Sistemas dinâmicos industriais, operam de maneira contínua, porém seus controladores operam de maneira amostrada em visto dos equipamentos adotados. Sendo assim, apesar da dinâmica continua, o projeto de controle utilizando-se ADP é discreto.

Os algoritmos de ADP, tem sido de grande aplicabilidade industrial, como por exemplo sistemas de mísseis, autopiloto, sistemas de potências, sistemas de comunicação, processos bioquímicos, etc. Na ótica de controladores ótimos, sistemas como esses, necessitam dos ajustes dos seus respectivos controladores de forma mais eficiente tendo em vista que as técnicas clássicas não são suficientes.

1.4 Contribuições

Dentre as principais contribuições dessa dissertação pode-se destacar:

1. Caracterização do problema de controle, como alocação de uma autoestrutura, que garanta a estabilidade do sistema no mapeamento no plano Z.

2. Estudo e sistematização de ADP para sistemas MIMO utilizando-se os algoritmos HDP e ADHDP.

3. Projeto de um controlador ótimo adaptativo *online*, que utiliza AR para resolver o problema do controle ótimo discreto.

4. Contribuições a cerca de formulações e análises matemáticas no desenvolvimento da ADP para o controlador ótimo discreto.

1.5 Organização da dissertação

A dissertação é organizada em Capítulos que descrevem o desenvolvimento metodológico para avaliação da convergência da solução DLQR e para as matrizes de sintonia.

No Capítulo 2, apresenta-se o controle ótimo discreto por Programação Dinâmica, uma proposta heurística para seleção das matrizes de sintonia Q e R do DLQR e uma abordagem introdutória sobre os aspectos essenciais de Aprendizagem por Reforço. O Capítulo 3, aborda métodos de soluções *offline* de sistemas de controle discreto através de controle ótimo utilizando-se como base a teoria de AR. Sugere-se a solução por Política de Iteração (PI) e Valor de Iteração (VI). No Capítulo 4 será visto como formular estes procedimentos em um método de AR *online* em tempo real usando-se dados medidos do sistema ao longo de sua trajetória. Estes métodos são amplamente chamados de ADP. São apresentados as estruturas básicas de ADP. É feita análise sobre cada uma delas. Soluções do DLQR por críticos adaptativos são apresentadas no Capitulo 5. Dois esquemas são apresentados, HDP e AD-HDP. A influência do fator de desconto e do ruído na ação de controle na equação de HJB serão analisados. No Capítulo 6, os algoritmos apresentados nos capítulos anteriores são avaliados por experimentos computacionais. A análise de convergência é feita baseada no número de iterações e no método heurístico de seleção das matrizes Q e R. No Capítulo 7, apresenta-se as conclusões, comentários e propostas futuras. Por fim, os Apêndices são direcionados para complementar a fundamentação da abordagem considerada. No Apêndice A tem-se a formulação do controle ótimo por cálculo variacional. O Apêndice B mostra a convergência dos algoritmos de HDP e AD-HDP para o DLQR.

1.6 Artigos publicados e aceitos

1.6.1 Artigos Publicados

1. Andrade, Gustavo A.; Neto, João Viana F.; Lopes, Leandro Rocha; , "A Framework for Modeling, Digital Control Design and Simulations of Dynamic", Computer Modelling and Simulation (UKSim), 2010 12th International Conference on , vol., no., pp.398-403, 24-26 March 2010.

2. Neto, João Viana F.; Sá, Denis Fabrício Sousa de; Lopes, Leandro Rocha; , "State Space Modeling of Thermal Actuators Based on Peltier Cells for Indirect Measurements and Optimal Control", Computer Modelling and Simulation (UKSim), 2010 12th International Conference on , vol., no., pp.392-397, 24-26 March 2010.

1.6.2 Artigo Aceito

1. Fonseca Neto, João and Leandro Rocha Lopes (n.d.). "On the Convergence of DLQR Control and Recurrences of *Riccati* and *Lyapunov* in Dynamic Programming". In:UKSim 13th International Conference on Computer Modelling and Simulation(UKSim2011). Cambridge, United Kingdom.

Capítulo 2

FUNDAMENTOS E ESTADO DA ARTE

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

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

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

Este capítulo tem enfoque na caracterização do controlador ótimo discreto por princípio de otimalidade e Programação Dinâmica partindo-se da discretização e, por fim, uma abordagem introdutória sobre AR e os aspectos essenciais como o processo de decisão *Markoviano*, Função Valor (*Value Function*) e uma classe de AR chamada de Ator-Crítico.

2.1 Projeto do LQR discreto

O problema do regulador é definido com referência a um sistema de entrada nula com o objetivo de guiar (orientar) os estados e saídas na vizinhança do estado de equilíbrio. As condições de entradas zero não são severas para o projeto. O regulador linear quadrático (tempo infinito) garante a estabilidade do sistema com algumas características de amortecimento e desempenho satisfatório quando

as entradas são diferentes de zero (KUO, 1980) (ATHANS e FALB, 1966) (LEWIS e SYRMOS, 1995).

2.1.1 Formulação do índice de desempenho quadrático para um sistema de dados amostrados

Considere o sistema linear em sua forma contínua dada por:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.1}$$

sendo $x(t)$ o vetor de estado $n \times 1$ e $u(t)$ o vetor de controle $p \times 1$.

O vetor de controle discreto é dada por:

$$u(t) = u(kt) \tag{2.2}$$

$$kT \leq t \leq (k+1)T \tag{2.3}$$

O desafio é encontrar o controle ótimo $u^*(kT)$, sendo k=0,1,2,...,$N-1$, para que o índice de desempenho dado por:

$$J = \frac{1}{2}x(tf)^T Sx(tf) + \frac{1}{2}\int_0^{tf} \left[x(t)^T Qx(t) + u(t)^T Ru(t)\right] \tag{2.4}$$

seja minimizado, para um horizonte de tempo $tf = NT$ dado. Neste caso, as matrizes Q e R do índice de desempenho são definidas positiva e semi-definida positivas respectivamente, T é o período de amostragem e N número de amostras.

Então a equação de estado discreta fica:

$$x[(k+1)T] = A_d x(kT) + B_d u(kT) \tag{2.5}$$

sendo:

$$A_d = e^{AT} \tag{2.6}$$

$$B_d = \int_0^T A_d(T-\tau)Bd\tau \tag{2.7}$$

O índice de desempenho discretizado em termos dos operadores fica:

$$\begin{aligned} J_{DLQR} &= \frac{1}{2}x(NT)^T Sx(NT) + \frac{1}{2}\sum_{k=0}^{N-1}[x(kT)^T \hat{Q}x(kT) \\ &+ 2x(kT)^T M(T)u(kT) + u(kT)^T \hat{R}u(kT)] \end{aligned} \tag{2.8}$$

sendo:

$$\hat{Q}(T) = \int_{kT}^{(k+1)T} A_d^T(t-kT)QA_d(t-kT)dt \tag{2.9}$$

$$M(T) = \int_{kT}^{(k+1)T} A_d^T(t-kT)QB_d(t-kT)dt \tag{2.10}$$

$$\hat{R}\ (T) = \int_{kT}^{(k+1)T} [B_d^T(t-kT)QB_d(t-kT) + R]dt \tag{2.11}$$

Tem-se como ponto de vista de propriedades, que $\hat{Q}(T)$ e $\hat{R}(T)$ são simétricas, semi-definidas positiva e definida positiva respectivamente e nada pode ser dito a respeito de $M(T)$. Em geral, o índice de desempenho quadrático é descrito sem a matriz M. É mais natural definir o índice de desempenho quadrático da seguinte forma:

$$J_{DLQR} = \frac{1}{2}x_N^T Sx_N + \frac{1}{2}\sum_{k=0}^{N-1}(x_k^T Qx_k + u_k^T Ru_k) \tag{2.12}$$

2.1.2 Problema de tempo finito

O problema do projeto do regulador linear discreto deve ser resolvido pelo uso do princípio do mínimo discreto.

O processo de controle digital é descrito por:

$$x_{k+1} = A_d x_k + B_d u_k \tag{2.13}$$

com x_0 dado. O objetivo é encontrar u_k^* para que o índice

$$J_N = \frac{1}{2}x_N^T S x_N + \frac{1}{2}\sum_{k=0}^{N-1}(x_k^T \hat{Q} x_k + x_k^T 2M u_k + u_k^T \hat{R} u_k) \tag{2.14}$$

seja minimizado.

O índice de desempenho J_N na Eq.(2.14) é para o intervalo de tempo de $[0, N]$, ou para um total de N estágios se o tempo não está envolvido com a variável independente. Define-se $J_{N-j}(x_j)$ como o índice de desempenho no intervalo de j a N, ou o ultimo estágio $N - j$, isto é:

$$J_{N-j}(x_j) = \frac{1}{2}x_N^T S x_N + \frac{1}{2}\sum_{k=j}^{N-1}(x_k^T \hat{Q} x_k + x_k^T 2M u_k + u_k^T \hat{R} u_k) \tag{2.15}$$

Quando $j = 0$,

$$J_{N-j}(x_0) = J_N \tag{2.16}$$

Quando $j = N$,

$$J_0(x_N) = \frac{1}{2}x_N^T S x_N \tag{2.17}$$

que é o índice de desempenho através do último estágio somente. Portanto, o valor do desempenho J pode ser dado por:

$$\min_{u_j} J_{N-j}(x_j) = \frac{1}{2}x_j^T P_j x_j \tag{2.18}$$

Sendo P_j a solução da DARE (*Discrete Algebraic Riccati Equation*).

Para o caso do regulador linear quadrático de tempo finito, não tem-se como exigência que o sistema seja controlável, observável ou até mesmo estável. O índice de desempenho pode ser finito para um finito N mesmo que os estados sejam não controláveis ou não estáveis.

2.1.3 Problema de tempo infinito

Para o caso de tempo infinito ou número infinito de estágios, $N = \infty$, o índice de desempenho é dado por:

$$J = \frac{1}{2}\sum_{k=0}^{\infty}(x_k^T \hat{Q} x_k + x_k^T 2M u_k + u_k \hat{R} u_k) \tag{2.19}$$

Neste caso o custo final é eliminado, desde que N se aproxima do infinito, o estado final x_N deve aproximar do estado de equilíbrio nulo, de modo que a restrição terminal não é mais necessária.

Para o projeto do regulador linear quadrático de tempo infinito, uma exigência é que, em malha fechada, o sistema seja assintoticamente estável. Para isso, o sistema dado pela Eq.(2.13) precisa ser controlável e observável.

A solução do regulador infinito pode ser obtido definindo-se $k \to \infty$. Como $N \to \infty$, o ganho da matriz P_k de *Riccati* torna-se constante.

$$\lim_{k\to\infty} P_k = P \tag{2.20}$$

O controle ótimo é:

$$u_k^* = -(\hat{R} + B_d^T P B_d)^{-1}(B_d^T P A_d + M^T)x_k^* \tag{2.21}$$

Então a matriz de realimentação, é uma matriz constante.

$$K = (\hat{R} + B_d^T P B_d)^{-1}(B_d^T P A_d + M^T) \tag{2.22}$$

O índice de desempenho ótimo para $N = \infty$ através da Eq.(2.18):

$$J_\infty^* = \frac{1}{2} x_0^T P x_0 \tag{2.23}$$

Para o projeto do LQR de tempo infinito, é necessário que o sistema seja controlável ou estável na realimentação de estados. Controlabilidade é uma exigência mais forte que a estabilidade, já que um sistema não controlável pode ser estabilizado se os estados não controláveis sejam estáveis (KUO, 1980).

2.2 Princípio da otimalidade e Programação Dinâmica

O projeto DLQR realizado nas seções anteriores podem ser afetados pelo uso do princípio da otimalidade. O projeto, usando o princípio da otimalidade, é conhecido como o método de Programação Dinâmica.

Independente do estado inicial e do controle nos estágios iniciais, deve-se fornecer um controle ótimo com relação ao estado resultante, a partir do controle

dos estágios iniciais. De outra forma, qualquer estratégia de controle que é ótima no intervalo de $[j, N]$ é necessariamente ótima $[j+1, N]$ para $j = 0, 1, 2, ..., N-1$.

O regulador discreto ótimo de tempo finito é dado por:

$$\min J = G(x_N) + \sum_{k=0}^{N-1} F(x_k, u_k) \tag{2.24}$$

sujeito a:

$$x_{k+1} = A_d x_k + B_d u_k \tag{2.25}$$

sendo:

$$G(x_N) = \frac{1}{2} x_N^T S x_N \tag{2.26}$$

$$\sum_{k=0}^{N-1} F(x_k, u_k) = \frac{1}{2} x_N^T \hat{Q} x_N + x_k^T M u_k + \frac{1}{2} u_k^T \hat{R} u_k \tag{2.27}$$

com x_0 dado.

Assumindo $J_{N-j}(x_j)$ seja o índice de desempenho no intervalo de $[j, N]$. Então:

$$J_{N-j}(x_j) = G(x_N) + \sum_{k=j}^{N-1} F_k(x_k, u_k) \tag{2.28}$$

$$j = 0, 1, 2, ..., N$$

O mínimo valor de $J_{N-j}(x_j)$ é representado por:

$$f_{N-j}(x_j) = \min_{u_i} J_{N-j}(x_j) \tag{2.29}$$

Para $j = N$, a Eq.(2.29) representa o índice de desempenho ou retorno sobre o estágio 0. Portanto:

$$f_0(x_N) = G(x_N) = \frac{1}{2} x_N^T S x_N \tag{2.30}$$

Para $j = N-1$, tem-se um estágio ou um intervalo processado que é o último estágio. Então, o índice de desempenho ótimo é dado por:

$$\begin{aligned} f_1(x_{N-1}) &= \min_{u_{N-1}} J_1(x_{N-1}) \\ &= \min_{u_{N-1}} G(x_N) + F_{N-1}(x_{N-1}, u_{N-1}) \end{aligned} \tag{2.31}$$

sabendo-se que:

$$G(x_N) = \frac{1}{2}(A_d x_{N-1} + B_d u_{N-1})^T S(A_d x_{N-1} + B_d u_{N-1}) \tag{2.32}$$

Rearranjando a Eq.(2.31) tem-se:

$$\begin{aligned} f_1(x_{N-1}) &= \min_{u_{N-1}} (\frac{1}{2} x_{N-1}^T (\hat{Q} + A_d^T S A_d) x_{N-1} \\ &+ x_{N-1}^T (M + \frac{1}{2} A_d^T S B_d) u_{N-1} \\ &+ \frac{1}{2} u_{N-1}^T B_d^T S A_d x_{N-1} \\ &+ \frac{1}{2} u_{N-1}^T (\hat{R} + B_d^T S B_d) u_{N-1}) \\ &= \min_{u_{N-1}} J_1(x_{N-1}) \end{aligned} \tag{2.33}$$

para o mínimo $J_1[x_{N-1}]$ tem-se:

$$\frac{\partial J_1(x_{N-1})}{\partial u_{N-1}} = 0 \tag{2.34}$$

o resultado é dado por:

$$[(M + \frac{1}{2} A_d^T S B_d)^T + \frac{1}{2} B_d^T S A_d] x_{N-1}^* + (\hat{R} + B_d^T S B_d) u_{N-1}^* = 0 \tag{2.35}$$

o controle ótimo é:

$$u_{N-1}^* = -(\hat{R} + B_d^T S B_d)^{-1} (M^T + B_d^T S A_d) x_{N-1}^* \tag{2.36}$$

Substituindo-se a Eq.(2.36) na Eq.(2.33), tem-se, depois da simplificação:

$$\begin{aligned} f_1(x_{N-1}) &= \frac{1}{2} x_{N-1}^T [\hat{Q} + A_d^T S A_d \\ &- (M^T + B_d^T S A_d)^T (\hat{R} + B_d^T S B_d)^{-1} (M^T + B_d^T S A_d)] x_{N-1} \end{aligned} \tag{2.37}$$

Definições:

$$P_N = S \tag{2.38}$$

$$P_{N-1} = \hat{Q} + A_d^T S A_d - (M^T + B_d^T S A_d)^T (\hat{R} + B_d^T S B_d)^{-1} (M^T + B_d^T S A_d) \tag{2.39}$$

Portanto, tem-se respectivamente:

$$f_0(x_N) = \frac{1}{2} x_N^T P_N x_N \tag{2.40}$$

$$f_1(x_{N-1}) = \frac{1}{2} x_{N-1}^T P_{N-1} x_{N-1} \tag{2.41}$$

Continuando o processo, assume-se $j = N - 2$. Isto é um problema de otimização consistindo nos dois últimos estágios.

$$\begin{aligned} f_2(x_{N-2}) &= \min_{u_{N-2}, u_{N-1}} J_2(x_{N-2}) \\ &= \min_{u_{N-2}, u_{N-1}} F_{N-2}(x_{N-2}, u_{N-2}) + F_{N-1}(x_{N-1}, u_{N-1}) \\ &+ G(x_N) \end{aligned} \tag{2.42}$$

sabendo-se que :

$$f_2(x_{N-2}) = \min_{u_{N-2}} (F_{N-2}(x_{N-2}, u_{N-2}) + f_1(x_{N-1}) \tag{2.43}$$

$$\begin{aligned} F_{N-2}(x_{N-2}, u_{N-2}) &= \frac{1}{2} x_{N-2}^T \hat{Q} x_{N-2} + x_{N-2}^T M u_{N-2} \\ &+ \frac{1}{2} u_{N-2}^T \hat{R} u_{N-2} \end{aligned} \tag{2.44}$$

e

$$f_1(x_{N-1}) = \frac{1}{2} (A_d x_{N-2} + B_d u_{N-2})^T P (A_d x_{N-2} + B_d u_{N-2}) \tag{2.45}$$

Ajustando:

$$\frac{\partial J_2(x_{N-2})}{\partial u_{N-2}} = 0 \tag{2.46}$$

Pode-se mostrar que o controle ótimo é dado por:

$$u^*_{N-2} = -[\hat{R} + B_d^T P_{N-1} B_d]^{-1}(M^T + B_d^T P_{N-1} A_d) x^*_{N-2} \tag{2.47}$$

e

$$\begin{aligned} f_2(x_{N-2}) = \frac{1}{2} x^T_{N-2}[\hat{Q} + A_d^T P_{N-1} A_d \\ -(M^T + B_d^T P_{N-2} A_d)^T(\hat{R} + B_d^T P_{N-2} B_d)^{-1} \\ (M^T + B_d^T P_{N-1} A_d)] x_{N-2} \end{aligned} \tag{2.48}$$

Assumindo-se:

$$\begin{aligned} P_{N-2} = \hat{Q} + A_d^T P_{N-1} A_d \\ -(M^T + B_d^T P_{N-1} A_d)^T(\hat{R} + B_d^T P_{N-1} B_d)^{-1}(M^T + B_d^T P_{N-1} A_d) \end{aligned} \tag{2.49}$$

A Eq.(2.48) é simplificada para:

$$f_1(x_{N-2}) = \frac{1}{2} x^T_{N-2} P_{N-2} x_{N-2} \tag{2.50}$$

Pode-se mostrar que em geral:

$$f_{N-j}(x_j) = \frac{1}{2} x_i^T P_j x_j \tag{2.51}$$

sendo:

$$\begin{aligned} P_j = \hat{Q} + A_d^T P_{j+1} A_d \\ -(M^T + B_d^T P_{j+1} A_d)^T(\hat{R} + B_d^T P_{j+1} B_d)^{-1}(M^T + B_d^T P_{j+1} A_d) \end{aligned} \tag{2.52}$$

O controle ótimo é:

$$u^*_j = -(\hat{R} + B_d^T P_{j+1} B_d)^{-1}[M^T + B_d^T P_{j+1} A_d] x^*_j \tag{2.53}$$

Assim, tem-se a equação de *Riccati* usando-se o princípio da otimalidade. O método de solução também é conhecido como Programação Dinâmica.

Para fins de simplicação das equações, será adotado a partir desse momento que $\hat{Q} = Q$, $\hat{R} = R$ e $M = 0$.

2.2.1 Solução recursiva da DARE

A equação a diferença de *Riccati* é geralmente resolvida por método de cálculo numérico computacional, método recursivo ou método dos autovalores e autovetores. O cálculo numérico computacional envolve soluções iterativas da equação a diferença não linear de *Riccati*. Como já discutido, a Programação Dinâmica é um método de solução recursivo do problema de controle ótimo. Assumindo-se a matriz de ganho de realimentação do regulador discreto ótimo como:

$$K_j = (R + B_d^T P_{j+1} B_d)^{-1} (B_d^T P_{j+1} A_d) \tag{2.54}$$

O controle ótimo da Eq.(2.53) é escrito como:

$$u_j^* = -K_j x_j^* \tag{2.55}$$

A Eq.(2.52) de *Riccati* é simplificada:

$$P_j = Q + A_d^T P_{j+1} A_d - (B_d^T P_{j+1} A_d)^T K_j \tag{2.56}$$

Começando-se com a condição limite $P_N = S$, as Eq.(2.54) e (2.56) são resolvidas por recursividade (BRYSON e HO, 1975) (LANCASTER e RODMAN, 1995).

2.2.2 Algoritmo de Programação Dinâmica para o LQR discreto

Com base em toda teoria exposta anteriormente, pode-se agora desenvolver um algoritmo computacional que mostre a resolução do problema do DLQR utilizando-se a Programação Dinâmica. Conforme a Eq.(2.57) do DLQR de horizonte finito exposta logo a seguir, tem-se o seguinte algoritmo:

$$J_{DLQR} = \frac{1}{2} \left\langle x_N^T S x_N \right\rangle + \frac{1}{2} \sum_{k=i}^{N-1} [x_k^T Q x_k + u_k^T R u_k] \tag{2.57}$$

ALGORITMO 1($PD - DLQR - HorizonteFinito$)

1 ▷ **Inicialização**
2 Sistema Dinâmico Discreto
3 A_d, B_d.
4 Condição Limite
5 P_N=S
6 Matrizes de Ponderação
7 Q e R.
8 Estados Iniciais
9 x_0
10 ▷ **Processo Iterativo**.
11 **for** $j \leftarrow N : -1 : 1$
12 **do**
13 ▷ **Ganho ótimo de realimentação**.
14 $K_j = (B_d^T S_N B_d + R)^{-1} B_d^T P_{j+1} A_d$
15 **for** $j \leftarrow N - 1 : -1 : 1$
16 **do**
17 ▷ **Recorrência de *Riccati***.
18 $P_j = (A_d - B_d K_j)^T P_{j+1}(A_d - B_d K_j) + K_j^T R K_j + Q$
19 ▷ **Controle Ótimo** u_j.
20 **for** $j \leftarrow 1 : 1 : N - 1$
21 **do**
22 $x_{j+1} = (A_d - B_d K_j) x_j$
23 **for** $j \leftarrow 1 : 1 : N$
24 **do** $u_j^* = -K_j x_j$
25 **Fim do Processo Iterativo**.

O algoritmo 1, expõe de maneira clara, os passos a serem seguidos para obter-se a solução do DLQR utilizando-se a Programação Dinâmica. Vale ressaltar a inicialização do processo. Destaca-se nesse trecho do algoritmo a condição limite, que nada mais é que o valor no instante de tempo final N assumido pela matriz P de *Riccati*, os estados inciais, e é claro, as matrizes de ponderação Q e R.

2.3 Seleção das matrizes de ponderações QR do índice de desempenho

As matrizes de ponderações do índice de desempenho são escolhidas pela característica de desempenho de controle. A dificuldade de escolha se deve ao fato de não existir um método sistemático para tal seleção, sendo normalmente adotada a forma diagonal para essas matrizes e tendo sua escolha realizada por meio de várias simulações, tentativa e erro. Os valores selecionados para as matrizes de ponderação, devem satisfazer os critérios estabelecidos pelo projetista, que por consequência, influenciam na determinação do ganho do controlador.

Algumas metodologias podem ser citadas. No trabalho de (JOHNSON e GRIMBLE, 1987), as técnicas disponíveis para seleção destas matrizes, são divididas em quatro metodologias: métodos heurísticos, controle ótimo modal, método do lugar das raízes ótimo assintótico e método da ponderação dinâmica.

A primeira técnica projetada para seleção das matrizes de ponderação do custo funcional quadrático foram as heurísticas.

2.3.1 Método de Sintonia Heurística QR

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

$$u_k(QR) = -K_{QR}x_k \tag{2.58}$$

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

O método de Sintonia Heurística QR (MSH-QR) (FONSECA NETO e LOPES 2011) baseia nas relações entre as matrizes Q e R quando a recorrência de *Riccati* que é dada por:

$$P = Q + A_d^T PA - A_d^T PB_d K_{ric} \tag{2.59}$$

é substituída na equação do ganho que é dada por:

$$K_{ric} = (R + B_d^T PB_d)^{-1} B_d^T PA_d \tag{2.60}$$

Desta forma, tem-se que o ganho ótimo da Eq.(2.60) é dado por:

$$K_{ric} = (R + B_d^T PB_d)^{-1} B_d^T (Q + A_d^T PA_d - A_d^T PB_d K_{ric}) A_d \tag{2.61}$$

Operando-se com a Eq.(2.61) no intuito de explicitar as relações para Q e R tem-se que:

$$\begin{aligned} K_{ric}(Q,R) = &\left\{ \left[(R + B_d^T PB_d)^{-1} B_d^T (Q + A_d^T PA_d) \right] \right. \\ &\left. - \left[(R + B_d^T PB_d)^{-1} B_d^T (A_d^T PB_d K_{ric}) \right] \right\} A_d \end{aligned} \tag{2.62}$$

A forma explicita final é dada por:

$$K_{ric}(Q,R) = K_{f1}(QR) + K_{f2}(R) \tag{2.63}$$

sendo:

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

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

$$K_{f1}(Q,R) \approx \left\{\left[R^{-1} B_d^T Q\right] A_d\right\} \tag{2.64}$$

$$K_{f2}(R) \approx -\left\{\left[R^{-1} B_d^T A_d^T P B_d R^{-1} B_d^T P A_d\right] A_d\right\} \tag{2.65}$$

Observa-se que se $R >> 0$ para o caso de matrizes diagonais, tem-se que $f_2(Q,R) \approx 0$ (FONSECA NETO e LOPES 2011).

2.4 Aprendizagem por Reforço (AR)

Quando se deseja que o agente tenha autonomia total, significa que este deverá ser capaz de aprender com base em informações do tipo recompensas ou reforços fornecidos por um "crítico"ou pelo próprio ambiente. Um sistema típico de Aprendizagem por Reforço (AR) constitui-se basicamente de um agente interagindo em um ambiente via sensores (percepção) e atuadores (ação). A ação, u_k, tomada muda de alguma forma o ambiente, afetando o estado na tentativa de alcançar o seu objetivo, e as mudanças são comunicadas ao agente através de um sinal de reforço, r_k, e o próximo estado, x_{k+1}, como mostra a Figura 2.1 (SI *et al.*, 2004).

