MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21d/2024/02.22.13.00-TDI

ALGORITMO BASEADO NO APRENDIZADO POR REFORÇO PARA O CONTROLE DO APONTAMENTO DE SATÉLITES UTILIZANDO REDES NEURAIS

Gabriel Goes Aragão Santana

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle, orientada pelo Dr. Ronan Arraes Jardim Chagas, aprovada em 20 de fevereiro de 2024.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34T/4AQ4DCB>

INPE
São José dos Campos
2024

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Coordenação de Ensino, Pesquisa e Extensão (COEPE)
Divisão de Biblioteca (DIBIB)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21d/2024/02.22.13.00-TDI

ALGORITMO BASEADO NO APRENDIZADO POR REFORÇO PARA O CONTROLE DO APONTAMENTO DE SATÉLITES UTILIZANDO REDES NEURAIS

Gabriel Goes Aragão Santana

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle, orientada pelo Dr. Ronan Arraes Jardim Chagas, aprovada em 20 de fevereiro de 2024.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34T/4AQ4DCB>

INPE
São José dos Campos
2024

Dados Internacionais de Catalogação na Publicação (CIP)

Santana, Gabriel Goes Aragão.

Sa59a Algoritmo baseado no aprendizado por reforço para o controle do apontamento de satélites utilizando redes neurais / Gabriel Goes Aragão Santana. – São José dos Campos : INPE, 2024.

xxiv + 157 p. ; (sid.inpe.br/mtc-m21d/2024/02.22.13.00-TDI)

Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2024.

Orientador : Dr. Ronan Arraes Jardim Chagas.

1. Controle de atitude. 2. Controle inteligente. 3. Aprendizado por reforço. 4. Redes neurais. 5. Linguagem Julia. I.Título.

CDU 629.7.062.2:004.822

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

DEFESA FINAL DE DISSERTAÇÃO DE GABRIEL GOES ARAGÃO SANTANA REG. 883804/2022 , BANCA Nº015/2024

No dia 20 de fevereiro de 2024, às 14h00 em teleconferência, o(a) aluno(a) mencionado(a) acima defendeu seu trabalho final (apresentação oral seguida de arguição) perante uma Banca Examinadora, cujos membros estão listados abaixo. O(A) aluno(a) foi **APROVADO(A)** pela Banca Examinadora, por unanimidade, em cumprimento ao requisito exigido para obtenção do Título de Mestre em Engenharia e Tecnologia Espaciais / Mecânica Espacial e Controle, com a exigência de que o trabalho final a ser publicado deverá incorporar as correções sugeridas pela Banca Examinadora, com revisão pelo(s) orientador(es).

Título: "ALGORITMO BASEADO NO APRENDIZADO POR REFORÇO PARA O CONTROLE DO APONTAMENTO DE SATÉLITES UTILIZANDO REDES NEURAIS."

Membros da banca:

Dr. Evandro Marconi Rocco – Presidente - Membro Interno – INPE

Dr. Ronan Arraes Jardim Chagas – Orientador/Membro Interno – INPE

Dra. Roberta Veloso Garcia – Membro Externo – USP

Documento assinado eletronicamente por **Evandro Marconi Rocco**, **Tecnologista**, em 04/03/2024, às 17:13 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Roberta Veloso Garcia (E)**, **Usuário Externo**, em 04/03/2024, às 18:01 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Ronan Arraes Jardim Chagas**, **Tecnologista**, em 05/03/2024, às 09:29 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.mcti.gov.br/verifica.html, informando o código verificador **11759447** e o código CRC **E0C31855**.

Referência: Processo nº 01340.001339/2024-11 SEI nº 11759447

"O tigre tem que correr
O pássaro tem que voar
Homem tem que sentar e se perguntar
Por quê, por quê, por quê?"

KURT VONNEGUT, *"Cama de Gato"*

AGRADECIMENTOS

Aos meus pais amados, pelo indispensável apoio e incentivo durante todo esse tempo. Sou muito grato por minha irmã querida pela fonte de inspiração. A minha querida vó Élea; ainda voaremos juntos pelo espaço infinito.

Agradeço ao Dr. Ronan Arraes pela sua disposição e empenho de me orientar, pela liberdade que me concedeu em desenvolver o trabalho e pelos muitos ensinamentos passados.

Aos professores da Pós-Graduação do INPE em Engenharia e Tecnologias Espaciais pelo esforço e coragem frente as grandes adversidades, em especial a Dr. Evandro Rocco, Dr. Antonio Bertachini e Dr. Valdemir Carrara.

Ao meu colega de sala André Simões, pelas muitas discussões produtivas e conselhos valiosos durante nosso tempo de convivência.

Aos estimados companheiros de curso Homero, Paulo e Shyenne pela constante ajuda e companheirismo.

Aos meus queridos amigos do nosso ilustre clube cinematográfico Limonada.

Às nobres mentes e almas que se dedicam à exploração do espaço, em especial às muitas dentro do INPE. Não há caminho fácil do céu às estrelas.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001

RESUMO

O presente trabalho aplica o paradigma do Aprendizado por Reforço (*Reinforcement Learning*, RL), um subcampo de *Machine Learning*, ao problema de controle de atitude de um satélite. Nessa formulação, um agente interage com um ambiente, modificando seu estado ao realizar ações e recebendo uma recompensa - um sinal escalar - de forma a punir ou recompensar suas decisões. Por um processo de tentativa e erro, o agente deve encontrar a forma ótima de agir. Em anos recentes, desenvolvimentos inspirados em novas ideias deram origem a uma variedade de algoritmos, capazes de serem aplicados em uma grande quantidade de ambientes desafiadores. Em sua essência, esses algoritmos utilizam-se de redes neurais artificias (RNAs) para representar suas funções de interesse. Dessa forma, surge a possibilidade de aplicar as técnicas do RL aos problemas de controle tradicional, como o controle de atitude. Dominar essas ferramentas traria ganhos práticos, à medida que permitiria um ajuste automático dos parâmetros do controlador, o controle em situações muito diferentes do cenário nominal e a possibilidade de realização de missões mais desafiadoras e que requeiram uma menor interferência humana. Três algoritmos modernos do RL foram selecionados: DDPG (*Deep Deterministic Policy Gradient*), TD3 (*Twin Delayed DDPG*) e SAC (*Soft Actor-Critic*). Tanto a implementação desses algoritmos quanto a dinâmica de atitude do satélite foram escritas na linguagem Julia. Um cenário em particular, no qual a matriz de momento do inércia do satélite é variável, também é simulado. De maneira a comparar a solução do RL, o controlador proporcional-derivativo (PD) do satélite Amazonia-1 serve como referência. Considerações práticas acerca da estrutura da rede, em termos de função de ativação, topologia e número de camadas são discutidas como forma de inserir conhecimento prévio e acelerar o aprendizado. Dentre os três algoritmos, o SAC mostra-se constantemente o mais estável, não apenas resolvendo o problema convencional como também sendo capaz de controlar de maneira adequada o problema de inércias variáveis. As redes obtidas são relativamente pequenas, o que indica que a implementação nos computadores de bordo é possível. Embora a questão fundamental da estabilidade do RL seja identificada como o maior problema existente para seu uso prático, os resultados indicam que a combinação de RL com ideias do controle convencional pode ser uma forma atraente para a resolução de problemas desafiadores na área espacial.

Palavras-chave: Controle de atitude. Controle inteligente. Aprendizado por reforço. Redes neurais. Linguagem Julia.

REINFORCEMENT LEARNING BASED ALGORITHM FOR THE CONTROL OF SATELLITE POINTING USING NEURAL NETWORKS

ABSTRACT

The present work applies the Reinforcement Learning (RL) paradigm, a subfield of Machine Learning, to the attitude control problem for a satellite. In this formulation, an agents interacts with an environment, changing its state by selecting actions and receiving a reward - a scalar sign - so as to punish or reward its decisions. By a trial-and-error approach, the agent should learn an optimum way to behave. Driven by novel ideas, recent years have witnessed major developments in the field, leading to a variety of algorithms capable of tackling numerous challenging environments. At its core, these algorithms employ artificial neural networks (ANNs) to represent their functions of interest. This way, a possibility arises of applying RL techniques to traditional control problems, which includes attitude control. Mastering this technique would be of practical importance, as it would allow the automatic tuning of controller parameters, control in situations far from the nominal scenario and it could enable more challenging missions to be carried out, requiring less human interference. Three modern RL algorithms were selected: DDPG (Deep Deterministic Policy Gradient), TD3 (Twin Delayed TD3) and SAC (Soft Actor-Critic). Their implementation as well as the satellite attitude dynamics were written in the Julia language. A particular scenario, in which the satellite's moment of inertia matrix is variable, is also simulated. The proportional-derivative (PD) control onboard the Amazonia-1 satellite is used as a reference, allowing a comparison with the RL solutions. Practical considerations concerning the desired network structure in terms of activation function, topology and number of hidden layers are discussed. These points are important as they help by providing previous knowledge to the agent and thus speed up the learning process. Among the three algorithms, SAC constantly proves itself to be the most stable, not only solving the conventional problem but also being able to adequately control the problem of variable inertia. The employed networks are relatively small, which indicates their implementation on real computers used in space missions is feasible. Even though the fundamental issue of stability is identified as the biggest hurdle to real applications, the results indicate that the combination of RL with conventional control ideas may be a promising approach to solve challenging problems in the space sector.

Keywords: Attitude control. Intelligent control. Reinforcement Learning. Neural Networks. Julia language.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

A3C	–	Asynchronous Advantage Actor-Critic
AI	–	Artificial Intelligence
DCM	–	Direct Cosine Matrix
DDPG	–	Deep Deterministic Policy Gradient
DPG	–	Deterministic Policy Gradient
DDQN	–	Double DQN
DQN	–	Deep Q-Network
INPE	–	Instituto Nacional de Pesquisas Espaciais
MDP	–	Processo de Decisão de Markov
ML	–	Machine Learning
POMDP	–	Processo de Decisão de Markov Parcialmente Observável
PPO	–	Proximal Policy Optimization
RL	–	Reinforcement Learning
RNA	–	Rede Neural Artificial
SAC	–	Soft Actor-Critc
TD3	–	Twin Delayed DDPG
TRPO	–	Trust Region Policy Optimization

LISTA DE SÍMBOLOS

γ	–	taxa de desconto
$\boldsymbol{\epsilon}$	–	parte vetorial do quatérnion
η	–	parte escalar do quatérnion
θ	–	ângulo genérico de rotação
π	–	política do agente
π^*	–	política ótima
π_θ	–	ator parametrizado por θ
ρ	–	densidade de estados
$\vec{\boldsymbol{\omega}}$	–	vetor velocidade angular
$\vec{\boldsymbol{a}}$	–	eixo de rotação na representação eixo-ângulo de Euler
$\mathcal{A}$	–	conjunto de todas as ações disponíveis ao agente
$\boldsymbol{A}^+$	–	matriz pseudoinversa
$\hat{\boldsymbol{e}}_{\boldsymbol{i}}$	–	versor no eixo i
$\mathbb{E}$	–	valor esperado
J	–	função custo
$\mathcal{H}$	–	entropia
$\boldsymbol{I}$	–	matriz identidade, momento de inércia
G_0	–	retorno ajustado obtido em um episódio
$\boldsymbol{k_d}$	–	ganhos da parte derivativa do controlador PD
$\boldsymbol{k_p}$	–	ganhos da parte proporcional do controlador PD
$\vec{\boldsymbol{L}}$	–	vetor momento angular
o	–	observação do ambiente
P	–	probabilidade de transição do ambiente
Q_π	–	valor de ação de π
Q_π^w	–	crítico parametrizado por w
r	–	recompensa
$\boldsymbol{R}$	–	matriz de rotação
s	–	estado atual do ambiente
s'	–	próximo estado do ambiente
$\mathcal{S}$	–	conjunto de todos os estados possíveis do ambiente
$\boldsymbol{q}$	–	quatérnion
$\bar{\boldsymbol{q}}$	–	quatérnion conjugado de $\bar{\boldsymbol{q}}$
$\vec{\boldsymbol{T}}$	–	vetor torque
$\boldsymbol{T_{rw}}$	–	vetor coluna com os torques aplicados às rodas de reação
$\vec{\boldsymbol{u}}$, $\vec{\boldsymbol{v}}$	–	vetor genérico
$\boldsymbol{u}$, $\boldsymbol{v}$	–	vetor coluna com coordenadas do vetor
V_π	–	valor de estado de π

SUMÁRIO

Pág.

1 INTRODUÇÃO

A inteligência artificial (AI, *artificial intelligence* em inglês), de maneira geral, diz respeito à inteligência exibida por agentes não naturais, isso é, criados pelo homem. O termo foi usado pela primeira vez em 1955, quando quatro pesquisadores de diferentes instituições propuseram a realização de um *workshop* sobre o tema (MCCARTHY et al., 2006). O evento aconteceu no ano seguinte, em 1956, no Dartmouth College, e é considerado o nascimento do campo de AI.

Em razão da multiplicidade de definições de inteligência, não é de se espantar que o termo AI também possua várias definições. Legg e Hutter (2007), por exemplo, compilaram uma lista de 70 definições de inteligência provenientes de 3 origens diferentes: coletivas (grupos e organizações), psicólogos e pesquisadores de AI. Analisando os pontos em comum, chegam a seguinte definição: "inteligência mede a habilidade de um agente em alcançar objetivos em uma vasta variedade de ambientes".

McCarthy (2007) definiu AI como a ciência e engenharia de fazer máquinas inteligentes, especialmente programas de computador.

Uma classificação que abrange a variedade de possibilidades foi exposta por Russell e Norvig (2010), que categorizam as definições existentes em 4 quadrantes de acordo com seus objetivos ao longo dos eixos racional/humano e pensar/agir. Dessa forma, uma abordagem centrada no aspecto humano tem como meta construir sistemas que pensem como humanos ou que ajam como humanos. Uma abordagem racionalista, por outro lado, pretende obter sistemas que pensem de maneira racional ou ajam de maneira racional.

Como pode ser visto, é difícil fornecer uma definição suficiente rigorosa, mas que ao mesmo tempo seja geral o bastante para ser aplicada ao grande número de possibilidades almejadas pelo campo de AI. Nesse trabalho, ao se falar de AI, entender-se-á a ideia de fazer máquinas que sejam mais semelhantes com humanos, imbuindo-as de poderosas capacidades geralmente associadas apenas ao homem, como raciocínio, abstração, criatividade e engenhosidade.

Se o objetivo do campo de AI é criar agente inteligentes, uma ideia que se torna atraente é a de aprendizado. As soluções de muitas tarefas que se deseja resolver são demasiado complexas para serem escritas explicitamente por um programador ou são mesmo inteiramente desconhecidas. Seria muito mais favorável inserir no agente uma capacidade de aprendizado. Conhecimento prévio ainda pode ser adicionado,

mas cabe ao agente aprender com suas experiências e encontrar soluções ótimas.

Essa é a ideia primordial por trás do campo de aprendizado de máquina (ML, *machine learning* em inglês), um subcampo de AI. O termo foi usado pela primeira vez por Samuel (1959), que desenvolveu um agente para aprender a jogar damas e realizou sua implementação em um computador IBM 704. O programa alcançou o desempenho de um jogador mediano em um período de 8 a 10 horas de execução. Curiosamente, algumas das ideias modernas do campo de aprendizado por reforço foram empregadas nesse trabalho, ainda que de maneira rudimentar, como o cálculo de funções de valor.

Mais formalmente, uma definição clara e concisa de ML foi dada por Mitchell (1997): "É dito que um programa de computador aprende com experiência E em relação a uma classe de tarefas T e medida de desempenho P se seu desempenho nas tarefas T, medido por P, melhora com a experiência E". Costuma-se subdividir o campo de ML em três paradigmas principais de aprendizado: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço.

No aprendizado supervisionado, o agente recebe vários exemplos compostos de uma série de características x juntamente com uma resposta alvo y, ou seja, tuplas com a estrutura (x, y). O objetivo do agente, então, é aprender um mapeamento entrada-saída $\hat{y} = f(x)$ a partir dos exemplos apresentados, de modo que a estimativa $\hat{y}$ seja próxima a y e uma boa generalização seja alcançada. O nome supervisionado refere-se à presença de um professor, visto que a resposta correta é fornecida em conjunto com as características. Visto dessa forma, o que se tem em essência é um problema de interpolação. Os problemas de regressão e classificação são exemplos de aprendizado supervisionado.

No aprendizado não supervisionado, apenas o conjunto de características x se encontra disponível, cabendo ao agente descobrir padrões ocultos presentes nos dados. Como as características não estão acompanhadas de uma resposta, o problema é dito não supervisionado, isso é, sem professor. Um exemplo desse paradigma é o problema de *clustering*, isso é, agrupar os dados de entrada em conjuntos de classes disjuntas.

O terceiro paradigma, de aprendizado por reforço, é foco desse trabalho. Se os dois outros envolvem encontrar padrões em um conjunto de dados, o aprendizado por reforço diz respeito a como um agente deve se comportar em um dado ambiente. A parte "por reforço" é uma referência à forma que o aprendizado ocorre: a partir

de estímulos externos de recompensa ou punição, que guiam o agente no sentido de alterar seu comportamento.

Um exemplo simples para o aprendizado por reforço é pensar em um agente que deve aprender a dirigir um carro de maneira próxima à humana. Seus sensores de entrada incluem câmeras, sensores de distância, velocímetro, dentre outros, de forma a conferir consciência do ambiente. O agente pode controlar o veículo por meio do volante e pedais, seus atuadores. O agente realiza em cada instante de tempo discreto uma ação[1], movendo seus atuadores de acordo com a indicação dos sensores. Se o agente colide com outros carros, sai da pista, dirige muito rápido, ele recebe uma punição. Se, por outro lado, ele se mantém em sua faixa, para nos sinais e obedece às leis de trânsito no geral, ele recebe um sinal de recompensa. Prosseguindo dessa forma, ao buscar maximizar seus ganhos, o agente pode modificar continuamente seu comportamento e alcançar um nível de direção semelhante ao humano. Essa breve descrição torna claro que esse paradigma pode ser aplicado à problemas do mundo físico, em ambientes dos mais variados tipos. Uma discussão mais detalhada deus seus fundamentos é postergada para a Seção 4.1.

Essa subdivisão do campo de ML em três paradigmas, contudo, deve ser visto como uma simplificação grosseira da realidade. Para certos problemas, a delimitação nem sempre é tão bem definida. Ademais, embora os três sejam os mais influentes, devem ser vistos apenas como exemplos dentre os muitos paradigmas do aprendizado existentes. A mente humana, por exemplo, é capaz de abstrações de alto nível e o processo de aprendizado não está limitado às técnicas citadas acima. Intuitivamente, um humano não aprende a dirigir um veículo apenas por meio de sinais de recompensa e punição, mas sim por uma interação complexa de muitos desses paradigmas. Um motorista pode buscar imitar seu instrutor ou aplicar os conhecimentos existentes de uma tarefa semelhante.

Nas últimas décadas, um outro subcampo de ML tem crescido em relevância: o aprendizado profundo (*deep learning* em inglês). Esse paradigma surge em parte dado a incapacidade dos algoritmos tradicionais de generalizar bem em certas tarefas de AI, como o reconhecimento de fala e de objetos. A generalização e o peso computacional se tornam mais desafiadoras em ambientes com número elevado de dimensões e o aprendizado profundo nasce com o objetivo de superar essas dificuldades (GOODFELLOW et al., 2016). O termo profundo é uma alusão ao elevado

[1] Ação, nesse contexto, deve ser entendida em geral como um vetor em $\mathbb{R}^n$, onde cada componente indica a ação de um atuador específico.

número de camadas existentes nas redes neurais.

Por último, menção deve ser feita ao conceito ambicioso de inteligência geral artificial (AGI, *artificial general intelligence* em inglês). Inteligência geral envolve a habilidade de alcançar uma variedade de objetivos e realizar uma variedade de tarefas em uma variedade de diferentes contextos e ambientes (GOERTZEL, 2014). O campo ainda permanece muito difuso e recente, mas suas ideias muito provavelmente permearão a AI em anos vindouros, à proporção que novas aplicações para tarefas mais amplas sejam desenvolvidas e princípios mais gerais sejam encontrados.

1.1 Objetivos

A breve exposição do aprendizado por reforço para o caso de direção de um veículo ilustra suas ideias gerais, bem como sua abordagem natural para resolver problemas de controle de uma forma mais ampla que o controle convencional. Reconhecendo suas potencialidades, bastante estendidas por ideias e algoritmos muito recentes, e tendo em conta o impacto e influência cada vez maior do campo de *machine learning*, esse trabalho tem como objetivo explorar a aplicação do paradigma do aprendizado por reforço ao controle de atitude de satélites.

Dominar essa abordagem alternativa poderia fornecer aos projetistas de missão uma ferramenta complementar para o controle de atitude, reduzindo os tempos e custos do projeto frente a estratégias convencionais, geralmente encontradas por um longo processo de tentativa e erro. Adicionalmente, tarefas mais desafiadoras, para os quais as técnicas do controle convencional não são propícias, poderiam ser atacadas dessa forma. De fato, em razão da generalidade de sua formulação, o aprendizado por reforço poderia ser aplicado a uma variedade de problemas da área espacial, muito além do controle de atitude.

Visando realizar essa investigação, a dinâmica de atitude correspondente ao ambiente de um satélite foi implementada e três algoritmos modernos do aprendizado por reforço, especificamente, DDPG, TD3 e SAC foram escolhidos para treinar um agente.

À vista disso, os seguintes objetivos gerais do trabalho podem ser delineados:

- empregar os algoritmos do aprendizado por reforço para o problema de controle de atitude de um satélite;
- comparar a solução obtida com a solução do controle convencional;

- comparar o desempenho dos três algoritmos apresentados, identificando fatores teóricos que influenciem nos seus resultados.

Alguns pontos mais específicos também devem ser mencionados:

- analisar a estabilidade e propriedades básicas do agente treinado;
- identificar heurísticas e otimizações propícias para o problema em questão;
- investigar a aplicação dessas técnicas ao problema de inércias variáveis, no qual os parâmetros básicos do satélite são desconhecidos pelo agente.

Isso é, buscar-se-á não apenas aplicar os algoritmos de maneira pura para a obtenção de seus resultados, mas também quando possível avaliá-los por um aspecto qualitativo, indicando os potenciais e as deficiências associadas a essa abordagem. É importante realçar essa observação, tendo em vista que por sua complexidade muitos algoritmos do campo de *machine learning* são vistos como "caixas pretas". Se as ideias do aprendizado por reforço devem encontrar mais uso na área de controle, então essa discussão é de grande importância.

Eis, em linhas gerais, a estrutura desse trabalho:

- o Capítulo 2 contém uma revisão bibliográfica dos desenvolvimentos mais importantes do campo do aprendizado por reforço, com especial atenção para os algoritmos mais recentes. O estado da arte no que diz respeito a sua aplicação em problemas de controle semelhantes também é apresentada;
- o Capítulo 3 tem como foco expor os fundamentos de atitude, em termos de suas representações mais comuns, sua cinemática, a mecânica de rotação de um corpo rígido e com rodas de reação e finalmente o controle de atitude em satélites propriamente dito;
- o Capítulo 4 expõe de forma extensa os fundamentos teóricos do aprendizado por reforço, culminando nos três modernos algoritmos a serem aplicados ao problema de controle. Tendo em vista o amplo uso de redes neurais como aproximadores de função, uma discussão mais breve sobre seus princípios também é apresentada;

- o Capítulo 5 brevemente discute a metodologia para a implementação dos algoritmos e das simulações do ambiente por meio da linguagem de programação Julia. Uma rápida visão do satélite Amazonia-1, cujos parâmetros foram empregados para algumas das simulações, é apresentada;
- o Capítulo 6 apresenta e discute os resultados das simulações de uma série de problemas de atitude, a saber, do controle unidimensional, no espaço e de inércias variáveis. Os resultados dos 3 algoritmos são comparados entre si e, quando possível, também com o controlador PD embarcado no Amazonia-1;
- o Capítulo 7 apresenta as conclusões advindas dos resultados obtidos e da experiência do autor ao longo do trabalho. Certas questões fundamentais ainda não bem esclarecidas são apontadas para possíveis trabalhos futuros.

Informações adicionais também estão contidas nos Anexos. Em particular, o Anexo C apresenta um relato do autor da experiência com a linguagem Julia e certas especificidades dos pacotes utilizados.

2 REVISÃO BIBILIOGRÁFICA

O capítulo atual tem como propósito apresentar uma visão geral do Aprendizado por Reforço e de trabalhos mais recentes aplicando-o no controle de atitude de veículos. A Seção 2.1 expõe de maneira resumida seu desenvolvimento histórico, incluindo ideias fundamentais e alguns desenvolvimentos recentes, que levaram aos seus modernos e poderosos algoritmos . A Seção 2.2 apresenta trabalhos com abordagem semelhante para o problema de controle de atitude.

2.1 Aprendizado por reforço moderno

O campo de Aprendizado por Reforço (RL, do inglês *Reinforcement Learning*) constitui um abordagem geral para a solução do problema de um agente interagindo com um ambiente a partir de sinais de recompensa ou punição. Em seus primórdios, no começo da segunda metade do século XX, foi profundamente influenciado por ideias e conceitos de dois outros campos: da psicologia do aprendizado, que busca descrever como animais aprendem, e do controle ótimo, cujo objetivo é otimizar a ação de um agente em um sistema dinâmico, sem se referir ao conceito de aprendizado. Dessa forma, combina o fim do controle ótimo com o meio do aprendizado.

A partir da década de 1980, ideias originais são introduzidas e o campo alcança sua independência, dando origem ao moderno aprendizado por reforço (SUTTON; BARTO, 2017). No que se segue, buscar-se-á expor brevemente esse desenvolvimento, enfatizando os resultados que são relevantes para esse trabalho.

A arquitetura do tipo ator-crítico foi apresentada por Barto et al. (1983), que a empregaram com sucesso para a solução do problema do pêndulo invertido montado em um veículo. Nessa arquitetura, na qual se baseiam grande parte dos algoritmos modernos, o ator representa a política, selecionando as ações a serem tomadas e alterando seu comportamento conforme o feedback do crítico.

Os métodos de diferença temporal (TD) foram discutidos em profundidade por Sutton (1988) e representam uma distinção fundamental entre RL e o aprendizado supervisionado . Eles receberam esse nome visto que o sinal de reforço advém da diferença da estimativa de um valor em tempos distintos. Ademais, eles incorporam dois aspectos essenciais do RL: a capacidade de aprender com a experiência por meio da interação com o ambiente e o uso de *bootstrapping*, tal como na programação dinâmica e controle ótimo, por meio da qual estimativas de valores são atualizadas em função de outras estimativas.

Watkins (1989) desenvolveu o algoritmo Q-learning, que permite a um agente encontrar uma forma ótima de agir em um ambiente. O ponto de partida do algoritmo é o conceito função de valor de ação $Q(s, a)$, que associa um escalar ao par estado e ação, estimando o quão favorável é tomar uma ação em um dado estado. A prova de convergência para o caso tabular estimulou outros desenvolvimentos no campo e o algoritmo se mostrou extremamente influente; os algoritmos mais modernos ainda buscam de uma forma ou outra estimar a função Q.

Tendo como base uma abordagem conexionista, Williams (1992) apresentou uma maneira simples de ajustar os pesos de uma rede com unidades estocásticas, provando que essas mudanças estão no sentido da esperança do gradiente da recompensa. Seu algoritmo foi denominado REINFORCE e generalizou alguns resultados anteriores. Apesar de suas limitações, representou uma contribuição importante para os métodos do tipo *policy gradient*.

Ao investigar a arquitetura de 8 tipos de agentes, Lin (1992) discutiu três estratégias para acelerar o aprendizado: uso de um replay de experiência- cujos elementos têm o formato $(estado\,inicial, ação, estado\,final, recompensa)$ -; o uso de um modelo de ação, de forma a simular o ambiente e permitir o planejamento de ações; e o aprendizado por meio de um expert externo, que indique ao agente uma solução adequada. Em suas simulaçõs, as três técnicas aceleraram o aprendizado; em particular, o autor realçou que o uso do replay também é de certa forma equivalente a aprender um modelo de ação.

O teorema *policy gradient* foi apresentado por Sutton et al. (1999). Esse resultado permite que uma política estocástica, parametrizada e diferenciável em seus parâmetros, possa ser sucessivamente movida em direção ao seu maior retorno por meio de um gradiente estocástico. Em particular, o teorema não requer conhecimento algum da dinâmica do ambiente - i.e. dito *model free*.

O gradiente para o caso de uma política determinística foi encontrado por Silver et al. (2014), que apontou a relação entre esse gradiente e o gradiente esperado da função valor de ação. De fato, dado condições especiais, prova-se que o gradiente de uma política estocástica cuja variância tende a zero converge rumo ao gradiente determinístico. Esse teorema constitui-se de um resultado notável, visto que é *model free*; até então, a existência desse gradiente era questionada ou acreditava-se que sua obtenção exigiria um modelo do ambiente. A classe de algoritmos ator-crítico fundamentada nele foi chamada de *Determinstic Policy Gradient* (DPG).

Aplicações em problemas mais complexos de RL, com número elevado de estados e ações, exigem a capacidade de generalização, o que por sua vez requer a aproximação de funções, de maneira similar ao que é comumente realizado no campo de aprendizado supervisionado. Métodos paramétricos são uma escolha natural; dentre eles, incluem-se métodos não lineares como redes neurais, parametrizadas por seus pesos. Nos últimos anos, o emprego de redes neurais - assim como outros métodos não lineares - em algoritmos de RL aptos a lidar com problemas de grande dimensão cresceu em popularidade, o que levou à definição do termo *Deep Reinforcement Learning* (DRL) (SUTTON; BARTO, 2017). Os parágrafos a seguir buscam expor de maneira breve esse desenvolvimento.

Mnih et al. (2015) apresentaram pela primeira vez um algoritmo eficiente para ambientes de alta dimensionalidade por meio do cálculo da função $Q(s, a)$ através de redes neurais profundas. O algoritmo foi denominado *Deep Q-Network* (DQN). De maneira a contornar os problemas de convergência enfrentados ao se usar aproximadores não lineares como redes neurais, duas ideias foram introduzidas: o uso de um replay de memória, para o armazenamento de transições no ambiente, e o emprego de uma segunda rede neural como o alvo da função perda; essa rede tem uma frequência de atualização inferior da outra rede, evitando instabilidades e permitindo a convergência. O algoritmo foi testado com uma série de jogos do console Atari 2600, superando o desempenho dos algoritmos disponíveis até então. Deve-se enfatizar que o DQN, dessa forma, é aplicável apenas a ambientes com espaço de ações discreto.

A tendência do algoritmo *Q-learning* em sobrestimar os valores de ação foi analisada em Hasselt et al. (2016) aplicada ao DQN. Esse comportamento decorre da preferência do *Q-learning* por valores sobrestimados frente aos subestimados. De maneira mais ampla, causas diversas como ruído do ambiente, aproximação de função e não estacionariedade também contribuem com um viés positivo, que tende a ser um fenômeno frequente e degrada o desempenho obtido. Reconhecendo sua ocorrência no DQN, uma solução foi proposta a partir da técnica de *Double Q-learning*. A idéia central consiste em usar duas funções $Q(s, a)$ diferentes: uma para selecionar a ação e a outra para avaliar a ação. O DQN foi ligeiramente modificado; a rede online determina a política e a rede alvo determina seu valor. O algoritmo foi chamado *Double* DQN (DDQN) e seu desempenho no mesmo ambiente do Atari 2600 é superior.

As ideias centrais do DQN em conjunto com a arquitetura ator-crítico do DPG foram aproveitadas por Lillicrap et al. (2015), que denominaram o algoritmo como *Deep*

Determinisitc Policy Gradient (DDPG). Esse algoritmo não só é capaz de resolver problemas de controle contínuos, como também possui a distinção de ser a primeira abordagem geral capaz de resolver problemas contínuos em altas dimensões. Redes neurais são empregadas tanto para o ator como para o crítico e as duas considerações da DQN, a saber, buffer de memória e redes alvos, permitem a convergência. O DDPG ainda utiliza-se da normalização de lote, de maneira a garantir que os componentes do vetor de estado tenham valores semelhantes. Uma técnica semelhante também vale para as camadas intermediárias. Embora a política, definida pelos parâmetros do ator, seja determinística, exploração do espaço de ações é garantida pela adição de um ruído à saída.

A persistência de valores de ação sobrestimados dentro da arquitetura ator-crítico foi identificada por Fujimoto et al. (2018). Os autores notaram que uma redefinição do DDPG semelhante ao feito com o DDQN traz poucos ganhos, visto que as redes atuais e alvos são muito semelhantes. Dessa forma, aproveitando ideias do *Double Q-learning*, três alterações ao DDPG foram apresentadas para melhorar a estabilidade e o desempenho: o emprego de um ator com dois críticos, com a estimativa subestimada sendo privilegiada para a atualização dos valores de ação na diferença temporal; uma atualização do ator menos frequente que a dos críticos, de maneira a garantir menor variância dos valores estimados; regularização dos valores de ação inserindo um ruído a ação usada no *bootstrapping*, com a consequência que os valores sejam melhor generalizados e mais suaves. Com essas modificações, o algoritmo foi denominado *Twin Delayed DDPG* (TD3).

Uma abordagem a partir da maximização da entropia foi desenvolvida por Haarnoja et al. (2018a), resultando no *Soft Actor-Critic* (SAC), um algoritmo cuja função de otimização considera, além do retorno, um termo associado a entropia da política. Dessa forma, políticas estocásticas são favorecidas e a política é incentivada a explorar o estado de ações. Um termo de temperatura ajusta a importância relativa dos dois termos; à medida que a temperatura diminui, a política tende a ser mais determinística. A motivação dos autores derivou da sensibilidade dos hiperparâmetros e da alta amostragem para a convergência observada em outros métodos. Os resultados comprovaram a estabilidade e robustez do SAC.

Pouco depois de sua publicação, Haarnoja et al. (2018b) aperfeiçoaram o SAC de maneira que o hiperparâmetro da temperatura seja ajustado automaticamente pelo algoritmo. Além de mostrar desempenho superior em ambientes de alta dimensão, esse trabalho é notável por aplicá-lo a 2 cenários no mundo real: um pequeno robô

quadrúpede com 8 atuadores e uma garra robótica com 9 graus de liberdade. Políticas robustas são aprendidas em ambos os casos; em particular, a garra aprende com sucesso a partir de um estado s composto de imagens brutas, isso é, pixels em formato RGB.

Ao debaterem fatores nocivos para a convergência, Sutton e Barto (2017) identificaram uma combinação de elementos que forma uma "tríade letal", cuja ocorrência em conjunto implica grandes dificuldades, formada por funções de aproximação, emprego de *bootstrapping* e métodos *off-policy*. Infelizmente, para ambientes mais complexos e desafiadores - nos quais aplicações modernas de RL têm sido testadas - a utilização dessas três ferramentas parece ser indispensável. Desse ponto de vista, as variações dos algoritmos acima representam um esforço teórico constante em direção a uma maior solidez da estimação dos valores $Q(s, a)$.

Mnih et al. (2016) propuseram uma abordagem assíncrona de *gradient descent*. Atores em paralelo interagem com suas próprias versões do ambiente, acumulando o gradiente, que é usado para atualização dos parâmetros. Quatro métodos, tanto *off-policy* como *on-policy* foram apresentados; o mais promissor é denominado *Asynchronous Advantage Actor-Critic* (A3C). Exploração é garantida à medida que os atores seguem políticas diferentes. O replay de memória não é usado; surpreendentemente, o uso de múltiplos atores tem um efeito estabilizante.

Um algoritmo alternativo em relação a abordagem ator-crítico acima foi descrito por Schulman et al. (2015), que o denominam *Trust Region Policy Optimization* (TRPO). A ideia principal do TRPO é variar os parâmetros de uma política estocástica de maneira que a sequência de otimizações seja monótona, i.e. as sucessivas políticas tragam retornos crescentes. Baseado em uma fundamentação teórica, o algoritmo é implementado como a maximização de uma função substituta sujeita a uma restrição da divergência das políticas, o que o torna um tipo de algoritmo minorização-maximização. Uma desvantagem do TRPO é sua complexidade amostral: novas experiências precisam ser adquiridas para avaliar novas políticas e experiências passadas não podem ser reaproveitadas.

Uma nova família de algoritmos foi proposta por Schulman et al. (2017), chamada *Proximal Policy Optimization* (PPO), inspirada na robustez do TRPO. A função substituta foi redefinida de forma que a política não varie de forma excessiva; outras generalizações também foram apresentadas, incluindo, por exemplo, termos de entropia, para incentivar a exploração. Assim, a restrição do TRPO não é inclusa de maneira explícita. Múltiplas épocas são utilizadas no gradiente estocástico para

a atualização da política. O PPO retém a estabilidade e confiabilidade do TRPO com melhor desempenho geral e uma maior simplicidade.

A despeito de suas diferenças, todos esses algoritmos citados são do tipo *model free*, ou seja, não aprendem diretamente um modelo da dinâmica do ambiente. A contrapartida a esse tipo são os algoritmos *model based*, que utilizam um modelo da dinâmica e o empregam juntamente com a interação com o ambiente para otimizar seu comportamento. Evidentemente, o problema cresce em complexidade, pois agora a dinâmica também precisa ser aprendida. No mundo real, a interação por tentativa e erro é cara e o uso de um modelo, mesmo que apenas próximo à realidade, oferece vantagens como uma menor complexidade amostral e a capacidade de planejamento, isso é, planejar as ações futuras dentro de um horizonte de tempo. As ideias ainda são difusas e não possuem a maturidade do caso *model free*, mas é de se esperar novos desenvolvimentos em anos vindouros (LUO et al., 2022).

2.2 Estado da arte

O moderno RL tem sido empregado em vários campos desafiadores com grande êxito. Por exemplo, o programa AlphaGo Zero empregou exclusivamente técnicas de RL para aprender a jogar o tradicional jogo de tabuleiro Go, dispensando auxílio humano por meio de exemplos de especialistas e alcançando desempenho super-humano (SILVER et al., 2017).

Uma área naturalmente propícia à aplicação do DRL é a robótica, tendo em vista a complexidade do hardware e do ambiente. Nesse campo, DRL tem sido aplicado para usos diversos, tais como manipulação precisa, rastreio de trajetória, navegação e planejamento de trajeto (ZHANG; MO, 2021).

Buşoniu et al. (2018) discutiram a teoria do RL e os desenvolvimentos recentes do DRL vistos da perspectiva da engenharia do controle convencional. Embora tenham reconhecido as similaridades e o potencial dessa abordagem para problemas práticos, identificaram algumas diferenças fundamentais e certos desafios ainda existentes. Em particular, a questão de estabilidade da solução foi julgada um problema ainda em aberto e a maior dificuldade para estender o RL a problemas de controle.

Ainda assim, dada a generalidade e capacidade dos algoritmos modernos, tem-se despertado o interesse de aplicar os métodos de RL para ambientes nos quais técnicas de controle clássico eram tradicionalmente empregadas. Um exemplo que tem atraído a atenção é o de veículos aéreos não-tripulados (UAVs, *Unmanned Aerial Vehicles*).

Para o controle de atitude, mais especificamente, estudos compararam implementações de controladores oriundos do DRL com o tradicional controlador proporcional-integral-derivativo (PID). Koch et al. (2018) implementaram um ambiente de simulação para um quadricóptero e avaliam controladores sintetizados pelo PPO, TRPO e DDPG. O controlador PPO supera o PID em quase todas as métricas. Bohn et al. (2021) utilizaram o SAC para simular o controle de atitude de uma aeronave de asa fixa, posteriormente o empregando em uma UAV real. O SAC se mostrou capaz de aprender uma política satisfatória usando apenas dados coletados em 3 minutos de voo.

Por outro lado, a aplicação do DRL no controle de atitude para satélites permanece um tópico menos explorado.

Ma et al. (2018) aplicaram com sucesso o DQN com um torque discretizado para o problema de captura de um veículo não cooperativo, uma tarefa que o controlador PD não é capaz de resolver. Su et al. (2019) aplicaram o DDPG modificado com recozimento para aperfeiçoar a otimização da política. Allison et al. (2019) treinaram um agente para o controle de atitude por *thrusters* e rodas de reação pelo PPO. Os resultados superam o controlador por *feedback* de quatérnion para uma larga faixa de massas de 0.1 kg a 100 toneladas.

No INPE, trabalhos recentes têm investigado aplicações de inteligência artificial no controle de atitude de um satélite. Carvalho (2021) analisou 4 arquiteturas de redes neurais do tipo proporcional-derivativa e as compara com o controlador PD em condições nominais e de falha. Marques (2021) aplicou o paradigma do DRL ao controle de atitude a partir de 3 algoritmos modernos já citados: DDPG, TD3 e SAC, que, apesar de convergirem, apresentam desempenhos diferentes. Apesar de suas diferenças, esses trabalhos mostram possibilidades de ganhos de um controlador inteligente frente ao controle convencional. Dessa maneira, esse trabalho buscar continuar esses esforços em direção ao controle inteligente.

Tipaldi et al. (2022) forneceram uma visão geral da literatura envolvendo métodos de RL aplicado a veículos espaciais, incluindo usos como sistemas de controle para pousos em planetas e asteroides, manobras de transferência orbital, controle de atitude, cenários de *rendezvous* e *docking* e tomada de decisões em espaçonaves e *rovers*.

Essa breve exposição atesta que as ferramentas do RL são capazes de encontrar soluções adequadas a problemas de controle, dentre os quais o controle de atitude

está incluso. Entretanto, os trabalhos que investigaram essa última possibilidade se concentraram, em sua essência, na aplicação dos algoritmos recentes conforme originalmente propostos, dando menor atenção a considerações de controle. Esse ponto engloba questões de estabilidade, heurísticas para o aprendizado e propostas do formato da solução. Um impeditivo para o uso mais difundido do RL em problemas do mundo real são os poucos resultados mais gerais do RL frente os poderosos resultados teóricos do controle convencional. Nesse sentido, o presente trabalho também investigará o RL a partir do ponto de vista do controle convencional.

3 ATITUDE

O presente capítulo tem como objetivo expor definições fundamentais de atitude, sua cinemática e dinâmica de propagação e uma visão geral da implementação do seu controle em satélites.

A importância do controle de atitude pode ser melhor exemplificada por alguns números. O satélite Amazonia-1 descreve uma órbita aproximadamente circular a 750 km da superfície terrestre. Um erro de apontamento de 1 centésimo de grau (36 segundos de arco) corresponde a um sensor apontado 130 metros para longe do seu alvo na superfície da Terra, que equivale a mais de 2 pixels do sensor ótico. Um caso extremo é o telescópio espacial Hubble, lançado em 1990. A necessidade de observar regiões do espaço profundo leva a requerimentos de apontamento extremamente estritos: ser capaz de estabilizar em um alvo com um erro médio quadrático não maior que 0,007 segundo de arco e retornar ao alvo com uma exatidão de 0,01 segundo de arco (BEALS et al., 1988).

Esses exemplos ilustram o papel da atitude na missão de veículos espaciais. É por meio do seu controle que satélites apontam seus sensores para regiões específicas, comunicam-se com estações terrestres, obtém energia do Sol através de painéis solares, realizam manobras para correções orbitais, dentre outras tarefas essenciais a serem executadas durante sua vida útil que são influenciadas pela atitude.

3.1 Representações de atitude

O termo atitude representa a orientação de um referencial em relação a um outro referencial escolhido. No espaço tridimensional, os graus de liberdade de um corpo rígido incluem não apenas sua posição, mas também sua orientação. Dessa forma, para um avião em voo por exemplo, sua atitude pode ser medida em relação ao horizonte local e descrever a inclinação de suas asas, o ângulo de seu nariz em relação ao horizonte e a proa que está tomando (norte, sul, etc). Claramente, o conhecimento apenas das coordenadas (x, y, z) da aeronave é insuficiente para determinar seu estado, visto que esta pode estar em voo de cruzeiro ou em uma emergência mergulhando com o nariz para baixo.

O exemplo apresentado apelou para a descrição da atitude a partir de 3 ângulos. De fato, eles correspondem à tradicional representação de atitude em termos dos ângulos de roll, pitch e yaw e podem ser facilmente visualizados. Outras representações são possíveis, com vantagens e desvantagens próprias. Na prática, o referencial inercial

costuma ser tomado como a referência, a partir do qual a orientação do referencial de interesse, geralmente o referencial atrelado a uma aeronave, satélite ou espaçonave, é expresso por meio de alguma representação.

Algumas das possibilidades para a descrição de atitude são apresentadas a seguir. Por convenção, variáveis não escalares serão escritas em negrito. Estas podem ser vetores, matrizes ou mesmo quatérnions; o seu significado exato deve se tornar claro de acordo com o contexto em que aparecem.

3.1.1 Direction cosine matrix

Consideremos de início um vetor qualquer $\vec{\boldsymbol{v}}$, assumido como sendo uma entidade própria, isso é, sua existência precede e independente de algum referencial específico. Sua representação em diferentes referenciais será, evidentemente, diferente. Considerando dois referenciais de mesma origem, mas diferentes orientações, dados pelas bases ortonormais $\{\hat{\boldsymbol{e_1}}, \hat{\boldsymbol{e_2}}, \hat{\boldsymbol{e_3}}\}$ e $\{\hat{\boldsymbol{e_1}}', \hat{\boldsymbol{e_2}}', \hat{\boldsymbol{e_3}}'\}$. Então, para o primeiro referencial, $\vec{\boldsymbol{v}}$ pode ser escrito como:

$$\vec{\boldsymbol{v}} = x\hat{\boldsymbol{e_1}} + y\hat{\boldsymbol{e_2}} + z\hat{\boldsymbol{e_3}}, \tag{3.1}$$

sendo x, y, z números reais que representam $\vec{\boldsymbol{v}}$ no primeiro referencial. Da mesma forma, como a representação é diferente do vetor em si, no segundo referencial:

$$\vec{\boldsymbol{v}} = x'\hat{\boldsymbol{e_1}}' + y'\hat{\boldsymbol{e_2}}' + z'\hat{\boldsymbol{e_3}}'. \tag{3.2}$$

A questão agora é estabelecer a relação entre essas representações, a saber, como (x, y, z) e (x', y', z') estão relacionados. Pela álgebra linear, o elemento $\hat{\boldsymbol{e_1}}$ pode ser reescrito projetando-o em $\{\hat{\boldsymbol{e_1}}', \hat{\boldsymbol{e_2}}', \hat{\boldsymbol{e_3}}'\}$, isso é:

$$\hat{\boldsymbol{e_1}} = \frac{\hat{\boldsymbol{e_1}}' \cdot \hat{\boldsymbol{e_1}}}{\hat{\boldsymbol{e_1}}' \cdot \hat{\boldsymbol{e_1}}'}\hat{\boldsymbol{e_1}}' + \frac{\hat{\boldsymbol{e_2}}' \cdot \hat{\boldsymbol{e_1}}}{\hat{\boldsymbol{e_2}}' \cdot \hat{\boldsymbol{e_2}}'}\hat{\boldsymbol{e_2}}' + \frac{\hat{\boldsymbol{e_3}}' \cdot \hat{\boldsymbol{e_1}}}{\hat{\boldsymbol{e_3}}' \cdot \hat{\boldsymbol{e_3}}'}\hat{\boldsymbol{e_3}}', \tag{3.3}$$

com $\cdot$ indicando o produto escalar. Sendo as bases ortonormais, então $\hat{\boldsymbol{e_1}}' \cdot \hat{\boldsymbol{e_1}}' = \hat{\boldsymbol{e_2}}' \cdot \hat{\boldsymbol{e_2}}' = \hat{\boldsymbol{e_3}}' \cdot \hat{\boldsymbol{e_3}}' = 1$. Então é possível reescrever a Equação 3.1 e igualar a 3.2:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{e_1}}' \cdot \hat{\boldsymbol{e_1}} & \hat{\boldsymbol{e_1}}' \cdot \hat{\boldsymbol{e_2}} & \hat{\boldsymbol{e_1}}' \cdot \hat{\boldsymbol{e_3}} \\ \hat{\boldsymbol{e_2}}' \cdot \hat{\boldsymbol{e_1}} & \hat{\boldsymbol{e_2}}' \cdot \hat{\boldsymbol{e_2}} & \hat{\boldsymbol{e_2}}' \cdot \hat{\boldsymbol{e_3}} \\ \hat{\boldsymbol{e_3}}' \cdot \hat{\boldsymbol{e_1}} & \hat{\boldsymbol{e_3}}' \cdot \hat{\boldsymbol{e_2}} & \hat{\boldsymbol{e_3}}' \cdot \hat{\boldsymbol{e_3}} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.4}$$

Essa equação descreve como a representação de $\vec{\boldsymbol{v}}$ muda de um referencial para o outro. A matriz 3x3 acima é chamada de Matriz de Cossenos Diretores ou em inglês *Direct Cosine Matrix* (DCM), visto que os produtos escalares são os cossenos dos

ângulos entre os versores e pode representar a atitude de um corpo em relação a um referencial escolhido.

