UNIVERSIDADE ESTADUAL DO MARANHÃO

CENTRO DE CIÊNCIAS TECNOLÓGICAS

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO E SISTEMAS - PECS

JOELSON MILLER BEZERRA DE SOUSA

ESTRATÉGIAS DE APRENDIZADO POR REFORÇO PARA CONTROLE ÓTIMO *ONLINE* DE MANIPULADORES ROBÓTICOS

SÃO LUÍS

2022

JOELSON MILLER BEZERRA DE SOUSA

ESTRATÉGIAS DE APRENDIZADO POR REFORÇO PARA CONTROLE ÓTIMO *ONLINE* DE MANIPULADORES ROBÓTICOS

Texto de dissertação apresentado ao Programa de Pós-Graduação em Engenharia da Computação e Sistemas da Universidade Estadual do Maranhão como requisito para obtenção do título de mestre em Engenharia da Computação.

Orientadora: Profa. Dra. Patrícia Helena Moraes Rêgo

SÃO LUÍS

2022

Sousa, Joelson Miller Bezerra de.

Estratégias de aprendizado por esforço para controle ótimo *online de* manipuladores robóticos / Joelson Miller Bezerra de. – São Luís, 2022.

97 f

Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia de Computação e Sistemas, Universidade Estadual do Maranhão, 2022.

Orientadora: Profa. Dra. Patrícia Helena Moraes Rêgo.

1.Aprendizado por esforço. 2.Manipuladores robóticos. 3.Controle ótimo.

JOELSON MILLER BEZERRA DE SOUSA

ESTRATÉGIAS DE APRENDIZADO POR REFORÇO PARA CONTROLE ÓTIMO *ONLINE* DE MANIPULADORES ROBÓTICOS

Texto de dissertação apresentado ao Programa de Pós-Graduação em Engenharia da Computação e Sistemas da Universidade Estadual do Maranhão como requisito para obtenção do título de mestre em Engenharia da Computação.

Aprovado em: 27/10/2022

Banca Examinadora

gov.br Documento assinado digitalmente
PATRICIA HELENA MORAES REGO
Data: 27/11/2022 10:37:05-0300
Verifique em https://verificador.iti.br

Profa. Patrícia Helena Moraes Rêgo, Dra. (Orientadora)
Doutora em Engenharia da Eletricidade
Universidade Estadual do Maranhão (UEMA)

gov.br Documento assinado digitalmente
HENRIQUE MARIANO COSTA DO AMARAL
Data: 04/01/2023 09:40:34-0300
Verifique em https://verificador.iti.br

Prof. Henrique Mariano Costa do Amaral, Me. (Examinador Interno)
Mestre em Engenharia de Sistemas e Computação
Doutorando em Engenharia Mecânica
Universidade Estadual do Maranhão (UEMA)

gov.br Documento assinado digitalmente
VILEMAR GOMES DA SILVA
Data: 22/12/2022 19:40:13-0300
Verifique em https://verificador.iti.br

Prof. Vilemar Gomes da Silva, Dr. (Examinador Externo)
Doutor em Engenharia Elétrica
Universidade Federal do Maranhão (UFMA)

À minha mãe (em memória), Eusimar, pela dedicação, amor e carinho, pelos quais sou eternamente grato.

AGRADECIMENTOS

Eu gostaria de prestar meus agradecimentos, em destaque, à minha mãe (em memória) pela dedicação, pelo amor e pelos ensinamentos oferecidos a mim. Agradeço por sempre me apoiar em minhas decisões e pela compreensão nos momentos difíceis. Grande parte do conquistei e do que sou é devido a você. Muito obrigado!

À minha orientadora, Profa. Patrícia Helena Moraes Rêgo, pela confiança, empenho e orientação valiosa e necessária. Agradeço também pela motivação e instruções fornecidas, essenciais para a conclusão deste trabalho.

Aos meus familiares, em especial a minha irmã, Jéssica, amigos e todos aqueles que contribuíram e me acompanharam nessa jornada.

À Universidade Estadual do Maranhão (UEMA) e ao Programa de Pós-Graduação em Engenharia da Computação e Sistemas (PECS) pelo apoio financeiro através da bolsa de mestrado.

RESUMO

As imprecisões e incertezas nos parâmetros de um manipulador robótico, tais como variações na carga de trabalho, medidas imprecisas da massa e/ou inércia dos elos, folgas ou atritos desconhecidos nas engrenagens, entre outras, afetam, muitas vezes de forma significativa, o desempenho do manipulador, ocasionando erros de regime e de seguimento de trajetória. Controladores adaptativos apresentam-se como uma boa alternativa para esses sistemas, pois possuem como principal característica a capacidade de aprenderem *online* usando estimação de parâmetros em tempo real. No entanto, controladores adaptativos geralmente não são projetados com a qualidade de serem ótimos no sentido de minimizarem funções de custo, conforme definido no contexto de controle ótimo, e, desta forma, não são viáveis para aplicações onde o uso de estratégias ótimas de controle é requerido. Neste trabalho, propõe-se uma abordagem unificada de controle adaptativo e controle ótimo que tem por base conceitos e métodos de aprendizado por reforço, tendo em vista o desenvolvimento de algoritmos para o projeto de sistemas de controle ótimo *online* com aplicações em controle de manipuladores robóticos. Uma estrutura paramétrica é utilizada para aproximar a função valor a fim de contornar o problema da maldição da dimensionalidade. A estimação desses parâmetros será realizada através do estimador dos Mínimos Quadrados Recursivos (*Recursive Least Squares* - RLS) a cada passo de tempo. Já em relação a política de controle, duas abordagens serão implementadas na etapa de atualização: a melhoria de política aproximada, em que uma representação via aproximadores de funções é utilizada, e a melhoria de política exata, onde as ações de controle são calculadas exatamente através da função valor. A principal vantagem da metodologia de controle proposta é que, para sua implementação, não é necessária nenhuma informação prévia dos parâmetros do manipulador, somente as medições dos estados e do sinal de controle são usadas. A avaliação dos esquemas de controle é feita em um modelo robótico UR10 do simulador V-REP para as tarefas de regulação, rastreamento e variações na carga de trabalho.

Palavras-Chaves: Aprendizado por Reforço; Manipuladores Robóticos; Controle Ótimo; Controle Adaptativo.

ABSTRACT

The inaccuracies and uncertainties in the parameters of a robot manipulator, such as payload variations, inaccurate measurements of the mass and/or inertia of the links, unknown backlash or friction in gears, among others, can significantly affect the robot performance, causing steady-state and trajectory following errors. Adaptive controllers are a suitable alternative for these systems since their main feature is the capability to learn online using real-time parameter estimation. Nevertheless, adaptive controllers are not usually designed to be optimal in the sense of minimizing cost functions as defined in the optimal control context, and thus, are not suitable to applications in which the use of optimal control strategies is required. In this work, an unified approach of adaptive control and optimal control based on concepts and methods of reinforcement learning is proposed aiming at the development of algorithms for the design of online optimal control systems with applications in robotic manipulator control. A parametric structure is used to approximate the value function in order to avoid the curse of dimensionality problem. The estimation of these parameters will be performed through the Recursive Least Squares (RLS) estimator at each time step. With respect to the control policy, two approaches will be implemented in the updating step: approximate policy improvement, where a representation by function approximators is used, and exact policy improvement, where control actions are calculated exactly through the value function. The main advantage of the proposed control methodology is that no prior information of the manipulator parameters is needed for its implementation, only the measurements of the states and the control signal are used. The evaluation of the control schemes is carried out in a UR10 robotic model on V-REP simulator for the regulation and tracking tasks, and payload variations.

Keywords: Reinforcement Learning, Robot Manipulator, Optimal Control, Adaptive Control.

LISTA DE FIGURAS

LISTA DE SIGLAS

AC	Ator-Crítico
API	*Application Programming Interface*
CC	Corrente Contínua
HJB	Hamilton-Jacobi-Bellman
LQR	*Linear Quadratic Regulator* (Regulador Linear Quadrático)
MATLAB	*Matrix Laboratory*
MRAC	*Model Reference Adaptive* Control (Controle Adaptivo por Modelo de Referência)
PD	Proporcional-Derivativo
PDM	Processo de Decisão Markoviano
PID	Proporcinal-Integrativo-Derivativo
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
RBF	*Radial Basis Function* (Função de Base Radial)
RNA	Rede Neural Artificial
RL	*Reinforcement Learning* (Aprendizado por Reforço)
RLS	*Recursive Least-Squares* (Mínimos Quadrados Recursivos)
ROS	*Robot Operating System*
SARSA	*State-Action-Reward-Next State-Next Action* (Estado-Ação-Recompensa- Estado Seguinte-Ação Seguinte)
SCARA	*Selective Compliance Assembly Robot Arm* (Braço Robótico para Montagem de Conformidade Seletiva)
TCP	Transmission Control Protocol (Protocolo de Controle de Transmissão)
TD	*Temporal Diference* (Diferença Temporal)
V-REP	*Virtual Robotics Experimentation Plataform*

SUMÁRIO

1 INTRODUÇÃO

A robótica é o ramo da ciência que tem como objetivo desenvolver sistemas automáticos para auxiliar ou substituir o homem nas mais diversas tarefas. O projeto, a construção e a manipulação de robôs são atividades que podem requerer habilidades em diversas áreas da tecnologia como teoria de controle, microeletrônica, inteligência artificial, mecânica, computação etc. (DUDEK & JENKIN, 2010).

Os robôs podem ser classificados a partir de diversos critérios. Do ponto de vista mecânico, por exemplo, eles podem ser divididos entre robôs móveis e robôs manipuladores (BEN-ARI, M. & MONDADA, F., 2018). Os robôs móveis são aqueles que são capazes de se locomover em espaços de grande escala, seja por meio terrestre, aéreo ou aquático (DUDEK & JENKIN, 2010). Já os manipuladores (ou articuladores) possuem sua locomoção limitada aos movimentos do mecanismo fixado à sua base. A estrutura mecânica de um robô articulador consiste em um conjunto de elos, ou articulações, conectados por juntas capazes de realizar movimentos, sendo este último o objeto de estudo desse trabalho (VIEIRA, 2005).

Na automação industrial, os robôs manipuladores tem se destacado nos últimos anos por proporcionarem mais exatidão, velocidade e segurança na execução de tarefas, além do fato que o seu custo de aquisição vem diminuindo constantemente (CRAIG, 2013). Os manipuladores também têm se destacado em atividades médicas, como cirurgias, possibilitando incisões menores, diminuindo o tempo de operação e proporcionando maior estabilidade em comparação com a mão humana (KUCUK & GUNGOR, 2016).

Estes avanços também são consequência de vários estudos e pesquisas direcionadas à área de robôs manipuladores que possibilitam o desenvolvimento de novas tecnologias de controle (BILLARD e KRAGIC, 2019). Basicamente, o sistema de controle de um manipulador é responsável por gerar comandos para os atuadores do robô de forma que o mecanismo execute um movimento pretendido. Desta forma, o objetivo é fazer o articulador seguir um sinal de referência usando uma lei de controle adequada, o modelo numérico do mecanismo e informações extraídas dos sensores (ZHANG e WEI, 2017).

Um tipo de tarefa comumente utilizada em manipuladores na indústria trata-se de um simples deslocamento entre duas posições, conhecido como apanhar e colocar (*pick-and-place*), em que o mecanismo deve agarrar um objeto em uma posição inicial e então dispô-lo em uma configuração final desejada (AWELEWA *et al.* 2013; CRAIG, 2013; GUALTIERI e PLATT, 2021). Entretanto existem tarefas em que se necessita que o robô siga um conjunto contínuo de

pontos (seguimento de trajetória), ou seja, não apenas as posições iniciais e finais são importantes, mas também todas as posições intermediárias (ZHANG e WEI, 2017). Pintura e soldagem são exemplos comuns nas indústrias de tarefas de seguimento de trajetória, que exigem maior grau de exatidão e, muitas vezes, maior velocidade de movimento (PANE *et al.*, 2019).

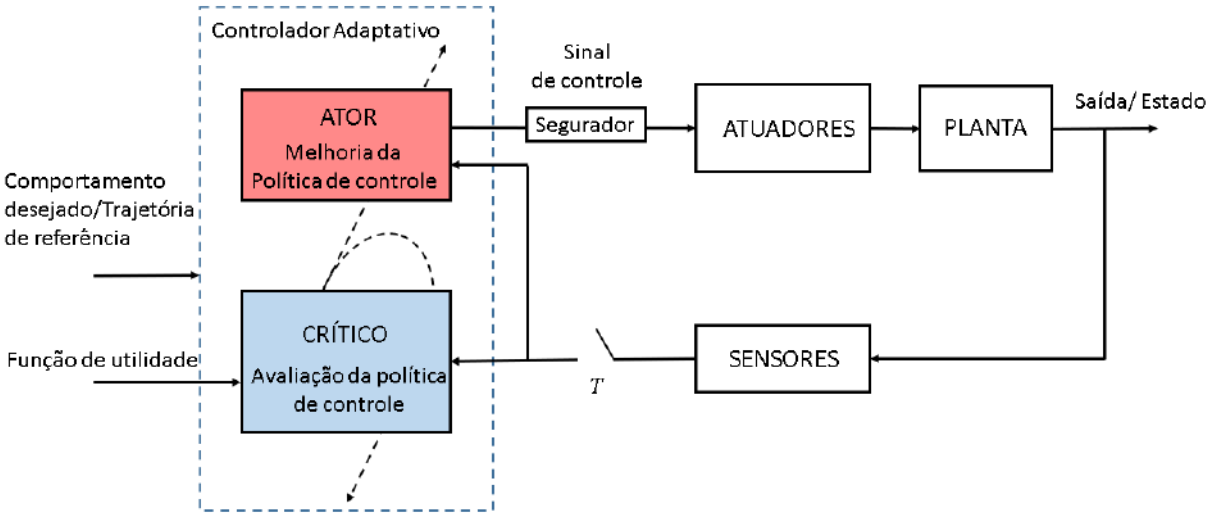
Demais tarefas, como montagem de peças, polimento e rebarbagem, não só se preocupam com o controle de velocidade e posição, mas também com o surgimento de possíveis pontos de instabilidade ou restrições de movimento devido a interação do manipulador com o ambiente. Desta forma, nesses processos, é desejável obter um sistema de controle capaz de lidar com o impacto do contato e, simultaneamente, com a força aplicada (CRAIG, 2013).

De modo a obter controladores mais eficientes em aplicações robóticas, metodologias de controle não-linear vêm sendo desenvolvidas, em que muitas dessas técnicas são baseadas em princípios de Inteligência Artificial (GONÇALVES, 2016). Neste contexto, este trabalho tem como objetivo desenvolver uma metodologia e algoritmos de controle de sistemas robóticos baseados em aprendizado por reforço (do inglês *Reinforcement Learning* - RL). Estes esquemas de aprendizado possibilitam a obtenção de controladores com características de adaptação (capazes de compensar variações nos parâmetros da planta) e otimização (minimizando critérios de desempenho estabelecidos pelo projetista).

Um esquema de aprendizado por reforço é ilustrado na Figura 1.1, onde se observa um sistema de controle com realimentação dentro de um contexto relacionado a controle adaptativo realizado pelos componentes Ator e Crítico. Este mecanismo de aprendizado possui duas etapas: a avaliação da política de controle (ação de controle), realizada pelo Crítico, seguida da melhoria da política de controle, executada pelo Ator. A vantagem dessa metodologia é a possibilidade de sua aplicação *online*, onde o comportamento ótimo pode ser alcançado observando as respostas do processo às ações de controle (LEWIS *et al.*, 2012a). Na literatura existem exemplos da estrutura Ator-Crítico (AC) aplicada a sistemas robóticos (KHAN *et al.*, 2012; LUY *et al.*, 2014; PANE *et al.*, 2016; PANE *et al.,* 2019; KAMBOJ *et al.*, 2020).

Visto os promissores resultados dos métodos de aprendizado por reforço na área de controle (KOBER *et al.*, 2013), este trabalho tem como proposta desenvolver algoritmos de aprendizagem dedicados ao controle de manipuladores robóticos.

Figura 1.1: Diagrama de blocos em alto nível da estrutura de aprendizado por reforço Ator-Crítico



Fonte: Autor

1.1 MOTIVAÇÃO E JUSTIFICATIVA

O desenvolvimento de estratégias de controle para manipuladores apresenta dificuldades decorrentes das próprias características do sistema, isto é, um robô articulador é um sistema complexo, dinamicamente não-linear, multivariável e com parâmetros incertos ou que variam no tempo (FATEH e FATEH, 2020). Pode-se citar como exemplos de parâmetros imprecisos: a massa e inércia dos elos, atritos ou folgas nas engrenagens das juntas, variações nas cargas de trabalho, localização do centro de massa (que pode mudar quando o robô estiver com carga), entre outras. Estas imprecisões paramétricas resultam em perdas de exatidão e velocidade nos movimentos do manipulador, que em determinadas aplicações é altamente indesejável. Já a dinâmica não-linear da planta dificulta o desenvolvimento de uma lei de controle adequada para diversos pontos de operação (CRAIG, 2013).

Controladores convencionais de realimentação, como o PID (Proporcional-Integral-Derivativo), são vastamente utilizados na indústria por serem simples, fáceis de implementar e por apresentarem bom desempenho em diversas aplicações. Entretanto, este esquema de controle possui resultados insatisfatórios quando aplicado a sistemas com não linearidades (KONSTANTOPOULOS e BALDIVIESO-MONASTERIOS, 2020).

Dentre os controladores clássicos aplicados a manipuladores existem os baseados em modelo (cinemático e/ou dinâmico para controle de posição, velocidade e força). Porém, estas abordagens necessitam do conhecimento completo das equações que descrevem o sistema,

sendo elas bastante complexas e com parâmetros que muitas vezes são incertos. A complexidade do modelo cresce também com o aumento de juntas e elos do manipulador, aumentando o custo computacional para solucionar estas equações (MOOSAVI *et al*., 2022).

Para sistemas com não-linearidades suaves, é frequentemente utilizada a abordagem de linearização do modelo na região de operação desejada. Entretanto, esta técnica não pode ser aplicada a todos os manipuladores robóticos, pois estes podem se mover vastamente por sua área de trabalho e, portanto, pode não existir linearização possível que possa ser calculada para todas as regiões (CRAIG, 2013).

Desta forma, as técnicas de controle adaptativo se apresentam como uma alternativa para contornar estes problemas. Esta abordagem permite compensar, de forma *online*¸ as variações e incertezas paramétricas do sistema garantindo que os critérios de desempenho desejados sejam alcançados (SUN *et al*., 2020). Entretanto este método garante somente a exatidão e a estabilidade do sistema, sem considerar a otimização da ação de controle (ZHU, 2020). Existem aplicações industriais em que é bastante vantajoso diminuir o tempo e a energia necessários para executar uma ação, de forma a diminuir o custo e maximizar a taxa de produção. Portanto, podem ser aplicadas técnicas de controle ótimo, que consistem basicamente em calcular uma lei de controle de maneira a minimizar um critério de desempenho.

Muitos procedimentos de otimização utilizados em controle determinam a lei de controle ótima *off-line* exigindo o conhecimento *a priori* do modelo da planta. Porém, os parâmetros de um manipulador variam e, portanto, estes métodos se tornam ineficientes. Desta forma, para garantir o desempenho ótimo do controlador, a lei de controle otimizada deve ser calculada em tempo real baseando-se nos estados atuais do processo (BOUKENS e BOUKABOU, 2013). Nesse caso, uma abordagem conjunta das técnicas de controle adaptativo e controle ótimo é desejada.

Os métodos de controle adaptativo podem ser divididos em duas abordagens: os controladores diretos e os controladores indiretos. Os métodos indiretos são executados em duas etapas, em que primeiramente estima-se um modelo para o sistema e em seguida usa-se essa informação para calcular o controle. Os métodos diretos determinam a lei de controle sem a necessidade das equações que regem o comportamento do sistema (VRABIE *et al*., 2013; LEWIS *et al.*, 2012a).

Controladores ótimos adaptativos podem ser obtidos através dos métodos de aprendizado por reforço, que se trata de uma área do aprendizado de máquina baseado nos estudos de comportamento animal e psicologia cognitiva (VRABIE *et al*., 2013). Esta

abordagem pode ser caracterizada como um tipo de controlador ótimo adaptativo, visto que a solução de controle ótimo pode ser obtida sem a necessidade do modelo dinâmico do sistema (VRABIE *et al*., 2013).

Diante do exposto, este trabalho tem como propósito desenvolver algoritmos e métodos aplicados ao controle ótimo de robôs manipuladores que possam compensar as variações nos parâmetros do sistema baseando-se em técnicas de aprendizado por reforço.

1.2 OBJETIVOS

O objetivo geral deste trabalho é desenvolver uma metodologia e algoritmos de controle ótimo *online* com base em paradigmas de aprendizado por reforço direcionados a robôs manipuladores.

De forma a alcançar o objetivo geral, apresentam-se os objetivos específicos a serem seguidos:

- Realizar revisão bibliográfica sobre o tema em estudo no intuito de investigar o estado da arte.
- Desenvolver modelos parametrizados para representar as políticas de controle e aproximar a função valor.
- Projetar e simular um controlador de estrutura parametrizada com garantias de desempenho ótimo.
- Modelar um manipulador robótico em um *software* de simulação a fim de fornecer um ambiente para os experimentos com o controlador proposto.
- Avaliar o desempenho dos algoritmos de controle quanto a sua capacidade de rastrear diferentes sinais de referência e de compensar variações na carga de trabalho.

1.3 ORGANIZAÇÃO DO TRABALHO

Esta dissertação está dividida em sete capítulos incluindo a Introdução. No Capítulo 2, é apresentada uma revisão bibliográfica a respeito das estratégias de controle aplicadas a manipuladores. A modelagem cinemática e dinâmica dos articuladores, o desenvolvimento das equações dinâmicas de robô com dois graus de liberdade e uma discussão sobre os atuadores e sensores empregados na robótica industrial são realizados no Capítulo 3. O Capítulo 4, por sua vez, é destinado à exposição dos conceitos fundamentais do aprendizado por reforço, assim

como os seus principais métodos e algoritmos. Além disto, serão discutidos outros tópicos relevantes ao projeto como aproximadores de função, redes neurais e controle ótimo. Os aspectos relacionados ao sistema de controle implementado neste trabalho serão expostos no Capítulo 5. Os resultados de simulação serão expostos no Capítulo 6 e as considerações finais serão apresentadas no Capítulo 7.

2 REVISÃO BIBLIOGRÁFICA

O problema de controle de manipuladores tem sido extensivamente estudado nas últimas décadas, resultando em diversas técnicas e métodos com a finalidade de resolver as variadas tarefas que podem ser executadas pelos articuladores.

Rubio *et al.* (2016) aplicaram com sucesso um controlador PID antivibratório a um manipulador, demostrando a estabilidade semiglobal da técnica para um determinado conjunto de erros iniciais para variáveis de velocidade e de posição.

A técnica de linearização por realimentação entrada-saída foi aplicada por Montoya-Cháirez e Moreno-Valenzuela (2021) para cancelar as não linearidades do modelo dinâmico de um mecanismo robótico com juntas flexíveis.

Melkou e Hamerlain (2019), por sua vez, recorreram a técnica de controlador robusto por modos deslizantes de ordem superior para lidar com as não linearidades e incertezas presentes no modelo de um braço robótico com músculos artificiais pneumáticos.

Paul *et al.* (2020) aplicaram um controlador PID com ganhos sintonizados via lógica *fuzzy* tipo-2 em um manipulador agrícola. O método de controle ativo adotado mostrou-se eficiente no objetivo de reduzir as vibrações do mecanismo.

Nawrocka *et al.* (2019) empregaram um controle não-linear baseado em redes neurais artificiais. Nesse esquema, duas redes neurais foram implementadas para controlar cada junta de um manipulador com dois graus de liberdade. O treinamento foi realizado *offline* utilizando o algoritmo de retropropagação de Lavenberg-Marquardt. O método aplicado mostrou-se estável e mais acurado que o controle via PID convencional, porém apresentando um comportamento levemente vibratório.

Li *et al*. (2016) propuseram um controle cinemático de manipuladores redundantes baseado em redes neurais recorrentes. Esta abordagem não necessita de treinamento antes da aplicação formal, mas faz uso do conhecimento parcial do modelo da planta. A estabilidade global do método foi comprovada neste trabalho e maior robustez e acurácia foram alcançadas nas simulações numéricas com o robô PUMA 560.

Os métodos de controle adaptativo possuem uma vasta literatura técnica em relação a aplicação em manipuladores robóticos por estes fornecerem bom desempenho em atender a altas demandas de precisão, exatidão, operabilidade, confiabilidade e flexibilidade. (AJWAD *et al*., 2015).

Dubowsky e Desforges (1979) são os pioneiros em empregar técnicas de controle adaptativo em robôs articulados. O controlador escolhido pelos autores foi o Sistema Adaptativo por Modelo de Referência (do inglês *Model Reference Adaptive Control* - MRAC) até então bastante estudado e aplicado a outros processos não lineares. A ideia deste controlador consiste em calcular um sinal de controle para o robô baseado no erro entre a saída da planta e a saída de um modelo especificado (com as características de controle desejadas) excitado pelo sinal de referência. Os autores, neste caso, utilizaram um modelo simplificado de um manipulador, considerando as juntas do mecanismo dinamicamente desacopladas.

Nos anos seguintes diversos pesquisadores realizaram estudos aplicando o método MRAC. Maliotis (1991) desenvolveu um controlador MRAC baseado no critério de estabilidade de Lyapunov. Já Al-Olimat e Ghandakly (2002) propuseram um algoritmo MRAC usando lógica *fuzzy* para ajustes dos parâmetros do modelo de referência para lidar com as condições ambientais complexas que o manipulador pode enfrentar. Em Zhang e Wei (2016), um controlador híbrido PID-MRAC foi projetado para controlar manipuladores com múltiplos graus de liberdade. Alqaudi *et al*. (2016), por sua vez, desenvolveram um algoritmo MRAC incluindo uma rede neural artificial (RNA) para aplicações em que existe a interação física entre o robô e o homem.

Em 2022, Yilmaz *et al*. propuseram um controlador *fuzzy*-adaptativo. A lei adaptativa, baseada em lógica *fuzzy* combinada com uma malha PD (Proporcional-Derivativo), foi desenvolvida para compensar as incertezas paramétricas do modelo cinemático do braço mecânico. Neste trabalho, a propriedade de autoajuste da estratégia proposta é obtida ajustando os centros e as larguras das funções de pertinência por meio de leis de atualização adaptativas dinâmicas. (YILMAZ *et al*., 2022).

Freire *et al.* (2018) realizaram um projeto de controle adaptativo aplicado em braços robóticos utilizando rede neurais. De forma a compensar as incertezas e não linearidades do mecanismo, uma rede neural de base radial foi usada para aproximar o modelo dinâmico do robô. Já o controlador sugerido pelos autores foi baseado no controle PID, em que os ganhos K_P, K_I e K_D estão incluídos na formulação matemática de uma rede neural e são ajustados usando o método gradiente descendente, semelhante aos pesos de uma rede do tipo *perceptron*. A estratégia proposta foi aplicada com sucesso em um manipulador do tipo SCARA (Braço Robótico para Montagem de Conformidade Seletiva) com dois graus de liberdade, demostrando um desempenho melhor do que controle PID convencional e capacidade de adaptação às não-linearidades, perturbações e incertezas do modelo.

Wu *et al*. (2019) implementaram um controlador adaptativo baseado em redes neurais e aprendizagem iterativa. As incertezas do modelo do robô manipulador são aproximadas por meio de uma rede neural de base radial, enquanto o controle por aprendizagem iterativa diminui os erros de rastreamento.

Kai e Huang (2014) aplicaram um controlador ótimo adaptativo a um sistema robótico articulado através do método LQR (Regulador Quadrático Linear). Entretanto, este controlador é majoritariamente usado em sistemas lineares. Portanto, para tornar prática esta aplicação o modelo não-linear do manipulador deve ser linearizado, o que é alcançado, neste trabalho, através de uma aproximação realizada por uma RNA. A estabilidade do método proposto é garantida pelo teorema de Lyapunov.

Em HU *et al.* (2012) utiliza-se de um controlador *backstepping* para lidar com as perturbações externas e incertezas dinâmicas e cinemáticas do manipulador. Esta técnica foi direcionada a tarefa de seguimento de trajetória.

De forma a obter estratégias ótimas e/ou que não necessitem do conhecimento completo das dinâmicas dos sistemas na execução de tarefas, métodos de aprendizado por reforço passaram a serem estudadas em aplicações robóticas. Nas últimas duas décadas, um vasto conjunto de autores voltaram seus esforços para tornar essas técnicas práticas.

No final dos anos 90, Kuan e Young (1998) propuseram uma estratégia que envolve métodos de aprendizado por reforço e controle robusto. Neste esquema, mecanismos de aprendizado determinam, baseando-se no erro entre o estado atual e o objetivo da tarefa, o sinal de comando que alimenta o controlador. Este, por sua vez, é um controlador robusto que garante a estabilidade do sistema na presença de incertezas paramétricas e perturbações externas. A eficiência do esquema proposto foi demostrada através de simulações baseadas em tarefas de rebarbagem.

Peters *et al.* (2003) investigaram em seu artigo diversos métodos de aprendizado por reforço para aplicações em robôs humanoides. A estrutura Ator-Crítico Natural apresentou resultados satisfatórios com rápida convergência.

Já Shah e Gopal (2009) desenvolveram um controlador baseado no algoritmo *Q-Learning*. Normalmente, este método é aplicado a sistemas discretos, portanto para tornar prática a aplicação desse método no espaço contínuo, diversos métodos de aproximação de funções foram investigados, como redes neurais, árvores de decisão e lógica *fuzzy*. O controlador *fuzzy Q-learning* foi o método destacado pelos pesquisadores neste artigo.

Khan *et al.* (2012) apresentaram uma visão geral sobre os métodos de aprendizado por reforço para obtenção de estratégias de controle ótimo adaptativo aplicados a sistemas robóticos. Ao final um esquema baseado no método programação dinâmica aproximada foi proposto para obtenção de políticas de controle ótimas *online.* A estratégia foi implementada no braço de um robô humanoide (Bristol Elumotion-Robotic-Torso II) considerando um caso sem restrições e outra com restrições de movimento.

Pane *et al.* (2016) em seu artigo forneceram validação experimental de um compensador baseado no esquema de aprendizado Ator-Crítico para aprimorar o desempenho de um robô manipulador. O método proposto dispensa a necessidade de aprender o modelo do sistema e pode ser utilizado em qualquer controlador por realimentação (PID, LQR etc.). A validação do método foi demonstrada através de experimentos com o robô manipulador com seis graus de liberdade.

Para controlar um braço robótico com parâmetros desconhecidos e sujeito a zonas mortas desconhecidas, Hu e Si (2018) implementaram um controle baseado em aprendizado por reforço Ator-Crítico com observador de estado via rede neural.

Khan *et al*. (2019) propuseram um controle de complacência adaptativo ótimo para um dispositivo robótico de auxílio à locomoção. O esquema de controle sugerido é fundamentado em Aprendizado Q e programação dinâmica aproximada. Esse esquema é completamente independente de modelo dinâmico e emprega realimentação da posição e velocidade da junta, bem como o torque detectado da junta (aplicado pelo usuário durante a caminhada) para controle de complacência. A eficiência do controlador é testada em simulação em um modelo de dispositivo robótico de auxílio à locomoção.