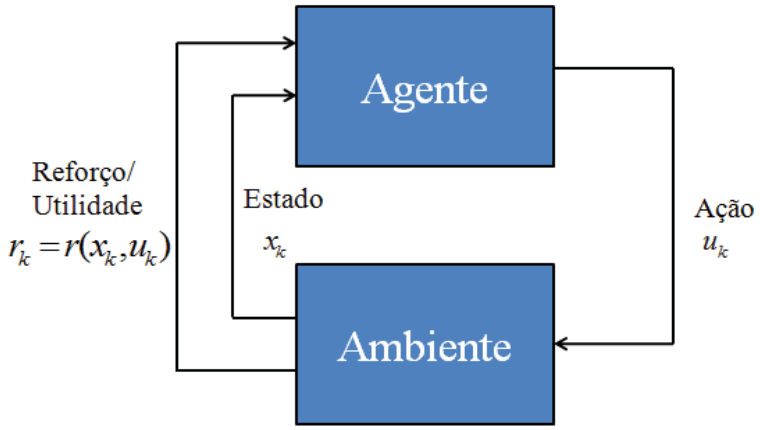

Figura 2.1: Um agente AR interagindo com seu ambiente.

Aprendizado por Reforço difere de Aprendizado Supervisionado principalmente no tipo de resposta recebida do meio ambiente. No Aprendizado Supervisionado, o equivalente a uma função "especialista"está disponível e conhece a saída correta, a *priori*, para cada uma das saídas do agente, e a aprendizagem é baseada em dados de erro da saída (SI *et al.*, 2004). Na AR, por outro lado, não há o conhecimento da saída correta, o agente recebe apenas a informação do ambiente por um reforço, e aplica uma ação, de maneira aumentar a quantidade de recompensa que ele recebe ao longo do tempo. Outra diferença de Aprendizado Supervisionado é que, no desempenho *online*, a avaliação do sistema é concomitante com a aprendizagem. AR pode ser aplicada quando Aprendizado Supervisionado padrão não é aplicável, e exige menos conhecimento a *priori* (SI *et al.*, 2004).

A cada instante de tempo k, o sistema de AR tem um número de ações de saída formando um vetor $\overrightarrow{u}_k$ chamado de "vetor de ação"ou "controle", em que o agente usa para influenciar o ambiente (SI *et al.*, 2004). O Problema se baseia em encontrar uma política de controle $\overrightarrow{u}_k$, que maximize o reforço/ganho acumulado no tempo, dado por:

$$max \left\langle \sum_{i=k}^{\infty} \gamma^{i-k} r_i \right\rangle \tag{2.66}$$

ou

$$max \left\langle \sum_{i=0}^{\infty} \gamma^{i} r_{k+i} \right\rangle \tag{2.67}$$

sendo r o reforço (utilidade) e γ o *fator de desconto*.

A política h é um mapeamento de estado para ação a um valor $h(x_k, u_k)$, ou seja, a probabilidade do agente tomar a ação u_k quando este se encontrar no estado x_k (SI *et al.*, 2004).

O comportamento do agente apresenta variações à medida que ele vai acumulando experiência a partir das interações com o ambiente. O aprendizado pode ser expresso em termos da convergência ate uma política ótima, h^*, que conduza a solução do problema de forma ótima (SI *et al.*, 2004) (LEWIS e VRABIE, 2009*a*). Vale ressaltar que as ações ótimas podem ser baseados em mínimo de combustível, energia mínima, risco mínimo, a recompensa máxima, e assim por diante.

O reforço é um sinal do tipo escalar, r_k, devolvido pelo ambiente ao agente. O reforço é emitido assim que uma ação tenha sido efetuada e uma transição de estado, $x_k \rightarrow x_{k+1}$, tenha ocorrido. As funções de reforço expressam o objetivo que o agente deve alcançar. O agente deve maximizar a quantidade total de reforços recebidos, chamado de retorno acumulado. No caso mais simples, modelo de horizonte finito, o retorno é dado por:

$$R_k = \left\langle \sum_{i=k}^{N} r_i \right\rangle \tag{2.68}$$

sendo N o número de amostras no horizonte de tempo discreto.

Para o caso em que deseja-se utilizar-se um *desconto de retorno*, modelo de horizonte infinito, tem-se que:

$$R_k = \left\langle \sum_{i=k}^{\infty} \gamma^{i-k} r_i \right\rangle \tag{2.69}$$

O *fator de desconto*, γ, $0 \leq \gamma \leq 1$, determina o grau de influência que têm os valores futuros sobre o reforço total (BUŞONIU *et al.*, 2010).

Se $\gamma = 0$ o agente tem uma visão míope dos reforços, maximizando apenas os reforços imediatos. Se $\gamma = 1$ a visão do reforço abrange todos os estados futuros dando a mesma importância para ganhos neste momento e qualquer ganho futuro.

Um conflito básico que surge na Aprendizagem por Reforço é entre a exploração (*Exploration*) e a descoberta (*Exploitation*). O dilema entre exploração e descoberta, pode ser caracterizado de tal maneira, que em cada estado, o agente deve escolher entre:

- uma ação para o qual o reforço esperado é de boa qualidade.
- ou uma ação em que a qualidade pode ser, neste instante preciso da escolha, não tão boa, porém as aplicações podem direcionar pare zonas promissoras, mas não exploradas.

Ou seja, o agente deve escolher, por vezes, ações que ele acredita ser de qualidade inferior, a fim de descobrir se elas poderiam realmente ser boas.

2.4.1 Processo de Decisão *Markoviano*

Formalmente, um agente de AR é caracterizado como um Processo de Decisão *Markoviano* (PDM), em que as transições entre estados são probabilísticas. Cada ação tem uma recompensa/custo, que depende do estado em que o processo se encontra. O PDM tem quatro componentes: estados, ações e distribuições de transição e recompensa (BUŞONIU *et al.*, 2010). Um PDM determinístico é definido pelo espaço de estados X do processo, o espaço de ação U do controlador, a função de transição f do processo (que descreve como os estados mudam com as ações de controle), e a função de reforço r (que avalia o desempenho do controle imediato).

Então tem-se a função de transição $f : X \times U \rightarrow X$

$$x_{k+1} = f(x_k, u_k) \tag{2.70}$$

Ao mesmo tempo, o controlador recebe um sinal de reforço $r_k : X \times U \rightarrow \Re$

$$r_k = r\left(x_k, u_k\right) \tag{2.71}$$

Considerando-se um processo estocástico, X, em que um número finito de valores pode ser, X=$\{x_o, x_1, ..., x_k\}$.

Em um Processo de Decisão *Markoviano*, as transições do estado x ao estado y depende apenas das ações permitidas no estado x. As Probabilidades de transição são denotadas então por (GLORENNEC, 2000):

$$p_{xy}(u) = P_r(\, x_{k+1} = x_y |\, x_k = x_x, u_k = u) \tag{2.72}$$

Sendo Pr o operador de probabilidade do estado x_k passar para o estado x_{k+1}, quando a ação u_k for tomada em k.

Pode-se caracterizar o PDM como:

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

2.4.2 Função Valor (*Value Function*)

A Função valor é o mapeamento do estado, ou par estado-ação, em um valor que é obtido a partir do reforço atual e dos reforços futuros. A função valor que considera so o estado x_k é denotada por $V(x_k)$ e denominada *função valor-estado*. A função valor que considera o par estado-ação (x_k, u_k) é denotada por $Q(x_k, u_k)$, e denominada *função valor-ação*(Q - *Function*).

Em todo estado x_k, um agente escolhe uma ação de acordo com uma determinada política, $h(x_k) = u_k$. O valor do estado x_k sobre uma política $h(x_k)$ é dado por:

$$V_h(x_k) = E\left\{\left\langle \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h(x_i)) \,|x_0 = x \right\rangle\right\} \forall x \in X \tag{2.73}$$

em que $E\{r(x_i, h(x_i))\}$ representa a experança de reforço quando se aplica a ação proposta pela política h no estado x_i. Adotando-se $E\{r(x_k, h(x_k))\} = R(x_k, u_k)$, a Eq.(2.73) fica:

$$V_h(x_k) = R(x_k, h(x_k)) + \gamma \sum_{y \in X} p_{xy}(h(x_k)) V_h(y) \tag{2.74}$$

Sabe-se que existe uma política ótima, $h^*(x_k)$ que define:

$$V_h^*(x_k) \geq V_h(x_k) \forall x \in X, \forall h \tag{2.75}$$

O valor ótimo é:

$$V_h^*(x_k) = \max_h E\left\{\left\langle \sum_{i=k}^{\infty} \gamma^{i-k} r_i \right\rangle\right\} \tag{2.76}$$

$V_h^*(x_k)$ satisfaz, no caso do horizonte infinito, a seguinte equação, conhecida como equação de otimalidade de *Bellman* (ou equação da Programação Dinâmica).

$$V_h^*(x_k) = \max_{u \in U_x} \left\{ R(x_k, u_k) + \gamma \sum_y p_{xy}(u_k) V_h^*(y) \right\}, \forall x \in X \tag{2.77}$$

A partir da Eq.(2.77), tem-se que a política ótima $h^*(x_k)$ é dada por:

$$h^*(x_k) = \arg\max_{u \in U_x} \left\{ R(x_k, u_k) + \gamma \sum_y p_{xy}(u_k) V_h^*(y) \right\} \tag{2.78}$$

Programação Dinâmica oferece um conjunto de métodos para resolver o problema de otimização, aproveitando a propriedade de *Markov*. Os dois principais métodos são chamados de Política de Iteração (PI) e Valor de Iteração (VI).

2.4.3 Aprendizagem por Ator-Crítico

AR está fortemente ligado pelo ponto de vista teórico de controle adaptativo direto e indireto. Uma classe de AR baseia-se na estrutura do Ator-Crítico (SUTTON e BARTO 1998), onde um componente Ator aplica uma política de ação ou controle para o ambiente, e um componente Crítico avalia o valor dessa ação. Com base nesta avaliação do valor, vários esquemas podem ser usados para modificar ou melhorar a ação no sentido de que a nova política possui um valor que é melhor em relação ao valor anterior.

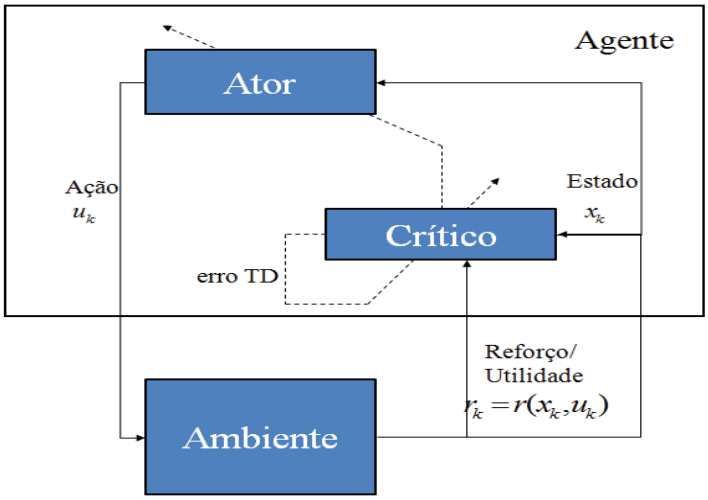

Figura 2.2: Estrutura de Aprendizado por Reforço com Ator/Crítico.

A estrutura Ator-Crítico, conforme a Figura 2.2, implica em duas etapas: política de avaliação pelo Crítico seguida de política melhoria. A etapa de políticas de avaliação é feita através da observação do ambiente os resultados e das ações em curso. Portanto, é de maior interesse, sistemas de Aprendizagem por Reforço que utilizam como estrutura Ator-Crítico, em que o Crítico avalia o valor das atuais políticas com base em algum tipo de critério de otimalidade (LEWIS e VRABIE, 2009*a*). Os Métodos Ator-Crítico são uma forma de aprendizagem Temporal-Diferencial (*Temporal Difference - TD*) que possui uma estrutura de memória separada para representar a política de forma independente da função de valor. Na aprendizagem TD, o conhecimento prévio, assim como, a experiência do agente, é utilizado para determinar qual decisão ele deve tomar. A atualização da função de valor pode ocorrer a cada instante de tempo, não necessitando de uma estimativa confiável da função de retorno. O algoritmo utilizado para atualizar $V(x_k)$ num dado instante k a uma ação, $h(x_k) = u_k$ que faz passar do estado x_k para x_{k+1} para um reforço imediato $r_k = (x_k, u_k)$ é dada por:

$$V_h(x_k) \leftarrow V_h(x_k) + \epsilon \left[r_k + \gamma V_h(x_{k+1}) - V_h(x_k) \right] \tag{2.79}$$

sendo ϵ a *taxa de aprendizagem*. Para mostrar que esta expressão é uma estimativa de $V_h(x_k)$, expandindo-se a Eq.(2.73) tem-se que:

$$\begin{aligned}
V_h(x_k) &= E\left\{ \left\langle \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h(x_i)) \,|x_0 = x \right\rangle \right\} \forall x \in X \\
V_h(x_k) &= E\left\{ r(x_k, h(x_k)) + \left\langle \sum_{i=k+1}^{\infty} \gamma^{i-k} r(x_i, h(x_i)) \,|x_0 = x \right\rangle \right\} \\
V_h(x_k) &= E\left\{ r(x_k, h(x_k)) + \gamma \left\langle \sum_{i=k+1}^{\infty} \gamma^{i-(k+1)} r(x_i, h(x_i)) \,|x_0 = x \right\rangle \right\} \\
V_h(x_k) &= E\left\{ r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) \right\}
\end{aligned} \tag{2.80}$$

Logo, ao atualizar a função de valor, é feita uma aproximação da equação acima, por utilizar V_k e não V_h real. Esta aproximação é utilizada como alvo na aprendizagem TD, de forma que V_k é atualizado a partir da sua diferença com a aproximação de V_h.

O Crítico assume a forma de um erro TD, sendo calculado como na Eq.(2.79). Consequentemente, o erro TD é dado por:

$$e_k = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k) \tag{2.81}$$

A regra de aprendizado é dada por:

$$V_h(x_k) \rightarrow V_h(x_k) + \epsilon e_k \tag{2.82}$$

é chamada de Regra Temporal Diferencial de Aprendizagem. Pode ser considerado como um erro de previsão entre as estimativas $V(x_k)$ e $V(x_{k+1})$.

Neste caso, $V_h(x_k)$ representa a Função Valor atual considerada pelo Crítico. Após cada decisão tomada pelo Ator, o Crítico avalia o novo estado do ambiente, de forma a determinar se a decisão tomada foi adequada ou não. Para cada política, o Crítico manterá funções de valor diferentes, de forma a manter coerência e permitir que políticas diferentes possam ser utilizadas para diferentes estados do ambiente. O Crítico atualiza, após cada iteração, a função valor daquele estado de acordo com a Eq.(2.79).

2.5 Conclusão

Foi exposto neste capítulo a formulação do LQR discreto e sua solução a partir de Programação Dinâmica. Pode-se perceber que a partir de um número N de iterações, consegue-se a convergência à solução do DLQR. Vale lembrar que deve-se partir de uma condição inicial dada pelo valor final assumido por P de *Riccati* tendo em vista que o processo de PD é de retrocesso no tempo.

No ponto de vista de controle ótimo, a determinação do ganho K do controlador, através do DLQR, pode ser verificada na melhor escolha das matrizes de ponderação Q e R. A escolha adequada destas matrizes pode acarretar uma melhor alocação de autovalores.

No método MSH-QR proposto, pode-se verificar a influência na escolha dos pesos das matrizes. Percebeu-se que o ganho K pode ser dividido em dois outros ganhos: um com a influência somente da matriz R e outro com a influência de ambas. Então para determinados valores de R ou Q tem-se uma variação na equação do ganho K do controlador.

Foi visto uma introdução com os principais aspectos sobre AR. Viu-se a importância da Função Valor e a evolução desde o Processo de Decisão *Markoviano* até chegar-se a uma estrutura de AR chamada de Ator-Crítico. Tal estrutura é relevante na resolução por ADP que serão apresentados nos capítulos seguintes.

Capítulo 3

APRENDIZAGEM POR REFORÇO PARA SISTEMAS DISCRETOS

No projeto de sistemas de controle com realimentação, são necessários algoritmos de projeto e técnicas de análise que possam garantir um bom desempenho e margens de segurança. Controles de realimentação, sem estabilidade, desempenho ou robustez garantidas, não são aceitos nas indústrias. A forma padrão para a prestação de tais garantias é usar a estrutura e as ferramentas fornecidas pela matemática (LEWIS e VRABIE, 2009*a*).

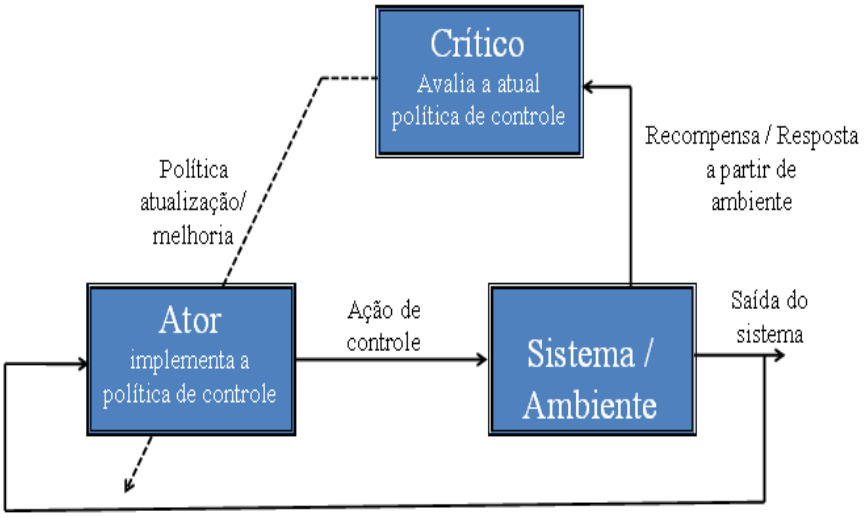

Figura 3.1: Aprendizado por Reforço com Ator/Crítico para sistema de controle.

Como já mencionado, a Aprendizagem por Reforço é a modificação das ações baseadas nas interações com o ambiente. Um ator ou agente interage com o ambiente e modifica suas ações, ou políticas de controle, baseado em estímulos

recebidos em resposta a essas ações. No Capítulo 2, uma classe de AR usando Ator-Crítico foi exposta. Esta, será utilizada para sistema de controle, conforme a Figura 3.1 (LEWIS e VRABIE, 2009*a*). Os algoritmos de Aprendizagem por Reforço são construídos a partir da idéia de que sucessivas decisões de controle devem ser lembradas por meio de um sinal de reforço (SI *et al.*, 2004)(LENDARIS, 2009).

Programação Dinâmica oferece um conjunto de métodos para resolver o problema de otimização, aproveitando a propriedade de *Markov*. Os dois principais métodos são chamados de Política de Iteração (PI) e Valor de Iteração (VI) (LEWIS e VRABIE, 2009*b*) (LEWIS e VAMVOUDAKIS, 2010*a*). A ADP é baseada neste dois métodos. Em contraste com Valor de Iteração, a Política de Iteração necessita de uma política admissível de controle inicial (LEWIS e VRABIE, 2009*a*).

Neste capítulo aborda-se métodos de soluções de sistemas de controle discreto através de controle ótimo utilizando-se como base a teoria de AR. A princípio, traça-se a estratégia de controle ótimo discreto generalizado com base na Função Valor. Em seguida, a partir da abordagem inicial, caracteriza-se o sistema discreto sob a visão de AR e PD. Logo a seguir, sugere-se a solução por PI e VI. Para finalizar o capítulo, toda a abordagem adotada de maneira generalista, é exposta no ponto de vista do DLQR e suas soluções para sistemas dinâmicos discretos. Vale frisar que os algoritmos aqui propostos são adequados para soluções *offline*, tendo em vista que se necessita do conhecimento da dinâmica do sistema.

3.1 Controle ótimo para tempo discreto

Os sistemas dinâmicos são descritos através de equações EDOs não lineares, por análises físicas e matemáticas, da forma $\dot{x} = f(x, u)$, com o estado $x(t) \in R^n$ e a entrada de controle $u(t) \in R^m$ no espaço contínuo. Alguns métodos padrões para a discretização do sistema continuo, convenientes para o controle por computador, podem ser utilizados. O resultado da discretização pode ser expresso em espaço de estados na forma de $x_{k+1} = F(x_k, u_k)$, com k sendo o tempo discreto.

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

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

com o estado $x_k \in R^n$ e a entrada de controle $u_k \in R^m$. A política de controle é definida como uma função de $h(\cdot) : R^n \rightarrow R^m$. Isto é, para cada estado x_k tem-se uma ação de controle definida da seguinte forma:

$$u_k = h(x_k) = -Kx_k \tag{3.2}$$

Tais mapeamentos são também conhecidos como controladores de realimentação. Vários métodos podem ser utilizados para a obtenção da política de controle de realimentação, dentre eles tem-se o controle ótimo através da resolução da equação de *Riccati*, controle adaptativo, controle no domínio da frequência, etc. No Aprendizado por Reforço, a política de controle é aprendida em tempo real a partir de estímulos recebidos do ambiente. Como já mencionado, na Aprendizagem por Reforço, o ator é o agente que gera a política de controle, matematicamente está representado pela Eq.(3.2), que tem como entrada x_k e saída u_k. Vale ressaltar que tudo fora do ator é considerado como ambiente, ou seja, o sistema da Eq.(3.1), assim como todos os distúrbios possíveis gerados, formam o ambiente.

Partindo-se do princípio de AR, defini-se uma função de custo, para horizonte infinito, como:

$$V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) = \sum_{i=0}^{\infty} \gamma^i r(x_{k+i}, u_{k+i}) \tag{3.3}$$

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

Assume-se que um sistema é estabilizável em um conjunto $\Omega \in R^n$, se existir uma política de controle $u_k = h(x_k)$ em que o sistema da Eq.(3.1) em malha fechada seja assintoticamente estável em Ω. A política de controle $u_k = h(x_k)$ é dita admissível se garante estabilidade e produz um custo finito $V_h(x_k)$ (LEWIS e VRABIE, 2009*a*).

O objetivo do controle ótimo é selecionar a política que minimiza:

$$V^*(x_k) = \min_{h(.)} (\sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h(x_i))) \tag{3.4}$$

A política ótima de controle é dada por:

$$u_k^* = h^*(x_k) = \arg\min_{h(.)}(\sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, h(x_i))) \tag{3.5}$$

Percebe-se que no campo da inteligência computacional, a Eq.(3.3) é interpretada como reforço e seu objetivo é maximizá-la. A maneira de como o valor de custo é encontrado é a principal diferença entre controle com realimentação e Aprendizagem por Reforço.

3.2 AR e princípio de otimalidade de *Bellman*

A Programação Dinâmica é uma técnica utilizada para a otimização de processos de decisão de multiestágios. A condição do processo dentro de cada estágio é denominada de estado. Cada estágio inclui uma tomada de decisão que pode alterar o estado do processo representando uma transição do estado corrente e o estado futuro. Dentro do processo multiestágios, o objetivo do tomador de decisão é encontrar uma política ótima proveniente das decisões.

A determinação de uma política ótima para um processo de decisão multiestágio está embasada no principio da otimalidade de *Bellman*: "Uma política ótima apresenta a propriedade segundo a qual, a despeito das decisões tomadas para assumir um estado particular num certo estágio, as decisões restantes a partir deste estado devem constituir uma política ótima"(KUO, 1980).

- Exemplo: Seja o sistema em que tem-se:
  - Vetor de **estado** x_k.
  - **Estágio** k.
  - Transformação do tipo $f(x_k, u_k, k)$.
  - u_k: vetor de **decisão (controle)** no estágio k.
  - Saindo de x_0 ,deseja-se levar o sistema para x_k no **Estágio** k utilizando-se controles (u_0, u_1, u_{k-1}).

O conjunto $u = (u_0, u_1, u_{k-1})$ é uma política admissível (ótima) quando os controles u_k e os estados x_k resultantes satisfazem restrições eventualmente impostas em cada estágio.

A Programação Dinâmica tem como características a sub-estrutura ótima, em que uma solução ótima pode ser encontrada a partir de soluções ótimas para seus subproblemas.

Escrevendo-se (3.3) como

$$V_h(x_k) = r(x_k, u_k) + \gamma \sum_{i=k+1}^{\infty} \gamma^{i-(k+1)} r(x_i, u_i) \tag{3.6}$$

Percebe-se que a equação a diferença equivalente a (3.3) é dada por:

$$V_h(x_k) = r(x_k, u_k) + \gamma V_h(x_{k+1}), \quad V_h(0) = 0 \tag{3.7}$$

Isto é, ao invés de se avaliar a soma infinita dada pela Eq.(3.3), pode-se resolver a equação a diferença (3.6) para se obter o valor da política de controle atual $u_k = h(x_k)$.

Esta é a equação não linear de *Lyapunov* conhecida como equação de *Bellman* (KIRK, 1970)(LEWIS e SYRMOS, 1995). Avaliar o valor da política atual usando a equação de *Bellman* é o primeiro conceito chave no desenvolvimento de técnicas de Aprendizagem por Reforço. O Hamiltoniano de tempo discreto pode ser definido como:

$$H(x_k, h(x_k), \Delta V(x_k)) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k) \tag{3.8}$$

Em que $\Delta V(x_k) = \gamma V_h(x_{k+1}) - V_h(x_k)$ é o operador a diferença de avanço. A função Hamiltoniana captura a energia contida ao longo da trajetória de um sistema como reflexo do ótimo desempenho desejado. A equação de *Bellman* requer que o Hamiltoniano seja igual a zero para o valor associado com a política prescrita (LEWIS e VRABIE, 2009*a*).

O valor ótimo pode ser escrito usando a equação de *Bellman* como:

$$V^*(x_k) = \min_h (r(x_k, h(x_k)) + \gamma V_h(x_{k+1})) \tag{3.9}$$

De acordo com o princípio de otimalidade de Bellman tem-se:

$$V^*(x_k) = \min_{u_k} (r(x_k, u_k) + \gamma V^*(x_{k+1})) \tag{3.10}$$

A Eq.(3.10) é conhecida como equação de Otimalidade de *Bellman*, ou *Hamilton-Jacobi-Bellman* (HJB) de tempo discreto. A política ótima de controle baseada

na Eq.(3.10):

$$h^*(x_k) = \arg\min_{u_k}(r(x_k, u_k) + \gamma V^*(x_{k+1})) \tag{3.11}$$

Como se precisa determinar a política ótima no tempo $k+1$, usando-se a Eq.(3.9), para se determinar a política ótima no tempo k , o Príncipio de *Bellman* é um procedimento de retrocesso no tempo (*backwards-in-time*) para resolver o problema do controle ótimo conhecido como Programação Dinâmica (PD). Métodos de Programação Dinâmica são métodos *offline* para projetos de controle em que geralmente necessita-se de todo conhecimento da dinâmica do sistema. Isto é, $f(x_k)$ e $g(x_k)$ devem ser conhecidos.