3.1.2 Ângulos de Euler

A DCM possui 9 parâmetros, representados pelos seus elementos. É possível ter uma representação com menos elementos? Para fornecer uma resposta, é interessante observar que, independente do referencial considerado, a norma e o ângulo formado entre dois vetores devem ser os mesmos, isso é, invariantes. Isso equivale a impor a restrição que o produto escalar entre a representação dos vetores $\vec{\boldsymbol{u}}$ e $\vec{\boldsymbol{v}}$ seja o mesmo. Escrevendo a DCM como $\boldsymbol{R}$, tem-se:

$$\vec{\boldsymbol{u}} \cdot \vec{\boldsymbol{v}} = \boldsymbol{u}^{\boldsymbol{T}}\boldsymbol{v} = (\boldsymbol{Ru})^{\boldsymbol{T}}\boldsymbol{Rv} = \boldsymbol{u}^{\boldsymbol{T}}\boldsymbol{R}^{\boldsymbol{T}}\boldsymbol{Rv}, \tag{3.5}$$

onde $\boldsymbol{u}$ e $\boldsymbol{v}$ são matrizes coluna que designam as coordenadas dos vetores $\vec{\boldsymbol{u}}$ e $\vec{\boldsymbol{v}}$ em um referencial de interesse. Portanto, a condição que deve ser satisfeita é $\boldsymbol{R}^{\boldsymbol{T}}\boldsymbol{R} = \boldsymbol{I}$, o que implica $\boldsymbol{R}^{-\mathbf{1}} = \boldsymbol{R}^{\boldsymbol{T}}$. É trivial mostrar que isso implica 6 equações de restrições, de maneira que apenas 3 elementos da DCM são efetivamente livres. Essa discussão leva a conclusão que pelo menos 3 parâmetros são necessários para a representação de atitude.

A representação por ângulos de Euler utiliza-se desse fato, descrevendo a transformação de referencial como três rotações sucessivas ao longo de uma sequência específica, levando o referencial inicial a coincidir com o referencial final. Os parâmetros são os ângulos θ_1, θ_2 e θ_3 dessa sequência. A matriz equivalente pode ser obtida por:

$$\boldsymbol{R} = \boldsymbol{R_3}(\theta_3)\boldsymbol{R_2}(\theta_2)\boldsymbol{R_1}(\theta_1). \tag{3.6}$$

Para uma rotação ao longo do eixo z, por exemplo, a matriz é:

$$\boldsymbol{R_z}(\theta) = \begin{bmatrix} cos(\theta) & sen(\theta) & 0 \\ -sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.7}$$

Aqui cabe deixar claro que as rotações e ângulos mencionados estão associados às transformações de referenciais, não ao vetor em si, que permanece invariante, apesar das suas diferentes representações. De fato, essa é uma fonte comum de confusão e na literatura é feita uma distinção entre rotações passivas, que envolvem transformações dos referenciais, e rotações ativas, que envolvem rotações do vetor em si. Os conceitos de atitude abordados nesse trabalho dizem respeito a rotações passivas.

Sendo a multiplicação matricial não comutativa, então a ordem das rotações impactará na matriz resultante. Visto que a sequência exata das rotações precisa ser especificada, existem 12 opções para os ângulos de Euler. Uma das mais comuns é a sequência aeronáutica *roll-pitch-yaw*, definida como uma rotação em z por ψ (*yaw*), seguida por uma rotação em y' de θ (*pitch*) e finalmente uma rotação em x'' de ϕ (*roll*).

Apesar de seu apelo intuitivo e facilidade de visualização, os ângulos de Euler têm a desvantagem da presença de singularidade no segundo ângulo, quando este alcança um valor do tipo $k\pi/2\ rad$, com k um número inteiro (CARRARA, 2012). A ocorrência dessa situação é denominada *gimbal lock*.

3.1.3 Eixo-ângulo de Euler

É possível provar que uma rotação no espaço - equivalente a uma mudança de referencial - pode ser representada por uma rotação ao redor de um eixo específico $\vec{\boldsymbol{a}}$ por um dado ângulo θ. Vetores dispostos ao longo desse eixo não tem sua representação alterada. A especificação dessas duas grandezas leva à representação de eixo-ângulo de Euler. Que esse eixo efetivamente existe é provado a seguir de maneira não muito rigorosa.

Sabendo que $\boldsymbol{R^T R = I}$, tem-se $det(\boldsymbol{R^T})det(\boldsymbol{R}) = det(\boldsymbol{I}) = 1$. Como uma matriz quadrada e sua transposta possuem o mesmo determinante, chega-se a $det(\boldsymbol{R}) = 1$, visto que o valor -1 é excluído por envolver reflexão. Como o determinante de uma matriz é igual ao produto de seus autovalores, $\lambda_1\lambda_2\lambda_3 = 1$. Como a rotação preserva o produto escalar e, consequentemente, a norma dos vetores, deve ser o caso que $|\lambda_1|^2 = |\lambda_2|^2 = |\lambda_3|^2 = 1$. Caso contrário, para um autovetor $\boldsymbol{u}$ e seu autovalor λ teria-se:

$$\boldsymbol{(Ru)^T Ru} = \lambda \boldsymbol{u^T} \lambda \boldsymbol{u} = \lambda^2 \boldsymbol{u^T u} \neq \boldsymbol{u^T u}. \tag{3.8}$$

Ou seja, todos os autovalores, reais ou complexos, tem norma igual a 1. O polinômio característico de $\boldsymbol{R}$ em λ é de grau 3. Um polinômio de grau 3 com coeficientes reais possui uma raíz real e duas complexas conjugadas ou três reais, de forma que é trivial identificar que ao menos um autovalor deve ser igual a 1[1]. Sendo todos os

[1] Um corolário interessante desse resultado é que em dimensões pares não necessariamente existe um autovalor $\lambda = 1$ ou, consequentemente, um autovetor real. Em $\mathbb{R}^2$, por exemplo, em geral não há um vetor invariante após a rotação. O mesmo vale para $\mathbb{R}^4$, $\mathbb{R}^6$ e assim por diante.

elementos de R reais, é possível mostrar que o autovetor associado a esse autovalor também deve ser real e corresponde ao eixo $\vec{a}$.

O vetor $\vec{\boldsymbol{a}}$ pode ser encontrado observando que sua representação é a mesma nos dois referenciais. Lembrando que $\boldsymbol{R^{-1}} = \boldsymbol{R^T}$, então:

$$\boldsymbol{Ra} = \boldsymbol{R^T a}. \tag{3.9}$$

Logo:

$$\boldsymbol{(R - R^T)a} = \boldsymbol{0}. \tag{3.10}$$

A matriz $\boldsymbol{R} - \boldsymbol{R^T}$ é antissimétrica. Desse modo, usando a propriedade do produto vetorial, $\boldsymbol{a}$ pode ser encontrado fazendo $\boldsymbol{a}^\times = \boldsymbol{R - R^T}$. O ângulo θ pode ser encontrado notando que a rotação em torno de $\boldsymbol{a}$ por θ equivale à matriz (CARRARA, 2012):

$$\boldsymbol{R} = cos(\theta)\boldsymbol{I} + (1 - cos(\theta))\boldsymbol{aa^T} - sen(\theta)\boldsymbol{a}^\times. \tag{3.11}$$

Realizando a expansão termo a termo, chega-se a:

$$\boldsymbol{R} = \begin{bmatrix} c\theta + a_1^2(1-c\theta) & a_1a_2(1-c\theta) + a_3s\theta & a_1a_3(1-c\theta) - a_2s\theta \\ a_1a_2(1-c\theta) - a_3s\theta & c\theta + a_2^2(1-c\theta) & a_2a_3(1-c\theta) + a_1s\theta \\ a_1a_3(1-c\theta) + a_2s\theta & a_2a_3(1-c\theta) - a_1s\theta & c\theta + a_3^2(1-c\theta) \end{bmatrix}, \tag{3.12}$$

com a ressalva que $\vec{\boldsymbol{a}}$ é um vetor de norma unitária. Dessa forma, escrevendo o traço - definido como a soma dos elementos da diagonal principal - da DCM $\boldsymbol{R}$ como $tr(\boldsymbol{R})$ mostra-se que:

$$cos(\theta) = \frac{1}{2}(tr(\boldsymbol{R}) - 1). \tag{3.13}$$

3.1.4 Quatérnions

Em aplicações espaciais, a representação de atitude por quatérnions encontra grande uso, em razão de sua compacidade e eficiência computacional. Quatérnions são extensões dos números complexos e seu conjunto é comumente designado por $\mathbb{H}$. Um quatérnion $\boldsymbol{q}$ é composto por 4 elementos e é definido por:

$$\boldsymbol{q} = q_0 + q_1\, i + q_2\, j + q_3\, k \quad q_0, q_1, q_2, q_3 \in \mathbb{R}. \tag{3.14}$$

O elemento q_0 costuma ser chamado de parte escalar de $\boldsymbol{q}$, enquanto q_1, q_2, q_3 formam sua parte vetorial. Os elementos $1, i, j, k$ formam uma base de $\mathbb{H}$ e obedecem à

seguinte propriedade:

$$i^2 = j^2 = k^2 = ijk = -1. \tag{3.15}$$

Por meio dessa propriedade, é trivial mostrar que a multiplicação dos elementos i, j, k segue uma permutação circular, isso é, $ij = k$, $jk = i$ e $ki = j$.

As operações algébricas de soma e multiplicação de quatérnions seguem de imediato. A soma é trivial e consiste na adição dos elementos dos quatérnions elemento a elemento. Escrevendo um quatérnion $\boldsymbol{q}$ por meio de sua parte escalar e vetorial, isso é, fazendo $\boldsymbol{q} = (a, \boldsymbol{v})$, então mostra-se sem maiores dificuldades que a multiplicação dos quatérnions $\boldsymbol{q}$ e $\boldsymbol{q'}$, representada pelo símbolo $\otimes$, é:

$$\boldsymbol{q} \otimes \boldsymbol{q'} = (a, \boldsymbol{v}) \otimes (a', \boldsymbol{v'}) = (aa' - \vec{\boldsymbol{v}} \cdot \vec{\boldsymbol{v}}', a\vec{\boldsymbol{v}}' + a'\vec{\boldsymbol{v}} + \vec{\boldsymbol{v}} \times \vec{\boldsymbol{v}}'). \tag{3.16}$$

A presença do produto vetorial no segundo termo $\vec{\boldsymbol{v}} \times \vec{\boldsymbol{v}}'$ implica que a multiplicação de quatérnions não comuta, isso é, a ordem dos fatores da multiplicação afeta o resultado.

De forma similar aos números complexos, o conjugado de $\boldsymbol{q}$ é escrito como $\bar{\boldsymbol{q}}$ e designa o quatérnion com a mesma parte escalar, mas parte vetorial de sinal oposto, isso é, $\bar{\boldsymbol{q}} = q_0 - q_1 i - q_2 j - q_3 k$.

Os quatérnions permitem definir a atitude de forma semelhante à notação por eixo-ângulo, a partir dos parâmetros simétricos de Euler η e $\boldsymbol{\varepsilon}$, também denominados parâmetros de Euler-Rodrigues, que também baseiam-se no vetor $\vec{\boldsymbol{a}}$ e ângulo θ e são dados por:

$$\eta = cos(\theta/2), \tag{3.17}$$

$$\boldsymbol{\varepsilon} = \boldsymbol{a}\, sen(\theta/2). \tag{3.18}$$

Seguindo a definição de quatérnion, η representa a parte escalar e $\boldsymbol{\varepsilon}$ a parte vetorial dos parâmetros simétricos de Euler. Notar que o argumento das funções trigonométricas envolve o $\theta/2$ e não θ. Sendo $\vec{\boldsymbol{a}}$ um vetor unitário, então é evidente que:

$$\eta^2 + \boldsymbol{\varepsilon}^{\boldsymbol{T}}\boldsymbol{\varepsilon} = \eta^2 + \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 = 1. \tag{3.19}$$

Ou seja, o quatérnion com esses componentes tem norma unitária. Fazendo uso das identidades trigonométricas $cos(\theta) = 2cos^2(\theta/2) - 1 = 1 - 2sen^2(\theta/2)$ e $sen(\theta) = 2sen(\theta/2)cos(\theta/2)$, a matriz $\boldsymbol{R}$ na Equação 3.11 pode ser reescrita em função dos

parâmetros de Euler como:

$$\boldsymbol{R} = (\eta^2 - \boldsymbol{\varepsilon}^{\boldsymbol{T}}\boldsymbol{\varepsilon})\boldsymbol{I} + 2\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^{\boldsymbol{T}} - 2\eta\boldsymbol{\varepsilon}^{\times}. \tag{3.20}$$

Se o quatérnion $\boldsymbol{q_1}$ representa uma rotação em torno de $\vec{\boldsymbol{a_1}}$ por θ_1 e o quatérnion $\boldsymbol{q_2}$, de forma similar, uma rotação ao redor de $\vec{\boldsymbol{a_2}}$ por θ_2, então a composição dessas rotações, obtida realizando primeiro $\boldsymbol{q_1}$ e em seguida $\boldsymbol{q_2}$, é dada pelo parâmetro de Euler $\boldsymbol{q_3}$, que é (CARRARA, 2012):

$$\boldsymbol{q_3} = \boldsymbol{q_1} \otimes \boldsymbol{q_2}. \tag{3.21}$$

É possível mostrar que a multiplicação de dois parâmetros de Euler seguindo a álgebra de quatérnions é igual a outro parâmetro de Euler, isso é, que $\boldsymbol{q_3}$ também tem norma unitária. Deve-se observar que a ordem da multiplicação é oposta a dada na Equação 3.6. Para a representação do vetor $\vec{\boldsymbol{v}}$, a transformação é:

$$(0, \boldsymbol{v'}) = \bar{\boldsymbol{q}} \otimes (0, \boldsymbol{v}) \otimes \boldsymbol{q}, \tag{3.22}$$

onde $(0, \boldsymbol{v'})$ e $(0, \boldsymbol{v})$ designam quatérnions de parte escalar nula e cuja parte vetorial é a representação do vetor $\vec{\boldsymbol{v}}$.

Convém realçar a diferença entre quatérnions e os parâmetros de Euler. A álgebra de quatérnions é definida pelas Equações 3.14 e 3.15. Os parâmetros de Euler são definidos adicionalmente por 3.17 e 3.18 e obedecem à Equação 3.19. Em outras palavras, todo parâmetro de Euler é um quatérnion (mais precisamente, pode ser escrito como um), mas a recíproca não é verdadeira (SHUSTER, 1993).

3.2 Cinemática

A representação de atitude escolhida, seja ela por ângulos de Euler, DCM ou quatérnions, especifica a orientação de um corpo em um certo instante de tempo. Para corpos com movimento de rotação, é de interesse saber como essa representação evolui ao longo do tempo. Na cinemática de uma partícula, esse problema é análogo ao de calcular a posição de um corpo no instante $x(t + \Delta t)$, conhecendo sua posição em $x(t)$ e as derivadas desse valor, sem fazer referência às forças agindo sobre a partícula.

Para a DCM, a solução desse problema requer propagar a matriz $\boldsymbol{R}(t)$ conhecida no instante t para o próximo instante $\boldsymbol{R}(t + \Delta t)$. É evidente que a relação entre essas

matrizes deve ter o formato:

$$\boldsymbol{R}(t+\Delta t) = \boldsymbol{\Phi}(t+\Delta t, t)\boldsymbol{R}(t), \tag{3.23}$$

com $\boldsymbol{\Phi}(t+\Delta t, t)$ indicando uma outra matriz de rotação. Fazendo $\Delta t \to 0$, então $cos(\theta) \approx 1$ e $sin(\theta) \approx \theta = \omega\Delta t$, onde ω é a velocidade angular medida no referencial $\boldsymbol{R}$, isso é, o vetor $[\omega_x\ \omega_y\ \omega_z]^T$ medido no próprio corpo. Dessa forma, aplicando essas considerações na Equação 3.11 e desprezando termos de segunda ordem:

$$\boldsymbol{\Phi}(t+\Delta t, t) = \boldsymbol{I} - \theta\boldsymbol{a}^{\times} = \boldsymbol{I} - \Delta t\boldsymbol{\omega}^{\times}. \tag{3.24}$$

Logo, calculando o limite $(\boldsymbol{R}(t+\Delta t) - \boldsymbol{R}(t))/\Delta t$ com $\Delta t \to 0$, tem-se:

$$\dot{\boldsymbol{R}}(t) = -\boldsymbol{\omega}^{\times}\boldsymbol{R}(t). \tag{3.25}$$

Ou seja, a propagação da DCM é dada por:

$$\dot{\boldsymbol{R}}(t) = \begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix} \boldsymbol{R}(t). \tag{3.26}$$

O desenvolvimento para a derivada do quatérnion é bastante semelhante com as expressões acima e conduz a (CARRARA, 2012):

$$\dot{\boldsymbol{q}}(t) = \begin{pmatrix} \dot{\eta}(t) \\ \dot{\boldsymbol{\varepsilon}}(t) \end{pmatrix} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}(t))\boldsymbol{q}(t), \tag{3.27}$$

sendo $\boldsymbol{\Omega}(\boldsymbol{\omega}(t))$ a matriz 4x4:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}(t)) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\omega}^{\times} & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^{\boldsymbol{T}} & 0 \end{bmatrix}. \tag{3.28}$$

Uma forma eficiente de realizar a integração da Equação 3.27 é através do método

de Runge-Kutta de quarta ordem(RK4):

$$\boldsymbol{k_1} = \dot{\boldsymbol{q}}(\boldsymbol{q}(t), \boldsymbol{\omega}(t))\Delta t, \tag{3.29}$$

$$\boldsymbol{k_2} = \dot{\boldsymbol{q}}(\boldsymbol{q}(t) + \boldsymbol{k_1}/2, \boldsymbol{\omega}(t))\Delta t, \tag{3.30}$$

$$\boldsymbol{k_3} = \dot{\boldsymbol{q}}(\boldsymbol{q}(t) + \boldsymbol{k_2}/2, \boldsymbol{\omega}(t))\Delta t, \tag{3.31}$$

$$\boldsymbol{k_4} = \dot{\boldsymbol{q}}(\boldsymbol{q}(t) + \boldsymbol{k_3}, \boldsymbol{\omega}(t))\Delta t, \tag{3.32}$$

$$\boldsymbol{q}(t + \Delta t) = \boldsymbol{q}(t) + \frac{1}{6}(\boldsymbol{k_1} + 2\boldsymbol{k_2} + 2\boldsymbol{k_3} + \boldsymbol{k_4}). \tag{3.33}$$

Embora a Equação 3.27 seja de primeira ordem e portanto disponha de uma solução analítica, esta requer a exponenciação da matriz $\boldsymbol{\Omega}(\boldsymbol{\omega}(t))$ e, portanto, não é favorável numericamente.

Uma vantagem de se usar a representação por quatérnions é que a condição de normalização é trivial de ser implementada, fazendo $\boldsymbol{q}/||\boldsymbol{q}||$. Para a DCM, a condição é que suas linhas e colunas possuam norma unitária - proveniente de $\boldsymbol{R}\boldsymbol{R}^{\boldsymbol{T}} = \boldsymbol{I}$ e equivalente às 6 restrições mencionadas - e essas correções, necessárias pela presença de erros numéricos, são mais intrincadas.

É importante observar que na representação de atitude por eixo-ângulo de Euler uma rotação em torno do eixo $\vec{\boldsymbol{v}}$ por um ângulo θ equivale a uma rotação em torno de $-\vec{\boldsymbol{v}}$ por $2\,\pi - \theta$. Consequentemente, é trivial mostrar que os quatérnions $\boldsymbol{q}$ e $-\boldsymbol{q}$ descrevem a mesma orientação. Entretanto, um desses quatérnions representa uma rotação por um ângulo, em módulo, superior a 180 graus. Assim, o quatérnion cujo componente escalar for não negativo será privilegiado - isso é, que tiver $cos(\theta/2) \geq 0$, visto que assim garante-se que $|\theta|$ seja igual ou inferior a 180 graus.

Embora expressões análogas existam para a propagação de atitude nas representações de ângulos de Euler e eixo-ângulo de Euler, elas são significativamente mais complexas, de forma que elas não serão discutidas. Na prática, é mais fácil realizar a propagação por DCM ou quatérnions e então converter seus valores para uma outra representação de interesse.

3.3 Dinâmica

A cinemática se preocupa em descrever o movimento sem fazer qualquer referência às suas causas. A parte da mecânica que se ocupa em descrever o efeito da ação de forças no movimento é a dinâmica. Para o caso de uma partícula, se a cinemática busca encontrar sua posição dada sua velocidade, aceleração e possivelmente

outras derivadas da posição, a dinâmica descreve, pelas leis de Newton, como forças atuantes na partícula impactam sua trajetória.

É de especial interesse analisar como a atitude de um corpo rígido muda em função dos torques atuantes. A dinâmica rotacional, como será mostrado adiante, pode ser separada da dinâmica de translação e tratada à parte, sem preocupar-se com o movimento de translação. As chamadas equações de Euler são a contrapartida às equações de Newton. Dada a maior complexidade presente, contudo, o seu desenvolvimento requer a introdução de alguns conceitos prévios.

3.3.1 Momento angular

Dado um referencial de origem no ponto O, aqui assumido como inercial, o vetor momento angular $\vec{\boldsymbol{l}}$ de uma partícula de posição $\vec{\boldsymbol{r}}$ e momento $\vec{\boldsymbol{p}}$ é definido como:

$$\vec{\boldsymbol{l}} = \vec{\boldsymbol{r}} \times \vec{\boldsymbol{p}} = \vec{\boldsymbol{r}} \times \boldsymbol{m}\vec{\boldsymbol{v}}, \tag{3.34}$$

onde a velocidade é $\vec{\boldsymbol{v}} = \dot{\vec{\boldsymbol{r}}}$. O momento angular total $\vec{\boldsymbol{L}}$ de um sistema de N partículas é simplesmente a soma vetorial do momento de cada partícula $\vec{\boldsymbol{l_i}}$, isso é:

$$\vec{\boldsymbol{L}} = \sum_{i=1}^{N} \vec{\boldsymbol{l_i}} = \sum_{i=1}^{N} \vec{\boldsymbol{r}}_i \times \boldsymbol{m}_i \vec{\boldsymbol{v}}_i. \tag{3.35}$$

Definindo o torque $\vec{\boldsymbol{t}}$ de uma força $\vec{\boldsymbol{F}}$ aplicada no ponto $\vec{\boldsymbol{r}}$ como:

$$\vec{\boldsymbol{t}} = \vec{\boldsymbol{r}} \times \vec{\boldsymbol{F}}, \tag{3.36}$$

então a variação temporal do momento $\vec{\boldsymbol{L}}$ pode ser determinada a partir das forças externas atuantes no sistema. Derivando 3.35 e reconhecendo que $\vec{\boldsymbol{F_i}} = m_i \dot{\vec{\boldsymbol{v}}}_i$, onde $\vec{\boldsymbol{F_i}}$ é a força resultante agindo em i, chega-se a:

$$\vec{\boldsymbol{T}} = \sum \vec{\boldsymbol{t}} = \dot{\vec{\boldsymbol{L}}}. \tag{3.37}$$

Em outras palavras, a variação do momento angular do sistema de N partículas é igual ao torque externo aplicado. Notar que torques internos, isso é, entre partículas, não alteram o momento angular total, pois os torques se cancelam por ação e reação.

Pode-se provar que, em geral, o momento angular de um sistema de partículas pode ser decomposto em duas somas: uma parte devido ao movimento do centro de massa do sistema e outra parte devido ao movimento relativo a esse centro de massa. Isso

leva ao resultado que a taxa de variação do momento angular em relação ao centro de massa é igual ao torque resultante medido em relação ao centro de massa (TAYLOR, 2013).

A consequência dessa observação é permitir desacoplar o problema de órbita do de atitude. Isso é, forças cuja linha de atuação passam pelo centro de massa não alteram a atitude do veículo, embora alterem seus parâmetros orbitais. De maneira similar, binários puros - thrusters atuando em sentidos opostos, por exemplo - influenciam unicamente a atitude. Na prática, essa interação é mais complexa: erros da atitude, por exemplo, levam à introdução de erros em uma manobra de queima de órbita, implicando em erros orbitais. Para o presente caso, contudo, esse observação será suficiente para tratar a propagação de atitude de maneira isolada, sem considerar a órbita do veículo.

3.3.2 Momento de inércia

De particular interesse é a aplicação do formalismo apresentado a um corpo rígido, que pode visto como um sistema de N partículas no qual as distâncias entre elas são fixas. Assumindo que o corpo gira ao redor de um eixo qualquer com velocidade angular $\boldsymbol{\omega} = [\omega_x \ \omega_y \ \omega_z]^T$, então a velocidade de um ponto qualquer é dada por $\vec{\boldsymbol{\omega}} \times \vec{\boldsymbol{r}}$ e o momento angular $\vec{\boldsymbol{L}}$ pode ser encontrado usando 3.35, com a distinção que o somatório se torna uma integral. Desenvolvendo esse termo, chega-se aos componentes de $\vec{\boldsymbol{L}}$:

$$L_x = \omega_x \int (y^2 + z^2) dm - \omega_y \int xy dm - \omega_z \int xz dm, \tag{3.38}$$

$$L_y = -\omega_x \int xy dm + \omega_y \int (x^2 + z^2) dm - \omega_z \int yz dm, \tag{3.39}$$

$$L_z = -\omega_x \int xz dm - \omega_y \int yz dm + \omega_z \int (x^2 + y^2) dm. \tag{3.40}$$

Essa equação pode ser escrita de forma compacta como $\boldsymbol{L} = \boldsymbol{I}\boldsymbol{\omega}$, onde $\boldsymbol{I}$ é o tensor momento de inércia do corpo, escrito como:

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}. \tag{3.41}$$

É evidente que $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$ e $I_{yz} = I_{zy}$. Logo, a matriz $\boldsymbol{I}$ é dita ser simétrica,

isso é, $\boldsymbol{I} = \boldsymbol{I}^{\boldsymbol{T}}$. Um teorema fundamental da álgebra linear assegura que qualquer matriz simétrica e com componentes reais pode ser transformada em uma matriz diagonal com elementos reais (LAY, 2013). Isso é, existe um sistema de coordenadas ortogonal $x'y'z'$ com mesma origem no qual a matriz de inércia $\boldsymbol{I'}$ do corpo tem o formato:

$$\boldsymbol{I'} = \begin{bmatrix} I_{x'x'} & 0 & 0 \\ 0 & I_{y'y'} & 0 \\ 0 & 0 & I_{z'z'} \end{bmatrix}. \tag{3.42}$$

Ademais, sendo $\boldsymbol{I}$ uma matriz definida positiva, então todos esses elementos serão positivos. Os eixos desse sistema de coordenadas são chamados de eixos principais e os termos diagonais acima são chamados de momentos de inércia principais. Dessa forma, todo corpo rígido possui um sistema de eixos principais. O significado físico desses eixos deve ser apreciado: se um corpo rotaciona em torno de um eixo principal com velocidade $\vec{\omega}$, então o seu momento angular $\vec{L}$ também estará no mesmo sentido de $\vec{\omega}$ (TAYLOR, 2013).

3.3.3 Equações de Euler para a dinâmica de rotação

As equações da mecânica tendem a assumir no referencial inercial sua forma mais simples. Entretanto, em muitos casos é de interesse prático expressá-las vistas de um referencial não-inercial. Embora as Equações 3.38, 3.39 e 3.40 forneçam o momento angular, elas o expressam conforme visto do referencial inercial. Ao longo do tempo, a orientação do corpo se altera, logo os momentos de inércia também devem ser função do tempo. É mais interessante analisar o movimento por um referencial preso ao corpo, de maneira que os momentos de inércia permaneçam constantes. Como agora se trata de um referencial girante, termos adicionais devem ser inclusos. Essas considerações levam às equações de Euler da rotação.

Como mencionado anteriormente, o momento angular em relação ao centro de massa pode ser analisado de forma separada, com o resultado que sua variação $\dot{\vec{\boldsymbol{L}}}_{\boldsymbol{CM}}$ em um referencial inercial é:

$$\vec{\boldsymbol{T}} = \dot{\vec{\boldsymbol{L}}}_{\boldsymbol{CM}}. \tag{3.43}$$

O chamado teorema do transporte permite correlacionar as variações de um vetor vistas em um referencial inercial $\mathtt{S}_0$ e em outro em rotação $\mathtt{S}$. Para um vetor $\vec{\boldsymbol{r}}$ qualquer tem-se que suas variações nos dois referenciais podem ser relacionadas

por (TAYLOR, 2013):

$$\left.\frac{d\vec{\boldsymbol{r}}}{dt}\right|_{S_0} = \left.\frac{d\vec{\boldsymbol{r}}}{dt}\right|_{S} + \vec{\boldsymbol{\omega}} \times \vec{\boldsymbol{r}}. \tag{3.44}$$

Sendo o vetor de interesse $\vec{\boldsymbol{L}}$, então chega-se a:

$$\dot{\vec{\boldsymbol{L}}} + \vec{\boldsymbol{\omega}} \times \vec{\boldsymbol{L}} = \vec{\boldsymbol{T}}, \tag{3.45}$$

onde para simplificar a notação a derivada do momento angular em relação ao referencial em rotação foi escrita como $\dot{\vec{\boldsymbol{L}}}$. Essa equação é conhecida como equação de Euler para a dinâmica rotacional e pode ser simplificada ainda mais usando $\boldsymbol{L} = \boldsymbol{I\omega}$ e assumindo que os eixos do sistema coincidem com os eixos principais. Mais especificamente, sendo I_k, ω_k e T_k respectivamente o momento de inércia, a velocidade angular e o torque externo aplicado referentes ao eixo principal k, então tem-se (CARRARA, 2012):

$$I_x\dot{\omega}_x = T_x + (I_y - I_z)\omega_y\omega_z, \tag{3.46}$$

$$I_y\dot{\omega}_y = T_y + (I_z - I_x)\omega_z\omega_x, \tag{3.47}$$

$$I_z\dot{\omega}_z = T_z + (I_x - I_y)\omega_x\omega_y. \tag{3.48}$$

Essas 3 equações formam um conjunto de equações acopladas e descrevem a variação da velocidade angular ao longo dos eixos principais do corpo em função dos torques externos aplicados. O acoplamento existente significa que não é possível em geral controlar cada eixo de maneira independente. Por exemplo, se um torque T_x for aplicado para controlar ω_x, então o aspecto dessas equações levará a uma variação de ω_y e ω_z, ainda que nenhum torque tenha sido aplicado nesses eixos.

De fato, no caso mais geral dado pela Equação 3.45 o acoplamento é ainda mais forte. O torque T_x, por exemplo, possui um impacto nos termos $I_{xx}\dot{\omega}_x$, $I_{xy}\dot{\omega}_y$ e $I_{xz}\dot{\omega}_z$. De maneira a facilitar o controle de atitude, os satélites tendem a ser projetados de forma que a matriz de inércia seja próxima a diagonal. Devida atenção é dada à distribuição interna das massas para garantir que os componentes não diagonais I_{xy}, I_{xz} e I_{yz} sejam muito menores que os elementos diagonais I_{xx}, I_{yy} e I_{zz}. Para o satélite Amazonia-1, por exemplo, a matriz de inércia é:

$$\boldsymbol{I_{Amaz}} = \begin{bmatrix} 310,0 & 1,11 & 1,01 \\ 1,11 & 360,0 & -0,35 \\ 1,01 & -0,35 & 530,7 \end{bmatrix} kg\ m^2. \tag{3.49}$$

Essa matriz está bem próxima de ser diagonal, de fato, seus autovalores - equivalentes aos momentos principais - são 309,971, 360,024 e 530,705. Mostra-se que esses dois sistemas de referência diferem por um ângulo de apenas 1,3 grau.

O efeito prático dessa medida é a virtual separação dos eixos na dinâmica de atitude, o que garante que os eixos possam ser controlados de maneira independente. Como as velocidades angulares encontradas em satélites também são pequenas, da ordem de 1 centésimo de radiano por segundo, os termos cruzados $\omega_i\omega_j$ em 3.46, 3.47 e 3.48 são muito pequenos e as equações se aproximam muito de $I_k\dot{\omega}_k = T_k$. Vale mencionar que um desacoplamento também pode ser alcançado ao fazer todos os momentos principais iguais, isso é, $I_x = I_y = I_z$. Nesse caso, que pode ser visualizado como o de uma esfera, os termos $I_i - I_j$ são todos nulos.

3.3.4 Dinâmica com rodas de reação

A interação das rodas com o satélite torna a equação da dinâmica de atitude mais complexa. Embora ambos possam ser vistos separadamente como corpos rígidos, cada roda possui um grau adicional de liberdade na forma da sua velocidade de rotação em torno do seu eixo. Dessa forma, considerando adicionalmente que elas não estão, em geral, dispostas ao longo dos eixos principais do satélite, a equação de dinâmica requer termos adicionais. Antes de apresentá-la, contudo, convém destacar alguns de seus aspectos qualitativos.

Sendo o torque aplicado às rodas $\vec{\boldsymbol{T}}_{\boldsymbol{rw}}$ interno, então o momento angular total do conjunto satélite-rodas deve permanecer constante, visto que o torque que o satélite aplica às rodas possui mesmo módulo mas sentido oposto ao aplicado pelas rodas no satélite. Essa propriedade de armazenar e trocar momento angular com o satélite faz o uso de rodas de reação atraente: esse controle de atitude não requer o consumo de propelentes e possibilita manobras com elevada precisão.

Tendo em vista que, por razões mecânicas, a velocidade angular das rodas é limitada a um dado valor máximo, um torque externo $\vec{\boldsymbol{T}}_{\boldsymbol{ext}}$ precisa ser aplicado de maneira periódica, de forma a extrair o momento angular em excesso do sistema satélite-rodas. O método mais comum de se fazer isso é a partir do uso bobinas magnéticas, aproveitando-se da interação da corrente elétrica com o campo magnético terrestre.

Para o satélite, denominando $\boldsymbol{I}_{\boldsymbol{sat}}$ seu momento de inércia em relação ao centroide, então a Equação 3.45 deve incluir o momento angular das rodas.

$$\boldsymbol{I_{sat}}\dot{\vec{\boldsymbol{\omega}}}_{\boldsymbol{sat}} = \vec{\boldsymbol{T}}_{\boldsymbol{ext}} - \vec{\boldsymbol{\omega}} \times \left(\vec{\boldsymbol{L}}_{\boldsymbol{sat}} + \vec{\boldsymbol{L}}_{\boldsymbol{rw}}\right) - \vec{\boldsymbol{T}}_{\boldsymbol{rw}}. \tag{3.50}$$

O termo adicionado $\vec{\boldsymbol{\omega}} \times \vec{\boldsymbol{L}}_{\boldsymbol{rw}}$ possui um significado físico que merece ser mencionado: ele corresponde ao efeito giroscópico advindo de girar às rodas em torno de um eixo diferente do seu eixo próprio de rotação. Como este eixo está fixo em relação ao satélite, um torque de resistência aparece.

A Equação 3.50 é apresentada em sua forma vetorial. Na prática, precisamos representá-la em alguma referencial para utilizar a notação matricial. Dessa forma, representando os vetores no referencial girante preso ao corpo, a equação para a dinâmica do satélite é (CARRARA, 2012):

$$\boldsymbol{I_{sat}}\dot{\boldsymbol{\omega}}_{\boldsymbol{sat}} = \boldsymbol{T_{ext}} - \boldsymbol{\omega}^{\times}\left(\boldsymbol{I_{sat}}\boldsymbol{\omega_{sat}} + \sum_{n=1}^{N} L_{rw,n}\boldsymbol{a_n}\right) - \sum_{n=1}^{N} T_{rw,n}\boldsymbol{a_n}, \tag{3.51}$$

onde N é o número total de rodas, $\boldsymbol{a_n}$ é a orientação da n-ésima roda em relação ao satélite, tal que $\boldsymbol{a_n^T a_n} = 1$ e os escalares $L_{rw,n}$ e $T_{rw,n}$ o momento angular e torque da n-ésima roda. $\boldsymbol{\omega_{sat}}$ deve ser entendido como a velocidade angular do satélite em relação ao referencial inercial medida no referencial do satélite. A dinâmica das rodas segue:

$$\dot{L}_{rw,n} = T_{rw,n}, \tag{3.52}$$

sendo $I_{rw,n}$ o momento de inércia da roda em torno do seu eixo, isso é, o momento axial, então sua velocidade angular $\omega_{rw,n}$ pode ser determinada por:

$$I_{rw,n}\omega_{rw,n} = L_{rw,n} - I_{rw,n}\boldsymbol{a_n^T}\boldsymbol{\omega_{sat}}. \tag{3.53}$$

O segundo termo acima corresponde à contribuição do momento angular devido a própria rotação do satélite e é, em geral, muito menor que o primeiro.

Resolvendo as Equações 3.51, 3.52 e 3.53, a aceleração angular $\dot{\omega}_{sat}$ do satélite é encontrada, o que permite a propagação de sua atitude, usando, por exemplo, as Equações 3.26 ou 3.27.

3.4 Controle

Conforme a Equação 3.50, o controle de atitude é possível a partir da aplicação adequada do torque $\vec{\boldsymbol{T}}_{\boldsymbol{rw}}$ às rodas. Sua reação $-\vec{\boldsymbol{T}}_{\boldsymbol{rw}}$ atua no satélite e portanto é

capaz de guiá-lo a uma orientação desejada. A questão do controle é encontrar uma lei que governe esse torque, que é função do Subsistema de Controle de Atitude (em inglês, *Attitude Control Subsystem*, ACS).

Para um controle de malha fechada, o estado do satélite - que inclui a atitude e velocidade angular - deve ser conhecido. Isso é a responsabilidade do Subsistema de Determinação de Atitude (em inglês, *Attitude Determination Subsystem*, ADS). O ADS se vale de sensores como giroscópios, sensores de estrelas, magnetômetros, sensores de horizonte, dentre outros, para estimar o estado da espaçonave a partir da combinação das indicações, o que geralmente é efetuado por um filtro de Kalman.

É evidente que na prática um erro de estimação da atitude impacta de alguma forma o controle. Por mais preciso que o controle seja, um erro permanente na estimação, por exemplo, deve levar a um erro permanente na atitude. Um resultado importante do controle clássico, entretanto, estabelece que os polos do controlador e do observador podem ser alocados separadamente, assumindo um sistema linear com coeficiente constantes (OGATA, 2010). As equações da atitude não são lineares, como pode ser visto pelas Equações 3.46, 3.47 e 3.48, entretanto essa separação dos problemas de estimação e controle foi aplicada com sucesso para a maioria das missões espaciais (MARKLEY; CRASSIDIS, 2014). Para a presente discussão, esse argumento justificará tomar o estado do satélite como um valor conhecido, isso é, o ACS tem acesso completo ao valor real.

O controle tem como objetivo trazer o satélite para o repouso a uma orientação específica, que sem perda de generalidade pode ser feito adotando um quatérnion de referência $\boldsymbol{q_{ref}} = 1 + 0i + 0j + 0k$, equivalente a $\theta = 0$ nas Equações 3.17 e 3.18. Na linguagem do controle, esse se trata do problema do regulador. Expressando $\vec{\boldsymbol{T}}_{\boldsymbol{rw}}$ no referencial do satélite, então uma escolha intuitiva para o controle é fazer:

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \boldsymbol{k_p} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} + \boldsymbol{k_d} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{3.54}$$

onde $\boldsymbol{k_p}$ e $\boldsymbol{k_d}$ são matrizes diagonais com termos positivos. Esse torque, por sua vez, precisa ser decomposto entre as N rodas de reação presentes no satélites, para obter o vetor $[T_1\ T_2\ \ldots\ T_N]^T$, onde cada elemento é o torque comandado a cada roda.

Desse modo, é preciso ter:

$$\begin{bmatrix} \boldsymbol{a_1} & \boldsymbol{a_2} & \dots & \boldsymbol{a_N} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{bmatrix} = T_1\boldsymbol{a_1} + T_2\boldsymbol{a_2} + \dots T_n\boldsymbol{a_n} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}. \tag{3.55}$$

É evidente que se $N < 3$, esse sistema não possuirá uma solução geral, exceto se o torque estiver na linha ou no plano das rodas. Se $N = 3$, esse sistema possuirá uma solução única. Intuitivamente, 3 rodas de reação são o mínimo necessário para se ter a capacidade de gerar um vetor torque arbitrário no espaço.

Caso existam mais que 3 rodas de reação operantes, então o sistema é indeterminado e infinitas soluções de $[T_1\ T_2\ \dots T_n]^T$ produzem o mesmo lado direito. Esse, de fato, é o caso mais comum na prática. As condições do ambiente espacial, a relativa alta frequência com que as rodas falham e a importância do controle de atitude levam à instalação de rodas adicionais. Isso garante uma redundância do controle, à medida que evita a ocorrência de um único ponto de falha no ACS. Mais rodas também permitem maior capacidade de armazenar momento angular e aplicação de torque, trazendo ganhos de desempenho.

O método específico para a determinação do torque varia de acordo com a missão. Contudo, uma das maneiras mais comuns de se realizar essa distribuição é a partir do uso da matriz pseudoinversa, também conhecida como inversa de Moore-Penrose, que generaliza a ideia de matriz inversa para matrizes não quadradas. Dada uma matriz $\boldsymbol{A_{M\times N}}$ de dimensões qualquer, sua pseudoinversa é denominada por $\boldsymbol{A^+}$.

A pseudoinversa obedece às seguintes propriedades: $\boldsymbol{AA^+A} = \boldsymbol{A}$, $\boldsymbol{A^+AA^+} = \boldsymbol{A^+}$ e simetria de $\boldsymbol{AA^+}$ e $\boldsymbol{A^+A}$. Se a matriz $\boldsymbol{A}$ possui rank completo, isso é, se suas colunas $\boldsymbol{a_1}$, $\boldsymbol{a_2}$ $\dots$ $\boldsymbol{a_N} \in \mathbb{R}^M$ constroem $\mathbb{R}^M$, então $\boldsymbol{A^+}$ é dada por (MARKLEY; CRASSIDIS, 2014):

$$\boldsymbol{A^+} = \boldsymbol{A^T}(\boldsymbol{AA^T})^{-1}. \tag{3.56}$$

Dessa definição, tem-se que $\boldsymbol{AA^+} = \boldsymbol{AA^T}(\boldsymbol{AA^T})^{-1} = \boldsymbol{I_M}$. Essa propriedade permite encontrar uma solução para ‘a Equação 3.55. Se em um sistema linear do tipo $\boldsymbol{A\,x} = \boldsymbol{b}$ sua solução é $\boldsymbol{x} = \boldsymbol{A^{-1}b}$, então, de forma análoga, o sistema na forma $\boldsymbol{A\,T_{rw,N}} = \boldsymbol{T_{rw}}$ será satisfeito se $\boldsymbol{T_{rw,N}} = \boldsymbol{A^+T_{rw}}$.

O uso da pseudoinversa permite mapear o torque no referencial do satélite ao torque aplicado às rodas. Essa solução possui a propriedade de minimizar a norma de $\boldsymbol{T_{rw,N}}$, isso é, o valor $T_1^2 + T_2^2 + \ldots T_N^2$ é mínimo.

4 APRENDIZADO POR REFORÇO

O presente capítulo expõe os conceitos fundamentais do Aprendizado por Reforço, assim como algumas ideias mais recentes que ampliaram seu alcance para problemas contínuos, culminando nos algoritmos modernos, três dos quais são apresentados e discutidos. Tendo em vista que por suas capacidades próprias redes neurais tornaram-se ubíquas em aplicações de RL, elas são brevemente abordadas ao final do capítulo.

4.1 Fundamentos

A presente seção almeja expor os fundamentos do aprendizado por reforço, ainda que de uma maneira pouco formal. O objetivo é introduzir as ideias do RL de maneira intuitiva, para que possam ser mais aprofundados nas seções seguintes.

O RL envolve o aprendizado de um agente por meio da interação com um ambiente específico. Em um dado instante de tempo discreto, o ambiente encontra-se num estado s. Assume-se que o estado descreve o ambiente de maneira completa e inequívoca e que é perfeitamente conhecido pelo agente, ou seja, o estado é completamente observável. O agente, seguindo algum comportamento particular, toma uma ação a, alterando o ambiente para um estado s' e recebendo uma recompensa escalar r. Caso o estado s' seja um estado não terminal, o processo continua; se s' for terminal, o episódio - isso é, a sequência de estado, ação, recompensa, estado - chega ao fim. Notar que apesar de ter acesso ao estado atual, não se assume que o agente tenha necessariamente acesso à dinâmica do ambiente, isso é, conhecimento da dinâmica de transição de estados.

O objetivo fundamental de RL é maximizar o retorno - definido pela soma das recompensas -, a partir de qualquer estado inicial. Isso envolve encontrar um comportamento ótimo, ou seja, uma forma de agir que atribuía a um dado estado uma ação específica - ou várias ações possíveis, para o caso estocástico -, de maneira que o retorno esperado seja o maior possível. Esse mapeamento é denominado política.

Essa ideia, embora simples, possui um apelo natural à forma como humanos e outros animais aprendem certas atividades. Uma criança, por exemplo, precisa aprender o controle apropriado dos músculos da perna para locomover-se. Esse aprendizado ocorre por um processo de tentativa e erro, no qual a distância percorrida é uma recompensa positiva e as quedas, representada pela dor, são uma recompensa negativa. Esse exemplo, evidentemente uma simplificação da realidade, ilustra como

De maneira geral, existem duas dificuldades associadas aos problemas no contexto de RL. A primeira é problema de atribuição de crédito, identificada primeiramente por Minsky (1961): após um episódio, o agente deve atribuir culpa ou crédito às decisões tomadas durante sua trajetória. Contudo, como a recompensa pode aparecer apenas após o final da interação com o ambiente – um xeque-mate no xadrez, por exemplo – e decisões iniciais limitam as decisões posteriores, a maneira de distribuir o crédito não é óbvia. Esse é o problema de ter recompensas atrasadas: se ela não é imediata, não é trivial qual ação tomar, pois o agente precisa projetar o futuro.

Segundo é a comumente chamada maldição da dimensionalidade, que aparece de diferentes formas no campo de AI. Os problemas mais interessantes que RL deve resolver possuem espaços de estados e ações extremamente grandes, em certos casos contínuos. Métodos tabulares tornam-se inviáveis e alguma forma de generalização é necessária. Considerando o xadrez novamente, um grande mestre tenderá a encontrar configurações do tabuleiro nunca vistas anteriormente e precisará encontrar o movimento apropriado nessas situações, usando o conhecimento de experiências similares e métodos heurísticos.

Esses dois pontos ilustram aspectos únicos do RL e sua discussão mais profunda pode trazer ganhos ao entendimento. Diferente do aprendizado supervisionado, no RL não há um professor para conferir uma resposta correta. Nesse primeiro caso, cabe ao agente adequar seus parâmetros internos de maneira que sua saída se aproxime da resposta, idealmente, zerando o erro para todas as entradas. No RL, por sua vez, algum sinal secundário, que não a recompensa, deve ser utilizado para guiar as ações a serem tomadas. Nesse sentido, as funções de valores $V(s)$ e $Q(s, a)$ cumprem esse papel. A arquitetura ator-crítico ilustra claramente esse ponto: o crítico atua como um "quase professor", fornecendo instruções para que o ator (a parte que efetivamente toma decisões) ajuste os parâmetros da sua política.

Igualmente, por desconhecer a resposta certa, o agente deve testar várias opções. Visto desse aspecto, RL é um problema de busca pelo melhor mapeamento estado-ação. Esse ponto leva a um dilema fundamental: *exploration* e *exploitation*. Por convenção, esse antagonismo será expresso respectivamente como exploração e proveito.

Exploração diz respeito ao agente explorar o espaço de ações disponíveis, mudando sua política, de maneira a adquirir novas experiências e, possivelmente, encontrar maiores retornos. A curto prazo, a exploração tenderá, muito certamente, a trazer perdas, mas essas podem ser compensadas por ganhos futuros. Por outro lado,

proveito refere-se à utilização por parte do agente de soluções já conhecidas e que tenham um bom retorno. Ao preferir o proveito, o agente agirá da melhor forma possível de acordo com seus conhecimentos prévios, mas ao se privar da exploração, poderá ignorar soluções ainda melhores.

Essas tendências antagônicas no RL podem ser vistas como o análogo da diferença entre extremos globais e locais: buscar um extremo global requer a realização de uma varredura do espaço, o que implica testar muitos pontos que estão longe de serem ótimos, enquanto achar um extremo local diz respeito a se concentrar em uma região específica bem conhecida, mas que não necessariamente é a melhor possível. Idealmente, o agente deve manter um equilíbrio entre esses dois termos, privilegiando de início o aspecto de exploração, para que tenha o mais rápido possível novas experiências, e depois tendendo lentamente para o proveito, para selecionar a melhor de suas experiências.

4.1.1 Formulação

A partir dos conceitos expostos acima, o objetivo agora se torna fornecer uma definição mais formal do problema enfrentado pelo RL. No instante discreto de tempo t, o estado do ambiente S_t é dado por $S_t = s$. A ação A_t realizada pelo agente é $A_t = a$, alterando o ambiente para o estado $S_{t+1} = s'$ no próximo instante e recebendo uma recompensa escalar r_{t+1}. Nota-se que a recompensa só está disponível no próximo instante de tempo, embora seja determinada por s e a, isso é: $r_{t+1} = r_{t+1}(S_t = s, A_t = a)$. Por ora, assume-se que tantos os estados como as ações são discretos.