A técnica de Aprendizado Q com aproximação via lógica *fuzzy* foi empregado por Kukker e Sharma (2021) em um manipulador robótico. De forma a contornar a escolha determinística da ação de controle desta técnica, que não é eficiente nos casos de sistemas não lineares e altamente acoplados, foi adotado algoritmo genético como otimizador estocástico para selecionar a ação ótima.

Kamboj *et al.* (2020) propuseram uma estratégia de controle ótimo cinemático em tempo discreto para um manipulador usando o esquema Ator-Crítico. Esta abordagem distingue-se em relação as soluções já existentes na literatura devido aos ganhos da rede neural da estrutura Crítico serem atualizados por uma lei derivada do teorema de estabilidade de Lyapunov. Desta forma, o controlador proposto tem estabilidade analítica garantida. A metodologia exposta foi aplicada a um modelo 3D de um manipulador com seis graus de liberdade em experimentos

realizados em um *software* de simulação. Em seguida, implementou-se a estratégia em um robô real do mesmo modelo do simulado.

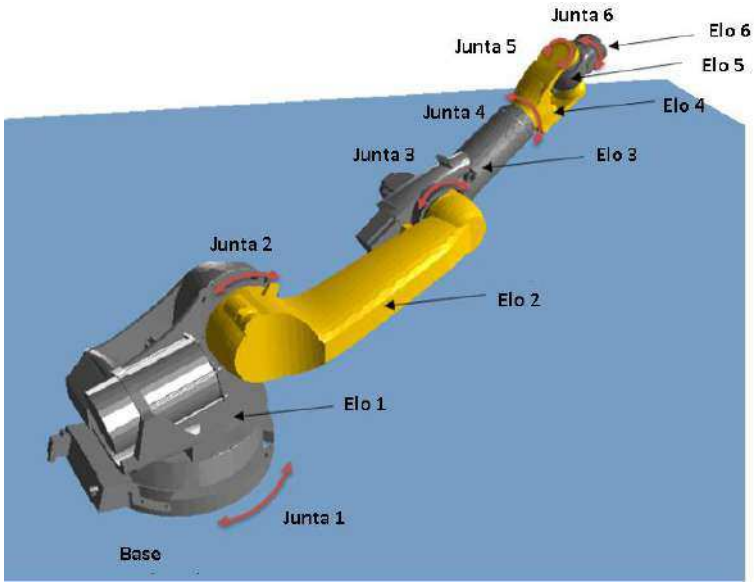
Um controlador de rastreamento baseado em RL Ator-Crítico para um manipulador também foi estudado por Cao *et al*. (2021). Neste trabalho, a técnica de modos deslizantes é utilizada para que a ação obtida pelo esquema Ator-Crítico garanta a convergência do erro de rastreamento em um tempo fixo. Além disso, um compensador *antiwindup* foi projetado para lidar com os efeitos da saturação do atuador da junta.

Técnicas de controle ótimo adaptativo via programação dinâmica aproximada aplicados a manipuladores robótico são encontrados na literatura (MENON *et al.*, 2019; MA *et al*.; NAM *et al*.; OUYANG *et al*.; YANG *et al*., 2020; HU *et al*., 2021).

3 DESCRIÇÃO DE UM MANIPULADOR ROBÓTICO

Um manipulador robótico, ou robô articulado, é formado por um conjunto de corpos individuais conectados entre si formando uma cadeia cinemática capaz de realizar tarefas através da interação com o ambiente. As duas peças fundamentais que compõem um robô articulado são os elos, ou articulações, e as juntas. Os elos são as estruturas físicas (rígidas ou flexíveis) que compõem o robô. Já as juntas são responsáveis por promover o movimento relativo entre as articulações por meio de acionadores. A Figura 3.1 ilustra uma sequência de elos e juntas de um braço robótico.

Figura 3.1: Elos e Juntas de um robô articulado

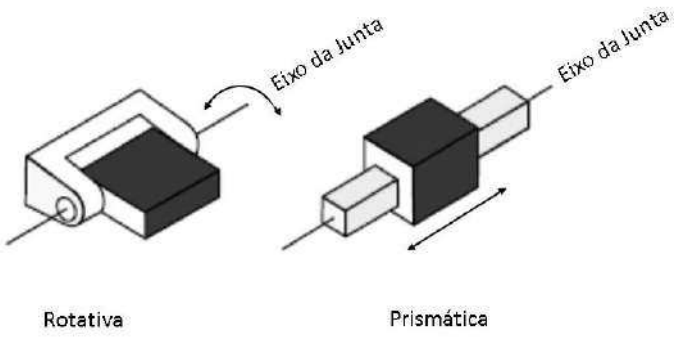


Fonte: (ABBAS, 2018)

As juntas são classificadas de acordo com a liberdade de movimento viabilizada por ela. Os tipos mais comuns encontrados na indústria são: as juntas rotacionais e as prismáticas. A Figura 3.2 ilustra estes dois tipos, onde é indicado o movimento realizado em relação ao eixo da junta.

As extremidades do robô articulador são denominadas de base e efetuador. A base fica ligada ao primeiro elo e fixa o mecanismo em algum ponto no espaço da tarefa. O efetuador (ou atuador) é uma ferramenta conectada ao último elo do articulador e é por este ponto que há a interação com ambiente. O tipo de atuador instalado dependerá da tarefa a ser executada.

Figura 3.2: Juntas Rotativas e Prismáticas



Fonte: http://www.cs.uu.nl/docs/vakken/moma/Robotics.pdf

3.1 MODELAGEM CINEMÁTICA

O estudo da cinemática tem como objetivo determinar o vetor posição e o vetor velocidade tanto do efetuador quanto das juntas de um manipulador. Essa análise não considera as forças que geram o movimento, centrando apenas em suas propriedades geométricas e temporais (CRAIG, 2013).

Os manipuladores são definidos como cadeias cinemáticas complexas, devido a isto, é necessário desenvolver técnicas para representar a posição de determinado ponto de seu mecanismo em função do tempo. Esta representação é escrita em função das variáveis do sistema, ou seja, nas medidas angulares das juntas rotativas e nas medidas de deslocamento das juntas prismáticas. Inicialmente, o procedimento fundamenta-se em definir sistemas de coordenadas associadas a cada elo e, assim, determinar a matriz de transformação que relaciona referenciais vizinhos (SILVA, 2017).

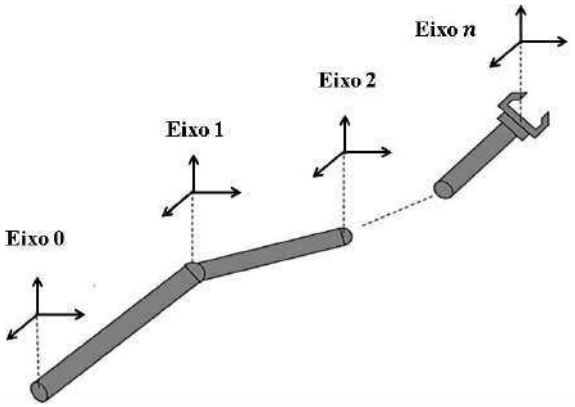
Os elos são numerados de 0 a n, no qual o elo 0 é normalmente afixado na base do robô. Um exemplo de enumeração dos elos é mostrado na Figura 3.3 na qual pode-se observar os eixos referenciais elencados e estabelecidos sobre a localização das juntas. A transformação entre sistemas de coordenadas consecutivos é definida pela matriz ${}_{i+1}^{i}T$, onde o sobrescrito se refere ao sistema de referência adotado para o elo i e o subscrito se refere ao sistema de referência do elo vizinho sucessor.

A matriz de transformação é constituída pelos elementos:

$$ {}_{i+1}^{i}T = \begin{bmatrix} {}_{i+1}^{i}R & {}^{i}P_{(i+1),o} \\ 0_{1\times 3} & 1 \end{bmatrix}, \tag{1} $$

em que ${}^{i}P_{(i+1),o}$ é o vetor de coordenadas que descreve a translação da origem do sistema $(i+1)$ com relação ao sistema de coordenadas (i) e ${}_{i+1}^{i}R$ é a matriz de rotação que representa a orientação do sistema de coordenadas $(i+1)$ tomando como referência o sistema de coordenadas (i).

Figura 3.3: Enumeração e afixação dos frames dos elos de manipulador

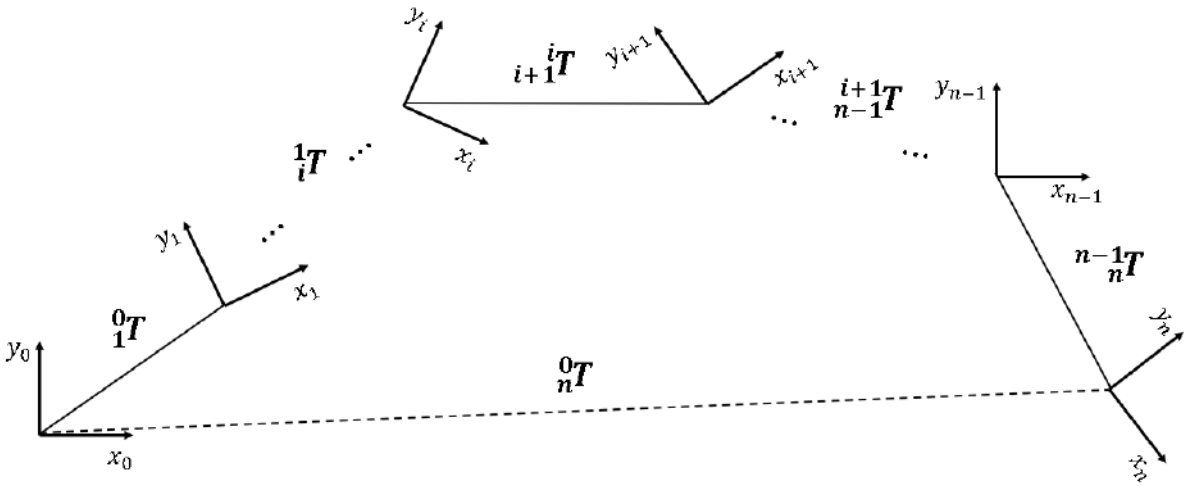


Fonte: (XU e QIAO, 2013)

Em seguida, essas transformações individuais podem ser utilizadas para calcular a posição e a orientação do elo n em relação ao elo 0, ou entre qualquer outro par de elos, através do produto de matrizes de transformação entre elos consecutivos (Figura 3.4). Desta forma, é possível localizar estes referenciais conforme a estrutura se move por meio de:

$$ {}_{n}^{0}T = {}_{1}^{0}T\ {}_{2}^{1}T\ \ldots {}_{n}^{n-1}T\ . \qquad (2) $$

Figura 3.4: Transformações genéricas de uma cadeia cinemática



Fonte: Autor

3.2 MODELAGEM DINÂMICA

A dinâmica dos manipuladores estuda a relação entre as forças aplicadas nos atuadores das juntas e o movimento do mecanismo. A formulação de Lagrange permite modelar o comportamento dinâmico de um corpo em termos das energias cinéticas e potenciais ao invés de considerar os momentos e forças aplicadas individualmente em cada junta.

A equação de Lagrange é expressa na Equação (3). O termo L é o lagrangeano que expressa a diferença entre a energia cinética $K(\cdot)$ e a energia potencial $U(\cdot)$ armazenada no mecanismo. Essa equação é escrita em termos das coordenadas generalizadas (q) do articulador e sua derivada no tempo $(\dot{q})$. O termo τ, por sua vez, representa o vetor generalizado de forças, incluindo as forças e os torque aplicados no sistema.

$$\tau = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} \tag{3}$$

$$L(\dot{q}, q) = K(\dot{q}, q) - U(q) \tag{4}$$

Para um elo rígido de um robô manipulador, a energia cinética pode ser escrita na forma:

$$k_i = \frac{1}{2} m_i v_{C_i}^T v_{C_i} + \frac{1}{2} \omega_i^T \, {}^{C_i}I_i \, \omega_i. \tag{5}$$

A energia cinética dispõe de duas componentes: uma relacionada a velocidade linear (v) e outra a velocidade angular (ω). O subscrito i indica os parâmetros e as variáveis relacionados ao i-ésimo elo do robô, m_i é massa e ${}^{C_i}I_i$ é a matriz de inércia tomando como referência o centro de massa do elo (C_i).

$$K(\dot{q}, q) = \sum_{i=0}^{n} k_i. \tag{6}$$

A energia potencial para um elo rígido pode ser expressa como:

$$u_i = m_i g^T P_{C_i} \tag{7}$$

$$U(q) = \sum_{i=0}^{n} u_i. \tag{8}$$

Neste caso, $U(q)$ depende da massa (m) de cada um dos elos, do vetor de gravidade (g) e da localização do centro de massa (P_C) de cada um dos elos relativos à base.

Após aplicar $L(q, \dot{q})$ na equação de Lagrange, pode-se reordenar os termos da expressão resultante conforme apresentado a seguir:

$$\tau = M(q)\ddot{q} + N(q, \dot{q}) + G(q), \tag{9}$$

em que $M(q)$ é a matriz $n \times n$ de massa do manipulador, $N(q, \dot{q})$ é um vetor de dimensão $n \times 1$ relacionado as forças de Coriolis e centrípeta, e $G(q)$ é um vetor $n \times 1$ com os termos que envolvem a gravidade.

Desse modo, o modelo não linear do manipulador pode ser escrito na forma em espaço de estado (KHAN *et al*., 2012):

$$\dot{x} = F(x) + G(x)u \tag{10}$$

em que:

$$F(x) = \begin{bmatrix} \dot{q} \\ -M^{-1}(N + G) \end{bmatrix},$$

$$G(x) = \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix},$$

$x = [q^T \quad \dot{q}^T]^T$ é o vetor de estado, e $u \equiv \tau$ é o vetor de entrada de controle.

3.3 MODELAGEM DE UM MANIPULADOR DE DOIS ELOS

Nesta seção serão derivadas as equações dinâmicas de um manipulador robótico com dois elos de juntas rotacionais. A Figura 3.5 apresenta o esquema do manipulador em questão, onde l_i é o comprimento do elo i, l_{g_i} é a distância entre o centro de gravidade do elo i a sua respectiva junta e θ_i é o ângulo da junta i, para $i = 1, 2$. Por simplicidade, será considerado que a massa de cada elo está concentrada no seu respectivo centro de gravidade.

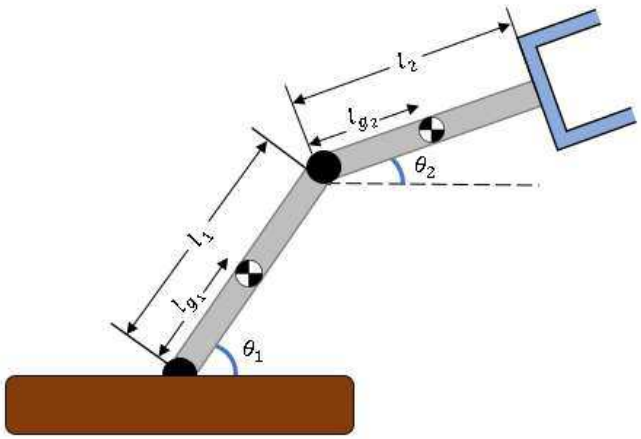
Para o elo 1, utilizando a Equação (5), tem-se:

$$k_1 = \frac{1}{2} m_1 v_{C_1}^T v_{C_1} + \frac{1}{2} \omega_1^T {}^{C_1}I_1 \, \omega_1 = \frac{1}{2} m_1 \dot{P}_{C_1}^T \dot{P}_{C_1} + \frac{1}{2} {}^{C_1}I_1 \dot{\theta}_1^{\,2},$$

em que $P_{C_1} = [l_{g_1} \cos\theta_1 \quad l_{g_1} \sin\theta_1]^T$. Resolvendo, tem-se:

$$k_1 = \frac{1}{2} m_1 l_{g_1}^2 \dot{\theta}_1^{\,2} + \frac{1}{2} {}^{C_1}I_1 \dot{\theta}_1^{\,2}. \tag{11}$$

Figura 3.5: Manipulador de dois elos com juntas rotativas



Fonte: Autor.

Já para o elo 2, escreve-se:

$$k_2 = \frac{1}{2} m_2 v_{C_2}^T v_{C_2} + \frac{1}{2} \omega_2^T {}^{C_2}I_2 \, \omega_2 = \frac{1}{2} m_2 \dot{P}_{C_1}^T \dot{P}_{C_1} + \frac{1}{2} {}^{C_2}I_2 \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2,$$

em que agora $P_{C_2} = [l_1 \cos\theta_1 + l_{g_2} \cos(\theta_1 + \theta_2) \quad l_1 \sin\theta_1 + l_{g_2} \sin(\theta_1 + \theta_2)]^T$. Logo:

$$\begin{aligned} k_2 = {} & \frac{1}{2} m_2 \left(l_1^2 \dot{\theta}_1^2 + l_{g_2} \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 + 2 l_1 l_{g_2} \dot{\theta}_1 \left(\dot{\theta}_1 + \dot{\theta}_2\right) \cos\theta_1 \right) \\ & + \frac{1}{2} {}^{C_2}I_2 \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2. \end{aligned} \tag{12}$$

Aplicando agora a Equação (7) para ambos os elos, tem-se:

$$u_1 = m_1 g^T P_{C_1} = m_1 g l_{g_1} \sin\theta_1 \tag{13}$$

$$u_2 = m_2 g^T P_{C_2} = m_2 g \left(l_1 \sin\theta_1 + l_{g_2} \sin(\theta_1 + \theta_2) \right). \tag{14}$$

Utilizando as Equações (4), (6), (8) e (3), obtêm-se as equações dinâmicas do manipulador em questão. Assim, tem-se:

$$\begin{aligned} \tau_1 = {} & \left[m_1 l_{g_1}^2 + {}^{C_1}I_1 + m_2 \left(l_1^2 + l_{g_2}^2 + 2 l_1 l_{g_2} \cos\theta_2 \right) + {}^{C_2}I_2 \right] \ddot{\theta}_1 \\ & + \left[m_2 \left(l_{g_2}^2 + l_1 l_{g_2} \cos\theta_2 \right) + {}^{C_2}I_2 \right] \ddot{\theta}_2 - m_2 l_1 l_{g_2} \sin\theta_2 \left(2\dot{\theta}_1 \dot{\theta}_2 + \dot{\theta}_2^{\,2} \right) \\ & + m_1 g l_{g_1} \cos\theta_1 + m_2 g \left(l_1 \cos\theta_1 + l_{g_2} \cos(\theta_1 + \theta_2) \right) \end{aligned}$$

$$\begin{aligned} \tau_2 = {} & \left[m_2 \left(l_{g_2}^2 + l_1 l_{g_2} \cos\theta_2 \right) + {}^{C_2}I_2 \right] \ddot{\theta}_1 + \left(m_2 l_{g_2}^2 + I_2 \right) \ddot{\theta}_2 + m_2 l_1 l_{g_2} \sin\theta_2 \, \dot{\theta}_1^{\,2} \\ & + m_2 g l_{g_2} \cos(\theta_1 + \theta_2), \end{aligned}$$

no qual τ_1 e τ_2 são os torques aplicados respectivamente nas juntas 1 e 2.

Organizando de maneira a tomar a forma da Equação (9), tem-se:

$$M = \begin{bmatrix} m_1 l_{g_1}^2 + {}^{C_1}I_1 + m_2\left(l_1^2 + l_{g_2}^2 + 2l_1 l_{g_2}\cos\theta_2\right) + {}^{C_2}I_2 & m_2\left(l_{g_2}^2 + l_1 l_{g_2}\cos\theta_2\right) + {}^{C_2}I_2 \\ m_2\left(l_{g_2}^2 + l_1 l_{g_2}\cos\theta_2\right) + {}^{C_2}I_2 & m_2 l_{g_2}^2 + {}^{C_2}I_2 \end{bmatrix}$$

$$N = \begin{bmatrix} -m_2 l_1 l_{g_2} \dot{\theta_2}^{\,2} \sin\theta_2 - 2m_2 l_1 l_{g_2}\dot{\theta_1}\dot{\theta_2}\sin\theta_2 \\ m_2 l_1 l_{g_2}\dot{\theta_1}^{\,2}\sin\theta_2 \end{bmatrix} \quad (15)$$

$$G = \begin{bmatrix} m_1 g l_{g_1}\cos\theta_1 + m_2 g\left(l_1\cos\theta_1 + l_{g_2}\cos(\theta_1+\theta_2)\right) \\ m_2 g l_{g_2}\cos(\theta_1+\theta_2) \end{bmatrix}.$$

3.3.1 Modelo Discreto no Tempo de um Manipulador

O controle de robôs, em geral, ocorre no espaço contínuo, mas com sua implementação dependente de um computador digital, como, por exemplo, microprocessadores. Portanto, de modo a verificar se o controlador proposto terá o comportamento desejado no tempo discreto, é importante simulá-lo com o modelo discretizado da planta. O manipulador robótico é um sistema contínuo com o sinal de controle $u(t)$ e estados $x(t)$, e, para prática de controle, uma referência de entrada $d(t)$. Para levar esse sistema para tempo discreto, deve-se amostrar $x(t)$ e $d(t)$ em períodos T_s de acordo com $x_k = x(kT_s)$ e $d_k = d(kT_s)$, respectivamente, onde $k = 0, 1, 2, \ldots$. Os sinais x_k e d_k são aplicados ao controlador digital, que, por sua vez, produz o sinal de controle u_k. Como o manipulador admite o sinal de entrada no espaço contínuo, tomando a forma de $u(t)$, u_k é mantido constante no intervalo $kT_s \leq t < (k+1)T_s$ para satisfazer esta condição.

Relacionando com o vetor de estado observado em (10) de um manipulador de 2 elos, define-se $x_k = [x_{k1} \quad x_{k2} \quad x_{k3} \quad x_{k4}]^T$ como o vetor de estado no instante de tempo k, onde $x_{k1} = q_1(kT_s)$ e $x_{k2} = q_2(kT_s)$ são, respectivamente, a posição angular da junta 1 e da junta 2, e $x_{k3} = \dot{q}_1(kT_s)$ e $x_{k4} = \dot{q}_2(kT_s)$ são, na ordem devida, a velocidade angular da junta 1 e da junta 2. O sinal de controle, naturalmente, é um vetor 2×1 em que $\tau(kT_s) = u_k$ é o torque aplicado nas juntas.

3.4 ATUADORES E SENSORES

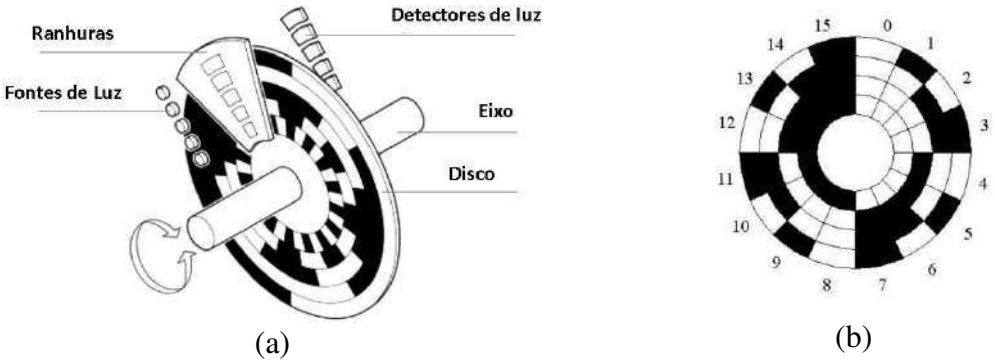
Dentre os elementos principais que constituem um manipulador robótico encontram-se os sensores e os atuadores. Os sensores são dispositivos que transformam o valor de uma grandeza física do ambiente em um sinal elétrico. Já os atuadores são unidades que convertem um tipo de energia em outro modificando uma determinada variável do ambiente (PAZOS, 2002).

3.4.1 Sensores em manipuladores robóticos

Um robô é equipado com diversos sensores a fim de monitorar as variáveis importantes para realização das tarefas. Estes dispositivos podem ser classificados entre externo e internos. Os externos são aqueles que observam as grandezas externas ao robô, como por exemplo, a distância do robô a um obstáculo. Por outro lado, os sensores internos coletam informações internas do sistema, como posição e velocidade das juntas do manipulador. Os principais sensores utilizados em robôs articulados são os *encoders*, potenciômetros, sensores inerciais, entre outros (RIBEIRO, 2004).

Os *encoders* são dispositivos eletromecânicos que traduzem as medidas de posição angular e linear em um sinal elétrico. Estes são constituídos por sensores óticos de barreira que detectam a passagem de luz de uma roda furada, conforme ilustrado na Figura 3.6 (a). Este disco é acoplado ao eixo do motor da junta. As informações obtidas pelos detectores fornecem uma codificação única para cada posição angular do eixo (Figura 3.6 (b)) (PAZOS, 2002).

Figura 3.6: Roda de um *encoder*



(a) (b)

Fonte: a) (AHMAD e AL-MAASHRI, 2007); b) (PAZOS, 2002)

Também existem sensores de posição baseado em potenciômetros. Estes sensores se baseiam na variação da resistência de um potenciômetro acoplado mecanicamente a um eixo.

3.4.2 Atuadores em manipuladores robóticos

No contexto dos manipuladores robóticos, os atuadores têm a função de transformar energia elétrica, hidráulica ou pneumática em movimento nas juntas.

Os atuadores hidráulicos utilizam um fluido para movimentar o braço e são normalmente aplicados quando se deseja operar cargas elevadas, onde é necessária grande potência e velocidade, mas com baixa precisão. Estes possuem a desvantagem de terem alto custo e a necessidade de pressurização do fluido hidráulico. Os pneumáticos, por sua vez, utilizam ar comprimido e são direcionados a manipuladores que realizam tarefas simples, repetitivas e que executam movimentos de alta velocidade entre duas posições específicas apresentando alta imprecisão em níveis intermediários (BOBROW e MCDONELL, 1998). Estes são mais baratos que os hidráulicos e são usados em robôs de pequeno porte. Por fim, os atuadores elétricos são atualmente os mais aplicados na robótica industrial, sendo direcionados a robôs de médio e pequeno porte (CARRARA, 2015). O principal atuador elétrico é o motor, tendo com função converter energia elétrica em movimento. Os principais tipos utilizados nos manipuladores são: motores de passo, servos motores, motores CC (corrente contínua), entre outros. Estes serão melhores apresentados a seguir.

Os motores de corrente contínua são dispositivos eletromecânicos alimentados por tensão e corrente contínua. Este se trata do tipo de atuador mais empregado na robótica, por sua simplicidade construtiva e facilidade de controle.

Já o motor de passo tem a característica de se movimentar de forma incremental, ou seja, em pequenos passos, de forma a rotacionar o eixo em um ângulo exato. Estes motores se caracterizam pelo baixo torque, repetição de movimentos bastante exatos, rotação em sentido horário e anti-horário, e possibilidade de controle digital. O acionamento de motores de passo se dá pela energização das bobinas, existindo atualmente circuitos especiais que produzem a sequência de sinais necessários para a realização de um movimento desejado. Devido a boa precisão dos motores de passo, os manipuladores que utilizam este tipo de atuador são, geralmente, controlados em malha aberta (ROMANO, 2002).

Os servos motores, por sua vez, são motores de corrente contínua composto por um sensor de posição interno, constituindo um sistema de realimentação. Desta forma, é possível controlar a posição angular do motor com precisão por meio de um sinal codificado na entrada. Na maioria dos casos, o tipo de sinal aplicado na entrada é o PWM (*Pulse Width Modulation*),

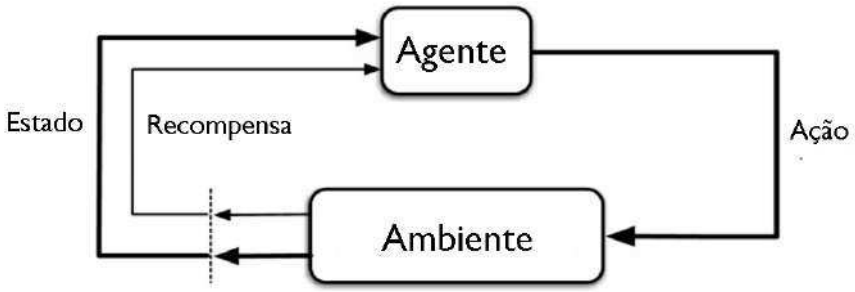
em que a posição do eixo do motor é proporcional à largura de pulso enviado. Os servos motores são pequenos, com ampla variação de torques e com posicionamento angular preciso.

4 FUNDAMENTAÇÃO TEÓRICA

O aprendizado por reforço é uma técnica de aprendizado de máquina que consiste em um agente interagir com o seu ambiente e modificar suas ações de acordo com a experiência adquirida. Esta técnica foi desenvolvida a partir de observações da natureza, onde constata-se a interação dos organismos vivos com o seu meio e uso dessas interações para melhorar suas próprias ações com o objetivo de sobreviverem e se desenvolverem. Os conceitos clássicos de RL aplicados a seres vivos foram inicialmente explorados por fisiologistas, como o russo Ivan Pavlov, que no início do século 20, realizou uma série de experimentos com cães, os quais eram estimulados com recompensas e punições simples, com a finalidade de modificar seus padrões de comportamento. Estas ideias também foram incorporadas nos estudos de Edward Thorndike, psicólogo norte-americano, que realizou experimentos com gatos dentro de "caixas de quebra-cabeças" (*puzzle box*). Nestes testes, o animal deve realizar uma série de ações para sair da gaiola e assim alcançar a comida, que a princípio estava fora do seu alcance. Após sucessivas experiências, o gato não só aprende o conjunto de ações que deve tomar para escapar, mas as executa de maneira mais rápida. Desta forma, o aprendizado por reforço é um conjunto de estratégias computacionais que permitem aos sistemas aprenderem a tomar ações direcionadas a um objetivo, observando o seu próprio comportamento e fazendo uso de mecanismos de reforço (recompensas e punições) (SUTTON e BARTO, 2015).

Essencialmente, os problemas de RL são em malha fechada, pois as ações do sistema influenciam nas entradas nos instantes de tempos seguintes (SUTTON e BARTO, 2015). A Figura 4.1 ilustra a interação entre o agente (sistema aprendiz e tomador de decisão) e o ambiente (o espaço real ou simulado onde o agente realiza suas ações).

Figura 4.1: Interação agente/ambiente no contexto de aprendizado por reforço



Fonte: (SUTTON e BARTO, 2015)

A formulação de problemas de aprendizado por reforço é geralmente fundamentada na estrutura de processo de decisão markoviano (PDM). Em um PDM determinístico, são

definidos o espaço de estados X, o espaço de ações U, a função de transição de estados f e a função de recompensa r. Desta forma, em cada instante tempo discreto k, o agente observa os estados x_k e aplica uma ação u_k, que levará ao processo de transição para um novo estado x_{k+1}, obedecendo $f(x_k, u_k)$. Nesta mudança de estados, é retornado o sinal de recompensa r_k (BUŞONIU *et al.*, 2011).