3.3 Política de Iteração, Equação de Ponto Fixo e Valor de Iteração

Após a apresentação do desenvolvimento da formulação matemática, agora, pode-se direcioná-la para a aplicação de AR para Controle Ótimo. Considere qualquer política de controle admissível $u_k = h(x_k)$ com $V_h(x_k)$. Determinar uma nova política a partir deste valor usando a operação:

$$h^{'}(x_k) = \arg\min_{h(.)}(r(x_k, u_k) + \gamma V(x_{k+1})) \tag{3.12}$$

É mostrado que a nova política $h^{'}(x_k)$ é melhorada na medida em que tem um valor $V_h^{'}(x_k)$ igual ou inferior ao antigo valor $V_h(x_k)$. Isto sugere o seguinte método iterativo para se determinar o controle ótimo, que é conhecido como Política de Iteração (LEWIS e VRABIE, 2009*a*).

3.3.1 Algoritmo de Política de Iteração

Nota-se por meio do aAlgoritmo 2, descrito a seguir, que na Política de Iteração, necessita de uma política inicial admissível, ou seja, que garanta a estabilidade do sistema. A avaliação do valor da política atual no PI utilizando a equação de *Bellman* leva a determinação da política $h_j(x_k)$ a partir de todos os estados atuais x_k. Isso é chamado de *backup* completo e pode envolver cálculo computacional significativo. Neste algoritmo, a etapa de avaliação das políticas

dada por:

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1}) \tag{3.13}$$

E a etapa de melhoria da política dada por:

$$h_{j+1}(x_k) = \arg\min_{u_k}(r(x_k, u_k) + \gamma V_{j+1}(x_{k+1})) \tag{3.14}$$

A seguir tem-se, o algoritmo para resolução do problema utilizando-se PI.

ALGORITMO 2($AR - PI$)

1 ▷ **Inicialização**
2 Selecionar qualquer política de controle admissível.
3 $h_0(x_k)$
4 Selecionar o Fator de Desconto:
5 $0 < \gamma \leq 1$
6 Iteração inicial.
7 $j = 0$.
8 ▷ **Processo Iterativo**.
9 **for** $j \rightarrow j + 1$
10 **do**
11 ▷ **Etapa de avaliação das políticas**: Determina o valor da política corrente usando a equação de *Bellman*.
12 $V_{j+1}(x_k) \leftarrow r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$
13 ▷ **Etapa de política de melhoria**: Determina a melhor política.
14 $h_{j+1}(x_k) \leftarrow \arg\min_{u_k}(r(x_k, u_k) + \gamma V_{j+1}(x_{k+1}))$
15 **if** $\gamma V_{j+1}(x_{k+1}) - V_{j+1}(x_k) \leftarrow 0$
16 **then**
17 **Fim do Processo Iterativo**.

Se a função de utilidade for dada por:

$$r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k \tag{3.15}$$

E a dinâmica do sistema for dada por:

$$x_{k+1} = f(x_k) + g(x_k) u_k \tag{3.16}$$

Então:

$$h_{j+1}(x_k) = -\frac{\gamma}{2} R^{-1} g^T(x_k) \nabla V_{j+1}(x_{k+1}) \tag{3.17}$$

Em que $\nabla V(x) = \frac{\partial V(x)}{\partial x}$ é o gradiente da função de custo interpretado como sendo uma coluna de vetores (AL-TAMIMI, *et al.*, 2008) (AL-TAMIMI e LEWIS, 2007).

Vale ressaltar que o algoritmo PI, exige para cada estado x_k a resolução da equação não-linear de *Lyapunov* que é dada na etapa de avaliação das políticas (LEWIS e VRABIE, 2009*a*).

3.3.2 Equação de Ponto Fixo

A equação de *Bellman* é uma equação de ponto fixo, visto que dado uma política admissível $u_k = h(x_k)$, tem-se um único ponto fixo $V_h(x_k)$ e o seguinte mapa de contração,

$$V^{i+1}(x_k) = r(x_k, h(x_k)) + \gamma V^i(x_{k+1}) \tag{3.18}$$

pode ser iterado a partir de qualquer $V^0(x_k)$ resultando em $V^i(x_k) \rightarrow V_h(x_k)$ (LEWIS e VRABIE, 2009*a*).

Pode-se substituir a Eq.(3.18), pela Eq.(3.13) em que a iteração em i é feita com a mesma política de controle $h_j(\cdot)$ ate a convergência. Assim, $V^i(x) \rightarrow V_{j+1}(x)$ quando $i \rightarrow \infty$, sugerindo o algoritmo chamado de Valor de Iteração.

3.3.3 Algoritmo de Valor de Iteração

O algoritmo de Valor de Iteração é baseada no fato de que a Equação de Otimalidade de *Bellman* (3.10), também é uma equação de ponto fixo. Por outro lado, o algoritmo VI depende da solução da etapa de política de melhoria, que é simplesmente uma equação de recursão (LEWIS e VRABIE, 2009*a*).

Neste algoritmo, a etapa de atualização do valor é dada por:

$$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_j(x_{k+1}) \tag{3.19}$$

E a etapa de política de melhoria dada por:

$$h_{j+1}(x_k) = \arg\min_{u_k}(r(x_k, u_k) + \gamma V_{j+1}(x_{k+1})) \tag{3.20}$$

A seguir tem-se o algoritmo para resolução do problema utilizando-se VI.

ALGORITMO 3($AR - VI$)

1 ▷ **Inicialização**
2 Selecionar qualquer política de controle não necessariamente admissível.
3 $h_0(x_k)$, $V^0(x_k)$.
4 Selecionar o Fator de Desconto:
5 $0 < \gamma \leq 1$
6 Iteração inicial.
7 $j = 0$.
8 ▷ **Processo Iterativo**.
9 **for** $j \rightarrow j + 1$
10 **do**
11 ▷ **Etapa de atualização do Valor**: Atualiza o valor usando.
12 $V_{j+1}(x_k) \leftarrow r(x_k, h_j(x_k)) + \gamma V_j(x_{k+1})$
13 ▷ **Etapa de política de melhoria**: Determina a melhor política.
14 $h_{j+1}(x_k) \leftarrow \arg\min_{u_k}(r(x_k, u_k) + \gamma V_{j+1}(x_{k+1}))$
15 **if** $\gamma V_{j+1}(x_{k+1}) - V_j(x_k) \leftarrow 0$
16 **then**
17 **Fim do Processo Iterativo**.

É importante notar que, agora, o valor antigo, V_j, é usado no lado direito, em contraste com a etapa de PI. Nota-se que o VI não exige uma política de estabilização inicial. Por outro lado, a etapa de atualização do valor no algoritmo VI, envolve menos capacidade computacional sendo chamado de *backup* parcial.

As equações de ponto fixo, com a formulação adequada, podem ser utilizadas para AR *online*, que aprendem através dos dados acumulados durante a trajetória do sistema (LEWIS e VRABIE, 2009*a*).

3.3.4 Algoritmo de Política Generalizada

Aprendizagem por Reforço sugere outro algoritmo chamado de Política de Iteração Generalizada (*Generalized Policy Iteration*- GPI). Isto é, poucos passos de horizonte $Khoz$ são tomados no sentido de avaliar o valor da política atual. O algoritmo é executado tomando-se $Khoz$ passos para a resolução da Equação de *Lyapunov* a cada iteração j.

ALGORITMO 4($AR - GPI$)

1 ▷ **Inicialização**
2 Selecionar qualquer política de controle não necessariamente admissível.
3 $h_0(x_k)$, $V_0(x_k)$.
4 Selecionar o Fator de Desconto:
5 $0 < \gamma \leq 1$
6 Iteração inicial.
7 $j = 0$.
8 ▷ **Processo Iterativo**.
9 **for** $j \rightarrow j + 1$
10 **do**
11 **for** $i \rightarrow 0 : Khoz - 1$
12 **do**
13 ▷ **Etapa de atualização do Valor**: Na etapa j, atualizar o valor usando.
14 $V_j^{i+1}(x_k) \leftarrow r(x_k, h_j(x_k)) + \gamma V_j^i(x_{k+1})$
15 ▷ **Etapa de política de melhoria**: Determina a melhor política.
16 $h_{j+1}(x_k) \leftarrow \arg\min_{u_k}(r(x_k, u_k) + \gamma V_{j+1}(x_{k+1}))$
17 **if** $\gamma V_j^{i+1}(x_{k+1}) - V_j^i(x_k) \leftarrow 0$
18 **then**
19 **Fim do Processo Iterativo**.

Para um número finito $Khoz$, tem-se como condições $V_j^0 = V_j$ e $V_{j+1} = V_j^{Khoz}$. Na GPI, em cada etapa de atualização política, apenas um backup parcial é feito dos valores.

A etapa de atualização do valor no algoritmo GPI é composto por $Khoz$ etapas da recursão usando o mesmo ganho fixo. Em um caso extremo, quando tem-se $Khoz = 1$, produz o Valor de Iteração, em que apenas um passo de *backup* é realizado. Quando define-se $Khoz = \infty$, a etapa de atualização do valor do algoritmo é realizada até a convergência, fornecendo assim a solução por Política de Iteração.

3.4 Solução do controlador discreto ótimo

Após as noções sobre AR, através das soluções por PI e VI, deseja-se, agora, mostrar que de fato elas podem ser aplicadas para sistemas de controle.

Sendo um sistema linear invariante no tempo na seguinte forma discreta :

$$x_{k+1} = A_d x_k + B_d u_k \tag{3.21}$$

com o estado $x_k \in R^n$ e a entrada de controle $u_k \in R^m$. A ação de controle definida da seguinte forma.

$$u_k = h(x_k) = -Kx_k \tag{3.22}$$

sendo K uma matriz de ganho a ser determinada.

Assume-se que (A_d, B_d) são estáveis, ou seja, existe umas matriz K em que garanta que o sistema em malha fechada da forma

$$x_{k+1} = (A_d - B_d K)x_k \equiv A_c x_k \tag{3.23}$$

seja assintoticamente estável.

A Função de custo em sua forma quadrática é definida como:

$$\begin{aligned} V_h(x_k) &= \sum_{i=k}^{\infty} \gamma^{i-k}(x_i^T Q x_i + u_i^T R u_i) = \sum_{i=0}^{\infty} \gamma^i (x_{k+i}^T Q x_{k+i} + u_{k+i}^T R u_{k+i}) \\ V_h(x_k) &= \sum_{i=k}^{\infty} \gamma^{i-k} x_i^T (Q + K^T R K) x_i = \sum_{i=0}^{\infty} \gamma^i x_{k+i}^T (Q + K^T R K) x_{k+i} \end{aligned} \tag{3.24}$$

Em que tem-se a função de utilidade $r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k$ com as matrizes de ponderação $Q = Q^T > 0$ e $R = R^T > 0$.

Assumindo-se agora que $(Q + K^T R K) = Y$, e utilizando-se a Eq. (3.23) tem-se:

$$V_h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} x_i^T Y x_i = \sum_{i=0}^{\infty} \gamma^i x_{k+i}^T Y x_{k+i} \tag{3.25}$$

$$V_h(x_k) = \sum_{i=0}^{\infty} \gamma^i [(A_c^T)_i x_k^T Y (A_c)_i x_k] \tag{3.26}$$

Esta soma será convergente uma vez que a matriz A_c tem todos os seus autovalores no circulo unitário.

O objetivo do projeto de controle ótimo, é encontrar uma matriz de ganho K, que minimize a função de custo $V_h(x_k) = V_K(x_k)$ para todos os estados x_k.

No caso do LQR (*Linear Quadratic Regulator*), seu valor ótimo é dado por $V_k(x_k) = x_k^T P x_k$ para uma dada matriz P Hermitiana ou real simétrica definida positiva a ser determinada. A Eq.(3.7) de *Bellman* para o caso do LQR discreto (LEWIS e VAMVOUDAKIS, 2010*b*) (LEWIS e VAMVOUDAKIS, 2010*a*) é dada por:

$$x_k^T P x_k = x_k^T Q x_k + u_k^T R u_k + \gamma(x_{k+1}^T P x_{k+1}) \tag{3.27}$$

em termos de ganho de realimentação é dada por:

$$\begin{aligned} x_k^T P x_k &= x_k^T (Q + K^T R K + \gamma[(A_d - B_d K)^T P (A_d - B_d K)]) x_k \\ x_k^T P x_k &= x_k^T (Y + \gamma A_c^T P A_c) x_k \end{aligned} \tag{3.28}$$

Para todos os estados x_k tem-se:

$$\begin{aligned} &\gamma[(A_d - B_d K)^T P (A_d - B_d K)] - P + Q + K^T R K = 0 \\ &\gamma A_c^T P A_c - P + Y = 0 \end{aligned} \tag{3.29}$$

Para a Eq.(3.26) tem-se que:

$$P_\infty = \sum_{i=0}^{\infty} \gamma^i [(A_c^T)_i Y (A_c)_i] \tag{3.30}$$

A Eq.(3.29), quando o K é fixado, é conhecida como equação de *Lyapunov*. Resolvendo esta equação, dado um ganho K, fornece $P = P^T > 0$, tal que $V_K(x_k) = x_k^T P x_k$ é o custo utilizando a política K, tem-se:

$$V_K(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} x_i^T (Q + K^T R K) x_i = \sum_{i=0}^{\infty} \gamma^i x_{k+i}^T (Q + K^T R K) x_{k+i} \tag{3.31}$$

escrevendo-se a equação de *Bellman* como:

$$x_k^T P x_k = x_k^T Q x_k + u_k^T R u_k + \gamma[(A_d x_k + B_d u_k)^T P (A_d x_k + B_d u_k)] \tag{3.32}$$

A minimização é obtida através da diferenciação em relação a u_k. Então para a Eq.(3.7) de *Bellman* (AL-TAMIMI, *et al.*, 2008) (AL-TAMIMI e LEWIS, 2007):

$$\frac{\partial V_h(x_k)}{\partial u_k} = \frac{\partial r(x_k, u_k)}{\partial x_k} \frac{\partial x_k}{\partial u_k} + \frac{\partial r(x_k, u_k)}{\partial u_k} + \gamma \left(\frac{\partial V_h(x_{k+1})}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial u_k} \right) \tag{3.33}$$

Minimizando-se a função de custo em relação ao controle u_k, tem-se:

$$\frac{\partial V_h(x_k)}{\partial u_k} = 0 \tag{3.34}$$

$$0 = \frac{\partial r(x_k, u_k)}{\partial x_k}\frac{\partial x_k}{\partial u_k} + \frac{\partial r(x_k, u_k)}{\partial u_k} + \gamma\left(\frac{\partial V_h(x_{k+1})}{\partial x_{k+1}}\frac{\partial x_{k+1}}{\partial u_k}\right) \tag{3.35}$$

$$\begin{aligned} &Ru_k + \gamma B^T P(A_d x_k + B_d u_k) = 0 \\ &u_k = -(R/\gamma + B_d^T P B_d)^{-1} B_d^T P A_d x_k \end{aligned} \tag{3.36}$$

o ganho ótimo de realimentação é:

$$K = (R/\gamma + B_d^T P B_d)^{-1} B_d^T P A_d \tag{3.37}$$

Substituindo-se a Eq.(3.37) na Eq.(3.32) tem-se o *Hamilton-Jacobi-Bellman* no tempo discreto ou equação de otimalidade de *Bellman* dado por

$$\gamma(A_d^T P A_d) - P + Q - \gamma[A_d^T P B_d (R/\gamma + B_d^T P B_d)^{-1} B_d^T P A_d] = 0 \tag{3.38}$$

ou seja, equação algébrica de *Riccati*.

Uma observação importante é o fato da formulação da Eq.(3.29) de *Lyapunov*, partindo-se da Eq.(3.27) de *Bellman*. Primeiro substitui-se x_{k+1} pela dinâmica do sistema para obter-se a Eq.(3.28), logo após o estado x_k é cancelado para obter-se Eq.(3.29). Estes passos tornam-se impossível a aplicação em tempo real *online* métodos de AR para encontrar o controle ótimo. Percebe-se através desses passos, que projeto de controladores ótimos é quase universalmente um processo *offline*, envolvendo soluções de equações de *Riccati* em que deve-se ter o conhecimento da dinâmica da planta, ou seja, as matrizes (A_d, B_d).

3.4.1 Algoritmo de Política de Iteração para o LQR discreto

Para o DLQR, a equação de *Bellman* (3.7) é escrita como (3.27) sendo equivalente a equação de *Lyapunov* (3.29) (LEWIS e VAMVOUDAKIS, 2010*a*). O algoritmo de Política de Interação para o LQR, tem como Etapa de avaliação das políticas:

$$\gamma[(A_d - B_d K_j)^T P_{j+1}(A_d - B_d K_j)] - P_{j+1} + Q + K_j^T R K_j = 0 \tag{3.39}$$

Com a política de atualização:

$$K_{j+1} = (R/\gamma + B_d^T P_{j+1} B_d)^{-1} B_d^T P_{j+1} A_d \tag{3.40}$$

Então tem-se o seguinte algoritmo de PI.

ALGORITMO 5($AR - PI - DLQR$)

1 ▷ - **Inicializações**
2 Ponderações e Sistema Dinâmico.
3 $[Q, R, A_d, B_d]$
4 Valores iniciais de P e K.
5 $[K_0, P_0]$
6 Selecionar o Fator de Desconto:
7 $0 < \gamma \leq 1$
8 Iteração Inicial
9 $j = 0$
10 ▷ **Processo Iterativo**
11 **for** $j \rightarrow j + 1$
12 **do**
13 $\Delta P_k \leftarrow 0$
14 ▷ **Recorrência de *Lyapunov***.
15 $\Delta P_k \leftarrow \gamma[(A_d - B_d K_j)^T P_{j+1}(A_d - B_d K_j)] + Q + K_j^T R K_j - P_{j+1}$
16 ▷ **Ganho Ótimo de realimentação** K.
17 $K_{j+1} \leftarrow (R/\gamma + B_d^T P_{j+1} B_d)^{-1} B_d^T P_{j+1} A_d$
18 **Fim do Processo Iterativo**.

Vale ressaltar, o fato já mencionado, que o algoritmo PI, necessita na inicialização, de uma política inicial admissível. Neste caso, K_0 tem que ser um ganho que garanta a estabilidade do sistema.

3.4.2 Algoritmo de Valor de Iteração para o LQR discreto

Em termos de valor de interação, a etapa de avaliação da política do LQR é:

$$P_{j+1} = \gamma[(A_d - B_d K_j)^T P_j (A_d - B_d K_j)] + Q + K_j^T R K_j \tag{3.41}$$

Em que a política de atualização (3.20) é dada pela Eq.(3.40). Então tem-se o seguinte algoritmo de VI.

ALGORITMO 6($AR-VI-DLQR$)

1 ▷ - **Inicializações**
2 Ponderações e Sistema Dinâmico.
3 $[Q, R, A_d, B_d]$
4 Valores iniciais de P e K.
5 $[K_0, P_0]$
6 Selecionar o Fator de Desconto:
7 $0 < \gamma \leq 1$
8 Iteração Inicial
9 $j = 0$
10 ▷ **Processo Iterativo**
11 **for** $j \rightarrow j + 1$
12 **do**
13 ▷ **Recorrência de *Lyapunov***.
14 $P_{j+1} \leftarrow \gamma[(A_d - B_d K_j)^T P_j (A_d - B_d K_j)] + Q + K_j^T R K_j$
15 ▷ **Ganho Ótimo de realimentação** K.
16 $K_{j+1} \leftarrow (R/\gamma + B_d^T P_{j+1} B_d)^{-1} B_d^T P_{j+1} A_d$
17 **if** $P_{j+1} - P_j \leftarrow 0$
18 **then**
19 **Fim do Processo Iterativo**.

Percebe-se que a Política de Interação envolve a completa solução da equação de *Lyapunov* (3.39) e requer um ganho estabilizante K_j em cada etapa j (*Backup* completo). Por outro lado o Valor de Interação envolve somente a recursividade de *Lyapunov* (3.41) em cada passo j, sendo de fácil computabilidade, não requerendo um ganho, K_0, inicial admissível (*Backup parcial*).

3.4.3 Algoritmo de Política Generalizada para o LQR discreto

Se K_j for um ganho estabilizante, então a Eq.(3.41) promove a solução de *Lyapunov* da Eq.(3.39), com um ganho fixo K_j, até a convergência. O algoritmo de *GPI*, promove a solução da equação de *Riccati*.

ALGORITMO 7($AR-GPI-DLQR$)

1 ▷ - **Inicializações**
2 Ponderações e Sistema Dinâmico.
3 $[Q, R, A_d, B_d]$
4 Valores iniciais de P e K.
5 $[K_0, P_0]$
6 Selecionar o Fator de Desconto:
7 $0 < \gamma \leq 1$
8 Iteração Inicial
9 $j = 0$
10 ▷ **Processo Iterativo**
11 **for** $j \rightarrow j + 1$
12 **do**
13 **for** $i \rightarrow 0 : Khoz - 1$
14 **do**
15 ▷ **Recorrência de *Lyapunov***.
16 $P_j^{i+1} \leftarrow \gamma[(A_d - B_d K_j)^T P_j^i (A_d - B_d K_j)] + Q + K_j^T R K_j$
17 ▷ **Ganho Ótimo de realimentação** K.
18 $K_{j+1} \leftarrow (R/\gamma + B_d^T P_{j+1} B_d)^{-1} B_d^T P_{j+1} A_d$
19 **if** $P_j^{i+1} - P_j^i \leftarrow 0$
20 **then**
21 **Fim do Processo Iterativo**.

Para um $Khoz$ inteiro finito, tem-se as condições de $P_j^0 = P_j$ e $P_{j+1} = P_j^{Khoz}$. Como já discutido, para o caso extremo de $Khoz = 1$ tem-se o algoritmo de VI e para $Khoz = \infty$ tem-se o algoritmo de PI através da solução da equação de *Lyapunov*.

3.5 Conclusão

Foi realizado o estudo sobre o controlador ótimo de tempo discreto de maneira generalizada para em seguida, caracterizar o controlador ótimo discreto por soluções de AR e PD. A teoria a cerca de AR, previamente exposta no Capítulo 2, foi direcionada para sistemas de controle. Percebe-se que diferente da abordagem usual, para sistema de controle ótimo discreto, a necessidade é de se minimizar o sinal de reforço.

Propostas de soluções por PI , VI e uma generalista (GPI), que aborda ambas soluções, foram vistas. A diferença principal entre ambas é a necessidade de uma política inicial admissível ou não. Em termos de controle, isso é visto como a necessidade de um ganho de controle K_0 inicial que aloque os autovalores do sistema em uma região de estabilidade no plano complexo Z, ou seja, garantir que os autovalores estejam dentro do circulo unitário. Para o DLQR, a solução é

obtida através da equação de *Riccati* ou *Lyapunov* para consequentemente obter-se o ganho K.

Capítulo 4

ADP PARA CONTROLE ÓTIMO *ONLINE*

A Programação Dinâmica como solução do controle ótimo, é um procedimento de retrocesso no tempo (*backwards-in-time*), sendo utilizado para planejamento *offline*. A Eq.(3.7) de *Bellman*, leva a vários métodos iterativos (PI e VI) para aprender a solução do controle ótimo sem ter que resolver a equação de *Hamilton-Jacobi-Bellman* (HJB).

Diversas contribuições para AR são feitas por meio da Aproximação da Função Valor (*Value Function Approximation* - VFA). Na aprendizagem da Função Valor por métodos de AR, é necessário armazenar o valor ótimo e o controle ótimo em função do vetor de estado $x \in R^n$. No Processo de Decisão *Markoviano*, que são em sistemas de estados discretos, o estado pode assumir apenas um número finito de valores discretos prescrito, o que leva a chamado complexidade computacional, mais conhecido por meio de *Bellman* por "Maldição da Dimensionalidade". Com o aumento de estados, mais informações devem ser guardadas, normalmente em forma de tabelas. No entanto, utilizando VFA, onde o crítico e, se desejar, o ator são parametrizados usando aproximadores de função, este problema é atenuado (VAMVOUDAKIS e LEWIS, 2009).

Neste capítulo, será visto como formular estes procedimentos em um método de AR *online* em tempo real usando-se dados medidos do sistema ao longo de sua trajetória. Estes métodos são amplamente chamado de *Approximate Dynamic Programming* - ADP (Programação Dinâmica Aproximada) ou *Neurodynamic Programming* - NDP (Programação Neurodinâmica). Existem dois principais ingredientes: Temporal Diferencial (*Temporal Difference* - TD) de erro e Aproximação da Função Valor (*Value Function Approximation* - VFA). As quatro principais estruturas de ADP propostas por Werbos , (WERBOS, 1974) (WERBOS, 1989), (WERBOS, 1990) e comumente utilizadas serão comentadas (LAN-

DELIUS e KNUTSSON, 1996).

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

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

$$e_k = H(x_k, h(x_k), \Delta V_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k) \tag{4.1}$$

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

$$0 = H(x_k, h(x_k), \Delta V_k) = r(x_k, h(x_k)) + \gamma V_h(x_{k+1}) - V_h(x_k) \tag{4.2}$$

A diferença temporal do erro pode ser considerada como, a previsão do erro entre o desempenho previsto e o desempenho observado em resposta a uma ação aplicada ao sistema.

A Eq.(4.2) é um elemento importante na resolução *online* da equação não linear de *Lyapunov* utilizando-se somente dados medidos ao longo da trajetória do sistema. Para sistemas não lineares, TD é de difícil solução.

Uma solução prática para solução do TD, é fazendo-se uma aproximação da função valor $V_h(\cdot)$, utilizando-se aproximação paramétrica chamada de Programação Dinâmica Aproximada (*Approximate Dynamic Programming* - ADP).

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

$$e_k = x_k^T Q x_k + u_k^T R u_k + \gamma(x_{k+1}^T P x_{k+1}) - x_k^T P x_k \tag{4.3}$$

Para simplificar-se a Eq. (4.3), utiliza-se o produto de *Kronecker* para escrever:

$$V_K(x_k) = x_k^T P x_k = vec(x_k^T P x_k) = (x_k \otimes x_k)(vec(P))^T \equiv \bar{p}^T \bar{x}_k \tag{4.4}$$

sendo $\otimes$ o produto de *Kronecker* (BREWER, 1978) e $vec(P)$ o vetor formado pelos elementos da matriz P em um vetor coluna. O produto de *Kronecker*, $\bar{x}_k = x_k \otimes x_k$, é um vetor quadrático contendo todos os produtos possíveis de n componentes de x_k. Vale ressaltar que a matriz P é simétrica, então tem-se somente $n(n+1)/2$ elementos independentes, assim, pode-se remover os elementos redundantes em $\bar{x}_k = x_k \otimes x_k$ para definir-se um conjunto de base quadrática, $\bar{x}_k$, com $n(n+1)/2$ elementos independentes. O vetor de parâmetros desconhecidos é $\bar{p} = vec(P)$, os elementos da matriz P.