A trajetória τ denota todos os estados visitados e ações escolhidas pelo agente no passado até o presente momento, isso é:

$$\tau = (S_0 = s_0, A_0 = a_0, S_1 = s_1, A_1 = a_1, \ldots, S_t = s_t, A_t = a_t). \tag{4.1}$$

O ambiente E define a forma como os estados transitam de S_t para S_{t+1}. No caso mais geral, E é estocástico e o próximo estado é dado por uma distribuição probabilística ao longo de todos os próximos estados possíveis. Essa probabilidade depende de toda a trajetória τ. A propriedade de Markov adotada pelo RL, contudo, assume que essa probabilidade pode ser escrita da seguinte forma:

$$P(S_{t+1} = s'|\tau) = P(S_{t+1} = s'|S_t = s, A_t = a). \tag{4.2}$$

Em outras palavras, a transição de um estado inicial para um outro estado depende apenas do estado inicial e da ação adotada nesse estado. O ambiente não possui memória da trajetória do agente e apenas o par $S_t = s, A_t = a$ influencia a transição. De início, a propriedade de Markov pode parecer uma condição restritiva. Contudo, muito frequentemente é trivial expandir o estado de forma a conter toda informação necessária do passado, de maneira que o estado atual seja suficiente para descrever a dinâmica. Evidentemente, a condição de normalização deve ser observada na Equação 4.2.

$$\sum_{s' \in \mathcal{S}} P(S_{t+1} = s' | S_t = s, A_t = a) = 1 \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \tag{4.3}$$

onde $\mathcal{S}$ representa o conjunto de estados e $\mathcal{A}$ o conjunto de ações, ambos finitos. Assim, o ambiente E define um mapeamento $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0; 1]$. Para simplificar a notação, as expressões do tipo $S_t = s$, $A_t = a$ serão omitidas.

A interação do agente com o ambiente inicia-se em um estado s_0 e prossegue até que um estado terminal $S_T = s_T$ seja encontrado no tempo T. A sequência $\tau = (s_0, a_0, s_1, a_1, \ldots s_{T-1}, a_{T-1}, s_T)$ define um episódio. A função $\rho_0(\mathcal{S})$ define a distribuição inicial dos estados, isso é, $\rho_0 : \mathcal{S} \to [0, 1]$.

O objetivo elementar do RL é maximizar a soma das recompensas ao interagir com o ambiente durante o episódio. Definindo a recompensa como um mapeamento do tipo $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, o retorno G_t, para um tempo t é a recompensa acumulada:

$$G_t = r_{t+1}(s_t, a_t) + r_{t+2}(s_{t+1}, a_{t+1}) + \ldots r_T(s_{T-1}, a_{T-1}) = \sum_{i=t}^{T-1} r_{i+1}(s_i, a_i). \tag{4.4}$$

O retorno G_0, ocultando a dependência de r, é:

$$G_0 = r_1 + r_2 + \cdots + r_T. \tag{4.5}$$

Em certas tarefas, entretanto, o agente interage com o ambiente de maneira contínua, sem alcançar um estado final, e não episódica. Uma definição mais geral, que evita a soma de termos infinitos, é descontar a recompensa por um valor $\gamma \in [0, 1]$, chamado de taxa de desconto. Esse parâmetro mede a importância de ganhos futuros em relação ao presente. Desse modo, o retorno ajustado G_t se torna:

$$G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_{i+1}(s_i, a_i). \tag{4.6}$$

O retorno G_0, por exemplo, agora é:

$$G_0 = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \ . \tag{4.7}$$

Essa definição generalizada é aplicável tanto para interações episódicas como contínuas. Equação 4.6 se reduz a 4.4 se $\gamma = 1$ e assumindo que um estado terminal transita para ele mesmo com recompensa zero.

A partir das considerações acima, o problema de RL pode ser descrito de maneira precisa como um processo de decisão de Markov (MDP, *Markov Decision Process* em inglês) - isso é, que obedece a 4.2 - definido pela tupla $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, cujo objetivo é a maximização de $\mathbb{E}[G_0]$, em outras palavras, do valor esperado de 4.7.

Um comentário teórico deve ser realizado. Em geral, o agente não tem acesso ao estado s diretamente, mas sim a uma observação o, que possivelmente depende do estado, isso é $o = o(s)$. Pode ser o caso do agente, por exemplo, não ser capaz de descobrir todos os elementos do estado, mas apenas uma fração deles. Nessa situação, um conjunto de estados é indistinguível, visto que produzem a mesma observação e o problema é dito ser um problema de decisão de Markov parcialmente observável (POMDP, *Partiallly Observable Markov Decision Process*, em inglês). Os algoritmos a serem discutidos pressupõem, entretanto, que os estados são totalmente observáveis, o que ocorre se $o = s$ ou se uma dada observação corresponder a um único estado.

4.1.2 Política e funções de valor

Pela definição de MDP, a função do agente é encontrar uma forma ótima de agir no ambiente E, o que envolve naturalmente a noção de associatividade: o agente deve associar estados específicos a ações específicas. Essa consideração leva a definição do conceito da política π de um agente, um mapeamento entre estados e ações tal que $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, com $\pi(a|s)$ representando a probabilidade condicional do agente adotar a ação a dado que o ambiente se encontra no estado s. Pela definição de probabilidade, a política deve obedecer a:

$$\sum_{a' \in \mathcal{A}} \pi(a'|s) = 1 \quad \forall s \in \mathcal{S}. \tag{4.8}$$

Se a política μ for determinística, então apenas uma ação possuirá probabilidade diferente de zero e o mapeamento terá o formato $\mu : \mathcal{S} \to \mathcal{A}$. O símbolo π denotará as políticas dos dois tipos, de maneira a generalizar os resultados. O objetivo do RL,

então, pode ser descrito de maneira ainda mais clara: para um dado MDP, encontrar uma política π que maximize $\mathbb{E}[G_0]$.

Se todos as políticas possíveis podem ser enumeradas, então uma abordagem de força bruta para o problema consistiria em listá-las e escolher a política com o maior retorno. Em particular, para políticas determinísticas, é fácil conferir que existem $|\mathcal{A}|^{|\mathcal{S}|}$ alternativas a serem comparadas. Para um dado ambiente, muito possivelmente apenas uma classe muito pequena desse total requereria a atenção do agente. Contudo, a necessidade de exploração também exige que uma grande fração das demais políticas sejam analisadas.

Esse breve argumento ilustra a maldição da dimensionalidade, uma limitação constantemente presente nos problemas de RL. Se, por exemplo, um dado ambiente contiver 10 estados e 2 ações possíveis em cada estado, o número de políticas determinísticas será de 1024. Caso o número de estados dobre para 20, então o número de políticas crescerá por um fator de 1000; caso o número dobre novamente, por um fator de 1 milhão. O retorno de cada uma dessas políticas também precisa ser estimado. Se o ambiente for estocástico, então muitas amostras seriam necessárias para obter uma boa estimativa.

Logo, é evidente que, exceto em problemas muito simples, uma análise por força bruta é inviável. Um outro formalismo é necessário para contornar essas dificuldades. O que se requer é definir algum sinal secundário, capaz de guiar o agente ao informá-la o quão bom é estar em um dado estado ou o quão bom é tomar uma dada ação. Essa é em essência a motivação por trás das chamadas funções de valor. A primeira delas é a de função de valor $V_\pi(s)$, definida para uma política π e estado s como (SUTTON; BARTO, 2017):

$$V_\pi(s) = \mathbb{E}_{a\sim\pi}[G_0|S_0 = s] = \mathbb{E}_{a\sim\pi}\left[\sum_{i=0}^{\infty}\gamma^i r_{i+1}(s_i, a_i)|S_0 = s\right]. \tag{4.9}$$

Em outras palavras, $V_\pi(s)$ denota o retorno ajustado esperado partindo do estado s e seguindo a política π. O operador esperança é necessário visto que em geral o ambiente e a política não são determinísticos.

De maneira semelhante, a função valor de ação $Q_\pi(s, a)$ indica o retorno ajustado

esperado ao se partir do estado s, tomar a ação inicial a e depois seguir a política π.

$$Q_\pi(s,a) = \mathbb{E}_{a_t\sim\pi, t>0}[G_0|S_0 = s, A_0 = a] = \mathbb{E}_{a\sim\pi}\left[\sum_{i=0}^{\infty}\gamma^i r_{i+1}(s_i,a_i)|S_0 = s, A_0 = a\right]. \tag{4.10}$$

Pela definição das Equações 4.9 e 4.10, é trivial mostrar que:

$$V_\pi(s) = \sum_{a\in\mathcal{A}}\pi(a|s)Q_\pi(s,a). \tag{4.11}$$

As funções de valor podem ser rescritas de uma forma alternativa usando a ideia de recursão:

$$V_\pi(s) = \mathbb{E}_{a\sim\pi}[r(s,a) + \gamma V_\pi(s')], \tag{4.12}$$

$$Q_\pi(s,a) = \mathbb{E}_{a_t\sim\pi, t>0}[r(s,a) + \gamma V_\pi(s')]. \tag{4.13}$$

O operador esperança pode ser substituído pelo conhecimento das probabilidades de transição P do ambiente :

$$V_\pi(s) = \sum_{a\in\mathcal{A}}\left(\pi(a|s)r(s,a) + \gamma\sum_{s'\in\mathcal{S}}P(s'|s,a)V_\pi(s')\right), \tag{4.14}$$

$$Q_\pi(s,a) = r(s,a) + \gamma\sum_{s'\in\mathcal{S}}P(s'|s,a)V_\pi(s') = r(s,a) + \gamma\sum_{s'\in\mathcal{S}}\left(P(s'|s,a)\sum_{a'\in\mathcal{A}}\pi(a'|s')Q_\pi(s',a')\right). \tag{4.15}$$

A Equação 4.14 aplicada a todos os estados forma um sistema linear de $|\mathcal{S}|$ equações e incógnitas. Portanto, para qualquer política π, pode ser resolvida, contanto que as probabilidades de transição P sejam todas conhecidas, juntamente com $r(s,a)$. A mesma lógica é válida para 4.15, que apresenta $|\mathcal{S}| \times |\mathcal{A}|$ incógnitas.

Conquanto viável matematicamente, essa opção é raramente explorada na prática. O RL busca lidar com ambientes desafiadores, sobre os quais pouco é conhecido. Métodos que encontram políticas ótimas sem referência explícita ao modelo do ambiente são chamados de métodos *model-free*. Os algoritmos a serem apresentados pertencem a essa classe.

4.1.3 Equação de Bellman

Uma política π^* é dito ser ótima se para qualquer outra política π e para todo estado $s \in \mathcal{S}$ tem-se $V_{\pi^*}(s) \geq V_\pi(s)$ (SUTTON; BARTO, 2017) Logo:

$$V_{\pi^*}(s) = \max_\pi V_\pi(s). \tag{4.16}$$

Em geral, pode existir mais de uma política ótima. É evidente que uma relação semelhante também é válida para $Q_{\pi^*}(s,a)$. Ademais, para π^* deve ser o caso que:

$$V_{\pi^*}(s) = \max_a Q_{\pi^*}(s,a). \tag{4.17}$$

A justificativa pode ser vista por uma simples prova por contradição. Se a Equação 4.17 não for válida, então o valor do estado pode ser aumentado modificando a política para a ação que maximiza Q no estado s. Mas isso contradiz a otimalidade de π^* por 4.16

Essas definições podem ser combinadas com as de função de valor conforme as Equações 4.14 e 4.15, chegando às equações de Bellman para otimalidade:

$$V_{\pi^*}(s) = \max_\pi V_\pi(s) = \max_a \left[r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V_{\pi^*}(s') \right], \tag{4.18}$$

$$Q_{\pi^*}(s,a) = \max_\pi Q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a'} Q_{\pi^*}(s',a'). \tag{4.19}$$

Essas equações possuem o formato recursivo e sua solução permite a determinação da política π^*, por exemplo, pelo argumento que maximiza o lado direito de 4.17. De maneira mais geral, elas pertencem aos métodos da programação dinâmica, um campo de otimização cuja principal ideia é particionar um dado problema em problemas menores de forma recursiva(BELLMAN, 1957).

De maneira similar ao discutido na Seção 4.1.2, o conjunto de equações pode ser resolvido, desde que o ambiente seja perfeitamente conhecido. Sua aplicação é limitada em razão do elevado número de equações e necessidade das probabilidades de transição. Contudo, essas equações possuem um valor teórico, à medida que fornecem uma forma direta de calcular π^*. Os métodos de RL buscam de formas diferentes encontrar essa mesma política.

4.1.4 Q-learning

Watkins (1989) apresenta o Q-learning, um algoritmo *model-free*, capaz de aprender uma política ótima π^* de maneira online ao interagir com o ambiente. Para todo estado não terminal, a estimativa do valor de ação $Q(s,a)$ é atualizada por:

$$Q(s,a) = Q(s,a) + \alpha(r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)), \tag{4.20}$$

com α sendo um pequeno número positivo. A Equação 4.20 possui o formato de uma equação de diferença temporal, atualizando a variável *valor* da seguinte forma:

$$valor \leftarrow valor + \alpha(alvo - valor). \tag{4.21}$$

No lado direito, *alvo* e *valor* indicam estimações da variável em instantes de tempo sucessivos e a diferença entre esses termos é que provoca a atualização. Dessarte, a variável caminha lentamente em direção ao *alvo*.

Os valores de $Q(s, a)$ convergem para $Q_{\pi^*}(s, a)$ com probabilidade igual a 1, contanto que os pares de estado e ação sejam visitados um número infinito de vezes, que as recompensas sejam finitas e que α obedeça certas propriedades especiais, diminuindo com o tempo(WATKINS; DAYAN, 1992). De posse de $Q_{\pi^*}(s, a)$, a determinação da política ótima é imediata.

Embora não seja evidente, o teorema acima é válido independente da política que seja seguida pelo agente, contanto que esta garanta a seleção com probabilidade não nula de todos os pares estado e ação. Isso é, a política precisa ser estocástica. Implícito aqui está a necessidade de garantir a exploração.

Uma opção para essa política de exploração é uma política ϵ-gananciosa. Essa política seleciona uma ação aleatória com probabilidade ϵ e é gananciosa com probabilidade $1 - \epsilon$, isso é, seleciona a melhor ação conforme:

$$a(s) = arg\max_{a'} Q(s, a'). \tag{4.22}$$

Para o Q-learning, as próprias estimativas $Q(s, a)$ podem ser utilizadas para determinar essa política.

Investigar uma política π ao interagir com o ambiente por meio de uma outra política β faz o Q-learning um algoritmo do tipo *off-policy*. A política π é chamada de política alvo, sobre a qual se deseja descobrir algo, e β política de comportamento, que efetivamente determina a interação com o ambiente e portanto as ações tomadas e os estados visitados. Ou seja, métodos *off-policy* obtém alguma informação a respeito de π de maneira indireta usando β. Em oposição aos métodos *off-policy*, métodos *on-policy* empregam a mesma política, isso é, $\pi = \beta$.

Uma pergunta óbvia é: por que usar duas políticas diferentes? A resposta é que, embora isso possa trazer instabilidades, as duas políticas tendem a possuir aspectos diferentes. A política alvo π é uma política ótima que otimiza os retornos e, por isso,

tende a ser determinística. A política β, por seu lado, tem o objetivo de garantir a exploração, de uma forma ou outra algo fundamental no RL, e consequentemente é estocástica. Em geral, sem definir uma distância específica, essas políticas são próximas uma da outra, tal como exemplificado pela política ϵ-gananciosa acima. A experiência tem mostrado que os métodos *off-policy* tendem a apresentar uma velocidade de aprendizado e desempenhos superiores ao métodos *on-policy*, principalmente em ambientes mais complexos, o que pode ser atribuído à exploração inata do ambiente proporcionada pela política β.

4.1.5 Iteração generalizada da política

A relevância das funções de valor no RL está na sua capacidade de sintetizar o quão favorável é para o agente estar em um dado estado ou executar uma dada ação estando em um certo estado. Desse modo, informação de retornos futuros esperados é trazida para o presente e o agente, ao ser ganancioso no valor de ação, maximizará seu retorno. Logo, esses sinais secundários $V(s)$ e $Q(s, a)$ permitem o ajuste da política π.

A contínua interação entre as funções de valor e a política rumo a uma política ótima pode ser formalizada pela ideia de iteração generalizada da política. O agente alterna entre a estimação da política e sua melhoria da seguinte forma (SUTTON; BARTO, 2017):

- estime para a nova política π o valor de ação $Q_\pi(s, a)$;
- melhore a política sendo ganancioso $\pi'(s) = arg\max_a Q_\pi(s, a)$, $\pi = \pi'$.

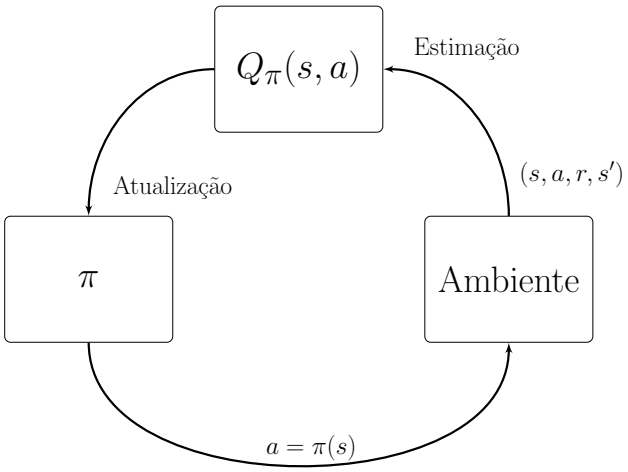
Essa ideia é ilustrada na Figura 4.1.

Uma ideia similar vale para $V_\pi(s)$. Pelas ideias da Seção 4.1.3 e 4.1.4, esse processo convergirá para π^*. Essa ideia de alternar estimação da política com atualização também está presente nos algoritmos a serem vistos na Seção 4.2.

4.1.6 Replay de memória

Um componente comum a todos os algoritmos a serem descritos é o de replay de memória, por vezes chamado de replay de experiência ou simplesmente buffer. À medida que o agente interage com o ambiente, o replay é preenchido por experiências, cuja estrutura é uma tupla do tipo (s, a, s', r). Na prática, um quinto elemento é adicionado, de maneira a estabelecer se o estado s' é terminal ou não. O replay suporta

Figura 4.1 - Iteração generalizada da política.



Fonte: Autor.

um tamanho fixo de experiências e conforme novas experiências são adquiridas, as mais antigas são excluídas.

A ideia do replay como forma de acelerar o aprendizado é primeiro introduzida e discutida por Lin (1992). Sendo o RL, em sua essência, um processo de tentativa e erro e tendo em vista o alto custo da interação com o ambiente - principalmente no mundo real -, é de interesse utilizar a experiência da forma mais eficiente possível. O Q-learning, conforme descrito na Seção 4.1.4, utiliza a experiência uma vez para atualizar a estimativa de $Q(s,a)$ e então a descarta. Ou seja, independente do valor da experiência, ela será esquecida pelo agente, a menos que ocorra novamente.

Uma alternativa simples a essa abordagem é armazenar as experiências e reutilizá-las futuramente. Em geral, seleciona-se aleatoriamente um número fixo de experiências do replay, que são utilizadas então para a atualização dos parâmetros do agente. Notar que como a política tende a ser continuamente alterada, as experiências presentes no replay em geral pertencerão a uma outra política diferente da política atual. Em outras palavras, o uso do replay torna o algoritmo *off-policy*.

4.1.7 Problema contínuo

Os métodos descritos até agora são ditos tabulares, pois assumem que todas as funções de valor podem ser escritas em uma tabela finita. Tanto o espaço de estados como o de ações formam um conjunto finito. Muitos problemas do mundo real - incluindo o controle tradicional -, entretanto, descrevem ambientes que variam de maneira contínua, que estão sujeitos a entradas que também são contínuas. Nesses casos, uma primeira abordagem pode simplesmente discretizar esses espaços. De fato, uma das primeiras aplicações do RL resolveu o problema do pêndulo invertido apoiado em um veículo dessa forma (MICHIE; CHAMBERS, 1968).

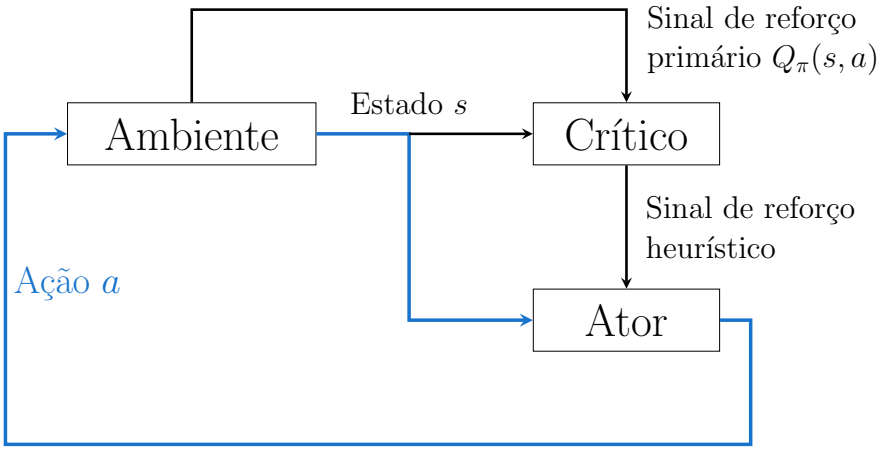
Contudo, uma análise mais atenta revela as limitações dessa abordagem. Uma discretização representativa deve exigir um número elevado de pontos, o que implicará uma elevada carga computacional. Por exemplo, no caso de um robô com muitos graus de liberdade, mesmo uma discretização grosseira leva a um modelo intratável. Ademais, parece inevitável que dispensar a continuidade do sistema leve a uma perda de informação e a políticas subótimas.

Consequentemente, o valor de ação $Q(s, a)$ não pode mais ser explicitamente escrito como uma tabela, mas sim ser determinado por uma função parametrizada $Q^w(s, a)$, onde w são seus parâmetros (e.g. coeficientes de um polinômio, pesos de uma combinação linear de características, pesos de uma rede neural). O mesmo vale para a política $\pi(a|s)$, escrita como $\pi_\theta(a|s)$. Visto que agora o ambiente possui infinitos estados e que o número de parâmetros é limitado, a função $Q^w(s, a)$ a rigor só pode se aproximar de $Q(s, a)$, reduzindo o erro mas sem nenhuma garantia de o zerar. O mesmo vale para a política $\pi_\theta(a|s)$ em relação a uma política desejada. Essa abordagem parametrizada dá origem à arquitetura do tipo ator-crítico, onde ator designa a política $\pi_\theta(a|s)$ e o crítico $Q^w(s, a)$.

A Figura 4.2 confere uma visão da arquitetura ator-crítico, muito semelhante ao que foi proposto inicialmente por Barto et al. (1983). Mais uma vez, convém enfatizar que no contexto do RL a resposta correta, isso é, a ação a ser realizada dado o estado do ambiente, não é previamente conhecida, visto que não há um professor. Nesse sentido, o crítico exerce uma função de um "pseudoprofessor", aprendendo um sinal primário a partir da interação com o ambiente.

De maneira similar ao problema tabular, a questão se torna como integrá-los de forma que os valores do crítico alterem os valores do ator na direção de políticas ótimas, isso é, como usar o sinal de reforço heurístico. Maximizar $Q^w(s, a)$ para um

Figura 4.2 - Visão da arquitetura ator-crítico.



Fonte: Adaptado de Haykin (2008).

dado estado de maneira a selecionar a melhor ação não é mais trivial, visto que o espaço de ações é contínuo. Resultados mais poderosos são necessários, a serem apresentados a seguir. ’

4.1.8 Gradiente da política determinística

Na arquitetura ator-crítico, a política passa a ser determinada por um vetor parâmetro $\theta \in \mathbb{R}^n$. De especial interesse são políticas determinísticas $\mu_\theta : \mathcal{S} \rightarrow \mathcal{A}$, tal que $a = \mu_\theta(s)$. Nesse formalismo, o objetivo é maximizar o desempenho $J(\mu_\theta) = E[G_0|\mu]$ no espaço de θ, isso é, encontrar a melhor política parametrizada.

O teorema do gradiente da política determinística é provado por Silver et al. (2014), o que permite a obtenção do gradiente $\nabla_\theta J(\mu_\theta)$. Esse resultado é significativo por certas razões. Alterar a política μ_θ não apenas altera as ações escolhidas, mas também a distribuição dos estados que é seguida, o que altera a recompensa $r(s,a)$. Mesmo assim, os autores provam que o gradiente pode ser obtido sem qualquer referência à dinâmica do ambiente, ou seja, pode ser utilizado por algoritmos *model-free*.

Mais especificamente, o gradiente $\nabla_\theta J$ é:

$$\begin{aligned} \nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right]. \end{aligned} \tag{4.23}$$

O lado direito desse teorema deve ser compreendido de forma matricial. O vetor $\theta = [\theta_1\, \theta_2 \ldots\, \theta_m]^T$ possui m elementos. A ação $\mu_\theta(s) = [a_1\, a_2 \ldots\, a_n]^T$ tem dimensão n. Logo, o jacobiano $\nabla_\theta \mu_\theta(s)$ é a matriz de dimensão $m \times n$:

$$\nabla_\theta \mu_\theta(s) = \begin{bmatrix} \frac{\partial a_1}{\partial \theta_1} & \frac{\partial a_2}{\partial \theta_1} & \cdots & \frac{\partial a_n}{\partial \theta_1} \\ \frac{\partial a_1}{\partial \theta_2} & \frac{\partial a_2}{\partial \theta_2} & \cdots & \frac{\partial a_n}{\partial \theta_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_1}{\partial \theta_m} & \frac{\partial a_2}{\partial \theta_m} & \cdots & \frac{\partial a_n}{\partial \theta_m} \end{bmatrix}. \tag{4.24}$$

De forma semelhante, o vetor $\nabla_a Q^\mu(s, a)$ é um vetor coluna com n elementos:

$$\nabla_a Q^\mu(s, a) = \begin{bmatrix} \frac{\partial Q^\mu(s,a)}{\partial a_1} \\ \frac{\partial Q^\mu(s,a)}{\partial a_2} \\ \vdots \\ \frac{\partial Q^\mu(s,a)}{\partial a_n} \end{bmatrix}. \tag{4.25}$$

Portanto, o produto é um vetor coluna com m elementos, correspondente ao gradiente de θ. A Equação 4.23 pode ser entendida intuitivamente pelo seguinte argumento. No caso tabular, a política μ deve ser alterada para selecionar a ação com o maior valor de $Q_\mu(s, a)$. Como no caso contínuo isso envolve uma maximização em cada passo, uma opção é alterar θ no sentido do gradiente $\nabla_\theta Q(s, a)$, em outras palavras, rumo a maiores retornos. Logo, os parâmetros devem ser atualizados conforme:

$$\theta \leftarrow \theta + \alpha \nabla_\theta Q(s, a). \tag{4.26}$$

Dessa forma, o vetor θ, composto pelos m elementos do ator, é alterado e a política é melhorada. Entretanto, o valor de ação não depende explicitamente de θ. Aplicando a regra da cadeia, o gradiente é $\nabla_\theta \mu_\theta(s) \nabla_a Q_\mu(s, a)$, onde a ação é $a = \mu_\theta(s)$. Infelizmente, o valor de ação $Q_\mu(s, a)$ não é previamente conhecido. Uma opção é usar seu valor estimado pelo crítico $Q^w_\mu(s, a)$. O caso *off-policy* é obtido usando a distribuição $s \sim \rho^\beta$.

Dessa forma, o valor esperado do gradiente pode ser estimado a partir de uma amostragem do ambiente. Esse resultado, em conjunto com a arquitetura ator-crítico, leva ao algoritmo DPG (*Deterministic Policy Gradient*) (SILVER et al., 2014).

4.1.9 Atualização do crítico

O teorema da política determinística fornece um importante resultado teórico para a atualização dos parâmetros do ator. Como o valor de ação $Q^{\mu}(s, a)$ não é exatamente conhecido, na prática os valores do crítico $Q^{w}(s, a)$ devem ser empregados. Por conseguinte, as estimativas do crítico têm grande influência na atualização do ator e no desempenho do agente. Uma questão que surge, entretanto, é como atualizar o crítico?

Primeiro, convém realizar uma observação. No caso tabular do *Q-learning*, havia $|\mathcal{S}| \times |\mathcal{A}|$ valores a serem preenchidos. Embora eles estejam relacionados entre si pela Equação 4.20, cada atualização altera apenas um valor em particular. Sendo o crítico parametrizado por w, então uma atualização desse parâmetro em um dado par estado-ação afetará a estimativa de todos os demais pares.

Ainda assim, os fundamentos do *Q-learning* podem fornecer um ponto de partida. Para uma interação com o ambiente com o formato (s, a, r, s'), sendo a política determinística $\mu_{\theta}(s)$, então uma possibilidade é definir uma função de perda $L^{w}(s, a)$

$$L^{w}(s, a) = \frac{1}{2}(Q^{w}(s, a) - r(s, a) - \gamma Q^{w}(s', \mu_{\theta}(s'))^2. \tag{4.27}$$

Desse modo, é possível realizar uma atualização de w no sentido de reduzir a Equação 4.27 a partir do gradiente $\nabla_{w} L^{w}(s, a)$. Convém observar que seu formato é $(valor - alvo)^2$. Tal como no *Q-learning*, a estimativa é atualizada em função de outra estimativa. Evidentemente, esse é um aspecto inerente ao RL: não há professor como no aprendizado supervisionado para conferir uma resposta.

No entanto, devido à presença do termo $Q^{w}(s', \mu_{\theta}(s'))$, o alvo também depende do vetor w. Em outras palavras, ao atualizar o parâmetro w, o alvo também será alterado. O crítico persegue um alvo móvel e essa propriedade traz consigo instabilidades, dificultando a convergência. Esse foi, de fato, um desafio enfrentado por alguns dos primeiros algoritmos do RL em ambientes complexos.

O DQN possui a distinção de resolver pela primeira vez esse problema de maneira satisfatória, com sua solução influenciando algoritmos posteriores. Primeiro, a Equação 4.27 não é mais minimizada para um único par estado-ação, mas para um conjunto destes extraído do replay de memória, conforme definido na Seção 4.1.6.

Segundo, de maneira a ter um alvo fixo para as atualizações, um segundo crítico

$Q^{w'}(s,a)$ é empregado, com parâmetros w'. Esse crítico fornece a estimativa do alvo $Q^{w'}(s', \mu_\theta(s'))$ para a atualização do crítico $Q^w(s,a)$, que é efetivamente utilizado para a seleção das ações. Os parâmetros w' são atualizados apenas após um dado número de atualizações de w (após 10000 atualizações, no trabalho original), sendo mantidos fixos nesse intervalo. Por meio dessas estratégias, a convergência do valor de ação é possível (MNIH et al., 2015).

4.2 Algoritmos

A presente seção almeja expor em mais detalhes os 3 algoritmos a serem usados para as simulações computacionais. Eles são: DDPG, TD3 e SAC. Antes de examiná-los, convém realçar seus pontos em comum. Todos são algoritmos capazes de resolver problemas com domínios de ação contínuo, possuem arquitetura ator-crítico, são do tipo *model-free* - isso é, não aprendem um modelo do ambiente -, utilizam-se de funções de valor e um replay de memória, o que os torna algoritmos *off-policy*. A iteração entre estimação e melhoria da política ocorre conforme a mesma abordagem da Seção 4.1.5, mas nenhum desses dois passos é executado até a convergência.

O DDPG e o TD3, em particular, diferem pouco um do outro, visto que o segundo agrega algumas modificações ao primeiro de maneira a evitar a sobrestimação da função $Q(s,a)$. O SAC, por sua vez, maximiza uma função diferente dos demais: a máxima entropia da política. Por sua definição, a política do SAC é estocástica, o que garante a exploração de maneira natural. Ruído precisa ser adicionado à política determinística dos outros dois algoritmos, de maneira a garantir a exploração.

Por fim, todos incorporam a mesma tríade letal apresentada na Seção 2.1. Seus elementos são: funções de aproximação não lineares, *bootstrapping* e métodos *off-policy*. Esses três aspectos qualitativos são a priori restrições para uma convergência das políticas para bons resultados, de maneira que há pouca fundamentação teórica que garanta que seus resultados sejam ótimos. Na prática, todavia, a adoção cuidadosa de certas heurísticas garante sua convergência para a absoluta maioria dos problemas de interesse, em especial, para o TD3 e SAC.

4.2.1 Deep Deterministic Policy Gradient

O DDPG combina as ideias que trouxeram o sucesso do DQN, que resolve problemas com espaços de ação discretos, com a equação do gradiente da política determinística do DPG. O DDPG é descrito no Algoritmo 1, conforme apresentado pelos autores no trabalho original (LILLICRAP et al., 2015).

Como o ator μ é determinístico, a ação tomada pelo agente é a soma da ação determinada pela política atual com o ruído gerado por um processo estocástico $\mathcal{N}$, o que garante a exploração.

O replay de memória R possui um tamanho fixo e coleta as experiências do agente à medida que interage com o ambiente, conforme discutido na Seção 4.1.6. A cada passo, um número fixo N de experiências é escolhido aleatoriamente do replay (chamadas de *minibatch*). Essas amostras são então usadas para atualizar os parâmetros do crítico Q, minimizando a função perda L com alvo Q', seguindo o apresentado na Seção 4.1.9.

O uso das redes alvos Q' e μ', cujos parâmetros variam lentamente, confere uma maior estabilidade ao algoritmo. Diferente do DQN, essas redes não variam após um numero fixo de iterações, mas sim caminham pouco a pouco rumo aos parâmetros das redes Q e μ; a velocidade dessa mudança é determinada pelo hiperparâmetro τ, uma pequena constante positiva.

Por fim, a política μ é alterada de acordo com o gradiente da política determinística, a Equação 4.23, por meio das estimativas do crítico Q.

Algorithm 1 Algoritmo DDPG

Inicialize aleatoriamente as redes do crítico $Q(s, a|\theta^Q)$ e ator $\mu(s|\theta^\mu)$ com pesos θ^Q e θ^μ
Inicialize as redes alvos Q' e μ' com pesos $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Inicialize o replay de memória R
for episódio $= 1, M$ **do**
  Inicialize um processo aleatório $\mathcal{N}$ para o ruído de exploração
  Receba o estado inicial s_1
  for $t = 1, T$ **do**
    Selecione $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ pela política atual e o ruído de exploração
    Execute ação a_t e observe recompensa r_t e novo estado s_{t+1}
    Armazene a transição (s_t, a_t, r_t, s_{t+1})
    Selecione um minibatch aleatório de N transições (s_i, a_i, r_i, s_{i+1}) de R
    $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Atualize o crítico pela minimização da função perda $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Atualização da política do ator pela amostragem do gradiente da política:
    $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
    Atualização das redes alvo:
    $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
    $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
  end for
end for

4.2.2 Twin delayed DDPG

Em sua formulação original, o DDPG apresenta uma tendência de sobrestimação dos valores de ação $Q(s, a)$. O TD3 insere algumas modificações no algoritmo original de maneira a evitar esse viés positivo, tornado o aprendizado mais robusto e permitindo a obtenção de melhores políticas. O algoritmo 2 descreve o TD3, seguindo a publicação original de seus autores (FUJIMOTO et al., 2018).

A sobrestimação é nociva à medida que a política favorece ações com maiores valores de ação. Dessa forma, ações que não são ótimas serão selecionadas e a política terá um desempenho inferior. O valor de Q não é conhecido de antemão, sendo estimado pelo agente. Logo, erros de estimação do valor de ação são inevitáveis, o que tende a levar a um viés positivo.

De maneira intuitiva, a sobrestimação pode ser melhor visualizada pelo seguinte argumento: o alvo da atualização TD é dado por $r+\gamma \max_{a'} Q(s', a')$, logo, o operador Q é empregado tanto para a seleção da ação (pelo argumento de máximo) como para a estimação do valor de ação. Dessa forma, valores otimistas são privilegiados. Uma possível solução para esse problema é desacoplar a seleção e estimação por meio de dois estimadores diferentes Q_1 e Q_2 (HASSELT et al., 2016)

Essa e mais outras duas ideias formam a base do TD3. A exploração pela adição de um ruído e o uso do replay de memória são idênticos ao DDPG. Contudo, duas redes Q_{θ_1} e Q_{θ_2}, juntamente com suas redes alvos $Q_{\theta'_1}$ e $Q_{\theta'_2}$, são utilizadas. Dessa maneira, o alvo para a atualização dos parâmetros θ_1 e θ_2 é $r + \gamma \min_{i=1;2} Q_{\theta'_i}(s', \tilde{a})$, ou seja, a menor estimação das redes alvo é escolhida. Embora esse alvo modificado induza um viés negativo, isso tende a ser compensado pela tendência de sobrestimação. Ademais, idealmente as duas redes tenderão à convergência.

Deve-se notar que a ação no alvo é $\tilde{a} = \pi_{\phi'}(s') + \epsilon$, quer dizer, um ruído ϵ é somado à ação da política. O objetivo desse termo adicional é suavizar a função $Q(s, a)$. O valor de ϵ é obtido a partir de uma distribuição normal $\mathcal{N}(0, \sigma)$, com a saída sendo limitada a um intervalo $[-c, c]$.

Finalmente, o ator π_ϕ é atualizado a uma frequência menor que a dos críticos Q_{θ_1} e Q_{θ_2}. O valor de d controla o intervalo entre essas atualizações. Um valor $d = 2$ é sugerido no trabalho original, isso é, uma atualização do ator a cada duas dos críticos. Dessa maneira, a política é atualizada com valores mais exatos do valor de ação. Notar que por convenção o ator é atualizado com os valores do crítico Q_{θ_1}.

Algorithm 2 Algoritmo TD3

Inicialize aleatoriamente as redes do crítico Q_{θ_1}, Q_{θ_2} e do ator π_ϕ com parâmetros θ_1, θ_2 e ϕ
Inicialize as redes alvos com parâmetros $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$ e $\phi' \leftarrow \phi$
Inicialize o replay de memória $\mathcal{B}$
for episódio $= 1, M$ **do**
  Selecione ação com ruído de exploração $a \sim \pi_\phi(s) + \epsilon$
  $\epsilon \sim \mathcal{N}(0, \sigma)$ e observe recompensa r e novo estado s'
  Armazene a transição (s, a, r, s')
  Selecione um minibatch aleatório de N transições (s, a, r, s') de $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$, $\epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Atualize os críticos $\theta_i = \arg\min_{\theta_i} \frac{1}{N} \sum (y - Q_{\theta_i}(s, a))^2$
  if $t \, mod \, d$ **then**
    Atualize ϕ pelo gradiente da política determinística:
    $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Atualize as redes alvo:
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

4.2.3 Soft Actor-Critic

A motivação do SAC é combater dois problemas principais que aflingem os algoritmos de RL: a complexidade amostral - isso é, o número elevado de interações com o ambiente necessário para a convergência - e a sensibilidade dos hiperparâmetros, que precisam ser ajustados de acordo com a tarefa para se obter um bom desempenho. Para tanto, o SAC possui uma abordagem diferente dos dois algoritmos acima, sendo baseado no princípio de maximização da entropia. Dessa forma, o ator busca simultaneamente obter sucesso na tarefa e agir de forma mais aleatória possível, o que o leva naturalmente à exploração. O algoritmo é descrito no algoritmo 3 (HAARNOJA et al., 2018b)

Mais especificamente, a função que o ator deve otimizar consiste agora do retorno somado à entropia da política, isso é:

$$J(\pi) = \sum_0^T \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [r_{t+1}(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))]. \tag{4.28}$$

O termo α representa uma temperatura que pondera a importância do termo entrópico frente ao retorno e é atualizado de maneira automática pelo algoritmo. Se

$\alpha = 0$, a função da Equação 4.28 se reduz à dos outros algoritmos. O termo γ foi omitido por conveniência, visto que sua presença torna a função mais complexa, mas é considerado pelo algoritmo.

O termo $\mathcal{H}$ representa a entropia da política. Em geral, a entropia de uma variável aleatória X está associada ao seu grau de incerteza. Se X é uma variável aleatória discreta, então sua entropia é:

$$\mathcal{H}(X) = \mathbb{E}[-log\, p(x)] = -\sum_{\mathbb{X}} p(x) log\, p(x), \tag{4.29}$$

com $p(x)$ indicando a probabilidade $X = x$ e $\mathbb{X}$ indicando o suporte de X. Dessa forma, a entropia conforme a Equação 4.29 é uma grandeza igual ou maior que zero. Ela é nula caso apenas uma probabilidade seja não nula, ou seja, se X tem um valor conhecido, e é máxima caso as probabilidades sejam todas iguais. Desse modo, a entropia de uma política determinística é zero, visto que não há incerteza quanto a suas ações.

A extensão para uma variável aleatória contínua é imediata, isso é, sendo $p(x)$ sua função densidade de probabilidade, então:

$$\mathcal{H}(X) = \mathbb{E}[-log\, p(x)] = -\int_{\mathbb{X}} p(x) log\, p(x) dx. \tag{4.30}$$

Deve-se notar que por essa definição a entropia pode mesmo ser negativa. Na realidade, a Equação 4.30 fornece a chamada entropia diferencial de X, correspondente à Equação 4.29 somada de um termo infinito (LATHI, 1998).

De maneira similar ao TD3, dois críticos θ_1 e θ_2 são utilizados, com o menor valor usado para o alvo. Consequentemente, duas redes alvos são também empregadas. A diferença é que o valor de ação é redefinido para considerar a entropia da política. Dessa forma, o alvo da Equação 4.27 é:

$$alvo = r(s,a) + \gamma \mathbb{E}_{s' \sim P}[V_\pi(s')], \tag{4.31}$$

com a função de valor $V_\pi(s)$ dada por:

$$V_\pi(s) = \mathbb{E}_{a \sim \pi}[Q_\pi(s,a) - \alpha\, log\, \pi(a|s)]. \tag{4.32}$$

Por meio dessas definições, as funções de perda $J_Q(\theta_i)$ são minimizadas para as estimativas dos dois críticos, usando-se as amostras extraídas do replay de memória.

A presença do termo de entropia implica que a política não pode ser determinística. Logo, o ator ϕ deve ser parametrizado de tal modo a apresentar um grau de aleatoriedade. A escolha mais comum é usar uma rede gaussiana, uma rede neural que fornece como saída a média e a covariância - assumida como diagonal - de uma distribuição normal.

Visto que a política é estocástica, a atualização ocorre de maneira diferente dos outros dois algoritmos. A ideia é que a política π_ϕ, uma distribuição do tipo $\pi_\phi(a|s)$, se aproxime da distribuição $e^{\frac{1}{\alpha}Q^\pi(s,a)}/Z$, onde Z é um fator de normalização, para que a Equação 4.8 seja satisfeita. Assim, a política favorecerá ações que tenham maiores valores de ação e, consequentemente, maiores retornos.

A referência usada para essa distância é a divergência[1] de Kullback-Leibler, que mede o quanto duas distribuições diferem. Sendo P e Q distribuições, então essa divergência é (HAYKIN, 2008):

$$D_{KL}(P||Q) = \sum_{\mathbb{X}} P(x) log \frac{P(x)}{Q(x)}. \tag{4.33}$$

Notar que essa divergência, embora sempre maior ou igual a 0, não comuta, isso é $D_{KL}(P||Q) \neq D_{KL}(Q||P)$. A divergência é 0 se as distribuições forem iguais. Logo, a função custo $J_\pi(\phi)$ para a política que deve ser minimizada é:

$$J_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi}[\alpha\, log(\pi(a|s)) - Q_\theta(s,a)]. \tag{4.34}$$

Com $\mathcal{D}$ indicando a amostragem realizada no *minibatch*. Evidentemente, como o valor esperado é calculado com amostras do replay, o algoritmo é *off-policy*. Não é difícil ver que a Equação 4.34 expressa o mesmo objetivo que 4.28: minimizar $-Q(s,a)$ equivale a maximizar $Q(s,a)$. Ademais, sendo o logaritmo uma função crescente, maiores valores de α incentivam a política a selecionar ações menos prováveis, isso é, com menor $\pi(a|s)$.

Por fim, a temperatura α é atualizada automaticamente minimizando a função $J(\alpha)$, dada por:

$$J(\alpha) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi}[-\alpha log\pi(a|s) - \alpha\bar{\mathcal{H}}], \tag{4.35}$$

com $\bar{\mathcal{H}}$ sendo a entropia alvo. Desse modo, espera-se que a temperatura inicial seja alta, para permitir a exploração, e que siga baixando à medida que o agente interage

[1] No campo da estatística, uma divergência é um tipo de distância entre distribuições menos restritivo que uma métrica. Por exemplo, embora uma divergência seja sempre positiva e não-negativa, ela não necessariamente comuta ou obedece à desigualdade triangular.

com o ambiente, até que a entropia da política alcance a entropia alvo. Os autores recomendaram $\bar{\mathcal{H}} = -dim(\mathcal{A})$.

Algorithm 3 Algoritmo SAC

Entradas: θ_1,θ_2,ϕ
$\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$
$\mathcal{D} \leftarrow \emptyset$
for cada iteração **do**
  for cada passo do ambiente **do**
    $a_t \sim \pi_\phi(a_t|s_t)$
    $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$
  end for
  for cada passo do gradiente **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ para $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i$ para $i \in \{1, 2\}$
  end for
end for
Saídas: θ_1, θ_2, ϕ

4.3 Redes neurais

Os algoritmos sobreditos são fundamentalmente agnósticos no que diz respeito à parametrização escolhida, contanto que esta seja diferenciável em seus parâmetros. Entretanto, em razão da sua capacidade de generalização, o ator e o crítico são muito frequentemente parametrizados por meio de redes neurais artificias (RNAs). Estas são parametrizadas, por sua vez, em função de seus pesos e vieses.

RNAs possuem uma inspiração biológica e são baseadas numa versão simplificada do funcionamento de células nervosas chamadas neurônios. Essa ideia foi primeiramente explorada por McCulloch e Pitts (1943) no modelo do perceptron, que construíram uma teoria lógica a partir da lei de tudo-ou-nada observada na atividade neuronal. Essa lei fundamenta-se na observação que um neurônio dispara - isso é, propaga o impulso elétrico - apenas se o valor de seu estímulo for superior a um limiar específico. A partir desse limiar, a saída do neurônio tem uma intensidade fixa.

As evidências da neurociência moderna, contudo, mostram que o funcionamento dos neurônios é significativamente mais complexo que o modelado por RNAs. Dessa

forma, para o presente tratamento, RNAs serão vistas como aproximadores de função não lineares, cuja principal utilidade reside na capacidade de generalização. Isso é, dado um vetor de entrada x, a rede produzirá um vetor de saída y definido por $y = f(x; \theta)$, onde θ são os parâmetros internos da RNA.

O elemento computacional básico da rede é o neurônio. Em geral, para o j-ésimo neurônio da camada k, que possui entradas $x_1, x_2 \dots x_n$, a saída y_j^k é dada por (HAYKIN, 2008):

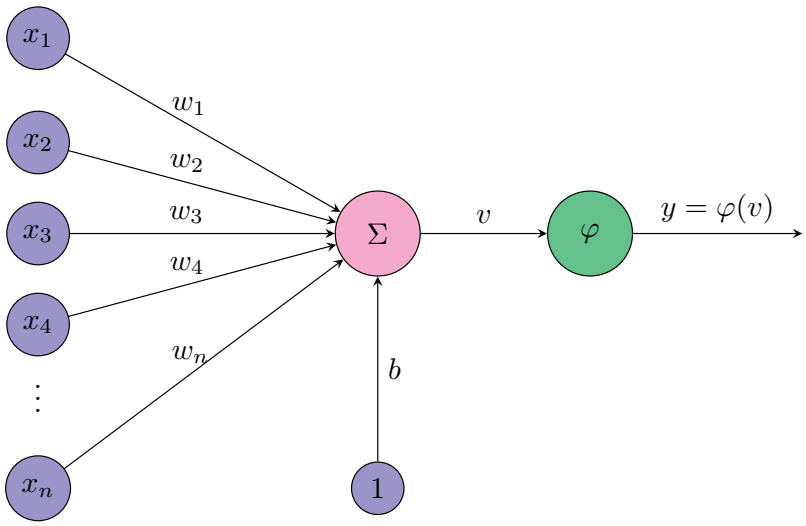
$$y_j^k = \varphi\left(\sum_{i=1}^{n} w_{ji}^k x_i + b_j^k\right), \tag{4.36}$$

φ é dita uma função de ativação, uma função não linear diferenciável que mapeia o termo entre parênteses à saída do neurônio. Os pesos w_{ji}^k ponderam a importância relativa das entradas x_i. O termo b_j^k é chamado de viés (em inglês, *bias*) e é uma constante adicionada à soma ponderada. Fazendo $x_0 = 1$ e $w_{j0}^k = b_j^k$, o viés pode ser incluso no somatório. Logo,a Equação 4.36 se torna:

$$y_j^k = \varphi\left(\sum_{i=0}^{n} w_{ji}^k x_i\right) = \varphi(v_j^k), \tag{4.37}$$

v_j^k é dita ser o campo induzido local. A Figura 4.3 ilustra o comportamento de um neurônio a partir de um vetor de entradas $[1\ x_1\ x_2\ \dots x_n]^T$.

Figura 4.3 - Visão esquemática de um neurônio.



Fonte: Autor.