A recompensa é definida como um valor numérico disponibilizado ao agente após a execução de uma ação (GONÇALVES, 2016). A função de recompensa é construída baseando-se no objetivo a ser alcançado e nas demandas exigidas pelo projetista, como custo mínimo de energia, mínimo risco etc. O agente, deste modo, escolhe as ações de forma a maximizar a recompensa recebida (KHAN, 2012) dada por:

$$r_k = r(x_k, u_k). \tag{16}$$

Relacionado com a ideia de recompensa, encontra-se o conceito da função valor $V(x_k)$. Esta é definida como a recompensa que se espera acumular a partir do estado atual x_k com o agente seguindo determinada estratégia de controle. Esta estratégia se refere ao comportamento que o agente vai ter dado o estado x_k. Em RL, o procedimento de escolha da ação no passo de tempo k é conhecido como política. Mais precisamente, uma política de controle é definida como uma função do espaço de estados para o espaço de ações $h(\cdot): X \rightarrow U$, ou seja, para cada estado x_k, a política define uma ação $u_k = h(x_k)$ (LEWIS e VRABIE, 2009).

Desta forma, r_k é o valor numérico da recompensa imediata, enquanto a função valor é a predição da soma das recompensas futuras. A ideia da função valor é importante no aprendizado por reforço, pois ela avalia o desempenho de uma determinada política para assim aprimorá-la (KHAN *et al*, 2012). A predição das recompensas futuras tem um papel essencial nesse tipo de aprendizado, pois em muitas situações, a lei de controle pode inicialmente levar o agente a receber recompensas ruins, mas a longo prazo, maximizar $V(x_k)$.

A noção da função valor é capturada pela expressão:

$$V^h(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i), \tag{17}$$

em que o parâmetro γ é conhecido como fator de desconto, limitado ao intervalo $0 < \gamma \leq 1$ e $u_i = h_i(x_i)$ é uma política de controle prescrita. O parâmetro γ determina a importância que será dada às recompensas futuras. Não há método genérico para determinar o seu valor. O

projetista deve determinar este parâmetro fazendo um balanço entre a qualidade da solução e a taxa de convergência (BUŞONIU *et al.*, 2011).

Normalmente, quando o modelo do ambiente é desconhecido, é utilizada a função valor de estado-ação, representada por $Q(x_k, u_k)$, ao invés de $V(x_k)$. Esta função retorna o valor do agente estar em determinado estado x_k e executar determinada ação u_k. De maneira mais formal, define-se a função Q, ou função valor de estado-ação, associada à política $u_k = h(x_k)$ como:

$$Q^h(x_k, u_k) = r(x_k, u_k) + \gamma V^h(x_{k+1}). \tag{18}$$

Uma vez que a função Q depende também da ação, ela já inclui informação sobre a qualidade das transições. Em contraste, a função valor de estado V somente descreve a qualidade dos estados. Neste caso, para inferir a qualidade das transições, estas devem ser explicitamente levadas em consideração. Consequentemente, um modelo do MDP é necessário na forma da dinâmica f e da função de utilidade r, enquanto na formulação usando função Q o problema de decisão markoviano é tratado sem referência à tais modelos.

Sutton e Barto (2015) listam três métodos principais para resolver problemas de decisão sequencial: Programação Dinâmica, Métodos de Monte Carlo e Métodos de Diferença Temporal.

4.1 PROGRAMAÇÃO DINÂMICA

O termo “programação dinâmica” refere-se a um conjunto de algoritmos que podem ser utilizados para calcular leis de controle ótimas dado o modelo do sistema (SUTTON e BARTO, 2015). Inicialmente, estes conceitos foram desenvolvidos no contexto de controle ótimo, mas se relacionam intimamente com as abordagens modernas de RL (RÊGO, 2014).

Algoritmos de programação dinâmica exigem o conhecimento do modelo do Processo de Decisão de Markov, incluindo a transição dinâmica e a função de recompensa (BERTSEKAS, 2007). Estes algoritmos são aplicados *offline* produzindo uma política que é utilizada para controlar o processo.

Reescrevendo a Equação (17), tem-se:

$$V^h(x_k) = r(x_k, u_k) + \gamma \sum_{i=k+1}^{\infty} \gamma^{i-(k+1)} r(x_i, u_i) \tag{19}$$

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

A Equação (19) é conhecida como equação de Bellman. Ela define o valor do agente estar no estado x_k como a soma da recompensa imediata, resultante da transição para o estado x_{k+1} pela aplicação de u_k, com o valor no estado seguinte (x_{k+1}) (RÊGO, 2014).

A programação dinâmica está baseada na ideia do princípio da Otimalidade de Bellman. Este princípio estabelece que em uma trajetória ótima, quaisquer que tenham sido as ações anteriores ao estado atual, as ações seguintes deverão ser ótimas, ou seja, uma política ótima é formada por subconjuntos de ações ótimas.

Determinando para a Equação (19) que a política avaliada pela função valor seja uma política ótima (h^*), tem-se:

$$V^{h^*}(x_k) = r\big(x_k, h^*(x_k)\big) + \gamma V^{h^*}(x_{k+1}). \tag{20}$$

Todas as políticas ótimas acumularão o máximo de recompensas a longo prazo, ou seja, retornarão ações que maximizarão a função valor. O valor ótimo pode ser expresso por:

$$V^*(x_k) = \max_{h(\cdot)} \Big(r\big(x_k, h(x_k)\big) + \gamma V^*(x_{k+1}) \Big). \tag{21}$$

Esta expressão é conhecida como a equação da Otimalidade de Bellman. Posto isto, a política ótima pode ser expressa por:

$$h^*(x_k) = \arg\max_{h(\cdot)} \Big(r\big(x_k, h(x_k)\big) + \gamma V^*(x_{k+1}) \Big). \tag{22}$$

Pode-se observar que para a Equação (21) é necessário o conhecimento da função valor no instante de tempo $(k+1)$ para determinar a função valor ótima no instante de tempo k. Desta forma para resolver este problema é realizado um procedimento “para trás no tempo” (*backward-in-time*).

Os algoritmos utilizados em programação dinâmica podem ser divididos em três subclasses baseado na estratégia tomada para determinar a política ótima: Iteração de Política, Iteração de Valor e Busca de Política (BUŞONIU *et al*., 2011).

4.1.1 Iteração de Política

Algoritmos de iteração de política aplicam duas operações iterativamente: a avaliação de política e a melhoria de política. A avaliação de política é feita através da solução da equação de Bellman (19) que avalia a qualidade da política de controle atual. Na etapa de melhoria de

política, uma nova política melhorada é determinada utilizando $V(x_k)$ obtida na operação anterior.

Desta forma, dada uma política inicial admissível, é realizado um processo iterativo para a determinação da função valor que melhor avalia a política atual h_j. De acordo com Lewis e Vrabie (2009), a equação de Bellman (19) é uma equação do ponto fixo. Portanto, iniciando com qualquer função valor V_0 é possível encontrar V^{h_j} através do seguinte mapeamento de contração

$$V_{i+1}(x_k) = r\left(x_k,\ h_j(x_k)\right) + \gamma V_i(x_{k+1}), \tag{23}$$

para $i = 0,\ 1,\ 2,\ \ldots$. A iteração ocorre até a convergência, ou seja, $V_i \rightarrow\ V^{h_j}$ quando $i \rightarrow \infty$.

Em seguida, é realizada a melhoria da política, na qual se estabelece uma nova política de controle h_{j+1} que é dada por:

$$h_{j+1}(x_k) = \arg\max_{h(\cdot)} \left(r\big(x_k, h(x_k)\big) + \gamma V_i(x_{k+1})\right). \tag{24}$$

Esse procedimento é descrito por Howard (1960) que prova que a nova política h_{j+1} é melhor, ou pelo menos igual à política h_j, de modo que $V^{h_{j+1}}(x_k) \geq V^{h_j}(x_k)$.

A ideia do algoritmo de iteração de política é apresentada na sequência:

$$h_0 \overset{E}{\rightarrow} V^{h_0} \overset{I}{\rightarrow} h_1 \overset{E}{\rightarrow} V^{h_1} \overset{I}{\rightarrow} h_2 \overset{E}{\rightarrow} \ldots \overset{I}{\rightarrow} h^* \overset{E}{\rightarrow} V^*\ ,$$

em que $\overset{E}{\rightarrow}$ denota a etapa de avaliação da política e $\overset{I}{\rightarrow}$ a melhoria da política (SUTTON e BARTO, 2015).

O algoritmo de iteração de política determinístico é apresentado no Algoritmo 1 (BUŞONIU *et al.*, 2011; SUTTON e BARTO, 2015).

4.1.2 Iteração de Valor

Algoritmos de iteração de valor direcionam sua busca, inicialmente, apenas pela função valor ótima $V^*(x_k)$, a qual é calculada iterativamente usando a equação da Otimalidade de Bellman (21). Somente após a convergência do processo é que a política ótima é computada (BUŞONIU *et al*., 2011).

A Iteração de Valor se baseia no fato de que a Equação da Otimalidade de Bellman (21) também é uma equação do ponto fixo. Portanto, o algoritmo é iniciado com uma função valor

arbitrária V_0 e a cada iteração i atualiza a função valor usando o seguinte mapeamento de contração:

$$V_{i+1}(x_k) = \max_{h(\cdot)} \Big(r\big(x_k, h(x_k)\big) + \gamma V_i(x_{k+1}) \Big), \tag{25}$$

para $i = 0,\ 1,\ 2$ ... Tal mapeamento é iterado até a função valor ótima, ou seja, $V_i \to V^*$.

A política ótima é, por fim, calculada de acordo com a seguinte equação:

$$h^*(x_k) = \arg\max_{h(\cdot)} \Big(r\big(x_k, h(x_k)\big) + \gamma V_{i+1}(x_{k+1}) \Big). \tag{26}$$

A sequência de passos do processo de Iteração de Valor é mostrada no Algoritmo 2 (BUŞONIU *et al.*, 2011; SUTTON e BARTO, 2015).

Algoritmo 1: Iteração de Política

Entrada: função de transição f, função de recompensa r, fator de desconto γ e as tolerâncias ϵ_v e ϵ_h.

Inicie com uma política $h_0(x)$ admissível, para todo $x_k \in X$, $j = 0$ e $i = 0$.

Repetir

// Avaliação de Política

Repetir:

$\forall\, x_k \in X: V_{i+1}(x_k) = r\left(x_k, h_j(x_k)\right) + \gamma V_i(x_{k+1})$

$i \leftarrow i + 1$

até $\|V_{i+1}(x_k) - V_i(x_k)\|_\infty < \epsilon_v$

// Melhoria de Política

$\forall\, x_k \in X: h_{j+1}(x_k) = \operatorname{argmax}_{h(\cdot)} \big(r(x_k, h(x_k)) + \gamma V_{i+1}(x_{k+1}) \big)$

$j \leftarrow j + 1$

até $\left\| h_{j+1}(x_k) - h_j(x_k) \right\|_\infty < \epsilon_h$

Saída: $V_i(x_k)$ e $h_j(x_k)$

4.1.3 Busca de Política

A busca de política constitui uma classe maior de métodos de programação dinâmica/aprendizado por reforço, que utilizam algoritmos de otimização para buscar diretamente uma política ótima. A princípio qualquer técnica de otimização pode ser utilizada, entretanto para problemas mais genéricos, deve-se considerar técnicas de otimização global aos

métodos de otimização local. Por exemplo, algoritmos genéticos, busca tabu, busca direta, entropia cruzada etc. (BUŞONIU *et al*., 2011).

Algoritmo 2: Iteração de Valor

Entrada: função de transição de estados f, função de recompensa r, fator de desconto γ e a tolerância ϵ_v.

Inicialize $V(x_k)$ arbitrariamente (por exemplo: $V_0(x_k) = 0$ para todo $x_k \in X$) e $i = 0$.

Repetir

$$\forall\ x_k\ \in\ X\colon V_{i+1}(x_k) = \max_{h(\cdot)} \left(r\left(x_k, h(x_k)\right) + \gamma V_i(x_{k+1})\right)$$

$$i \leftarrow i + 1$$

até $\|V_{i+1}\,(x_k) - V_i(x_k)\|_\infty < \epsilon_v$

Retorna uma política, h^*, de modo que:

$$h^*(x_k) = \arg\max_{h(\cdot)} \left(r\left(x_k,\ h(x_k)\right) + \gamma V_{i+1}\,(x_{k+1})\right)\ \forall\ x_k \in X$$

Saída: $V_i(x_k)$ e $h^*(x_k)$

4.2 MÉTODOS DE MONTE CARLO

Diferente dos métodos de programação dinâmica, os métodos de Monte Carlo não exigem o conhecimento completo do ambiente, mas apenas a experiência adquirida, ou seja, uma amostra de uma sequência composta por: estados medidos, ação aplicada e recompensa recebida (SUTTON e BARTO, 2015).

O termo "Monte Carlo" é usado geralmente de forma mais ampla, incluindo diversos métodos de estimação que envolvem componentes randômicos. No contexto de aprendizado por reforço, os métodos de Monte Carlo fundamentam-se na média dos retornos obtidos ao longo da exploração do agente no ambiente.

Quando o modelo do ambiente não é conhecido é apropriado utilizar a função valor de estado-ação Q ao invés da função valor estado V. Desta forma, inicialmente é realizada a etapa de predição, onde a função Q é estimada para a política atual. Uma maneira intuitiva de estimar essa função pela experiência adquirida é simplesmente calculando a média das recompensas recebidas para cada par estado-ação visitado. Portanto, quanto mais determinado estado é visitado, mais retornos são adquiridos e consequentemente a média converge para o valor esperado (SUTTON e BARTO, 2015). Então, quando $N(x_k, u_k) \to \infty$ a função $Q(x_k, u_k)$ se

aproxima do valor verdadeiro, onde $N(x_k, u_k)$ é o incremento que registra o número de vezes que determinado par estado-ação (x_k, u_k) foi visitado (MORALES, 2020).

Entretanto, para assegurar que os retornos estejam devidamente disponíveis, o cálculo da média é realizado ao fim de cada ciclo de ação aplicadas, ou seja, a experiência adquirida é dividida entre episódios. Portanto, este método é geralmente aplicado em tarefas com característica episódica, pois só após a conclusão de um ciclo de ações é que a função valor é estimada e a política é aprimorada (SUTTON e BARTON, 2015).

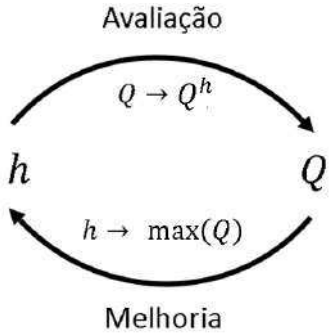
A função Q pode também ser atualizada de forma incremental através de:

$$Q(x,\, u) \leftarrow Q(x,u) + \frac{1}{N(x,u)}\left[G_j - Q(x,u)\right], \tag{27}$$

para cada $x \in X$ e $u \in U$ e onde G_j é o retorno recebido em cada par (x, u) no j-ésimo episódio.

Com a função Q devidamente estimada, é possível realizar a melhoria de política, para isso utiliza-se a mesma ideia do algoritmo de Iteração de Política da Programação Dinâmica. Entretanto, na estratégia padrão, a política a ser avaliada é fixada e um processo de atualizações da função valor ocorre até se obter a convergência, ou seja, até encontrar a função valor que melhor avalia a política atual. Nesse caso, será utilizada uma variação desse método onde há uma interação entre a etapa de avaliação e de melhoria da política. Desse modo, a função Q é repetidamente alterada para se aproximar mais da função valor de estado-ação da política atual (Q^h), e a política é repetidamente melhorada em relação a função Q atual (SUTTON e BARTON, 2015). A Figura 4.2 ilustra este processo iterativo.

Figura 4.2: Avaliação e melhoria da política



Fonte: (SUTTON e BARTO, 2015)

Como a função Q é aplicada, não há necessidade do modelo para construir a política. Portanto, para cada $x \in X$, determina-se a ação que maximiza o valor ação:

$$h(x) = \arg\max_{h(\cdot)} Q(x_k, u_k). \quad (28)$$

4.3 MÉTODOS DE DIFERENÇAS TEMPORAIS

Os métodos de Monte Carlo possuem uma desvantagem relacionada ao fato de que são muito difíceis de serem implementados *online*, desde que a política só é atualizada de forma episódica. Neste caso, os métodos de Diferenças Temporais (do inglês *Temporal Diferences - TD*) possuem a característica de dispensarem a necessidade do modelo do ambiente e, em contraste aos métodos de Monte Carlo, podem ser implementados *online* passo-a-passo, ou seja, a função valor é estimada e a política é melhorada a cada transição de estado do ambiente (SUTTON e BARTO, 2015). Atualmente, os métodos TD também se tornaram esquemas bastante populares no quadro de aprendizado por reforço (KHAN *et al.*, 2012).

4.3.1 SARSA

SARSA é um algoritmo de diferença temporal da classe dos métodos de política-*on* (*on-policy*), ou seja, a estimação da função valor de estado-ação é realizada a partir das informações obtidas seguindo a política atual $h(x_k)$. Neste procedimento, inicialmente é tomada uma ação $u_k = h(x_k)$ que leva o ambiente para um novo estado x_{k+1} em que se observa a recompensa imediata r_k. Em seguida, uma nova ação é determinada utilizando ainda a política atual, ou seja, $u_{k+1} = h(x_{k+1})$. Estas informações são então utilizadas para melhor estimar a função Q de modo a satisfazer a condição de consistência que é dada por:

$$Q^h(x_k, u_k) = r(x_k, u_k) + \gamma Q^h(x_{k+1}, h(x_{k+1})). \quad (29)$$

A esta equação dá-se o nome de equação de Bellman para a função Q. É possível obter a Equação (29) a partir das Equações (18) e (19), observando-se a relação $Q^h(x_k, h(x_k)) = V^h(x_k)$. A sequência de dados $(x_k, u_k, r_k, x_{k+1}, u_{k+1})$ exigida para o algoritmo inspirou o nome dado a esse método (Estado, Ação, Recompensa, Estado Seguinte, Ação Seguinte, do inglês - SARSA).

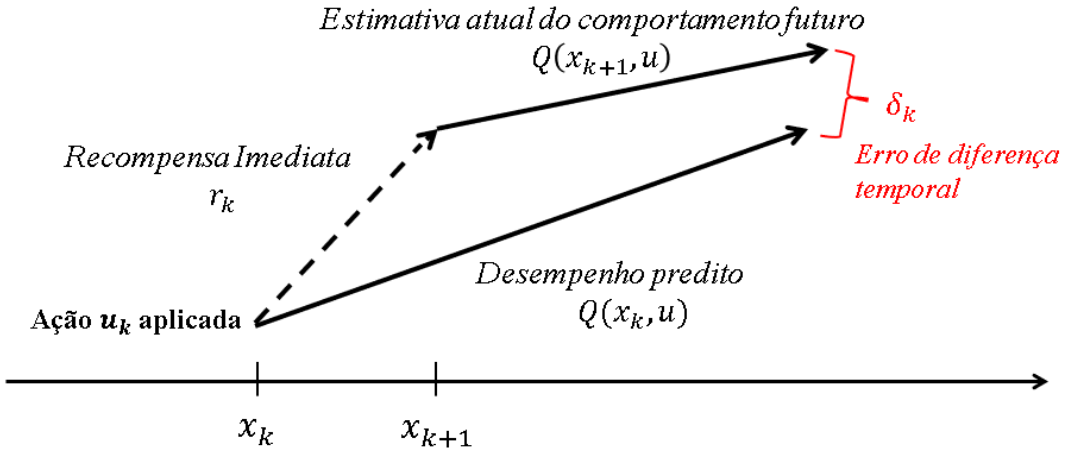
A etapa de estimação da função valor de estado-ação apoia-se na minimização do erro de diferença temporal δ_k, que pode ser entendido como a diferença entre o valor observado e o

valor predito. A função δ_k é expressa na Equação (30), podendo ser calculado a cada passo de tempo k:

$$\delta_k = r_k + \gamma Q^h(x_{k+1}, u_{k+1}) - Q^h(x_k, u_k). \tag{30}$$

A Figura 4.3 ilustra a ideia do erro de diferença temporal. Essa diferença é calculada entre a estimativa atualizada $r_k + \gamma Q(x_{k+1}, u_{k+1})$, obtida após a aplicação da ação u_k, e a estimativa atual $Q(x_k, u_k)$ (RÊGO, 2014). É importante notar que a estimativa atualizada envolve o sinal de recompensa, desta forma, r_k progressivamente "injeta realidade" a estimação da função Q (MORALES, 2020).

Figura 4.3: Erro de Diferença Temporal



Fonte: Autor

No SARSA, a cada passo de tempo, a função Q é atualizada do seguinte modo:

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha[r_k + \gamma Q_k(x_{k+1}, u_{k+1}) - Q_k(x_k, u_k)]. \tag{31}$$

SARSA combina uma política gulosa[1] na função Q atual com exploração usando, por exemplo, a exploração ε-gulosa (ε-*greedy*). Este método permite lidar com o dilema típico do aprendizado por reforço: o balanço entre a exploração e explotação *(exploration/exploitation*). Com a aplicação de ε-*greedy*, a seleção da ação é feita de acordo com:

$$u_k \leftarrow \begin{cases} arg\,\max_u Q_k(x_k, u) & \text{com probabilidade } 1 - \varepsilon_k \text{ (explotação)} \\ ação\ aleatória\ em\ U & \text{com probabilidade } \varepsilon_k \text{ (exploração)} \end{cases} \tag{32}$$

em que $\varepsilon_k \in (0,1)$ é a probabilidade de exploração no passo de tempo k. Normalmente, deseja-se que a exploração diminua com o tempo, que modo que a política se torne ótima. Isto pode

[1] Política que escolhe ações que maximizam Q

ser realizado fazendo com que ε_k se aproxime de 0 conforme k aumenta, como por exemplo $\varepsilon_k = 1/k$ (BUŞONIU *et al.*, 2011).

A atualização da política pode ser realizada pela Equação (28). O algoritmo SARSA com exploração ε-gulosa é mostrado na Algoritmo 3.

Algoritmo 3: SARSA com exploração ε-gulosa

Entrada: fator de desconto γ, o fator de exploração ε_k e a taxa de aprendizado α.

Inicie arbitrariamente $Q(x_k, u_k)$ para todo $x_k \in X$ e $u_k \in U$

Meça o estado x_0 inicial

$$u_0 \leftarrow \begin{cases} \arg\max_u Q_0(x_0, u) & \text{com probabilidade } 1 - \varepsilon_0 \\ ação\ aleatória\ em\ U & \text{com probabilidade } \varepsilon_0 \end{cases}$$

Repetir:

Aplique u_k, meça o próximo estado x_{k+1} e receba a recompensa r_k

$$u_{k+1} \leftarrow \begin{cases} \arg\max_u Q_k(x_{k+1}, u) & \text{com probabilidade } 1 - \varepsilon_k \\ ação\ aleatória\ em\ U & \text{com probabilidade } \varepsilon_k \end{cases}$$

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha[r_k + \gamma Q_k(x_{k+1}, u_{k+1}) - Q_k(x_k, u_k)]$$

$$k \leftarrow k + 1$$

até satisfazer o critério de parada

4.3.2 Aprendizagem Q

Aprendizagem Q (Q-*Learning*) foi inicialmente proposta em Watkins (1989) tratando-se de um dos métodos mais amplamente utilizados Buşoniu *et al*. (2011). Neste caso, refere-se a um método de política-*off* (*off-policy*), ou seja, a estimação da função valor de estado-ação independe da política atual. A equação de atualização é expressa na Equação (33) que utiliza a tupla de dados (x_k, u_k, r_k, x_{k+1}):

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha \left[r_k + \gamma \max_{u \in U} Q_k(x_{k+1}, u) - Q_k(x_k, u_k) \right]. \quad (33)$$

Nota-se que esta expressão se assemelha a equação de atualização do SARSA, com a diferença que a parcela entre parênteses inclui um termo de otimalidade ($\max_{u \in U} Q_k$). Os passos do método de aprendizagem Q são apresentados no Algoritmo 4.

Algoritmo 4: *Q-Learning* com exploração ε-gulosa

Entrada: fator de desconto γ, o fator de exploração ε_k e a taxa de aprendizado α

Inicialize arbitrariamente $Q(x_k, u_k)$ para todo $x_k \in X$ e $u_k \in U$

Meça o estado x_0 inicial

Repetir:

$$u_k \leftarrow \begin{cases} \arg\max_u Q_k(x_k, u) & \text{com probabilidade } 1 - \varepsilon_k \\ ação\ aleatória\ em\ U & \text{com probabilidade } \varepsilon_k \end{cases}$$

Aplique u_k, meça o próximo estado x_{k+1} e receba a recompensa r_k

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha \left[r_k + \gamma \max_u Q(x_{k+1}, u) - Q(x_k, u_k) \right]$$

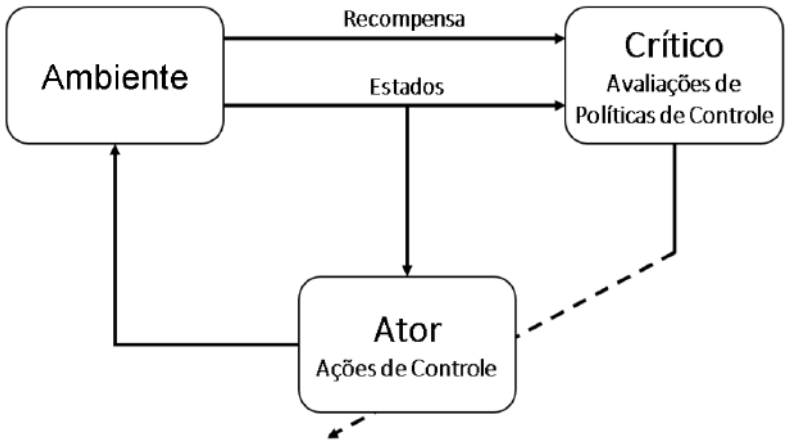
$k \leftarrow k + 1$

até satisfazer o critério de parada

4.3.3 Métodos Ator-Crítico

Os métodos Ator-Crítico foram inicialmente introduzidos em trabalhos como Witten (1977) e Barto *et al.* (1983). Para tanto, são utilizadas duas estruturas paramétricas distintas para realizar o aprendizado. Uma delas é direcionada a avaliação da política, recebendo o nome de Crítico, enquanto a segunda realiza a melhoria da política, chamada de Ator (KHAN *et al.*, 2012). A Figura 4.4 ilustra a estrutura Ator-Crítico.

Figura 4.4: Estrutura do método de aprendizagem Ator-Crítico



Fonte: Autor

No contexto de controle, o Ator representa o agente interagindo com o ambiente, calculando as ações a serem seguidas, ou seja, o Ator corresponde a um controlador

determinando as ações de controle. O Crítico, por sua vez, avalia as ações tomadas pelo Ator (controlador) através do método de diferença temporal (30) para assim aprimorar sua estimação da função Q e a política (KHAN *et al.*, 2012; REGÔ, 2014).

Atualmente, os métodos Ator-Crítico despertam um interesse particular nos projetistas por duas vantagens significativas: exigem cálculos computacionais mínimos para selecionar as ações e a possibilidade de aprenderem políticas explicitamente estocásticas (SUTTON e BARTON, 2015).

4.4 APROXIMAÇÃO DA FUNÇÃO VALOR

A principal desvantagem dos métodos convencionais de aprendizado por reforço é a necessidade de estimarem o valor estado ação para todo o espaço dos estados e das ações do problema. A representação tabular da função Q mostra-se inadequada quando a dimensão desses espaços é muito grande ou trata-se de espaços contínuos. Este problema limita o uso dos métodos de programação dinâmica em sistemas mais complexos, o que é conhecido como "maldição da dimensionalidade" (LEWIS e VRABIE, 2009). Por exemplo, o número de estados do jogo de tabuleiro gamão chega a 10^{20}, já no jogo de tabuleiro GO existem 10^{127} estados. Em jogos eletrônicos, como os de Atari, cada *frame* tem 210-por-160 *pixels* e três canais de cores o que resulta em um número enorme de estados (MORALES, 2020). Na robótica, tanto o espaço das ações quanto o espaço dos estados são contínuos. Desta forma, para esses problemas existe um gargalo relacionado a sua aplicação prática devido a grande quantidade de memória computacional exigida.

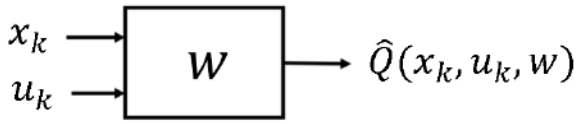
Para tornar prática as implementações de técnicas de aprendizado por reforço, aproximadores de funções são utilizados para substituir a representação tabular da função Q e assim tornando os algoritmos mais eficientes (MORALES, 2020). Esta abordagem tem sido conhecida como Programação Dinâmica Aproximada ou Programação Neurodinâmica. A Figura 4.5 ilustra a nova representação da função valor de estado-ação, que é obtida através de uma função parametrizada e expressa por:

$$Q(x_k, u_k) \approx \hat{Q}(x_k, u_k, w). \tag{34}$$

Os problemas de RL podem ser combinados com diversas técnicas de aproximação de função parametrizadas como: regras *fuzzy*, redes neurais, máquinas de vetores de suporte, combinação linear de características, entre outros (SHAH e GOPAL, 2009). Atualmente, redes

neurais são amplamente utilizadas para este propósito. A vantagem encontrada no uso de redes neurais é a sua capacidade de lidar com representações mais complexas da função valor com o número menor de parâmetros comparado aos outros métodos (RASTOGI, 2017).

Figura 4.5: Aproximação da função valor de estado-ação

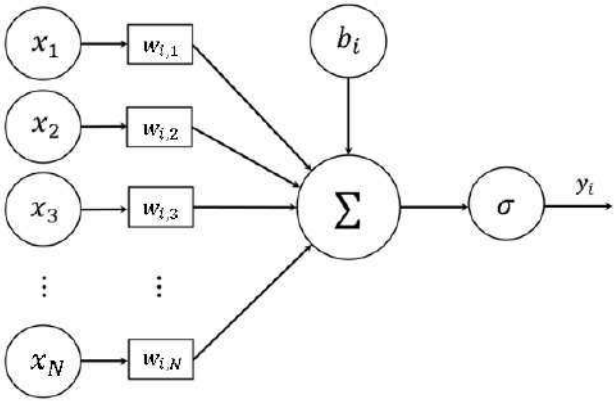


Fonte: Autor

4.5 REDES NEURAIS

Redes Neurais Artificiais (RNA) são estruturas computacionais inspiradas na forma como o cérebro humano processa informações. Similarmente ao cérebro humano, as RNA's têm como a unidade básica de processamento o neurônio. A Figura 4.6 ilustra um neurônio artificial.