O erro TD, passa ser escrito como:

$$\begin{aligned} e_k &= x_k^T Q x_k + u_k^T R u_k + \gamma(\bar{p}^T \bar{x}_{k+1}) - \bar{p}^T \bar{x}_k \\ e_k &= r(x_k, u_k) + \gamma(\bar{p}^T \bar{x}_{k+1}) - \bar{p}^T \bar{x}_k \end{aligned} \tag{4.5}$$

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

4.2 ADP- AR *online* para controle ótimo

Uma aproximação da função $V_h(x)$ pode ser dada da seguinte forma:

$$V_h(x) = W_{V_h}^T \phi(x) + \varepsilon_L(x) \tag{4.6}$$

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

Assumindo-se agora, a aproximação da seguinte forma:

$$V_h(x) = W_{V_h}^T \phi(x) \tag{4.7}$$

Substituindo-se a aproximação da Eq.(4.7) na Eq.(4.1) tem-se:

$$e_k = r(x_k, u_k) + \gamma W_{V_h}^T \phi(x_{k+1}) - W_{V_h}^T \phi(x_k) \tag{4.8}$$

Procedimentos iterativos para resolver a equação TD podem ser utilizados, incluindo a Política de Iteração e Valor de iteração.

4.2.1 Algoritmo de Política de Iteração *online*

As equações para o desenvolvimento deste algoritmo são baseadas na aproximação da função valor e nas equações do algoritmo PI do Capítulo 3. Então tem-se na etapa da avaliação das políticas:

$$W_{V_{j+1}}^T(\phi(x_k) - \gamma\phi(x_{k+1})) = r(x_k, h_j(x_k)) \tag{4.9}$$

e a política de melhoria é dada por:

$$h_{j+1}(x_k) = \arg\min_{u_k}(r(x_k, u_k) + W_{V_{j+1}}^T\varphi(x_{k+1})) \tag{4.10}$$

O algoritmo 8 utiliza a aproximação da função valor por Política de Iteração.

Algoritmo 8($ADP - PI$)

1 ▷ **Inicialização**
2 Selecionar qualquer política de controle admissível.
3 $h_0(x_k)$
4 Iteração Inicial
5 $j = 0$
6 Aproximação da Função Valor
7 $V_j(x_k) = W_{V_j}^T\phi(x_k)$
8 ▷ **Processo Iterativo**.
9 **for** $j \rightarrow j + 1$
10 **do**
11 ▷ **Etapa de avaliação das políticas**: Determina $W_{V_{j+1}}$ por LS.
12 $W_{V_{j+1}}^T(\phi(x_k) - \gamma\phi(x_{k+1})) \leftarrow r(x_k, h_j(x_k))$
13 ▷ **Etapa de política de melhoria**: Determina a melhor política.
14 $h_{j+1}(x_k) \leftarrow \arg\min_{u_k}(r(x_k, u_k) + W_{V_{j+1}}^T\varphi(x_{k+1}))$
15 **if** $W_{V_{j+1}}^T\phi(x_k) - \gamma W_{V_{j+1}}^T\phi(x_{k+1}) \leftarrow 0$
16 **then**
17 **Fim do Processo Iterativo**.

Se a função de utilidade for dada pela Eq.(3.15) e a dinâmica do sistema for Eq.(3.16) então:

$$h_{j+1}(x_k) = -\frac{\gamma}{2}R^{-1}g^T(x_k)\nabla\phi^T(x_{k+1})W_{V_{j+1}} \tag{4.11}$$

Percebe-se que se tem o vetor de aproximação $W_{V_{j+1}} \in R^L$ com L elementos. No instante de tempo $k + 1$ tem-se o valor do estado x_k, a política de controle $u_k = h_j(x_k)$, o próximo estado x_{k+1} e a função de utilidade $r(x_k, h_j(x_k))$, gerando-se assim uma equação escalar que se repete para próximos intervalos de tempo utilizando a mesma política de controle $h_j(\cdot)$, até que se tenha ao menos

L equações para determinar-se a solução de $W_{V_{j+1}}$ por LS ou *Batch* LS (Mínimos Quadrados Batelada) (VRABIE *et al.*, 2009).

Escrevendo-se a Eq. (4.9) como:

$$W_{V_{j+1}}^T \Phi(k) = W_{V_{j+1}}^T (\phi(x_k) - \gamma\phi(x_{k+1})) = r(x_k, h_j(x_k)) \tag{4.12}$$

com

$$\Phi(k) = (\phi(x_k) - \gamma\phi(x_{k+1})) \tag{4.13}$$

sendo o vetor de regressores.

No passo j do algoritmo tem-se somente uma política de controle fixa $u = h_j(x)$. A cada intervalo de tempo k, tem-se um conjunto de dados medidos, $(x_k, x_{k+1}, r(x_k, h_j(x_k))$, então pode-se obter a solução da Eq.(4.12) através de *Recursive Least Squares* -RLS (Mínimos Quadrados Recursivos). O vetor $\Phi(k) = (\phi(x_k) - \gamma\phi(x_{k+1}))$ precisa estar permanentemente excitado (VRABIE *et al.*, 2009).

4.2.2 Algoritmo de Valor de Iteração *online*

Da mesma forma que a Política de Iteração, um algoritmo utilizando AR pode ser dado com base no Valor de Iteração. As equações para o desenvolvimento deste algoritmo são baseadas na aproximação da função valor e nas equações do algoritmo VI do Capítulo 3.

Então tem-se na etapa da avaliação das políticas:

$$W_{V_{j+1}}^T \phi(x_k) = r(x_k, h_j(x_k)) + \gamma W_{V_j}^T \phi(x_{k+1}) \tag{4.14}$$

e a política de melhoria é dada pela Eq.(4.10).

O algoritmo de VI, utilizando-se a aproximação da função valor pode ser visto logo a seguir.

ALGORITMO 9($ADP - VI$)

1 ▷ **Inicialização**
2 Selecionar qualquer política de controle não necessariamente admissível.
3 $h_0(x_k)$, $V^0(x_k)$.
4 Iteração Inicial
5 $j = 0$
6 Aproximação da Função Valor
7 $V_j(x_k) = W_{V_j}^T \phi(x_k)$
8 ▷ **Processo Iterativo**.
9 **for** $j \rightarrow j + 1$
10 **do**
11 ▷ **Etapa de atualização do Valor**: Determina $W_{V_{j+1}}$ por LS.
12 $W_{V_{j+1}}^T \phi(x_k) \leftarrow r(x_k, h_j(x_k)) + \gamma W_{V_j}^T \phi(x_{k+1})$
13 ▷ **Etapa de política de melhoria**: Determina a melhor política.
14 $h_{j+1}(x_k) \leftarrow \arg\min_{u_k}(r(x_k, h_j(x_k)) + \gamma W_{V_{j+1}}^T \phi(x_{k+1}))$
15 **if** $W_{V_{j+1}}^T \phi(x_k) - \gamma W_{V_j}^T \phi(x_{k+1}) \leftarrow 0$
16 **then**
17 **Fim do Processo Iterativo**.

O valor antigo se encontra do lado direito da Eq.(4.14) e o vetor de regressores é agora dado por $\phi(x_k)$, que precisa estar constantemente excitado para convergência do LS.

Aprendizagem por Reforço, como já visto, é um método indireto de controle adaptativo em que os parâmetros da Função Valor, Eq.(4.7), são estimados e o controle é obtido através da Eq.(4.10). O controle ótimo é diretamente calculado em termos de parâmetros de aprendizagem utilizando-se a Eq.(4.10), caracterizando-se um esquema de controle adaptativo direto. Dentro deste contexto, Aprendizagem por Reforço promove soluções de aprendizagem *online* de controle adaptativo que convergem para soluções de controle ótimas.

Para o caso do LQR discreto, a AR utilizando-se a Aproximação da Função Valor, resolve a equação de *Riccati online*, sem que se necessite, a *priori*, do conhecimento da dinâmica do sistema, (A_d, B_d), mas somente da observação dos dados $(x_k, x_{k+1}, r(x_k, h_j(x_k))$ em cada intervalo de tempo ao longo da trajetória do sistema.

4.3 Estruturas básicas de ADP

Os algoritmos de AR utilizando a estrutura Ator Crítico, podem ser utilizadas para determinação de uma lei de controle ótimo para um processo dinâmico tanto *online* como *offline* (SI *et al.*, 2004).

Na implentação *offline*, necessita-se da dinâmica do sistema a ser controlado e os valores dos estados são obtidos através de simulações. O algoritmo gera uma ação de controle u_k e uma função valor V_j, para todos os estados, que se repetem para $j = 0, 1, 2, ...$, até que o algoritmo produza valores ótimos de $h^*(x_k)$ e $V^*(x_k)$.

Ao contrário da equação de *Euler-Lagrange* (Apêndice A) e PD, na implementação *online*, o processo computacional fica dependente somente da variável de estado x_k, não envolvendo o conhecimento da condição final do processo. Os vetores de estado, tornam-se disponíveis progressivamente no tempo $\{x_k|k = 0, 1, 2, ...\}$. Assumindo-se que a planta é totalmente observável, o valor do atual estado x_k, é determinado a partir de medições de saída disponíveis. O próximo estado x_{k+1}, é previsto através do modelo da Eq.(3.1). A lei de controle ótima pode ser determinada *online* para sistemas em que suas características são expostas somente durante a operação.

Uma família de estruturas de ADP foi proposta por Werbos no começo dos anos 90 sendo amplamente utilizada (LENDARIS, 2009) (WANG *et al.*, 2009). A formulação original, é baseada em uma implementação utilizando-se Redes Neurais, porém, qualquer estrutura de aprendizagem,pode ser utilizada. As quatro principais estruturas de ADP propostas por Werbos , (WERBOS, 1974) (WERBOS, 1989), (WERBOS, 1990) são: *Heuristic Dynamic Programming* (HDP), *Dual* HDP(DHP), *Action Dependent* HDP (AD-HDP) (*Q-Learning*) (BRADTKE, 1993), Action Dependent DHP (AD-DHP). A Figura 4.1 expõe as características de cada categoria de ADP.

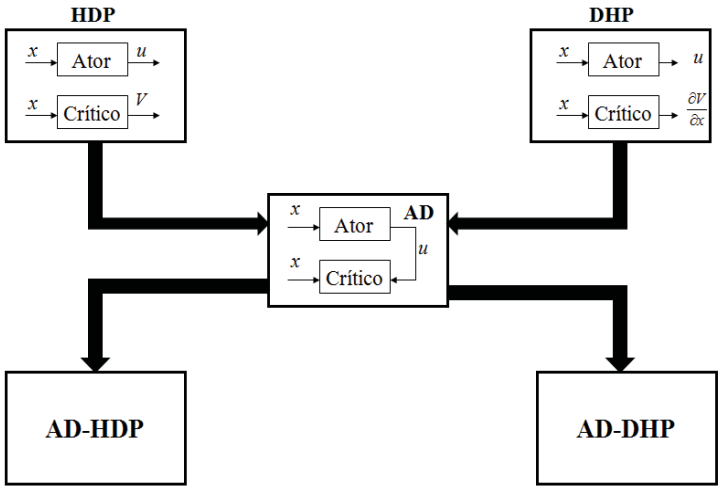

Figura 4.1: Modelos de ADP propostos por Werbos.

Algumas características importantes devem ser notadas nas diferentes estruturas de ADP. A entrada do crítico recebe a informação do estado do sistema (e do modelo de referência da planta, se for o caso). Na estrutura AD (*Action Dependent*) o Crítico também promove as saídas do controlador. Na estrutura HDP, a saída do Crítico promove uma aproximação da Função Valor $V_h(x_k)$. Na estrutura DHP, tem-se uma aproximação do gradiente de $V_h(x_k)$, denotado por $\nabla V_h(x_k) = \frac{\partial V_h(x_k)}{\partial x_k}$.

Existem formulações em que se necessita de um ciclo de treinamento do sistema, porém as já citadas, necessita-se de dois ciclos de treinamento, um para o Crítico e outro para a ação de controle. Dependendo da estrutura da ADP, um ou ambos ciclos de treinamento, irão necessitar do modelo da planta.

Os componentes básicos no processo de ADP são ação/controlador e a planta a ser controlada. O controlador recebe as informações do estado x_k corrente da planta e tem como saída a ação de controle u_k. A planta recebe a ação de controle u_k, e direciona-se para o próximo estado x_{k+1}. Os dados de x_k é fornecido para o Crítico e para a função de utilidade $r(x_k, u_k)$. Todos esses dados são necessários para o cálculo de treinamento do Ator e do Crítico. O treinamento é baseado na Eq.(3.7) de *Bellman*.

4.3.1 *Heuristic Dynamic Programming* (HDP)

HDP é a estrutura de ADP mais básica e amplamente aplicada. A Estrutura de HDP é mostrado na Figura 4.2.

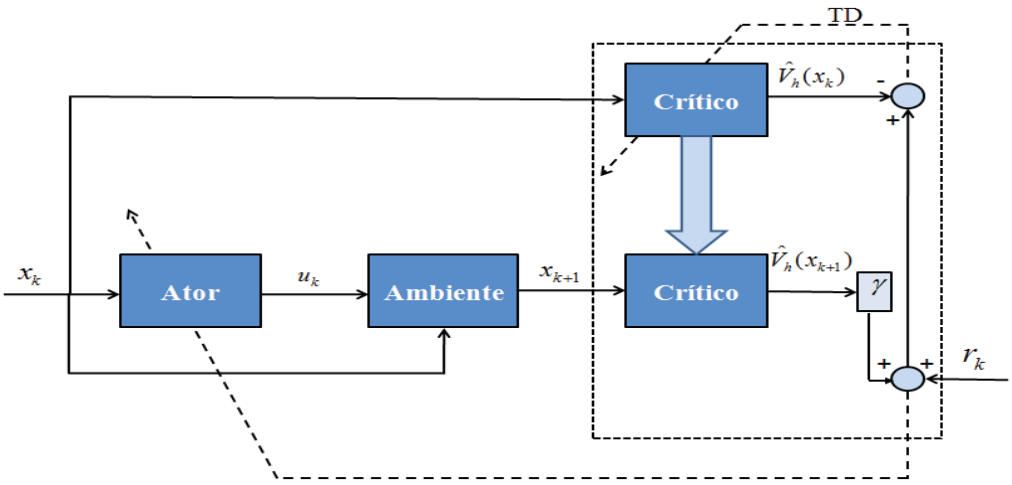

Figura 4.2: Estrutura de HDP.

Neste caso o Crítico estima $V_h(x_k)$ baseado diretamente do estado x_k da planta. O Crítico não necessita do modelo da planta para cálculo. O treinamento do controlador, por outro lado, necessita encontrar as derivadas $\partial \widehat{V}_h(x_k)/\partial u_k$ em cada instante k. Assim, o algoritmo HDP utiliza o modelo da planta somente para a atualização do controlador (WANG *et al.*, 2009).

4.3.2 *Dual Heuristic Programming* (DHP)

Neste algoritmo, o Crítico utiliza o co-estado, ou seja, estima diretamente as derivadas de $V_h(x_k)$ em relação aos estados da planta, $\nabla V_h(x_k) = \frac{\partial V_h(x_k)}{\partial x_k}$.

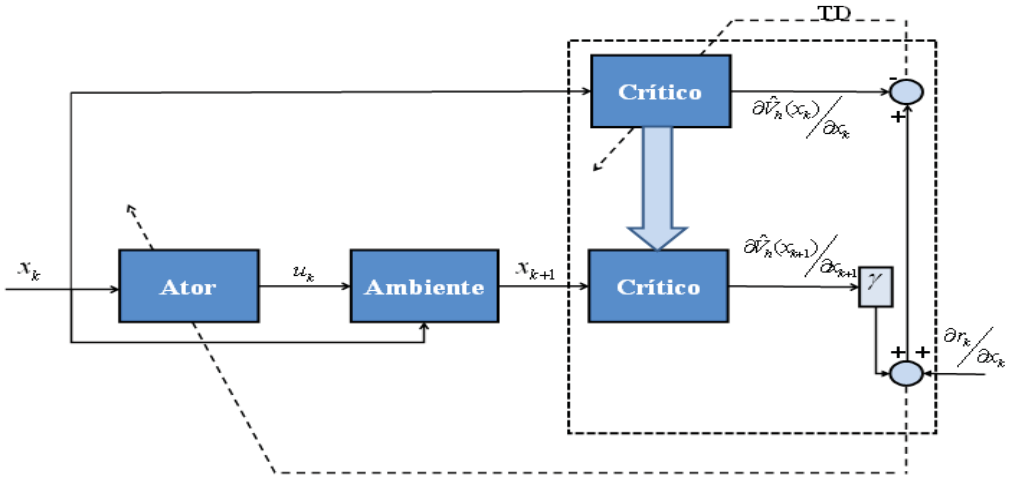

Figura 4.3: Estrutura de DHP.

Para executar o algoritmo, é necessário encontrar a equação de ponto fixo para o co-estado. Então tem-se:

$$\frac{\partial V_h(x_k)}{\partial x_k} = \frac{\partial r(x_k, u_k)}{\partial x_k} + \gamma \frac{\partial V_h(x_{k+1})}{\partial x_k} \tag{4.15}$$

ou

$$\begin{aligned} \nabla V_h(x_k) &= \frac{\partial r(x_k, u_k)}{\partial x_k} + \frac{\partial r(x_k, u_k)}{\partial h(x_k)} \frac{\partial h(x_k)}{\partial x_k} \\ &+ \gamma \left(\frac{\partial x_{k+1}}{\partial x_k} + \frac{\partial x_{k+1}}{\partial h(x_k)} \frac{\partial h(x_k)}{\partial x_k} \right) \nabla V_h(x_{k+1}) \end{aligned} \tag{4.16}$$

Infelizmente, para estimar os valores do lado direito da Eq.(4.16), necessita-se

do conhecimento da dinâmica da planta, uma vez que $\frac{\partial x_{k+1}}{\partial x_k} = f(x_k)$, $\frac{\partial x_{k+1}}{\partial h(x_k)} = g(x_k)$. Além disso, isto requer a implementação *online* de RLS para um n-éssimo vetor, sendo computacionalmente intensivo. A ação de controle u_k é determinada da mesma forma que na HDP, porém como o crítico necessita do conhecimento da planta, então, consequentemente, o controlador será dependente do conhecimento prévio do sistema a ser controlado (WANG *et al.*, 2009).

4.3.3 *Action Dependent Heuristic Dynamic Programming* (AD-HDP)

No algoritmo AD-DHP usa-se tanto o estado x_k quanto o controle u_k como entrada do Crítico, $Q(x_k, u_k)$.

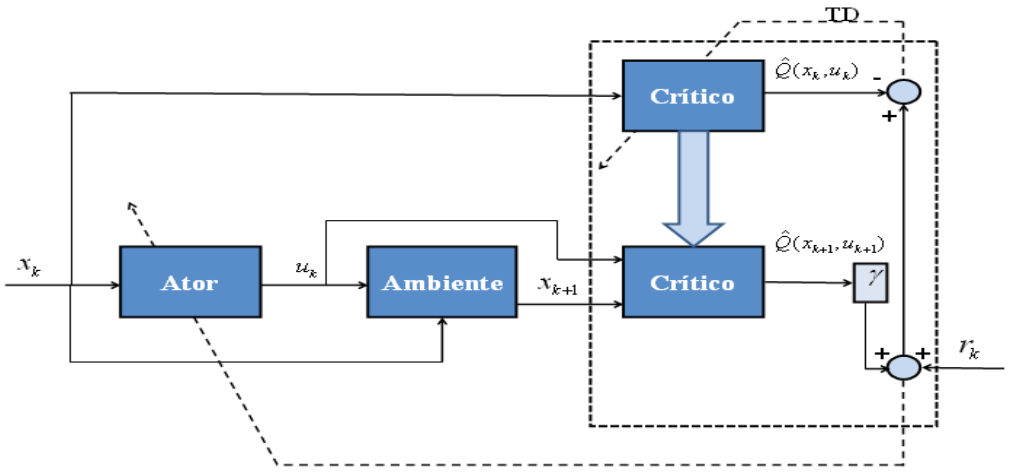

Figura 4.4: Estrutura de ADHDP.

O modo de operação é similar ao do algoritmo HDP. O controle é obtido através da derivada $\frac{\partial Q(x_k,u_k)}{\partial u_k}$. O algoritmo AD-HDP, não necessita do conhecimento da planta para treinamento do processo (LEE *et al.*, 2009) (AL-TAMIMI, *et al.*, 2007*a*).

4.3.4 *Action Dependent Dual Heuristic Programming* (AD-DHP)

No algoritmo AD-DHP usa-se tanto o estado x_k quanto o controle u_k como entrada do Crítico, e como saída tem-se o gradiente de $V_h(x_k)$ em relação aos

estados ($\frac{\partial V_h(x_k)}{\partial x_k}$) e ao controle ($\frac{\partial u_k}{\partial x_k}$).

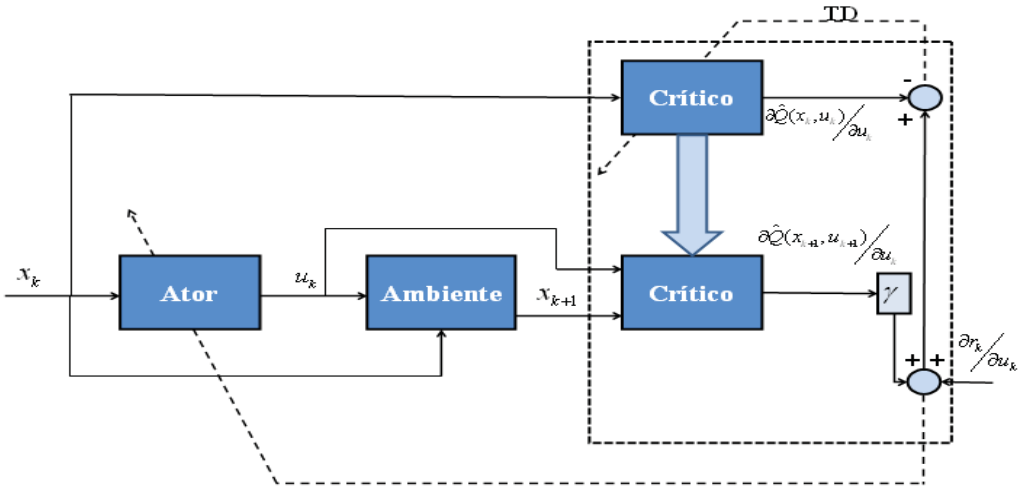

Figura 4.5: Estrutura de ADDHP.

Este método utiliza o mesmo treinamento do Crítico do DHP, porém recebe as derivadas necessárias para o treinamento do controlador diretamente da saída do Crítico. Por isso, AD-DHP necessita do modelo da planta para o treinamento do Crítico porém não para o treinamento do controlador.

Os resultados das análises feitas para os diferentes tipos de ADP já citadas podem ser encontradas na Tabela 4.1 (LENDARIS, 2009).

Tabela 4.1: Estruturas de ADP e a necessidade do modelo da planta para treinamento.

Estrutura de ADP	**Necessidade do Modelo para treinamento do**	
	Crítico	Controlador
HDP		X
AD-HDP		
DHP	X	X
AD-DHP	X	

4.4 Conclusão

Neste capítulo viu-se que TD e a Aproximação da Função Valor são tópicos importantes na implementação de uma ADP. Foi exposto a estrutura de AR *on-line*, para controle ótimo discreto . Para esta situação, os algoritmos de PI e VI do Capítulo 3, foram desenvolvidos de forma que não se necessita do conhecimento da dinâmica do sistema, ou seja, matrizes A_d e B_d. Para o DLQR, um aspecto importante na Aproximação da Função Valor é o produto de *Kronecker* e a vetorização.

As estruturas básicas de ADP propostas por Werbos foram abordadas para viabilizar o desenvolvimento dos esquemas Ator-Crítico. A importância do modelo Ator-Crítico de AR nestas estruturas de ADP pode ser percebido. No modelo HDP viu-se que ainda se necessita do modelo da planta para atualização da ação de controle, enquanto que sua variante, AD-HDP, torna-se um modelo totalmente livre da necessidade deste modelo.

Capítulo 5

SOLUÇÕES PARA O LQR DISCRETO UTILIZANDO ADP

Neste capítulo, algoritmos de ADP modelados por críticos adaptativos são utilizados na resolução do DLQR. Estes algoritmos são direcionados para resolução da equação de *Riccati* de forma *online*. Uma estrutura paramétrica é usada para aproximar a função de custo da política de controle corrente.

Dois esquemas são apresentados, HDP e AD-HDP. No algoritmo AD-HDP, mostra-se a independência do conhecimento da dinâmica do sistema para a aproximação do crítico ou da ação de controle, diferentemente do algoritmo HDP, em que necessita-se do conhecimento da dinâmica do sistema para atualizar a ação de controle.

Formulações sobre a influência do fator de desconto e do ruído na equação de HJB serão expostos e analisados.

5.1 Algoritmo HDP para o LQR discreto

Considerando o sistema da Eq.(3.21), a ação de controle da Eq.(3.22) e a Função Valor dada por Eq.(3.24), utilizando-se um custo inicial $V_0(x) \geq 0$ não necessariamente ótimo, pode-se encontrar um valor $V_j(x)$ a partir de $j = 0$ através do algoritmo VI da seguinte forma:

$$V_{j+1}(x) = \min \left\{ x_k^T Q x_k + u_k^T R u_k + \gamma V_j(x_{k+1}) \right\} \tag{5.1}$$

A Eq.(5.1) é uma relação de recorrência para solucionar o custo ótimo em avanço no tempo. Esta política gulosa (*greedy*) é denotada por $h_j(x_k) = u_k$ e V_{j+1} é dada por :

$$V_{j+1}(x) = x_k^T Q x_k + h_j^T(x_k) R h_j(x_k) + \gamma V_j(x_{k+1}) \tag{5.2}$$

Na aproximação HDP, a função $V_j(x)$, é geralmente difícil de se obter no sistema de malha fechada exceto em casos especiais. Entretanto, em geral, uma estrutura paramétrica $\widehat{V}(x, \bar{p}_j)$, é usada para aproximar o valor atual $V_j(x)$. Similarmente uma estrutura paramétrica é usada para obter uma representação de malha fechada para a ação de controle $\hat{h}(x, K)$ (AL-TAMIMI, *et al.*, 2007*b*). Sabendo-se que a Função Valor no estado x é quadrática, $V(x) = x^T P x$, a ação de controle $h(x)$ é linear, então uma escolha para uma estrutura paramétrica pode ser dada por

$$\widehat{V}(x, \bar{p}_j) = W_{V_j}^T \phi(x) = \bar{p}_j^T \bar{x} \tag{5.3}$$

$$\hat{h}(x, K_j) = -K_j^T x \tag{5.4}$$

sendo $\bar{x} = (x_1^2, ..., x_1 x_n, x_2^2, x_2 x_3, ..., x_{n-1} x_n, x_n^2)$, o vetor base polinomial do produto de *Kronecker* (BREWER, 1978), e $\bar{p} = vec(P)$, sendo $v(\cdot)$ uma função vetorial que atua sobre uma matriz $n \times n$ e tem como saída um vetor coluna $\frac{n(n+1)}{2} \times 1$. O vetor de saída $vec(\cdot)$ é construído pelo empilhamento das colunas da matriz quadrada em um único vetor coluna com os elementos fora da diagonal somados conforme $P_{ij} + P_{ji}$. As estruturas paramétricas (5.3) e (5.4) fornecem uma representação exata das funções em (5.2).