Para os m neurônios da camada k, então o campo local induzido $\boldsymbol{v^k}$ pode ser calculado por:

$$\boldsymbol{v^k} = \begin{bmatrix} v_1^k \\ v_2^k \\ \vdots \\ v_m^k \end{bmatrix} = \begin{bmatrix} w_{10}^k & w_{11}^k & w_{12}^k \dots & w_{1n}^k \\ w_{20}^k & w_{21}^k & w_{22}^k \dots & w_{2n}^k \\ \vdots & \vdots & \ddots & \vdots \\ w_{m0}^k & w_{m1}^k & w_{m2}^k \dots & w_{mn}^k \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}. \tag{4.38}$$

A saída dessa camada $\boldsymbol{y^k}$, que pode servir de entrada para a camada $k+1$, pode então ser calculada por:

$$\boldsymbol{y^k} = \begin{bmatrix} y_1^k \\ y_2^k \\ \vdots \\ y_m^k \end{bmatrix} = \varphi(\boldsymbol{v^k}) = \begin{bmatrix} \varphi(v_1^k) \\ \varphi(v_2^k) \\ \vdots \\ \varphi(v_m^k) \end{bmatrix}. \tag{4.39}$$

4.3.1 Função de ativação

As funções de ativação definem a saída de um neurônio em função de seu campo induzido. As condições gerais para uma função φ foram mencionadas na Seção 4.3. A não-linearidade é uma condição essencial, visto que caso contrário a rede pode ser reescrita como uma transformação linear $y = Ax$. Em razão dos poucos requisitos, uma grande número de funções são concebíveis e uma extensa variedade existe na literatura. Entretanto, o enfoque aqui será restrito a três funções específicas comumente empregadas. Escrevendo $\varphi = \varphi(v)$, com v sendo o campo local de um neurônio qualquer, a função logística é:

$$\varphi(v) = \frac{1}{1+e^{-v}}. \tag{4.40}$$

A função tangente hiperbólico é:

$$tanh(v) = \frac{senh(v)}{cosh(v)} = \frac{e^v - e^{-v}}{e^v + e^{-v}}. \tag{4.41}$$

Por fim, a função $ReLU$, abreviatura para unidade linear retificada, é definida por[2]:

$$ReLU(v) = \max(v, 0). \tag{4.42}$$

A Figura 4.4 apresenta os gráficos dessas funções de ativação. De maneira geral, o formato de cada uma as fazem indicadas para aplicações diferentes. As funções logística e tangente hiperbólico são ditas funções sigmoides, visto que tendem assintoticamente para valores específicos em seus dois extremos, tendo um formato de S. Nesses extremos, diz que a função saturou. A imagem no intervalo $[0; 1]$ torna a sigmoide propícia aos problemas de classificação, onde sua saída denota o grau de confiança de pertencimento de um vetor a uma classe específica. A função tangente hiperbólico, com imagem em $[-1; 1]$, é, dentre outras possibilidades, adequada para problemas de controle onde a saída está limitada a dois extremos de mesmo módulo mas sinal oposto. Por último, a imagem $[0; \infty[$ da $ReLU$ é atrativa em problemas de controle nos quais não há limitação ao módulo da saída, como, por exemplo, nos estados do sistema ou, para o presente caso, a estimação de $Q(s, a)$.

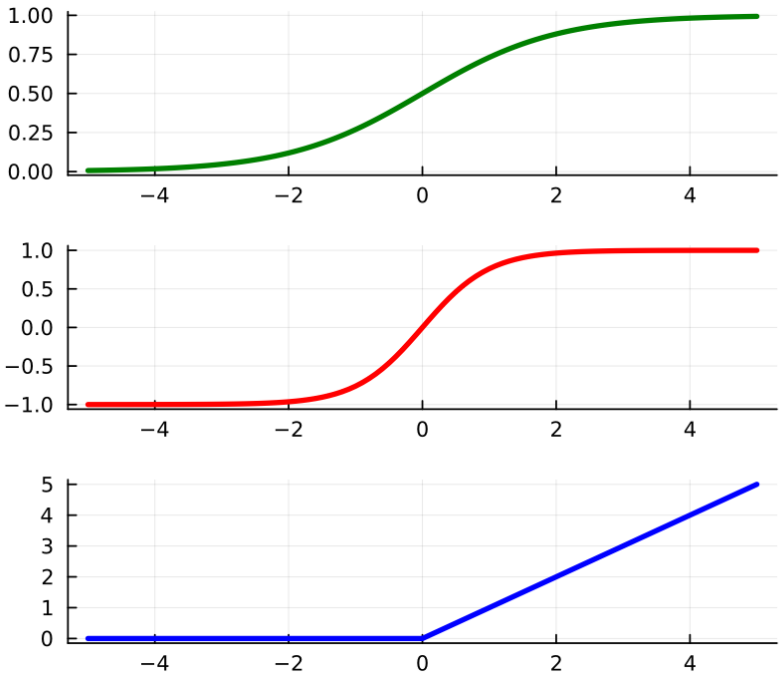
A função $ReLU$, em particular, difundiu-se de maneira significativa em anos recentes e tem encontrando grande uso no campo de *deep learning*. Isso pode ser explicado pelas propriedades do seu gráfico. Para uma rede com neurônios $ReLU$, metade dos neurônios estarão ativos, em geral, para uma dada entrada. A ausência de saturação, presente nas funções sigmoide, significa uma menor probabilidade do problema *vanishing gradient*, que ocorre quando o gradiente é próximo a zero. Esses dois fatores contribuem para um aprendizado mais rápido e, portanto, para um melhor treinamento, especialmente em redes com muitas camadas. Adicionalmente, embora seja uma função não-linear, o cálculo de sua saída é significativamente mais simples, reduzindo os custos computacionais.

4.3.2 Topologia

RNAs são compostas por vários neurônios interconectados dispostos de acordo com uma topologia específica. A Figura 4.5 ilustra o caso de uma rede multicamada *feedforward* totalmente conectada, na qual todos os neurônios de uma camada estão conectados com todos os neurônios da camada seguinte. O termo *feedforward* se refere à forma como informação flui pela rede: da esquerda para direita, sem nenhum *feedback* das camadas seguintes às anteriores. Consequentemente, a saída de uma

[2]Notar que por sua definição, a função $ReLU$ não é diferenciável em $v = 0$. Implementações numéricas devem lidar com esse caso particular, por exemplo, atribuindo à derivada nesse ponto um valor entre 0 e 1.

Figura 4.4 - Funções de ativação comumente usadas.



Gráficos da função logística (verde), tangente hiperbólico (vermelho) e ReLU (azul)
Fonte: Autor.

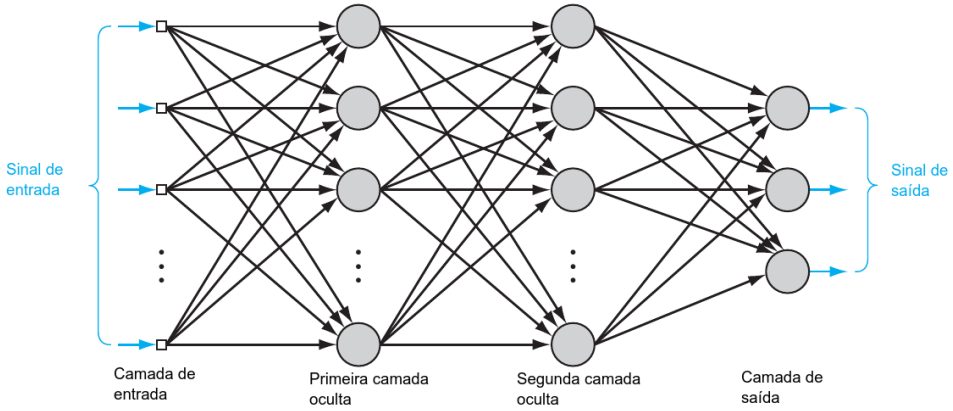
camada é a entrada da seguinte. Esse é o caso de interesse e que será utilizado nas simulações.

Na literatura, ainda é comum encontrar a designação perceptron de múltiplas camadas (MLP, em inglês *Multilayer Perceptron*) para essa arquitetura. Entretanto, o perceptron original utilizava uma função de ativação do tipo degrau e, portanto, era um classificador binário. Muitas outras arquiteturas são possíveis; para processamento de imagens, por exemplo, redes convolutivas, nas quais os neurônios estão ligados apenas à uma fração específica dos neurônios da camada posterior, foram empregadas com sucesso (MNIH et al., 2015).

A computação, à semelhança das estruturas neuronais biológicas, ocorre de maneira paralela e distribuída. Um neurônio apenas possui um mapeamento do tipo $\mathbb{R}^m \to \mathbb{R}$, isso é, um vetor em um escalar. Logo, uma camada, composta de vários neurônios, mapeia um vetor a um outro vetor, não necessariamente de mesma dimensão. Dada uma entrada x na camada de entrada, a função da rede para a Figura 4.5, por exemplo, é:

$$y(x;\theta) = f^4(f^3(f^2(f^1(x)))), \tag{4.43}$$

Figura 4.5 - RNA com arquitetura totalmente conectada e duas camadas ocultas.



Fonte: Adaptado de Haykin (2008).

onde f^1 representa a camada de entrada, f^2 a primeira camada oculta, f^3 a segunda camada oculta e f^4, visto ser a última função, a camada de saída. O parâmetro θ representa todos os pesos e vieses existentes de uma camada a outra. As camadas ocultas, que recebem esse nome por não serem diretamente observáveis, possuem um papel importante na generalização, à medida que mapeiam a entrada da camada anterior a um outro espaço de características (GOODFELLOW et al., 2016).

4.3.3 Treinamento

Na condição de aproximadores de função, busca-se alterar os parâmetros θ da rede de maneira a obter um bom mapeamento entre um conjunto de entradas e saídas desejadas. Mais formalmente, deve-se minimizar uma função custo $J(\theta)$, encontrando um valor θ^* ótimo:

$$\theta^* = \arg\min_{\theta} J(\theta). \tag{4.44}$$

Na maior parte dos casos, a entropia cruzada entre a distribuição dos dados e a distribuição do modelo é adotada como função custo (GOODFELLOW et al., 2016). Entretanto, a abordagem aqui será mais simplificada. Dado um conjunto ζ de N amostras com formato:

$$\zeta = \{x(n), d(n)\}_{n=1}^{N}, \tag{4.45}$$

onde $x(n)$ é o n-ésimo exemplo e $d(n)$ sua saída correspondente (HAYKIN, 2008). Definindo a saída de rede como um vetor $y(x(n); \theta)$, então uma escolha atraente

para a função custo é o erro médio quadrático, isso é:

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} (d_j(n) - y_j(x(n); \theta))^2, \tag{4.46}$$

onde j denota o j-ésimo componente do vetor e C indica todos os neurônios da camada de saída. O problema se torna treinar uma rede de maneira que os parâmetros definam uma boa função de aproximação conforme os exemplos em ζ.

Métodos de descida do gradiente podem ser empregados para minimizar 4.46, o que requer o conhecimento do vetor $\frac{\partial J}{\partial \theta}$. É evidente que todos os componentes distribuídos possuem uma influência no resultado. A questão é como distribuir a culpa a estes parâmetros. O problema aqui é o mesmo de distribuição de crédito do RL, mas agora em uma versão estrutural. Mais especificamente, os parâmetros da camada de saída têm uma influência direta na saída, e seu gradiente pode ser prontamente calculado, mas a estimação do gradiente dos parâmetros das camadas intermediárias não é trivial.

O algoritmo de *backpropagation*, apresentado por Rumelhart et al. (1986), resolve o problema do gradiente de maneira eficiente para as redes do tipo *feedforward*. A ideia central do algoritmo - e a razão de seu nome - é realizar o cálculo em duas etapas: na primeira, os pesos são mantidos constantes, e a rede propaga o sinal da entrada, da esquerda para a direita, até a camada de saída. Na segunda, o sinal de erro é propagado no sentido inverso, da direita para a esquerda, e os termos individuais $\frac{\partial J}{\partial \theta_i}$ são calculados para os parâmetros θ_i. Visto de uma forma mais primitiva, o algoritmo é uma aplicação da regra da cadeia a uma função composta.

Deve-se enfatizar que o algoritmo de *backpropagation* apenas fornece o vetor gradiente $\frac{\partial J}{\partial \theta}$, não definindo uma forma específica de se atualizar os pesos. Estes devem ser modificados por algum método de gradiente estocástico. Um exemplo simples é uma atualização do tipo:

$$\theta \leftarrow \theta - \eta \frac{\partial J}{\partial \theta}. \tag{4.47}$$

Essa atualização é conhecida como *gradient descent*. O valor η é uma pequena constante positiva, chamada de taxa de aprendizado, geralmente entre 0,1 e 0,0001. Em geral, o formato das equações de atualização é significativamente mais complexo que 4.47 e inclui termos e hiperparâmetros adicionais de momento, que ajudam a aumentar a estabilidade e evitar mínimos locais. Um algoritmo eficiente para esta tarefa e que possui esse formato é ADAM (KINGMA; BA, 2017).

4.3.4 Aproximação de função

A breve discussão torna claro que RNAs podem ser vistos como aproximadores de funções, da mesma forma, por exemplo, que um polinômio interpolador. Isso é, independente de sua topologia específica, uma rede pode ser vista como uma caixa preta, mapeando um dado vetor no espaço $\mathbb{R}^m$ para um outro no espaço $\mathbb{R}^n$. O algoritmo de *backpropagation* fornece uma forma geral do ajuste dos pesos internos de forma a minimizar o erro. Uma pergunta mais geral, todavia, é o quão exata essa aproximação pode ser, em outras palavras, se a rede é capaz de aproximar uma função desejada com erro tão pequeno quanto se queira. Alguns resultados teóricos fornecem uma resposta positiva a essa pergunta.

Hornik et al. (1989) estabeleceram que redes multicamadas do tipo *feedforward* com ao menos um camada oculta são capazes de aproximar uma função Borel mensurável a qualquer nível arbitrário de precisão, contanto que um número suficiente de neurônios ocultos esteja disponível. Esse resultado impunha algumas restrições sobre a função de ativação, que tinha o formato o sigmoide. Hornik (1991) expandiu essas restrições a uma grande variedade de funções de ativação. Esse resultado mostra que a capacidade de representação de RNAs não está associada a escolha das funções de ativação em si, mas da presença das camadas ocultas.

É importante observar que esses teoremas apenas realizam afirmações a cerca da existência de redes com tais propriedades. Isso é, eles não são teoremas construtivos, visto que provam a existência, mas não conferem uma abordagem geral para a construção da rede. Ademais, a propriedade de aproximação é válida apenas dentro de uma região específica; a extrapolação para fora dessa região não está assegurada pelos resultados acima. As consequências práticas desses resultados, contudo, são extremamente poderosas, à medida que garantem a capacidade de RNAs em representar com grande exatidão uma vasta quantidade de funções de interesse.

5 METODOLOGIA

No presente capítulo, a forma como as simulações computacionais foram realizadas e algumas considerações adicionais de aspecto mais prático são apresentadas.

5.1 Ambiente de simulação

A linguagem computacional Julia foi escolhida para realizar as simulações da propagação de atitude do satélite e implementar os modernos algoritmos de RL anteriormente discutidos. Possuindo uma compilação *just-in-time* e baseada no paradigma de *multiple dispatch*[1], Julia é uma linguagem de alto nível e elevado desempenho, especialmente adequada para problemas científicos e de análise numérica.

As implementações dos algoritmos foram realizadas a partir do código contido no pacote *ReinforcementLearning*, um pacote voltado para o RL e integralmente escrito em Julia. Seus dois principais tipos abstratos, nos quais os experimentos de RL se baseam, são *AbstractPolicy* e *AbstractEnv* (TIAN; CONTRIBUTORS, 2020).

A abstração *AbstractPolicy* refere-se a política e retorna ações de acordo com o estado do ambiente. Dentre os seus tipos concretos, estão *DDPGPolicy*, *TD3Policy* e *SACPolicy*, correspondente aos algoritmos de interesse. Para a interação com o ambiente e atualização de seus parâmetros a partir da experiência, um outro tipo concreto chamado *Agent* está disponível. Esse tipo é composto de dois campos: *AbstractPolicy* e *AbstractTrajectory*, que acumula experiências pelo replay de memória. Dessa forma, selecionando aleatoriamente amostras do replay, a atualização da política é efetuada seguindo a lógica de cada algoritmo.

Por sua vez, a abstração *AbstractEnv* diz respeito ao ambiente do experimento. Seus tipos concretos devem ser implementados pelo usuário, seguindo a lógica particular do ambiente em questão. Embora nenhuma imposição seja feita em relação a sua dinâmica ou os campos de sua estrutura, a interface com o restante do pacote exige que os seguintes valores sejam definidos: estado do ambiente (geralmente um vetor), recompensa recebida, espaço de estados, espaço de ação e uma variável booleana que indique se o episódio acabou ou não.

Durante a interação com o ambiente, o agente muda continuamente seus parâmetros em busca de maiores retornos. De maneira a observar essa evolução, o pacote

[1] Paradigma no qual uma mesma função pode ter várias implementações diferentes, de acordo com os tipos dos argumentos que lhe são passados. Ao ser chamada, a implementação mais específica que corresponda aos tipos é executada.

também dispõe da abstração *AbstractHook*, que permite ao usuário registrar alguma informação de interesse, como retorno médio ou número de passos por episódio do agente. Um tipo concreto pode ser chamado tanto antes como depois de uma passo ou de um episódio.

Definindo-se uma condição de parada, tal como número de passos ou de episódios, então a simulação é executada chamando a função *run* e passando como argumentos o agente, o ambiente, a condição e alguma variável opcional do tipo *AbstractHook*.

Alguns outros pacotes merecem ser destacados. Para todos algoritmos, o ator e o crítico foram definidos como redes neurais, o que foi possível por meio do pacote *Flux*. Em muitas das expressões desenvolvidas, a otimização exigia o conhecimento de gradientes de certos parâmetros, isso é, de derivadas parciais. Na prática, isso é realizado pelo *Zygote*, um pacote de diferenciação automática extremamente sofisticado, capaz de retornar o valor numérico das derivadas ao analisar o funcionamento do código.

Uma discussão mais detalhada sobre os pacotes empregados, observações pessoais e dificuldades enfrentadas encontra-se no Anexo C.

5.2 Amazonia-1

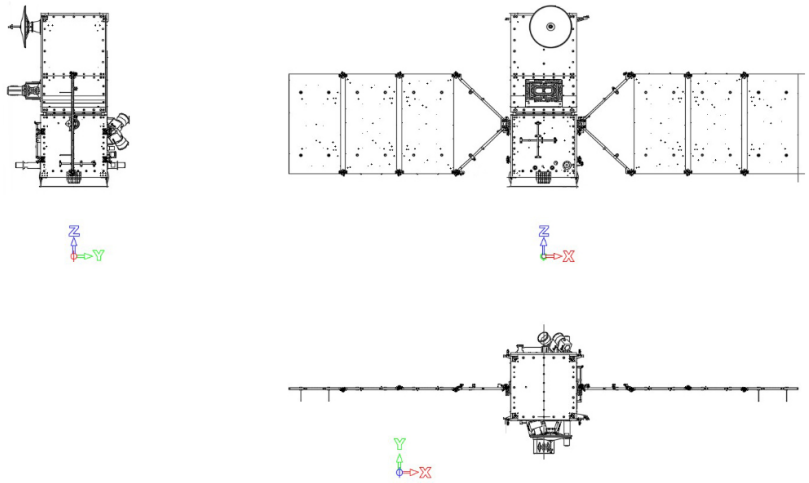
De modo a selecionar um ambiente E representativo para o emprego das ferramentas do moderno RL em aplicações espaciais, os valores reais do satélite Amazonia-1 - lançado em 2021 e o primeiro satélite de observação terrestre a ser completamente projetado, testado, integrado e operado pelo Brasil - foram selecionados. Projetado pelo Instituto Nacional de Pesquisas Espaciais (INPE), também possui a distinção de ser o primeiro satélite totalmente brasileiro estabilizado em três eixos, isso é, seu controle de atitude é completo no espaço (Instituto Nacional de Pesquisas Espaciais, 2021). A Figura 5.1 fornece vistas ortográficas do satélite.

Com uma massa de aproximadamente 640 kg, seu momento de inércia em relação ao seu centroide é, repetindo o apresentado na Equação 5.1, dado por:

$$\boldsymbol{I_{Amaz}} = \begin{bmatrix} 310,0 & 1,11 & 1,01 \\ 1,11 & 360,0 & -0,35 \\ 1,01 & -0,35 & 530,7 \end{bmatrix} kg\ m^2. \tag{5.1}$$

O controle de atitude é implementado pelo uso de rodas de reação, com o emprego de magnetômetros para a eliminação de momento angular excessivo do sistema satélite-

Figura 5.1 - Vistas do satélite Amazonia-1.



Direção de órbita corresponde a +Z; -Y fica apontado para a Terra.
Fonte: Instituto Nacional de Pesquisas Espaciais (2021).

rodas. Mais especificamente, 4 rodas são empregadas de acordo com a configuração NASA *Standard*, na qual 3 rodas são dispostas ao longo dos três eixos do satélite e um quarta roda é disposta equidistante das demais, apontando para a diagonal do cubo (MARKLEY; CRASSIDIS, 2014). A Figura 5.2 ilustra esse disposição.

Desse modo, a matriz $\boldsymbol{A}$ é:

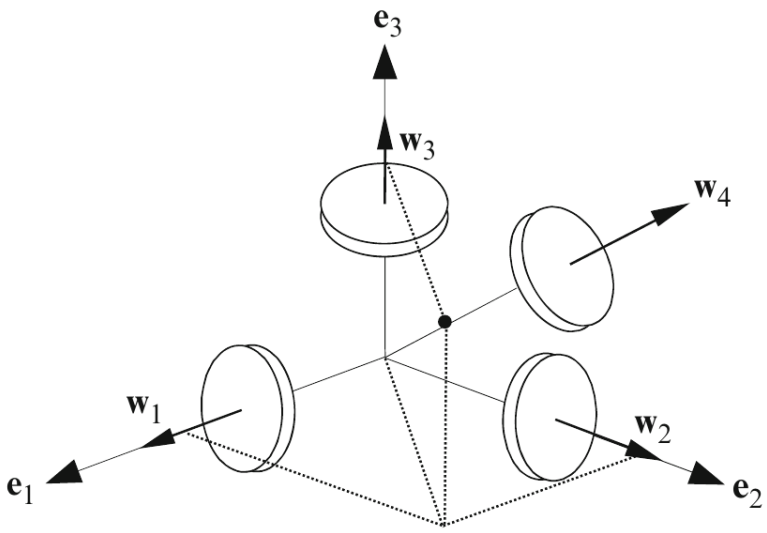
$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & \frac{\sqrt{3}}{3} \\ 0 & 1 & 0 & \frac{\sqrt{3}}{3} \\ 0 & 0 & 1 & \frac{\sqrt{3}}{3} \end{bmatrix}. \tag{5.2}$$

Logo, sua pseudoinversa $\boldsymbol{A}^{+}$ é:

$$\boldsymbol{A}^{+} = \begin{bmatrix} \frac{5}{6} & -\frac{1}{6} & -\frac{1}{6} \\ -\frac{1}{6} & \frac{5}{6} & -\frac{1}{6} \\ -\frac{1}{6} & -\frac{1}{6} & \frac{5}{6} \\ \frac{\sqrt{3}}{6} & \frac{\sqrt{3}}{6} & \frac{\sqrt{3}}{6} \end{bmatrix}. \tag{5.3}$$

O torque aplicado ao longo dos três eixos do satélite é governado por um lei de controle do tipo PD. No referencial do satélite, o torque $\boldsymbol{T_{rw}}$ aplicado às rodas é dado por:

Figura 5.2 - Configuração NASA Standard.



Fonte: Markley e Crassidis (2014).

$$\boldsymbol{T_{rw}}` = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} 0,6253 & 0 & 0 \\ 0 & 0,6748 & 0 \\ 0 & 0 & 1,019 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} + \begin{bmatrix} 25,95 & 0 & 0 \\ 0 & 28,03 & 0 \\ 0 & 0 & 42,21 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{5.4}$$

O torque dado pela Equação 5.4 é traduzido para o torque de cada roda T_1, T_2, T_3 e T_4 por meio da pseudoinversa. As 4 rodas possuem as mesmas características e seu torque máximo está limitado a 0.075 Nm. Se o torque comandado a uma roda em particular foi superior a esse valor, fala-se em saturação de seu torque.

O controle PD do satélite fornece uma referência básica para comparação do desempenho obtido pelos métodos de RL. As duas matrizes na Equação 5.4, correspondentes aos ganhos $\boldsymbol{k_p}$ e $\boldsymbol{k_d}$ na Equação 3.54, foram cuidadosamente selecionados em um longo processo iterativo de tentativa e erro. Eles estão muito próximos de um controle ótimo para condições nominais. Contudo, em situações adversas, como falhas de rodas, é de se esperar que o desempenho desse controle se deteriore de maneira significativa, em razão das condições particulares para as quais seus parâmetros foram ajustados.

5.3 Inicialização

Ao se estabelecer uma definição formal do problema de RL, devida atenção foi dada a como os estados são inicializados seguindo uma função densidade de probabilidade ρ. Em muitos problemas práticos, é de interesse expor o agente a um conjunto de estados diferentes, visto que expô-lo a apenas uma fração dos estados pode comprometer sua capacidade de generalização e portanto o desempenho final. Para o problema de controle de atitude, por exemplo, o agente deve ser capaz de controlar o satélite dada uma orientação inicial qualquer. Dessa forma, garantir uma orientação inicial suficiente aleatória possui grande importância durante o treinamento. Sendo a atitude representada pelo quatérnion, então o objetivo é selecionar de maneira uniforme um quatérnion $\boldsymbol{q}$.

A restrição da norma do quatérnion implica que os componentes de $\boldsymbol{q}$ devem satisfazer $\eta^2 + \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2$. Logo, seus elementos podem ser visualizados em $\mathbb{R}^4$ como pontos de uma hiperesfera. Realizar uma distribuição uniforme equivale, de maneira informal, a garantir que todas as hipersuperfícies dessa esfera com mesmo volume tenham a mesma densidade de pontos selecionados.

Uma forma eficiente de garantir essa distribuição é apresentada por Shoemake (1992) e reproduzida aqui. Selecionando três números u_1, u_2 e u_3 uniformemente distribuídos em $[0, 1]$, então um quatérnion pode ser gerado por:

$$\eta = \sqrt{1 - u_1}\, sen(2\pi u_2), \tag{5.5}$$
$$\varepsilon_1 = \sqrt{1 - u_1}\, cos(2\pi u_2), \tag{5.6}$$
$$\varepsilon_2 = \sqrt{u_1}\, sen(2\pi u_3), \tag{5.7}$$
$$\varepsilon_3 = \sqrt{u_1}\, cos(2\pi u_3), \tag{5.8}$$

cuja norma é unitária, por definição.

O eixo da velocidade angular também deve ser selecionado de maneira a satisfazer essa propriedade de uniformidade. Nesse caso, o objetivo é garantir a mesma densidade de pontos na superfície de uma esfera, isso é, garantir que superfícies de mesma área tenham a mesma probabilidade de serem selecionadas.

Selecionar aleatoriamente cada componente de um vetor para então realizar sua

normalização produz uma distribuição distorcida na diagonal do cubo. Igualmente, selecionar uniformemente os ângulos polares θ e ϕ distorce a distribuição nos polos. A forma correta é selecionar u uniformemente em $[-1, 1]$ e θ em $[0, 2\,\pi]$ e encontrar os componentes por:

$$x = \sqrt{1-u^2}cos(\theta), \tag{5.9}$$

$$y = \sqrt{1-u^2}sen(\theta), \tag{5.10}$$

$$z = u. \tag{5.11}$$

Esse método produz a distribuição correta, uma vez que em coordenadas cilíndricas o elemento de área infinitesimal de uma esfera de raio unitário é $dA = d\theta dz$. De posse desse vetor unitário, sua norma pode ser multiplicada pelo módulo ω de interesse.

6 RESULTADOS

Esse capítulo descreve os resultados obtidos para o problema de controle de atitude de satélites por meio de RL, usando a implementação dos algoritmos na linguagem Julia. A Seção 6.1 apresenta o caso simples de controle em torno de único eixo. A simplicidade desse caso permite apresentar e discutir ideias fundamentais do RL. A Seção 6.2 expande essa aplicação para o controle em três dimensões, assumindo torques externos. A Seção 6.3 considera o problema com as rodas de reação inclusas. Finalmente, as Seções 6.4 e 6.5 abordam o problema de controle de atitude de um satélite cujos parâmetros são previamente desconhecidos pelo agente, isso é, com momentos de inércia variáveis, primeiro ao problema unidimensional e depois para o caso em três dimensões.

6.1 Controle em um único eixo

De forma a adquirir experiência com o problema e testar a abordagem inicial, o controle de atitude foi simulado inicialmente em torno de um único eixo. O problema torna-se significativamente mais simples e a visualização dos resultados é mais intuitiva. A atitude pode ser então simplesmente descrita pelo ângulo polar θ e a velocidade angular por $\dot{\theta}$. Realizando a discretização dessas variáveis com um passo temporal Δt, as equações da cinemática são então dadas por:

$$\theta(k+1) = \theta(k) + \dot{\theta}(k)\Delta t + \frac{1}{2}\ddot{\theta}(k)\Delta t^2, \tag{6.1}$$

$$\dot{\theta}(k+1) = \dot{\theta}(k) + \ddot{\theta}(k)\Delta t, \tag{6.2}$$

com k representando o k-ésimo passo temporal. A aceleração angular é encontrada pela equação dinâmica $\ddot{\theta}(k) = T(k)/I$, com $T(k)$ sendo o torque aplicado e I o momento polar de inércia. De maneira a selecionar valores representativos, o eixo z do satélite Amazonia-1 foi escolhido, com uma inércia de $I_z = 530,7\ kg\ m^2$.

Seguindo a definição do RL exposta na Seção 4.1.1, então precisamos definir os termos da tupla $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$. Uma escolha natural para o estado é $s_t = [\theta(t)\ \dot{\theta}(t)]^T$. Entretanto, de forma a ser coerente com a representação de atitude por quatérnions adotada pelo Amazonia-1, o estado será definido como $s_t = [sen(\theta(t)/2)\ \dot{\theta}(t)]^T$. O ângulo θ será representado no intervalo $[-\pi, \pi]\, rad$.

A ação corresponde ao torque aplicado pelo agente, que no caso em questão

pode ser representado como um escalar. O valor do torque máximo para cada roda no Amazonia-1 é $0,075\ Nm$. Logo a ação deve estar contida no intervalo $[-0,075\ ,\ 0,075]Nm$. Por simplificação, a rede neural do ator possui uma saída entre $[-1,1]$ e esse valor é então multiplicado por $0,075$ para a obtenção do torque real.

O ambiente P é determinístico e é descrito pelas Equações 6.1 e 6.2.

A recompensa $r(s_t, a_t)$ deve guiar o agente rumo ao objetivo de trazer o satélite ao repouso em $\theta = 0$. Não há uma escolha única para esse fim e várias funções são capazes de transmitir essa mensagem. Por comodidade, a recompensa r será simplesmente função do ângulo θ, mais especificamente:

$$r(s_t, a_t) = r(s_t) = -\frac{|\theta|}{\pi}. \tag{6.3}$$

Dessa forma, a recompensa é sempre negativa, está no intervalo $[-1,0]$ e possui um valor máximo em $\theta = 0$. Notar que a velocidade angular não está diretamente inclusa no formato da recompensa. Ainda assim, implicitamente o agente deve concluir que a velocidade angular também deve ser zerada. Ademais, a recompensa não incorpora nenhum termo envolvendo a ação selecionada. A inclusão de um termo que punisse ações poderia levar ao controle do tipo *bang-off-bang* observado no controle ótimo, com tempos de assentamento elevados.

Para assegurar que o agente aprenda a controlar o satélite satisfatoriamente, ele deve ser exposto a uma variedade de estados. Dessa forma, a distribuição inicial de estados ρ_0 possui um papel importante no resultado final. O ângulo inicial é selecionado aleatoriamente de uma distribuição uniforme contínua com suporte em $[-\pi, \pi]$. A velocidade angular inicial também segue a mesma distribuição, mas com suporte em $[-0,025\ ,\ 0,025]\, rad/s$, que é um intervalo considerado razoável para as velocidades que o agente deve encontrar em uma aplicação real.

Os episódios são inicializados conforme o esquema acima e prosseguem até que uma de duas condições de parada seja satisfeita. A primeira é se a norma do vetor estado for inferior a 0,0001, ou um ângulo menor que 1 centésimo de grau. A segunda é se o número de passos na simulação alcançar 4000. Sendo o passo adotado $\Delta t = 1\, s$, então isso equivale a 1 hora, 6 minutos e 40 segundos no ambiente do satélite. A taxa de desconto escolhida foi $\gamma = 0{,}99$.

O ator e o crítico foram parametrizados por redes neurais do tipo *feedforward* total-

mente conectadas. A configuração a ser escolhida, em termos de número de camadas, neurônios por camada e função de ativação, também é de certo modo arbitrária. A rede deve possuir um número de parâmetros suficiente para representar a política ótima π^*, isso é, ser capaz de realizar o mapeamento ótimo entre estados e ações, no caso do ator, e ser capaz de representar o valor de ação verdadeiro $Q_{\pi^*}(s, a)$ da forma mais próxima possível, no caso do crítico.

Infelizmente, a forma analítica da política π^* é desconhecida. Mesmo se a conhecêssemos, conforme a discussão da Seção 4.3.4, não teríamos resultados gerais para estimar o número de neurônios necessários para sua aproximação. Assim esses valores devem ser escolhidos a partir de um julgamento um tanto quanto subjetivo. Uma vez que para todos os algoritmos considerados a atualização da política depende profundamente do conhecimento do valor de ação, tomou-se o cuidado de atribuir mais parâmetros ao crítico que ao ator.

Seguindo a observação de Marques (2021), o viés não deve ser incluso na rede neural do ator, de maneira a garantir que a ação seja nula quando o estado for nulo, isso é, $f(x) = 0$ quando $x = 0$. Adicionalmente, uma propriedade do controlador PD é ser uma função com simetria ímpar no estado, isso é, $f(x) = -f(-x)$. Isso é uma consequência da simetria no espaço: o torque deve mudar de sinal quando θ e $\dot{\theta}$ mudam de sinal.

O agente deve incorporar esse aspecto; uma possibilidade para fazê-lo é usar funções de ativação que possuam a simetria ímpar. Desse modo, a função de ativação tangente hiperbólico foi adotada para a rede do ator, visto que $tanh(x) = -tanh(-x)$, o que permite conferir conhecimento prévio ao agente da solução do problema e assim acelerar o aprendizado. Sendo $tanh(x) \in [-1, 1] \, \forall x \in \mathbb{R}$, então a saída também estará limitada, o que está de acordo com as limitações do atuador.

A Tabela 6.1 apresenta os hiperparâmetros[1] comuns às simulações. O ator, com 32 neurônios por camada, 2 camadas ocultas e sem o termo de viés, possui 1120 parâmetros. O crítico, com 128 neurônios por camadas e viés, possui 17153 parâmetros, ou quase 16 vezes mais que o ator. Dessa forma, espera-se que o valor de ação estimado pelo crítico seja exato o bastante para garantir atualizações do ator rumo a maiores retornos. Notar que como o número de passos total é igual a 10 milhões e o intervalo de atualização é de 10 passos, então o número total de atualizações do

[1] No contexto de *machine learning*, o termo hiperpârametro se refere a uma variável escolhida de antemão cujo valor não se altera durante o treinamento. Em contrapartida, parâmetros descrevem valores que são ajustados durante esse processo, como os pesos de uma rede neural.

crítico é ligeiramente inferior a 1 milhão.

Tabela 6.1 - Hiperparâmetros usados na simulação.

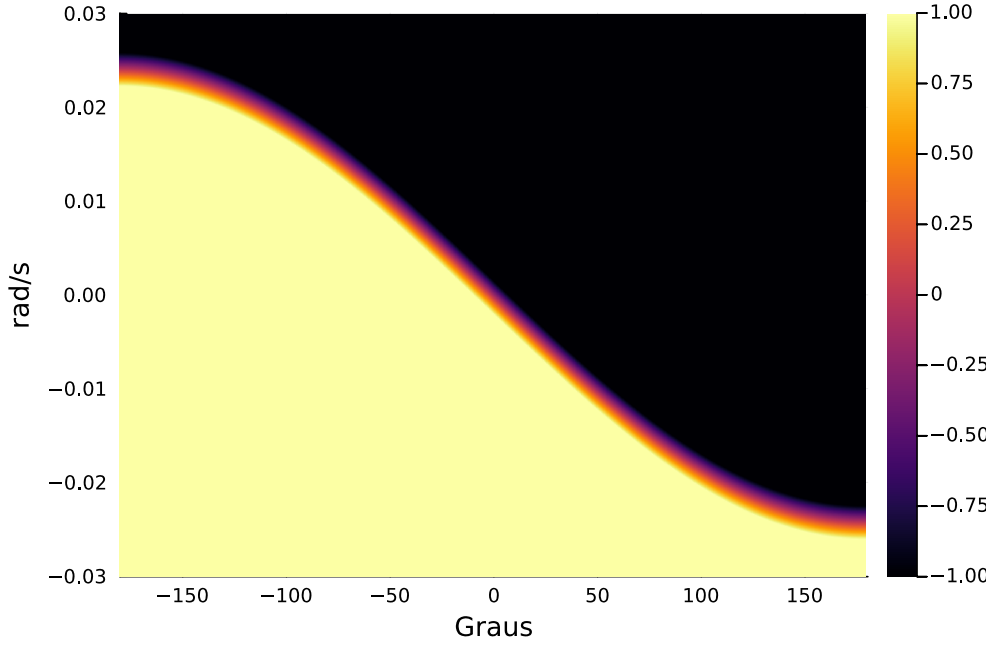
Hiperparâmetro	**Valor**
Taxa de desconto	0,99
Camadas ocultas (ator)	2
Neurônios por camada (ator)	32
Função de ativação (ator)	tanh
Camadas ocultas (crítico)	2
Neurônios por camada (crítico)	128
Função de ativação (crítico)	ReLU
Tamanho do replay	10000
Tamanho do minibatch	32
Número de passos	10^7
Passos entre atualizações	10
Otimizador	ADAM
Taxa de aprendizado	0,001

Devida atenção também deve ser dada ao tamanho do replay de memória. Como discutido anteriormente, a função do replay é reaproveitar experiências passadas. As constantes atualizações da rede do ator levam a mudanças na política, de forma que as interações contidas no replay são invariavelmente resultado de políticas anteriores, o que faz todos os algoritmos serem do tipo *off-policy*. Se por um lado esse aspecto favorece a exploração e a capacidade de generalização do agente, por outro lado pode levar a instabilidade, à medida que o agente relembra experiências que não estão de acordo com a política atual. Essa discussão qualitativa busca realçar o seguinte ponto: o replay não deve ser curto o bastante para comprometer a velocidade de aprendizado, nem longo o bastante para trazer divergências.

De forma a fornecer uma referência, a Figura 6.1 apresenta o torque comando pelo controlador PD do satélite Amazonia-1 em torno do eixo z, em função do ângulo e da velocidade angular. O mapa de calor foi normalizado; logo os valores estão no intervalo $[-1, 1]$, correspondentes aos limites de atuação de $\pm 0,075\, Nm$.

Como pode ser visto facilmente, as duas regiões de saturação na diagonal superior e inferior são separadas por uma fina região não saturada, onde o torque é dado de acordo com a lei proporcional-derivativa. O formato senoidal dessa região deve-se ao uso do componente $sen(\theta/2)$ do quatérnion para a parte proporcional do controlador.

Figura 6.1 - Fração do torque nominal comandado pelo controlador PD do Amazonia-1, eixo z.

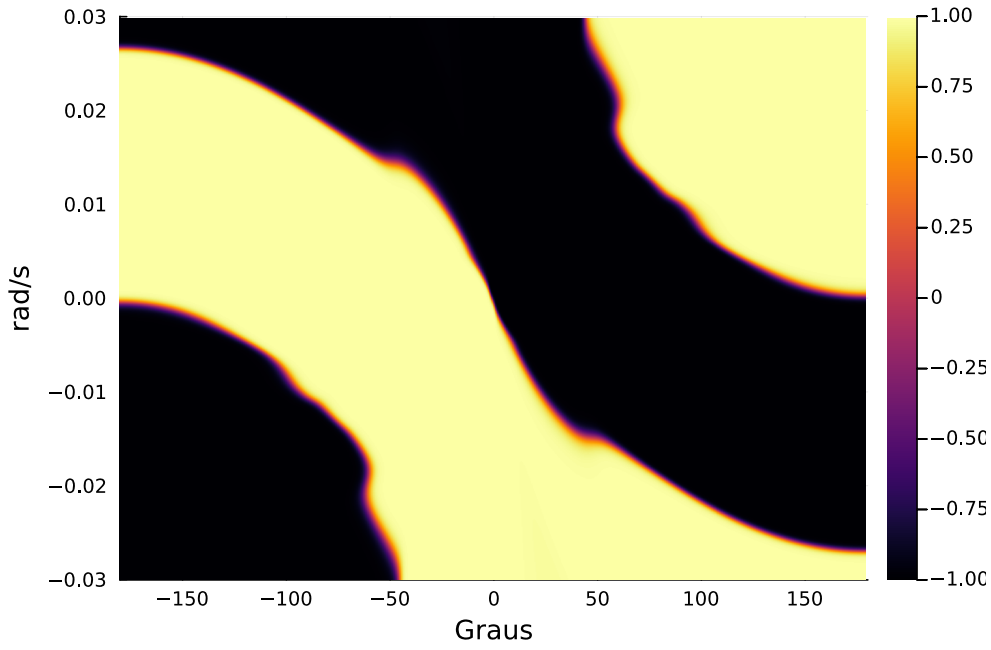


Fonte: Autor.

A Figura 6.2 apresenta o mapa de calor do torque obtido para o DDPG, a Figura 6.3 o do TD3 e a Figura 6.4 o do SAC, todos obtidos conforme a política determinística do algoritmo em questão. O uso da função *tanh* no ator faz com que todos esses gráficos apresentem uma simetria ímpar.

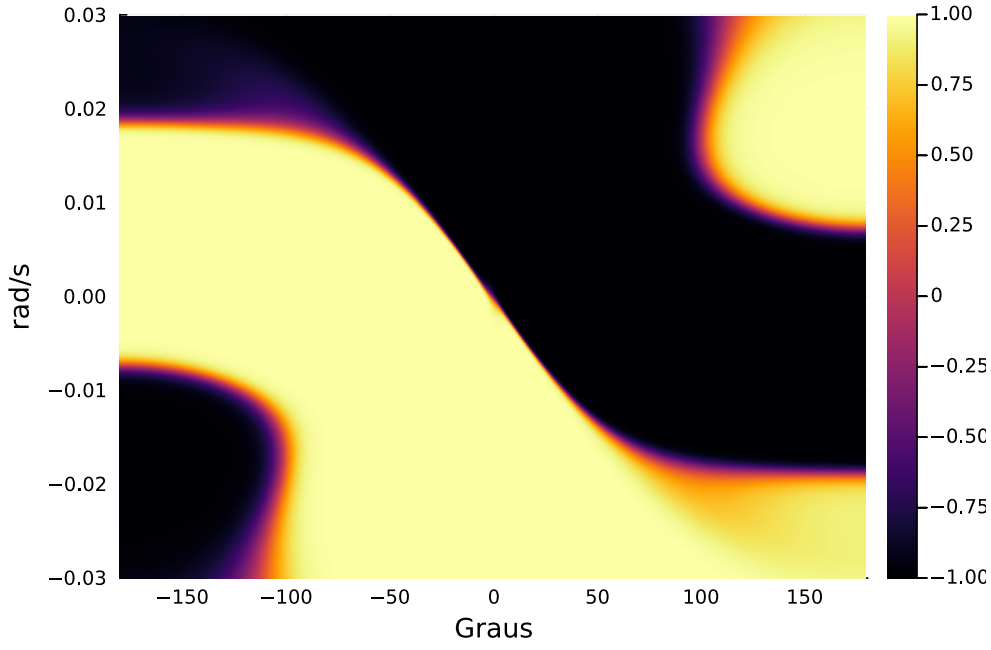
Os gráficos do DDPG e TD3 possuem um formato similar. As duas regiões de saturação estão presentes como no PD, mas são acrescidas agora de duas outras regiões de sinal oposto no canto superior esquerdo e inferior direito dos gráficos. Essas regiões correspondem a ângulos e velocidades altas e de mesmo sinal, de tal modo que o ângulo do satélite cresce inicialmente com o tempo. O controlador PD busca simplesmente desacelerar o veículo e portanto aplica um torque de sinal contrário ao movimento. O agente, contudo, concluiu que essa velocidade angular é benéfica pela dinâmica do ambiente - afinal, $2\pi\,rad = 0\,rad$ - e pode ser utilizada para alcançar a origem mais rapidamente. Assim, por exemplo, se tanto o ângulo como a velocidade angular forem positivos, o satélite chegará à origem pelo sentido horário ($\dot{\theta} < 0$) por meio do PD e pelo sentido anti-horário ($\dot{\theta} > 0$) com o DDPG e o TD3. A presença dessas regiões ilustra a capacidade dos algoritmos de RL em encontrar soluções não triviais em problemas de controle.

Figura 6.2 - Fração do torque nominal comandado pelo ator do DDPG.



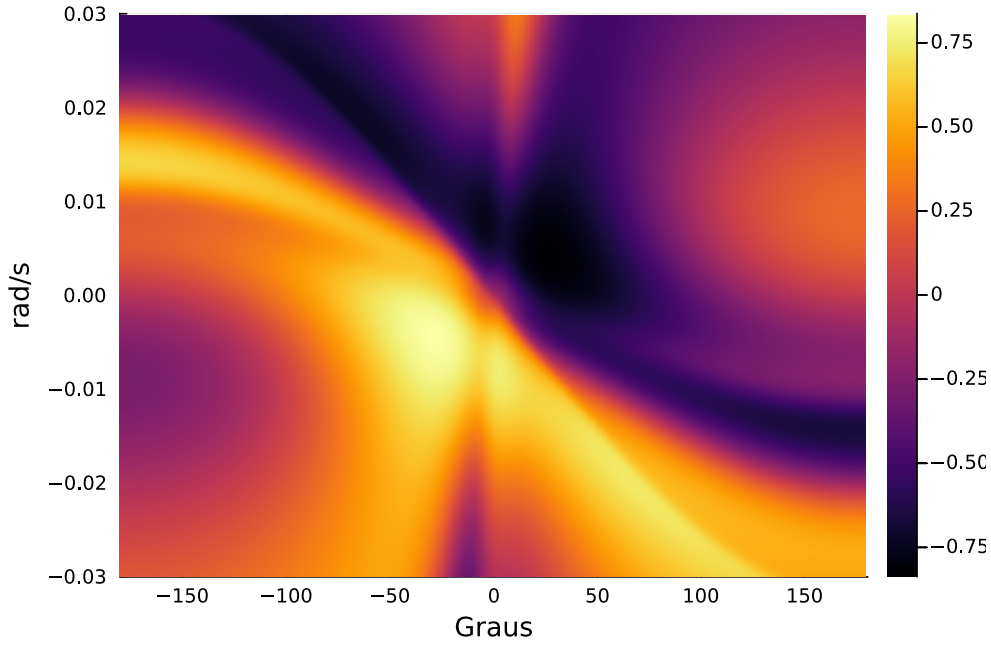
Fonte: Autor.

Figura 6.3 - Fração do torque nominal comandado pelo ator do TD3.



Fonte: Autor.

Figura 6.4 - Fração do torque nominal comandado pelo ator do SAC.



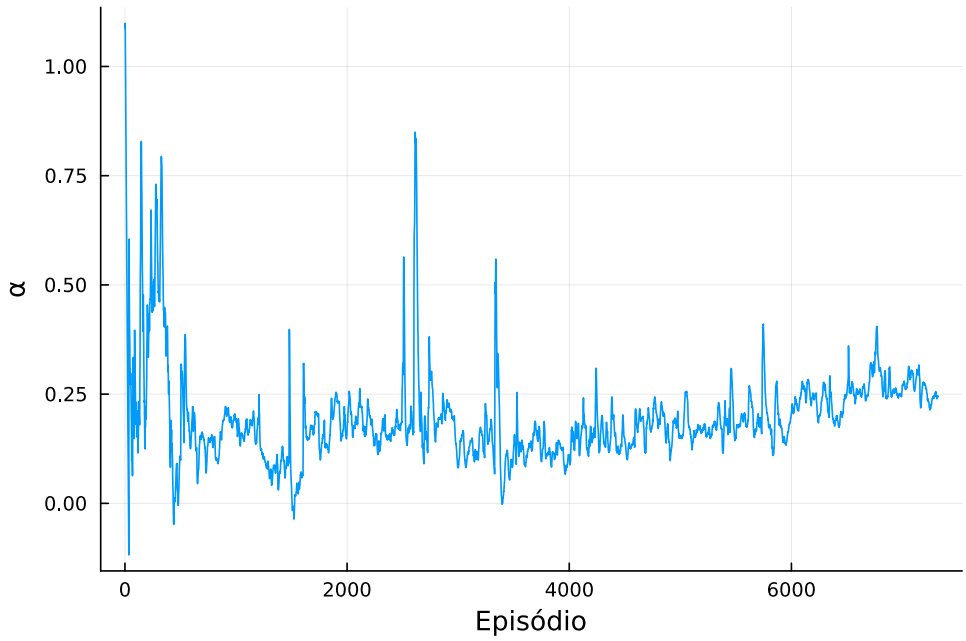
Fonte: Autor.