Figura 4.6: Neurônio Artificial



Fonte: Autor

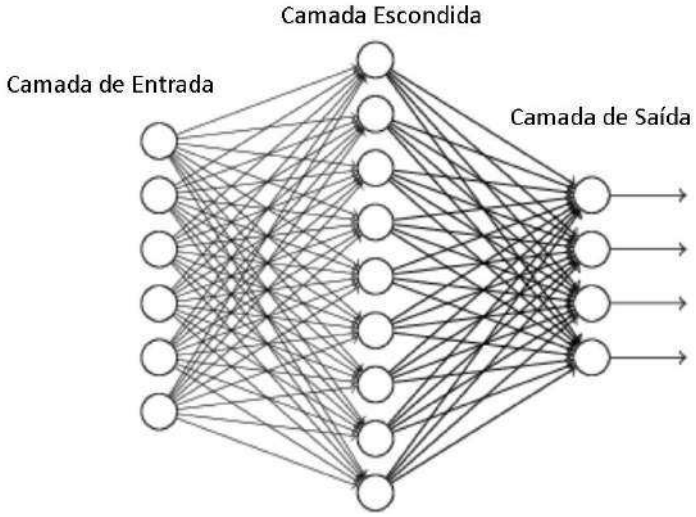
Matematicamente, um neurônio completo pode ser escrito como:

$$y_i = \sigma\left(\sum_{j=1}^{N} w_{i,j}x_j + b_i\right), \tag{35}$$

no qual x_1, x_2, ...,x_N são os sinais de entrada; $\sigma(\cdot)$ é a função de ativação; $w_{i,1}$, $w_{i,2}$, ...,$w_{i,N}$ são os pesos sinápticos do neurônio; b_i é o *bias*; e y_i é o sinal de saída (HAYKIN, 1999).

Na imensa maioria das aplicações práticas, uma RNA é constituída por camadas de neurônios, possuindo necessariamente uma camada de neurônios de entrada e uma camada de saída e um número arbitrário de camadas escondidas (HAYKIN, 1999). A informação passa da camada de entrada para as camadas escondidas e, na sequência, para a camada de saída, onde a resposta do processamento é adquirido (Figura 4.7). Cada neurônio de cada uma das camadas realiza o mesmo cálculo apresentado na Equação (35).

Figura 4.7: Rede neural multicamadas



Fonte: (NIELSEN, 2015)

No contexto de aprendizado supervisionado, as redes neurais são treinadas utilizando conjunto de dados de entrada e de saída coletados de um determinado problema. Entretanto, no contexto de aprendizado por reforço, as RNA's são utilizadas para aproximar tanto a função Q quanto a política $h(x_k)$. Este processo deve ser realizado usando apenas as informações disponíveis (estados, sinais de controle e sinal de recompensa). Desta forma, ao invés de treinar a rede com um conjunto de dados de uma única vez, a rede é treinada conforme o agente interage com o ambiente (MOTA, 2018).

RNA's com múltiplas camadas possuem a propriedade de serem aproximadores universais de função. O teorema de aproximação universal, mostrado pela primeira vez por Cybenko (1989), afirma que uma rede neural com pelo menos uma camada escondida e com a camada de saída linear é capaz de aproximar qualquer função suave $f(x)$: $\mathbb{R}^N \rightarrow \mathbb{R}^m$ dado um conjunto de pesos ideais:

$$f(x) = W^o \sigma(W^H x) + \epsilon_L, \tag{36}$$

no qual $W^H \in \mathbb{R}^{L\times N}$ é a matriz de pesos entre a camada de entrada e a escondida; $W^o \in \mathbb{R}^{m\times L}$ é a matriz de pesos entre a camada escondida e a de saída. O erro de aproximação, denotado por ϵ_L, geralmente diminui uniformemente conforme o número de neurônios L da camada escondida cresce. Este teorema tem sido provado com o uso de diversas funções de ativação nos neurônios da camada escondida, como funções sigmoides, RBF (funções de base radial) (CHEN *et al*., 2014), entre outras (LEWIS *et al.,* 1998; LEWIS *et al.*, 2004; LEWIS e VRABIE, 2009).

Os pesos ideais da RNA, definida em (36), que melhor promovem a aproximação de uma dada função não linear são difíceis de determinar. Entretanto, no contexto de controle, para um dado erro de aproximação ϵ_{max}, tal que $\epsilon_{max} > 0$ e $\|\epsilon_L\| < \epsilon_{max}$, existe um conjunto de pesos da RNA próximos o suficiente dos valores ideais. Desta forma, uma estimativa de $f(x)$ pode ser dada por:

$$\hat{f}(x) = \widehat{W}^O \sigma(\widehat{W}^H x), \tag{37}$$

em que $\widehat{W}^H$ e $\widehat{W}^o$ são estimados através de algum algoritmo de sintonia (LEWIS *et al.,* 1998).

Um caso particular da RNA definida em (37), ocorre quando apenas os pesos da camada de saída são estabelecidos como os parâmetros a serem estimados. Desse modo, pode-se simplificar a representação fazendo $\phi(x) = \sigma(\widehat{W}^H x)$, visto que $\widehat{W}^H$ é constante. Logo:

$$\hat{f}(x) = \widehat{W}^O \phi(x), \tag{38}$$

em que $\phi(x) = [\phi_1(x) \quad \phi_2(x) \quad \dots \quad \phi_L(x)]: \mathbb{R}^N \to \mathbb{R}^L$. A expressão (38) descreve o esquema de uma rede neural linear nos parâmetros da camada de saída. Esta RNA tem como grande vantagem ser mais fácil de treinar que (36) e garante uma boa capacidade de aproximar funções suaves desde que $\phi(\cdot)$ seja escolhido com vetor de funções de base (LEWIS *et al.,* 1998; LEWIS *et al.*, 2004).

4.6 CONTROLE ÓTIMO

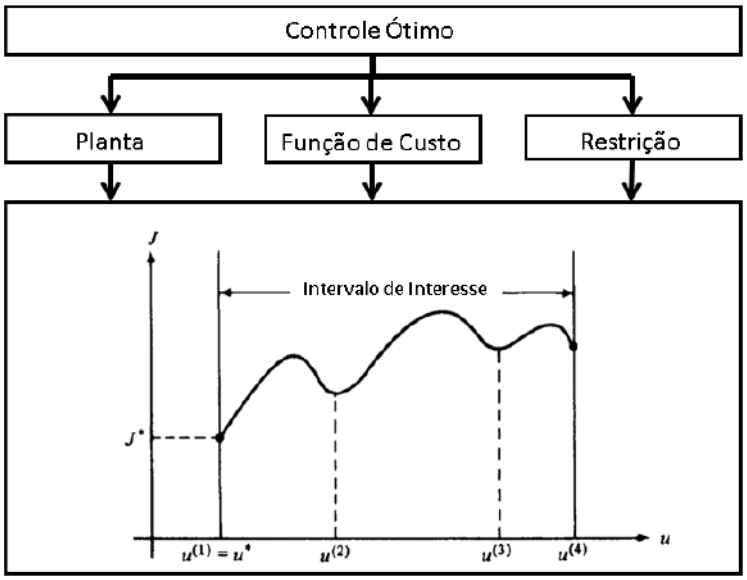
O controle ótimo é um ramo da teoria de controle que lida com o problema de determinar leis de controle para um sistema dinâmico de modo a obter o melhor desempenho. Esta abordagem computa a estratégia de ações otimizada minimizando uma função de custo, ou função de utilidade, que inclui as exigências do projeto, os objetivos do problema e as restrições do sistema. A escolha da função de custo possui um papel crucial no projeto de controle ótimo,

por este quantificar o índice de desempenho do controlador possibilitando sua avaliação, desta forma, esta seleção está intimamente relacionada com o contexto do problema de controle em análise.

Resumidamente, o problema de controle ótimo para sistemas em tempo discreto é direcionado a determinar uma lei de controle u_k^* no intervalo de interesse $[i, N]$ que leva o sistema a seguir uma trajetória admissível x_k^* de modo a minimizar a função de custo $J(\cdot)$ (LEWIS *et al*., 2012b).

A formulação matemática do problema de controle ótimo exige a definição de alguns itens: a modelagem do processo a ser controlado, o estabelecimento das restrições físicas do sistema e a especificação da função de utilidade. A Figura 4.8 ilustra estas condições.

Figura 4.8: Representação do problema de controle ótimo



Fonte: Modificado de (NAIDU, 2002; KIRK, 2004).

A função de utilidade toma a forma geral:

$$J = \phi_N(x_N) + \sum_{k=i}^{N-1} L_k(x_k, u_k, \cdot), \tag{39}$$

em que $[i, N]$ é o intervalo de interesse, $\phi_N(\cdot)$ retorna o custo para o sistema no estado terminal x_N e $L_k(\cdot)$ define o custo dos estados x_k e dos sinais de controle u_k em cada instante k no intervalo $[i, N)$.

A seguir será apresentado algumas funções de utilidade comuns para determinados problemas de controle ótimo (KIRK, 2004; LEWIS *et al*., 2012b).

a) ***Problemas de minimização de tempo***: Neste caso, o objetivo é determinar os sinais de controle que levem o sistema o mais rápido de um dado estado inicial x_0 a o estado final desejado $x \in \mathbb{R}^n$.

$$J = \sum_{k=0}^{N-1} 1 = N \tag{40}$$

b) ***Problemas de minimização do combustível:*** A finalidade neste caso é levar o sistema do estado inicial x_0 a um estado desejado $x \in \mathbb{R}^n$ consumido o mínimo de combustível. O termo "mínimo de combustível" depende das particularidades físicas do processo. A função de custo é definida tomando como fundamento que o sinal de controle é proporcional ao combustível gasto.

$$J = \sum_{k=0}^{N-1} |u_k| \tag{41}$$

c) ***Problemas de minimização de energia:*** A função de custo neste caso toma a forma:

$$J = \frac{1}{2} x_N^T S x_N + \frac{1}{2} \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k), \tag{42}$$

em que R e S são matrizes diagonais e definidas positiva e Q é simétrica e semi-definida positva. Neste caso, caso seja mais importante que os estados sejam pequenos, deve-se escolher Q com valores grandes, porém se é mais relevante que a energia do sinal de controle seja pequena, então deve-se escolher valores grandes para a matriz R. Por sim, dado que seja mais interessante que o estado final se torne pequeno, então nesse caso S deve ser elevado.

d) ***Problemas de rastreamento:*** Este problema de controle consiste em manter o estado x_k o mais próximo possível dos estados desejados d_k no intervalo $[i, N]$. Portanto, a função de custo tem a forma:

$$J = e_N^T e_N + \sum_{k=i}^{N-1} (e_k^T Q e_k + u_k^T R u_k), \tag{43}$$

em que se define $e_k = x_k - d_k$. As matrizes Q e R são diagonais e positivas definidas. Os elementos de Q são selecionados de forma a ponderar a importância dos componentes do vetor de erros.

5 PROJETO DO CONTROLADOR ÓTIMO BASEADO EM APRENDIZADO POR REFORÇO PARA UM MANIPULADOR ROBÓTICO

Os esquemas de aprendizado por reforço apresentam um atrativo relativo à possibilidade de implementação de um controlador ótimo *online* sem a necessidade do conhecimento da dinâmica robô. O método Ator-Crítico se destaca pelo fato do cálculo da estimativa da função Q ser realizada a cada passo de tempo, usando apenas os estados observados do processo. Neste projeto, duas técnicas de aprendizado por reforço foram implementadas para a solução do problema de rastreamento ótimo em um manipulador robótico com dois graus de liberdade e juntas rotativas. Estas técnicas diferem entre si quanto a representação da política e, por consequência, quanto a estratégia de atualização dos parâmetros da estrutura Ator. No primeiro esquema, a política de controle não é representada explicitamente, em vez disso, o cálculo de atualização do Ator é executado por meio de uma expressão exata que decorre da função Q. Já no segundo arranjo, a definição da política é explícita com o emprego, por exemplo, de uma estrutura de aproximação paramétrica.

5.1 ABORDAGEM DE CONTROLE

No contexto de controle ótimo e aprendizado por reforço, a noção de maximizar as recompensas futuras ponderadas é modificada para minimizar o custo de controle. Desta forma, o objetivo é determinar uma política de controle $h^*(x_k,\, d_k) = u_k^*$ que minimize o índice de desempenho (função valor)

$$V(x_k, d_k) = \sum_{i=k}^{\infty} \gamma^{i-k}\, r(x_i, u_i, d_i), \tag{44}$$

tendo como restrição o seguinte sistema não-linear de tempo discreto:

$$\begin{aligned} x_{k+1} &= f(x_k) + g(x_k)u_k \\ y_k &= x_k\,, \end{aligned} \tag{45}$$

em que $x_k \in \mathbb{R}^m$ é o vetor de estado, $u_k \in \mathbb{R}^n$ é o vetor de entrada de controle, $y_k \in \mathbb{R}^m$ é o vetor de saída, d_k é o vetor de trajetória desejada, $0 < \gamma \leq 1$ é o fator de desconto, e $r(\cdot)$ é a função de utilidade que retorna o custo de controle em um passo de tempo. Uma função de utilidade razoavelmente geral em problemas de minimização de energia é dada por:

$$r(x_i, u_i, d_i) = \tilde{r}(x_i, d_i) + u_i^T R u_i, \tag{46}$$

sendo R uma matriz definida positiva. O vetor d_i pode ser descrito como uma demanda de projeto, fazendo com que $\tilde{r}(\cdot)$ represente o custo para executar a tarefa desejada, como por exemplo, o custo de rastreamento.

Usando o princípio da otimalidade de Bellman, o índice de desempenho ótimo pode ser escrito como:

$$V^*(x_k, d_k) = \min_{u_k}\big(r(x_k, u_k, d_k) + \gamma V^*(x_{k+1}, d_{k+1})\big). \tag{47}$$

Em RL, uma variante da função valor $V(\cdot)$, chamada função Q, é usada. Tal função tem uma aplicação apropriada nos projetos de controle em que o modelo da planta não está disponível. A função Q associada à uma política de controle h é definida por:

$$Q^h(x_k,\, u_k,\, d_k) = r(x_k,\, u_k,\, d_k) + \gamma V^h(x_{k+1},\, d_{k+1}), \tag{48}$$

e a função Q ótima satisfaz a seguinte equação:

$$Q^*(x_k,\, u_k,\, d_k) = r(x_k,\, u_k,\, d_k) + \gamma V^*(x_{k+1},\, d_{k+1}). \tag{49}$$

Combinando as Equações (47) e (49), a equação da otimalidade de Bellman em termos da função Q é dada por:

$$V^*(x_k,\, d_k) = \min_{u_k}\big(Q^*(x_k,\, u_k, d_k)\big). \tag{50}$$

e a política de controle ótima é obtida por:

$$h^*(x_k,\, d_k) = \arg\min_{u_k} Q^*(x_k, u_k, d_k). \tag{51}$$

Desta forma, o sinal de controle pode ser calculado resolvendo-se:

$$\frac{\partial Q^*(x_k, u_k, d_k)}{\partial u_k} = 0. \tag{52}$$

5.1.1 Estrutura do Crítico

Os esquemas de aprendizado por reforço escolhidos para esse projeto utilizam-se da estratégia de aproximação da função valor de estado-ação. A estrutura parametrizada que aproxima a função Q, em sua forma geral, é dada por:

$$\hat{Q}_i(x_k,\, u_k,\, d_k,\, w_i) = w_i^T \phi(x_k,\, u_k,\, d_k), \tag{53}$$

sendo w_i a i-ésima estimação do vetor de pesos da rede neural e $\phi(\cdot)$ é o vetor de funções de ativação. Para este estudo, foi definida uma rede neural como em (38). Esta arquitetura de RNA pode ser capaz de aproximar funções contínuas desde que $\phi(\cdot)$ seja escolhido de modo a formar um vetor de funções de base (LEWIS *et al.,* 2004).

Para a estrutura parametrizada da função Q, considera-se que o valor desejado para estimação do parâmetro w_i é dado por:

$$\Delta_{objetivo} = \tilde{r}(x_k, d_k) + u_k^T R u_k + \gamma \hat{Q}_i(x_{k+1}, u_{k+1}, d_{k+1}) \tag{54}$$

O vetor de pesos w_i é calculado pela minimização, em um sentido dos mínimos quadrados, do erro de diferencial temporal, que é definido por:

$$\delta_k = r_k + \gamma \hat{Q}_i(x_{k+1}, u_{k+1}, d_{k+1}, w_i) - \hat{Q}_i(x_k, u_k, d_k, w_i). \tag{55}$$

Neste estudo, o vetor de função de ativação da função Q é construído por polinômios de ordem superior. O objetivo é inserir alguns elementos quadráticos e termos de até quarta ordem dos erros de rastreamento, dos estados e dos sinais de controle, de modo que a rede neural possa aprender as não-linearidades do manipulador. Abu-Khalaf e Lewis (2005), Vamvoudakis *et al.* (2011), Khan *et al.* (2012), Pradhan e Subudhi (2012), LIU *et al.* (2014), Zhang *et al.* (2017) e GUO *et al.* (2020) são alguns exemplos de estudos em que foram utilizadas uma estrutura parametrizada com funções de base polinomiais de ordem superior para aproximar a função valor em problemas de controle de sistemas não lineares.

Por simplificação, $\phi(\cdot)$ será representado utilizando o produto de Kronecker $\otimes$ com a exclusão dos termos redundantes (AL-TAMIMI *et al.*, 2007). Esta exclusão é necessária para que os elementos que compõem o vetor de funções de base $\phi(\cdot)$ tornam-se linearmente independentes. Portanto,

$$\phi(z_k) = z_k \otimes z_k\,, \tag{56}$$

em que:

$$z_k = [u_k^T \quad e_k^T \quad e_{k1}^2 \quad \ldots \quad e_{km}^2 \quad \dot{q}_{k1}^2 \quad \ldots \quad \dot{q}_{kn}^2]^T \tag{57}$$

de modo que $e_k = x_k - d_k$ é o erro de rastreamento.

A função Q definida neste trabalho para um manipulador genérico, toma a forma:

$$\begin{aligned}\hat{Q}_i(x_k, u_k, d_k, w_i) \\ &= w_{i,1}^T \phi_1(z_k) + w_{i,21}^T \phi_2(z_k) u_{k1} + w_{i,22}^T \phi_2(z_k) u_{k2} + \cdots \\ &+ w_{i,2n}^T \phi_2(z_k) u_{kn} + w_{i,31} u_{k1}^2 + w_{i,32} u_{k2}^2 + \cdots + w_{i,3n} u_{kn}^2,\end{aligned} \tag{58}$$

em que n é o número de juntas do robô. Logo, reescreve-se a Equação (56) baseado em (58) para obter $\phi(z_k) = [\phi_1^T(z_k) \quad \phi_2^T(z_k)u_{k1} \quad \phi_2^T(z_k)u_{k2} \quad \ldots \quad \phi_2^T(z_k)u_{kn} \quad u_{k1}^2 \quad \ldots \quad u_{kn}^2]^T$. Os elementos que compõem os vetores $\phi_1(\cdot)$ e $\phi_2(\cdot)$ são independentes de u_k.

5.1.2 Estrutura do Ator

Primeiramente será considerada a melhoria de política no caso em que a política não é representada explicitamente. Ao invés disso, ações gulosas são calculadas, de maneira exata, sob a demanda da função Q, de acordo com a Equação (51). Nesta situação, diz-se que a melhoria de política é exata.

Alternativamente, a política também pode ser representada explicitamente mediante uma aproximação paramétrica. À vista disso, uma das soluções para o cálculo da melhoria da política consiste em resolver o problema de regressão para atualizar os pesos do Ator visando a minimização dos valores estado-ação. Isto posto, tem-se:

$$\theta_{k+1} = \theta^+ \mid \theta^+ \in \arg\min_{\theta} Q^*(x_k, h(x_k, d_k, \theta), d_k), \tag{59}$$

em que θ é matriz de parâmetros ajustáveis do Ator (BUŞONIU *et al*., 2011).

5.1.2.1 Melhoria de política exata

A melhoria de política exata ocorre aplicando-se a Equação (52) para o caso em que a função Q assume a forma dada pela Equação (58). Portanto, tem-se, para o mecanismo com n graus de liberdade:

$$\frac{\partial \hat{Q}_i(x_k, u_k, d_k, w_i)}{\partial u_{kn}} = w_{i,2n}^T \phi_2(z_k) + 2w_{i,3n} u_{kn} = 0$$

$$u_{kn} = -\frac{1}{2w_{i,3n}} w_{i,2n}^T \phi_2(z_k) \tag{60}$$

Reorganizando na forma matricial, a política de controle pode ser escrita como:

$$h_i(x_k, d_k) = \begin{bmatrix} u_{k1} \\ u_{k2} \\ \vdots \\ u_{kn} \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} w_{i,31} & 0 & \dots & 0 \\ 0 & w_{i,32} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_{i,3n} \end{bmatrix}^{-1} \begin{bmatrix} \phi_2^T(z_k) & \mathbf{0_{1\times 5n}} & \dots & \mathbf{0_{1\times 5n}} \\ \mathbf{0_{1\times 5n}} & \phi_2^T(z_k) & \dots & \mathbf{0_{1\times 5n}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0_{1\times 5n}} & \mathbf{0_{1\times 5n}} & \dots & \phi_2^T(z_k) \end{bmatrix} w_{i,2}, \tag{61}$$

em que $w_{i,2} = \begin{bmatrix} w_{i,21}^T & w_{i,22}^T & \dots & w_{i,2n}^T \end{bmatrix}^T$.

Para o caso particular de um manipulador com dois graus de liberdade, a expressão simplifica-se para:

$$h_i(x_k, d_k) = -\frac{1}{2}\begin{bmatrix} w_{i,31} & 0 \\ 0 & w_{i,32} \end{bmatrix}^{-1} \begin{bmatrix} \phi_2^T(z_k) & \mathbf{0_{1\times 10}} \\ \mathbf{0_{1\times 10}} & \phi_2^T(z_k) \end{bmatrix} w_{i,2}, \tag{62}$$

em que $w_{i,2} = \begin{bmatrix} w_{i,21}^T & w_{i,22}^T \end{bmatrix}^T$. Dessa maneira, a RNA a ser implementada neste trabalho possui 78 neurônios.

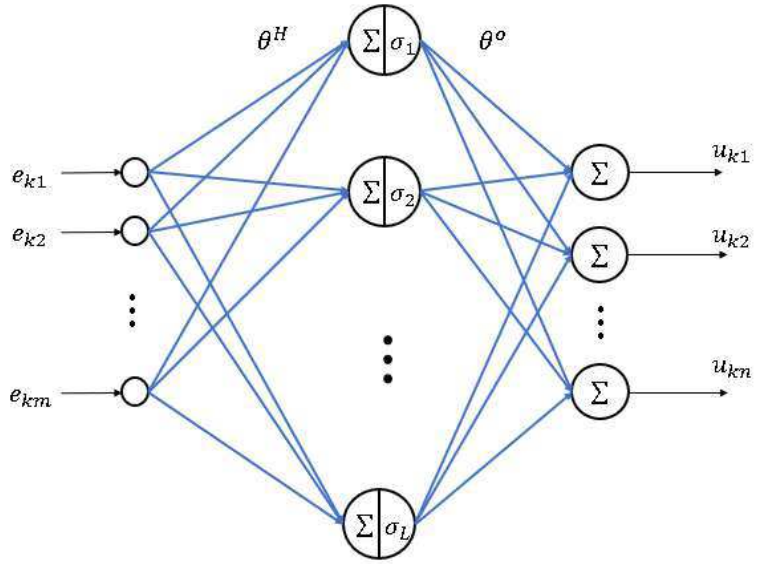
5.1.2.2 Melhoria de política aproximada

Já para o caso em que a política é representada explicitamente, a expressão de atualização da política dependerá da estrutura de aproximação escolhida. Para este projeto, uma rede neural artificial foi implementada para aproximar a política de controle do manipulador robótico. A topologia da rede escolhida é expressa pela Equação (37) e pode ser observada em mais detalhes na Figura 5.1. Os pesos entre a camada de entrada e a escondida (θ^H) são fixos e iniciados aleatoriamente, e somente os pesos da camada de saída (θ^o) são atualizados de maneira a minimizar os erros de aproximação. Logo:

$$h(x_k, d_k) = \theta^o \sigma(\theta^H e_k) \tag{63}$$

em que o vetor de funções de ativação para o Ator escolhido neste projeto é $\sigma = [\sigma_1 \quad \dots \quad \sigma_L]$, sendo $\sigma_i(s) = \frac{1-e^{-s}}{1+e^s}$ para $i = 1, \dots, L$, como observado nos trabalhos de controle não-linear via métodos Ator-Crítico GUO *et al*. (2014); HU e SI (2018), HUANG *et al*. (2018), e AL-DABOONI e WUNSCH (2019).

Figura 5.1: Esquema da rede do Ator



Fonte: Autor

A estratégia de aprendizado da rede do Ator consiste em indiretamente possibilitar que sejam produzidas ações de controle promissoras zerando o custo r_k a longo prazo. Portanto, os pesos são ajustados por meio da retropropagação do erro entre o objetivo final desejado U_c e a função Q, em que $U_c \equiv 0$. A política de controle é atualizada pela minimização da função objetivo $E_{a(k)} = \frac{1}{2} e_{a(k)}^2$, no qual $e_{a(k)}$ se resume a função Q como observada em (53). Pela regra de atualização dos parâmetros baseada em gradiente, tem-se:

$$\begin{aligned} \theta_{k+1}^o &= \theta_k^o + \Delta\theta_k^o = \theta_k^o - \eta \frac{\partial E_{a(k)}}{\partial \theta_k^o} \\ &= \theta_k^o - \frac{1}{2}\eta \left(\frac{\partial E_{a(k)}}{\partial \widehat{Q}_\iota} \frac{\partial \widehat{Q}_\iota}{\partial \phi_k} \frac{\partial \phi_k}{\partial u_k} \frac{\partial u_k}{\partial \theta_k^o} \right), \end{aligned} \tag{64}$$

em que $\eta > 0$ é a taxa de aprendizado. A expressão final de ajuste para o caso de um manipulador com duas juntas é:

$$\theta_{k+1}^o = \theta_k^o - \frac{1}{2}\eta\, \widehat{Q}_\iota (\sigma_k\, w_i^T [\varphi_1 \quad \varphi_2])^T \tag{65}$$

sendo w_i vetor de pesos do Crítico, $\varphi_1 = [2u_{k1} \quad u_{k2} \quad e_k^T \quad e_{k1}^2 \quad \dots \quad e_{k4}^2 \quad \dot{q}_{k1}^2 \quad \dot{q}_{k2}^2 \quad \mathbf{0_{1\times 66}}]^T$ e $\varphi_2 = [0 \quad u_{k1} \quad \mathbf{0_{1\times 10}} \quad 2u_{k2} \quad e_k^T \quad e_{k1}^2 \quad \dots \quad e_{k4}^2 \quad \dot{q}_{k1}^2 \quad \dot{q}_{k2}^2 \quad \mathbf{0_{1\times 55}}]^T$.

Por questões de convergência os pesos estimados do Ator são restritos a um intervalo apropriado fazendo:

$$\theta_k^o = \frac{\theta_k^o}{\max|\theta_k^o|}, \text{se } \max|\theta_k^o| > \theta_{max}, \tag{66}$$

em que $\max|\cdot|$ é o valor absoluto máximo entre as componentes do argumento e θ_{max} é o limite estabelecido para os pesos ajustáveis da política (GUO *et al*., 2014).

5.2 ALGORITMO DE CONTROLE

Para o problema de rastreamento considerado nesse estudo, a função de utilidade é definida como:

$$r(x_k,\, e_k,\, u_k) = e_k^T Q_c e_k + u_k^T R u_k + (u_{k+1} - u_k)^T S (u_{k+1} - u_k), \tag{67}$$

sendo $Q_c \in \mathbb{R}^{4\times4}$, $R \in \mathbb{R}^{2\times2}$ e $S \in \mathbb{R}^{2\times2}$ matrizes definidas positivas e diagonais.

O objetivo do algoritmo de controle engloba atualizar o vetor de parâmetros do Crítico e do Ator. Para realizar este procedimento *online*, o algoritmo iterativo dos mínimos quadrados recursivos (*Recursive Least-Squares* - RLS) será utilizado para aprender os parâmetros ótimos da função Q. O método RLS tem aplicações em diversas tarefas, tendo como uma de suas características a rápida convergência, o que se mostra adequado em problemas de estimação *online* e controle (XU *et al*., 2002).

Portanto, aplicando o algoritmo RLS, a estimação de w_i a cada passo de tempo k é dada por (ASTROM, 1989):

$$w_{k+1} = w_k + K_k \delta_k \tag{68}$$

em que:

$$K_k = \frac{P_k \phi(z_k)}{\lambda + \phi(z_k)^T P_k \phi(z_k)} \tag{69}$$

$$P_{k+1} = \frac{1}{\lambda}\left(P_k - \frac{P_k \phi(z_k) \phi(z_k)^T P_k}{\lambda + \phi(z_k)^T P_k \phi(z_k)}\right) \tag{70}$$

e λ é um parâmetro conhecido como fator de esquecimento de forma que $0 < \lambda \leq 1$.

Astrom (1989) recomenda que a matriz de correlação inversa, P, deve ser inicializada e redefinida periodicamente como βI para acelerar a convergência, onde β trata-se de um número positivo de valor alto e I é a matriz identidade com as dimensões apropriadas. Outro requisito de grande importância para a convergência do estimador RLS é o de excitação persistente. Esta

condição é relevante em problemas de identificação de sistemas, por gerar mais informações necessárias para a estimação. A excitação persistente também contribui em problemas de controle ótimo, pois favorece que a política convirja para sua solução ótima (GUO *et al*., 2020). Desse modo, para satisfazer essa condição, será injetado um sinal de ruído nas entradas de controle.

O método de aprendizado por reforço empregado, cujo Ator é representado implicitamente, exige uma política de controle inicial estável. A finalidade é manter o controlador estável durante os instantes iniciais até que o agente adquira experiência suficiente (observando o ambiente) para que uma nova política possa ser calculada. Por simplificação, os ganhos da rede neural devem ser inicializados de modo a resultar em um controlador Proporcional-Derivativo discreto.

Portanto o algoritmo de controle para esse esquema inicia-se com os ganhos da RNA definidos arbitrariamente para produzir o efeito de um controle PD e com a matriz P (do algoritmo RLS) inicializada com valores altos. Durante os primeiros instantes, não há atualização na política de controle para garantir a estabilidade durante o aprendizado inicial, entretanto o vetor de pesos w_k é calculado a cada passo aplicando a Equação (68). O sinal de controle é obtido em cada instante de tempo k usando (62). Ao fim desse período inicial, os pesos do controlador são atualizados, iniciando-se um novo período de aprendizagem. Para fornecer robustez ao algoritmo, a atualização dos parâmetros da política é obtida por:

$$w_{i+1(ctrl)} = \alpha w_{k+1} + (1-\alpha) w_{i(ctrl)} \tag{71}$$

em que $w_{i(ctrl)}$ são os parâmetros do controlador implementado durante o i-ésimo ciclo, $0 < \alpha \leq 1$ é o fator de aprendizado. Nesse instante, a matriz P é redefinida. Os pesos do controlador são novamente mantidos inalterados até que o ciclo em curso tenha se concluído. O processo é repetido até a convergência dos parâmetros da rede.

O algoritmo de aprendizado por reforço implementado neste estudo com melhoria de política exata é apresentado no Algoritmo 5.

Já para o caso do algoritmo de aprendizado com a política representada de forma explicita, a estimação da função Q manterá o esquema apresentado em (68). A cada instante de tempo k, o sinal de controle é calculado resolvendo (65). De modo a fornecer robustez ao esquema, um sinal de controle PD é adicionado a ação obtida pelo Ator para gerar o torque aplicado ao manipulador. O algoritmo de aprendizado por reforço implementado neste estudo com melhoria de política aproximada é apresentado no Algoritmo 6.

Algoritmo 5: Algoritmo de Controle RL com Melhoria de Política Exata

Entrada: fator de desconto γ, fator de aprendizado α, valor inicial da matriz de covariância β e o fator de esquecimento λ.