A ação de controle, K_j de (5.4) pode ser encontrada através da Eq.(5.5).

$$K_j = (R/\gamma + B_d^T P_j B_d)^{-1} B_d^T P_j A_d \tag{5.5}$$

Nota-se que na atualização da ação de controle, é necessário o conhecimento das matrizes A_d e B_d do modelo da planta (AL-TAMIMI, *et al.*, 2007*b*).

Depois de se determinar a ação de controle pela Eq.(5.5), substituindo-se em Eq.(5.2), tem-se:

$$d(x_k, \bar{p}_j) = x_k^T Q x_k + (-K_j^T x_k) R (-K_j x_k) + \gamma \bar{p}_j^T \bar{x}_{k+1} \tag{5.6}$$

que é a função de objetivo a ser alcançada a partir da estimação de $\widehat{V}(x, \bar{p}_{j+1})$ por LS para se encontrar $\bar{p}_{j+1}$.

$$\bar{p}_{j+1}^T \bar{x}_k = d(x_k, \bar{p}_j) \tag{5.7}$$

O parâmetro $\bar{p}_{j+1}$ é encontrado minimizando-se o erro entre o valor da função (5.6) e (5.7) através do LS em um conjunto compacto Ω dado por:

$$\bar{p}_{j+1} = \arg\min_{\bar{p}_{j+1}} \{ \int_{\Omega} \left| \bar{p}_{j+1}^T \bar{x} - d(x_k, \bar{p}_j) \right|^2 dx \} \tag{5.8}$$

5.1.1 Formulação *online* do algoritmo HDP para sistemas MIMO

O algoritmo pode ser implementado em tempo real coletando-se pontos de dados suficientes de $d(x_k, \bar{p}_j)$ pela Eq.(5.6) utilizando-se LS. Necessita-se então o conhecimento da informação dos estados x_k, x_{k+1} como a dinâmica envolvida no tempo, assim como da função de reforço $r(x_k, u_k)$. Como já mencionado, no algoritmo HDP, não é necessário o modelo da planta para treinamento do Crítico, porém é necessário para o cálculo da ação de controle (AL-TAMIMI, *et al.*, 2007*b*).

Para satisfazer a condição de excitação no problema de LS, o número de pontos coletados deve ser dado por:

$$N \geq n(n+1)/2 \tag{5.9}$$

sendo n o número de estados. Assim, após vários passos de tempo que são suficientes para garantir a condição de excitação, tem-se o problema dos LS batelada dado por:

$$\bar{p}_{j+1} = (XX^T)^{-1} XY \tag{5.10}$$

sendo

$$\begin{aligned} X &= \left[\begin{array}{cccc} \bar{x}|_{x_{k-N-1}} & \bar{x}|_{x_{k-N-2}} & ... & \bar{x}|_{x_{k-1}} \end{array} \right] \\ Y &= \left[\begin{array}{cccc} d(x_{k-N-1}, \bar{p}_j) & d(x_{k-N-2}, \bar{p}_j) & ... & d(x_{k-1}, \bar{p}_j) \end{array} \right]^T \end{aligned} \tag{5.11}$$

O desenvolvimento do algoritmo HDP, segue através das iterações entre as equações (5.5) e (5.10).

5.1.2 Influência do fator de desconto γ

Na aplicação de ADP, um aspecto importante é como o fator de desconto influencia no processo de convergência do algoritmo. Através dos algoritmos propostos, percebe-se que as saídas do Crítico são utilizadas para treiná-lo (TD). Então, nos estágios iniciais do processo, a componente da Eq.(3.7), tem uma contribuição ao Crítico equivalente a um "ruído"(SI *et al.*, 2004).

Para a estrutura HDP, a escolha do fator de desconto é fundamental para a convergência do Crítico. Uma prática comum é começar o treinamento com valores baixo de γ e depois aumenta-lo incrementalmente. Valores baixo de γ representam um fator de desconto que anula os valores do termo do lado direito da recursão de *Bellman*. Para o Crítico, isso resulta em uma aproximação somente de $r(x_k, u_k)$. Incrementar progressivamente o valor de γ, faz com que o Crítico aprenda como os custos primários acumulam ao longo do tempo para formar a função de valor $V_h(x_k)$.

5.2 Algoritmo AD-HDP para o LQR discreto

Como já mencionado, infelizmente, no algoritmo de HDP, necessita-se do conhecimento das matrizes A_d e B_d. Isto, porque na minimização, como segue a Eq.(3.33), deve-se diferenciar em relação ao controle u_k como mostra a Eq.(5.12).

$$\frac{\partial x_{k+1}}{\partial u_k} = g(x_k) \tag{5.12}$$

Entretanto é exigido do sistema a matriz de entrada $g(x_k)$. Para se evitar a necessidade de qualquer conhecimento do sistema, deve-se fornecer uma alternativa para se obter derivadas parciais em relação à entrada de controle.

Neste seção, o princípio da otimalidade de *Bellman* é formulado usando o conceito de Função-Q (*Q-function*) (WERBOS, 1989), (BRADTKE, 1993) ao invés da Função Valor padrão. A equação de *Bellman* nos permite calcular o valor usando qualquer política de controle admissível. Entretanto, pode-se definir a Função-Q associada com a política $u = h(x)$ da seguinte forma:

$$Q_h(x_k, u_k) = r(x_k, u_k) + \gamma V_h(x_{k+1}) \tag{5.13}$$

Percebe-se que a Função-Q, fica em função do estado x_k e do controle u_k no

tempo k. A Função-Q ótima é dada por:

$$Q^*(x_k, u_k) = r(x_k, u_k) + \gamma V^*(x_{k+1}) \tag{5.14}$$

Em termos de Q^*, pode-se escrever a equação de otimalidade de *Bellman* da seguinte forma:

$$V^*(x_k) = \min_{u_k}(Q^*(x_k, u_k)) \tag{5.15}$$

e o controle ótimo da forma:

$$h^*(x_k) = \arg\min_{u_k}(Q^*(x_k, u_k)) \tag{5.16}$$

O valor mínimo então pode ser obtido por:

$$\frac{\partial Q^*(x_k, u_k)}{\partial u_k} = 0 \tag{5.17}$$

Ao contrário da Eq.(5.12), assumindo-se que se conheça a Função-Q para todo (x_k, u_k), não necessita-se do conhecimento da dinâmica do sistema. No algoritmo de HDP, deve-se aprender e armazenar o valor ótimo para todos os estados x_k possíveis. Em contraste, na aprendizagem pela Função-Q, deve-se armazenar o valor ótimo da Função-Q para todos os valores de (x_k, u_k) possíveis, isto é, para toda ação de controle realizada a cada estado possível. Assim, tem-se muito mais informação a ser processada.

Para determinar-se a equação de ponto fixo para a Função-Q, nota-se que:

$$Q_h(x_k, h(x_k)) = V_h(x_k) \tag{5.18}$$

Então, a "equação de *Bellman*"para o Q é:

$$Q_h(x_k, u_k) = r(x_k, u_k) + \gamma Q_h(x_{k+1}, h(x_{k+1})) \tag{5.19}$$

O valor Q ótimo é:

$$Q^*(x_k, u_k) = r(x_k, u_k) + \gamma Q^*(x_{k+1}, h^*(x_{k+1})) \tag{5.20}$$

A Eq.(5.19) é a equação de ponto fixo ou "equação de *Bellman*"para a Função-Q. A partir dela pode-se aplicar qualquer técnica de AR para solução, incluindo

PI e VI.

5.2.1 Caracterização da configuração Função-Q para o LQR discreto

A partir da formulação já proposta usando Função-Q, agora leva-se a uma aplicação para o caso do DLQR. Partindo-se da Eq.(5.13) tem-se:

$$Q_K(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + \gamma[x_{k+1}^T P x_{k+1}] = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{5.21}$$

Sendo P a solução da equação de *Lyapunov* para um determinado ganho de controle K e uma matriz H associada a solução de *Lyapunov*. Desenvolvendo-se a Eq.(5.21) tem-se:

$$\begin{aligned} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} &= x_k^T Q x_k + u_k^T R u_k + \gamma x_{k+1}^T P x_{k+1} \\ &= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ &+ \gamma \left(\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} A_d^T \\ B_d^T \end{bmatrix} P \begin{bmatrix} A_d \\ B_d \end{bmatrix}^T \begin{bmatrix} x_k \\ u_k \end{bmatrix} \right) \end{aligned} \tag{5.22}$$

Desenvolvendo-se a Eq.(5.22) para o lado direito de (5.21) tem-se que:

$$Q_K(x_k, u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + \gamma A_d^T P A_d & \gamma A_d^T P B_d \\ \gamma B_d^T P A_d & R + \gamma B_d^T P B_d \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{5.23}$$

A Eq.(5.23) é a Função-Q , quadrática em (x_k, u_k), para o LQR discreto. A matriz H então é dada por:

$$H = \begin{bmatrix} Q + \gamma A_d^T P A_d & \gamma A_d^T P B_d \\ \gamma B_d^T P A_d & R + \gamma B_d^T P B_d \end{bmatrix} \tag{5.24}$$

Através da Eq.(5.20), para o caso do LQR discreto, pode-se obter:

$$\begin{aligned}\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T H \begin{bmatrix} x_k \\ u_k \end{bmatrix} &= r(x_k, u_k) + \gamma Q^*(x_{k+1}, u_{k+1}) \\ &= x_k^T Q x_k + u_k^T R u_k + \gamma \begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix}^T H \begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix} \end{aligned} \quad (5.25)$$

Atribuindo-se $h(x_k) = u_k = -Kx_k$:

$$\begin{aligned} &= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ &+\gamma \begin{bmatrix} A_d x_k + B_d u_k \\ -K(A_d x_k + B_d u_k) \end{bmatrix}^T H \begin{bmatrix} A_d x_k + B_d u_k \\ -K(A_d x_k + B_d u_k) \end{bmatrix} \\ &= \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\ &+\gamma \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} A_d & B_d \\ -KA_d & -KB_d^T \end{bmatrix}^T H \begin{bmatrix} A_d & B_d \\ -KA_d & -KB_d \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \end{aligned} \quad (5.26)$$

Então H pode ser escrito da seguinte forma (YU e ZHONG-PING, 2010):

$$\begin{aligned}\begin{bmatrix} H_{xx} & H_{xu} \\ H_{ux} & H_{uu} \end{bmatrix} &= G + \gamma \begin{bmatrix} A_d & B_d \\ -KA_d & -KB_d^T \end{bmatrix}^T H \begin{bmatrix} A_d & B_d \\ -KA_d & -KB_d \end{bmatrix} \\ &= G + \gamma \begin{bmatrix} A_d^T \\ B_d^T \end{bmatrix} \begin{bmatrix} I & -K^T \end{bmatrix} H \begin{bmatrix} I \\ -K \end{bmatrix} \begin{bmatrix} A_d & B_d \end{bmatrix} \end{aligned} \quad (5.27)$$

sendo

$$\begin{aligned} G &= \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \\ P &= \begin{bmatrix} I & -K^T \end{bmatrix} H \begin{bmatrix} I \\ -K \end{bmatrix} \end{aligned} \quad (5.28)$$

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

$$\begin{aligned} V^*(x_k) &= \min_{u_k} Q^*(x_k, u_k) \\ &= \begin{bmatrix} x_k^T & u_k^T \end{bmatrix} H \begin{bmatrix} x_k^T & u_k^T \end{bmatrix}^T \end{aligned} \tag{5.29}$$

Para se minimizar em função de u_k, aplica-se a Eq. (5.17) e obtém-se:

$$\begin{aligned} 0 &= H_{ux}x_k + H_{uu}u_k \\ u_k &= (H_{uu})^{-1}H_{ux}x_k \end{aligned} \tag{5.30}$$

Como $u_k = -Kx_k$, então tem-se que:

$$K = (H_{uu})^{-1}H_{ux} \tag{5.31}$$

A Eq.(5.23) pode ser utilizada para se obter o ganho da Eq.(5.31) em função da matriz P. Nota-se que o ganho K do controlador, Eq.(5.31), fica dependente somente da matriz H, não necessitando do conhecimento das matrizes A_d e B_d (LEE *et al.*, 2009) (AL-TAMIMI, *et al.*, 2007*a*).

5.2.2 Formulação *online* do algoritmo AD-HDP para sistemas MIMO

O algoritmo *Q-Learning* tem sido aplicado para resolver o problema do LQR discreto (LANDELIUS e KNUTSSON, 1996). No *Q-Learning*, uma estrutura paramétrica é usada para aproximar a Função-Q da política de controle atual. (AL-TAMIMI, *et al.*, 2007*c*).

No *Q-Learning*, inicia-se com uma Função-Q inicial $Q_0(x, u) \geq 0$ não necessariamente ótima, e depois encontra-se $Q_1(x, u)$ resolvendo-se a Eq.(5.32) com $i = 0$ (AL-TAMIMI *et al.*, 2007*c*).

$$\begin{aligned} Q_{j+1}(x_k, u_k) &= \left\{ x_k^T Q x_k + u_k^T R u_k + \min_{u_k} \gamma Q_j(x_{k+1}, u_{k+1}) \right\}, \\ &= \left\{ x_k^T Q x_k + u_k^T R u_k + V_j(x_{k+1}) \right\} \\ &= \left\{ x_k^T Q x_k + u_k^T R u_k + \gamma V_j(A_d x_k + B_d u_k) \right\} \end{aligned} \tag{5.32}$$

Consequentemente tem-se que:

$$\min_{u_k} Q_{j+1}(x_k, u_k) = \min_{u_k} \left[\begin{array}{cc} x_k^T & u_k^T \end{array} \right] H_{j+1} \left[\begin{array}{cc} x_k^T & u_k^T \end{array} \right]^T \tag{5.33}$$

De acordo com a Eq.(5.31), a política de atualização de realimentação é dada por:

$$K_j = (H_{uu}^j)^{-1} H_{ux}^j \tag{5.34}$$

sendo

$$h_j(x_k) = -K_j x_k \tag{5.35}$$

Este é o método de política de iteração gulosa (*greedy*) que é baseado na Função-Q (AL-TAMIMI *et al.*, 2007*c*). Então:

$$\begin{aligned} \min_{u_k} Q_{j+1}(x_k, u_k) &= V_{j+1}(x_k) \\ &= \min_{u_k} \left\{ x_k^T Q x_k + u_k^T R u_k + \gamma V_j(A_d x_k + B_d u_k) \right\}, \end{aligned} \tag{5.36}$$

Para obter-se a solução em avanço no tempo, deve-se substituir a Eq.(5.35) na Eq.(5.32), obtendo-se a seguinte relação de recorrência.

$$\begin{aligned} Q_{j+1}(x_k, u_j(x_k)) &= x_k^T Q x_k + h_j^T(x_k) R h_j(x_k) \\ &+ \gamma \left[\begin{array}{cc} x_{k+1}^T & h_j^T(x_{k+1}) \end{array} \right] H_j \left[\begin{array}{cc} x_{k+1}^T & h_j^T(x_{k+1}) \end{array} \right]^T \end{aligned} \tag{5.37}$$

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

Em um sistema realimentado, na aproximação usando ADHDP, a Função-Q é geralmente difícil de se obter. Entretanto, uma estrutura paramétrica pode ser utilizada para se aproximar $Q_j(x, u)$ (AL-TAMIMI *et al.*, 2007*c*). Similarmente, uma estrutura paramétrica é utilizada para aproximar a ação de controle $\hat{u}_j(x, K)$ (AL-TAMIMI, *et al.*, 2007*a*). Então tem-se que:

$$\hat{h}_j(x) = -K_j x \tag{5.38}$$

$$\widehat{Q}(\bar{z}, \bar{H}_j) = z^T H_j z = \bar{H}_j^T \bar{z}_k \tag{5.39}$$

sendo $z = \begin{bmatrix} x^T & u^T \end{bmatrix}^T$ $z \in R^{n+m=q}$, $\bar{z} = (z_1^2, ..., z_1 z_q, z_2^2, z_2 z_3, ..., z_{q-1} z_q, z_q^2)$ é o vetor base polinomial do produto de *Kronecker*, e $\bar{H} = vec(H)$ com $vec(\cdot)$ send o vetor função que atua sobre a matriz $q \times q$ e dá como saída o vetor coluna $\frac{q(q+1)}{2} \times 1$. A saída de $vec(\cdot)$ é construída com o empilhamento das colunas da matriz quadrada em um único vetor coluna com os elementos fora da diagonal somados como $H_{ij} + H_{ji}$. As estruturas paramétricas (5.38) e (5.39) fornecem uma representação da Eq.(5.37).

A aproximação da Eq.(5.37), utilizando-se as estruturas paramétricas propostas , é dada por:

$$d(z_k(x_k), H_j) = x_k^T Q x_k + \hat{h}_j^T(x_k) R \hat{h}_j(x_k) + \gamma Q_j(x_{k+1}, \hat{h}_j(x_{k+1})) \tag{5.40}$$

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

$$\bar{H}_{j+1}^T \bar{z}(x_k) = d(\bar{z}(x_k), \bar{H}_j) \tag{5.41}$$

O parâmetro $\bar{H}_{j+1}$ é encontrado pela minimização do erro entre (5.39) e (5.40) através do LS em um dado conjunto compacto Ω.

$$\bar{H}_{j+1} = \arg\min_{\bar{H}_{j+1}} \left\{ \int_{\Omega} \left| \bar{H}_{j+1}^T \bar{z}(x_k) - d(\bar{z}(x_k), \bar{H}_j) \right|^2 dx_k \right\} \tag{5.42}$$

Resolvendo-se o problema do LS obtém-se;

$$\bar{H}_{j+1} = \left(\int_{\Omega} \bar{z}(x_k) \bar{z}(x_k)^T dz \right)^{-1} \int_{\Omega} \bar{z}(x_k) d(\bar{z}(x_k), \bar{H}_j) dx \tag{5.43}$$

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

$$
\begin{aligned}
z(x_k) &= \begin{bmatrix} x_k^T & (\hat{h}_j(x_k)) \end{bmatrix}^T && (5.44)\\
&= \begin{bmatrix} x_k^T & (-K_j x_k) \end{bmatrix}^T && (5.45)\\
&= \left(x_k^T \begin{bmatrix} I & -K_j^T \end{bmatrix}^T \right)^T && (5.46)
\end{aligned}
$$

Através da Eq.(5.46) pode-se notar a dependencia linear de x_k em $\hat{h}_j$. Portanto tem-se que:

$$\int_{\Omega} \bar{z}(x_k)\bar{z}(x_k)^T dx_k \tag{5.47}$$

não possui inversa o que torna o problema sem solução por LS. Para contornar este problema. adiciona-se um ruído branco a entrada. Então, tem-se:

$$\hat{h}_{ej}(x_k) = -K_j x_k + n_k \tag{5.48}$$

sendo $n(0, \sigma)$ com variância σ^2. Portanto $z(x_k)$ torna-se:

$$z(x_k) = \begin{bmatrix} x_k \\ \hat{h}_{ej}(x_k) \end{bmatrix} = \begin{bmatrix} x_k \\ -K_j x_k + n_k \end{bmatrix} = \begin{bmatrix} x_k \\ -K_j x_k \end{bmatrix} + \begin{bmatrix} 0 \\ n_k \end{bmatrix} \tag{5.49}$$

Modificando-se a Eq.(5.40) tem-se:

$$d(z_k(x_k), H_j) = x_k^T Q x_k + \hat{h}_{ej}(x_k)^T R \hat{h}_{ej}(x_k) + \gamma Q_j(x_{k+1}, \hat{h}_j(x_{k+1})) \tag{5.50}$$

A inversa da matriz dada por:

$$\bar{H}_{j+1} = (ZZ^T)^{-1} ZY \tag{5.51}$$

é garantida pela condição de excitação. Então tem-se:

$$\begin{aligned} d(z_k(x_k), H_j) &= x_k^T Q x_k + \hat{h}_{ej}(x_k)^T R \hat{h}_{ej}(x_k) \\ &+ \gamma \begin{bmatrix} x_{k+1}^T & (-K_j x_{k+1})^T \end{bmatrix} H_j \begin{bmatrix} x_{k+1}^T & (-K_j x_{k+1})^T \end{bmatrix}^T \end{aligned} \quad (5.52)$$

sendo

$$x_{k+1} = A_d x_k + B_d \hat{h}_{ej}(x_k) \quad (5.53)$$

A solução por LS, Eq.(5.51), pode ser resolvida em tempo real através de medições suficientes gerados por $d(z_k, H_j)$ na Eq.(5.50). Necessita-se então do conhecimento dos estados x_k, x_{k+1}, da função de reforço $r(z_k) = x_k^T Q x_k + \hat{h}_{ej}(x_k)^T R \hat{h}_{ej}(x_k)$ e Q_j. Portando, o algoritmo *Q-Learning*, não necessita do modelo do sistema para atualização do Crítico ou da Ação de Controle, sendo assim, um modelo de ajuste livre (*Model-Free Tunning*) (AL-TAMIMI *et al.*, 2007*c*).

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

$$\begin{aligned} Z &= \begin{bmatrix} \bar{z}(x_{k-N-1}) & \bar{z}(x_{k-N-2}) & \dots & \bar{z}(x_{k-1}) \end{bmatrix} \\ Y &= \begin{bmatrix} d(\bar{z}(x_{k-N-1}), \bar{H}_j) & d(\bar{z}(x_{k-N-2}), \bar{H}_j) & \dots & d(\bar{z}(x_{k-1}) , \bar{H}_j) \end{bmatrix}^T \end{aligned} \quad (5.54)$$

No desenvolvimento do algoritmo AD-HDP, O parâmetro da ação de controle é atualizado de acordo com a Eq.(5.34).

5.2.3 Influência do ruído de controle e fator de desconto

Para analisar a influência do ruído na entrada de controle, considere a equação de *Bellman*, Eq.(3.27), com a entrada de controle dada por $\hat{h}_e(x_k) = u_k + n_k$, sendo n_k o ruído branco adicionado. Então tem-se:

$$x_k^T \hat{P} x_k = x_k^T Y x_k + n_k^T R n_k + u_k^T R n_k + n_k^T R u_k + \gamma [\hat{x}_{k+1}^T \hat{P} \hat{x}_{k+1}] \quad (5.55)$$

então

$$\begin{aligned} x_k^T \hat{P} x_k &= x_k^T Y x_k + n_k^T R n_k + u_k^T R n_k + n_k^T R u_k \\ &+ \gamma[(A_d x_k + B_d \hat{h}_e(x_k))^T \hat{P}(A_d x_k + B_d \hat{h}_e(x_k))] \\ &= x_k^T Y x_k + n_k^T R n_k + u_k^T R n_k + n_k^T R u_k \\ &+ \gamma[(A_d x_k + B_d(u_k + n_k))^T \hat{P}(A_d x_k + B_d(u_k + n_k))] \end{aligned} \tag{5.56}$$

Desenvolvendo-se a Eq.(5.56) tem-se que:

$$\begin{aligned} x_k^T \hat{P} x_k &= x_k^T Y x_k + n_k^T R n_k + u_k^T R n_k + n_k^T R u_k \\ &+ \gamma[(A_d x_k + B_d u_k)^T \hat{P}(A_d x_k + B_d u_k) \\ &+ (A_d x_k + B_d u_k)^T \hat{P} B_d n_k + (B_d n_k)^T \hat{P}(A_d x_k + B_d u_k) \\ &+ (B_d n_k)^T \hat{P}(B_d n_k)] \end{aligned} \tag{5.57}$$

Utilizando-se $tr\{AB\} = tr\{BA\}$ e assumindo-se que o ruído branco é independente de u_k e x_k de modo que $E\left\{R u_k n_k^T\right\} = 0$ e $E\left\{\hat{P} B_d n_k (A_d x_k + B_d u_k)^T\right\} = 0$, então, calculando-se repetidas ações de controle com diferentes sequências de ruídos tem-se que:

$$\begin{aligned} x_k^T \hat{P} x_k &= x_k^T Q x + u_k^T R u_k + \gamma[(A_d x_k + B_d u_k)^T \hat{P}(A_d x_k + B_d u_k)] \\ &+ n_k^T (R + \gamma B_d^T \hat{P} B_d) n_k \end{aligned} \tag{5.58}$$

como $u_k = -K x_k$ então tem-se:

$$\begin{aligned} x_k^T \hat{P} x_k &= x_k^T Q x_k + (-K x_k)^T R(-K x_k) + \gamma[(A_c x_k)^T \hat{P}(A_c x_k)] \\ &+ n_k^T (R + \gamma B_d^T \hat{P} B_d) n_k \\ &= x_k^T Q x_k + x_k^T (K^T R K) x_k + x_k^T (\gamma A_c^T \hat{P} A_c) x_k \\ &+ n_k^T (R + \gamma B_d^T \hat{P} B_d) n_k \end{aligned} \tag{5.59}$$

Sendo $A_c = (A_d - B_d K)$. Pode-se Escrever a Eq.(5.59) da seguinte forma:

$$x_k^T[\hat{P} - Q - K^T RK - \gamma A_c^T \hat{P} A_c]x_k = n_k^T(R + \gamma B_d^T \hat{P} B_d)n_k \tag{5.60}$$

A Eq.(5.58) é a equação de *Bellman* mais um termo dependente do ruído de controle adicionado. A influência do fator γ é percebida no termo $(R+\gamma B_d^T \hat{P} B_d)$. Através da Eq.(3.3), para um $\gamma < 1$, pode-se verificar a influência do fator de desconto na diminuição do efeito do ruído. Então a Eq.(3.3) modificada é dada por:

$$\begin{aligned} V_{h_e}(x_k) &= \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, \hat{h}_e(x_i)) \\ &= \sum_{i=0}^{\infty} \gamma^{i} r(x_{k+i}, \hat{h}_e(x_{k+i})) \end{aligned} \tag{5.61}$$

Desenvolvendo-se a Eq.(5.61) tem-se que:

$$V_{h_e}(x_k) = r(x_k, \hat{h}_e(x_k)) + \gamma \sum_{i=1}^{\infty} \gamma^{i-1} r(x_{k+i}, \hat{h}_e(x_{k+i})) \tag{5.62}$$

Particularizando-se a Eq.(5.62) para $r(x_k, \hat{h}_e(x_k)) = x_k^T Y x_k + n_k^T R n_k$, sendo $Y = (Q + K^T RK)$. Então:

$$\begin{aligned} V_{h_e}(x_k) &= x_k^T Q x + u_k^T R u_k + n_k^T R n_k \\ &+ \gamma \sum_{i=1}^{\infty} \gamma^{i-1} (x_{k+i}^T Q x_{k+i} + u_{k+i}^T R u_{k+i} + n_{k+i}^T R n_{k+i}) \end{aligned} \tag{5.63}$$

Então

$$\begin{aligned} V_{h_e}(x_k) &= x_k^T Y x_k + n_k^T R n_k \\ &+ \gamma \sum_{i=1}^{\infty} \gamma^{i-1} [(x_{k+i})^T (Y)(x_{k+i}) + n_{k+i}^T R n_{k+i}] \end{aligned} \tag{5.64}$$

Sabendo-se que $u_k = -Kx_k$ e que $A_c = (A_d - B_d K)$ então tem-se a seguinte relação:

$$\begin{aligned} V_{h_e}(x_k) &= x_k^T Y x_k + n_k^T R n_k \\ &+ \gamma \sum_{i=1}^{\infty} \gamma^{i-1} [x_k^T (A_c^T)_i Y (A_c)_i x_k + n_k^T B_d^T Y B_d n_k + n_{k+i}^T R n_{k+i}] \end{aligned} \quad (5.65)$$

sendo

$$V_{h_e}(x_{k+1}) = \sum_{i=1}^{\infty} \gamma^{i-1} [x_k^T (A_c^T)_i Y (A_c)_i x_k + n_k^T B_d^T Y B_d n_k + n_{k+i}^T R n_{k+i}] \quad (5.66)$$

sabendo-se que $\hat{V}_{h_e}(x_{k+1}) = \hat{x}_{k+1}^T \hat{P} \hat{x}_{k+1}$

$$\hat{x}_{k+1}^T \hat{P} \hat{x}_{k+1} = \sum_{i=1}^{\infty} \gamma^{i-1} [x_k^T (A_c^T)_i Y (A_c)_i x_k + n_k^T B_d^T Y B_d n_k + n_{k+i}^T R n_{k+i}] \quad (5.67)$$

Então, para o caso particular, a Eq.(5.62) fica:

$$\begin{aligned} V_{h_e}(x_k) &= \sum_{i=0}^{\infty} \gamma^{i} [x_k^T (A_c^T)_i Y (A_c)_i x_k + n_k^T B_d^T Y B_d n_k + n_{k+i}^T R n_{k+i}] \\ &= \sum_{i=k}^{\infty} \gamma^{i-k} [x_k^T (A_c^T)_{i-k} Y (A_c)_{i-k} x_k + n_k^T B_d^T Y B_d n_k + n_i^T R n_i] \end{aligned} \quad (5.68)$$

Generalizando-se

$$\begin{aligned} V_{h_e}(x_k) &= \sum_{i=k}^{\infty} \gamma^{i-k} (x_i^T Y x_i + n_i^T R n_i) \\ &= \sum_{i=0}^{\infty} \gamma^{i} (x_{k+i}^T Y x_{k+i} + n_{k+i}^T R n_{k+i}) \end{aligned} \quad (5.69)$$

Nota-se agora, que o fator de desconto γ tem importante influência no decaimento do efeito da ação do ruído n_k. A escolha de γ, para um sistema com ação de controle dada por $\hat{h}_e(x_k) = u_k + n_k$, tem influência direta na convergência do algoritmo do sistema, para remover os efeitos de polarização dinâmica não modelada.