O SAC em particular, visto na Figura 6.4, possui em sua formulação um mecanismo para o ajuste automático da temperatura α, que pondera a importância relativa da entropia da política e incentiva o agente a explorar. A Figura 6.5 apresenta a variação da temperatura ao longo dos episódios: seu valor inicial de 1,0 decresce rapidamente até estabilizar-se em um valor próximo de 0,25. Uma consequência dessa temperatura é que o agente não possui o incentivo para saturar sua ação na maior parte dos estados, como nos outros dois algoritmos. Ainda assim, o aspecto discutido acima pode ser visualizado nas mesmas regiões inferior direita e superior esquerda do gráfico.

Apesar da solidez desse ajuste automático para uma grande variedade de problemas, o ambiente em questão pode trazer maior retorno em uma dada temperatura específica. Menores temperaturas levam a políticas mais determinísticas e favorecem o retorno $E[G_0]$ frente à entropia da política. Assim, uma segunda simulação para o SAC foi feita usando-se um valor fixo $\alpha = 0,01$. A Figura 6.6 apresenta o resultado para esse caso.

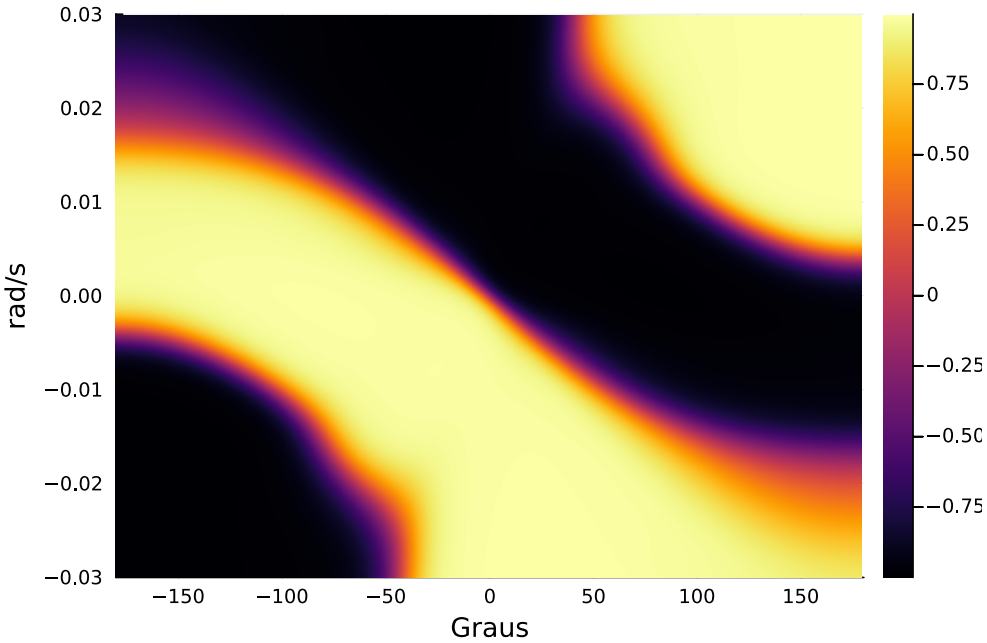
De forma a comparar os agentes, a Figura 6.7 fornece o ângulo e o torque aplicado pelo controlador PD, partindo da condição inicial de 60° e velocidade angular nula.

Figura 6.5 - Variação da temperatura α ao longo dos episódios.



Fonte: Autor.

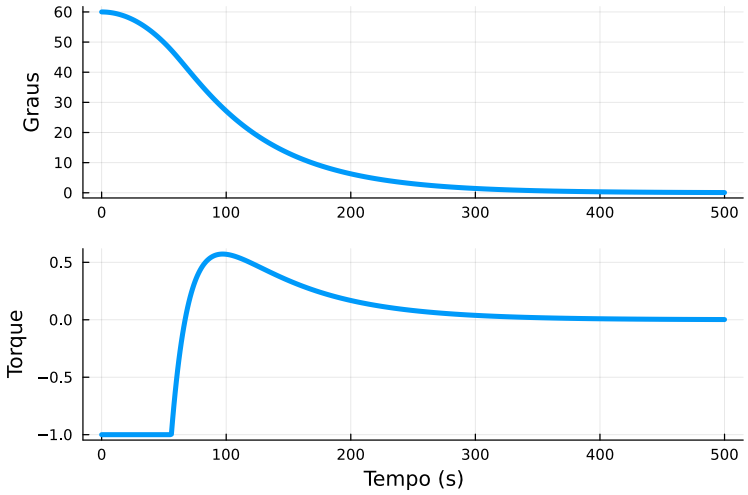
Figura 6.6 - Fração do torque nominal comandado pelo ator do SAC ($\alpha = 0,01$).



Fonte: Autor.

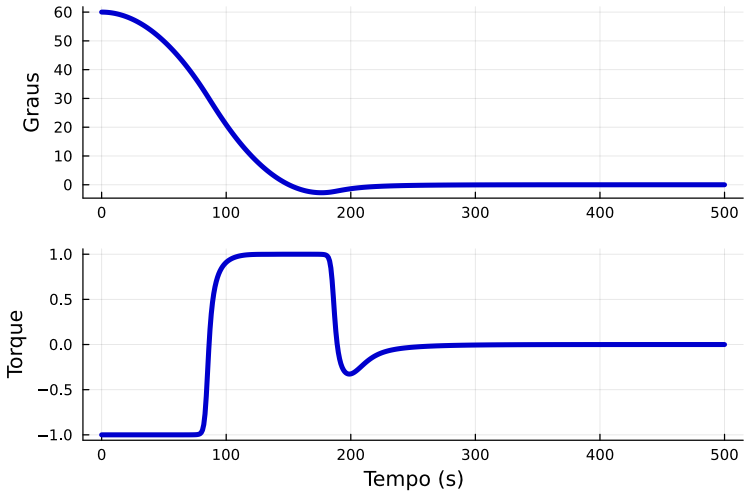
As Figuras 6.8, 6.9 e 6.10 apresentam a mesma informação para o DDPG, TD3 e SAC (a temperatura fixa). O desempenho desses algoritmos é bastante similar ao PD e todos trazem o satélite ao repouso, com ângulo nulo. Adicionalmente, todos possuem uma curva de torque suave, sem variações bruscas.

Figura 6.7 - Controle PD, condições iniciais $\theta = 60°$ e $\omega = 0,00\,rad/s$.



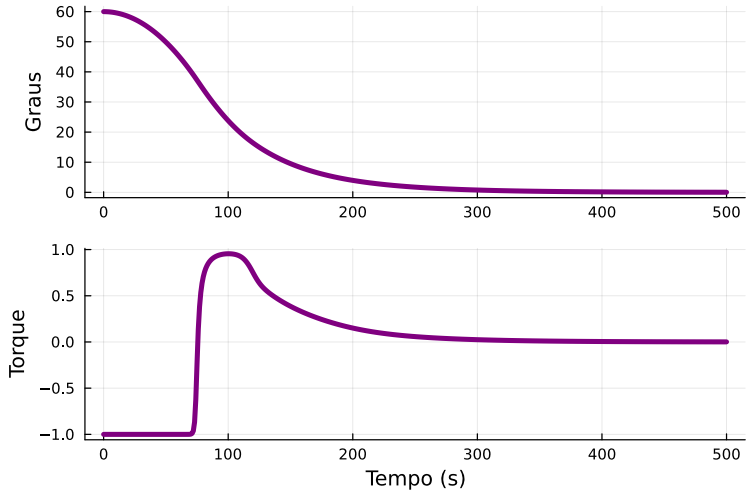
Fonte: Autor.

Figura 6.8 - Controle DDPG, condições iniciais $\theta = 60°$ e $\omega = 0,00\,rad/s$.



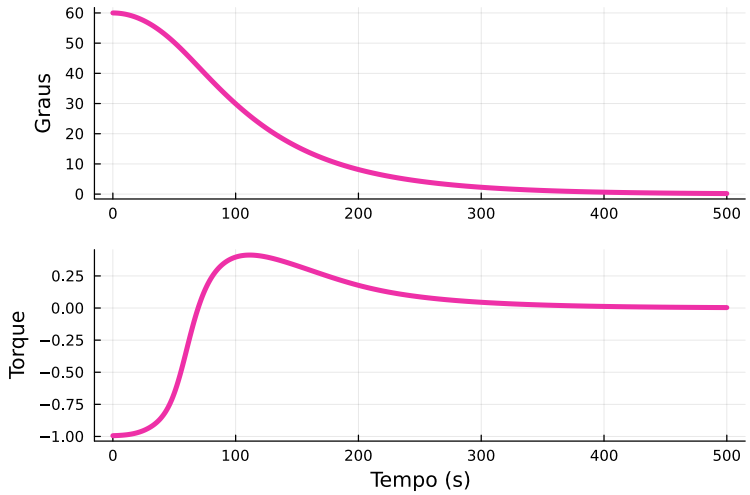
Fonte: Autor.

Figura 6.9 - Controle TD3, condições iniciais $\theta = 60°$ e $\omega = 0,00\,rad/s$.



Fonte: Autor.

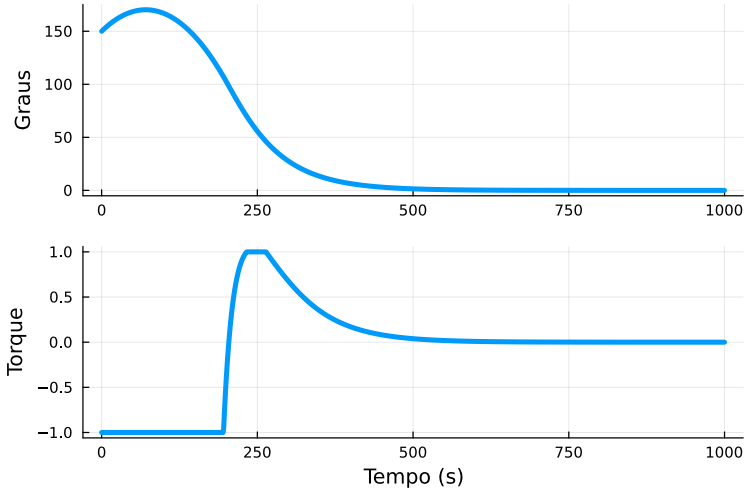
Figura 6.10 - Controle SAC ($\alpha = 0,01$), condições iniciais $\theta = 60°$ e $\omega = 0,00\,rad/s$.



Fonte: Autor.

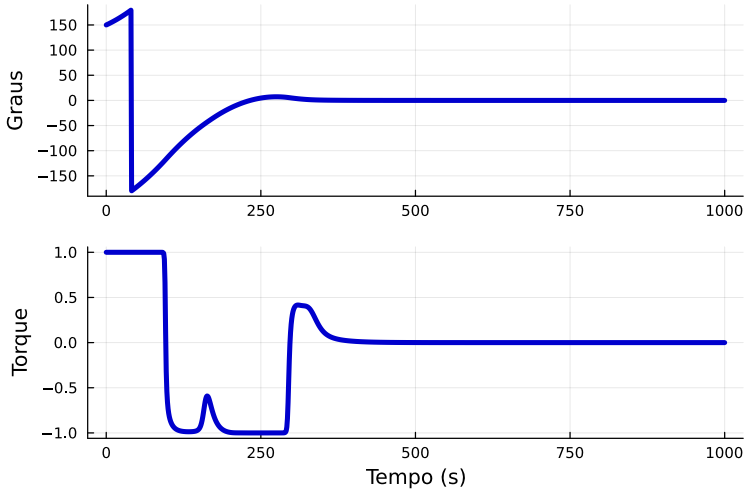
Um cenário diferente é exposto na Figura 6.11, com ângulo inicial $\theta = 150°$ e velocidade angular de 0,01 rad/s. Conforme debatido anteriormente, o PD simplesmente aplicará torque máximo no sentido oposto ao do movimento. Por outro lado, como ilustrado nas Figuras 6.12, 6.13 e 6.14 para o DDPG, TD3 e SAC($\alpha = 0,01$) respectivamente, os agentes aprenderam que essa velocidade inicial era benéfica e, portanto, são inicialmente acelerados e alcançam o repouso mais rapidamente pelo sentido anti-horário.

Figura 6.11 - Controle PD, condições iniciais $\theta = 150°$ e $\omega = 0,01 rad/s$.



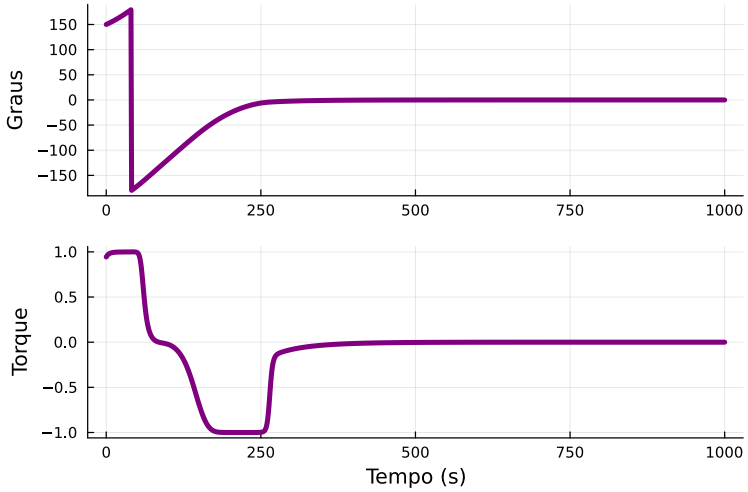
Fonte: Autor.

Figura 6.12 - Controle DDPG, condições iniciais $\theta = 150°$ e $\omega = 0,01 rad/s$.



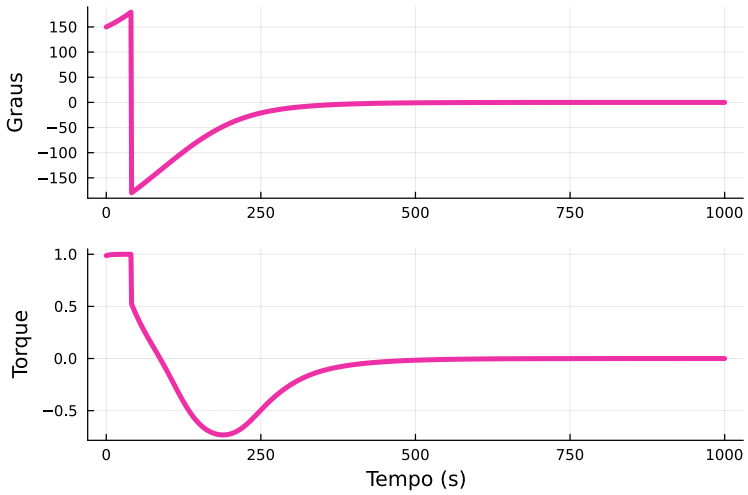
Fonte: Autor.

Figura 6.13 - Controle TD3, condições iniciais $\theta = 150°$ e $\omega = 0,01\,rad/s$.



Fonte: Autor.

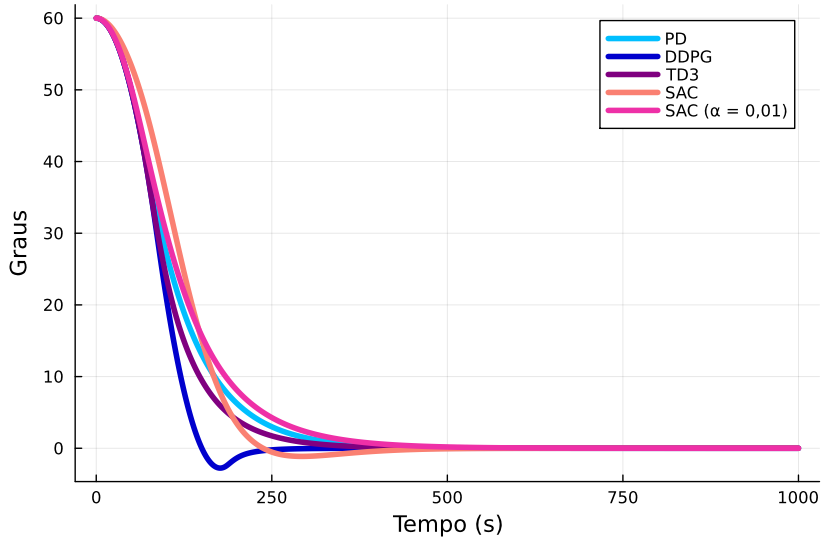
Figura 6.14 - Controle SAC ($\alpha = 0,01$), condições iniciais $\theta = 150°$ e $\omega = 0,01\,rad/s$.



Fonte: Autor.

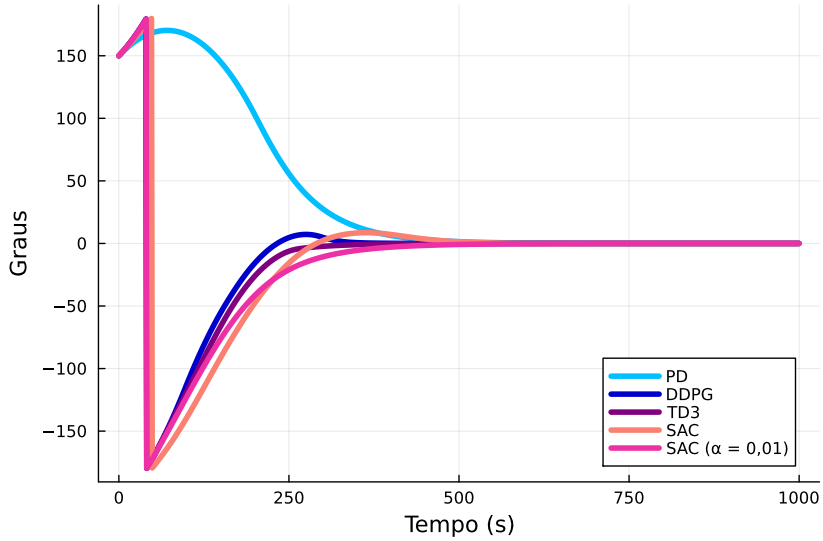
As Figuras 6.15 e 6.16 comparam o controlador PD e os 4 agentes treinados pelo paradigma do RL para as duas condições iniciais anteriormente mostradas. O DDPG possui o menor tempo de assentamento, embora possua uma tendência de sobrepasso. O TD3 é ligeiramente mais lento que o DDPG, mas não possui essa tendência. Como esperado, o SAC com temperatura fixa é superior ao SAC em termos de tempo de assentamento e sobrepasso.

Figura 6.15 - Comparação dos agentes, condições iniciais $\theta = 60°$ e $\omega = 0,00\,rad/s$.



Fonte: Autor.

Figura 6.16 - Comparação dos agentes, condições iniciais $\theta = 150°$ e $\omega = 0,01\,rad/s$.



Fonte: Autor.

A Tabela 6.2 apresenta o retorno G_0 obtido para cada agente apresentado, usando o valor médio após a simulação de 1 milhão de episódios aleatoriamente inicializados. Sendo a recompensa sempre menor ou igual a zero, então o retorno deve ser negativo. Se o valor absoluto de G_0 for interpretado como um custo, então o DDPG e o TD3 possuem respectivamente um custo $6,9\%$ e 5% menor que o controlador PD. A formulação do SAC com temperatura livre possui um custo maior que o PD, mas adotando um valor mais adequado de $\alpha = 0,01$ o custo é $1,5\%$ menor. Muito possivelmente, temperaturas ainda mais baixas levariam a retornos mais próximos do DDPG e TD3.

Tabela 6.2 - Retorno ajustado médio (1 milhão de episódios).

Algoritmo	$\mathbb{E}[G_0]$
PD	-38,18
DDPG	-35,54
TD3	-36,28
SAC	-42,18
SAC ($\alpha = 0.01$)	-37,61

As atualizações do ator e portanto seu desempenho final estão relacionados com a capacidade do crítico em estimar de maneira adequada o valor de ação $Q_\pi(s, a)$. Sendo o ambiente determinístico, então seu valor real pode ser calculado e então comparado com a indicação do crítico. Considerando que o estado possui 2 elementos, é mais fácil visualizar a função de valor $V_\pi(s)$, que pode ser estimada pelo crítico indiretamente notando que $V_\pi(s) = Q_\pi(s, \pi(s))$ se a política é determinística. Os valores reais $V_\pi(s)$ foram obtidos calculando o retorno ajustado G_0 ao se começar no estado s e selecionando ações conforme a política determinística. A Tabela 6.3 apresenta o erro dessas estimativas para o DDPG e o TD3. O SAC foi omitido, visto que a definição do crítico inclui termos entrópicos, conforme a Equação 4.32.

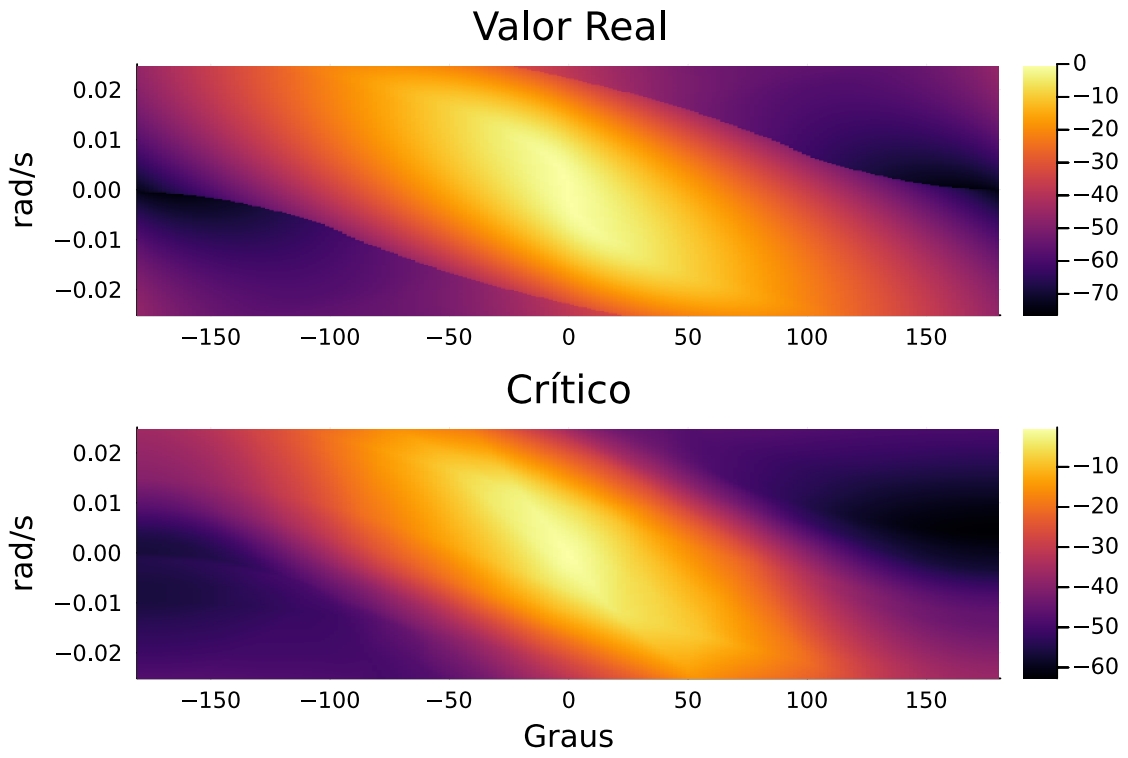
Tabela 6.3 - Erro $V_\pi(s) - V_\pi^w(s)$ do crítico.

Algoritmo	**Erro médio**	**Raíz do erro médio quadrático**
DDPG	2,57	4,96
TD3	-4,53	5,50

Como apontado teoricamente, o DDPG possui uma leve tendência a sobrestimação

do valor de função. Para o TD3, o crítico com menor indicação foi selecionado, de acordo com a ideia original do algoritmo, o que por sua vez leva a subestimação do valor de função. Sendo a recompensa sempre entre em $[-1, 0]$ e $\gamma = 0,99$, então para todos os estados $V_\pi(s)$ deve ser limitado inferiormente por $-1/(\gamma - 1) = -100$ e superiormente por 0.

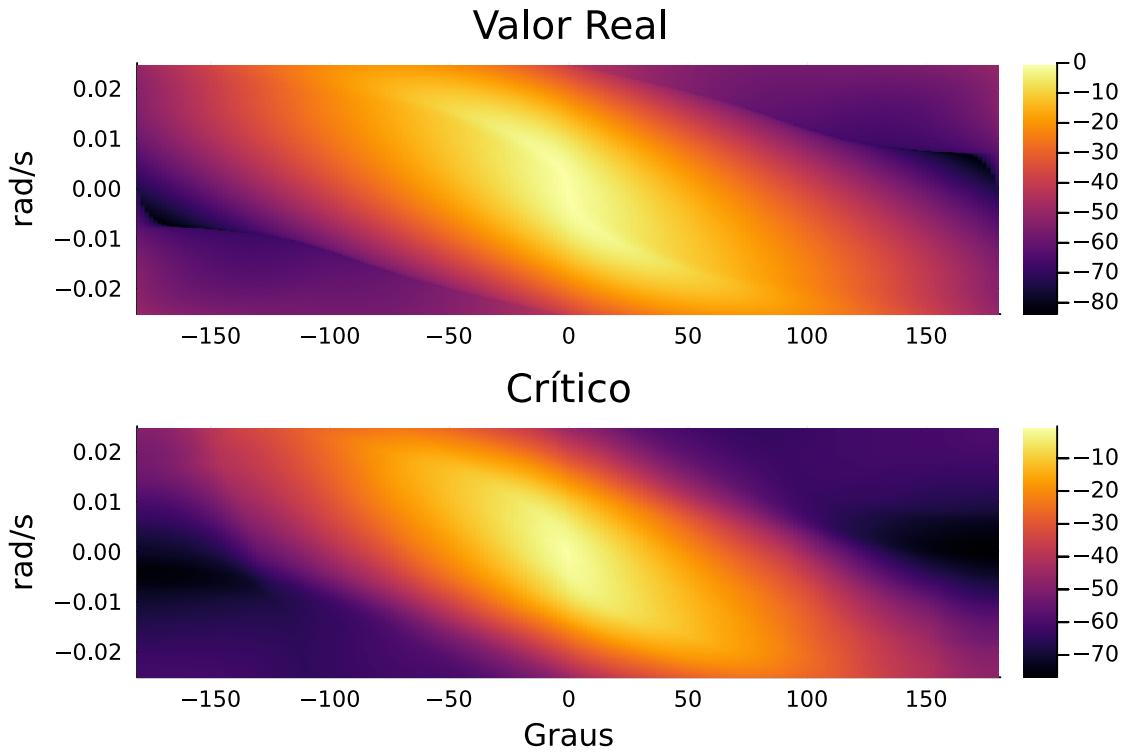
Figura 6.17 - Valor real $V_\pi(s)$ e estimativa do crítico para o DDPG.



Fonte: Autor.

As Figuras 6.17 e 6.18 comparam os valores reais de $V_\pi(s)$ com a indicação do crítico para o DDPG e TD3, respectivamente. Devido às simetrias do ator e da recompensa, o valor real apresenta uma simetria par no estado, isso é, $V_\pi(s) = V_\pi(-s)$. É imediato da análise que os estados com maior retorno estão na diagonal que se estende do canto superior direito do gráfico ao inferior esquerdo: nesses casos, a velocidade auxilia o satélite a alcançar o repouso mais rapidamente. Em contrapartida, os estados com menor retorno possuem um ângulo próximo de 180 graus e uma velocidade angular próxima a zero; assim, precisam de tempo para serem acelerados e alcançarem a origem.

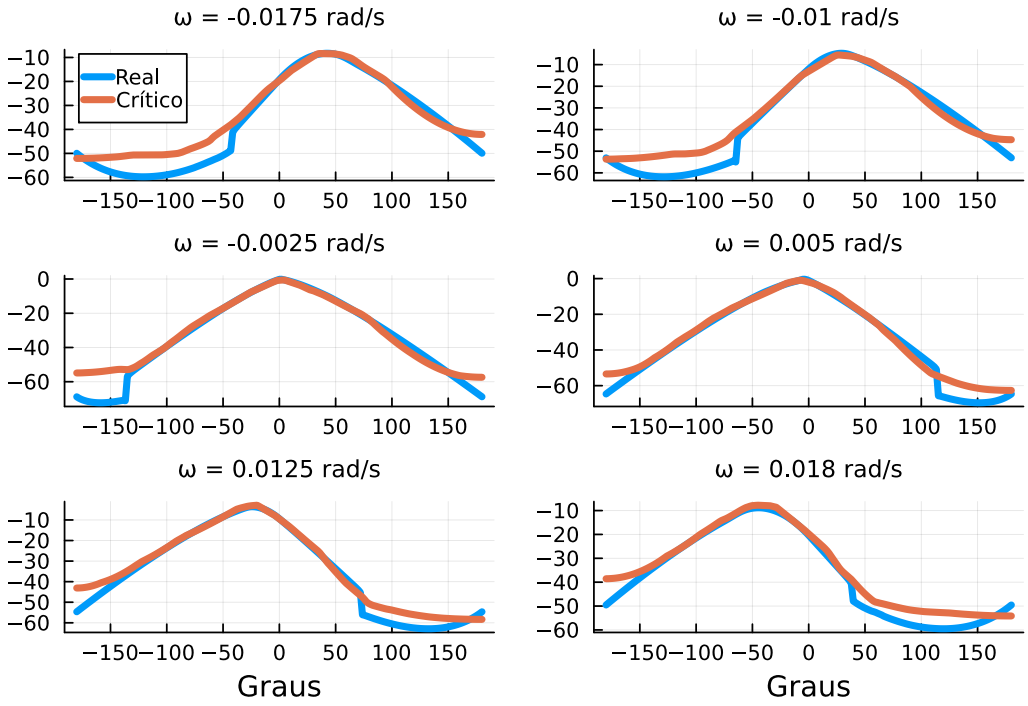
Figura 6.18 - Valor real $V_\pi(s)$ e estimativa do crítico para o TD3.



Fonte: Autor.

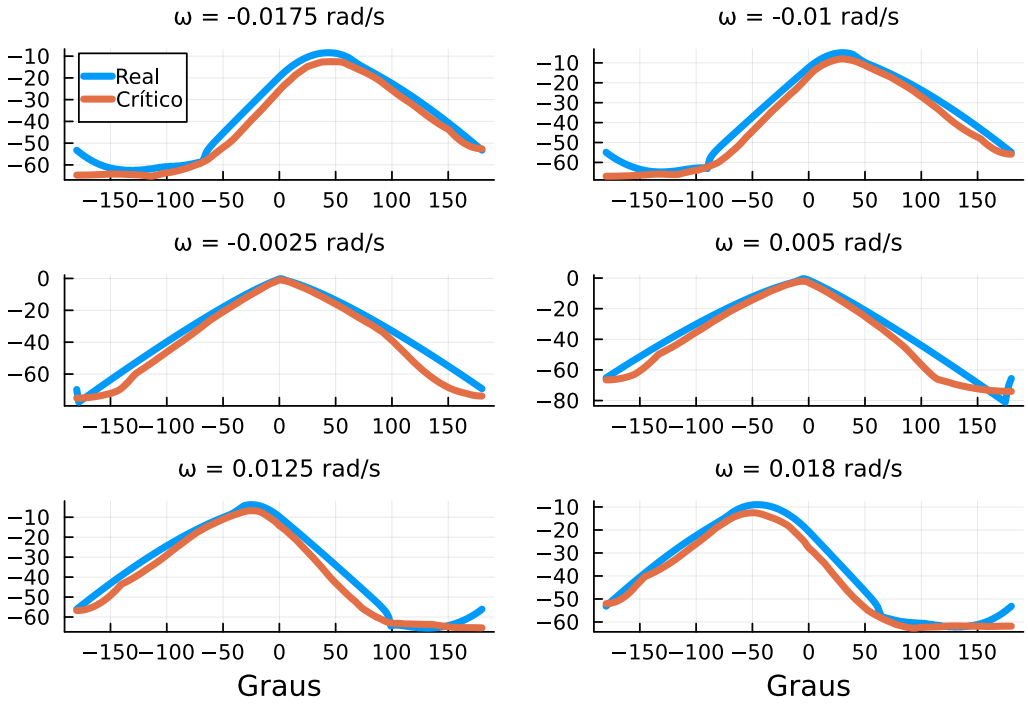
As Figuras 6.19 e 6.20 fornecem uma visão mais detalhada, considerando alguns valores específicos de velocidade angular e variando o ângulo de -180 a 180 graus. Os críticos foram capazes de generalizar a função $V_\pi(s)$, mas a tendência de sobrestimação do DDPG e de subestimação do TD3 ainda pode ser identificada.

Figura 6.19 - Valor real $V_\pi(s)$ e estimativa do crítico do DDPG para alguns valores de velocidade angular.



Fonte: Autor.

Figura 6.20 - Valor real $V_\pi(s)$ e estimativa do crítico do TD3 para alguns valores de velocidade angular.



Fonte: Autor.

Apesar da argumentação e dos exemplos apresentados, não há garantias teóricas acerca da estabilidade desses controladores, em razão da não linearidade inata das redes neurais. Uma forma rudimentar, contudo, de explorar essa questão é realizar uma aproximação do torque ao redor do estado $[0; 0]$ por:

$$T(sen(\theta/2), \dot{\theta}) = T(0,0) + \frac{\partial T}{\partial sen(\theta/2)} sen(\theta/2) + \frac{\partial T}{\partial \dot{\theta}} \dot{\theta}. \tag{6.4}$$

Sendo $T(0,0) = 0$ e aproximando $sen(\theta/2) \approx \theta/2$, então a equação dinâmica se torna:

$$I_z \ddot{\theta} = \frac{\partial T}{\partial sen(\theta/2)} \frac{\theta}{2} + \frac{\partial T}{\partial \dot{\theta}} \dot{\theta}. \tag{6.5}$$

Aplicando a transformada de Laplace para θ e supondo condições iniciais nulas, então a equação acima pode ser reescrita no domínio da frequência como:

$$\theta(s) \left(I_z s^2 - \frac{\partial T}{\partial \dot{\theta}} s - \frac{1}{2} \frac{\partial T}{\partial sen(\theta/2)} \right) = 0. \tag{6.6}$$

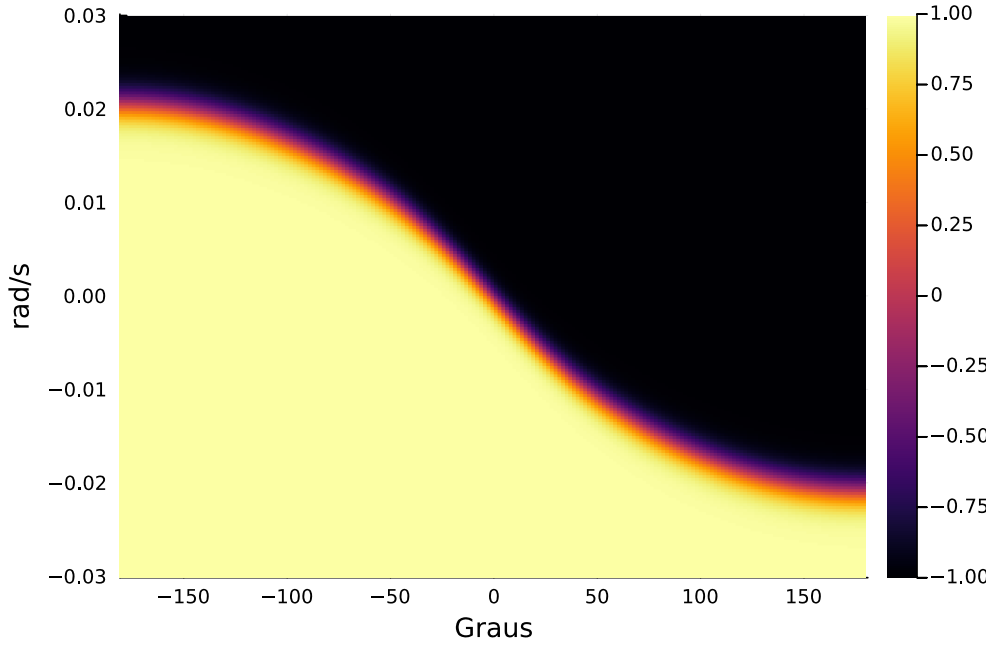
As derivadas parciais podem ser calculadas pelo pacote *Zygote* por meio da função *gradient*; o anexo C fornece um exemplo simples desse cálculo. Agora as raízes do polinômio de segunda ordem devem ser analisadas; se a parte real de ambas raízes forem negativas, então o sistema deve ser estável. As raízes são apresentadas na Tabela 6.4, como todas possuem partes reais negativas, então um argumento pode ser feito que o ponto $[0, 0]$ é estável, isso é, o controlador compensará pequenas perturbações.

Tabela 6.4 - Raízes dos agentes perto da origem.

Algoritmo	**Polinômio**	**Raízes**
PD	$530,7s^2 + 42,21s + 0,512$	$-0,015,\ -0,065$
DDPG	$530,7s^2 + 174,05s + 4,95$	$-0,031,\ -0,296$
TD3	$530,7s^2 + 127,44s + 1,92$	$-0,016,\ -0.224$
SAC	$530,7s^2 + 18,81s + 0.2$	$-0,18 \pm 0,008i$
SAC ($\alpha = 0,01$)	$530,7s^2 + 50,08s + 0,55$	$-0,013,\ -0,082$

Por fim, cabe mencionar que dada a estocasticidade dos algoritmos - uma vez que os estados iniciais e o ruído adicionado à ação são variáveis aleatórias -, simulações adicionais usando os mesmos hiperparâmetros produzem resultados diferentes. A Figura 6.21, por exemplo, apresenta o mapa de calor de um outro agente treinado pelo TD3 com os hiperparâmetros da Tabela 6.1. Como pode ser visto, o torque aproxima-se muito do controlador PD, de fato, as regiões de torque de sinal oposto nos canto superior direito e inferior esquerdo sequer estão presentes.

Figura 6.21 - Fração do torque nominal aplicado pelo TD3 para uma outra simulação.



Fonte: Autor.

6.2 Controle em três eixos: torques externos

O problema de controle de atitude é agora expandido para o caso mais geral no espaço, no qual o satélite deve ser estabilizado ao redor de três eixos. Nessa seção, torques externos, de intensidade limitada, devem ser aplicados para trazer o veículo ao repouso, alinhando-se com o quatérnion $\boldsymbol{q} = 1 + 0i + 0j + 0k$.

O ambiente é descrito pela equação de Euler da dinâmica rotacional de um corpo rígido:

$$\dot{\vec{\boldsymbol{L}}}_{\boldsymbol{sat}} = \vec{\boldsymbol{T}}_{\boldsymbol{ext}} - \vec{\boldsymbol{\omega}} \times \vec{\boldsymbol{L}}_{\boldsymbol{sat}}, \tag{6.7}$$

com o termo $\vec{\boldsymbol{L}}_{\boldsymbol{sat}}$ indicando o momento angular no referencial em rotação com o corpo, dado nesse referencial por $\boldsymbol{L}_{\boldsymbol{sat}} = \boldsymbol{I}_{\boldsymbol{sat}}\boldsymbol{\omega}$, onde $\boldsymbol{\omega}$ é a velocidade angular do corpo em relação ao referencial inercial expressa nas coordenadas do sistema do corpo. Na prática, a determinação dessa velocidade ocorre primariamente por meio de giroscópios embarcados.

Escrita de modo matricial nessa referencial em rotação, a Equação 6.7 permite calcular os componentes da aceleração angular do veículo:

$$\dot{\boldsymbol{\omega}}_{\boldsymbol{sat}} = \boldsymbol{I}_{\boldsymbol{sat}}^{-\mathbf{1}} \left(\boldsymbol{T}_{\boldsymbol{ext}} - \boldsymbol{\omega}_{\boldsymbol{sat}}^{\times} \boldsymbol{I}_{\boldsymbol{sat}} \boldsymbol{\omega}_{\boldsymbol{sat}} \right). \tag{6.8}$$

A matriz $\boldsymbol{I}_{\boldsymbol{sat}}$ para o Amazonia-1 foi apresentada na Equação 5.1. Em relação ao problema discutido na Seção 6.1, uma possível dificuldade adicional para o aprendizado do agente é o acoplamento existente pela diferença dos termos I_{xx}, I_{yy}, I_{zz} e presença de I_{xy}, I_{xz} e I_{yz} não nulos, como discutido na Seção 3.3.2. Isso faz com que o torque em um dado eixo afete um eixo perpendicular a este.

A velocidade angular $\boldsymbol{\omega}_{\boldsymbol{sat}}$ então pode ser propagada realizando uma simples integração pelo método de Euler, do instante de tempo k para $k+1$:

$$\boldsymbol{\omega}_{\boldsymbol{sat}}(k+1) = \boldsymbol{\omega}_{\boldsymbol{sat}}(k) + \Delta t\, \dot{\boldsymbol{\omega}}_{\boldsymbol{sat}}(k). \tag{6.9}$$

O conhecimento da velocidade angular em cada tempo permite a propagação da atitude, que, em razão da eficiência computacional e amplo uso na área espacial, será representada por quatérnions. A propagação obedece então à Equação 3.27 e é resolvida numericamente pelo método de Runge-Kutta 4, realizando-se a normalização do quatérnion ao final.

Essa escolha, por sua vez, também define o estado s para representar o ambiente, que contém os 3 elementos da parte vetorial do quatérnion e os 3 componentes da velocidade angular, isso é, $s = [q_1\ q_2\ q_3\ \omega_x\ \omega_y\ \omega_z]^T$. Portanto, o objetivo do agente é alcançar o estado de norma nula.

A ação efetuada pelo agente a - fisicamente, o torque - é dada por um vetor de 3

elementos, onde cada elemento é limitado ao mesmo módulo máximo de $0.075\,Nm$. Novamente, por convenção, os valores extremos foram mapeados para $[-1, 1]$ e esse valor incluso direto na dinâmica do sistema. Logo, $\mathcal{A}$ pode ser visualizada em três dimensões como um cubo, centrado na origem e se estendendo no intervalo $[-1, 1]$ em cada eixo.

Após dificuldades iniciais, a recompensa r precisou ser ligeiramente alterada para esse novo ambiente. Simulações iniciais tiveram problemas de convergência, que foram resolvidos adicionando um termo que punisse velocidades angulares altas e fortemente incentivasse o agente a chegar à origem. Dessa forma, a nova recompensa tem o formato:

$$r(s_t, a_t) = -\frac{|\theta|}{\pi} + c - 0,2 = -\frac{2\,cos^{-1}(q_0)}{\pi} + c - 0,2. \tag{6.10}$$

O termo c foi definido como:

$$c = \begin{cases} - \quad 150 \quad \text{se} \quad |\boldsymbol{\omega}| > 0,03 \ rad/s, \\ \quad\ 200 \quad \text{se} \quad |s'| < 0,001. \end{cases} \tag{6.11}$$

Notar que embora o termo c considere o valor de s', a recompensa ainda é função de (s, a), pois como o ambiente é determinístico $s' = f(s, a)$. Adicionalmente, é claro que c pode assumir apenas um valor, pois se a velocidade exceder 0,03 rad/s o estado terá uma norma maior que 0,001. Tirando o caso em que $|\boldsymbol{\omega}| > 0,03$ rad/s, a velocidade angular não é punida na Equação 6.10; puni-la poderia desincentivar o agente a diminuir o $|\theta|$ de forma mais rápida.

Adicionalmente, uma constante com valor -0,2 foi inclusa na recompensa para cado passo. Sua função é incentivar o agente a concluir o episódio de forma mais rápida possível trazendo o satélite para um estado terminal perto da origem. Em outras palavras, fisicamente essa recompensa estimula o agente a diminuir seu tempo de assentamento.

Da mesma forma que em muitos problemas de RL, uma variável lógica *healthy* foi definida, de modo a indicar se o satélite está ou não "saudável". Seu valor é verdadeiro sempre que a velocidade angular do satélite, em módulo, for inferior a 0,03 rad/s, caso contrário é falso. A ideia é que, independente da política que o agente esteja seguindo, alcançar uma velocidade dessa ordem significa que o agente não está

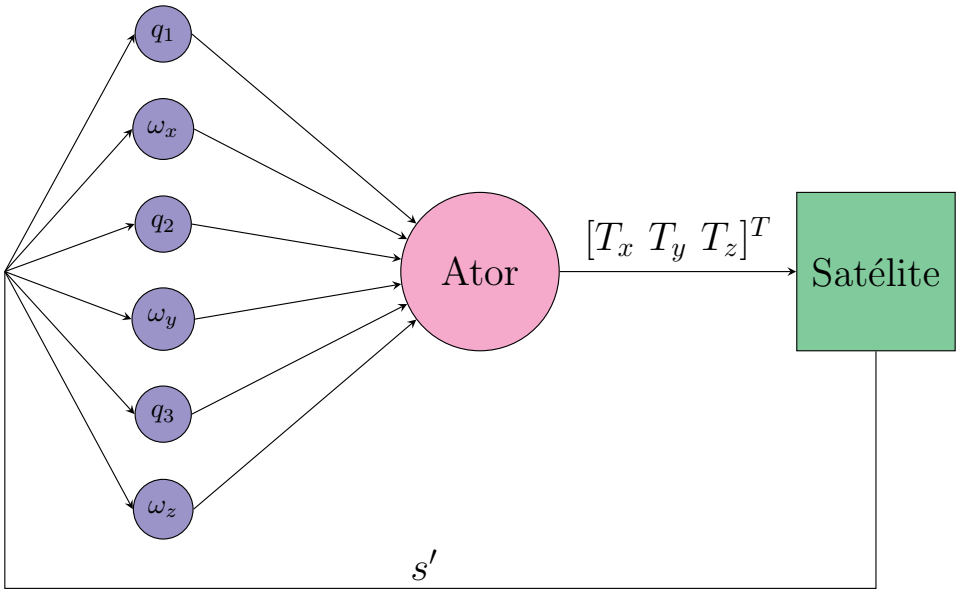
aprendendo, visto que tais velocidades são muito elevadas para serem encontradas nas simulações, uma vez que a velocidade máxima com o que o agente é inicializado é 0,024 rad/s. Desse modo, espera-se que o agente prefira explorar outras políticas e o aprendizado seja acelerado.

A ocorrência de qualquer uma das três seguintes condições leva ao fim do episódio: caso a variável *healthy* se torne falsa - com o agente recebendo uma punição de -150 -, caso o número de passos no episódio alcance 4000, ou caso o agente atinja um estado com norma inferior a 0,001, recebendo uma recompensa de 200.

Os estados são inicializados aleatoriamente seguindo uma distribuição ρ, com a orientação e velocidade angular iniciais sendo calculados de acordo com o discutido na Seção 5.3. Em todos os instantes do tempo, o quatérnion é calculado de forma que sua parte escalar $\eta = cos(\theta/2)$ seja sempre positiva.

A simetria ímpar permanece no problema tridimensional, pois, assim como no controlador PD, é de se esperar que o torque $\boldsymbol{T}$ obedeça a $\boldsymbol{T}([q_1, q_2, q_3, \boldsymbol{\omega}]) = -\boldsymbol{T}([-q_1, -q_2, -q_3, -\boldsymbol{\omega}])$. Portanto, a mesma função de ativação tangente hiperbólico foi empregada nos neurônios do ator. Visto que não há viés na rede e que $tanh(0) = 0$, a condição de torque nulo com entradas nulas foi atendida. O esquema geral do controle de atitude é ilustrado na 6.22.

Figura 6.22 - Visão esquemática do controle de atitude por uma única rede.



Fonte: Autor.

Novamente, o ator e o crítico foram parametrizados por redes neurais totalmente conectadas. Problemas iniciais foram enfrentados ao se usar redes com 2 camadas ocultas no ator, como foi feito no problema unidimensional. Após uma análise cuidadosa sobre essa questão, chegou-se à conclusão que uma arquitetura com apenas uma camada oculta para o ator seria capaz de representar o torque desejado. Dessa forma, o ator possui 64 neurônios na camada oculta, o que equivale a 576 parâmetros a serem ajustados.

Embora redes com mais parâmetros tenham uma maior capacidade de aproximação, esse ganho frequentemente ocorre à custa da generalização: a rede pode aprender tão bem os pares entrada/saída que lhe são fornecidos, tornando o erro muito pequeno pelo ajuste de seus parâmetros, que perde a capacidade de aprender o aspecto geral da solução. No campo de ML, esse problema é conhecido como *overfitting*.

Conforme discutido na Seção 4.3.4, redes neurais com uma camada oculta são aproximadores universais, contanto que um número suficiente de neurônios esteja disponível. O torque ótimo deve ser uma função de s relativamente simples, contínua e bem comportada. Portanto, um número de parâmetros relativamente baixo deve ser capaz de realizar esse aproximação. Dessa forma, a rede neural do ator pode possuir apenas uma camada oculta. Adicionalmente, redes neurais menos profundas favorecem a função de ativação *tanh*, aliviando o problema do gradiente nulo nos limites de saturação, à medida que o caminho pelo qual gradiente é propagado é menor.

Tabela 6.5 - Hiperparâmetros usados na simulação em três dimensões.