Inicialize os pesos da rede neural de forma a garantir um controlador PD estável.

Meça os estados x_0 e os erros de trajetória e_0 iniciais.

Inicialize as matrizes $P_0 = \beta I$ e Q_c, S e R arbitrariamente e $i = 0$.

Repetir

// **Ruído de Controle como componente de exploração**

$\xi = [\,]$

$u_k = h_i(x_k, d_k) + \xi$

Aplique o torque $\tau_k = u_k$ nas juntas do mecanismo e meça os estados x_{k+1}

$u_{k+1} = h_i(x_{k+1}, d_{k+1})$

$e_{k+1} = x_{k+1} - d_{k+1}$

$r_k = e_k^T Q_c e_k + (u_{k+1} - u_k)^T S(u_{k+1} - u_k) + u_k^T R u_k$

// **Mínimos Quadrados Recursivos - RLS**

$\Delta_{objetivo} = r_k + \gamma \hat{Q}(x_{k+1},\, u_{k+1},\, e_{k+1})$

$\widehat{W}_k = w_k^T \phi_k$

$K_k = \frac{P_k \phi_k}{\lambda + \phi_k^T P_k \phi_k}$

$w_{k+1} = w_k + K_k\left(\Delta_{objetivo} - \widehat{W}_k\right)$

$P_{k+1} = \frac{1}{\lambda}\left(P_k - \frac{P_k \phi_k \phi_k^T P_k}{\lambda + \phi_k^T P_k \phi_k}\right)$

Se fim de um período de aprendizado:

$w_{i+1(ctrl)} = \alpha w_{k+1} + (1 - \alpha) w_{i(ctrl)}$

// **Atualização da Política**

$h_{i+1} \leftarrow$

$$-\frac{1}{2}\begin{bmatrix} w_{i+1(ctrl),31} & 0 \\ 0 & w_{i+1(ctrl),32} \end{bmatrix}^{-1} \begin{bmatrix} \phi_2^T(z_k) & \mathbf{0_{1\times 10}} \\ \mathbf{0_{1\times 10}} & \phi_2^T(z_k) \end{bmatrix} w_{i(ctrl),2}$$

$P_{k+1} = \beta I$

$i = i + 1$

fim-se

até satisfazer o critério de parada

Algoritmo 6: Algoritmo de Controle RL com Melhoria de Política Aproximada

Entrada: fator de desconto γ, fator de aprendizado α, valor inicial da matriz de covariância β e o fator de esquecimento λ.

Inicialize os pesos da rede neural de forma a garantir um controlador PD estável.

Meça os estados x_0 e os erros de trajetória e_0 iniciais.

Inicialize as matrizes $P_0 = \beta I$ e Q_c, S e R arbitrariamente e $i = 0$.

Repetir

//Cálculo do sinal PD

$u_{k_{PD}} = K_P e_k + K_D \dot{e}_k$

// **Ruído de Controle como componente de exploração**

$\xi = [\,]$

$y_k = h(x_k, d_k)$

$u_k = y_k + u_{k_{PD}} + \xi$

Aplique o torque $\tau_k = u_k$ nas juntas do mecanismo e meça os estados x_{k+1}

$u_{k+1} = h(x_{k+1}, d_{k+1})$

$e_{k+1} = x_{k+1} - d_{k+1}$

$r_k = e_k^T Q_c e_k + (u_{k+1} - u_k)^T S (u_{k+1} - u_k) + u_k^T R u_k$

// **Mínimos Quadrados Recursivos - RLS**

$\Delta_{objetivo} = r_k + \gamma \hat{Q}(x_{k+1},\, u_{k+1},\, e_{k+1})$

$\widehat{W}_k = w_k^T \phi_k$

$K_k = \frac{P_k \phi_k}{\lambda + \phi_k^T P_k \phi_k}$

$w_{k+1} = w_k + K_k\left(\Delta_{objetivo} - \widehat{W}_k\right)$

$P_{k+1} = \frac{1}{\lambda}\left(P_k - \frac{P_k \phi_k \phi_k^T P_k}{\lambda + \phi_k^T P_k \phi_k}\right)$

// **Atualização da Política**

$\theta_{k+1}^o = \theta_k^o - \frac{1}{2}\eta\, \widehat{Q}_i\left(\sigma_k \;\; w_i^T [\varphi_1 \quad \varphi_2]\right)^T$

até satisfazer o critério de parada

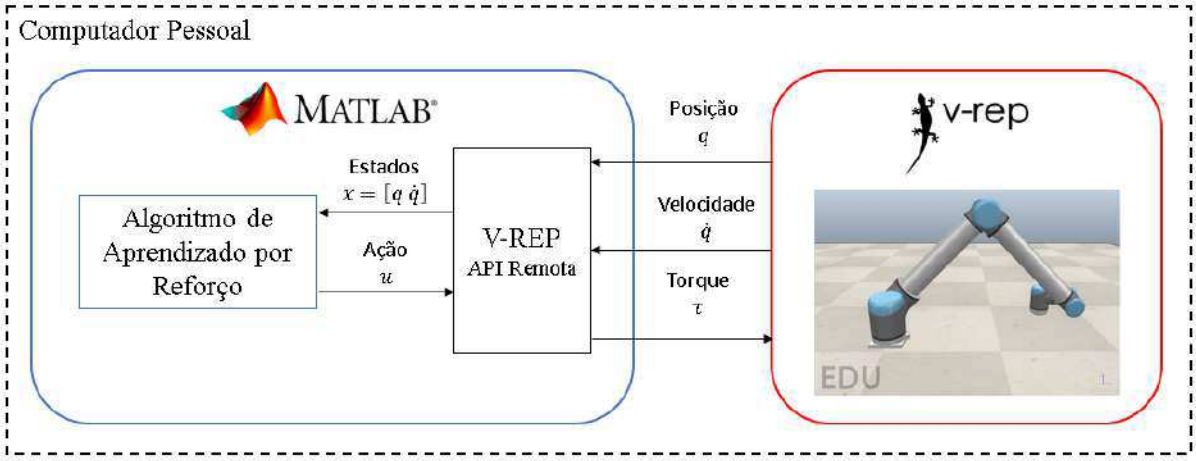
5.3 ESTRUTURA DE SIMULAÇÃO

De modo a fornecer uma estrutura de simulação que permita desenvolver os algoritmos propostos neste trabalho e realizar os experimentos, foi utilizado o software V-REP (*Virtual Robotics Experimentation Plataform*) em conjunto com o MATLAB (*Matrix Laboratory*). O

V-REP é um simulador para robôs de propósito geral desenvolvido pela empresa *coppelia Robotics* que fornece vários motores de física para as simulações, diversos modelos robóticos, múltiplas configurações do ambiente, entre outros recursos. Nesta plataforma, é possível personalizar todos os objetos da cena, incluindo os parâmetros dos sensores e atuadores, permitindo assim atingir resultados mais próximos a realidade (ROHMER *et al*., 2013).

No V-REP são disponibilizados diferentes meios de controlar os objetos/modelos na cena, seja através de rotinas embarcadas, nós do ROS (*Robot Operating System*) (QUIGLEY *et al*., 2015), API (*Application Programming Interface*) remota, um *plugin* ou alguma solução personalizada. Os algoritmos de controle podem ser escritos em C/C++, Python, Java, Lua e MATLAB. Os três elementos principais desse simulador são os objetos (juntas, corpos, sensors etc.), os módulos de cálculos (cinemático, dinâmico, de detecção de colisões etc.) e mecanismos de controle (API remota, rotinas embarcadas etc.). Uma vantagem também do programa é o fato de ele ser multiplaforma, o que significa que ele pode ser executado em Windows e Linux (SHAMSHIRI *et al*., 2018). Neste estudo, o modelo robótico usado no simulador é controlado por uma rotina externa desenvolvida na plataforma MATLAB fazendo uso da API remota. A Figura 5.2 ilustra a comunicação entre o controlador e o ambiente de simulação.

Figura 5.2: Esquema de controle do V-REP por API remota via MATLAB



Fonte: Autor

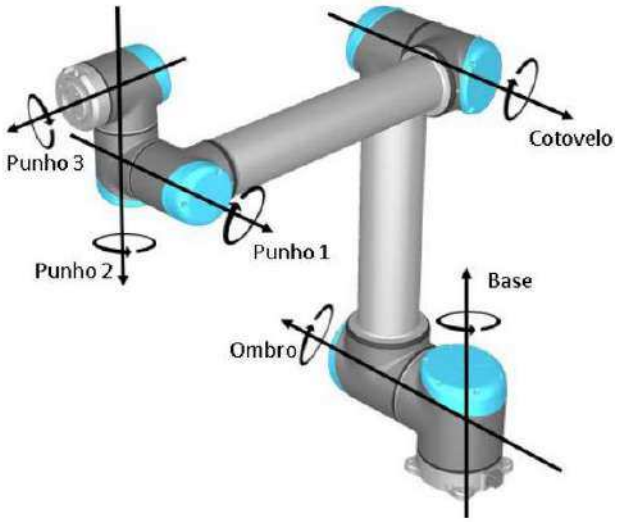
A API remota realiza a comunicação entre o MATLAB e o simulador mediante o protocolo TCP (*Transmission Control Protocol* - Protocolo de Controle de Transmissão), que possibilita, por exemplo, executar o controlador e o V-REP em máquinas distintas. Na rotina criada no MATLAB, deve constar o comando que estabelece a conexão com o servidor operando dentro do simulador antes de iniciar o ambiente. Em seguida, com a conexão estabelecida, o modo de sincronização deve ser configurado para permitir que o laço do

algoritmo do controlador externo execute a cada passo de tempo da simulação. Uma vez operando a rotina de controle e o ambiente no simulador, é possível acessar os objetos da cena para ler ou escrever nas variáveis desejadas (PLUŠKOSKI *et al*., 2019).

6 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados obtidos através de simulações dos esquemas de controle propostos neste trabalho. Os experimentos computacionais montados visam avaliar o comportamento do sistema para o problema de regulação, rastreamento e variações na carga de trabalho. Para execução desses experimentos computacionais foi utilizado o modelo disponível do braço robótico UR10 no simulador V-REP. Visto que este articulador possui seis graus de liberdade, nestes ensaios o torque gerado pela lei de controle será aplicado apenas nas juntas do ombro e do cotovelo (Figura 6.1) enquanto as demais juntas são desativadas e bloqueadas em suas respectivas posições de equilíbrio (0º). O controle foi realizado utilizando a API remota através de rotinas implementadas na plataforma MATLAB.

Figura 6.1: Juntas do manipulador UR10



Fonte: Autor

6.1 ESQUEMA DE CONTROLE RL COM MELHORIA DE POLÍTICA EXATA

Neste primeiro esquema de aprendizado, é necessário que a política inicial seja estável. Um controlador discreto PD foi escolhido para atender esta condição. Isto pode ser implementado modificando os pesos da Equação (62), onde observa-se que:

$$\phi_2(z_k) = [e_{k1} \quad e_{k2} \quad e_{k3} \quad e_{k4} \quad e_{k1}^2 \quad e_{k2}^2 \quad e_{k3}^2 \quad e_{k4}^2 \quad x_{k3}^2 \quad x_{k4}^2], \tag{72}$$

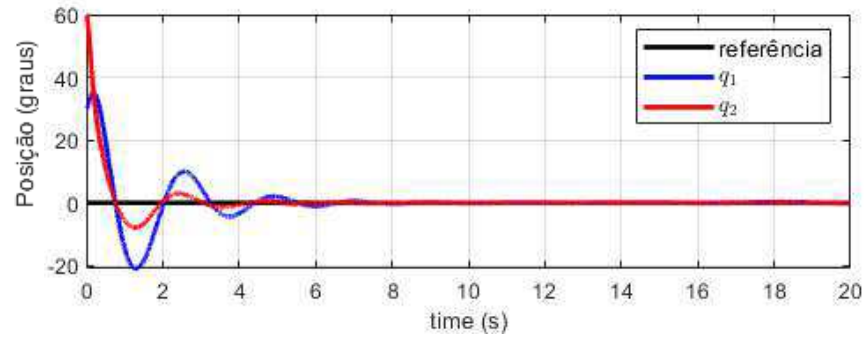
ou seja, $h(x_k, d_k)$ depende diretamente dos erros de posição e velocidade dos elos. Isto posto, constata-se que facilmente pode-se obter um controle PD fazendo, por exemplo:

$$w_{i,31} = w_{i,32} = \frac{1}{2},$$
$$w_{i,21} = [K_{P_1} \quad 0 \quad K_{D_1} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T, \tag{73}$$
$$w_{i,22} = [0 \quad K_{P_2} \quad 0 \quad K_{D_2} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T,$$

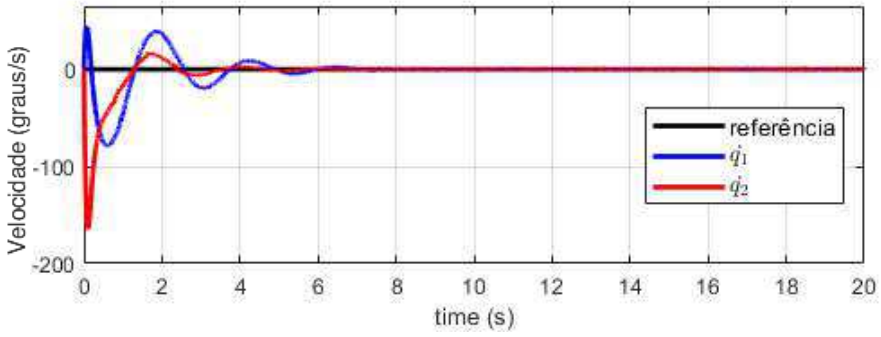
sendo K_{P_1} e K_{D_1}, e K_{P_2} e K_{D_2} os ganhos proporcional e derivativo, respectivamente, das juntas do ombro e do cotovelo O ajuste desses parâmetros será realizado por tentativa e erro.

O comportamento dos estados para o caso de regulação é apresentado na Figura 6.2. A configuração inicial das juntas foi definida como $x_0 = [\pi/6 \quad \pi/3 \quad 0 \quad 0]^T$ e os parâmetros do controlador foram ajustados para os seguintes valores $K_{P_1} = K_{P_2} = 150$, $K_{D_1} = K_{D_2} = 30$, $\gamma = 0{,}98$, $Q_c = diag(250,\ 250,\ 0{,}001,\ 0{,}001)$, $R = diag(0{,}0001,\ 0{,}0001)$, $\alpha = 0{,}2$ e $P_0 = 10^4 I_{78\times 78}$. O ciclo de aprendizado para esta simulação foi de 0,8 s. O esforço de controle aplicado nas juntas é apresentado na Figura 6.3 e a atualização dos pesos do Ator na Figura 6.5. Na Figura 6.4, por sua vez, é exibido o comportamento dos custos de controle, assim como também a função de custo estimada pela função $\hat{Q}$, em que se observa ambos os sinais convergindo para a região próxima de zero.

Figura 6.2: Esquema RL com Melhoria de Política Exata - Regulação: a) Posição e b) Velocidade de Ambas as Juntas



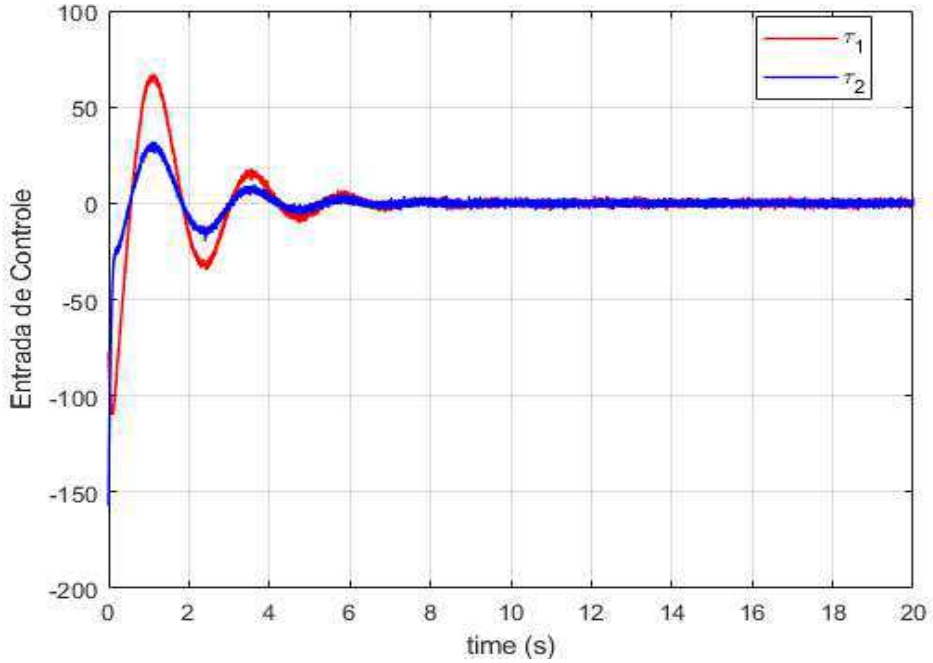
(a)



(b)

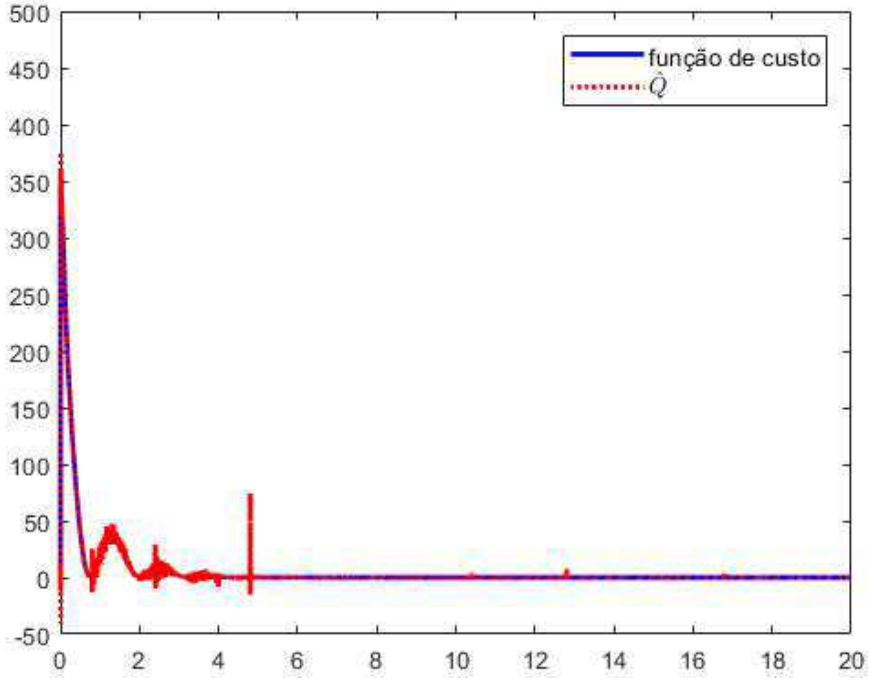
Fonte: Autor

Figura 6.3: Esforço de Controle pelo Esquema RL com Melhoria de Política Exata – Regulação



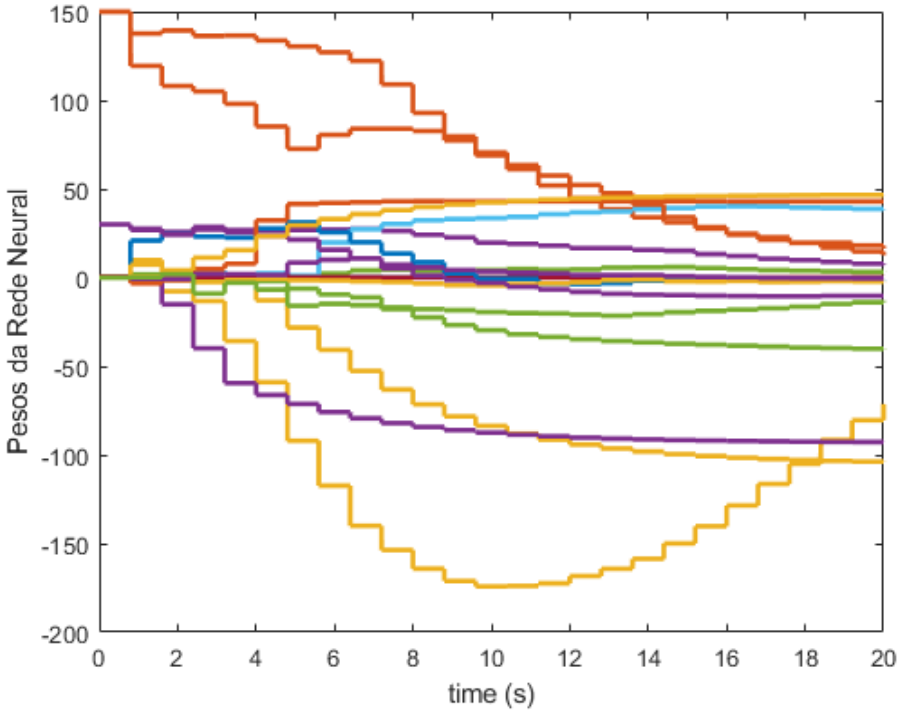
Fonte: Autor

Figura 6.4: Função de Custo vs Função de Custo Estimada ($\hat{Q}$) pelo Esquema RL com Melhoria de Política Exata - Regulação



Fonte: Autor

Figura 6.5: Atualização dos Pesos da Rede do Ator pelo Esquema RL com Melhoria de Política Exata - Regulação



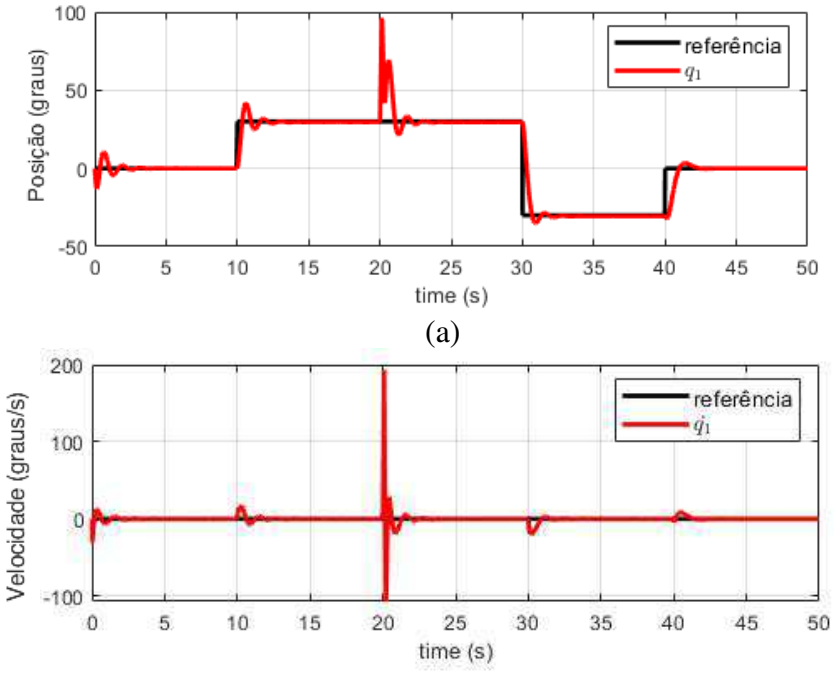
Fonte: Autor

Na segunda experiência sugerida para validar o controlador implementado, foi utilizado um sinal de referência de múltiplos degraus, de modo a simular a tarefa de pegar e colocar (*pick and place*), comumente realizada por manipuladores. Para este experimento o estado inicial foi configurado em $x_0 = [0 \quad 0 \quad 0 \quad 0]^T$. Os parâmetros de controle foram os mesmos utilizados para o caso de regulação exceto para os valores seguintes $Q_c = diag(100, 100, 0{,}001, 0{,}001)$, $\alpha = 0{,}1$, $K_{P_1} = K_{P_2} = 500$, $K_{D_1} = K_{D_2} = 50$ e ciclo de aprendizado alterado para 2 s. Sob estes ajustes, a resposta de rastreamento e o torque aplicado nas juntas são apresentados nas Figuras 6.10 o comportamento dos pesos do Ator.

Figura 6.6 a 6.8. Como visto, as juntas são capazes de rastrear o sinal de referência com erros dentro dos limites toleráveis e a estabilidade do sistema é mantida durante todo o tempo de simulação. É mostrado também que no instante de tempo de 20 s houve um aumento no sinal de controle causando um sobressinal indesejado, porém nos tempos seguintes, a partir da 15ª atualização da política (30 s) observou-se um aprimoramento no rastreamento em relação a política inicial (primeiros 2 s), consequência do aprendizado adquirido.

Já na Figura 6.9, é mostrado que as estimativas da função de custo pela função $\hat{Q}$ são indesejadas nos instantes em que há variação na amplitude do sinal de entrada. Entretanto, algumas iterações após estes instantes, o comportamento esperado é estabelecido. Por fim é mostrado na Figura 6.10 o comportamento dos pesos do Ator.

Figura 6.6: Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal de Múltiplos Degraus: a) Posição e b) Velocidade Simuladas da Junta do Ombro

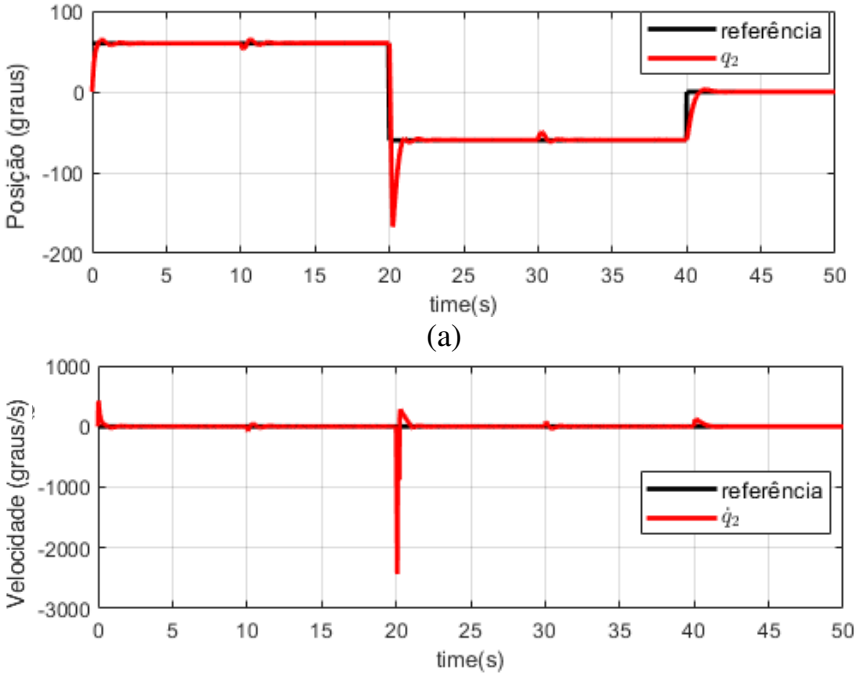


(a)

(b)

Fonte: Autor

Figura 6.7: Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal de Múltiplos Degraus: a) Posição e b) Velocidade Simuladas da Junta do Cotovelo

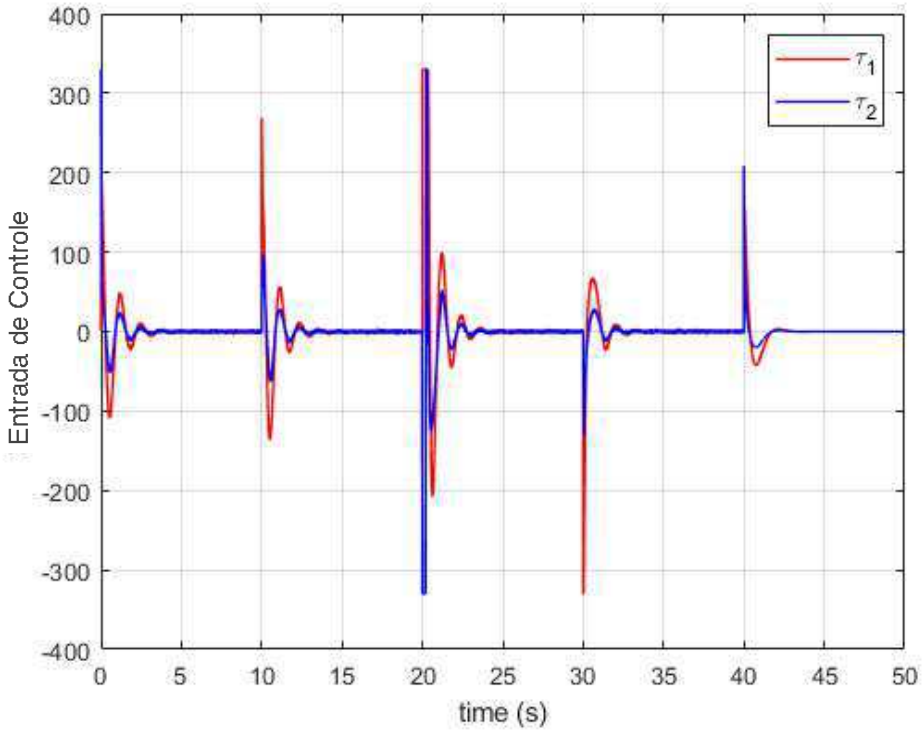


(a)

(b)

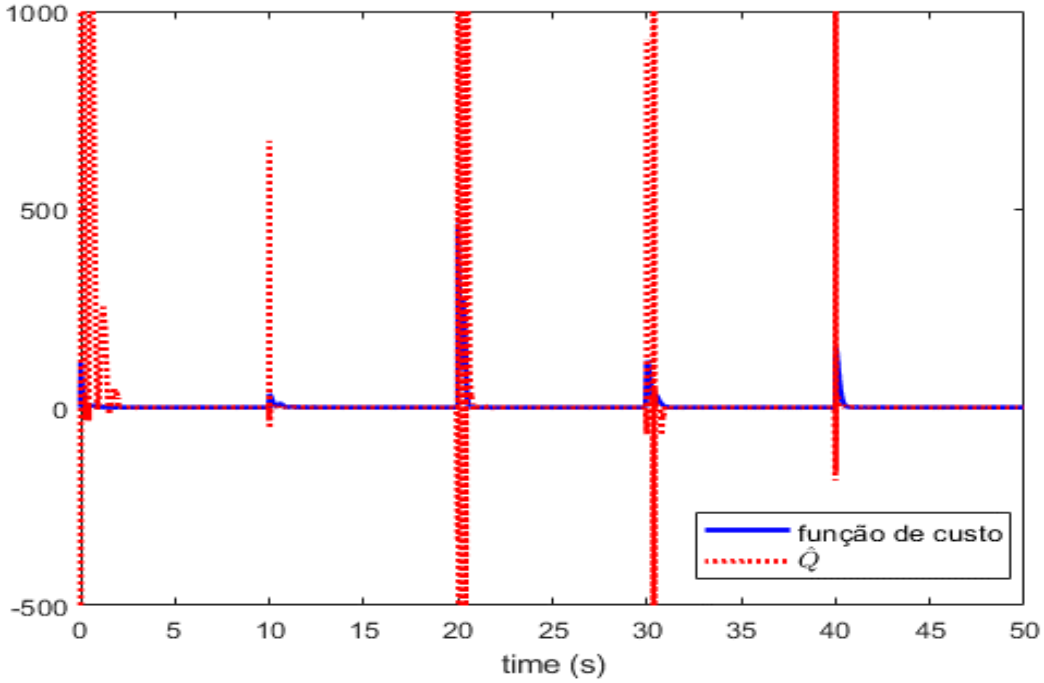
Fonte: Autor

Figura 6.8: Esforço de Controle pelo Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal de Múltiplos Degraus