5.3 Conclusão

Neste capítulo duas técnicas de ADP foram introduzidas para resolução do DLQR de forma *online*. Estas duas técnicas são HDP e AD-HDP. No HDP, necessita-se de uma quantidade menor para convergência por LS do que o algoritmo AD-HDP. Isto se deve ao fato de que na AD-HDP, trabalha-se também com a ação de controle na construção da matriz dos regressores.

Observa-se que na determinação do ganho K do controlador, somente no algoritmo HDP que se necessita do conhecimento da dinâmica do sistema. No algoritmo AD-HDP, a determinação fica em virtude somente da matriz H.

Apesar de alguns pesquisadores colocarem o algoritmo AD-HDP como sendo um modelo totalmente livre do conhecimento da dinâmica da planta, percebe-se que algum conhecimento é necessário para saber-se a dimensão da matriz H, ou seja, os modelos de ADP, de certa forma, dependem do conhecimento da dinâmica do sistema.

A escolha do fator de desconto γ exerce forte influência na convergência dos algoritmos. O ruído n_k adcionado na ação de controle do algoritmo AD-HDP, pode ser minimizado pelo fator de desconto γ conforme foi mostrado através da equação de HJB.

Capítulo 6

AVALIAÇÃO DE DESEMPENHO DOS ALGORITMOS DE AR E ADP

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

Os resultados são organizados em:

- Programação Dinâmica.
- Aprendizagem por Reforço por PI e VI.
- ADP - Implementações por HDP e AD-HDP.

6.1 Modelo do sistema dinâmico

O sistema de avião F-16, extraido do livro (STEVENS e LEWIS, 1992) é utilizado como sistema base para avaliação o projeto dos controladores. O modelo contínuo é dado por:

$$\dot{x}(t) = \begin{bmatrix} -1.10188 & 0.90528 & -0.00212 \\ 4.0639 & -0.7013 & -0.16919 \\ 0 & 0 & -10 \end{bmatrix} x(t) + \begin{bmatrix} 0 & 0 \\ 0 & 10 \\ 10 & 0 \end{bmatrix} u(t) \qquad (6.1)$$

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

A discretização do modelo é obtida pelo método padrão do segurador de ordem zero (*zero order hold* - zoh) com intervalo de amostra de 0.1s. O modelo discreto então fica:

$$x_{k+1} = \begin{bmatrix} 0.9124 & 0.0829 & -0.0007 \\ 0.3724 & 0.9428 & -0.0103 \\ 0 & 0 & 0.3679 \end{bmatrix} x_k + \begin{bmatrix} -0.0003 & 0.0427 \\ -0.0061 & 0.9682 \\ 0.6321 & 0 \end{bmatrix} u_k \qquad (6.2)$$

O modelo da Eq.(6.2) será utilizado para o projeto do controlador ótimo pelas metodologias propostas.

6.2 Convergência QR

Aspectos de estabilidade na implementação de uma ADP num sistema realimentado tem sido estudado durante os anos (BALAKRISHNAN *et al.*, 2008). A análise de convergência QR consiste na avaliação dos autovalores das matrizes Q e R e suas relações com a alocação de autovalores de sistemas MIMO no plano Z por meio dos controladores ótimos. Os resultados são apresentados na forma de tabelas montadas seguindo uma heurística que é estabelecida a partir das Equações (2.64) e (2.65) do Capítulo 2 (FONSECA NETO e LOPES 2011). O processo iterativo para variações sistemáticas nas matrizes Q e R da função de custo seguem um padrão de crescimento da matriz Q, enquanto que a matriz R é uma matriz identidade durante todo o processo de solução.

6.3 Resultados do algoritmo de PD para o LQR discreto

Logo a seguir será analisado os resultados obtidos com o algoritmo 1 de PD (Programação Dinâmica) desenvolvido no Capítulo 2 para o DLQR do sistema MIMO de terceira ordem. São analisados aspectos de convergência do algoritmo em relação a número de iterações e a relação dos valores das matrizes Q e R com alocação de autovalores no plano Z.

6.3.1 Implementação *offline* do algoritmo de PD para sistemas MIMO

Considerando-se as matrizes A_d e B_d do sistema do F-16 já exposto e as matrizes Q e R como identidades para fins de referência, tem-se que a solução de *Riccati* pelo método de *Schur* é dada por:

$$P_{schur} = \begin{bmatrix} 5.3947 & 0.7427 & -0.0078 \\ 0.7427 & 1.6203 & -0.0067 \\ -0.0078 & -0.0067 & 1.1037 \end{bmatrix} \tag{6.3}$$

Consequentemente a política ótima é dada por:

$$K_{schur} = \begin{bmatrix} -0.0050 & -0.0039 & 0.1782 \\ 0.5644 & 0.6129 & -0.0066 \end{bmatrix} \tag{6.4}$$

Para os mesmos valores de A_d, B_d, Q e R, pode-se encontrar a mesma solução através de PD para um número N de iterações. Os parâmetros para os cálculos da matriz P de *Riccati* e do ganho K do controlador são as Eq.(2.56) e Eq.(2.54) respectivamente.

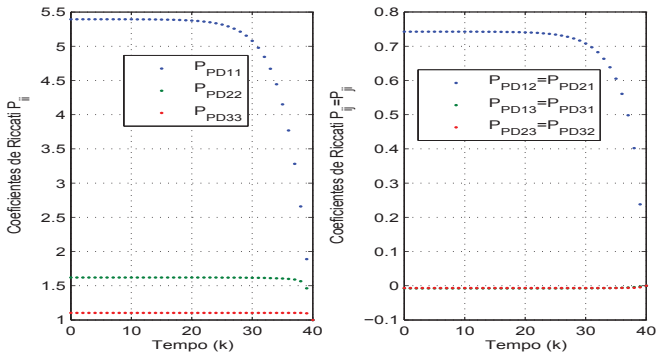

Figura 6.1: Convergência dos coeficientes da matriz P de *Riccati* por Programação Dinâmica

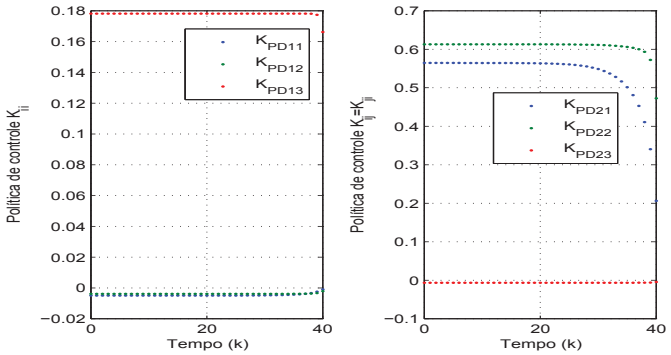

Figura 6.2: Convergência da política ótima K por Programação Dinâmica

Aplicando-se a PD para uma condição final S sendo a identidade, então consegue-se a convergência $P_{PD} = P_{Schur}$ e $K_{PD} = K_{Schur}$ após $N = 40$ iterações como mostra a Figuras 6.1 e 6.2. A trajetória ótima para cada estado x_k e a ação de controle u_k podem ser verificadas nas Figuras 6.3 e 6.4. Percebe-se que pelo método de Programação Dinâmica, os estados alcançam o estado de equilíbrio $x_k = 0$.

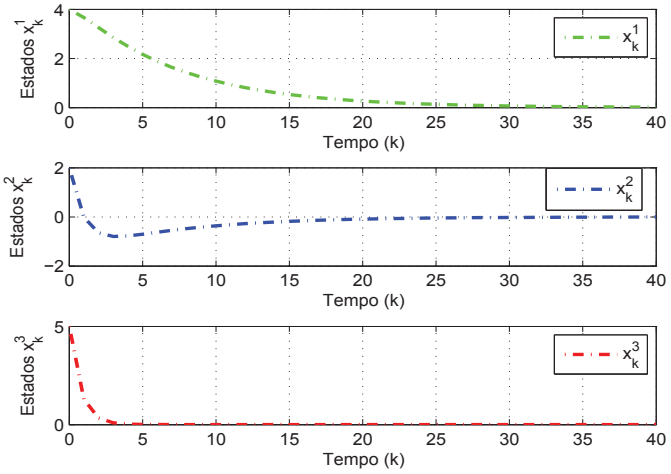

Figura 6.3: Trajetória dos Estados x_k por Programação Dinâmica

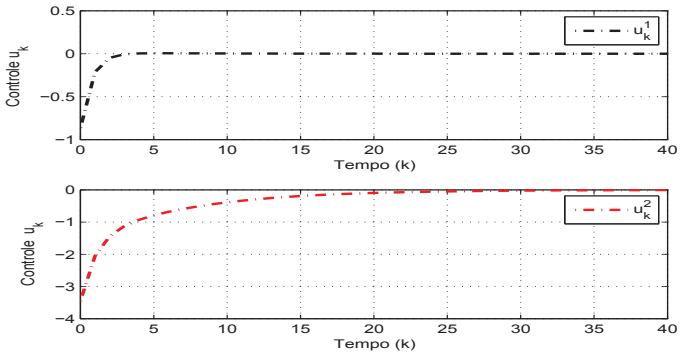

Figura 6.4: Ação de Controle u_k por Programação Dinâmica

A evolução dos autovalores σ a cada iteração pode ser verificada na Figura 6.5. Para cada ganho K obtido por cada iteração, gera um autovalor que aloca o sistema em determinado local no plano Z complexo.

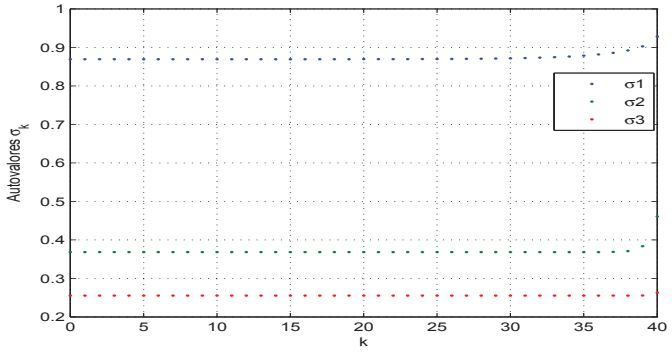

Figura 6.5: Autovalores σ a cada iteração por Programação Dinâmica

Ressalta-se que a solução por Programação Dinâmica é um método *offline* pois necessita do conhecimento da dinâmica do sistema. Outro empecilho é o fato da necessidade de se já conhecer o custo final do sistema, tendo em vista que a solução se da por retrocesso no tempo.

A análise da convergência, em relação a heurística da escolha dos valores de Q e R pode ser analisada na Tabela 6.1. Os números na coluna $Q(q_i)$ representam os valores numéricos das matrizes $Q(q_i) = 10^{q_i} I_{3\times3}$. O mesmo raciocínio para as matrizes R, $R(r_i) = 10^{q_i} I_{2\times2}$, sendo que para todos os casos tem-se $r_i = 0$ para $\forall\ i$ (FONSECA NETO e LOPES 2011). Para esta situação tem-se que $q_i = \{2, 1, 0, 1, -2\}$.

Tabela 6.1: DLQR-PD e Variações da Matriz Q.

q_i	r_i	N	Autovalores σ
2	0	57	0.8674, 0.0097 e 0.0090
1	0	49	0.8676, 0.0826 e 0.0721
0	0	40	0.8693, 0.3683 e 0.2553
-1	0	37	0.8808, 0.6584 e 0.3517
-2	0	33	0.9001, 0.7401 e 0.3662

Observou-se que a medida que se varia os valores de q_i para valores acima da referência, tem-se um aumento do número de iterações necessárias para convergência do algoritmo. Porém, pode-se verificar que dois dos três autovalores tem um deslocamento significativo em direção a origem no plano complexo.

6.4 Resultados de AR para o LQR discreto

Logo a seguir será analisado os resultados obtidos com os algoritmos 5 e 6 de Política de Iteração (PI) e Valor de Iteração (VI), respectivamente, para o DLQR, do sistema MIMO de terceira ordem, previamente desenvolvido no Capítulo 3. Serão analisados aspectos de convergência dos algoritmos em relação a número de iterações e a relação dos valores das matrizes Q e R com alocação de autovalores no plano Z. Como já visto, tais técnicas, diferentemente da Programação Dinâmica, tem o enfoque em avanço no tempo. Assim, o que as distingue, é somente a necessidade, ou não, de uma política inicial admissível.

6.4.1 Implementação *offline* do algoritmo PI para sistemas MIMO

Para resolução por PI, considera-se as matrizes A_d e B_d do sistema do F-16 já exposto e as matrizes Q e R como identidades para fins de referência. Para solução pelo método PI, necessita-se na inicialização de uma política inicial admissível, então para tal, tem-se $P_0 = I_{3\times3}$. A matriz P de *Riccati* é determinada pela Eq.(3.39), enquanto o ganho do controlador é determinado pela Eq.(3.40). Foi

adotado como fator de desconto $\gamma = 1$. Outros valores podem ser adotados, tendo em vista que $0 < \gamma \leq 1$.

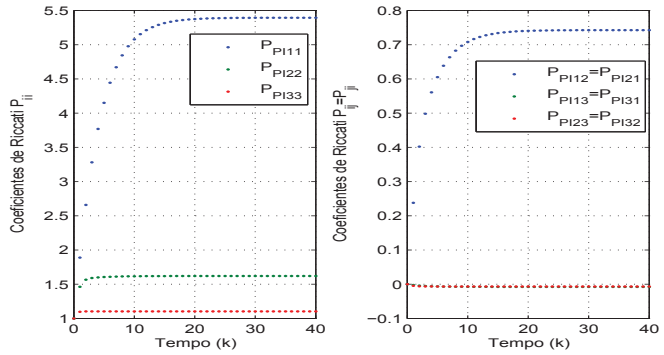

Figura 6.6: Convergência dos coeficientes da matriz P de *Riccati* por PI

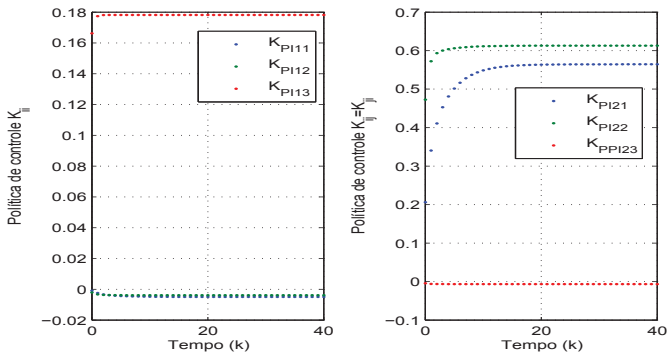

Figura 6.7: Convergência da política ótima K por PI

Para $N = 40$ iterações, chega-se a convergência do algoritmo como mostram as Figuras 6.6 e 6.7, e a solução obtida por PI é a igual a solução obtida pelo método de *Schur*, sendo assim, $P_{PI} = P_{Schur}$ e $K_{PI} = K_{Schur}$. A trajetória ótima para cada estado x_k e a ação de controle u_k podem ser verificadas nas Figuras 6.8 e 6.9. Percebe-se que pelo algoritmo PI, os estados também alcançam o estado de equilíbrio.

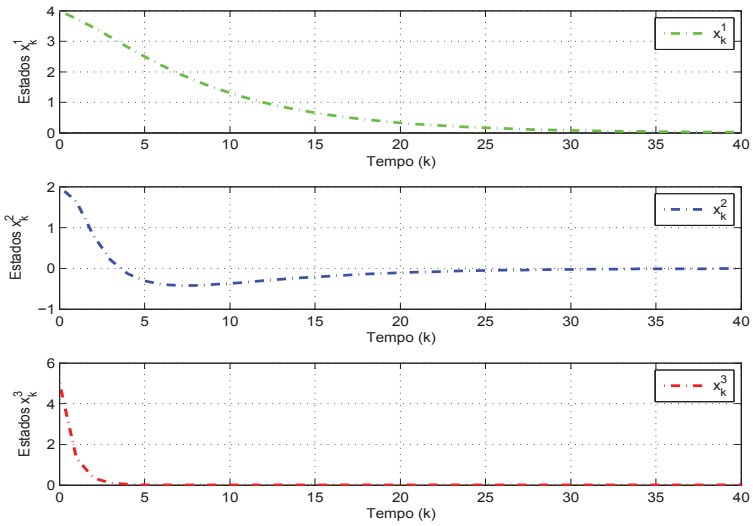

Figura 6.8: Trajetória dos Estados x_k por PI

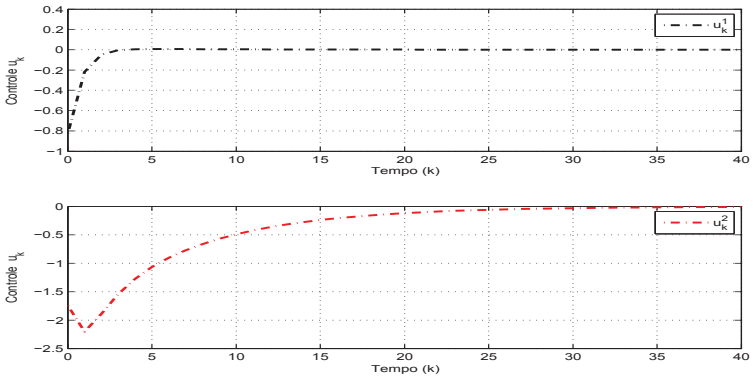

Figura 6.9: Ação de Controle u_k por PI

A evolução dos autovalores σ a cada iteração pode ser verificada na Figura 6.10. Para cada ganho K obtido por cada iteração, gera um autovalor que aloca o sistema em determinado local no plano Z complexo.

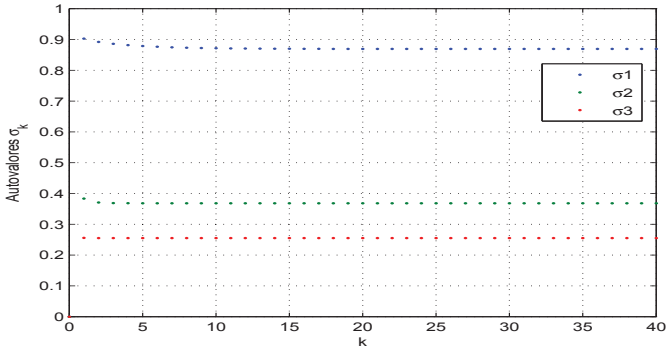

Figura 6.10: Autovalores σ a cada iteração por Programação Dinâmica

Lembrando que tanto a solução por PI, quanto a solução por VI que será exposta logo a frente, são soluções *offline*, pois necessitam do conhecimento da dinâmica do sistema, (A_d e B_d), como já visto anteriormente.

A análise da convergência QR por PI, é feita logo a seguir e exposta na Tabela 6.2. Para esta situação tem-se que $q_i = \{2, 1, 0, 1, -2\}$.

Tabela 6.2: DLQR-AR-PI e Variações da Matriz Q.

q_i	r_i	N	Autovalores σ
2	0	57	0.8674, 0.0097 e 0.0090
1	0	49	0.8676, 0.0826 e 0.0721
0	0	40	0.8693, 0.3683 e 0.2553
-1	0	37	0.8808, 0.6584 e 0.3517
-2	0	33	0.9001, 0.7401 e 0.3662

As mesmas observações feitas quando utilizou-se PD, podem ser notadas neste caso utilizando PI. A semelhança dos valores se deve também pela escolha de $P_0 = I_{3\times3}$, que neste caso, é igual a condição final $S = I_{3\times3}$ dada por PD.

6.4.2 Implementação *offline* do algoritmo VI para sistemas MIMO

Para o algoritmo VI, utilizou-se as matrizes de ponderação Q e R como identidades. A inicialização do sistema, leva em consideração uma política inicial não necessariamente admissível. É prática comum, se escolher como valor inicial

$P_0 = 0$. Utiliza-se a Eq.(3.41) para determinação da matriz P de *Riccati*, enquanto a Eq.(3.40) é utilizada na determinação do ganho ótimo do controlador. O fator de desconto para este caso também foi de $\gamma = 1$.

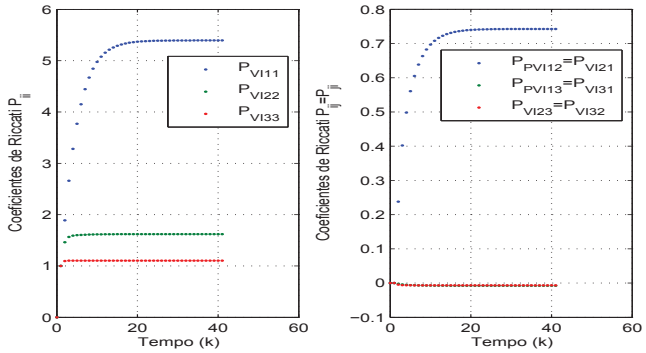

Figura 6.11: Convergência dos coeficientes da matriz P de *Riccati* por VI

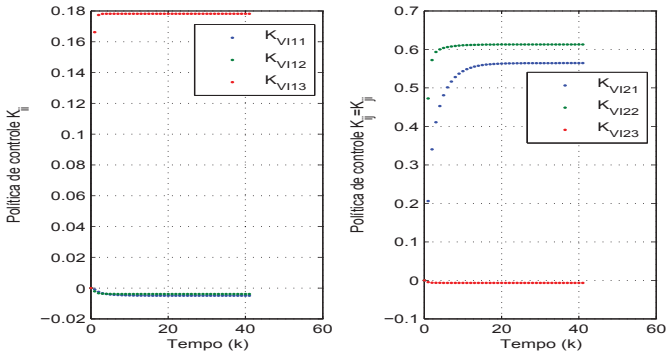

Figura 6.12: Convergência da política ótima K por VI

Verifica-se que para esse caso a convergência é alcançada após $N = 41$ iterações como mostram as Figuras 6.11 e 6.12, ou seja, $P_{VI} = P_{Schur}$ e $K_{VI} = K_{Schur}$. A trajetória ótima para cada estado x_k e a ação de controle u_k podem ser verificadas nas Figuras 6.13 e 6.14. Percebe-se que, da mesma forma dos outros algoritmos, os estados alcançam o estado de equilíbrio $x_k = 0$. A escolha adequada das matrizes de ponderação Q e R tem influência relevante na convergência.

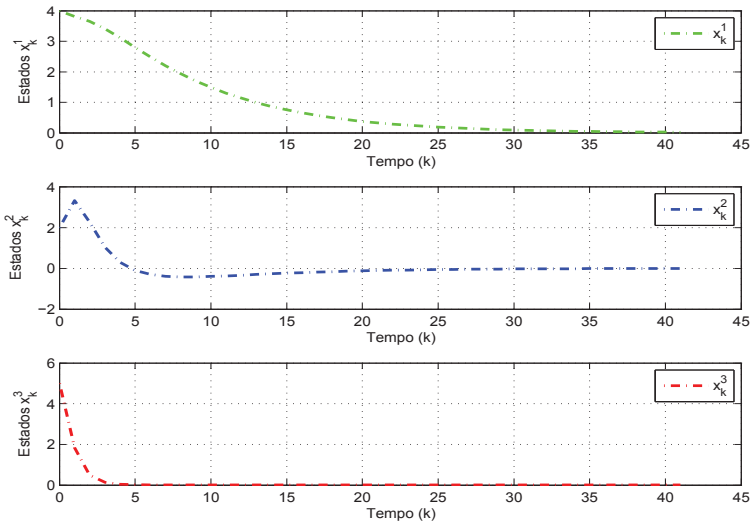

Figura 6.13: Trajetória dos Estados x_k por VI

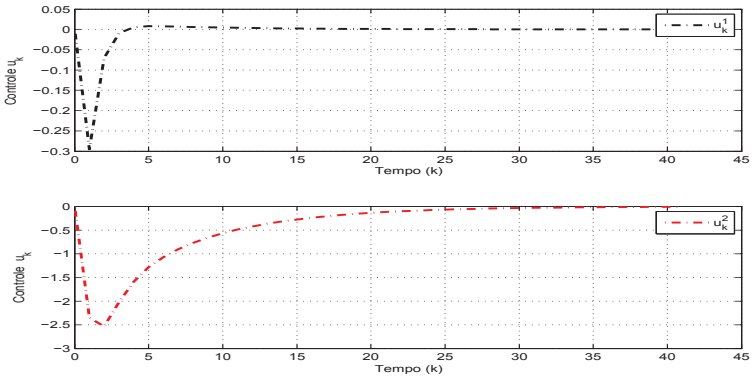

Figura 6.14: Ação de Controle u_k por VI

Os autovalores σ a cada iteração pode ser verificada na Figura 6.15. Para cada ganho K obtido por cada iteração, gera um autovalor que aloca o sistema em determinado local no plano Z complexo.