Hiperparâmetro	**Valor**
Taxa de desconto	0,99
Camadas ocultas (ator)	1
Neurônios por camada (ator)	64
Função de ativação (ator)	tanh
Camadas ocultas (crítico)	2
Neurônios por camada (crítico)	128
Função de ativação (crítico)	ReLU
Tamanho do replay	250000
Tamanho do minibatch	128
Número de passos	10^7
Passos entre atualizações	10
Otimizador	ADAM
Taxa de aprendizado (ator)	0,00003
Taxa de aprendizado (crítico)	0,001

Os hiperparâmetros são listados na Tabela 6.5. De forma a melhorar a estabilidade e permitir a convergência da política, a taxa de aprendizado do ator foi baixada e o tamanho do replay aumentado. A taxa de aprendizado do crítico é muito maior que a do ator, de forma a garantir que os valores de ação possam convergir mais rapidamente.

Uma implementação concreta do tipo *AbstractHook* é utilizada para analisar o desempenho do agente a cada 50000 passos, estimando o tempo médio para alcançar-se um estado com norma inferior a 0,001 a partir de 30 estados iniciais escolhidos aleatoriamente. Caso o estado não seja alcançado, o valor de 4000 é atribuído. Nessas simulações, o ruído do ator é desconsiderado, isso é, a política determinística é usada. O agente com melhor desempenho é salvo enquanto o algoritmo prossegue em suas atualizações. Os estados empregados para obter essa métrica são listados no Anexo B.

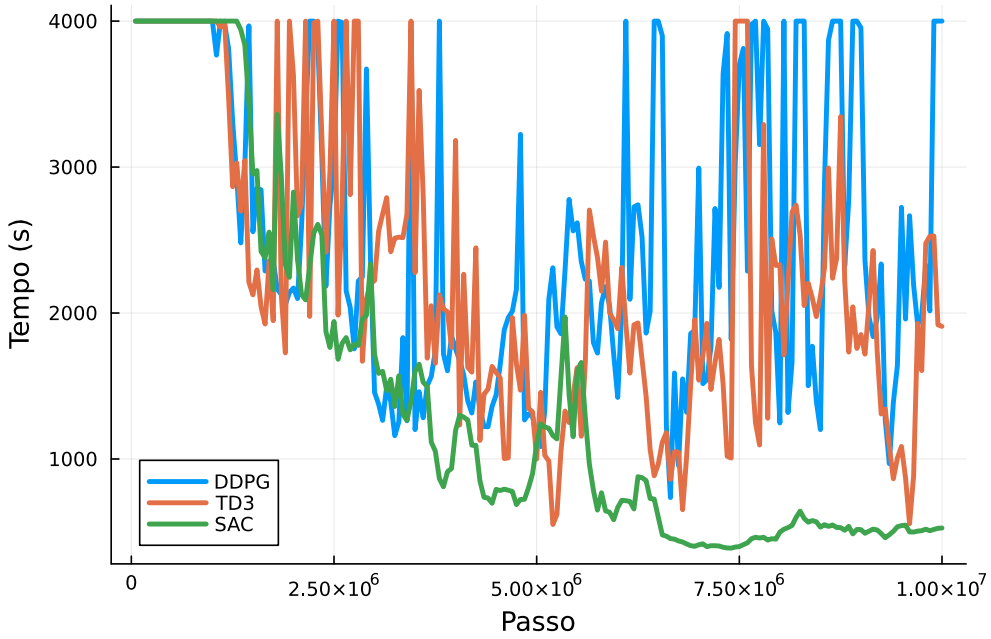
A Figura 6.23 indica a evolução dessa métrica ao longo da simulação para o DDPG, TD3 e SAC. Sendo o ator inicializado aleatoriamente, esse valor corresponde a 4000 de início para todos algoritmos e começa a diminuir após cerca de 1 milhão de passos, quando o agente adquire experiência suficiente para se mover rumo a maiores retornos e, consequentemente, menores tempos. Entretanto, a evolução posterior ocorre de maneira muita distinta.

Os tempos do DDPG e do TD3 declinam inicialmente, mas eventualmente passam a apresentar um caráter oscilatório, variando entre o valor máximo de 4000 e um valor abaixo de 2000, sem jamais se estabilizar em um platô. Isso é, as atualizações do ator ora melhoram seu desempenho, ora o pioram. Essas variações são ainda mais bruscas para o DDPG, que ao final da simulação ainda retorna ao valor inicial. O TD3, por outro lado, embora também oscile, é capaz de, ao final, manter-se dentro de um intervalo mais confinado.

O SAC, por sua vez, apresenta uma evolução virtualmente decrescente. Oscilações ainda estão presentes, mas possuem uma amplitude muito menor em relação aos outros dois agentes. Pouco depois da metade da simulação, o agente alcança um valor mínimo e então se estabiliza próximo a esse valor, sem sofrer grandes alterações.

Essa robustez do SAC é muito possivelmente consequência do ajuste automático da temperatura α, que permite ao agente equilibrar adequadamente as tendências de exploração e proveito ao longo de sua interação com o ambiente. O agente começa quente, favorecendo a exploração, e vai esfriando, tendendo a um comportamento

Figura 6.23 - Evolução da métrica para os três algoritmos ao longo da simulação.



Menor tempo é melhor.

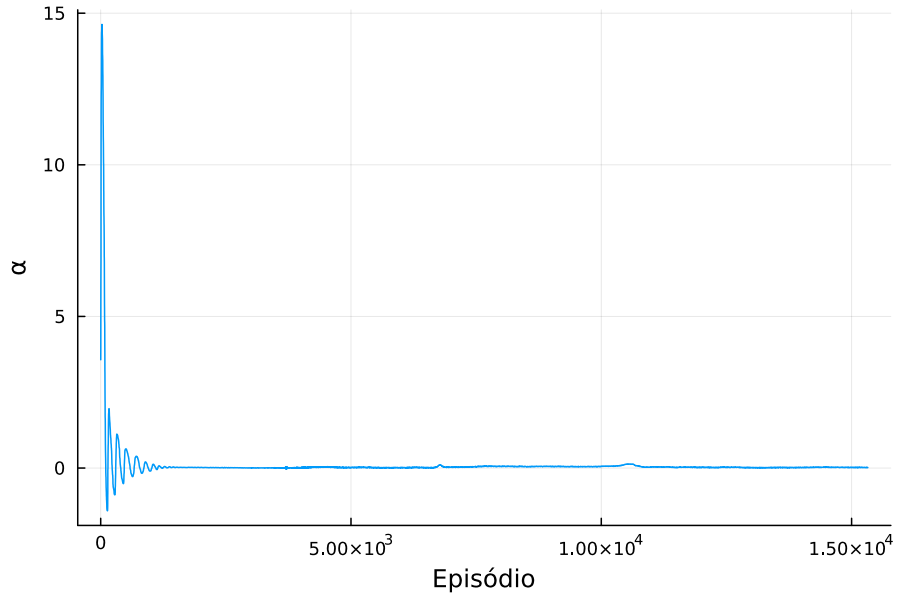
Fonte: Autor.

mais determinístico. Por outro lado, o DDPG e o TD3 garantem a exploração adicionando um ruído de distribuição normal à saída da rede neural do ator, cuja variância permanece inalterada à medida que o agente aprende mais sobre o ambiente.

Certamente, é possível que o ajuste de alguns dos hiperparâmetros da Tabela 6.5, como taxa de aprendizado, tamanho da rede ou do replay, levasse à estabilização do DDPG e do TD3 e portanto melhorasse o seu desempenho de forma significativa. Entretanto, esses resultados indicam que, para ambientes mais complexos, com maiores dimensões, a necessidade de balancear exploração com procura torna-se mais importante e certos algoritmos tornam-se mais frágeis em relação à escolha de seus hiperparâmetros. Provavelmente, o ruído gaussiano do DDPG e do TD3 é vantajoso nas primeiras interações para garantir a exploração do ambiente, mas torna-se desvantajoso em assegurar a convergência a uma política ótima à medida que o número de interações cresce. O SAC, em contrapartida, ajusta a temperatura α para equilibrar essas tendências. A Figura 6.24 apresenta sua evolução. Após um período curto extremamente oscilatório - incluindo temperaturas negativas -, a temperatura estabiliza-se em um valor próximo a 0; seu último valor foi de 0,016.

Em virtude do desempenho claramente superior do SAC, ele será o foco do restante

Figura 6.24 - Evolução do parâmetro temperatura ao longo da simulação.



Fonte: Autor.

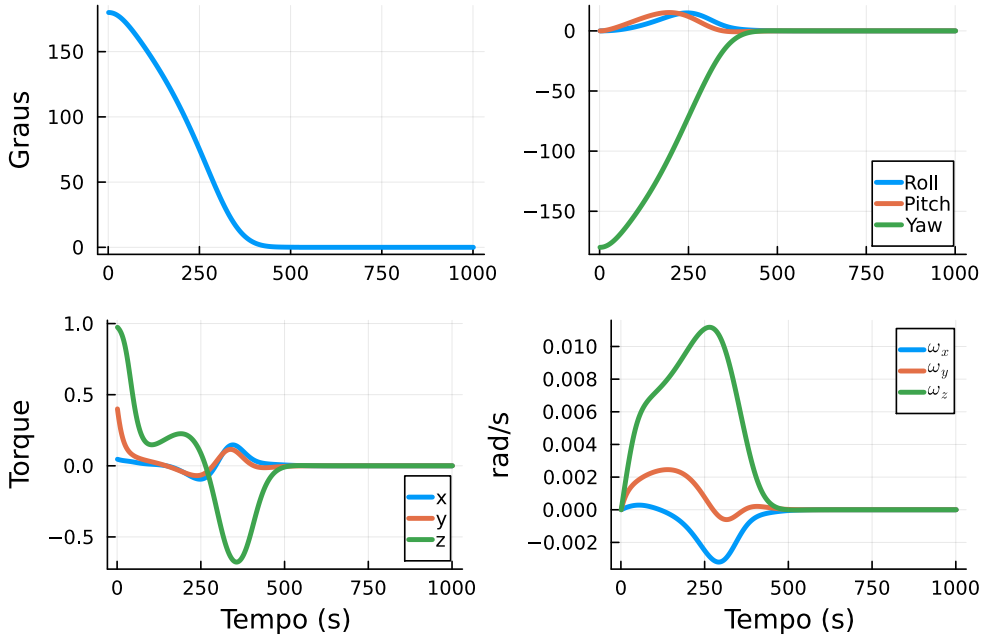
da discussão. A Tabela 6.6 indica as condições iniciais de 3 cenários escolhidos para simular o ator obtido. Em particular, o cenário 1 foi construído de maneira a representar uma condição de erro máximo de apontamento de 180 graus e nenhuma velocidade angular inicial.

Tabela 6.6 - Condições iniciais dos cenários.

Cenário	Roll	Pitch	Yaw	ω_x [rad/s]	ω_y [rad/s]	ω_z [rad/s]
1	0,0°	0,0°	-180,0°	0,00	0,00	0,00
2	90,0°	-60,0°	120,0°	0,01	0,01	0,01
3	30,0°	60,0°	90,0°	0,02	-0,01	0,02

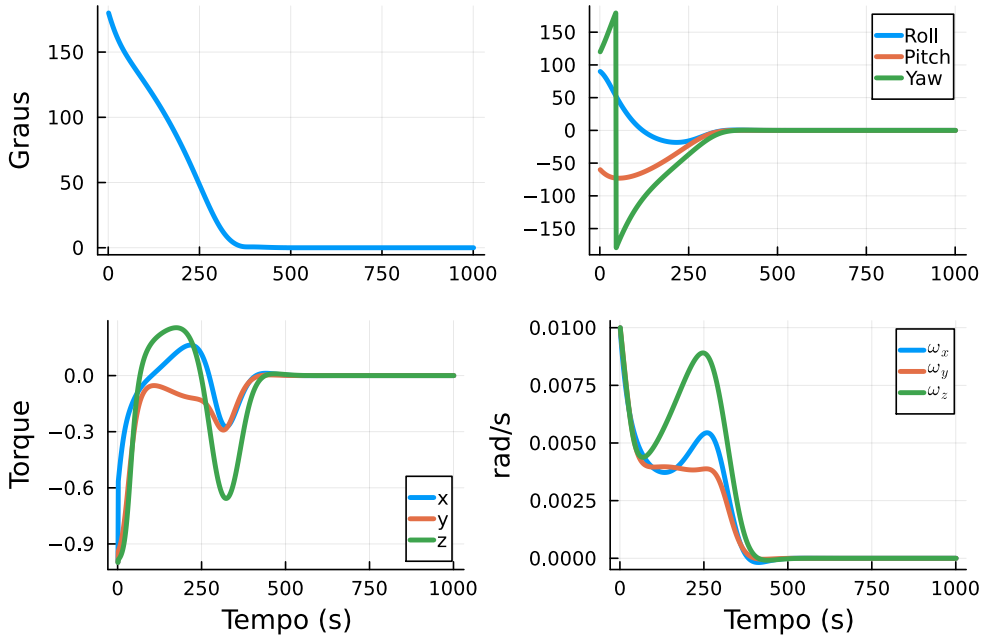
As Figuras 6.25, 6.26 e 6.27 ilustram o erro angular, os valores de *roll*, *pitch* e *yaw*, o torque aplicado em cada eixo e a velocidade angular do satélite para cada cenário. O agente foi capaz de controlar o satélite e trazê-lo ao repouso com erro de apontamento nulo. Deve-se destacar que esse erro decresce de maneira monótona e não apresenta o aspecto oscilatório de uma solução subamortecida. Adicionalmente, o torque aplicado é suave e não possui variações abruptas.

Figura 6.25 - SAC: simulação de controle em 3 eixos, cenário 1.



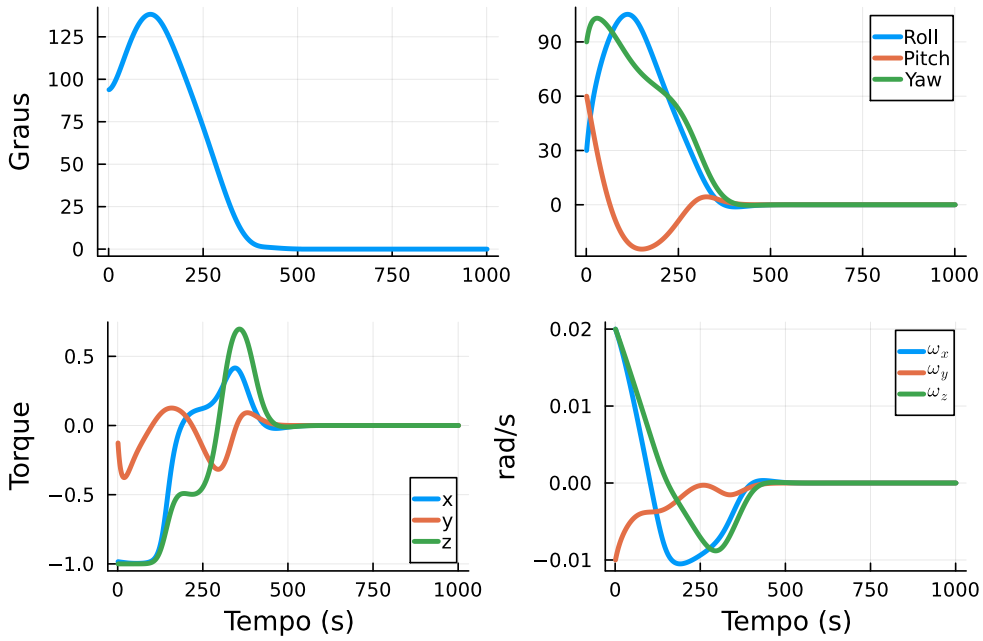
Fonte: Autor.

Figura 6.26 - SAC: simulação de controle em 3 eixos, cenário 2.



Fonte: Autor.

Figura 6.27 - SAC: simulação de controle em 3 eixos, cenário 3.



Fonte: Autor.

Uma desvantagem, contudo, desse agente é a integração inerente dos neurônios e sua influência na saída da rede neural. Dessa forma, todos os componentes do vetor estado s afetam, por exemplo, o torque ao longo do eixo x. Na Figura 6.25, certamente seria mais razoável que o torque em x fosse inicialmente 0, visto que não há velocidade angular ao longo desse eixo e o componente de *roll* também é nulo. Entretanto, em razão dessa característica da rede neural, um pequeno valor de torque é ainda assim aplicado. Consideração semelhante é válida para o eixo y.

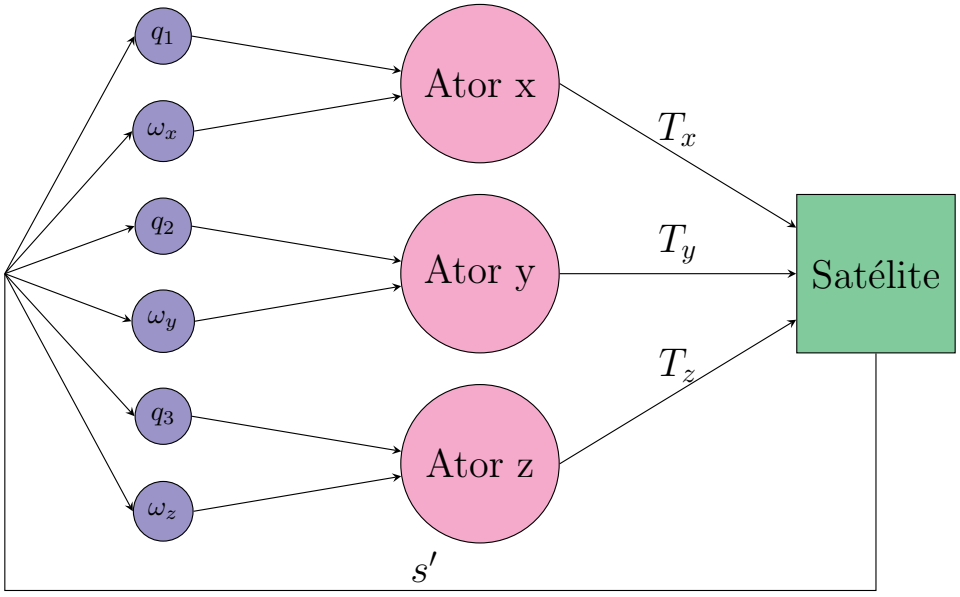
Igualmente, embora a rede seja capaz de controlar o veículo em 3 dimensões, o torque não possui a saturação característica do controlador PD, o que pode comprometer a rapidez com o que o satélite atinge o repouso.

Sendo os eixos fracamente acoplados, uma outra alternativa é treinar 3 redes neurais de forma independente para cada eixo - como na Seção 6.1 - e então combiná-las para ter o controle no espaço. Como o vetor de estado em uma dimensão era do tipo $[sen(\theta/2), \dot{\theta}]$, então de maneira análoga cada rede neural terá como entradas o componente do quatérnion correspondente e a velocidade em torno desse eixo, isso é $[q_1, \omega_x]$ para o eixo x, $[q_2, \omega_y]$ para y e $[q_3, \omega_z]$ e para z. A Figura 6.28 fornece uma visão do esquema de controle para esse caso; a comparação com 6.22 em particular

torna as diferenças claras.

Tendo em conta que a solução anterior de uma única rede neural para determinar o vetor torque contava com uma camada oculta, as três redes adotadas também possuem essa arquitetura, contendo 32 neurônios nessa camada. Os agentes foram treinados pelo SAC em torno de cada eixo; detalhes são apresentados no Anexo A. Ao todo, a rede neural resultante possui 288 parâmetros, exatamente a metade da solução integrada.

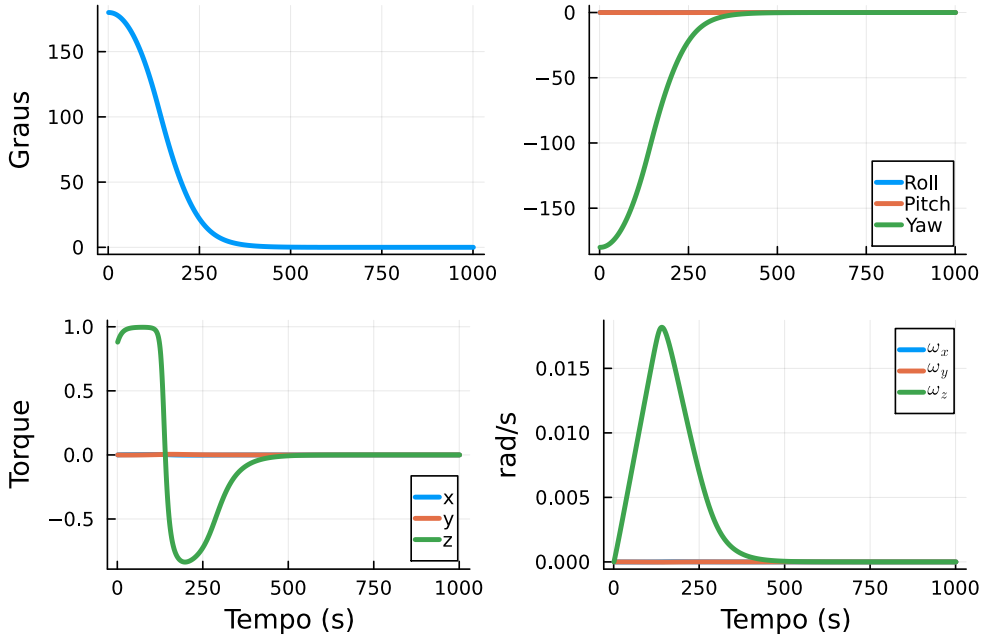
Figura 6.28 - Visão esquemática do controle de atitude em três dimensões por redes independentes.



Fonte: Autor.

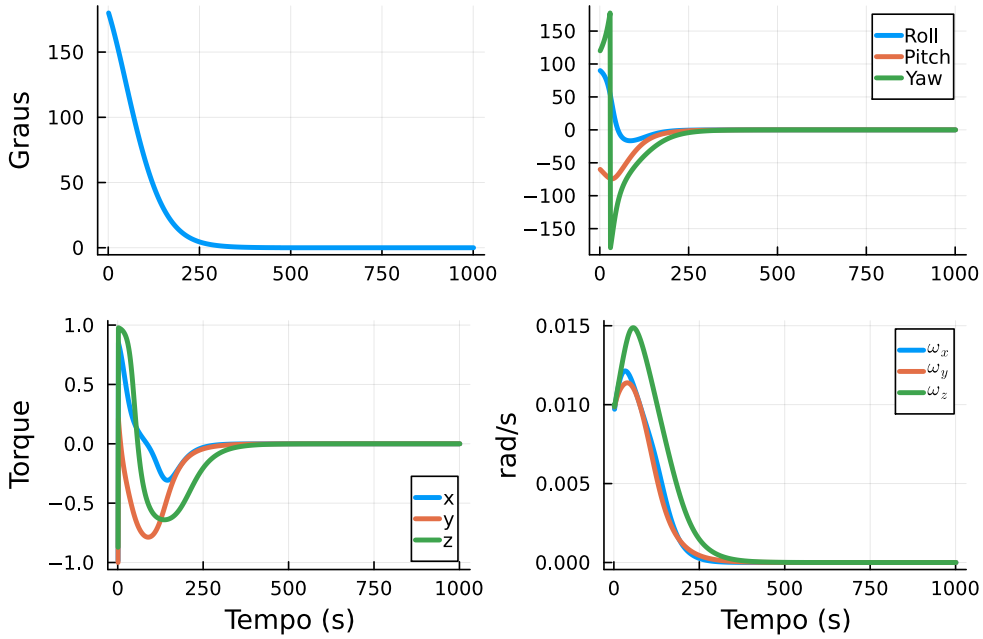
As Figuras 6.29, 6.30 e 6.31 ilustram os mesmos 3 cenários para esse agente. O torque agora satura, o que leva a tempos de assentamento ligeiramente inferiores. Por fim, a comparação entre esses dois agentes e o controlador PD do Amazonia-1 para os 3 cenários é mostrada nas Figuras 6.32, 6.33 e 6.34.

Figura 6.29 - SAC: simulação do controle em 3 eixos, rede desacoplada, cenário 1.



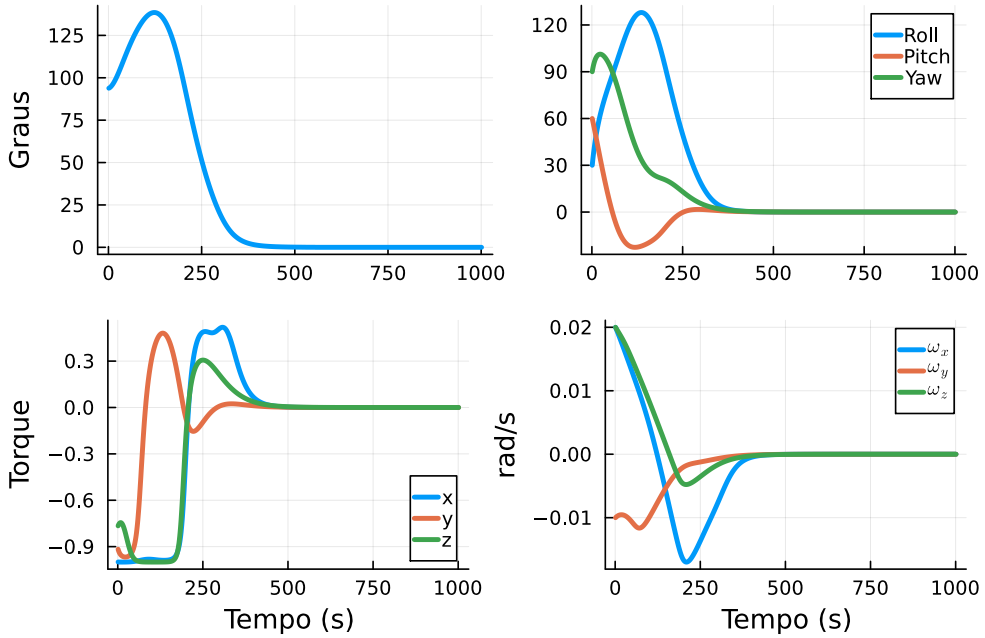
Fonte: Autor.

Figura 6.30 - SAC: simulação do controle em 3 eixos, rede desacoplada, cenário 2.



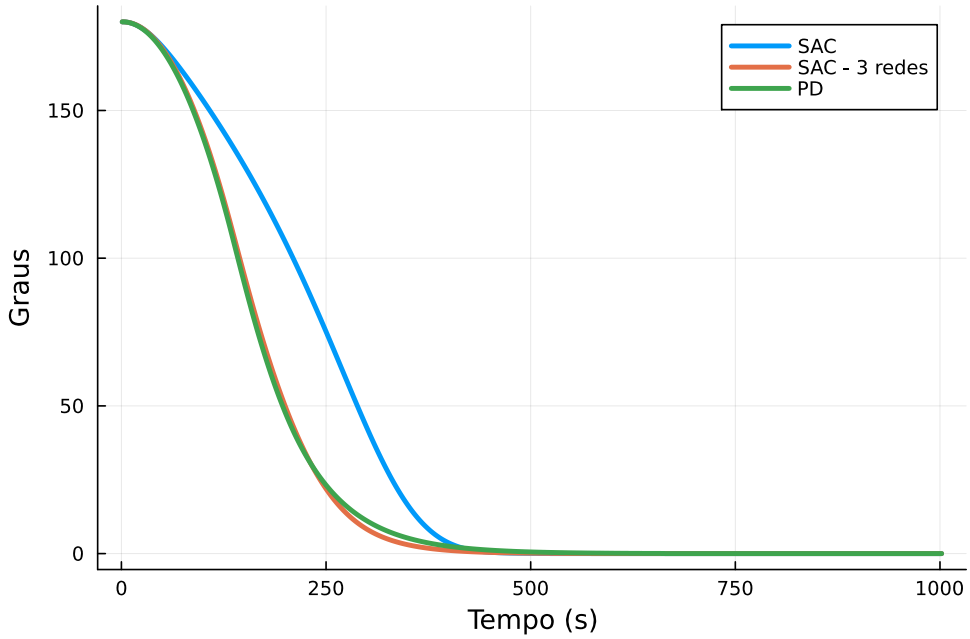
Fonte: Autor.

Figura 6.31 - SAC: simulação do controle em 3 eixos, rede desacoplada, cenário 3.



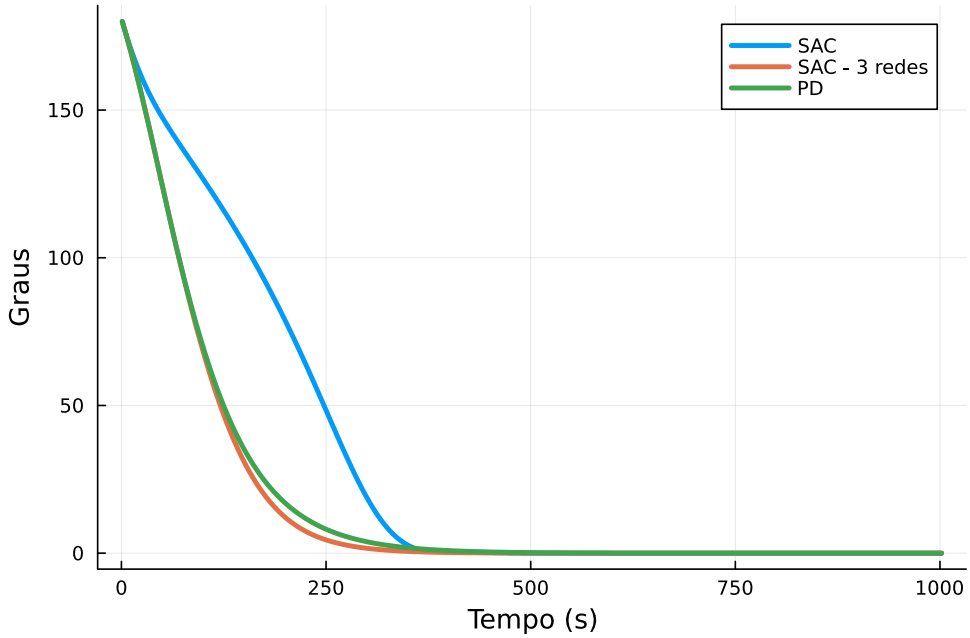
Fonte: Autor.

Figura 6.32 - Comparação para o cenário 1.



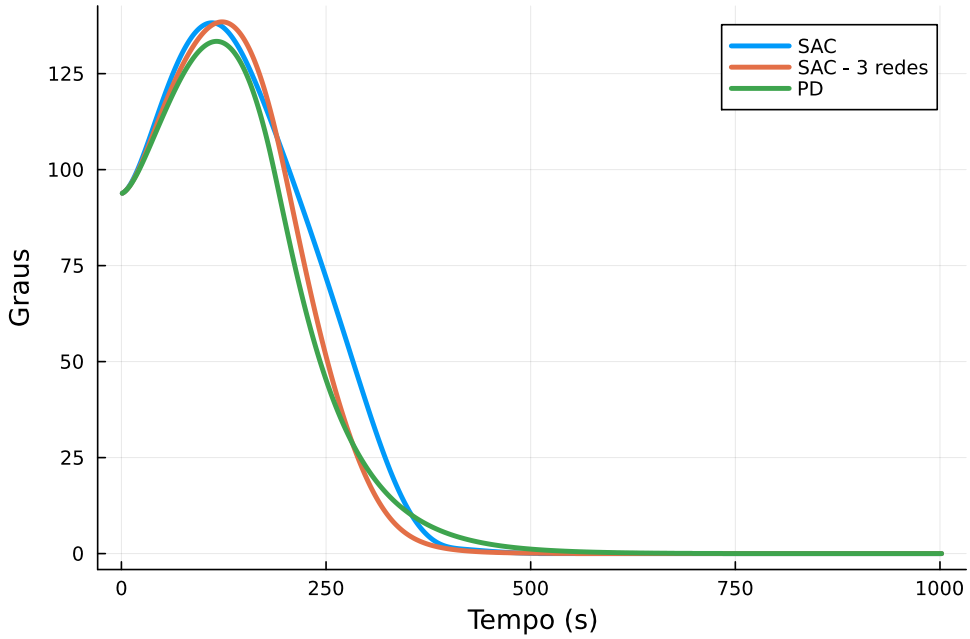
Fonte: Autor.

Figura 6.33 - Comparação para o cenário 2.



Fonte: Autor.

Figura 6.34 - Comparação para o cenário 3.



Fonte: Autor.

O controlador SAC com 3 redes independentes converge para um comportamento muito semelhante ao controlador PD, aparentando ser ligeiramente mais rápido. Surpreendentemente, o controlador SAC com uma única rede, embora frente aos outros dois controladores passe uma maior parte do tempo mais longe do ângulo de referência, alcança a condição final desejada em um tempo muito semelhante, como pode ser visto na Tabela 6.7. Qualitativamente, isso se deve a forma como as acelerações e desacelerações tendem a ocorrer de uma forma mais suave.

Tabela 6.7 - Tempo em segundos para que $|s| < 0,001$.

Cenário	PD	SAC	SAC (3 redes)
1	605	493	512
2	536	467	433
3	657	495	502

Esse tempo é provavelmente consequência do formato da recompensa na Equação 6.10, que estimula o agente o atingir o estado terminal no menor tempo possível. Essa simples análise ilustra certos aspectos do RL: a importância da recompensa no desempenho do agente, que pode assumir comportamentos pouco intuitivos no sentido de maximizar o retorno.

6.3 Controle em três eixos: rodas de reação

A abordagem da Seção anterior pode ser expandida para o controle de atitude com rodas de reação. Como discutido anteriormente, por razões de redundância normalmente satélites contam com ao menos 4 rodas de reação, de forma que essa roda adicional dá origem a um torque com componentes em ao menos dois dos eixos das outras rodas.

O desenvolvimento acima considerou torques externos agindo em um corpo rígido pela equação de Euler. A presença das rodas faz com que o problema não possa mais ser tratado dessa maneira - embora tanto o satélite como as rodas sejam corpos rígidos - e termos adicionais são inclusos para considerar esse acoplamento, levando à Equação 3.51.

Assim, a velocidade de rotação das rodas em torno do seu eixo possui influência na propagação da atitude e portanto para obedecer à propriedade markoviana o estado precisa ser redefinido para $s = [q_1\, q_2\, q_3\, \omega_x\, \omega_y \omega_z\, \omega_{rw,1}\, \omega_{rw,2}\, \omega_{rw,3}\, \omega_{rw,4}]^T$. Utilizar a definição anterior do estado com apenas os 6 primeiros elementos tornaria o problema

parcialmente observável (POMDP).

Um estado maior contendo 10 elementos faz com que encontrar a solução ótima seja computacionalmente mais desafiador. Adicionalmente, a hipótese de viés nulo, que garante um torque nulo quando o satélite estiver em repouso alinhado com a referência, não é mais válida: o agente deve aprender a aplicar torque nulo sempre que os 6 elementos iniciais sejam nulos, independente da velocidade angular das rodas. Assim, o viés é um parâmetro a mais a ser aprendido. Como a simetria não é mais aplicável, uma função de ativação do tipo ReLU precisaria ser usada para esse novo ator.

Há dois modos de evitar essa dificuldade. O primeiro é simplesmente manter o estado antigo[2] e aceitar o problema como um POMDP. Nesse caso, o agente tem conhecimento imperfeito do ambiente, mas isso pode ser compensado pela simplicidade resultante.

O segundo é assumir os 4 torques do vetor $[T_x\ T_y\ T_z\ T_w]^T$ como torques externos - onde w designa o eixo $[\frac{1}{\sqrt{3}}\ \frac{1}{\sqrt{3}}\ \frac{1}{\sqrt{3}}]^T$ - e utilizar a mesma dinâmica de corpo rígido da Seção 6.2, desprezando inteiramente a dinâmica das rodas, transferindo essa solução posteriormente para o caso em que elas são inclusas. De fato, é interessante notar que o controlador PD do Amazonia-1 não se utiliza do valor da velocidade de rotação das rodas. Visto que ademais os algoritmos discutidos pressupõem que o problema seja MDP, essa será a solução adotada. Notar que diferente do controlador PD, o ator agora controla diretamente o torque aplicados às rodas, não necessitando de uma matriz pseudoinversa para realizar a conversão.

Tendo em conta a maior estabilidade e desempenho apresentados, o SAC será usado para essa simulação. A recompensa permanece com o mesmo formato da Equação 6.10 e os hiperparâmetros são os mesmos dos da Tabela 6.5.

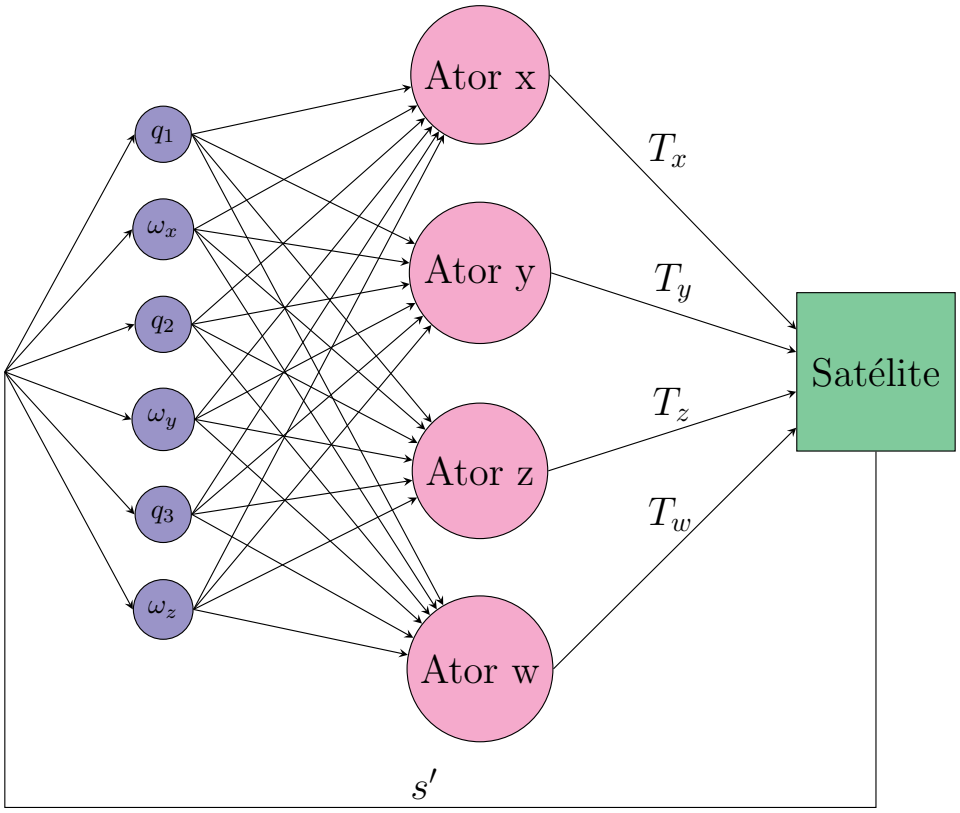
O formato do ator - que deve fornecer um vetor com 4 elementos - foi alterado a partir da experiência obtida com o caso anterior. Ao usar um ator tal que cada eixo fosse parametrizado por sua própria rede, o desempenho aproximou-se do controlador PD. Entretanto, isso exigiu que cada rede fosse treinada em seu eixo separadamente e não para o problema completo. Isso foi possível pois foi assumido a independência entre os eixos. Por sua vez, usar uma única rede neural para gerar o vetor torque

[2] A rigor, manter o vetor observação com 6 elementos. O estado é definido pela dinâmica do ambiente - possuindo 10 elementos no caso com as rodas de reação - e em problemas parcialmente observáveis não pode ser reconstruído unicamente a partir da observação.

permitiu ao agente utilizar do conhecimento adquirido da iteração com a dinâmica tridimensional. Contudo, esse formato gera uma certa dependência entre os componentes da saída, pois como a rede é totalmente conectada o ajuste do peso de uma conexão afeta mais de um componente.

A arquitetura ideal deve combinar aspectos dessas duas soluções. A forma encontrada para fazê-lo foi usar uma rede neural parcialmente conectada, composta de 4 sub-redes independentes de 1 camada oculta com 16 neurônios que definem o torque para cada roda. O estado completo serve de entrada. A Figura 6.35 esquematiza esse arranjo.

Figura 6.35 - Visão esquemática do controle de torque das 4 rodas.



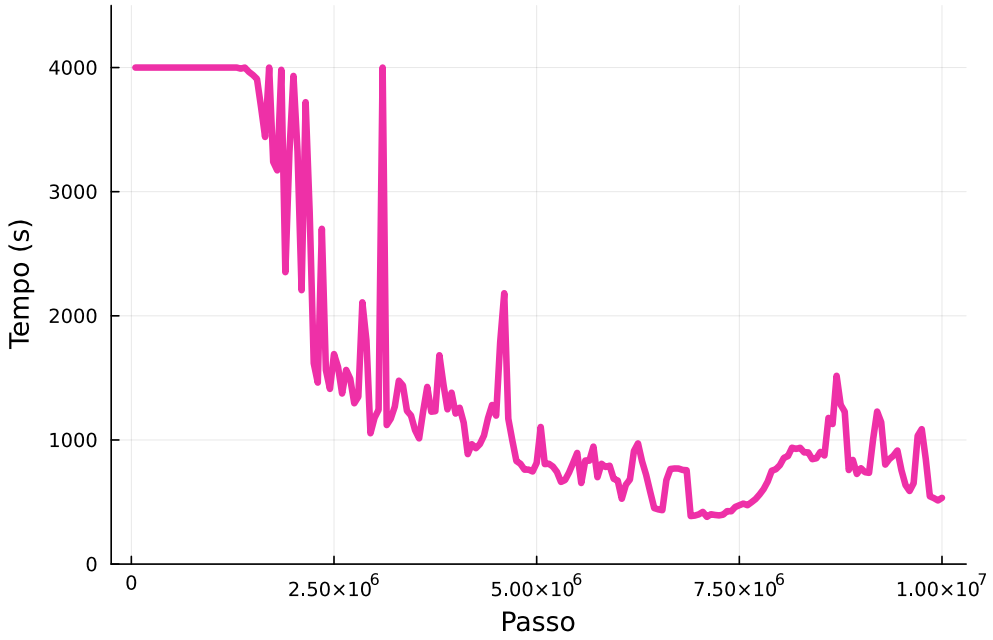
Fonte: Autor.

Com esse formato, o torque no eixo x, por exemplo, pode ser alterado sem modificar o torque no eixo y. De certa forma, esse paralelismo cria uma independência no aprendizado: o agente aprende a controlar cada eixo sem influenciar o outro diretamente. Cada sub-rede dispõe de 112 parâmetros; a rede inteira possui, portanto,

448 parâmetros.

A Figura 6.36 apresenta a evolução do aprendizado do agente, usando a mesma métrica definida na Seção 6.2. Embora o platô não seja tão claro, o agente se estabiliza em um tempo abaixo de 1000 segundos.

Figura 6.36 - Evolução do desempenho do SAC, torque em 4 eixos.

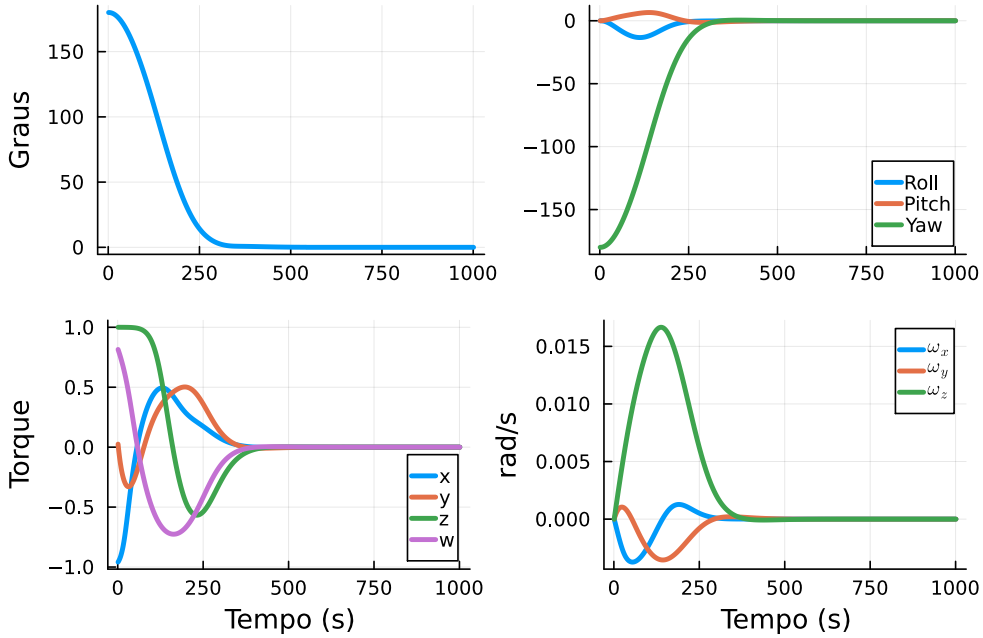


Fonte: Autor.

Novamente, o melhor agente é selecionado. As Figuras 6.37, 6.38 e 6.39 apresentam seu desempenho para os 3 cenários escolhidos. Novamente, o agente consegue trazer o satélite ao repouso com erro nulo e o torque aplicado em todos os eixos é suave. Em função da presença da roda ao longo do eixo w, que aumenta o torque disponível, os tempos de assentamento são ligeiramente inferiores em relação aos dos agentes da Seção 6.2.

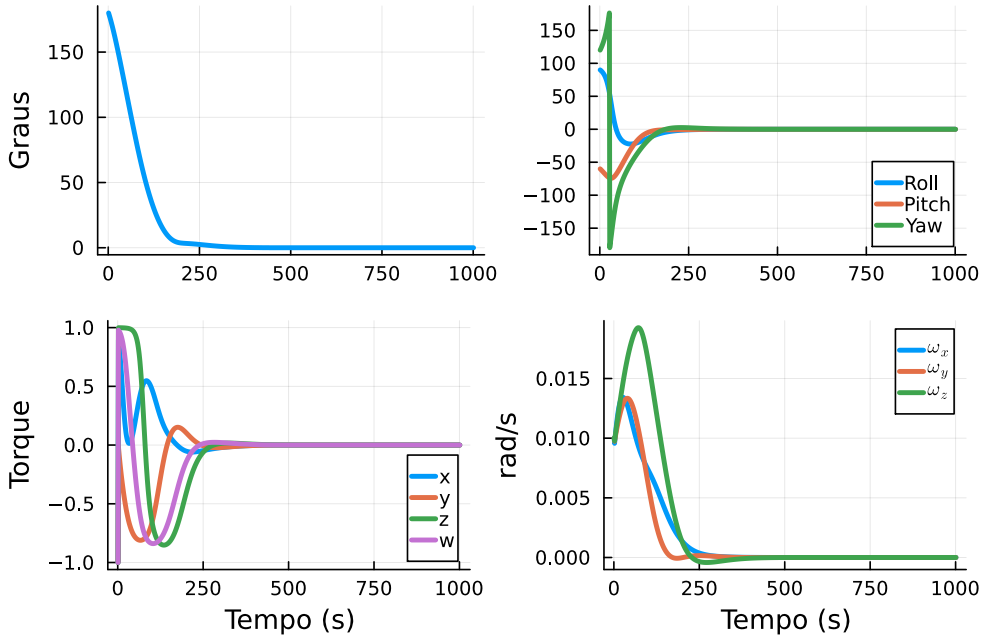
Um fato surpreendente é que o agente é capaz de controlar ao satélite, mesmo a partir de condições extremas para o qual não foi treinado. Por exemplo, a Figura 6.40 mostra o controle em um desses cenários, no qual a velocidade angular em torno de cada eixo é 0,05 rad/s, equivalente a uma velocidade angular total de aproximadamente 5 graus por segundo, mais que o dobro da velocidade angular máxima com que o satélite podia ser inicializado em seu treinamento.

Figura 6.37 - SAC: simulação de torque em 4 eixos, cenário 1.



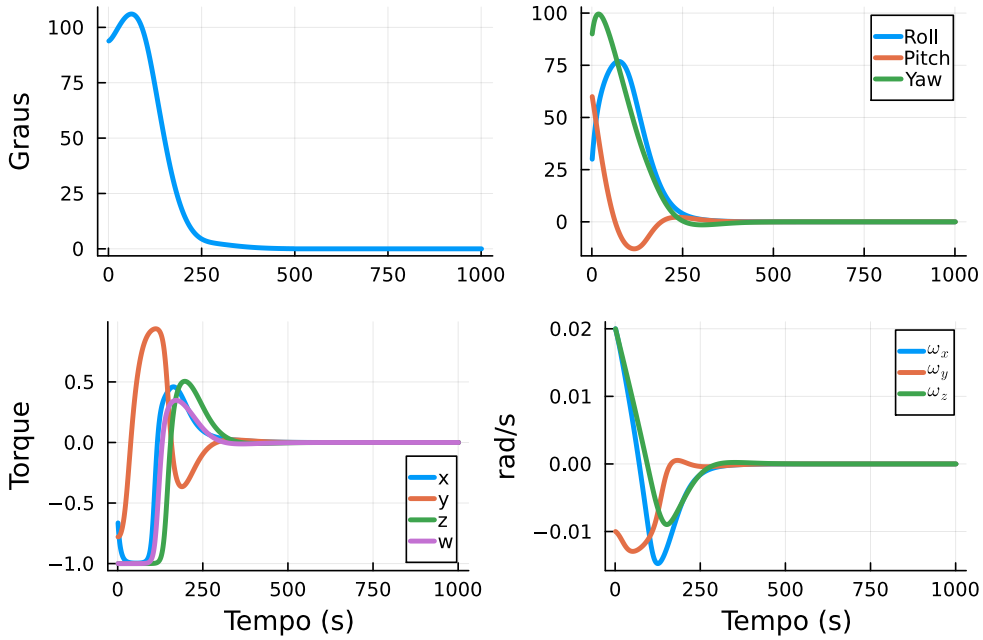
Fonte: Autor.