Fonte: Autor

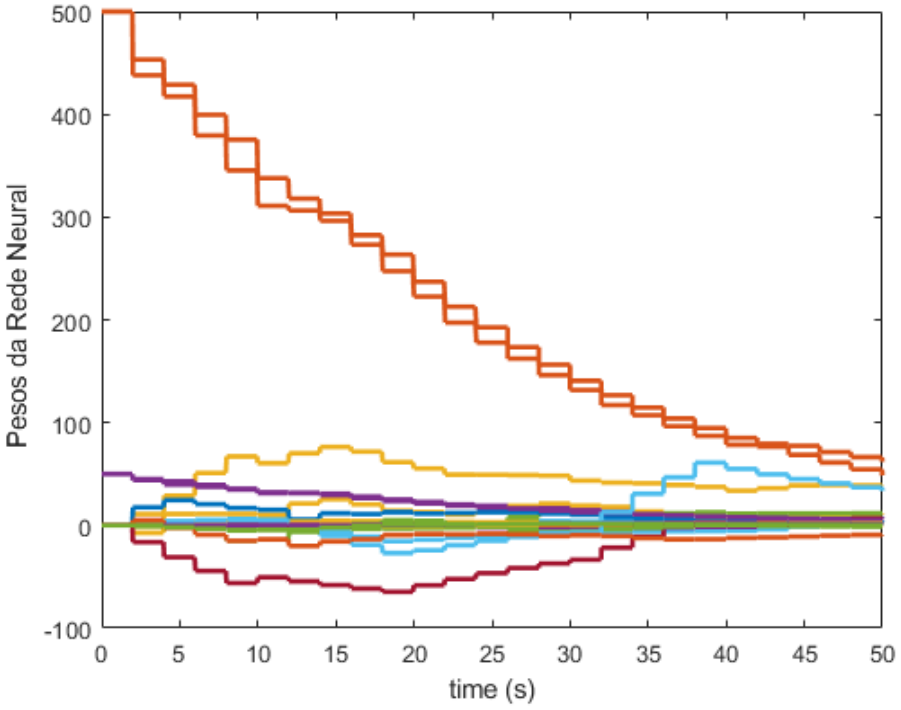
Figura 6.9: Função de Custo vs Função de Custo Estimada ($\hat{Q}$) Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal de Múltiplos Degraus



Fonte: Autor

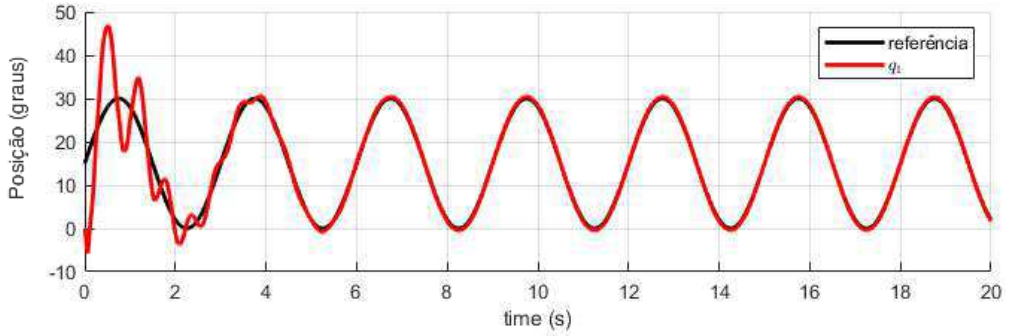
Para o terceiro experimento proposto, um sinal senoidal foi estabelecido como referência para as juntas do articulador sob os seguintes ajustes $Q_c = diag(200, 200, 0{,}001, 0{,}001)$, $K_{P_1} = 4000$, $K_{P_2} = 2000$, $K_{D_1} = 50$, $K_{D_2} = 20$ e $\alpha = 0{,}4$. Os demais parâmetros foram configurados nos mesmos valores do experimento 2. Os resultados da simulação são observados nas Figuras 6.11 a 6.15. De acordo com as Figuras 6.11 e

Figura 6.12, onde é mostrado o desempenho de rastreamento, observa-se o aprimoramento do seguimento de trajetória ao fim de cada ciclo de aprendizado (intervalos de 2 s). A partir do terceiro ciclo os erros de rastreamento se estabilizam dentro de limites toleráveis.Figura 6.10: Atualização dos Pesos da Rede do Ator pelo Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal de Múltiplos Degraus

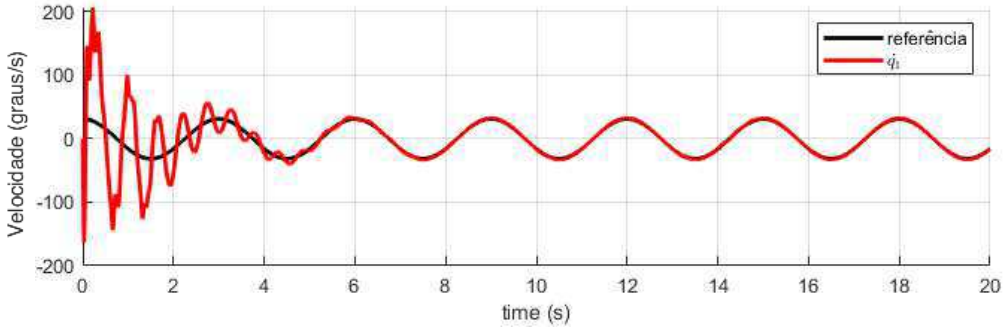


Fonte: Autor

Figura 6.11 Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal Senoidal: a) Posição e b) Velocidade Simuladas da Junta do Ombro



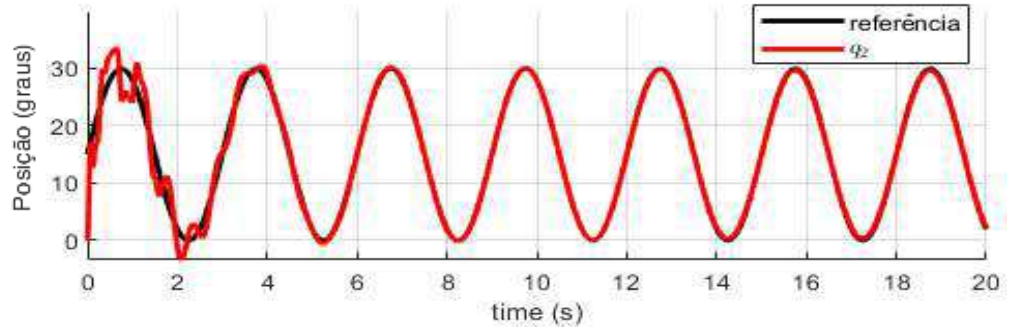
(a)



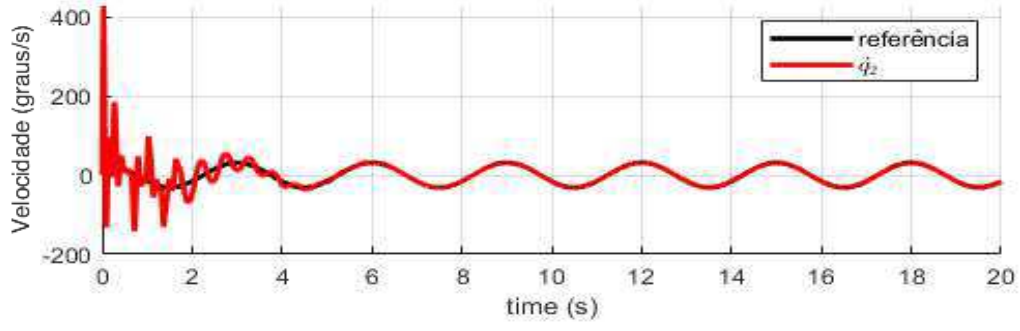
(b)

Fonte: Autor

Figura 6.12: Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal Senoidal: a) Posição e b) Velocidade Simuladas da Junta do Cotovelo



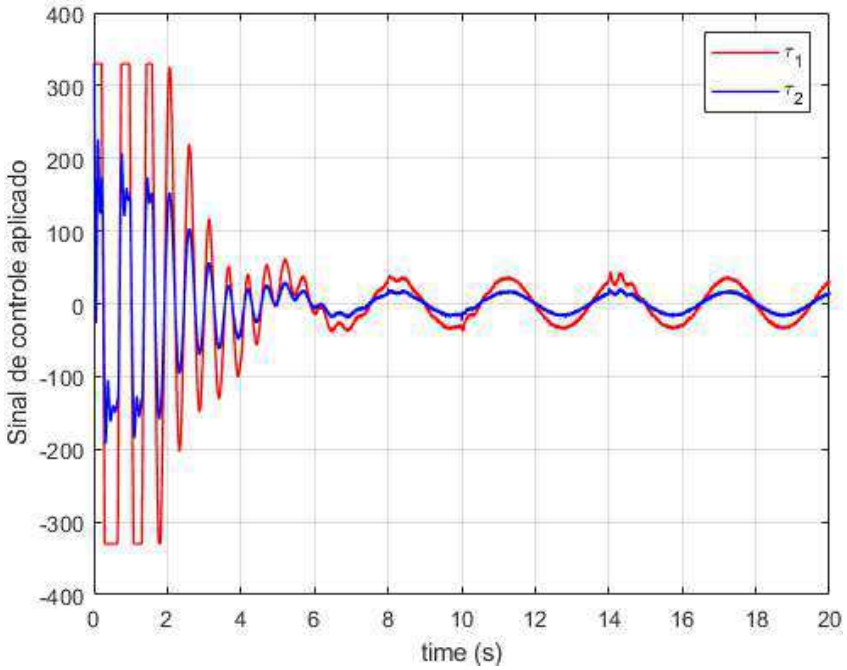
(a)



(b)

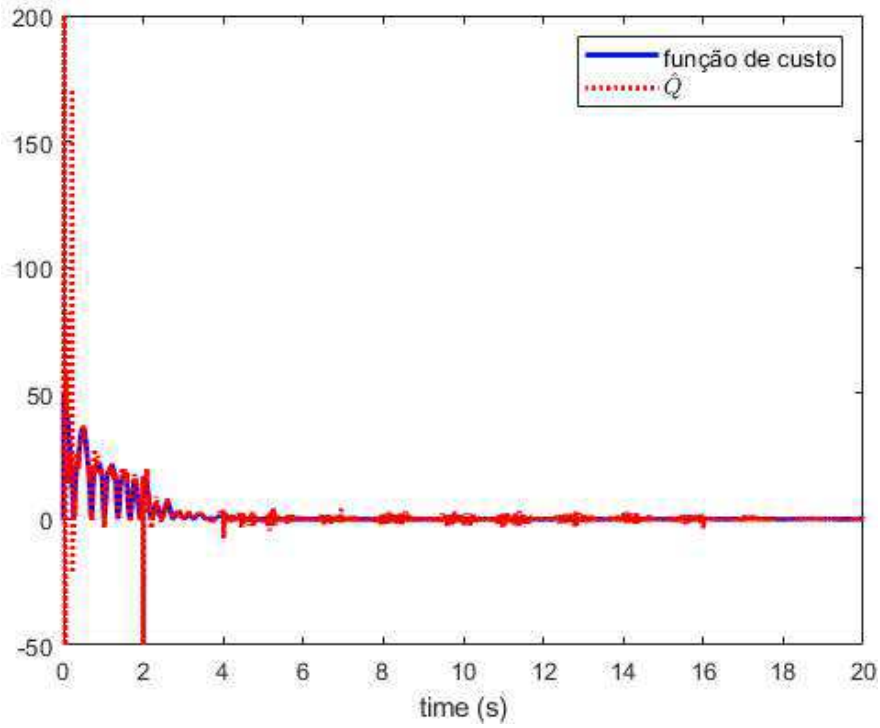
Fonte: Autor

Figura 6.13: Esforço de Controle pelo Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal Senoidal



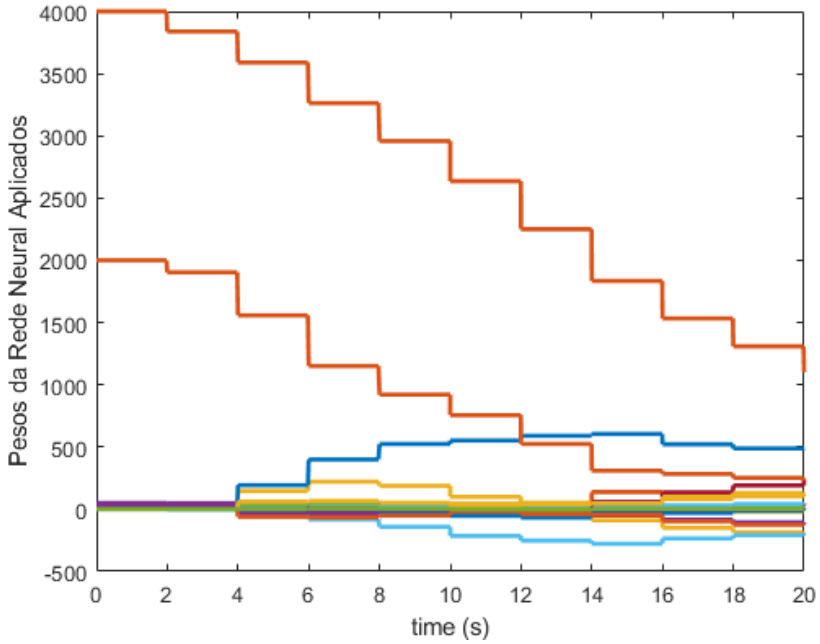
Fonte: Autor

Figura 6.14: Função de Custo vs Função de Custo Estimada ($\hat{Q}$) pelo Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal Senoidal



Fonte: Autor

Figura 6.15: Atualização dos Pesos da Rede do Ator pelo Esquema RL com Melhoria de Política Exata - Rastreamento de um Sinal Senoidal

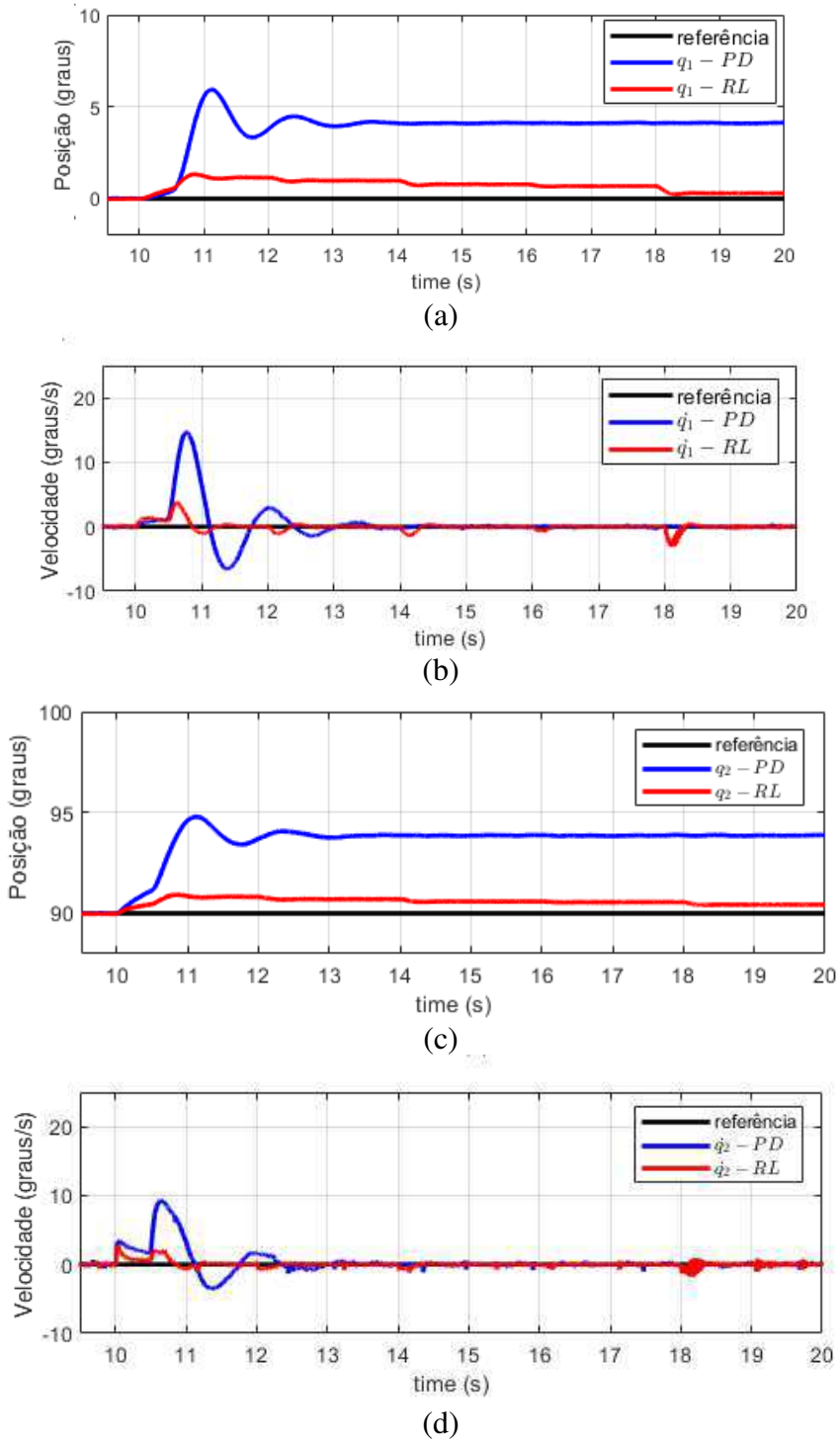


Fonte: Autor

Por fim, variações nos parâmetros do manipulador foram realizadas através da adição de uma carga no efetuador do mecanismo. Três valores para a massa da carga foram definidos para os testes: 5 kg, 2 kg e 1 kg e o comportamento dos estados são observados nas Figuras 6.16 a 6.18. Um degrau unitário foi escolhido como sinal de referência, sendo $d_k = \begin{bmatrix} 0 & \pi/2 & 0 & 0 \end{bmatrix}$. Em cada experimento, a carga de trabalho foi adicionada no instante de 10 s, após a acomodação das variáveis de estado, e mantida constante até o fim da simulação. Para demostrar o efeito compensador da estratégia em estudo, é exibido em azul o desempenho da política inicial adotada (controle PD), em contraste ao comportamento observado com a aplicação do controlador proposto (em vermelho).

Para o primeiro caso (carga com massa igual a 5 kg), os parâmetros do controlador foram ajustados para os seguintes valores $K_{P_1} = K_{P_2} = 500$, $K_{D_1} = K_{D_2} = 50$, $\gamma = 0{,}94$, $Q_c = diag(750,\ 750,\ 0{,}001,\ 0{,}001)$, $R = diag(0{,}0001,\ 0{,}0001)$, $\alpha = 0{,}5$, $P_0 = 10^4 I_{78\times 78}$ e ciclo de aprendizado igual a 2 s. Neste experimento, nota-se uma resposta lenta à perturbação inserida, mas ainda sim uma capacidade de rastreamento superior a política inicial, apresentando erros que diminuem a cada fim de um ciclo de aprendizado. Ao final do tempo estabelecido de simulação, o erro de posicionamento máximo foi de 0.43º o que é aceitável em diversas aplicações.

Figura 6.16: Esquema RL com Melhoria de Política Exata - Carga de Trabalho de 5 kg: a) Posição e b) Velocidade Simuladas da Junta do Ombro, e c) Posição e d) Velocidade Simuladas da Junta do Cotovelo

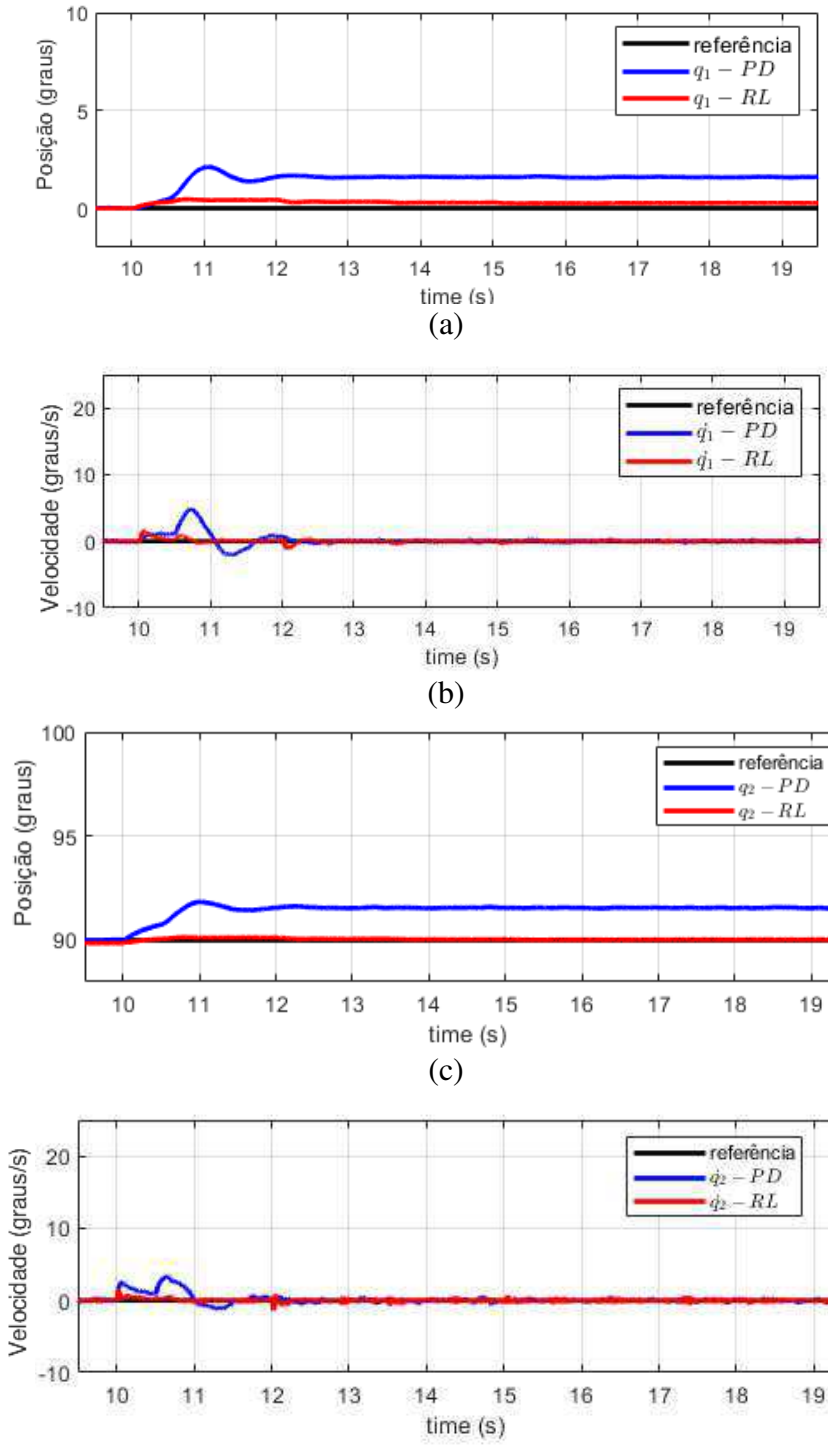


Fonte: Autor

Já para o experimento com carga de 2 kg, os mesmos parâmetros do controlador para o caso anterior foram implementados, exceto: $Q_c = diag(700, 700, 0{,}001, 0{,}001)$, $\alpha = 0.5$ e ciclo de aprendizado 1.5 s. Observa-se o mesmo comportamento do teste com 5 kg, mas com resposta mais rápida e erros menores. Por fim, na terceira situação, o desempenho do

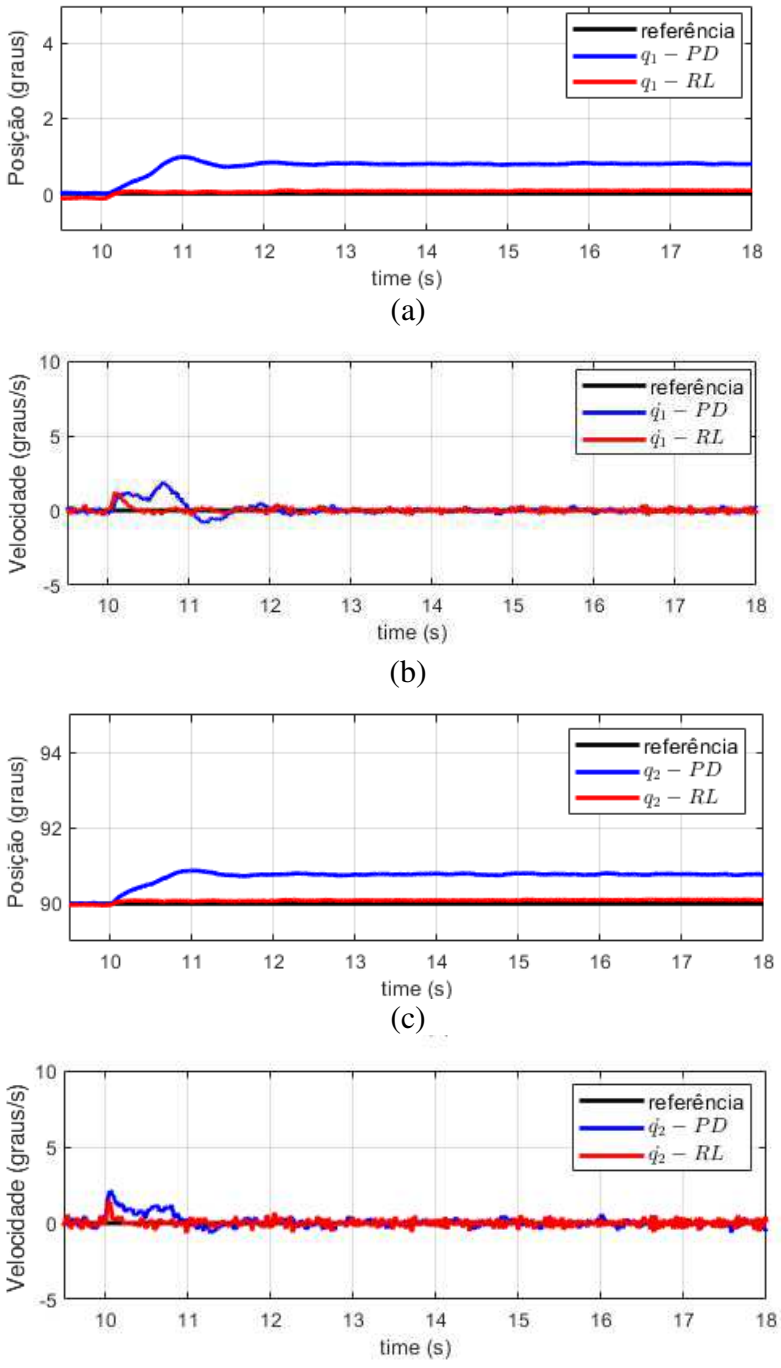
controlador foi altamente satisfatório, mantendo a posição de rastreamento próxima de zero já no instante após a adição da carga de 1 kg, lidando rapidamente com a perturbação. Os parâmetros do controlador para este caso são mesmos do experimento com carga de 2 kg.

Figura 6.17: Esquema RL com Melhoria de Política Exata - Carga de Trabalho de 2 kg: a) Posição e b) Velocidade Simuladas da Junta do Ombro, e c) Posição e d) Velocidade Simuladas da Junta do Cotovelo



Fonte: Autor

Figura 6.18: Esquema RL com Melhoria de Política Exata - Carga de Trabalho de 1 kg. a) Posição e b) Velocidade Simuladas da Junta do Ombro, e c) Posição e d) Velocidade Simuladas da Junta do Cotovelo



Fonte: Autor

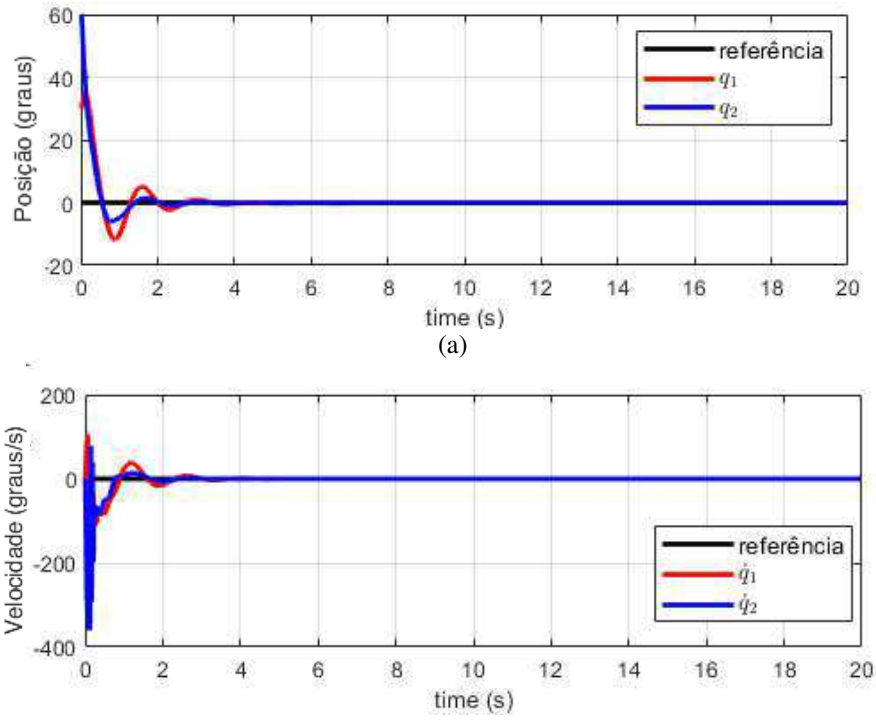
6.2 ESQUEMA DE CONTROLE RL COM MELHORIA DE POLÍTICA APROXIMADA

Já nesta estratégia, para proporcionar robustez ao sistema, uma malha PD é incluída, somando-se ao sinal gerado pelo Ator para produzir o torque aplicado nas juntas do robô. Os

sinais de referência utilizados neste esquema possuem as mesmas caraterísticas dos aplicados na estratégia anterior. Já em relação as simulações a seguir, os mesmos ganhos proporcional e derivativo serão empregados, sendo: $K_{P_1} = K_{P_2} = 500$ e $K_{D_1} = K_{D_2} = 50$. Os demais parâmetros do controlador também são iguais para os diferentes sinais de referência e são os seguintes: $Q_c = diag(200, 200, 0{,}001, 0{,}001)$, $R = diag(0{,}0001, 0{,}0001)$, $\gamma = 0{,}9$, $P_0 = 10^4 I_{78\times78}$, $\eta = 0.3$ e $L = 12$, em que L é o número de neurônios da camada escondida da rede do Ator. Já a matriz de pesos ϑ^H, que pondera as entradas da rede do Ator, é iniciada com valores aleatórios no intervalo de -1 a 1 e mantida fixa durante toda a simulação.