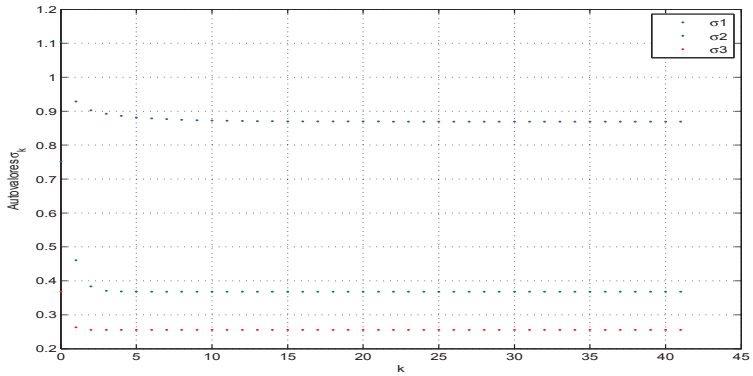

Figura 6.15: Autovalores σ a cada iteração por VI

Análise da convergência QR por VI, é exposta na Tabela 6.3. Para esta situação seguiu-se a heurística da escolha de $q_i = \{2, 1, 0, 1, -2\}$.

Tabela 6.3: DLQR-AR-VI e Variações da Matriz Q.

q_i	r_i	N	Autovalores σ
2	0	57	0.8674, 0.0097 e 0.0090
1	0	49	0.8676, 0.0826 e 0.0721
0	0	41	0.8693, 0.3683 e 0.2553
-1	0	46	0.8808, 0.6584 e 0.3517
-2	0	61	0.9001, 0.7401 e 0.3662

Percebe-se que a medida que se varia q_i, tem-se uma variação de N iterações, não obedecendo a mesma lógica imposta quando se utiliza o algoritmo PI. A condição não admissível imposta na na inicialização do algoritmo, exerce forte influência no número de iterações para convergência do algoritmo. No ponto de vista dos autovalores, não há modificações em comparação com os obtidos por PD e PI.

6.5 Resultados de ADP para o LQR discreto

Nesta seção, serão analisados os resultados obtidos por ADP, especificamente os algoritmos 10 e 11, HDP e AD-HDP respectivamente, para o DLQR, do sistema MIMO de terceira ordem. Análises sobre os aspectos de convergência do algoritmo

em relação a número de iterações e a relação dos valores das matrizes Q e R com alocação de autovalores no plano Z serão feitas.

6.5.1 Implementação *online* do algoritmo HDP para sistemas MIMO

O algoritmo *online* HDP, demonstrado logo a seguir, é utilizado para controle *online* do avião F-16 em avanço no tempo.

ALGORITMO 10($ADP - HDP - DLQR$)

▷ **Inicialização**
$\bar{p}_0 = vec(P_0) \geq 0$; $P_0 \geq 0$; $x_0 \leftarrow [\quad\quad]$;$j = 0$
Ponderações e Sistema Dinâmico.
$[Q, R, A_d, B_d]$
Selecionar o Fator de Desconto:
$0 < \gamma \leq 1$
▷ **Processo Iterativo**.
for $j \rightarrow j + 1$
    do
        ▷ **Política de Iteração (Ação de Controle)**
        $K_j \leftarrow (R/\gamma + B_d^T P_j B_d)^{-1} B_d^T P_j A_d$
        $h_j(x_j) = -K_j x_j$
        ▷ **Sistema Dinâmico**
        $x_{j+1} = A_d x_j + B_d u_j$
        ▷ **Resolução através do LS Batelada**:
        $X = \left[\begin{array}{cccc} \overline{x}|_{x_{k-N-1}} & \overline{x}|_{x_{k-N-2}} & \ldots & \overline{x}|_{x_{k-1}} \end{array} \right]$
        $Y = \left[\begin{array}{cccc} d(x_{k-N-1}, \bar{p}_j) & d(x_{k-N-2}, \bar{p}_j) & \ldots & d(x_{k-1}, \bar{p}_j) \end{array} \right]^T$
        if $j == N$
          then
                ▷ **Recorrência de *Riccati***
                $\bar{p}_{j+1} = (XX^T)^{-1} XY$
                $P_{j+1} = f(\bar{p}_{j+1})$
                ▷ **Reinicialização dos estados**
                $x_{j+1} \leftarrow [\quad\quad]$
        if $\|\bar{p}_{j+1} - \bar{p}_j\|_F < \varepsilon$
          then
Fim do Processo Iterativo.

Para manter a condição de excitação, pode-se usar vários esquemas padrões, incluindo reinicialização dos estados ou injeção de um sinal de ruído branco (AL-TAMIMI, *et al.*, 2007*b*). A reinicialização de estados apareceu em (MURRAY *et al.*, 2002) para resolução da equação HJB associadas para controle ótimo em tempo contínuo.

No projeto de HDP, os estados do avião são inicializados com $x_0 = \left[\begin{array}{ccc} 4 & 2 & 5 \end{array} \right]$. Qualquer valor pode ser selecionado. As matrizes de ponderação, Q e R, são

inicializadas como identidade com suas respectivas dimensões. É selecionado o fator de desconto, $\gamma = 1$. Os parâmetros do crítico e do ator são inicializados com zero. Após esta etapa de inicialização, a dinâmica do avião é executada em avanço no tempo, e o ajuste das estruturas parâmetros é realizada por meio da observação dos estados *online*.

Nas Figuras 6.16 e 6.17, tem-se os estados e as entradas de controle em relação ao tempo. Neste exemplo, foi utilizado a reinicialização dos estados periodicamente com valores de $x_0 = \begin{bmatrix} 7 & 2 & -5 \end{bmatrix}$ para impedir a singularidade.

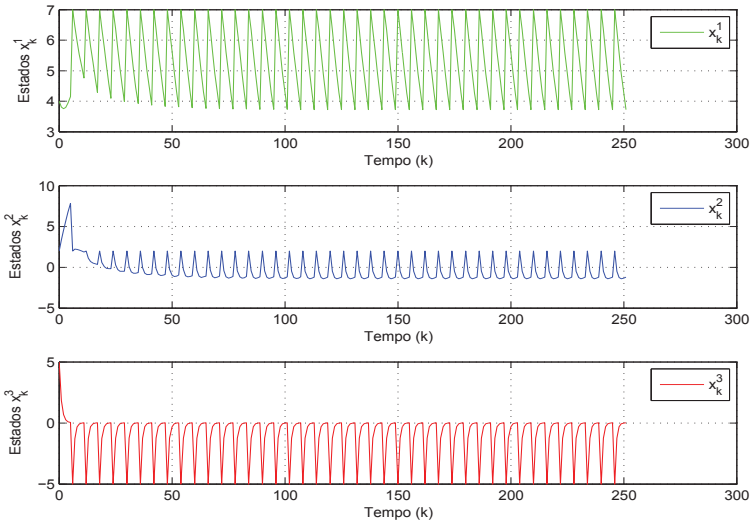

Figura 6.16: Trajetória dos Estados x_k com reinicialização por HDP

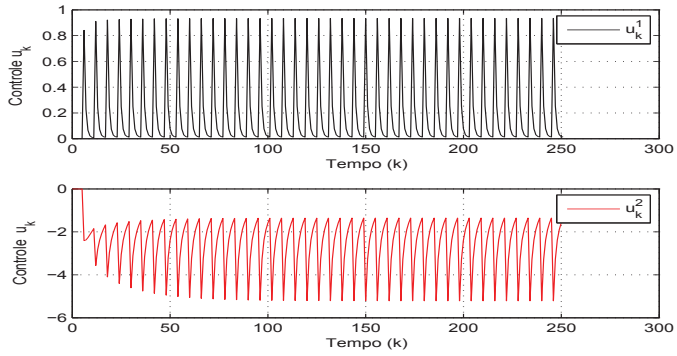

Figura 6.17: Ação de Controle u_k por HDP

Nas Figuras 6.18 e 6.19, a convergência dos parâmetros do crítico e da ação são mostrados. Como esperado, os parâmetros do crítico converge para P de *Riccati*. O crítico leva 251 intervalos de tempo para convergir para P. A razão

para isto é que 6 leituras são necessárias para ajustar o crítico a cada atualização para resolver cada P_j.

É importante frisar, que a reinicialização de estados fornece as condições excitação necessárias para obter a convergência dos parâmetros. Uma vez que esses parâmetros sejam conhecidos, o controlador DLQR foi encontrado. Então, pode-se utilizar os parâmetros da ação de controle como os parâmetros finais do controlador em qualquer controle *online*, sem ter que inserir deliberadamente quaisquer sinais de excitação para o sistema.

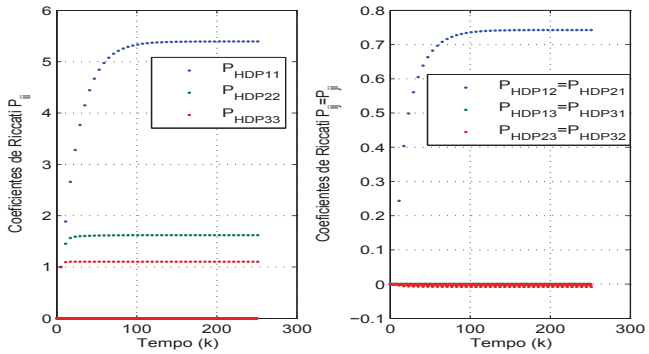

Figura 6.18: Convergência dos parâmetros P do crítico por HDP

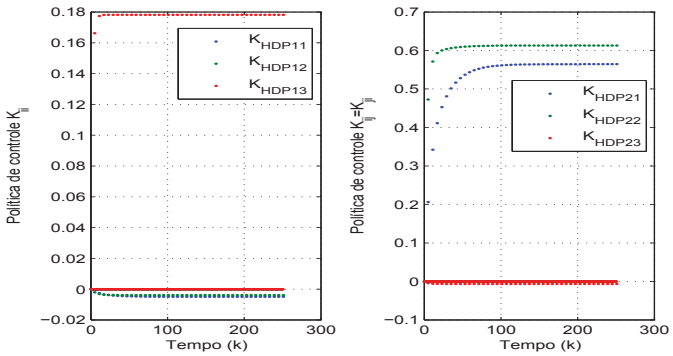

Figura 6.19: Convergência da política ótima K por HDP

Os autovalores a cada iteração pode ser verificada na Figura 6.20.

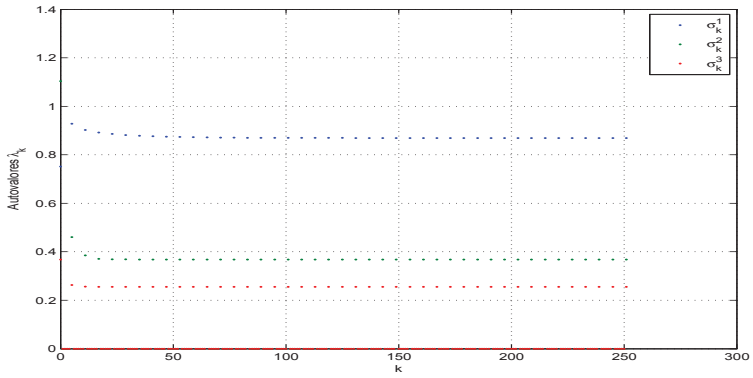

Figura 6.20: Autovalores σ a cada iteração por HDP

Análise da convergência QR por HDP, é exposta na Tabela 6.4. Para esta situação seguiu-se a heurística da escolha de $q_i = \{1, 0, 1, -2\}$.

Tabela 6.4: DLQR-ADP-HDP e Variações da Matriz Q.

q_i	r_i	N	Autovalores σ
1	0	-	-
0	0	251	0.8693, 0.3683 e 0.2553
-1	0	275	0.8808, 0.6584 e 0.3517
-2	0	365	0.9001, 0.7401 e 0.3662

Observa-se que as variações na matriz Q, conduziram ao mapeamento de pólos reais limitados ao eixo Z real e ao semi plano direito. Uma investigação para mapeamento em outras regiões do plano Z envolve outras heurísticas para variações de Q e R. Verifica-se também que não se obtém solução para valores acima de $q_i > 0$. Para valores de $q_i < 0$, os autovalores tendem a ficar mais distantes da origem e as iterações aumentam.

6.5.2 Implementação *online* do algoritmo AD-HDP para sistemas MIMO

O algoritmo AD-HDP (*Q-Learning*), é utilizada para controle *online* do avião F-16 em avanço no tempo.

ALGORITMO 11($ADP - AD\text{-}HDP - DLQR$)

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

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

Nas Figuras 6.21 e 6.22, tem-se os estados e as entradas de controle em relação ao tempo.

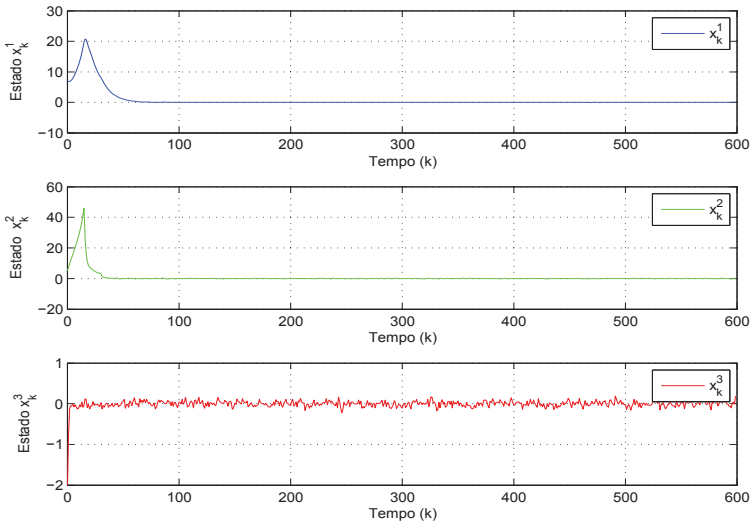

Figura 6.21: Trajetória dos Estados x_k por AD-HDP

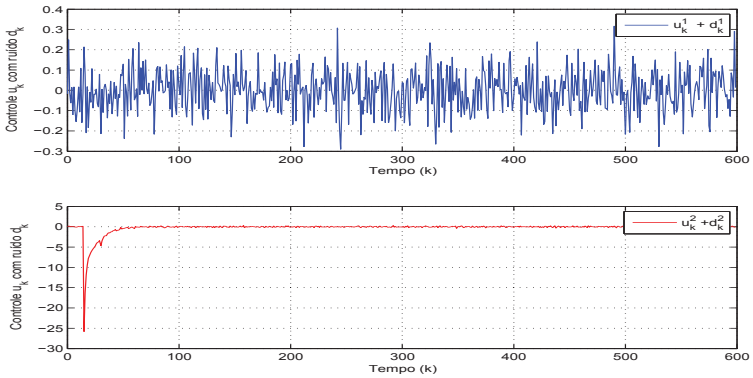

Figura 6.22: Ação de Controle u_k por AD-HDP

Para manter a condição de excitação, utilizou-se, aqui, a injeção de um ruído no controle, que pode ser visto na Figura 6.23. Assim, tem-se a condição de excitação persistente necessária para convergência do LS se evitando desvio de parâmetros. O ruído de controle está associado a convergência de P como visto no Capítulo 5.

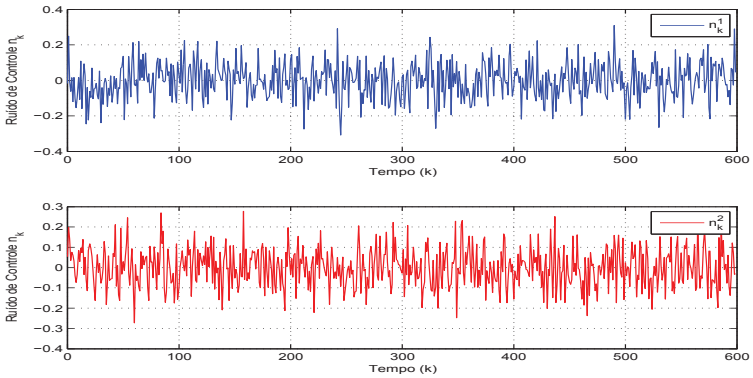

Figura 6.23: Ruído de controle n_k

A matriz H_j da Eq.(5.27), é encontrada de forma *online* através do algoritmo proposto.

$$H_{AD-HDP} = \begin{bmatrix} 6.2200 & 1.6388 & -0.0202 & -0.0153 & 1.4620 \\ 1.6388 & 2.5934 & -0.0197 & -0.0144 & 1.5876 \\ -0.0202 & -0.0197 & 1.1496 & 0.2568 & -0.0197 \\ -0.0153 & -0.0144 & 0.2568 & 1.4411 & -0.0143 \\ 1.4620 & 1.5876 & -0.0197 & -0.0143 & 2.5901 \end{bmatrix} \tag{6.5}$$

Através dela, pode-se encontrar os parâmetros de P_j. Neste caso, por se trabalhar também com a ação u, 15 leituras são necessárias para ajustar o crítico a cada atualização para resolver cada H_j. O crítico leva 599 intervalos de tempo para convergir para H e consequentemente para P. Nas Figuras 6.24 e 6.25, é mostrado a convergência da ação de controle e a localização dos autovalores.

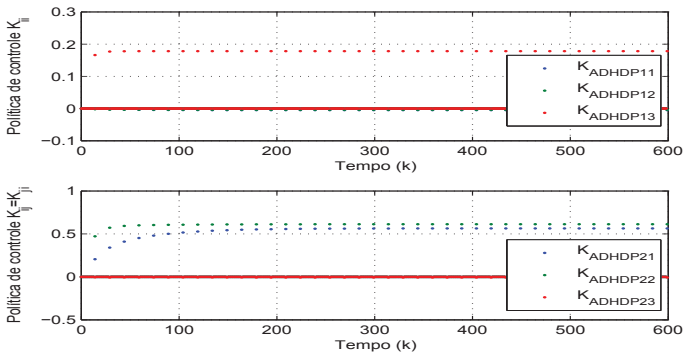

Figura 6.24: Convergência da política ótima K por AD-HDP

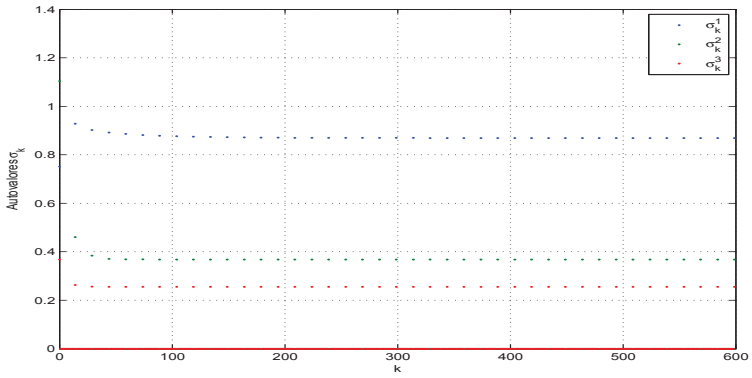

Figura 6.25: Autovalores σ a cada iteração por AD-HDP

Afim de verificar a independência da ação de controle em relação a matriz A_d, na iteração 300, modificou-se os elementos $A_d(1,1) = -0.5$ e $A_d(3,2) = -1$. Nas Figuras 6.27 e 6.26 , é mostrado a convergência da ação de controle e a localização dos autovalores.

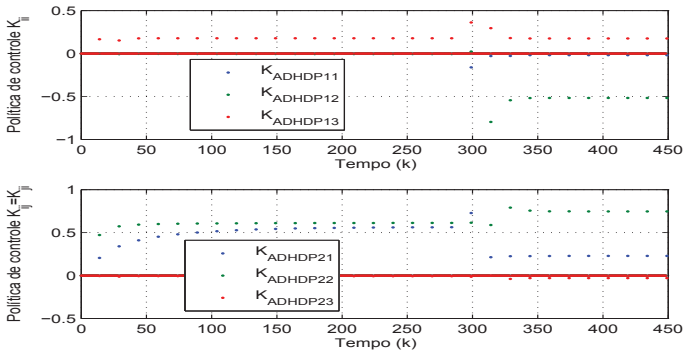

Figura 6.26: Convergência da política ótima K por AD-HDP para modificação em A_d na 300ª iteração

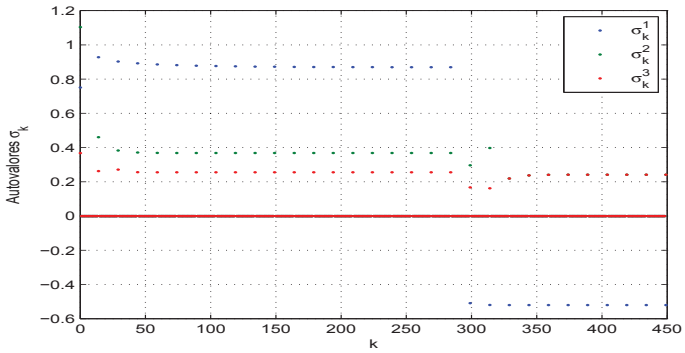

Figura 6.27: Autovalores σ a cada iteração por AD-HDP para modificação em A_d na 300ª iteração

Para excitar mais o sistema dinâmico, simulou-se uma mudança brusca nos estados na iteração 209. Então, no instante posterior, os estados passaram a ser $x_{210} = \begin{bmatrix} 4 & 2 & 5 \end{bmatrix}$. As Figuras 6.28 e 6.29 retratam o comportamento dos estados e da ação de controle.

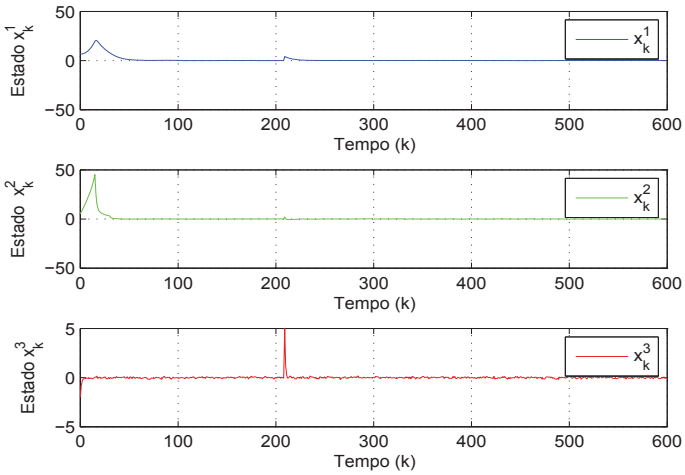

Figura 6.28: Trajetória dos Estados x_k por AD-HDP com mudança no estado x_{210}

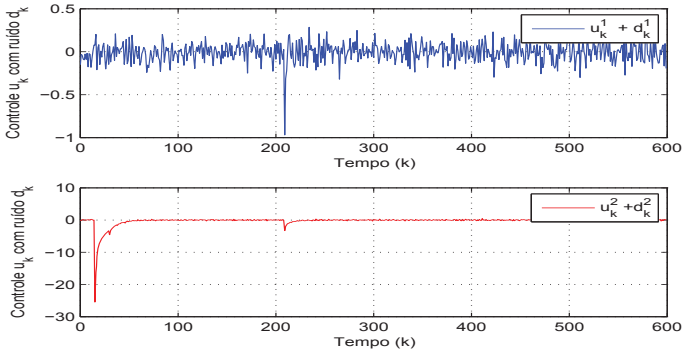

Figura 6.29: Ação de Controle u_k por AD-HDP com mudança no estado x_{210}

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

Análise da convergência QR por AD-HDP, é exposta na Tabela 6.5. Para esta situação seguiu-se a heurística da escolha de $q_i = \{5, 2, 1, 0, 1, -2, -5\}$.

Tabela 6.5: DLQR-ADP-ADHDP e Variações da Matriz Q.

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

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

6.6 Conclusão

Neste Capítulo foi visto os resultados obtidos através dos diversos algoritmos propostos.A *priori* desenvolveu-se o projeto do DLQR pro Programação Dinâmica. Observou-se que o algoritmo chega a solução P de *Riccati* para 40 iterações.

Logo após, implementou-se para ocaso do DLQR, os algoritmos de resolução de AR *offline*, PI e VI. Verificou-se que a convergência por PI, para o caso base, foi mais rápida do que por VI. Isto se deve ao fato de se inicializar com uma política inicial admissível.

Em seguida, teve-se a implementação *online* dos algoritmos HDP e AD-HDP. Observou-se um número maior de iterações para a convergência no algoritmo AD-HDP. Isto se deve ao fato da quantidade de pontos coletados para construção da matrizes dos regressores. Devido ao modelo do F-16 se tratar de um sistema MIMO de três estados e duas entradas de controle, para o HDP necessitou-se de 6 pontos de coleta de dados, enquanto que para o AD-HDP necessitou-se de 15 pontos.

O método utilizado para seleção das matrizes de ponderação QR foi o MSH-QR. Para cada algoritmo proposto se fez uma análise da variação da matriz Q em relação a convergência por número de iterações e alocação de autovalores.

Um fato interessante, e que vale ser comentado, é que a medida que se varia q_i para valores abaixo da referência, no algoritmo PI a convergência é alcançada para iterações cada vez menores. Em contra partida, no algoritmo VI, quando q_i varia para valores abaixo da referência, o número de iterações tendeu a aumentar. No mesmo algoritmo, para $q_i = 2$ teve-se a convergência na iteração de número 57, enquanto para $q_i = -2$ teve-se uma convergência na iteração de número 61, ou seja, nenhum padrão foi observado em relação ao número de iteração a variação de q_i.

Nos algoritmos *online*, HDP e AD-HDP, também foi adotado a heurística da variação das matrizes QR. Para o HDP, não obteve-se convergência para valores de $q_i > 0$. Para o algoritmo AD-HDP, obteve-se convergência para diversos valores de q_i, porém nenhum padrão foi verificado em relação a variação de q_i e o número de iterações.

Capítulo 7

CONCLUSÃO

Nesta dissertação, foi apresentado o projeto de controlador ótimo discretos do tipo DLQR através de ADP. A técnica de ADP viabiliza a a sintonia da política de controle e a resolução do problema da Programação Dinâmica *online* em avanço no tempo, caracterizando-se assim um controle adaptativo ótimo direto. Mostrou-se que TD e a Aproximação da Função Valor são tópicos importantes na implementação de uma ADP.

A solução *online* de AR através do algoritmo HDP, mostrou uma convergência mais rápida através de LS em comparação com o algoritmo AD-HDP. Isto se deve ao fato de que para o AD-HDP, na construção da matriz dos regressores, necessita-se também da informação da ação de controle u. Viu-se que para o modelo Ator-Crítico da estrutura HDP, a determinação da política de controle fica dependente da dinâmica do sistema, sendo assim um modelo parcialmente livre (*partially model-free*). Na estrutura AD-HDP, em contrapartida, pode-se considerar um modelo livre (*model-free*), por não necessitar do conhecimento da dinâmica na determinação de K.