Figura 6.38 - SAC: simulação de torque em 4 eixos, cenário 2.



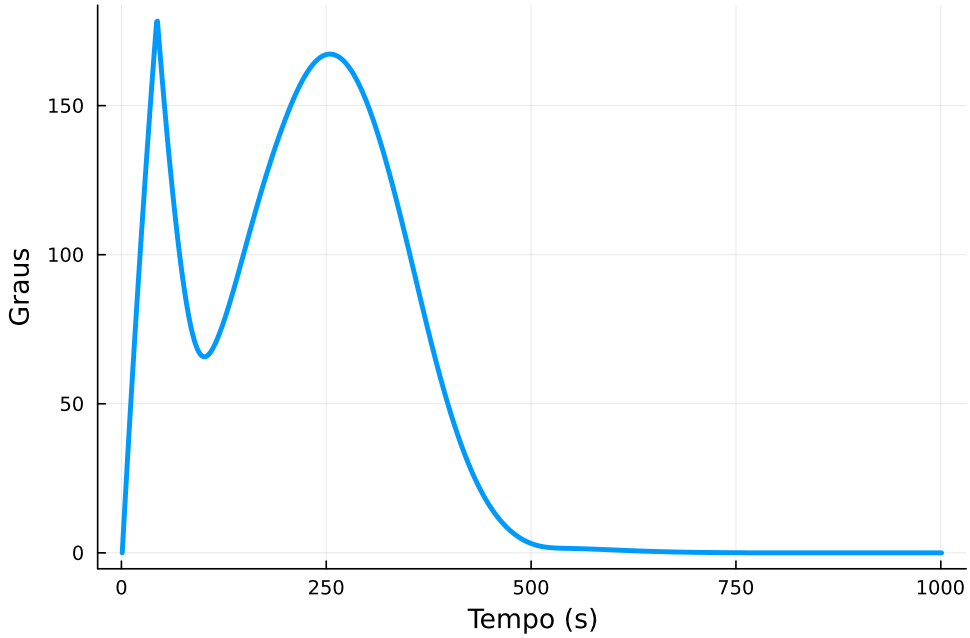
Fonte: Autor.

Figura 6.39 - SAC: simulação de torque em 4 eixos, cenário 3.



Fonte: Autor.

Figura 6.40 - SAC: erro angular em uma condição extrema, simulação de torque em 4 eixos.



Condição inicial: ângulo nulo e velocidade angular de 0.05 rad/s em cada eixo

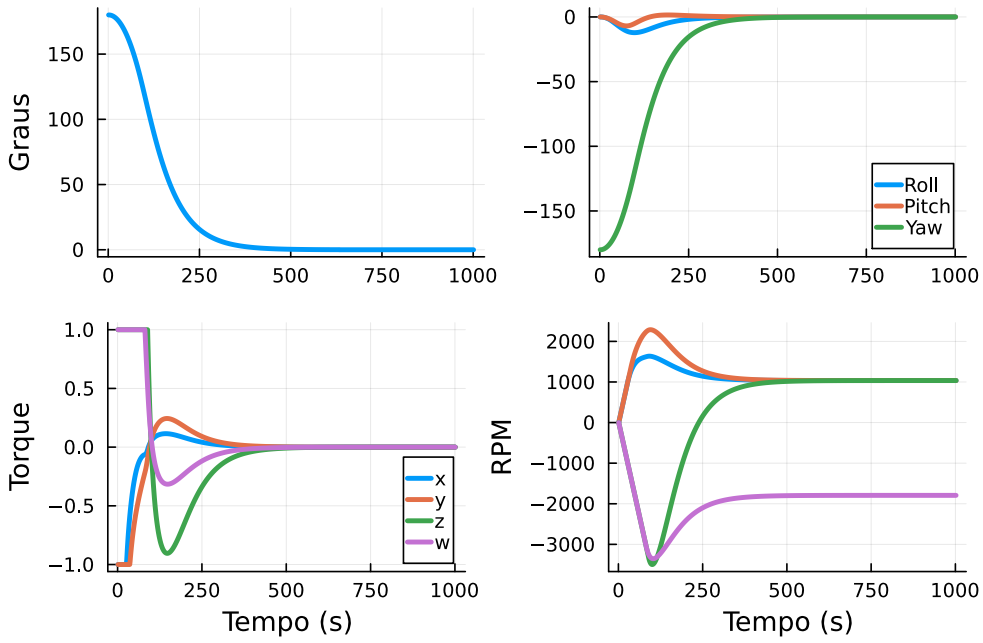
Fonte: Autor.

Esses resultados assumiram que o torque era externo ao satélite. É de interesse realizar a mesma análise incluindo a dinâmica das rodas de reação, que efetivamente fornecem o torque ao resto do satélite para o controle de atitude. O agente agora fornece a torque a ser aplicado às rodas. O valor de 6000 RPM usado no Amazonia-1 foi fixado como a velocidade máxima para todas as rodas.

As Figuras 6.41, 6.43 e 6.45 apresentam o desempenho do controlador PD nos 3 cenários e servem como referência para a resposta esperada. As Figuras 6.42, 6.44 e 6.46 fornecem a resposta do controlador SAC anteriormente apresentado com a inclusão das rodas.

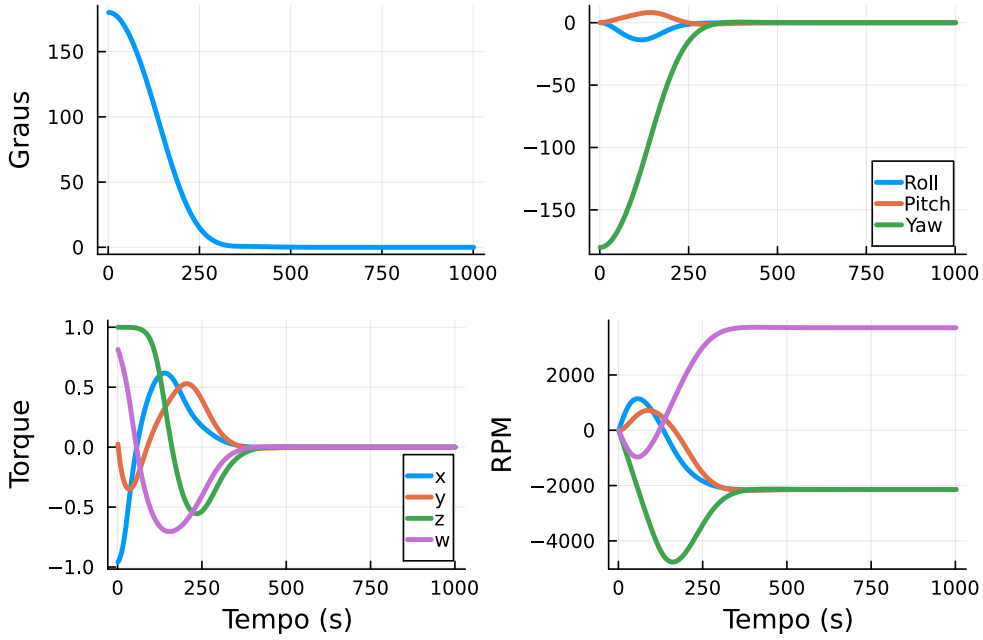
A presença das rodas altera a dinâmica pelo efeito giroscópico, de forma que pequenas distorções nos ângulos são visíveis em relação ao formato das curvas anteriores. Ainda assim, o agente foi capaz de controlar o satélite e trazê-lo ao repouso, mantendo as velocidades angulares das rodas dentro de limites aceitáveis, que nos cenários 2 e 3 foram próximas aos valores do controlador PD. Esses dois pontos são um tanto surpreendentes, visto que o agente não foi treinado nesse ambiente e, portanto, não possuía qualquer informação a respeito de como distribuir o torque entre as rodas.

Figura 6.41 - PD: simulação de torque em 4 eixos com rodas de reação, cenário 1.



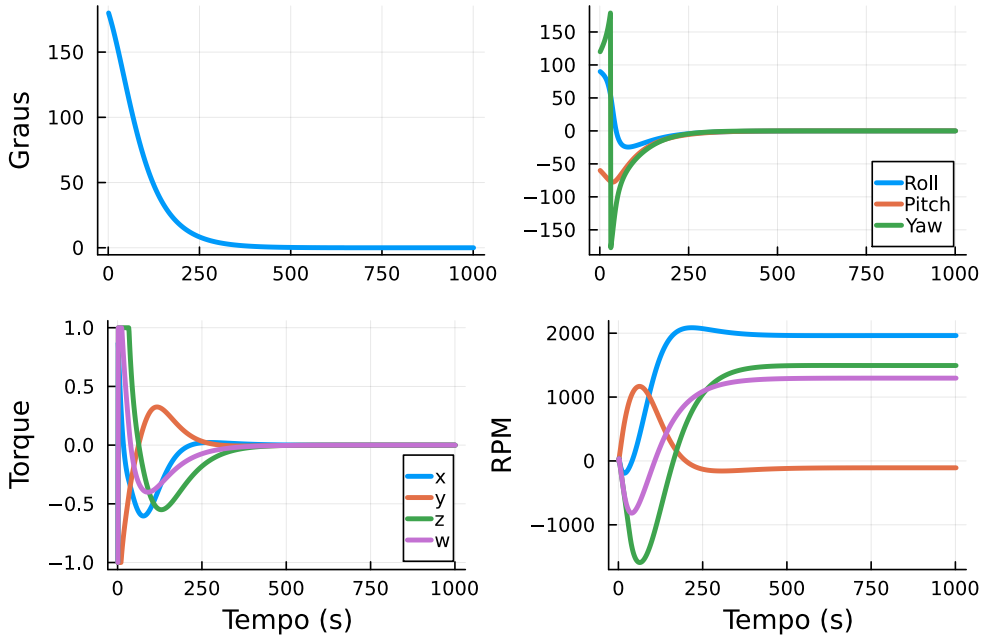
Fonte: Autor.

Figura 6.42 - SAC: simulação de torque em 4 eixos com rodas de reação, cenário 1.



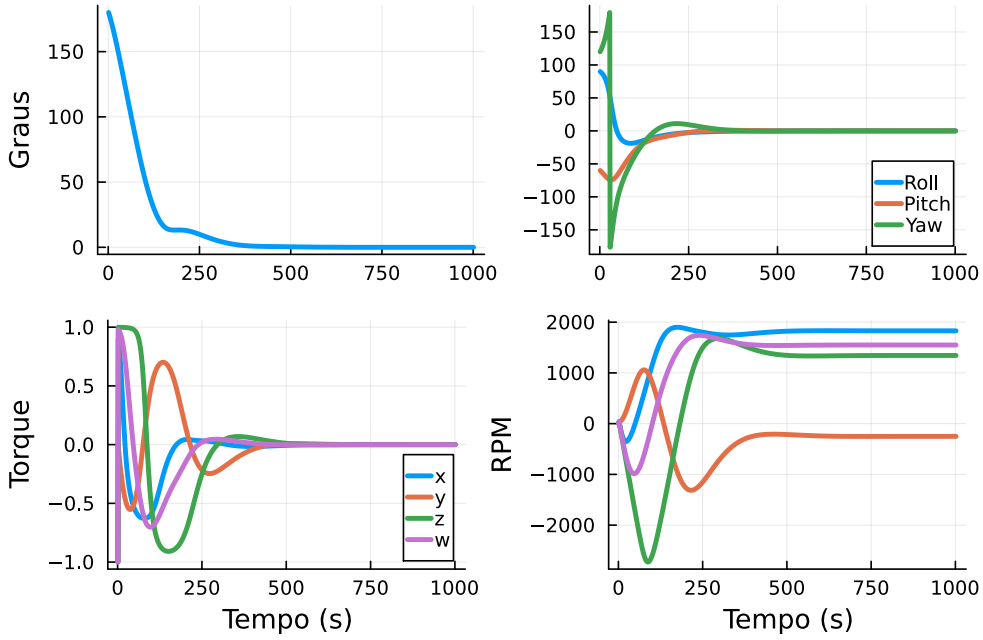
Fonte: Autor.

Figura 6.43 - PD: simulação de torque em 4 eixos com rodas de reação, cenário 2.



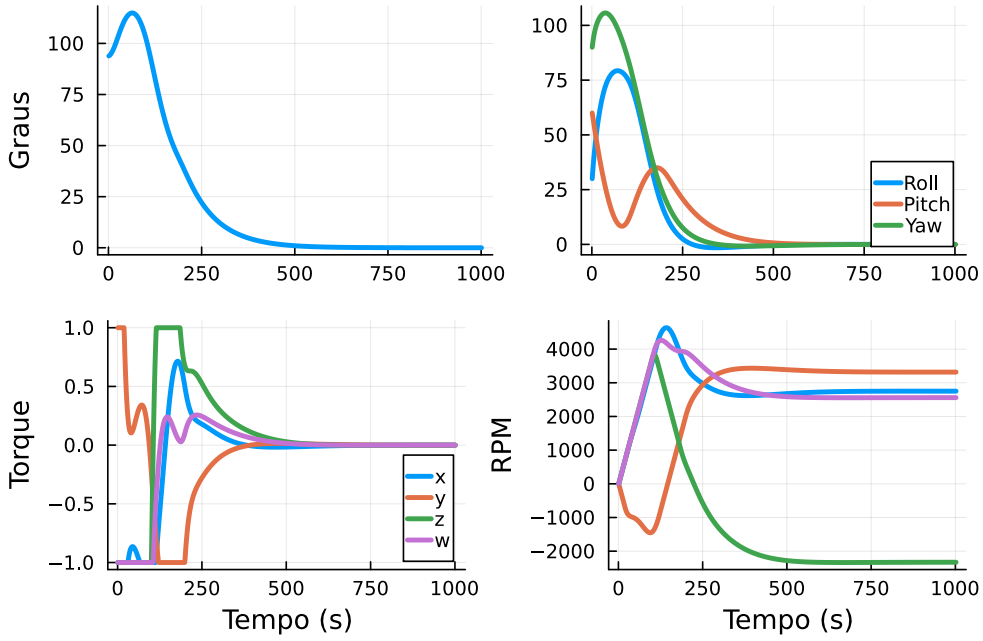
Fonte: Autor.

Figura 6.44 - SAC: simulação de torque em 4 eixos com rodas de reação, cenário 2.



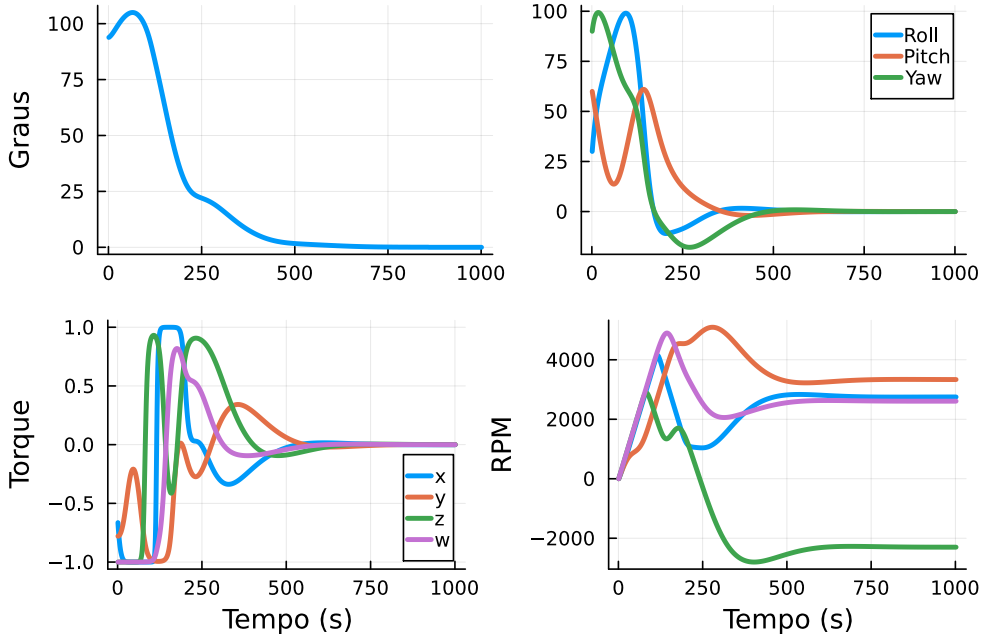
Fonte: Autor.

Figura 6.45 - PD: simulação de torque em 4 eixos com rodas de reação, cenário 3.



Fonte: Autor.

Figura 6.46 - SAC: simulação de torque em 4 eixos com rodas de reação, cenário 3.

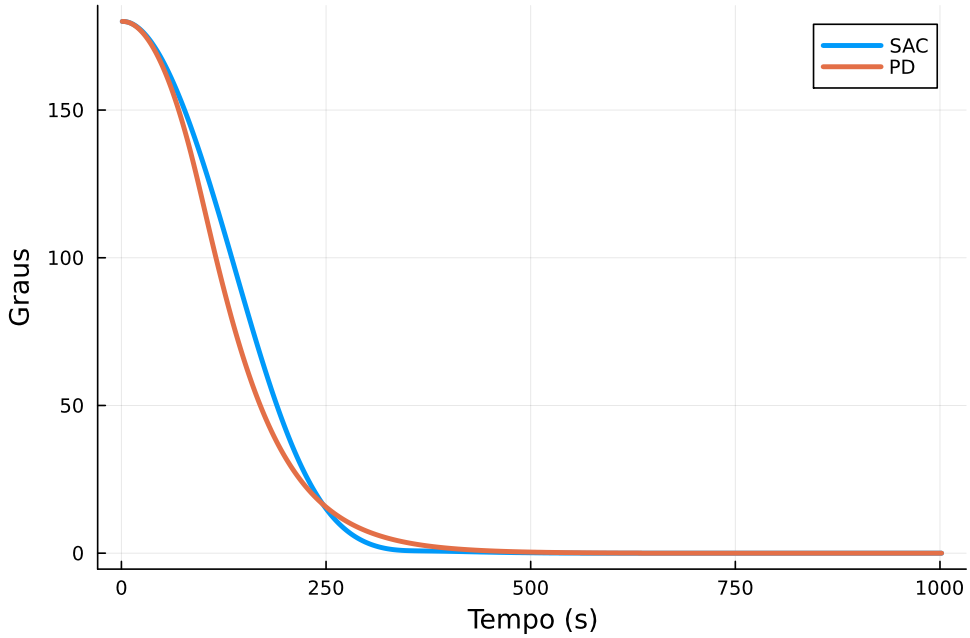


Fonte: Autor.

As Figuras 6.47, 6.48 e 6.49 comparam a comportamento do controlador PD e do SAC para os três cenários. No primeiro cenário, o SAC foi ligeiramente mais rápido, enquanto nos outros dois, apesar de apresentar um desempenho inicial superior, o ângulo sofre uma desaceleração repentina por volta de 200 segundos e então prossegue essencialmente da mesma forma que o controlador PD. Essa desaceleração é decorrente do sobrepasso do ângulo de *yaw* e se deve fisicamente ao efeito giroscópico das rodas, que cresce à medida que as rodas são aceleradas. De fato, esse sobrepasso não é identificável nas Figuras 6.38 e 6.39. Novamente, convém realçar que o agente está sendo aplicando em um ambiente ligeiramente diferente do que foi treinado e, portanto, o resultado final está sujeito a essas distorções.

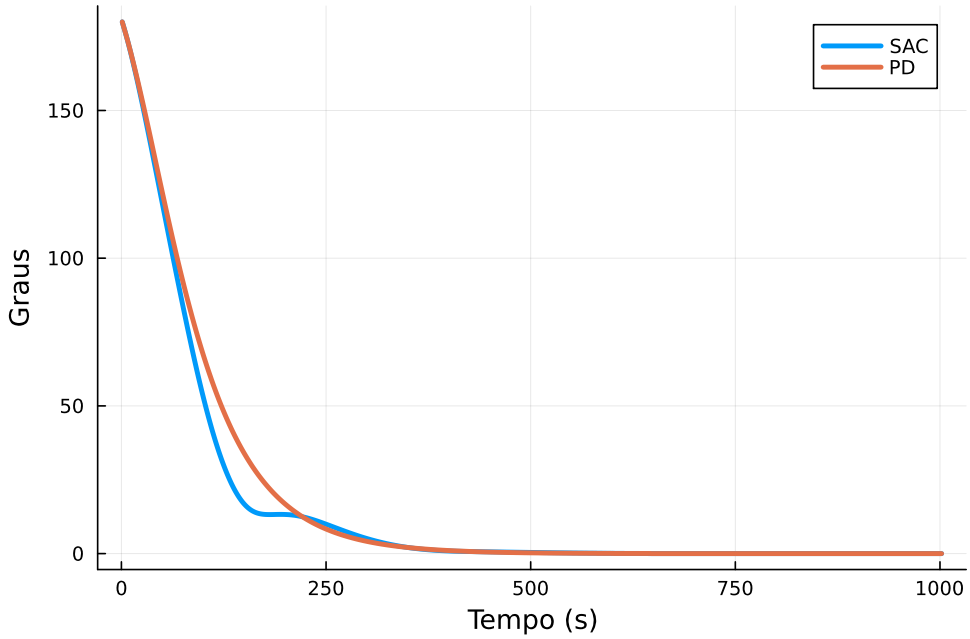
Ainda assim, o agente foi capaz de efetivamente controlar a atitude de um satélite, controlando as 4 rodas e virtualmente se igualando ao controlador PD. A rede do ator é muita pequena considerando os computadores modernos e é possível que um número relativamente baixo de neurônios, e portanto de parâmetros, seja não apenas suficiente como também necessário para a obtenção de uma solução ótima seguindo os algoritmos apresentados. A forma de uma lei de controle PID, aplicada com sucesso em muitos sistemas do mundo real, é relativamente simples de ser representada por uma rede neural com baixo número de parâmetros. Expandir a rede significa mais variáveis a serem ajustadas, o que aumenta a possibilidade de *overfitting* nos exemplos extraídos do *minibatch*. Uma rede com menos parâmetros é mais capaz de generalizar e para funções de controle, que tendem a ser bem comportadas, isso parece ser mais importante.

Figura 6.47 - Comparação do SAC e PD, cenário 1.



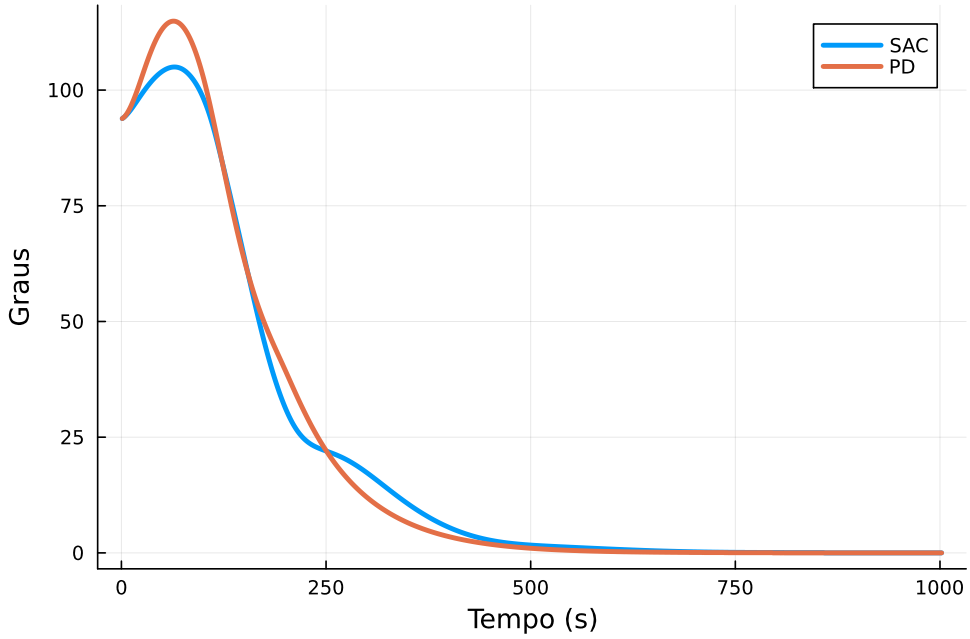
Fonte: Autor.

Figura 6.48 - Comparação do SAC e PD, cenário 2.



Fonte: Autor.

Figura 6.49 - Comparação do SAC e PD, cenário 3.



Fonte: Autor.

6.4 Inércias variáveis: controle em um único eixo

Os agentes apresentados nas seções anteriores forem treinados um de cada vez em um mesmo ambiente específico, modelados com os mesmos parâmetros, no qual foram capazes de aprender a agir de forma ótima. De maneira a explorar as capacidades do RL, é interessante avaliar a possibilidade do agente a generalizar suas ações não apenas em um dado ambiente, mas em um conjunto de ambientes com características semelhantes.

Assim, um agente seria capaz de controlar um satélite independente de suas características, aplicando um torque diferente de acordo com o veículo em questão. Esse controle seria capaz de ser aplicado a uma variedade de missões e portanto seria uma espécie de controlador universal. Uma analogia do RL com a faculdade humana de direção de veículos pode tornar mais clara essa ideia.

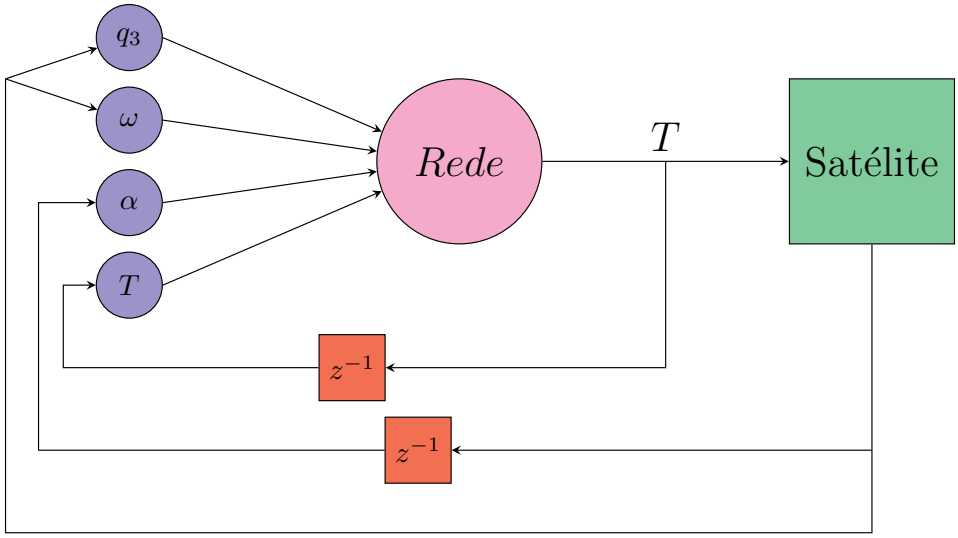
Humanos são capazes de aproveitar o conhecimento existente em um domínio de aplicação para um outro domínio que seja próximo o bastante desse primeiro. Por exemplo, uma pessoa que saiba dirigir um veículo específico pode em um intervalo de tempo curto aprender a conduzir um outro veículo com propriedades diferentes, por exemplo, com menor massa e maior potência. Isso é possível visto as semelhanças entre as tarefas: embora as dinâmicas dos ambientes sejam diferentes, as equações possuem o mesmo formato, com a mudança única dos parâmetros. Mais formalmente, a capacidade de migrar o conhecimento para outros fins constitui-se em um recente paradigma no campo de ML: o aprendizado por transferência (em inglês, *transfer learning*.

Realizar um simples mapeamento do tipo $[q_1\, q_2\, q_3\, \omega_x\, \omega_y\, \omega_z]^T \rightarrow [T_x\, T_y\, T_z]^T$ claramente não deve ser o suficiente: a resposta do ambiente será diferente de acordo com os parâmetros do satélite - a saber, o momento de inércia do eixo. Mais precisamente, essa abordagem torna o problema parcialmente observável e portanto compromete a aplicação dos algoritmos até agora utilizados. Deste modo, uma informação adicional é necessária para que o agente possa tomar uma decisão. O que se deseja efetivamente é algum tipo de *feedback*, com o intuito de permitir que o agente identifique o ambiente em questão e possa agir de acordo. Isso está em sintonia com o exemplo de direção humana: medindo-se a aceleração obtida ao pisar no acelerador, é possível adquirir uma boa ideia do veículo.

Dessa forma, de maneira a testar a validade dessa abordagem para o problema em torno de um único eixo, o estado s_t no passo t deve ser expandido para

$[sen(\theta(t)/2)\ \dot{\theta}(t)\ \alpha(t-1)\ T(t-1)]^T$, onde os dois últimos componentes indicam a aceleração angular e o torque no passo anterior. Isso equivale a realimentar o torque para a entrada da rede e utilizar a informação da aceleração angular resultante, que pode ser obtida por meios práticos a partir de giroscópios ou outros sensores. Assim, a rede possui um aspecto de uma rede neural recorrente; a arquitetura é ilustrada na Figura 6.50, onde símbolo z^{-1} denota um atraso unitário e q_3 o componente $sen(\theta/2)$ do quatérnion.

Figura 6.50 - Rede recorrente para o controle com inércia variável em um eixo.



Fonte: Autor.

Para que o agente seja capaz de realizar uma generalização e efetuar o controle com sucesso, durante o treinamento ele é exposto a uma variedade de ambientes diferentes, sendo cada episódio inicializado com parâmetros de momento de inércia e torque máximo diferentes.

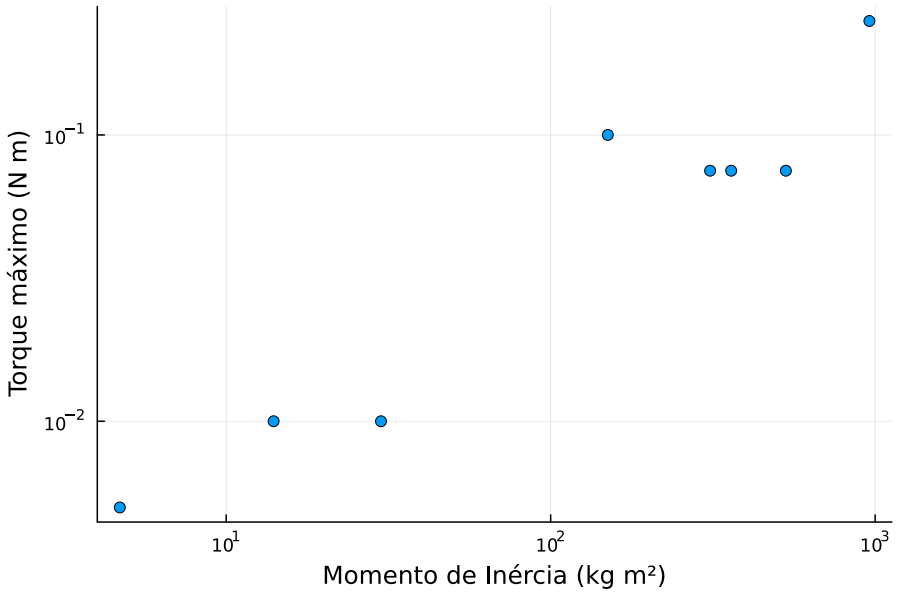
Definir valores representativos de missões reais foi uma tarefa não trivial, visto que os parâmetros de momento de inércia raramente são disponibilizados publicamente. Assim, esses valores precisaram ser grosseiramente estimados a partir de dados de massa e dimensões de missões. O SCD-1, por exemplo, projetado pelo INPE e lançado em 1993, pode ser aproximado como um cilindro de massa uniforme, o que permite calcular seu momento de inércia axial e transversal. Igualmente, o SATEC, que deveria ter sido lançado pelo foguete VLS em 2003, possuía a simples geometria de um paralelepípedo, o que permite estimar seu momento de inércia. A Tabela 6.8

fornece os parâmetros adotados. Deve-se notar que esses dois satélites foram projetados para contar com estabilização por rotação, ainda assim um atuador fictício lhes foi atribuído. A Figura 6.51 plota esses parâmetros; notar que os dois eixos estão em escala logarítmica.

Tabela 6.8 - Parâmetros dos satélites usados na simulação de inércia variável.

Satélite	$\mathbf{I_{sat}}$**(kg m²)**	$\mathbf{T_{max}}$**(N m)**	**Razão**
Kepler (eixo axial)	960,0	0,25	3840
Amazonia-1 (eixo-z)	530,7	0,075	7076
Amazonia-1 (eixo-y)	360,0	0,075	4800
Amazonia-1 (eixo-x)	310,0	0,075	4133
Hipotético	150,0	0,1	1500
SCD-1 (eixo tranversal)	30,0	0,01	3000
SCD-1 (eixo axial)	14,0	0,01	1400
SATEC	4,7	0,005	940

Figura 6.51 - Combinação momento de inércia e torque máximo para o problema de inércia variável.



Fonte: Autor.

Em razão da estabilidade e desempenho, o SAC será empregado para esse problema. Novamente, a ação realizada pelo agente correspondente ao torque, que foi normalizado para o intervalo $[-1, 1]$ para cada ambiente. Esse valor corresponde a valores

físicos diferentes de acordo com o tipo do satélite. A ideia da rede neural para esse problema, conforme vista na Figura 6.50, é que ao ter acesso à aceleração angular o agente possa descobrir as características do ambiente e assim aplicar o torque adequado.

A recompensa também precisa ser redefinida para considerar o novo problema. Claramente, a punição pelo erro do ângulo deve estar presente. Contudo, o estado agora também é composto pela último torque aplicado. É de se esperar que o controle ótimo seja suave, isso é, que o torque varie lentamente e tenha um aspecto contínuo. A realimentação do torque, entretanto, pode trazer instabilidades: variações no torque serão alimentadas a rede, que com novas entradas terá uma nova saída e assim por diante. Logo, a recompensa escolhida foi:

$$r(s_t, a_t) = -\frac{|\theta|}{\pi} - 0,25 - 0,5\ (a_t - s_{t,4})^2. \tag{6.12}$$

Visto que $s_{t,4}$ é o torque no passo anterior, o termo $(a_t - s_{t,4})^2$ pune variações bruscas da saída da rede. O termo -0,25 estimula o agente a concluir o episódio.

No controle digital, a frequência de atuação possui uma grande influência no desempenho do sistema a ser controlado. Um dado sistema, por exemplo, pode tornar-se instável se a frequência do controlador for muito baixa. A frequência de 1 Hz é o suficiente para controlar a atitude de um satélite como o Amazonia-1, mas não é de evidente se para outros satélites esse valor é adequado. De fato, quanto menor a razão I/T_{max} é de se esperar que maior seja a frequência empregada. Uma dificuldade para a abordagem do RL é que o passo Δt deve ser igual em todos os episódios, para garantir que a propriedade markoviana seja satisfeita e, portanto, não é um parâmetro que pode ser ajustado. Assim, um passo de 0,2 s, ou uma frequência de 5 Hz, foi adotado.

A simulação prossegue até que a norma do estado seja inferior a 0,0001 ou que um número de passos de 10000 seja atingido, equivalente a 2000 s. A taxa de desconto γ foi redefinida para 0,999, visto que o episódio agora possui mais passos que o exemplo unidimensional da Seção 6.1

Tendo em conta que o tipo do ambiente agora muda de um episódio para o outro, o tamanho do replay foi aumentado, de forma a garantir que experiências em vários ambientes sejam armazenadas e, portanto, sejam selecionadas no *minibatch* para realizar as atualizações.

Os hiperparêmtros são listados na Tabela 6.9. O ator é composto de 64 neurônios em uma única camada oculta e possui, portanto, 320 parâmetros.

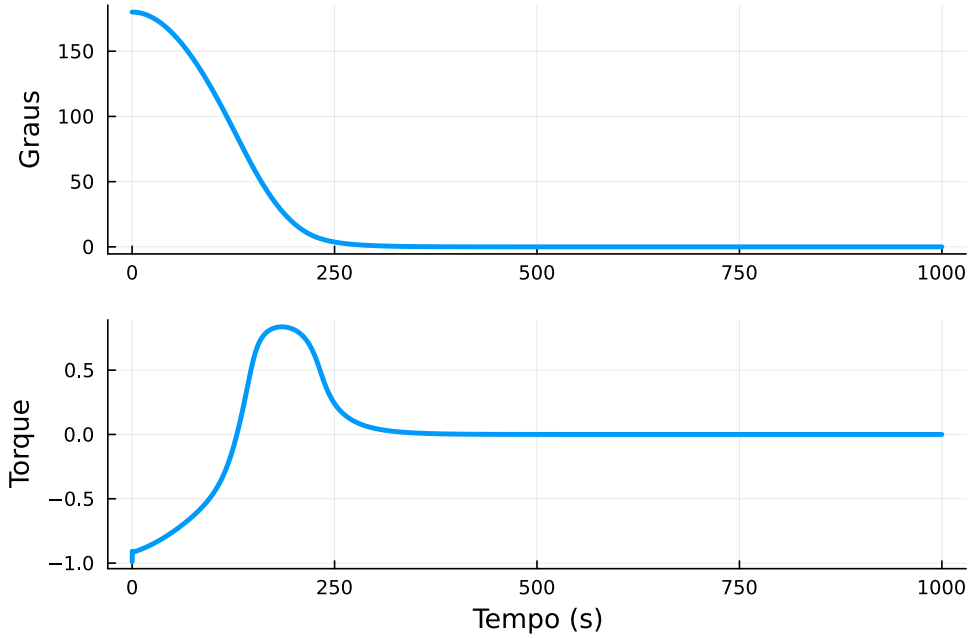
Tabela 6.9 - Hiperparâmetros usados na simulação com inércias variáveis.

Hiperparâmetro	**Valor**
Taxa de desconto	0,999
Camadas ocultas (ator)	1
Neurônios por camada (ator)	64
Função de ativação (ator)	tanh
Camadas ocultas (crítico)	2
Neurônios por camada (crítico)	128
Função de ativação (crítico)	ReLU
Tamanho do replay	10^6
Tamanho do minibatch	128
Número de passos	10^7
Passos entre atualizações	10
Otimizador	ADAM
Taxa de aprendizado	0,001

As Figuras 6.52, 6.53, 6.54 e 6.55 apresentam a evolução do ângulo e o torque aplicado para 4 satélites diferentes, todos com ângulo inicial de 180 graus e velocidade angular nula. O agente é capaz de lidar com essas 4 inércias diferentes e zerar o erro em todos os casos. Igualmente, o torque aplicado é suave, sem possuir variações bruscas, assemelhando-se em todos os exemplos. Um de seus aspectos surpreendentes é a não saturação: exceto brevemente nos primeiros momentos, o torque comandado jamais alcança o valor extremo de -1.

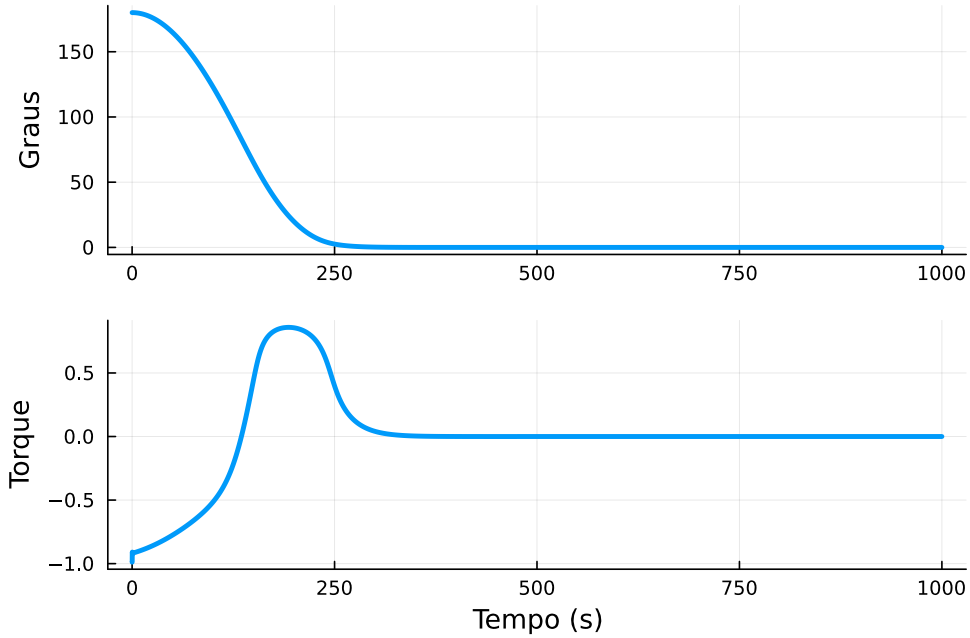
É muito possível que o controlador PD, com seus parâmetros ajustados para um satélite em particular, fosse mais rápido que o agente apresentado. Ainda assim, a capacidade de generalizar o controle para uma variedade de ambientes torna-se clara pelos gráficos.

Figura 6.52 - Inércias variáveis problema unidimensional, $I_{sat} = 960\ kg\ m^2$.



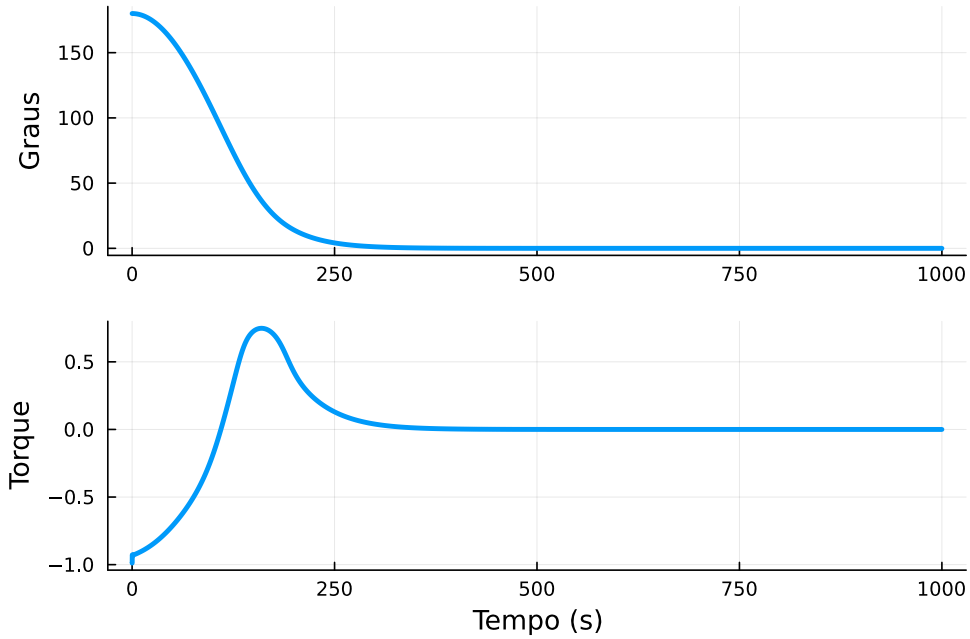
Fonte: Autor.

Figura 6.53 - Inércias variáveis problema unidimensional, $I_{sat} = 310\ kg\ m^2$.



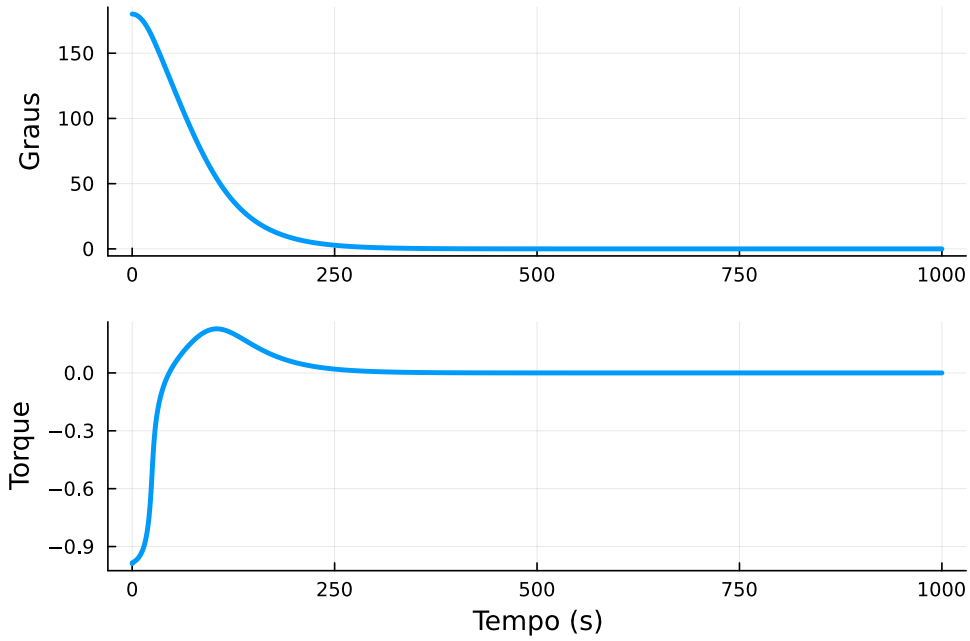
Fonte: Autor.

Figura 6.54 - Inércias variáveis problema unidimensional, $I_{sat} = 30\ kg\ m^2$.



Fonte: Autor.

Figura 6.55 - Inércias variáveis problema unidimensional, $I_{sat} = 4,7\ kg\ m^2$.

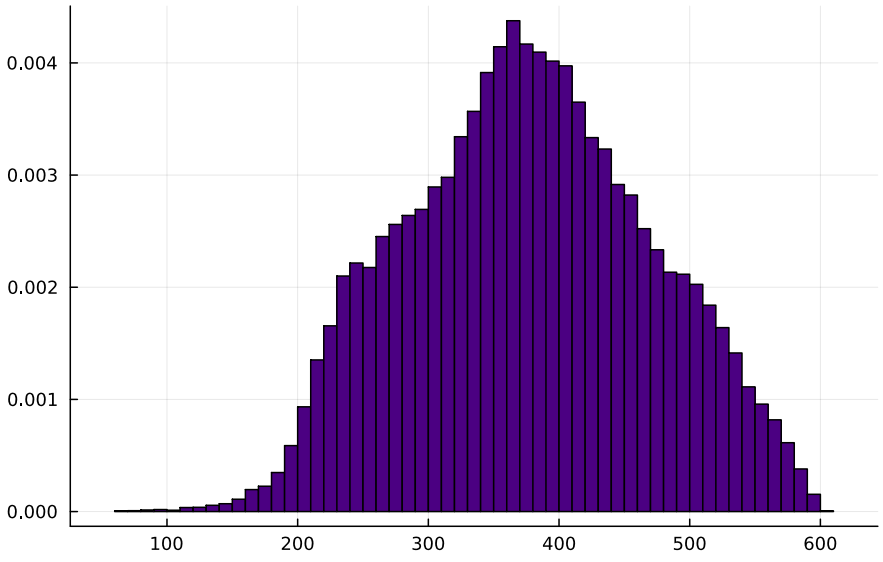


Fonte: Autor.

Novamente, em razão da não linearidade e elevado número de parâmetros, assegurar a estabilidade do agente permanece uma questão não trivial. Dada a incapacidade de métodos puramente teóricos de fornecer uma resposta definitiva a essa questão, uma alternativa é usar o método de Monte Carlo, realizando várias simulações e estimando alguma métrica de desempenho, como o tempo de assentamento.

A Figura 6.56 apresenta a distribuição do tempo de assentamento - definido como o tempo para que a norma do vetor $[sen(\theta/2)\ \dot{\theta}]^T$ seja menor que 0,0001 - usando como parâmetros do satélite o eixo y do Amazonia-1. Para a obtenção desse histograma, 50 mil episódios foram inicializados aleatoriamente, com ângulo em $[-\pi; \pi]$ rad e velocidade angular em $[-0,025\ ;\ 0,025]$ rad/s. Sendo o histograma limitado superiormente, então pode-se argumentar que o agente, ao menos para esses parâmetros do Amazonia-1, é estável.

Figura 6.56 - Histograma do tempo de assentamento do eixo y do Amazonia-1, usando o agente de inércias variáveis.



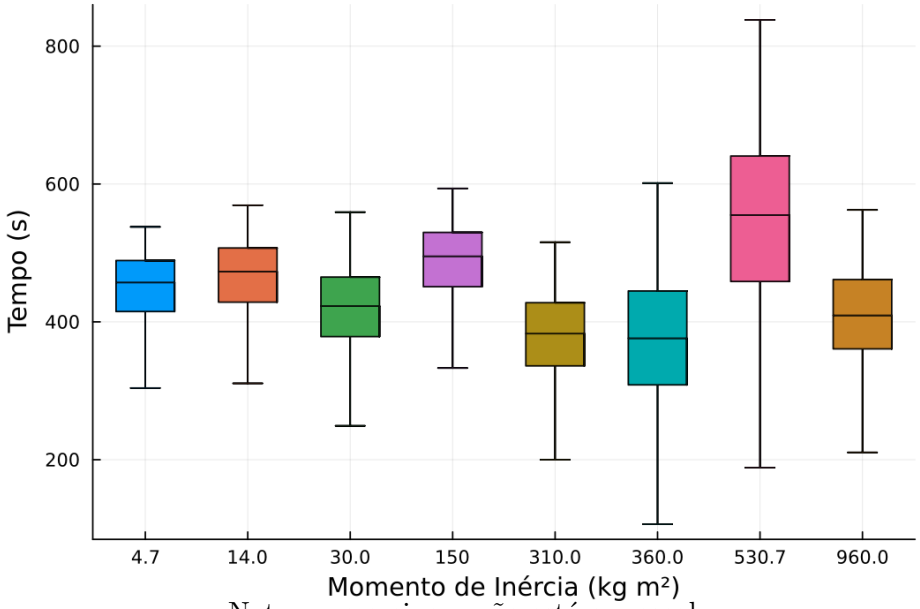
$I = 360\ kg\ m^2$ e $T = 0,075\ Nm$.