No problema de regulação, o comportamento dos estados, do sinal de controle, da função de custo e da convergência dos pesos da política são observados nas Figuras 6.19 a 6.22, respectivamente. Nota-se pelos resultados um bom desempenho de rastreamento, com erros dentro da faixa de precisão desejada e apresentando estabilidade durante todo o período de simulação. No primeiro segundo, observa-se também uma forte oscilação no sinal de controle, deixando explícita a fase de aprendizagem do Ator. Constata-se também uma boa estimação da função de custo pela função $\hat{Q}$ já nos primeiros instantes de simulação.

Figura 6.19: Esquema RL com Melhoria de Política Aproximada - Regulação: a) Posição e b) Velocidade Simuladas de Ambas as Juntas

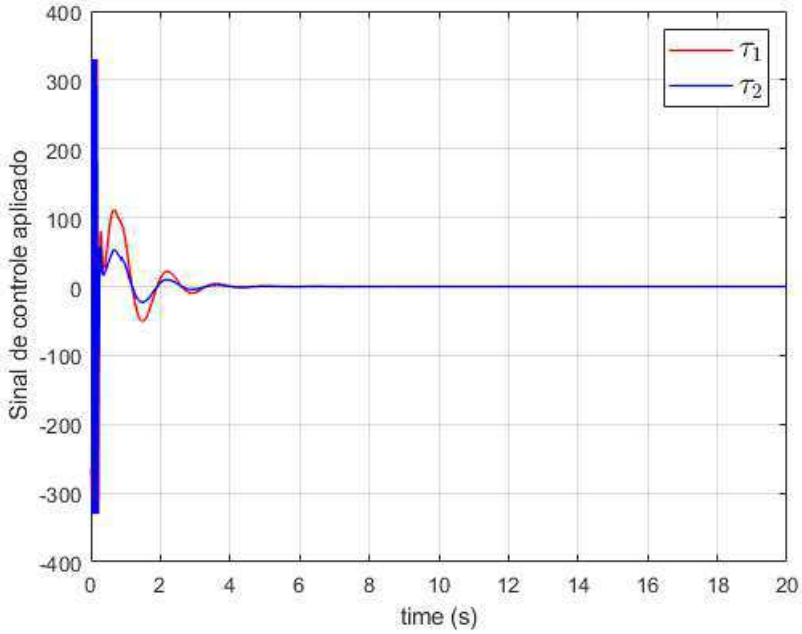


(a)

(b)

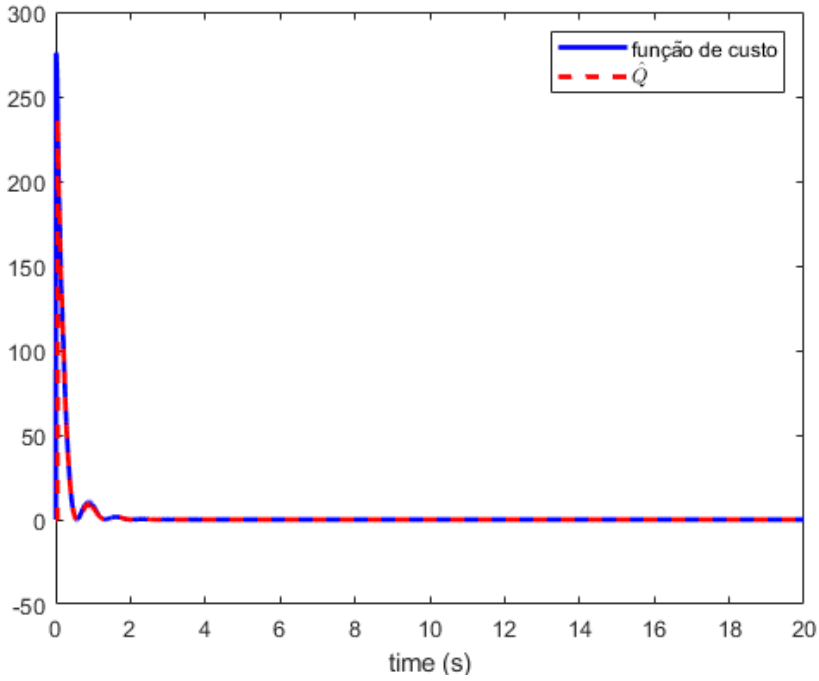
Fonte: Autor

Figura 6.20: Esforço de Controle pelo Esquema RL com Melhoria de Política Aproximada - Regulação



Fonte: Autor

Figura 6.21: Função de Custo vs Função de Custo Estimada ($\hat{Q}$) pelo Esquema RL com Melhoria de Política Aproximada - Regulação

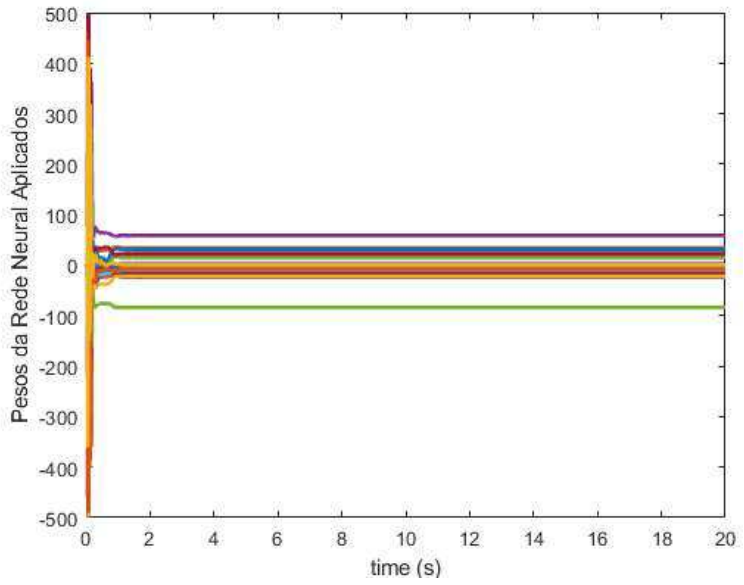


Fonte: Autor

O sinal de referência de múltiplos degraus foi adotado no segundo teste em simulação. Novamente, as juntas do mecanismo rastrearam o sinal de referência, como ilustrado nas Figuras 6.23 a 6.25. É observado que nos primeiros 30 s de simulação a junta do ombro apresentou sobressinais maiores que a junta do cotovelo, enquanto esta segunda exibiu erros de

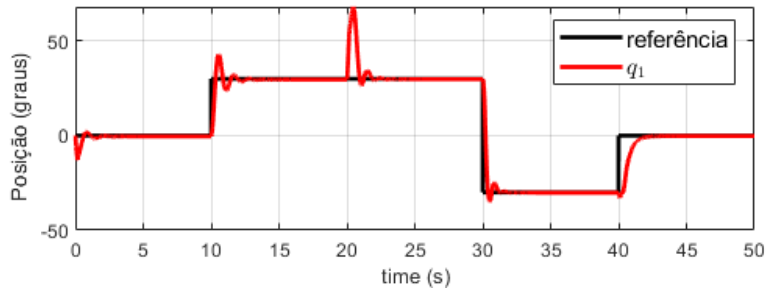
regime superiores à da primeira junta. Entretanto, após este intervalo, o comportamento dos estados tornou-se mais desejável, com oscilações e erros menores. Por fim, nas Figuras 6.26 e 6.27 são mostradas, respectivamente, a função de custo e a atualização dos pesos do Ator.

Figura 6.22: Atualização dos Pesos da Rede do Ator pelo Esquema RL com Melhoria de Política Aproximada - Regulação

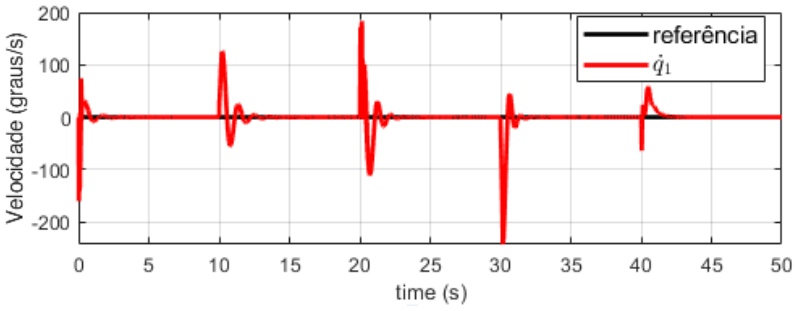


Fonte: Autor

Figura 6.23: Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal com Múltiplos Degraus: a) Posição e b) Velocidade Simuladas da Junta do Ombro



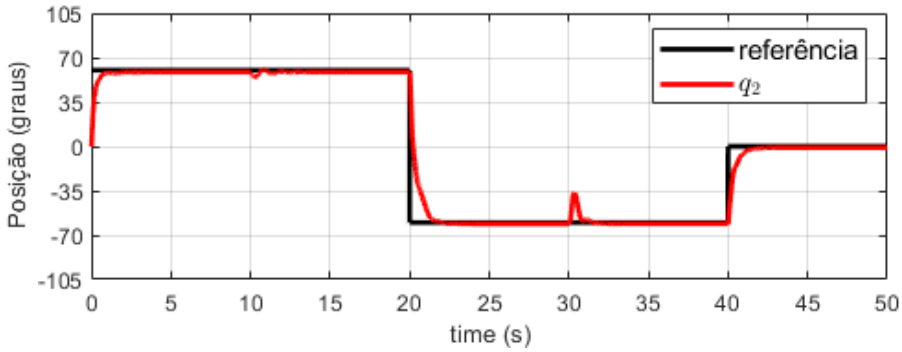
(a)



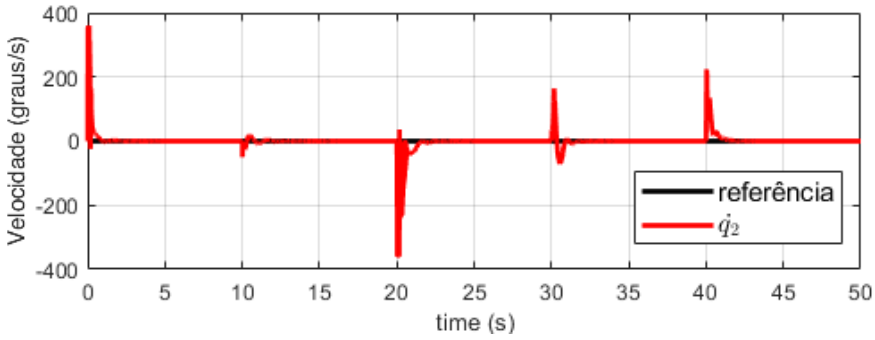
(b)

Fonte: Autor

Figura 6.24: Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal com Múltiplos Degraus: a) Posição e b) Velocidade Simuladas da Junta do Cotovelo



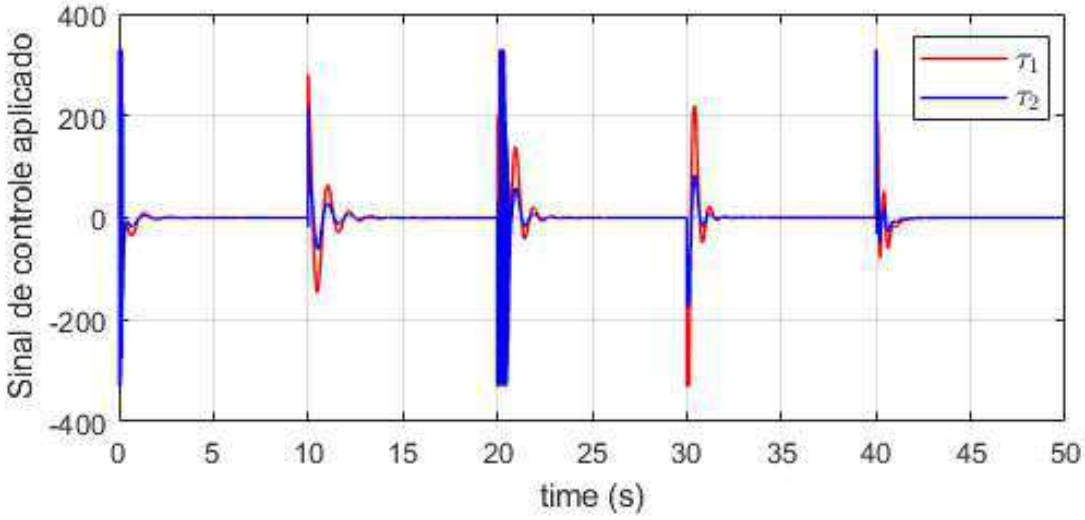
(a)



(b)

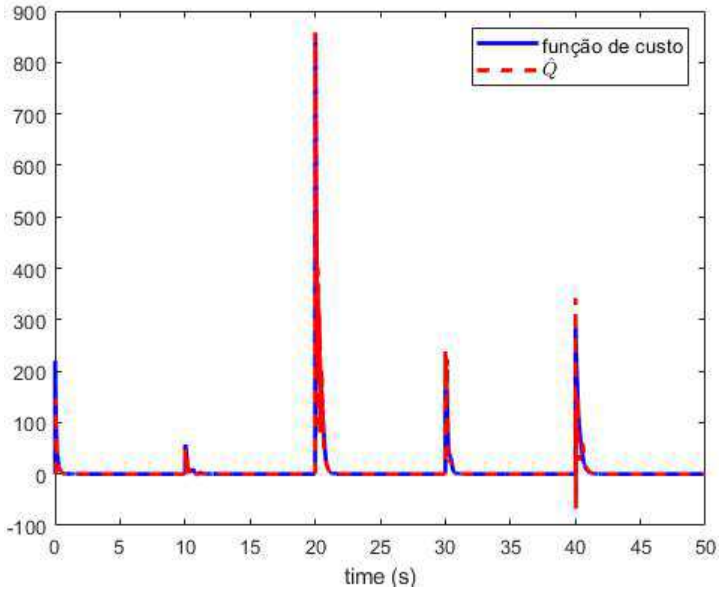
Fonte: Autor

Figura 6.25: Esforço de Controle Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal com Múltiplos Degraus



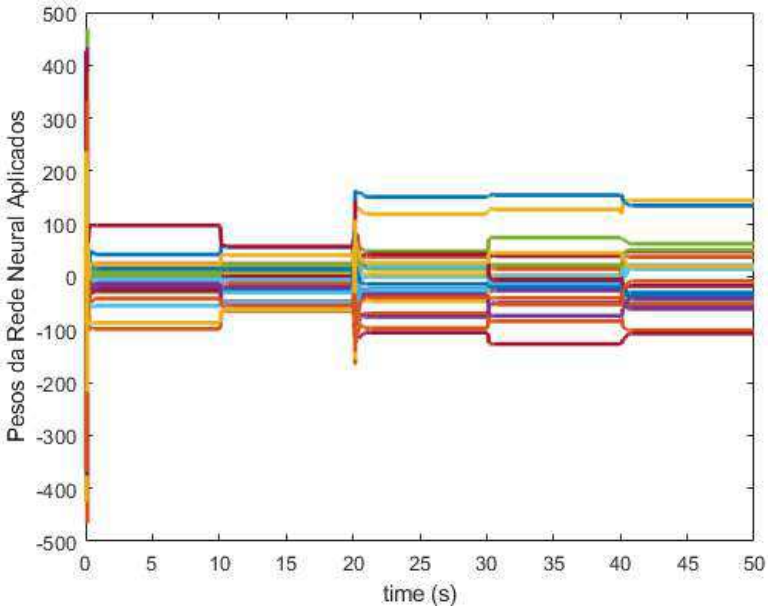
Fonte: Autor

Figura 6.26: Função de Custo vs Função de Custo Estimada ($\hat{Q}$) pelo Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal com Múltiplos Degraus



Fonte: Autor

Figura 6.27: Atualização dos Pesos da Rede do Ator pelo Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal com Múltiplos Degraus

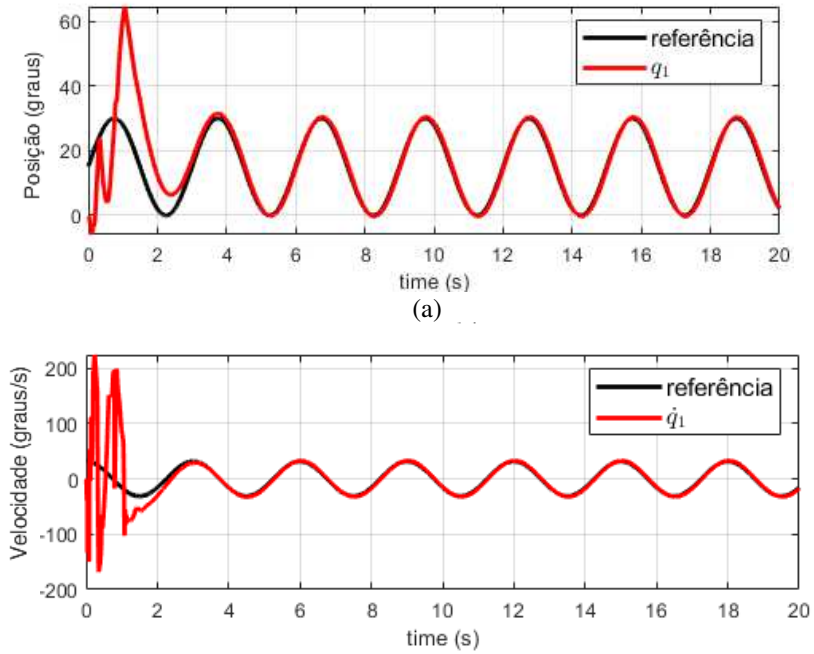


Fonte: Autor

No terceiro caso, um sinal senoidal foi adotado como referência. Os resultados de simulação são observados nas Figuras 6.28 a 6.32. Evidencia-se no comportamento do torque aplicado e na trajetória de atualização dos pesos da política de controle, o processo de aprendizado nos primeiros 2 s de simulação. Após este intervalo os pesos do Ator apresentam

comportamento de convergência, o sinal de controle torna-se menos irregular e os erros de rastreamento e o custo de controle são minimizados para valores admissíveis.

Figura 6.28: Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal Senoidal: a) Posição e b) Velocidade Simuladas da Junta do Ombro

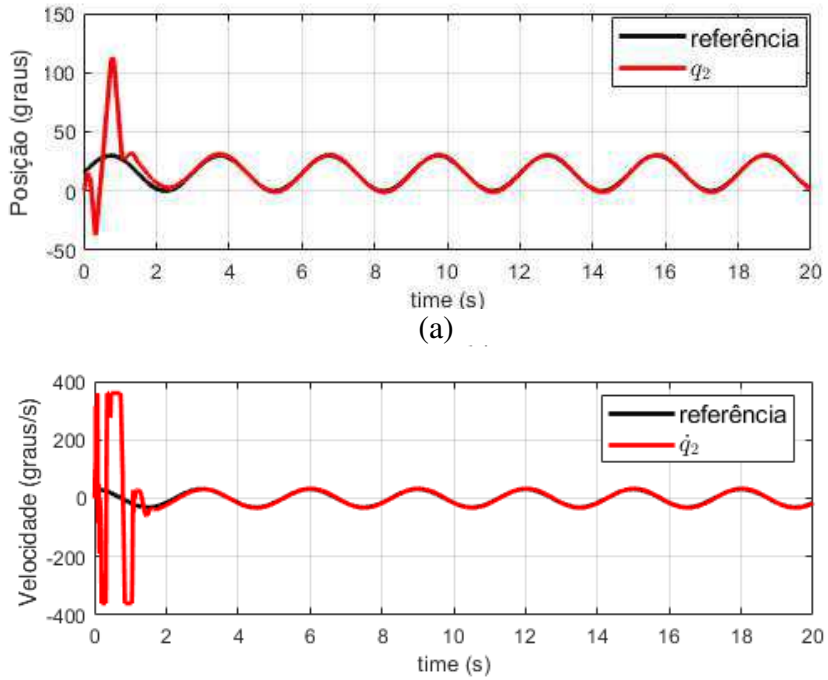


(a)

(b)

Fonte: Autor

Figura 6.29: Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal Senoidal: a) Posição e b) Velocidade Simuladas da Junta do Cotovelo

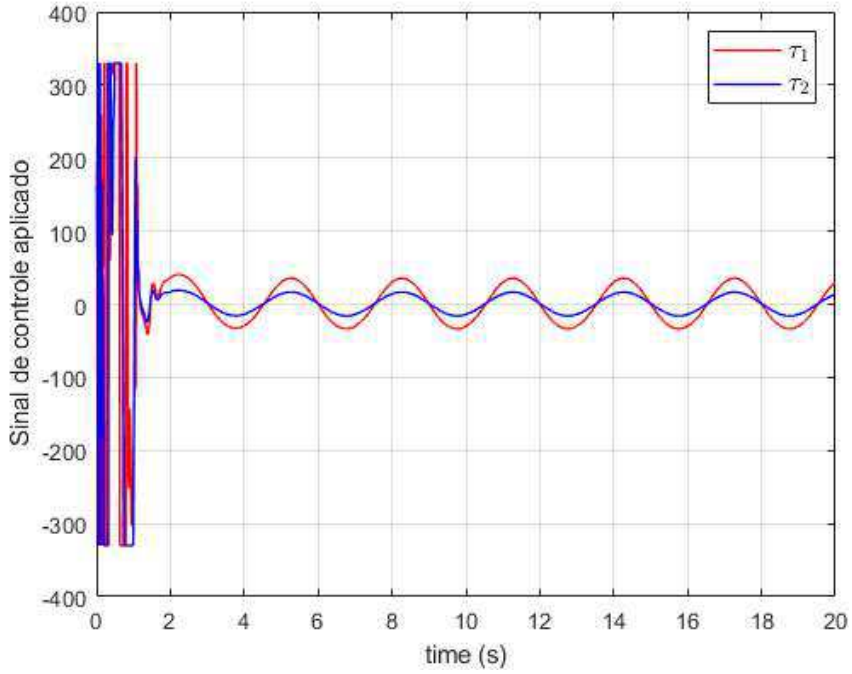


(a)

(b)

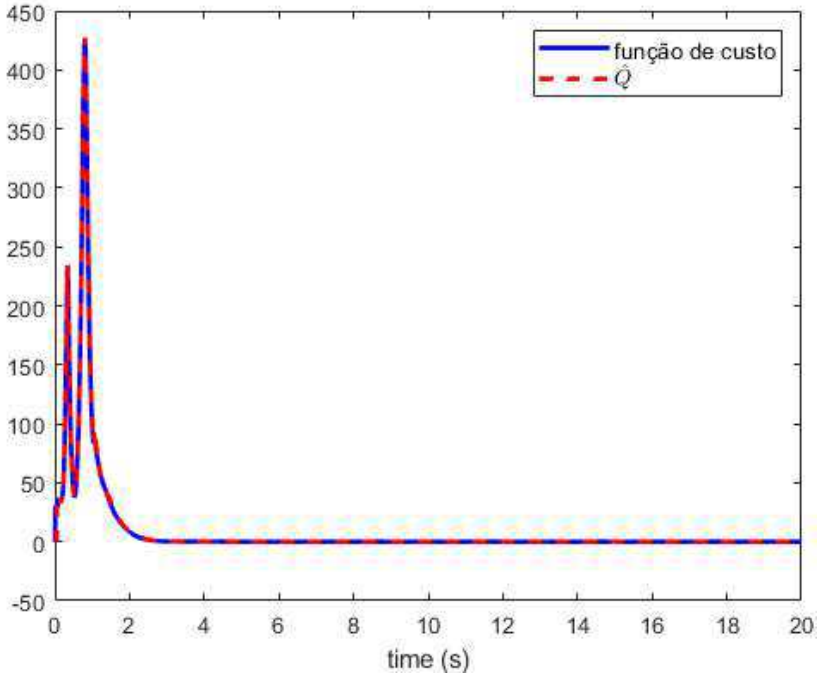
Fonte: Autor

Figura 6.30: Esforço de Controle pelo Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal Senoidal



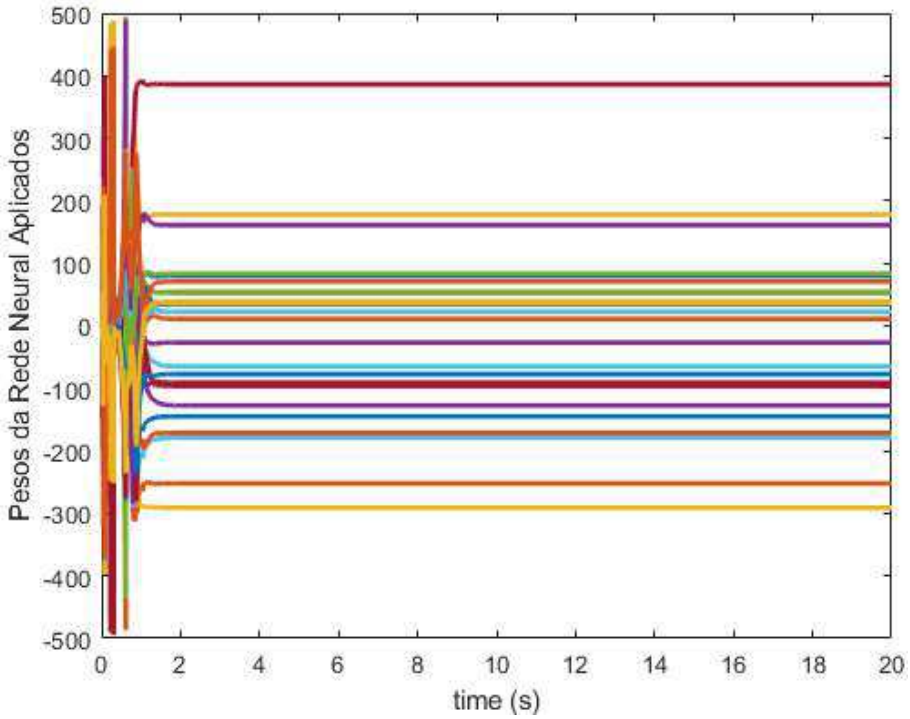
Fonte: Autor

Figura 6.31: Função de Custo vs Função de Custo Estimada ($\hat{Q}$) pelo Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal Senoidal



Fonte: Autor

Figura 6.32: Atualização dos Pesos da Rede do Ator pelo Esquema RL com Melhoria de Política Aproximada - Rastreamento de um Sinal Senoidal

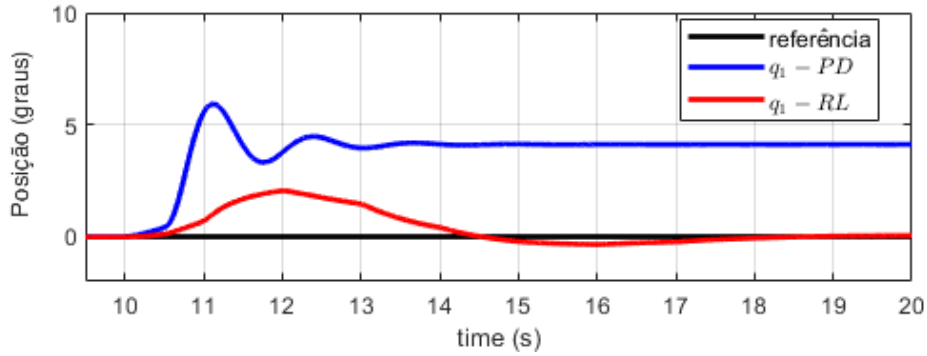


Fonte: Autor

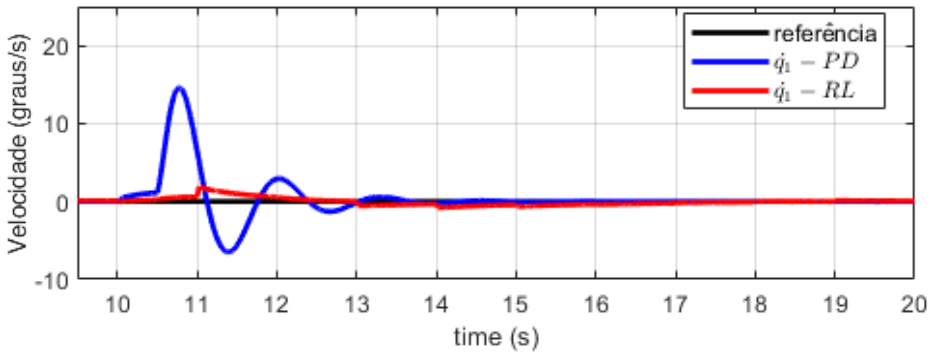
Na sequência, experimentos com variação nos parâmetros do manipulador foram realizados. Para tanto, o mesmo sinal de referência e os mesmos valores da carga utilizados nos testes com o esquema de melhoria de política exata foram aplicados nas experiências discutidas a seguir. Além disso, nas Figuras 6.33 a 6.35, onde são apresentados os resultados desses experimentos, o sinal em azul demonstra o desempenho isolado do controlador PD (aplicado para fornecer robustez ao esquema em avaliação), enquanto em vermelho é exibido o resultado da atuação do controle baseado em RL. Os parâmetros do controlador foram os mesmos aplicados nas tarefas de regulação e rastreamento, exceto para: $Q_C = diag(700, 700, 0{,}001, 0{,}001)$, $R = diag(0{,}01, 0{,}01)$ e $\gamma = 0{,}95$.

O comportamento observado nos testes foi semelhante para as diferentes cargas aplicadas, onde constata-se o desvio dos estados em relação a referência quando a perturbação é adicionada, com este sendo compensado com sucesso antes do fim estabelecido para simulação. Além disso, é evidente que a capacidade de rastreamento do esquema proposto é superior ao controlador PD, apresentando nos instantes finais da simulação erros próximos a zero.