Os experimentos computacionais, mostraram as diversas soluções alcançadas, para um sistema MIMO de 3ª ordem de um avião F-16, pelos algoritmos de Programação Dinâmica, AR *offline* por Política de Iteração e Valor de Iteração e pelos métodos *online* de ADP: HDP e AD-HDP . A análise de convergência pelo número de iterações e pela sintonia das matrizes Q e R foram apresentadas.

O método de solução *online* de ADP dado pelo algoritmo AD-HDP teve maior habilidade, comparando com o HDP, em realizar um mapeamento mais amplo dos autovalores no plano Z estável. Algo interessante é o fato de que para qualquer um destes algoritmos propostos, ao menos algum conhecimento do sistema é necessário. Na HDP fica explicito na determinação do ganho do controlador,

enquanto que no AD-HDP fica implícito na determinação da dimensão da matriz H.

algoritmos de ADP aqui desenvolvidos, mostraram-se, dentro de suas limitações, processos de soluções viáveis e com suas soluções aplicáveis na prática. Na ADP, foi mostrado que tendo somente uma pequena informação sobre os estados do sistema através de sensores, e extraídos do sistema apenas em momentos específicos, $(x_k, x_{k+1}, r(x_k, u_k))$, pode-se obter a solução de *Riccati online*. Consequentemente tem-se um controlador ótimo discreto e adaptativo.

7.1 Trabalhos Futuros

Alguns estudos e desenvolvimentos sobre ADP podem ser abordados para trabalhos futuros. Especificamente, a pesquisa pode seguir para os seguintes desenvolvimentos e investigações.

- Enfocar soluções de ADP por outras técnicas como DHP e AD-DHP.
- Aplicar como técnica de soluções de aproximação da Função Valor *Recursive Least Squares - RLS*(Mínimos Quadrados Recursivos), Redes Neurais ou Lógica *Fuzzy*.
- Aplicação de outros métodos de sintonia das matrizes de ponderação Q e R investigando-se sua relação com a convergência do algoritmo de ADP.
- Análise da relação da quantização de entradas e estados para sistemas MIMO com a convergência por ADP.

Apêndice A

FORMULAÇÃO DO ÍNDICE DE DESEMPENHO POR CÁLCULO VARIACIONAL

Este apêndice complementa os estudos do Capítulo 2. A abordagem é direcionada para a formulação do controle ótimo discreto por cálculo variacional levando-se em consideração o multiplicador de *Lagrange* e o princípio do máximo e mínimo discreto. O desenvolvimento matemático aqui utilizado tem como referência (KUO, 1980).

A.1 Equação discreta *Euler-Lagrange*

Considere o seguinte problema de otimização:

$$\min J = \sum_{k=0}^{N-1} F[x_k, x_{k+1}, u_k, k] \tag{A.1}$$

sujeito a:

$$x_{k+1} = f[x_k, u_k, k] \tag{A.2}$$

considerando o multiplicador de *Lagrange*:

$$\lambda_{k+1}$$

A função Lagrangiana fica:

$$\ell(x, \lambda) = F[x_k, x_{k+1}, u_k] + \lambda_{k+1}^T [x_{k+1} - f[x_k, u_k, k] \tag{A.3}$$

Considerando as variações de x_k, x_{k+1}, u_k e λ_{k+1} sendo:

$$x_k = x_k^* + \varepsilon\eta_k \tag{A.4}$$

$$x_{k+1} = x_{k+1}^* + \varepsilon\eta_{k+1} \tag{A.5}$$

$$u_k = u_k^* + \delta\mu_k \tag{A.6}$$

$$\lambda_{k+1} = \lambda_{k+1}^* + \gamma\omega_{k+1} \tag{A.7}$$

sendo x_k^*, x_{k+1}^*, u_k^* e λ_{k+1}^* as trajetórias ótimas; η_k, μ_k, ω_k são variáveis arbitradas. Assim, fazendo as manipulações matemáticas das equações (A.4) a (A.7) na equação de custo modificada pelo multiplicador de *Lagrange*, então:

$$\begin{aligned} J_c = & \sum_{k=0}^{N-1} F[x_k^o + \varepsilon\eta_k,\ x_{k+1}^* + \varepsilon\eta_{k+1}, u_k^* + \delta\mu_k, k] \\ & + \langle \lambda_{k+1}^* + \gamma\omega_{k+1}, x_{k+1}^* + \varepsilon\eta_{k+1} \\ & - f[x_k^* + \varepsilon\eta_k, u_k^* + \delta\mu_k, k] \rangle \end{aligned} \tag{A.8}$$

Simplificando-se a Eq.(A.8) tem-se:

$$J_c = \sum_{k=0}^{N-1} F_c[x_k,\ x_{k+1}, \lambda_{k+1}, u_k, k] \tag{A.9}$$

Expandindo-se F_c em série de *Taylor* em torno de x_k^*, x_{k+1}^*, u_k^o e λ_{k+1}^*:

$$\begin{aligned} & F_c[x_k,\ x_{k+1}, \lambda_{k+1}, u_k, k] = F_c[x_k^*,\ x_{k+1}^*, \lambda_{k+1}^*, u_k^*, k] \\ & + \left\langle \varepsilon\eta_k, \frac{\partial F_{c_k^*}}{\partial x_k^*} \right\rangle + \left\langle \varepsilon\eta_k, \frac{\partial F_{c_k^*}}{\partial x_{k+1}^*} \right\rangle + \left\langle \gamma\omega_{k+1}, \frac{\partial F_{c_k^*}}{\partial \lambda_{k+1}^*} \right\rangle \\ & + \left\langle \delta\mu_k \frac{\partial F_{c_k^*}}{\partial u_k^*} \right\rangle + \textit{termos} \text{ de ordem superior} \end{aligned} \tag{A.10}$$

Condição necessária para obtermos o mínimo de J_c.

$$\left.\frac{\partial J_c}{\partial \varepsilon}\right|_{\varepsilon=\delta=\gamma=0} = 0 \tag{A.11}$$

$$\left.\frac{\partial J_c}{\partial \gamma}\right|_{\varepsilon=\delta=\gamma=0} = 0 \tag{A.12}$$

$$\left.\frac{\partial J_c}{\partial \varepsilon}\right|_{\varepsilon=\delta=\gamma=0} = 0 \tag{A.13}$$

Substituindo a expansão da série de *Taylor* na Eq.(A.8) e aplicando-se as condições necessárias para se obter o mínimo J_c, tem-se:

$$\sum_{k=0}^{N-1}\left[\left\langle \eta_k, \frac{\partial F_{c_k^*}}{\partial x_k^*}\right\rangle + \left\langle \eta_{k+1}, \frac{\partial F_c^*(k)}{\partial x_{k+1}^*}\right\rangle\right] = 0 \tag{A.14}$$

$$\sum_{k=0}^{N-1}\left[\left\langle \omega_{k+1}, \frac{\partial F_{c_k^*}}{\partial \lambda_{k+1}^*}\right\rangle\right] = 0 \tag{A.15}$$

$$\sum_{k=0}^{N-1}\left[\left\langle \mu_k, \frac{\partial F_{c_k^*}}{\partial \lambda_k^*}\right\rangle\right] = 0 \tag{A.16}$$

A equação pode ser escrita como:

$$\begin{aligned}\sum_{k=0}^{N-1}\left\langle \eta_k, \frac{\partial F_{c_k^*}}{\partial x_k^*}\right\rangle &= -\sum_{k=1}^{N}\left\langle \eta_k, \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*}\right\rangle \\ &= -\sum_{k=0}^{N-1}\left\langle \eta_k, \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*}\right\rangle \\ &+ \left\langle \eta_k, \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*}\right\rangle\Big|_{k=0} - \left\langle \eta_k, \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*}\right\rangle\Big|_{k=N}\end{aligned} \tag{A.17}$$

sendo:

$$F_{c_{k-1}^*} = F_c[x_{k-1}^*, x_k^*, \lambda_k^*, u_{k-1}^*, k-1] \tag{A.18}$$

Organizando os termos da Eq.(A.17):

$$\begin{aligned}&\sum_{k=0}^{N-1}\left\langle \eta_k, \frac{\partial F_{c_k^*}}{\partial x_k^*} + \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*}\right\rangle \\ &+ \left\langle \eta_k, \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*}\right\rangle\Big|_{k=0}^{k=N} = 0\end{aligned} \tag{A.19}$$

Lema do cálculo variacional:

$$\frac{\partial F_c^*}{\partial x_k^*} + \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*} = 0 \tag{A.20}$$

$$\left\langle \eta_k, \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*} \right\rangle \Bigg|_{k=0}^{k=N} = 0 \tag{A.21}$$

A Eq.(A.20) é chamada de equação discreta de *Euler-Lagrange* que é a condição necessária que deve satisfazer J_c para ser extremo e a Eq.(A.21) é conhecida como condição de transversalidade.

Condição sem restrição 2.

$$\frac{\partial F_{c_k^*}}{\partial \lambda_{j_{k+1}^*}} = 0 \quad p/ \quad j = 1, 2, 3, ..., n \tag{A.22}$$

então, tem-se que esta derivada resulta na satisfação da equação da trajetória ótima.

$$x_{k+1}^* = f[x_k^*, u_k^*, k] \tag{A.23}$$

Restrição 3.

$$\frac{\partial F_{c_k^*}}{\partial u_{j_{k+1}^*}} = 0 \quad p/ \quad j = 1, 2, 3, ..., p \tag{A.24}$$

Fornece a lei de controle em termos de λ_{k+1}^*.

Na maioria dos problemas de projeto o estado inicial x_0 é dado no *outset*. Então a perturbação para x_k e $k = 0$ é zero desde que x_0 seja fixado. Logo $\eta_0 = 0$.

A condição de transversalidade fica reduzida a :

$$\left\langle \eta_k, \frac{\partial F_{c_{k-1}^*}}{\partial x_k^*} \right\rangle \Bigg|_{k=N} = 0 \tag{A.25}$$

Os problemas de controle podem ser classificados de acordo com as condições finais. Considerando x_N como um dado ponto fixo, o problema é definido como problema do ponto fixo. Por outro ado se x_N é livre , então tem-se o problema do ponto livre.

Problema do ponto fixo:

- x_N=fixo
- $\eta_N = 0$

logo

$$\left.\frac{\partial F_{c^*_{k-1}}}{\partial x^*_k}\right|_{k=N} \tag{A.26}$$

Problema do ponto final livre:

- x_N=livre
- $\eta_N \neq 0$

logo

$$\left.\frac{\partial F_{c^*_{k-1}}}{\partial x^*_k}\right|_{k=N} = 0 \tag{A.27}$$

A.2 Princípio do máximo (mínimo) discreto

Princípio introduzido por *Pontryagin* é um método de solução de grande classe de sistemas de controle com dados contínuos. O projeto do princípio máximo baseia-se no cálculo variacional mas os mecanismos são mais elegantes e refinados do que a utilização da equação de *Euler-Lagrange*. O problema baseia-se em determinar a lei de controle ótimo u^*_k no intervalo $[0, N]$ que satisfaça:

$$\min J = G[x_N, N] + \sum_{k=0}^{N-1} F[x_k, u_k, k] \tag{A.28}$$

sujeito a :

$$x_{k+1} = f[x_k, u_k, k] \tag{A.29}$$

Definir o vetor de co-estado p_k (nx1) de forma que o problema de otimização seja formulado como:

$$\min J_c = G[x_N, N] + \sum_{k=0}^{N-1} [F[x_k, u_k, k] + \left\langle p^T_{k+1}[x_{k+1} - f(x, u, k)]\right\rangle] \tag{A.30}$$

A função objetivo nova é chamado de Hamiltoniana e representada por:

$$H[x_k, u_k, p_{k+1}, k] \tag{A.31}$$

Ainda:

$$H[x_k, u_k, p_{k+1}, k] = F[x_k, u_k, k] - \left\langle p_{k+1}^T f[x_k, u_k, k] \right\rangle \tag{A.32}$$

Em geral o projeto de controle com o tempo ótimo é definido como o problema de conduzir (levar) o estado x_0 para x_N em um tempo mínimo.

O contorno é restrito a uma amplitude.

$$|u(Kt)| \leq U \tag{A.33}$$

Apêndice B

CONVERGÊNCIA DOS ALGORITMOS HDP E AD-HDP

A apresentação das provas de convergência tem por objetivo fornecer as desenvolvimento da teoria e elucidar fenômenos do processo iterativo do Capítulo 5. É apresentado neste apêndice as provas de convergência dos algoritmos HDP e AD-HDP do LQR discreto baseado nos artigos (AL-TAMIMI, *et al.*, 2007*b*) e (AL-TAMIMI *et al.*, 2007*c*).

B.1 Convergência do Algoritmo HDP

A análise da convergência do algoritmo de HDP proposto nesta dissertação é demonstrado logo a seguir. A iteração das equações (3.40) e (5.8) é equivalente a iteração da equação de *Riccati* a seguir.

$$P_{j+1} = Q + \gamma \left(A_d^T P_j A_d - A_d^T P_j B_d \left(R/\gamma + B_d^T P_j B_d \right)^{-1} B_d^T P_j A_d \right) \tag{B.1}$$

O problema de LS definido por (5.8) pode ser escrito como:

$$\int_{\Omega} (2\bar{x}\bar{x}^T \bar{p}_{j+1} - 2\bar{x} d^T(x, \bar{p}_j)) dx = 0 \tag{B.2}$$

e implica que:

$$\bar{p}_{j+1} = \left(\int_{\Omega} \bar{x}\bar{x}^T dx \right)^{-1} \int_{\Omega} \bar{x} d(x,\bar{p}_j) dx \tag{B.3}$$

Sobe a hipótese condição de excitação, o operador inverso existe. Substituindo-se (5.6) em (B.3).

$$\begin{aligned}\bar{p}_{j+1} &= \left(\int_{\Omega} \bar{x}_k \bar{x}_k^T dx\right)^{-1} \int_{\Omega} \bar{x}_k (x_k^T (Q + K_j^T R K_j \\ &+ \gamma (A - BK_j)^T P_j (A - BK_j) x_k) dx \end{aligned} \tag{B.4}$$

Utilizando-se o produto de *Kronecker* (BREWER, 1978), a Eq.(B.4) pode ser escrita como:

$$\begin{aligned}\bar{p}_{j+1} &= \left(\int_{\Omega} \bar{x}_k \bar{x}_k^T dx\right)^{-1} \left(\int_{\Omega} \bar{x}_k \bar{x}_k^T dx\right) \\ &\times v\left(Q + K_j^T R K_j + \gamma (A - BK_j)^T P_j (A - BK_j)\right) \\ &= v\left(Q + K_j^T R K_j + \gamma (A - BK_j)^T P_j (A - BK_j)\right) \end{aligned} \tag{B.5}$$

sendo v a função vetorização do produto de *Kronecker*.

Desde que a matriz P_{j+1}, que é reconstruída a partir de $\bar{p}_{j+1}$, seja simétrica então a iteração em $\bar{p}_j$ é equivalente a seguinte iteração.

$$P_{j+1} = Q + K_j^T R K_j + \gamma \left[(A - BK_j)^T P_j (A - BK_j)\right] \tag{B.6}$$

A Eq.(B.6) pode ser escrita na forma da Eq.(B.1)

Se a sequência de LS na Eq.(5.8) for solúvel, ou seja, as condições de excitação correspondente vigora, então o algoritmo HDP converge para o valor de *Riccati* quando se inicializa com $P_0 \geq 0$.

A prova de convergência do algoritmo HDP acaba de ser criado supondo-se que o problema do LS é resolvido completamente, ou seja, a condição de excitação é satisfeita. Note-se que uma maneira fácil de inicializar o algoritmo é selecionando $P_0 = 0$.

B.2 Convergência do Algoritmo AD-HDP

A análise da convergência do algoritmo de AD-HDP proposto nesta dissertação é demonstrado logo a seguir. A iteração das equações (5.32) e (5.54) é equivalente a

$$H_{j+1} = G + \gamma \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d^T \end{bmatrix}^T H_j \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d \end{bmatrix} \tag{B.7}$$

A Eq.(5.50) é equivalente a

$$d(\bar{z}_k(x_k), \bar{h}_j) = \bar{z}_k^T \times v \left(G + \gamma \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d^T \end{bmatrix}^T H_j \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d \end{bmatrix} \right) \tag{B.8}$$

Utilizando-se o produto de *Kronecker* (BREWER, 1978), o LS é dado por:

$$\bar{H}_{j+1} = \underbrace{(ZZ)^{-1}(ZZ)}_{I} \times v \left(G + \gamma \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d^T \end{bmatrix}^T H_j \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d \end{bmatrix} \right) \tag{B.9}$$

Desde que a matriz H_{j+1}, que é reconstruída a partir de $\bar{H}_{j+1}$, seja simétrica então a iteração em $\bar{H}_j$ é equivalente a seguinte iteração:

$$H_{j+1} = G + \gamma \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d^T \end{bmatrix}^T H_j \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d \end{bmatrix} \tag{B.10}$$

As matrizes H_{j+1} e K_{j+1} podem ser escritas como

$$H_{j+1} = \begin{bmatrix} Q + \gamma A_d^T P_j A_d & \gamma A_d^T P_j B_d \\ \gamma B_d^T P_j A_d & R + \gamma B_d^T P_j B_d \end{bmatrix} \tag{B.11}$$

$$K_{j+1} = (R/\gamma + B_d^T P_j B_d)^{-1} B_d^T P_j A_d \tag{B.12}$$

sendo P_j dado por:

$$P_j = \begin{bmatrix} I & -K_j^T \end{bmatrix} H_j \begin{bmatrix} I \\ -K_j \end{bmatrix} \tag{B.13}$$

A Eq.(B.7) pode ser escrita então como:

$$\begin{aligned} H_{j+1} &= G + \gamma \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d^T \end{bmatrix}^T H_j \begin{bmatrix} A_d & B_d \\ -K_j A_d & -K_j B_d \end{bmatrix} \\ &= G + \gamma \begin{bmatrix} A_d^T \\ B_d^T \end{bmatrix} \begin{bmatrix} I & -K_j^T \end{bmatrix} H_j \begin{bmatrix} I \\ -K_j \end{bmatrix} \begin{bmatrix} A_d & B_d \end{bmatrix} \end{aligned} \tag{B.14}$$

Desde que tenha-se (5.34) e (B.11) então, consequentemente tem-se (B.12)
A iteração de H_j é similar a iterar P_j. De (B.13) tem-se que:

$$P_{j+1} = \begin{bmatrix} I & -K_{j+1}^T \end{bmatrix} H_{j+1} \begin{bmatrix} I \\ -K_{j+1} \end{bmatrix} \tag{B.15}$$

Utilizando-se (B.11)tem-se:

$$\begin{aligned} P_{j+1} &= \begin{bmatrix} I & -K_{j+1}^T \end{bmatrix} \begin{bmatrix} Q + \gamma A_d^T P_j A_d & \gamma A_d^T P_j B_d \\ \gamma B_d^T P_j A_d & R + \gamma B_d^T P_j B_d \end{bmatrix} \begin{bmatrix} I \\ -K_{j+1} \end{bmatrix} \\ &= Q + K_{j+1}^T R K_{j+1} + \gamma \left[(A_d - B_d K_{j+1})^T P_j (A_d - B_d K_{j+1}) \right] \end{aligned} \tag{B.16}$$

Substituindo-se (B.12) em (B.16) tem-se:

$$P_{j+1} = \gamma \left(A_d^T P_j A_d - A_d P_j B_d \left({}^{R}/_{\gamma} + B_d P_j B_d \right)^{-1} B_d P_j A_d \right) + Q \tag{B.17}$$

A iteração a partir da Eq.(B.7), com $H_0 = 0$, $K_0 = 0$ converge com $H_j \to H$, sendo H correspondente a $Q^*(x_k, u_k)$ e $xPx = \min_u Q^*(x, u)$, com P sendo a solução de *Riccati*. Então tem-se H_J dado por:

$$H_j \to \begin{bmatrix} Q + \gamma A_d^T P A_d & \gamma A_d^T P B_d \\ \gamma B_d^T P A_d & R + \gamma B_d^T P B_d \end{bmatrix} \tag{B.18}$$

Para $i \to \infty$, tem-se $P_j \to P^*$, consequentemente $H_j \to H^*$ e $Q_j \to Q^*$.
A prova de convergência do algoritmo AD-HDP acaba de ser feita supondo-se

que o problema do LS é resolvido completamente, ou seja, a condição de excitação é satisfeita.

Referências Bibliográficas

Sutton, Richard S. and Andrew G. Barto (1998). Reinforcement learning i: Introduction.

Al-Tamimi, A. and F. Lewis (2007). Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof. In: *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*. pp. 38 –43.

Al-Tamimi, A., D. Vrabie, M. Abu-Khalaf and F.L. Lewis (2007*a*). Model-free approximate dynamic programming schemes for linear systems. In: *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*. pp. 371 –378.

Al-Tamimi, A., F.L. Lewis and M. Abu-Khalaf (2008). Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **38**(4), 943 –949.

Al-Tamimi, A., M. Abu-Khalaf and F.L. Lewis (2007*b*). Adaptive critic designs for discrete-time zero-sum games with application to control. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **37**(1), 240 –247.

Al-Tamimi, Asma, Frank L. Lewis and Murad Abu-Khalaf (2007*c*). Model-free q-learning designs for linear discrete-time zero-sum games with application to h-infinity control. *Automatica* **43**(3), 473 – 481.

Athans, Michael and L. Peter Falb (1966). *OPTIMAL CONTROL- An Introduction to the Theory and Its Apllications*. McGRAW-Hill Book Company . United States of America.

Balakrishnan, S.N., Jie Ding and F.L. Lewis (2008). Issues on stability of adp feedback controllers for dynamical systems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **38**(4), 913 –917.

Bellman, Richard (1958). Dynamic programming and stochastic control processes. *Information and Control* **1**(3), 228–239.

Bellman, Richard Ernest (2003). *Dynamic Programming*. Dover Publications, Incorporated.

Bertsekas, Dimitri P. (1995). *Dynamic Programming and Optimal Control, Two Volume Set*. Athena Scientific.

Bradtke, S.J., B.E. Ydstie and A.G. Barto (1994). Adaptive linear quadratic control using policy iteration. In: *American Control Conference, 1994*. Vol. 3. pp. 3475 – 3479 vol.3.

Bradtke, Steven J. (1993). Reinforcement learning applied to linear quadratic regulation. In: *In Advances in Neural Information Processing Systems 5*. Morgan Kaufmann. pp. 295–302.

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

Bryson, Arthur E. and Yu-Chio Ho (1975). *Applied Optimal Control*. Taylor and Francis. UK.

Buşoniu, L., R. Babuška, B. De Schutter and D. Ernst (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press. Boca Raton, Florida.

Doyle, J. C. and G. Stein (1981). Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis. *IEEE Transactions on Automatic Control* **26**, 4–16.

Dreyfus, Stuart E. and Averill M Law (1977). *The art and theory of dynamic programming / Stuart E. Dreyfus, Averill M. Law*. Academic Press, New York :.

Fonseca Neto, João and Leandro Rocha Lopes (2011). On the convergence of DLQR control and recurrences of riccati and lyapunov in dynamic programming. In: *UKSim 13th International Conference on Computer Modelling and Simulation(UKSim2011)*. Cambridge, United Kingdom.

Glorennec, Pierre Yves (2000). Reinforcement learning: an overview. In: *Europeia Sym. booktitle on Intelligent Techniques*.

Gupta, Madan M. and Sinha, Naresh K., Eds.) (1995). *Intelligent Control Systems: Theory and Applications*. IEEE Press. Piscataway, NJ, USA.

Johnson, M.A. and M.J. Grimble (1987). Recent trends in linear optimal quadratic multivariable control system design. *Control Theory and Applications, IEE Proceedings D* **134**(1), 53 –71.

Kirk, Donald E. (1970). *Optimal Control Theory: An Introduction*. Prentice-Hall Network Series. Prentice-Hall Inc.. Englewood Cliffs, New Jersey.

Kuo, Benjamin C. (1980). *Digital Control Systems*. Harcourt Brace College Publishers.

Lancaster, Peter and Leiba Rodman (1995). *Algebraic Riccati Equations*. Clarendon Press -Oxford. New York - USA.

Landelius, T. and H. Knutsson (1996). Greedy adaptive critics for LQR problems: Convergence proofs. Report LiTH-ISY-R-1896. Computer Vision Laboratory. SE-581 83 Linköping, Sweden.

Lee, Jae Young, Jin Bae Park and Yoon Ho Choi (2009). Model-free approximate dynamic programming for continuous-time linear systems. In: *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*. pp. 5009 – 5014.

Lendaris, G.G. (2009). A retrospective on adaptive dynamic programming for control. In: *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. Vol. 0. pp. 1750 –1757.

Lewis, F. L. and K. G. Vamvoudakis (2010*a*). Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using

measured output data. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **PP**(99), 1 –12.

Lewis, F.L. and D. Vrabie (2009*a*). Reinforcement learning and adaptive dynamic programming for feedback control. *Circuits and Systems Magazine, IEEE* **9**(3), 32 –50.

Lewis, F.L. and Kyriakos G. Vamvoudakis (2010*b*). Optimal adaptive control for unknown systems using output feedback by reinforcement learning methods. In: *Control and Automation (ICCA), 2010 8th IEEE International Conference on*. pp. 2138 –2145.

Lewis, Frank L. and Draguna Vrabie (2009*b*). Adaptive dynamic programming for feedback control. In: *Asian Control Conference, 2009. ASCC 2009. 7th*. pp. 1402 –1409.

Lewis, Frank L. and Vassilis L. Syrmos (1995). *Optimal Control*. John Wiley and Sons, Inc.. USA.

Meyer, Gerben G., Kary Främling and Jan Holmström (2009). Intelligent products: A survey. *Comput. Ind.* **60**(3), 137–148.

Murray, J.J., C.J. Cox, G.G. Lendaris and R. Saeks (2002). Adaptive dynamic programming. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*.

Powell, Warren B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience.

Si, Jennie, Andrew G. Barto, Warren Buckler Powell and Don Wunsch (2004). *Handbook of Learning and Approximate Dynamic Programming (IEEE Press Series on Computational Intelligence)*. Wiley-IEEE Press.

Stevens, Brian L. and Frank L. Lewis (1992). *Aircraft control and simulation / Brian L. Stevens, Frank L. Lewis*. Wiley, New York :.

Vamvoudakis, K.G. and F.L. Lewis (2009). Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem. In: *Neural*

Networks, 2009. IJCNN 2009. International Joint Conference on. pp. 3180 –3187.

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

Wang, Fei-Yue, Huaguang Zhang and Derong Liu (2009). Adaptive dynamic programming: An introduction. *Computational Intelligence Magazine, IEEE* **4**(2), 39 –47.

Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD thesis. Harvard University. Cambridge, MA.

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

Werbos, Paul J. (1990). Neural networks for control. Chap. A menu of designs for reinforcement learning over time, pp. 67–95. MIT Press. Cambridge, MA, USA.

Werbos, P.J. (1989). Neural networks for control and system identification. In: *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on*. pp. 260 –265 vol.1.

Yu, Jiang and Jiang Zhong-Ping (2010). Approximate dynamic programming for output feedback control. In: *Control Conference (CCC), 2010 29th Chinese*. pp. 5815 –5820.