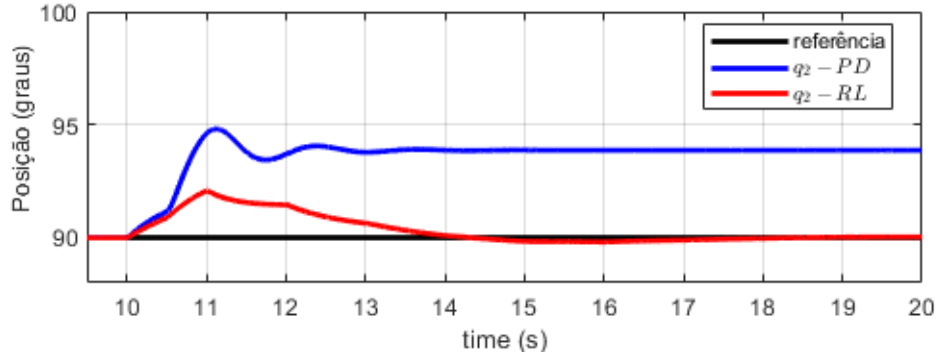
Figura 6.33: Esquema RL com Melhoria de Política Aproximada - Carga de Trabalho de 5 kg. a) Posição e b) Velocidade Simuladas da Junta do Ombro, e c) Posição e d) Velocidade Simuladas da Junta do Cotovelo



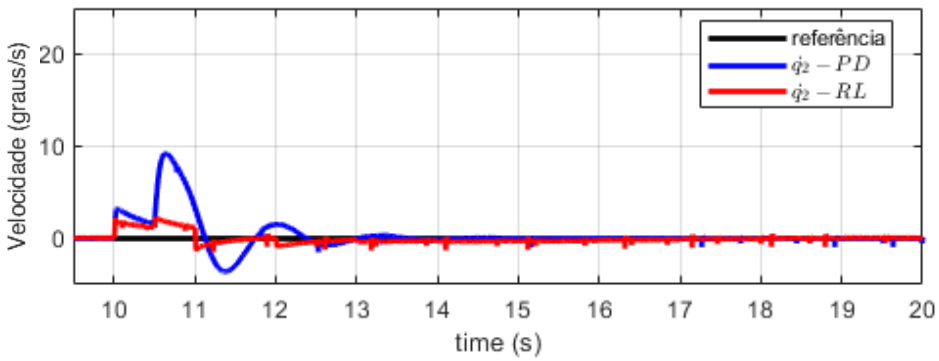
(a)



(b)



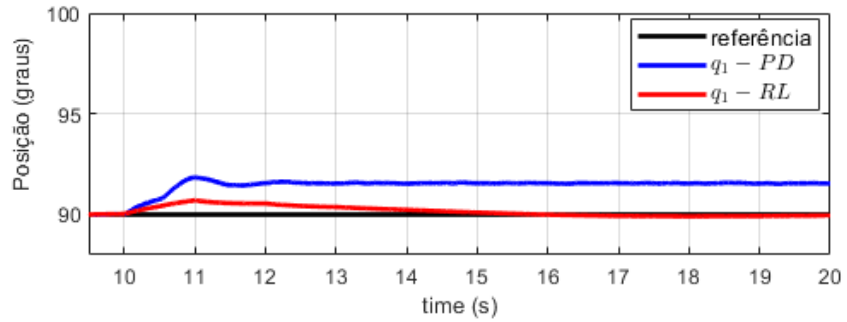
(c)



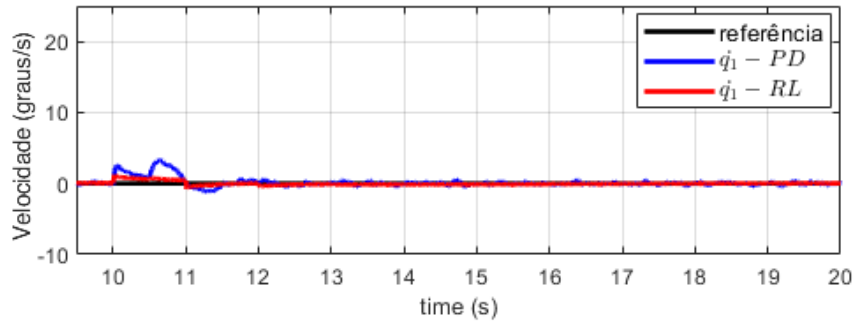
(d)

Fonte: Autor

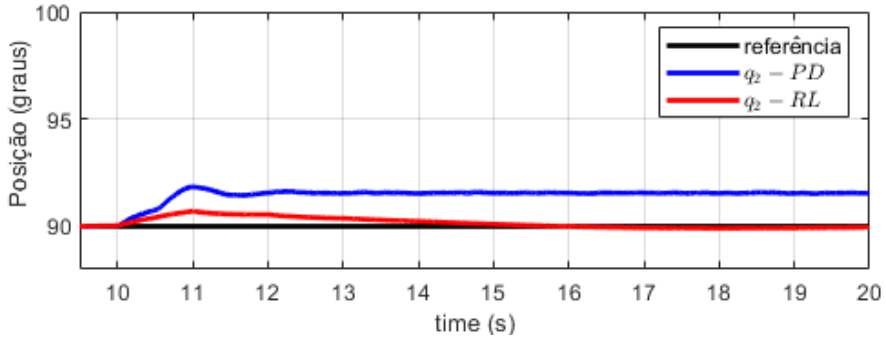
Figura 6.34: Esquema RL com Melhoria de Política Aproximada - Carga de Trabalho de 2 kg. a) Posição e b) Velocidade Simuladas da Junta do Ombro, e c) Posição e d) Velocidade Simuladas da Junta do Cotovelo



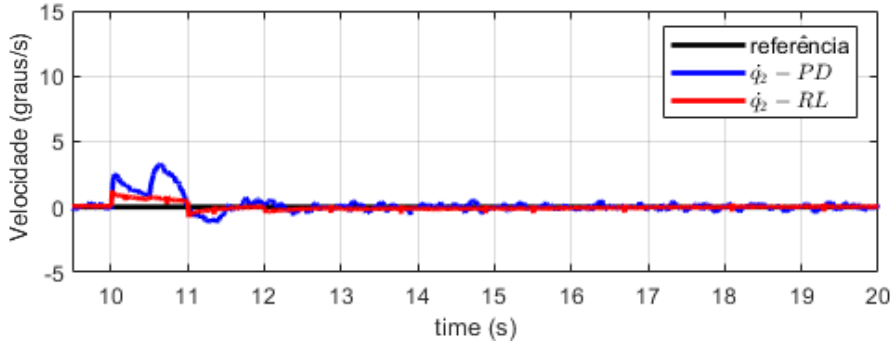
(a)



(b)



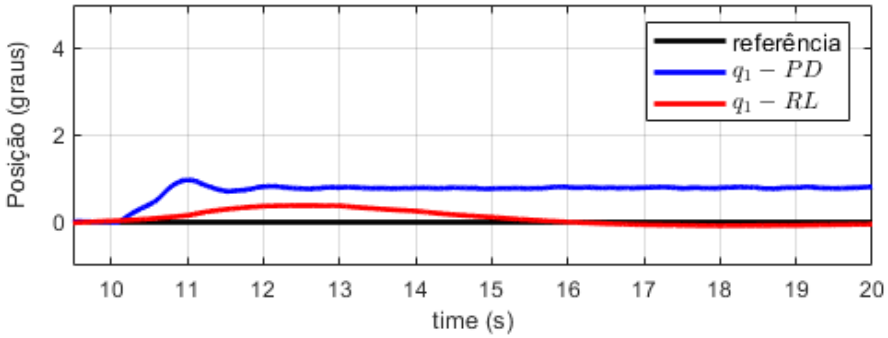
(c)



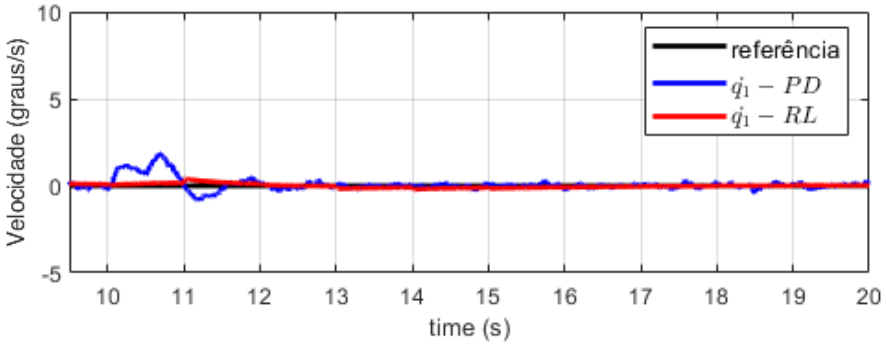
(d)

Fonte: Autor

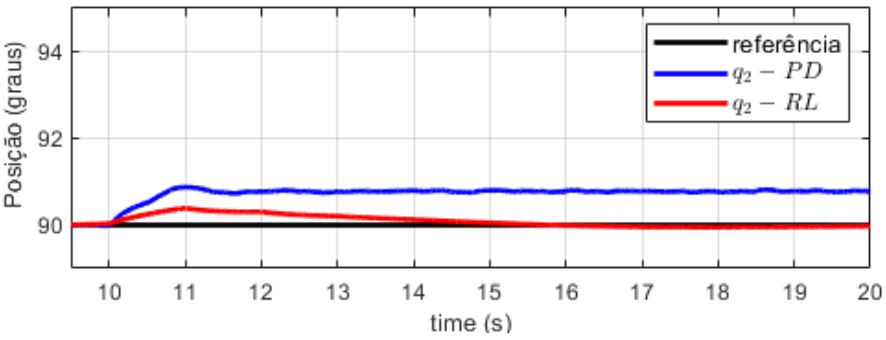
Figura 6.35: Esquema RL com Melhoria de Política Aproximada - Carga de Trabalho de 1 kg. a) Posição e b) Velocidade Simuladas da Junta do Ombro, e c) Posição e d) Velocidade Simuladas da Junta do Cotovelo



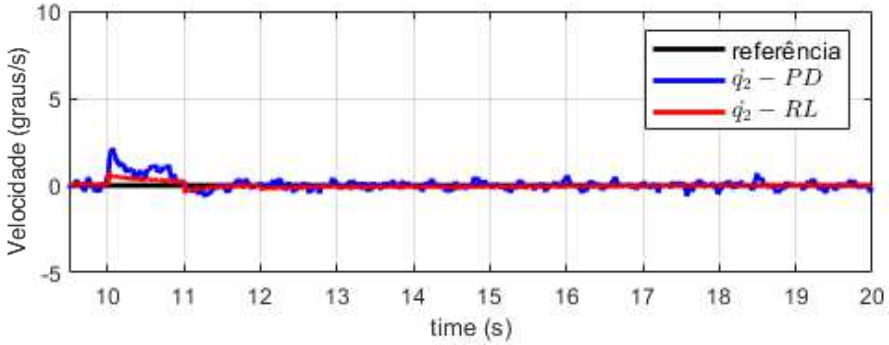
(a)



(b)



(c)



(d)

Fonte Autor

7 CONSIDERAÇÕES FINAIS

Neste trabalho, apresentou-se uma metodologia e algoritmos de aprendizado por reforço para o projeto de um controlador ótimo de um sistema manipulador robótico sem a necessidade do conhecimento do modelo dinâmico da planta. Dois esquemas de controle baseados em RL foram propostos para o problema de regulação e rastreamento ótimos mediante variações na carga de trabalho. Ambos os esquemas aproximam a função Q treinando uma rede neural usando apenas as medidas reais do sistema via o estimador RLS, mas diferem quanto a representação e atualização da política de controle. Nas duas abordagens, uma rede neural polinomial foi escolhida para aproximar a função valor de estado-ação, mostrando-se adequada para aprender as não linearidades do manipulador. Já especificamente em relação ao primeiro esquema, a etapa de atualização da política ocorre através de uma expressão que é obtida minimizando a função Q. Este procedimento só acontece ao fim de um número fixo de iterações (ciclo de aprendizado) com o objetivo de fornecer robustez ao esquema, mantendo, assim, a política constante durante este intervalo. Com respeito a segunda abordagem, uma rede neural foi implementada para aproximar a política de controle, onde os pesos dessa rede são treinados de modo a minimizar a função Q e gerar ações de controle ótimas.

Experimentos computacionais com a metodologia apresentada foram realizados utilizando o modelo do robô UR10 no simulador V-REP. As simulações incluíram a realização da tarefa de regulação, seguimento de trajetória dos sinais senoidal e de múltiplos degraus, e testes com variação na carga de trabalho. É importante apontar que o esquema com melhoria de política exata apresentou frequentes instabilidades durante o desenvolvimento dos experimentos, exigindo maior esforço na etapa de ajustes dos parâmetros do algoritmo (Q_c, R, γ, α etc.), de modo a determinar os valores que satisfizessem as exigências de controle ao mesmo tempo que fornecesse maior estabilidade durante a simulação. Por outro lado, este primeiro esquema demonstrou uma capacidade de compensação a variação na carga de trabalho (2 kg e 1 kg) maior que a abordagem com política aproximada, minimizando com sucesso o sobressinal já nos instantes iniciais após a modificação do parâmetro. Por fim, em vista dos resultados obtidos nas simulações, controladores implementáveis em manipuladores robóticos foram obtidos, fornecendo estabilidade para as variáveis de estado durante todo o tempo de simulação, capacidade de rastreamento nos diferentes sinais de referência e compensação nas variações na carga de trabalho, mesmo sem o conhecimento explícito do sistema.

7.1 TRABALHOS FUTUROS

Como trabalhos futuros a serem desenvolvidos a partir do estágio atual atingido neste trabalho de dissertação, podem-se citar:

- a realização de um estudo sobre a metodologia de implementação do controlador discutido neste trabalho em um protótipo real; e
- a implementação dos esquemas de aprendizado em um manipulador planar com juntas rotativas e grau de liberdade maior que dois.

REFERÊNCIAS

ABBAS, Z. **Motion Control of Robotic Arm Manipulator Using PID Sliding Mode Technique.** Tese de Doutorado. Universidade de Ciência de Tecnologia de Islamabad. 2018.

ABU-KHALAF, M. & LEWIS, F. **Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach**. Automatica. vol. 41, pp. 779-791. 2005.

AHMAD, A. e AL MAASHRI, A. Design and Analysis of Newly Innovated Set of Binary code-words of Unitary Hamming Distance. IEEE international conference on communication, computer and power. 2007.

AJWAD, S.A., IQBAL, J., ULLAH, M.I. e MEHMOOD, A. **A systematic review of current and emergent manipulator control approaches**. Frontiers of Mechanical Engineering, vol. 10, pp. 198–210, 2015.

AL-DABOONI, S. e WUNSCH, D. **The Boundedness Conditions for Model-Free HDP(λ)**. IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 7, pp. 1928-1942, July 2019.

AL-OLIMAT, K. S. e GHANDAKLY, A. A. **Multiple model reference adaptive control algorithm using on-line fuzzy logic adjustment and its application to robotic manipulators.** Conference Record of the 2002 IEEE Industry Applications Conference. 37th IAS Annual Meeting, Pittsburgh, PA, USA, vol.2, pp. 1463-1466, 2002.

ALQAUDI, B., MODARES, H., RANATUNGA, I. *et al*. **Model reference adaptive impedance control for physical human-robot interaction.** Control Theory Technol. vol. 14, p. 68–82, 2016.

AL-TAMIMI, A., LEWIS, F. L., e ABU-KHALAF, M. **Model-free Q-learning designs for discrete-time zero-sum games with application to H-infinity control.** 2007 European Control Conference (ECC), pp. 1668-1675, 2007

AWELEWA, A., MBANISI, K., MAJEKODUNMI, S., ODIGWE, I., AGBETUYI, A. e SAMUEL, I. Development of a Prototype Robot Manipulator for Industrial Pick-and-Place Operations. International Journal of Mechanical & Mechatronics Engineering, vol. 13, pp 20-28, 2013.

BEN-ARI, M. e MONDADA, F. **Robots and Their Applications. In: Elements of Robotics.** Springer, Cham. 2018.

BILLARD, A. e KRAGIC, D. **Trends and challenges in robot manipulation**. Science, vol. 364, no. 6446, 2019.

BOBROW, J. E. e McDONELL, W. **Modeling, Identification, and Control of a Pneumatically Actuated, Force Controllable Robot.** IEEE Transactions on Robotics and Automation, VOL. 14, NO. 5, OCTOBER 1998.

BOUKENS, M. e BOUKABOU, A. **Neuro-optimal controller for robot manipulators**. International Conference on Artificial Neural Networks. Springer, Berlin, Heidelberg, p. 503-510, 2013.

BUŞONIU, L., BABUŜCA, R., DE SCHUTTER, B. E. e ERNST, D. **Reinforcement Learning and Dyamic Programming Using Functions Approximator**s. CRC Press. Boca Raton, Florida. 2011.

CAO, S. SUN, L., JIANG, J. e ZUO, Z. **Reinforcement Learning-Based Fixed-Time Trajectory Tracking Control for Uncertain Robotic Manipulators With Input Saturation**. IEEE Transactions on Neural Networks and Learning Systems, pp. 1-12, 2021.

CARRARA, V. **Introdução à Robótica Industrial.** 2015. Disponível em: http://urlib.net/8JMKD3MGP3W34P/3K5JPL8

CHEN, W., FU, Z. J. e CHEN, C. S. **Recent Advances in Radial Basis Function Collocation Methods**. 1ª Edição. Springer, 2014.

CRAIG, JOHN J. **Introdução à Robótica**, 3a. Edição, Pearson, 2013.

DUBOWSKY, S. e DESFORGES, D. T. **The Application of Model-Referenced Adaptive Control to Robotic Manipulators.** ASME. J. Dyn. Sys., Meas., Control. SeptembeR; 101(3): 193–200, 1979.

DUDEK, G. e JENKIN, M. **Computational principles of mobile robotics**. Cambridge university press. 2010.

FATEH, S. e FATEH, M. M. **Adaptive Fuzzy Control of Robot Manipulators with Asymptotic Tracking Performance**. Journal Control Autom Electrical System, vol. 31, pp. 52–61, 2020.

GONÇALVES, D. V. **Controle Adaptativo de processo de nível utilizando aprendizado por reforço ator-crítico.** Monografia. Universidade de Brasília. 2016.

GUALTIERI, M e PLATT, R. **Robotic Pick-and-Place With Uncertain Object Instance Segmentation and Shape Completion**. IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1753-1760, April 2021.

GUO, W., LIU, F., SI, J., e MEI, S. **Online adaptation of controller parameters based on approximate dynamic programming**. 2014 International Joint Conference on Neural Networks (IJCNN), pp. 256-262, 2014.

GUO, X., YAN, W., e CUI, R. **Reinforcement Learning-Based Nearly Optimal Control for Constrained-Input Partially Unknown Systems Using Differentiator**. IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 11, pp. 4713-4725, Nov. 2020.

HAYKIN, S., **Redes Neurais Princípios e prática**, editora Bookman 2ª ed., Porto Alegre - RS, 1999.

HOWARD, R. A. **Dynamic programming and Markov processes**. John Wiley and Sons, Inc. New York, 1960.

HU, Q., XU, L. e ZHANG, A. **Adaptive backstepping trajectory tracking control of robot manipulator**. Journal of the Franklin Institute. Vol. 349, no. 3, p. 1087-1105, 2012.

HU, Y. e SI, B. **A reinforcement learning neural network for robotic manipulator control**. Neural Computation, vol. 30, no., pp. 1983–2004, 2018.

HU, Y., CUI L., e CHAI, S. **Optimal Tracking Control for Robotic Manipulator using Actor-Critic Network**. 2021 40th Chinese Control Conference (CCC), pp. 1556-1561, 2021.

HUANG, Z., ZHENG H., CHEN W., ZHANG, Q., WU, Q., TAN, Q. e YANG, Z. **Approximate dynamic programming solution for the optimal nitrogen oxides/particulate matter trade-off control of a WAPS engine**. Advances in Mechanical Engineering, vol. 10, no. 2, 2018.

KAI, C. e HUANG, A. Adaptive LQ control of robot manipulators. 2014 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, pp. 770-774, 2014.

KAMBOJ, A., PRAKASH, R., MOHANTA, J. K. E L. BEHERA. **Discrete-Time Lyapunov based Kinematic Control of Robot Manipulator using Actor-Critic Framework**. 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, United Kingdom, 2020, pp. 1-7, 2020.

KHAN, S. G., TUFAIL, M., SHAH, S. H. e ULLAH, I. **Reinforcement learning based compliance control of a robotic walk assist device**. Advanced Robotics, vol. 33, no. 24, pp. 1281-1292, 2019.

KHAN, S.; HERRMANN, G.; LEWIS, F. L.; PIPE, T.; e MELHUISH, C. **Reinforcement Learning and Optimal Adaptive Control: An Overview and Implementation Examples**. 2012). Annual Reviews in Control. vol. 36, pp. 42-59, 2012.

KIRK, D. E. **Optimal Control Theory: An Introduction.** Prentice-Hall, 2004.

KOBER, J. e BAGNELL, J. e PETERS, J. **Reinforcement Learning in Robotics: A Survey**. The International Journal of Robotics Research, vol 32, p.: 1238-1274, 2013.

KONSTANTOPOULOS, G. C. e BALDIVIESO-MONASTERIOS, P.R. **State-limiting PID controller for a class of nonlinear systems with constant uncertainties**. Int J Robust Nonlinear Control, vol. 30, pp. 1770– 1787, 2020.

KUAN, C., e YOUNG, K. **Reinforcement learning and robust control for robot compliance tasks**. Journal of Intelligent and Robotic Systems, 23, 165–182, 1998.

KUCUK, S. e GUNGOR, B. D. **Inverse kinematics solution of a new hybrid robot manipulator proposed for medical purposes**. Medical Technologies National Congress (TIPTEKNO), Antalya, 2016, pp. 1- 4.

KUKKER, A. e SHARMA, R. **Stochastic Genetic Algorithm-Assisted Fuzzy Q-Learning for Robotic Manipulators**. Arabic Journal for Science and Engineering, vol. 46, pp. 9527–9539, 2021.

LEWIS, F. L. e VRABIE, D. **Reinforcement learning and adaptive dynamic programming for feedback control.** IEEE Circuits and Systems Magazine, vol. 9, no. 3, pp. 32-50, 2009.

LEWIS, F. L., VRABIE, D. e VAMVOUDAKIS, K. **Reinforcement Learning and Feedback Control: Using Natural Decision Methods to Design Optimal Adaptive Controllers**. IEEE Control Systems Magazine, vol. 32, no. 6, pp. 76-105, Dec. 2012a.

LEWIS, F., VRABIE, D., e SYRMOS, V. L. **Optimal Control**. 3ª Edição. John Wiley & Sons, Inc. New Jersey. 2012b.

LEWIS, F., YESILDIRAK, A., e JAGANNATHAN, S. **Neural Network Control of Robot Manipulators and Nonlinear Systems**. 1998.

LEWIS, F.L., DAWSON, D.M., e ABDALLAH, C. **Robot Manipulator Control: Theory and Practice**. Segunda Edição. 2004.

LI, S. ZHANG, Y. e JIN, L. **Kinematic Control of Redundant Manipulators Using Neural Networks**. IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 10, pp. 2243-2254, Oct. 2016.

LIU, D., WANG, D., e LI, H. **Decentralized Stabilization for a Class of Continuous-Time Nonlinear Interconnected Systems Using Online Learning Optimal Control Approach**. IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 2, pp. 418-428, Feb. 2014.

LUY, N.T., THANH, N.T. e TRI, H.M. **Reinforcement learning-based intelligent tracking control for wheeled mobile robot. Transactions of the Institute of Measurement and Control**, vol 36, no 7, p.: 868-877, 2014.

MA, B., DONG, B., ZHOU, F. e LI, Y. **Adaptive Dynamic Programming-Based Fault-Tolerant Position-Force Control of Constrained Reconfigurable Manipulators**. IEEE Access, vol. 8, pp. 183286-183299, 2020.

MALIOTIS, G. A. **Hybrid Model Reference Adaptive Control/Computed Torque Control Scheme for Robotic Manipulators. Proceedings of the Institution of Mechanical Engineers,** Part I: Journal of Systems and Control Engineering, vol 205, no. 3, p. 215–21, 1991.

MELKOU, L. e HAMERLAIN, M. **High Order Homogeneous Sliding Mode Control for a Robot Arm with Pneumatic Artificial Muscles**. IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, pp. 394-399, 2019.

MENON, A., PRAKASH, R. e BEHERA, L. **Adaptive Critic Based Optimal Kinematic Control for a Robot Manipulator**. 2019 International Conference on Robotics and Automation (ICRA), pp. 1316-1322, 2019.

MONTOYA-CHÁIREZ, J. e MORENO-VALENZUELA, J. **An Input-Output Feedback Linearization Approach to the Motion Control of Flexible Joint Manipulators**. 2021 60th IEEE Conference on Decision and Control (CDC), pp. 1426-1431, 2021

MOOSAVI, S. K. R., ZAFAR, M. H. e. SANFILIPPO, F. **Forward Kinematic Modelling with Radial Basis Function Neural Network Tuned with a Novel Meta-Heuristic Algorithm for Robotic Manipulator**s. Robotics, vol. 11, no. 2, pp 43, 2022.

MORALES, M. **Grokking Deep Reinforcement Learning**. Simon e Schuster.2020.

MOTA, I. C. **Aprendizado por reforço utilizando Q-learning e redes neurais artificiais em jogos eletrônicos**. Monografia. Universidade de Brasília. 2018.

NAIDU, D. S. **Optimal Control Systems**, CRC Press, 2002.

NAM, D. P., NGUYEN, H. Q., TRAN, P. N., e YEN, T. T. H. **Adaptive dynamic programing based optimal control for a robot manipulator**. International Journal of Power Electronics and Drive Systems, vol. 11, no. 3, pp. 1123, 2020.

NAWROCKA, A., NAWROCKI, M. e KOT, A. **Neural Network Control for Robot Manipulator**. 2019 20th International Carpathian Control Conference (ICCC), pp. 1-4, 2019.

NIELSEN, M. A. **Neural networks and deep learning**. Determination Press, 2015.

PANE, Y. P., NAGESHRAO, S. P. e BABUŠKA, R. **Actor-critic reinforcement learning for tracking control in robotics**. 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, pp. 5819-5826, 2016.

PANE, Y. P., NAGESHRAO, S. P., KOBER, J e BABUŠKA, R. **Reinforcement learning based compensation methods for robot manipulators**. Engineering Applications of Artificial Intelligence, vol 78, pp 236-247, 2019.

PAUL, S., ARUNACHALAM, A., KHODADAD, D.e RUBANENKO, O. ***Fuzzy* Tuned PID Controller for Vibration Control of Agricultural** Manipulator. 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), pp. 1-5, 2020.

PAZOS, F. **Automação de Sistemas e Robótica**. Editora Axcel Books do Brasil. Rio de Janeiro. 2002.

PETERS, VIJAYAKUMAR, J., S., e SCHAAL, S. **Reinforcement learning for humanoid robotics**. In Third IEEE-RAS international conference on humanoid robots. Karlsruhe, Germany, 2003.

PLUŠKOSKI, A., CIGANOVIĆ, I. e JOVANOVIĆ, M. D. **Benefits of Residual Networks in Reinforcement Learning using V-Rep Simulator**. 6th International Conference IcETRAN. 2019.

PRADHAN, S. K. e SUBUDHI, B. **Real-Time Adaptive Control of a Flexible Manipulator Using Reinforcement Learning**. IEEE Transactions on Automation Science and Engineering, vol. 9, no. 2, pp. 237-249, April 2012.

QUIGLEY, M., GERKEY, B., e SMART, W. D. **Programming Robots with ROS: a practical introduction to the Robot Operating System**. O'Reilly Media, Inc. 2015.

RASTOGI, D. **Deep Reinforcement Learning for Bipedal Robots.** Dissertação de mestrado. Universidade de Tecnologia de Delft. 2017.

RIBEIRO, M. I. **Sensores em Robótica**, Enciclopédia Nova Activa Multimédia, Volume de Tecnologias, pags. 228-229. Portugal, 2004.

ROHMER, E., SINGH, S. P. N., e FREESE, M. **CoppeliaSim (formerly V-REP): a Versatile and Scalable** Robot Simulation Framework. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2013.

ROMANO, V. F. **Robótica industrial: Aplicação na indústria de manufatura e de processos**. São Paulo: Edgard Blücher, ISBN 8521203152, 2002

RUBIO, J. D. J., CRUZ, P., PARAMO, L. A., MEDA, J. A., MUJICA, D. e ORTIGOZA, R. S. **PID Anti-Vibration Control of a Robotic Arm**. IEEE Latin America Transactions, vol. 14, no. 7, pp. 3144-3150, Julho, 2016

SHAH, H., e GOPAL, M. **Reinforcement learning control of robot manipulators in uncertain environments**. In IEEE international conference on industrial technology, (ICIT 2009) (pp. 1–6), 2009.

SHAMSHIRI, R. R., HAMEED, I. A., WELTZIEN, C. e KARKEE, M. **Robotic Harvesting of Fruiting Vegetables: A Simulation Approach in V-REP, ROS and MATLAB**. In (Ed.), Automation in Agriculture - Securing Food Supplies for Future Generations. IntechOpen.2018.

SILVA, L. G. O. **Controle de um Manipulador Robótico Leve de 4-DOF com Software Baseado em ROS e QT**. Trabalho de monografia. Universidade Federal do Rio de Janeiro – Escola Politécnica. 2017.

SPONG, M.W. e VIDYASAGAR, M. **Robot Dynamics and Control**. John Wiley & Sons, 2ª Edição. 2004.

SUN, N., LIANG, D., WU, Y., CHEN, Y., QIN, Y. e FANG, Y.**Adaptive Control for Pneumatic Artificial Muscle Systems With Parametric Uncertainties and Unidirectional Input Constraints**. IEEE Transactions on Industrial Informatics, vol. 16, no. 2, pp. 969-979, Fev. 2020.

SUTTON, R. S.; BARTO, A. G. e WILLIAMS, R. J. **Reinforcement Learning is Direct Adaptive Optimal Control**. American Control Conference, Boston, MA, 1991.Universidade Federal do Rio Grande do Norte. Natal, p. 95. 2005.

VAMVOUDAKIS, K. G.; VRABIE, D.; e LEWIS, F. L. **Online adaptive learning of optimal control solutions using integral reinforcement learning.** 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2011, pp. 250-257.

VIEIRA, F. C. **Controle Dinâmico de Robôs Móveis com Acionamento Diferencial**.

VRABIE, D., VAMVOUDAKIS, K. G., e LEWIS, F. L. **Optimal adaptive control and differential games by reinforcement learning principles**. (Vol. 2). IET. 2013.

WATKINS, C. J. C. H. **Learning from Delayed Rewards**. Tese de Doutorado. Universidade de Cambridge. Cambridge, Inglaterra. (1989).

WITTEN, I. H. **An Adaptive Optimal Controller for Discrete-Time Makov Environments.** Information and Control, vol. 34, pp. 286-295, 1977.

WU, L., YAN, Q. e CAI, J. **Neural Network-Based Adaptive Learning Control for Robot Manipulators With Arbitrary Initial Errors**. IEEE Access, vol. 7, pp. 180194-180204, 2019.

XU, J., e QIAO, L. **Robust Adaptive PID Control of Robot Manipulator with Bounded Disturbances**. Mathematical Problems in Engineering. p. 1-13. 2013.

XU, X., HE, H., e HU, D. **Efficient Reinforcement Learning Using Recursive Least-Squares Methods.** Journal of Artificial Intelligence Research. 16. 259-292. 10.1613. 2002.

YANG, J., WANG, Y., WANG, T., e YU, X. **Optimal Tracking Control For A Two-link Robotic Manipulator Via Adaptive Dynamic Programming**. 2020 Chinese Automation Congress (CAC), pp. 2687-2692, 2020.

YILMAZ, B. M., TATLICIOGLU, E., SAVRAN, A. e ALCI, M. **Self-Adjusting Fuzzy Logic Based Control of Robot Manipulators in Task Space**. IEEE Transactions on Industrial Electronics, vol. 69, no. 2, pp. 1620-1629, Feb. 2022.

ZHANG, D e WEI, B. **A review on model reference adaptive control of robotic manipulators**. Annual Reviews in Control, vol. 43, pp. 188-198, 2017.

ZHANG, D. e WEI, B. **Design, analysis and modelling of a hybrid controller for serial robotic manipulators.** Robotica, vol 35, p. 1888-1905, 2016.

ZHANG, K., ZHANG, H., XIAO, G., e SU, H. **Tracking control optimization scheme of continuous-time nonlinear system via online single network adaptive critic design method.** Neurocomputing. 251, C, 127–135, 2017.

ZHU, X., MA, B., DONG, B. e LI, Y. **Adaptive Dynamic Programming-Based Sliding Mode Optimal Position-Force Control for Reconfigurable Manipulators with Uncertain Disturbance**. Chinese Control and Decision Conference, 2020, p. 421-427.