Fonte: Autor.

A Figura 6.57 traça o *boxplot* da distribuição do tempo de assentamento para todos os satélites com que o agente foi treinado, usando 50 mil episódios cada. Os *outliers* inferiores, correspondente a pontos com orientação e velocidade angular iniciais muito favoráveis, foram excluídos. Uma vez que todos são limitados superiormente,

o agente é estável para todos esses ambientes. A Figura 6.58, por sua vez, apresenta a distribuição do tempo de assentamento para um satélite diferente do que o agente foi treinado. Novamente, a solução é estável. Surpreendentemente, embora a razão entre o torque e o momento de inércia seja maior nesse caso, o tempo de assentamento médio é maior.

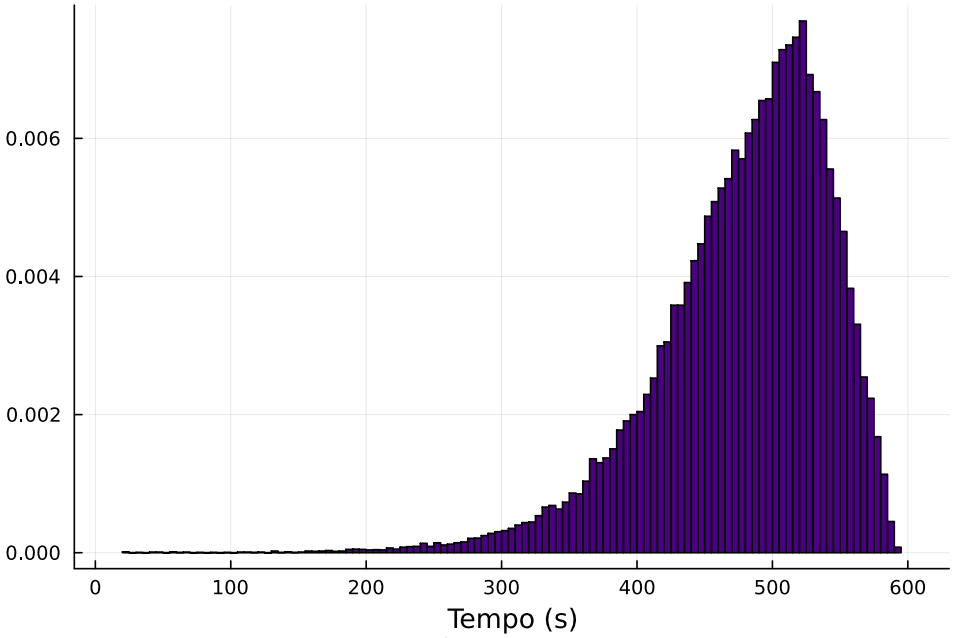
Figura 6.57 - Distribuição do tempo de assentamento para todos os satélites com que o agente foi treinado.



Notar que o eixo x não está em escala.

Fonte: Autor.

Figura 6.58 - Histograma do tempo de assentamento para um satélite diferente do treinado.



$I = 80\ kg\ m^2$ e $T = 0,05\ Nm$.

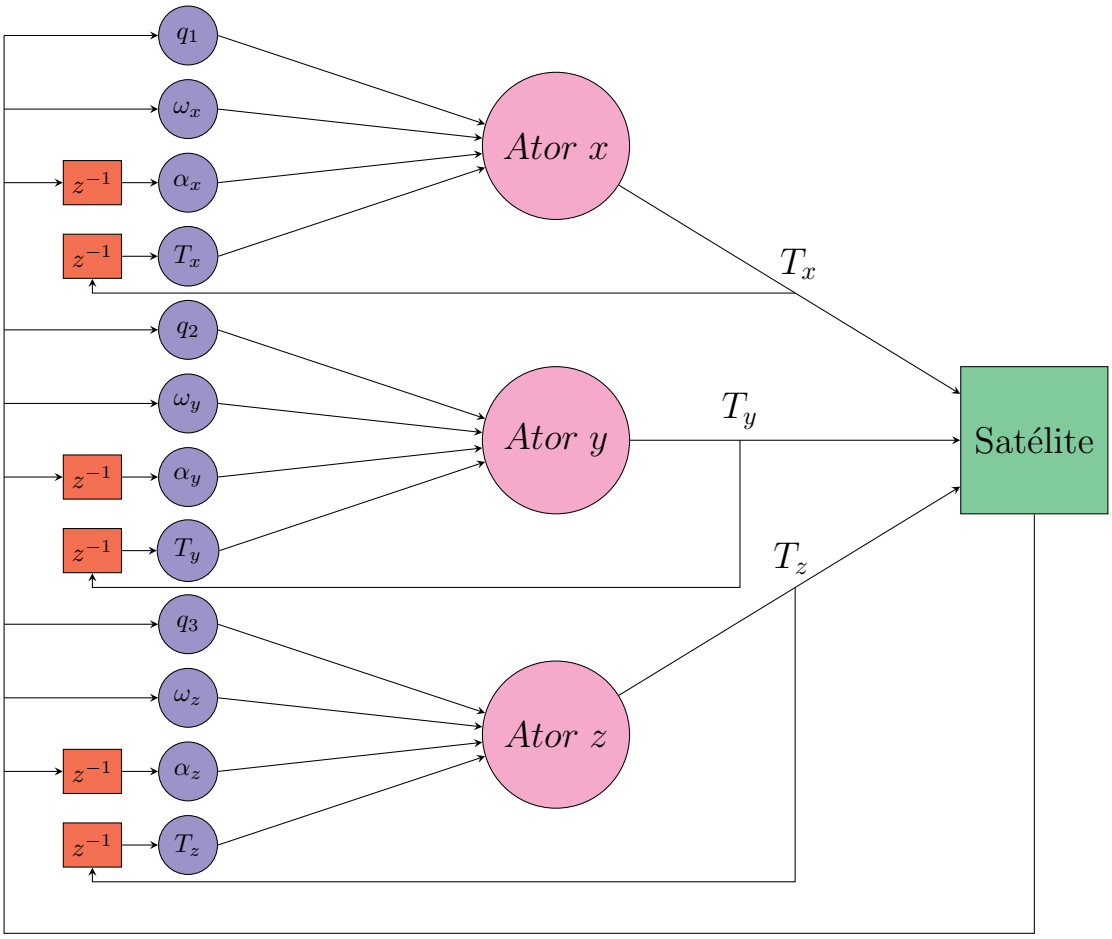
Fonte: Autor.

6.5 Inércias variáveis: controle em três eixos

O sucesso obtido com essa abordagem sugere uma expansão para o caso em três dimensões. De maneira a reutilizar a rede encontrada no problema unidimensional, a arquitetura que foi adotada está ilustrada Figura 6.59: 3 redes determinam a fração do torque nominal a ser aplicado em torno de cada eixo. Essas redes possuem todas os mesmos parâmetros encontrados na simulação anterior, mas seus vetores de entrada diferem, sendo seus elementos a parte vetorial do quatérnion, a velocidade angular, a aceleração angular e o torque aplicado correspondente aos seus respectivos eixos.

Novamente, o acoplamento das equações da dinâmica rotacional introduz possíveis complicações, à medida que um eixo pode influenciar um outro. De fato, como pode ser visto pelas Equações 3.46, 3.47 e 3.48, um eixo pode experimentar uma aceleração angular ainda que o torque aplicado seja nulo.

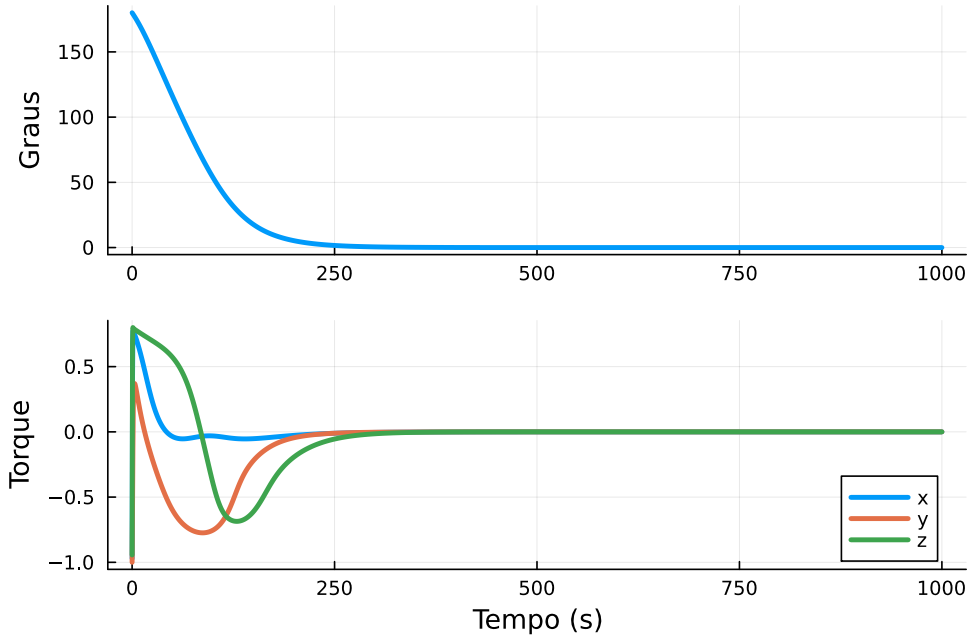
Figura 6.59 - Rede recorrente para o controle com inércia variável em três eixos.



Fonte: Autor.

As Figuras 6.60, 6.61, 6.62 e 6.63 apresentam o erro angular e o torque aplicado em cada eixo pelo agente para diferentes combinações de momento de inércia e torque máximo. A Tabela 6.6 foi usada para as condições iniciais. Os torques foram assumidos como externos e a matriz de inércia em todos os casos foi diagonal. Notar que a Figura 6.63 corresponde aos valores do Amazonia-1. Nesse caso em particular, um sobrepasso do ângulo pode ser observado.

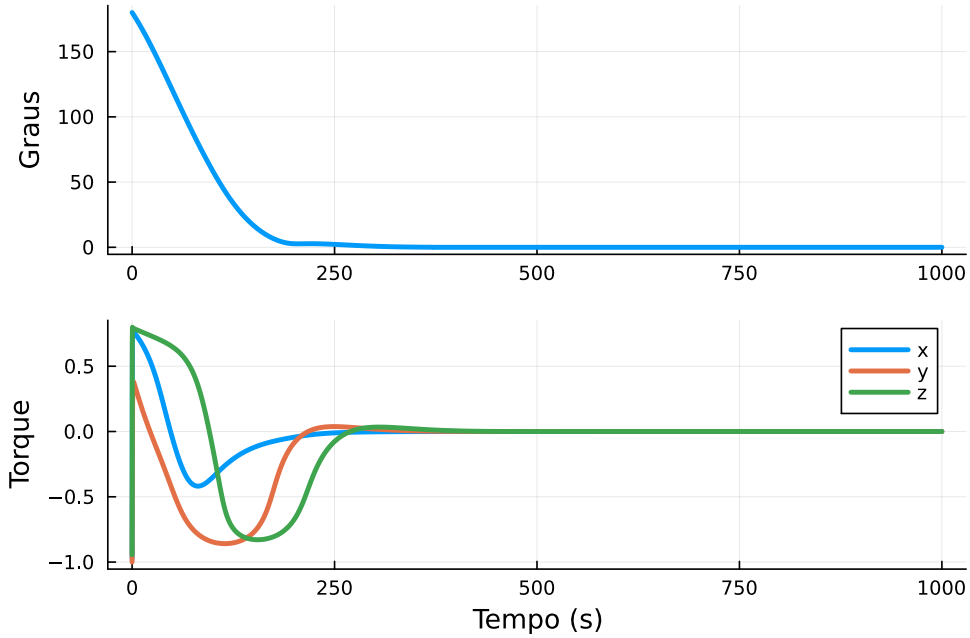
Figura 6.60 - Rede recorrente para o controle com inércia variável em três eixos.



Condições iniciais dadas pelo Cenário 2 da Tabela 6.6. Matriz de inércia diagonal, de componentes 15, 30 e 40 $kg\ m^2$, torque máximo de 0,01 Nm.

Fonte: Autor.

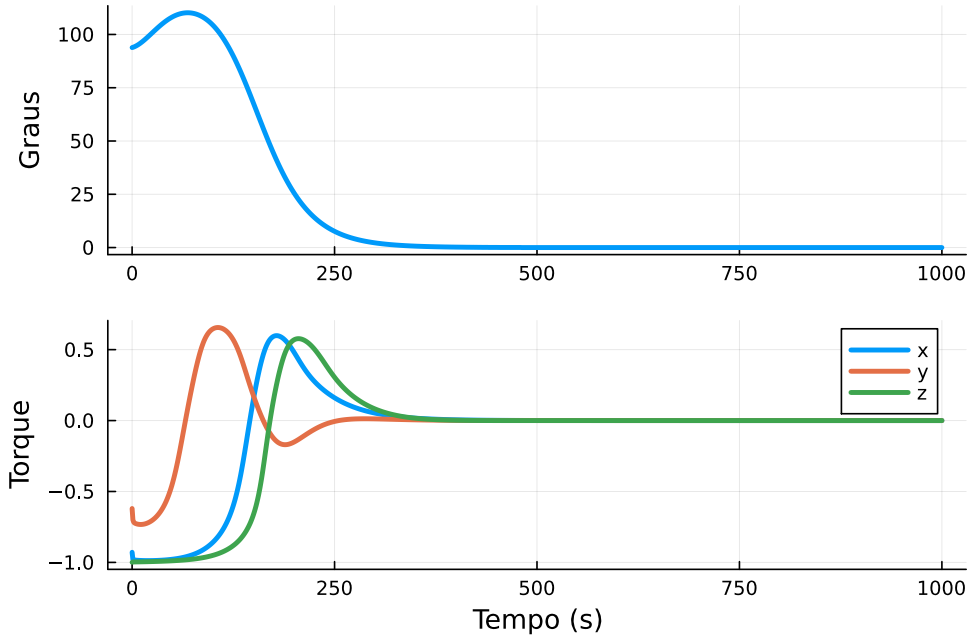
Figura 6.61 - Rede recorrente para o controle com inércia variável em três eixos.



Condições iniciais dadas pelo Cenário 2 da Tabela 6.6. Matriz de inércia diagonal, de componentes 100, 150 e 180 $kg\ m^2$, torque máximo de 0,03 Nm.

Fonte: Autor.

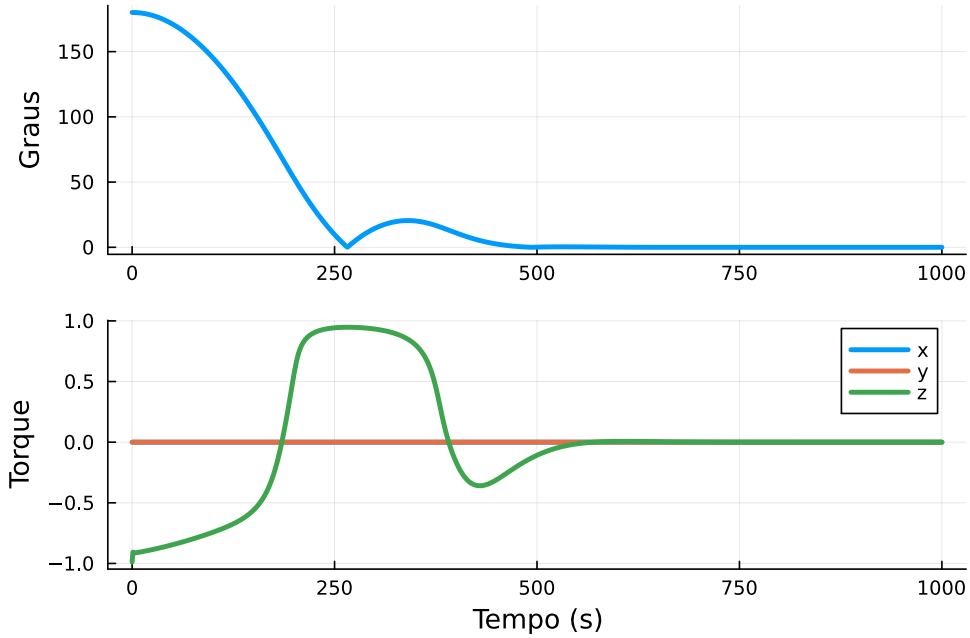
Figura 6.62 - Rede recorrente para o controle com inércia variável em três eixos.



Condições iniciais dadas pelo Cenário 3 da Tabela 6.6. Matriz de inércia diagonal, de componentes 600, 750 e 1000 $kg\ m^2$, torque máximo de 0,2 Nm.

Fonte: Autor.

Figura 6.63 - Rede recorrente para o controle com inércia variável em três eixos.



Condições iniciais dadas pelo Cenário 1 da Tabela 6.6. Matriz de inércia diagonal, de componentes 310, 360 e 530,7 $kg\ m^2$, torque máximo de 0,075 Nm.

Fonte: Autor.

Mais uma vez, o controlador é capaz de trazer o satélite ao repouso, com erro nulo para uma série de condições iniciais, adaptando-se a diferentes parâmetros de momento de inércia e torque máximo disponível. O torque aplicado também é suave, sem variações bruscas. Esses resultados indicam que essa solução é capaz de controlar satélites com massas dentro de um vasto intervalo e, portanto, comporta-se como um controlador universal. Mais especificamente, usando a linguagem do controle, o agente pode ser dito como sendo do tipo adaptativo, isso é, apto a tolerar amplas variações dos parâmetros do sistema a ser controlado.

Ao longo de todas essas simulações, o momento de inércia foi mantido o mesmo do início ao fim. Em razão da capacidade do agente de se adequar a variações nos parâmetros, é de interesse verificar seu comportamento quando o momento de inércia é repentinamente alterado. No caso real, isso poderia corresponder a um desacoplamento ou ejeção de uma carga útil, como um *lander* para pouso na superfície de um outro planeta. De forma semelhante, esse também seria o caso no acoplamento com outro veículo ou na captura de um outro objeto, como um detrito espacial. Ainda de modo alternativo, essa variação poderia decorrer da extensão de elementos flexíveis do veículo, como painéis solares ou membros articulados, sem que houvesse alteração da massa do conjunto. Essa mudança súbita dos parâmetros do sistema é um desafio para técnicas mais convencionais, mas o agente aqui discutido poderia ser capaz de rapidamente se adaptar a essa nova situação.

As Figuras 6.64, 6.65, 6.66, 6.67, 6.68 e 6.69 simulam variados cenários com mudanças repentinas de inércia. Em todos eles, o satélite com torque máximo de 0,2 Nm possui um momento de inércia inicial dado por:

$$\boldsymbol{I} = \begin{bmatrix} 600 & 0 & 0 \\ 0 & 750 & 0 \\ 0 & 0 & 1000 \end{bmatrix} kg\ m^2. \tag{6.13}$$

Essa matriz é alterada no instante $t = 100\ s$. Nas Figuras 6.64, 6.65 e 6.66, o valor de seus componentes é reduzido em 40%, de forma que a matriz se torna:

$$\boldsymbol{I'} = \begin{bmatrix} 360 & 0 & 0 \\ 0 & 450 & 0 \\ 0 & 0 & 600 \end{bmatrix} kg\ m^2. \tag{6.14}$$

De maneira semelhante, nas Figuras 6.67, 6.68 e 6.69 esse valor é aumentado para:

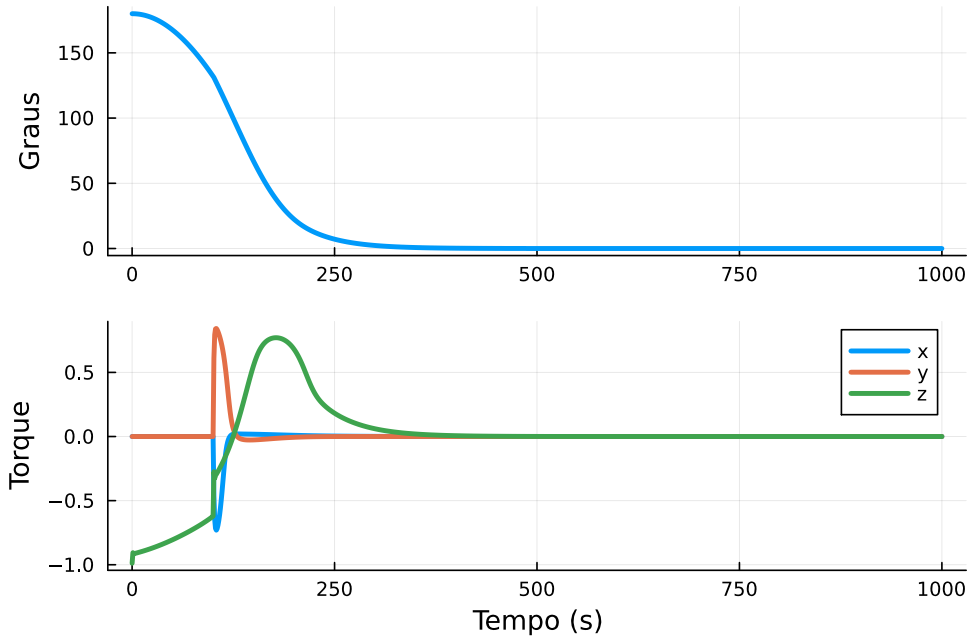
$$\boldsymbol{I''} = \begin{bmatrix} 840 & 0 & 0 \\ 0 & 1050 & 0 \\ 0 & 0 & 1400 \end{bmatrix} kg\ m^2. \tag{6.15}$$

Um termo perturbativo é somado à velocidade angular da espaçonave, de modo a simular as forças e momentos que podem aparecer durante o processo de mudança de inércia. Essa variável corresponde a um vetor no espaço escolhido aleatoriamente, com norma entre 50 a 100% da velocidade angular no momento da mudança.

Como pode ser visto, em todos os casos o agente conseguiu controlar a atitude do satélite, trazendo-o ao repouso e zerando o erro angular. O torque aplicado foi relativamente suave, embora variações abruptas ocorram no momento que a inércia é alterada, devido a variações na velocidade angular acima discutidas. Ainda assim, o agente rapidamente se adapta a esses novos parâmetros e o torque é suave nos instantes posteriores.

Isso muito provavelmente é resultado do treinamento ao que a rede foi submetida: visto que variações bruscas no torque eram punidas, a rede possui uma saída com aspecto suave. Assim, mesmo que a aceleração angular seja subitamente alterada, a rede tem a tendência a aplicar um torque muito próximo ao torque do instante anterior. Um outro ponto positivo dessa observação - com consequências práticas - é que mesmo com uma estimativa ruidosa para a aceleração angular, o torque ainda será contínuo.

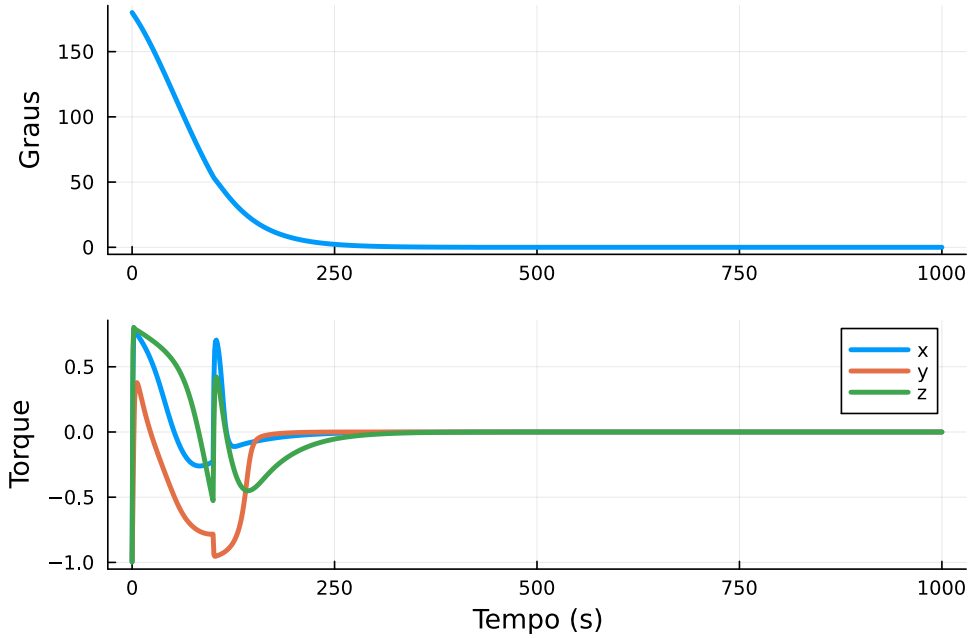
Figura 6.64 - Controle de atitude com redução do momento de inércia em $t = 100\ s$.



Condições iniciais dadas pelo Cenário 1 da Tabela 6.6.

Fonte: Autor.

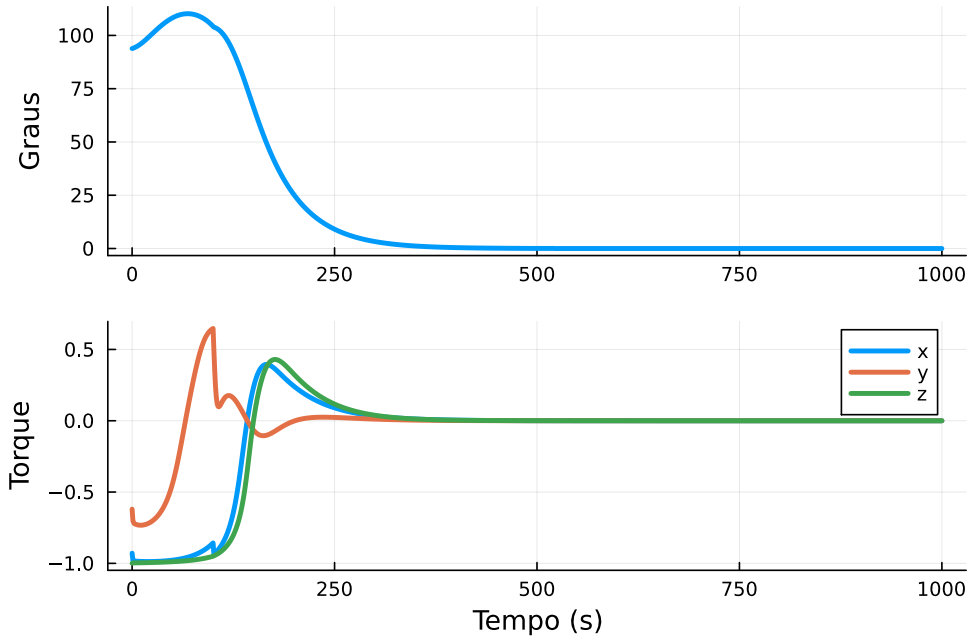
Figura 6.65 - Controle de atitude com redução do momento de inércia em $t = 100\ s$.



Condições iniciais dadas pelo Cenário 2 da Tabela 6.6.

Fonte: Autor.

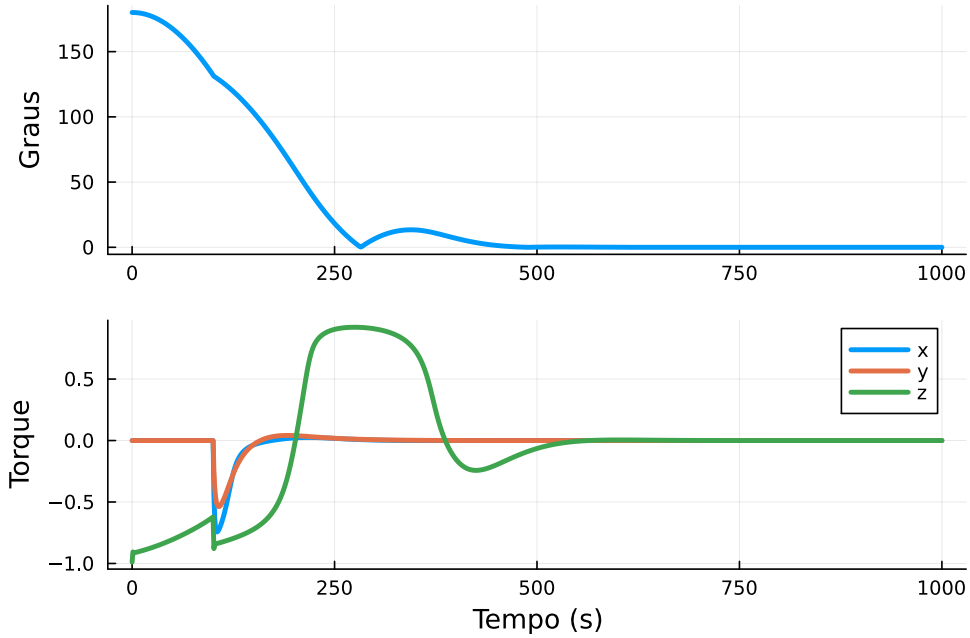
Figura 6.66 - Controle de atitude com redução do momento de inércia em $t = 100\ s$.



Condições iniciais dadas pelo Cenário 3 da Tabela 6.6.

Fonte: Autor.

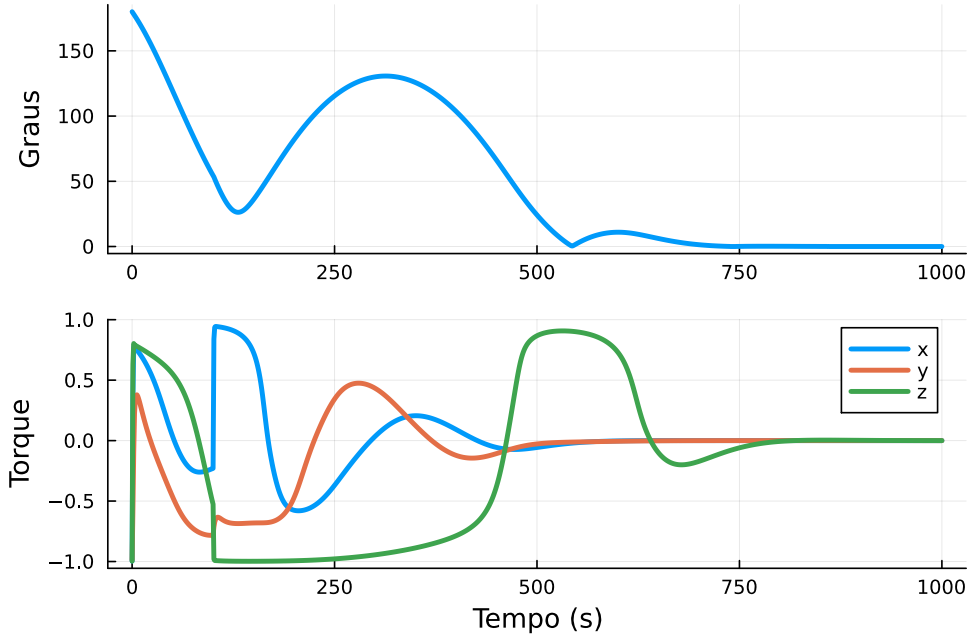
Figura 6.67 - Controle de atitude com aumento do momento de inércia em $t = 100\ s$.



Condições iniciais dadas pelo Cenário 1 da Tabela 6.6.

Fonte: Autor.

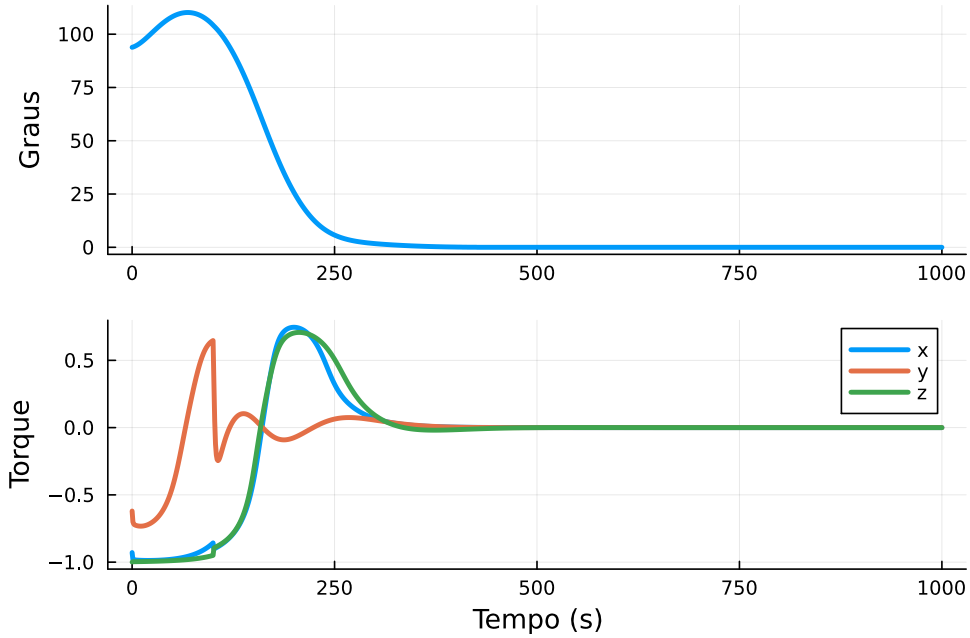
Figura 6.68 - Controle de atitude com aumento do momento de inércia em $t = 100\ s$.



Condições iniciais dadas pelo Cenário 2 da Tabela 6.6.

Fonte: Autor.

Figura 6.69 - Controle de atitude com aumento do momento de inércia em $t = 100\ s$.

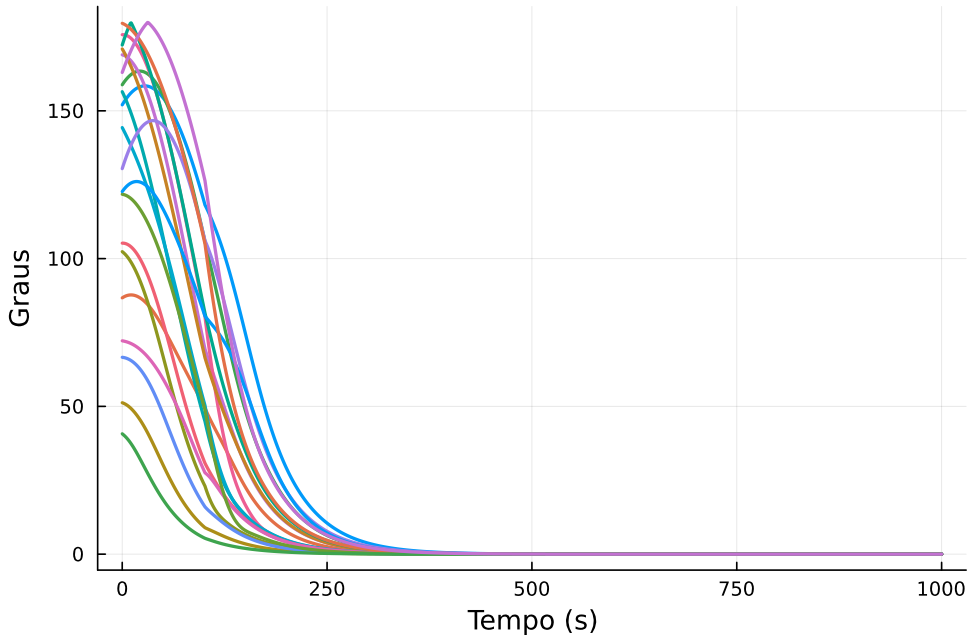


Condições iniciais dadas pelo Cenário 3 da Tabela 6.6.

Fonte: Autor.

Evidentemente, esses cenários correspondem a condições iniciais específicas. A própria velocidade angular após a alteração da inércia é uma variável aleatória. Seria interessante investigar o comportamento desse agente para uma série de cenários. A Figura 6.70 apresenta a mesma situação de redução de inércia no tempo $t = 100\ s$, mas com 20 condições iniciais diferentes. A Figura 6.71 contém informação similar, mas para o aumento de inércia. Como é tornado claro pela análise dos gráficos, todas as curvas convergem para erro nulo, sem apresentar caráter oscilatório. Igualmente, como esperado, é nítido que os cenários de aumento de inércia possuem maiores tempos de assentamento.

Figura 6.70 - Simulações superpostas com condições iniciais aleatórias e redução do momento de inércia em $t = 100\ s$.

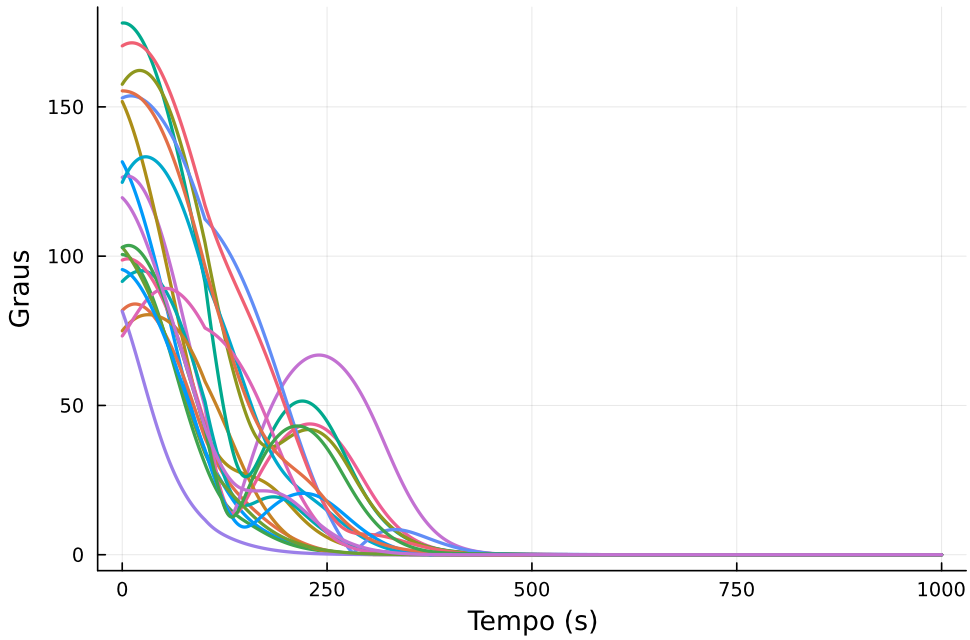


O gráfico apresenta 20 simulações ao todo.

Fonte: Autor.

Esses resultados fornecem um argumento adicional em favor da estabilidade do agente encontrado, visto que este foi capaz controlar o satélite mesmo após variar o momento de inércia dentro de um extenso intervalo para várias condições iniciais. Portanto, baseando-se na simples arquitetura exposta na Figura 6.50 o problema de controle de atitude com parâmetros variáveis - ou mais precisamente, desconhecidos pelo agente - foi resolvido de maneira satisfatória. Tendo que conta o tamanho da rede de 64 neurônios, ou 320 parâmetros, sua execução no *hardware* real seria per-

Figura 6.71 - Simulações superpostas com condições iniciais aleatórias e aumento do momento de inércia em $t = 100\ s$.



O gráfico apresenta 20 simulações ao todo.

Fonte: Autor.

feitamente viável considerando a capacidade atual dos computadores embarcados. Muito possivelmente, o tamanho da rede poderia ser reduzido ainda mais usando técnicas mais tradicionais de ML.

A simulação realizada - assim como todas as outras ao longo desse trabalho - possui um caráter estocástico, isso é, muito certamente uma outra simulação, com os mesmos hiperparâmetros, produziria resultados diferentes. Isso é inerente ao RL: a necessidade de exploração requer uma aleatoriedade e o replay de memória introduz outro componente randômico, à medida que as interações são sorteadas para atualizar o ator e o crítico.

Outros elementos essenciais, como os hiperparâmetros, o formato da recompensa e a escolha do vetor estado, são escolhidos de maneira um tanto quanto livre e podem também ter grande influência no resultado final. É evidente que para problemas de controle do mundo real, onde más ações têm consequências desastrosas, assegurar as propriedades do controlador e garantir a estabilidade são objetivos fundamentais. Provar de maneira rigorosa que esses requisitos são atendidos para os agentes apresentados ainda está além dos métodos disponíveis. Contudo, os exemplos apre-

sentados, em conjunto com certas heurísticas adotadas - a ausência do viés na rede, função de ativação ímpar, dentre outras -, indicam a capacidade dos agentes em resolver problemas desafiadores de controle.

Destarte, como visto ao longo desse capítulo, a abordagem moderna do RL, combinando o aprendizado do agente ao interagir com o ambiente com a generalização por redes neurais, pode ser uma alternativa para problemas convencionais do controle, em especial para os seus mais desafiadores. Na área espacial, isso inclui o controle de atitude, mas muito certamente problemas de outra natureza, como realização de manobras de *rendezvous* ou pousos controlados, também são aplicações possíveis. À proporção que missões futuras cada vez mais desafiadoras e exigindo menor intervenção humana sejam concebidas e planejadas, as técnicas do RL podem mostrar-se uma escolha atraente para atender a essas demandas.

7 CONCLUSÕES

O objetivo desse trabalho foi aplicar o paradigma do RL por meio de seus algoritmos mais modernos ao problema de controle de atitude de um satélite. Os agentes encontrados foram bem sucedidos em atender esse objetivo. Como visto anteriormente, as soluções foram capazes de zerar o erro angular e aplicar um torque suave, conforme esperado de uma resposta ótima. Não apenas isso, mas em alguns desses problemas o agente encontrou respostas não triviais: no caso unidimensional reconheceu uma combinação favorável entre velocidade angular e ângulo de mesmos sinais e no caso com inércias variáveis rapidamente se adaptou a mudanças repentinas de parâmetros do sistema. Além disso, a rede do ator é relativamente pequena, apresentando apenas uma camada oculta, o que favorece sua implementação no *hardware* real. Esses pontos confirmam o potencial do RL para problemas mais tradicionais de controle, em especial para aqueles que sejam mais desafiadores para técnicas convencionais.

Dentre os três algoritmos apresentados, o SAC mostrou-se o mais robusto, obtendo o melhor desempenho nas tarefas de controle. O DDPG e o TD3, por outro lado, apresentaram instabilidades e muitas variações nos problemas tridimensionais. A razão desses resultados provavelmente está nos fundamentos dos algoritmos: o SAC parte do princípio de máxima entropia enquanto os outros dois baseiam-se no teorema do gradiente da política determinística.

O princípio de máxima entropia busca simultaneamente maiores retornos com uma aleatoriedade da política, o que permite a exploração do ambiente. Sendo a temperatura ajustada automaticamente, o agente é incentivado inicialmente a ser mais exploratório, tendendo ao proveito à medida que o número de interações cresce. O DDPG e o TD3, por outro lado, inserem a exploração por um ruído gaussiano, cuja variância não é ajustada ao longo do tempo. O desempenho superior do SAC, logo, pode ser atribuído ao ajuste adequado entre exploração e proveito. Essa reflexão sugere que esse antagonismo e seu equilíbrio ainda é uma questão primordial no contexto do RL, mesmo em seus algoritmos mais modernos.

Ao longo desse desenvolvimento algumas outras limitações inerentes ao RL de uma forma mais geral também foram identificados. No paradigma do RL, a escolha dos termos presentes no vetor estado que descreve o ambiente e a função de recompensa associada a interação possuem influência significativa no resultado final. Contudo, o formato exato desses elementos é até certo ponto arbitrário.

O estado, em geral, deve ser definido de modo a garantir que a propriedade marko-

viana seja satisfeita. No desenvolvimento apresentado, optou-se pela representação de atitude em quatérnions, mais especificamente, os componentes vetoriais do quatérnion, visto que o elemento escalar pode ser encontrado observando que a norma é unitária. Embora essa escolha tenha sido julgada como adequada para o problema, dada a multiplicidade de representações existentes, é possível que alguma outra seja mais favorável em termos de velocidade aprendizado e de desempenho no contexto do RL.

A recompensa talvez seja ainda mais complexa de ser definida. Seu requisito é guiar o agente rumo a um comportamento ótimo, todavia várias funções podem satisfazer essa condição. Uma recompensa de formato $-|\theta|$ foi empregada para o erro da atitude, visto que seu máximo ocorre em $\theta = 0$, ou seja, o agente é estimulado a atingir o repouso. Ainda assim, outras funções como $-|\theta|^2$, $-|sen(\theta/2)|$ dentre muitas outras concebíveis atendem a esse requisito, bastando serem decrescentes com $|\theta|$. Ademais, termos adicionais envolvendo $\vec{\boldsymbol{\omega}}$ poderiam ter sido inclusos. A dificuldade de definir uma boa recompensa também está ligada a uma razão mais sutil: seu formato deve ser geral o bastante para indicar o objetivo a ser alcançado e específico o bastante para acelerar o aprendizado, mas sem induzir o agente a um comportamento particular que não seja ótimo no contexto da tarefa.

De certa forma, essas duas dificuldades também estão presentes no contexto mais geral do controle clássico e do controle ótimo. A abordagem do RL moderno, contudo, traz algumas novas dificuldades. O uso de aproximadores não lineares como redes neurais por um lado permite a representação de funções complexas para uma série de tarefas desafiadoras; por outro lado dificulta verificar que as propriedades da rede sejam apropriadas para o problema. Apesar dos métodos modernos de otimização de RNAs serem considerados eficientes, o problema fundamental de ficar preso em um extremo local ao invés de um global permanece presente.

Para o controle, mais particularmente, deve-se garantir que o sistema obtido seja estável. O controlador PD pode ser analisado a partir das ferramentas do controle clássico, como lugar das raízes e métodos no domínio da frequência, e sua estabilidade provada. Evidentemente, o controle em aplicações espaciais requer um conhecimento preciso acerca de suas propriedades. Sendo uma RNA uma função não linear, garantir sua estabilidade torna-se um problema teórico mais intrincado. Os casos apresentados, embora forneçam indícios muito fortes da estabilidade, estão longe de serem uma prova rigorosa.

Ainda assim, o controlador obtido pelas técnicas do RL foi capaz de superar o PD no

cenário unidimensional, apresentando tempos de assentamento ligeiramente inferiores para mesmas condições iniciais e identificando a condição singular de velocidade angular e ângulos elevados e de mesmo sinal, como discutido anteriormente. Nos dois cenários tridimensionais considerados, a diferença foi mínima. Ainda assim, o controlador desacoplado, com cada rede treinada separadamente, foi superior ao PD em termos de tempo de assentamento. Novamente, isso sugere que dotar o ator de certas características próprias para o problema influencia significativamente o desempenho da política encontrada.

A eliminação do viés na RNA do autor, além de ser uma forma de acelerar o aprendizado, garante que o torque aplicado seja nulo quando o estado for nulo, o que confere uma certa estabilidade à solução. Do mesmo modo, a escolha da função de ativação do tipo tangente hiperbólico acelerou o aprendizado, tendo em conta a simetria espacial do problema. Essas considerações de inserir conhecimento prévio são frequentemente importantes no contexto de *Machine Learning*, mas para o problema de controle adquirem uma importância ainda maior: garantir que a solução possua certas propriedades desejadas. Portanto, trabalhos futuros devem esforçar-se em identificar particularidades do problema que permitam utilizar um conhecimento prévio no agente.

Logo, as dificuldades fundamentais enfrentadas para o controle a partir das técnicas do Aprendizado por Reforço devem ser vistas não como específicas do ambiente analisado - que de fato é relativamente simples -, mas sim como uma questão de alcance mais geral envolvendo diferenças elementares entre o controle e o RL. O controle parte do princípio de estabilidade: os sistemas de seu interesse devem primariamente ser estáveis. O RL, por outro lado, parte da noção de maximizar um retorno esperado. Idealmente, essas duas abordagens devem levar à mesma solução, com a recompensa incentivando o agente a ser estável. Ainda assim, essa noção é transmitida de forma indireta. Garantir a estabilidade dos agentes obtidos pelo RL permanece uma questão ainda aberta e, muito possivelmente, é a questão mais importante a ser resolvida para sua difusão em aplicações associadas ao controle convencional.

Embora resultados teóricos a respeito do controle com RNA não estejam disponíveis, é a percepção do autor que novos desenvolvimentos devem incluir argumentações do controle convencional. Uma fusão de ideias desses dois campos pode ser extremamente proveitosa, combinando as capacidades do RL moderno com o suporte teórico do controle. Por exemplo, ainda que os algoritmos apresentados tenham obtido su-

cesso usando a formulação de redes neurais, eles são agnósticos no que concerne a parametrização do ator e do crítico, contanto que a diferenciabilidade seja garantida. É possível que uma parametrização alternativa possa ser eficaz para o problema de controle de atitude, ao mesmo tempo que garante que condições elementares sejam atendidas.

Novamente, convém realçar que os algoritmos descritos foram desenvolvidos em anos recentes. O campo de RL como um todo tem-se mostrado prolífico; outros focos de investigação também oferecem perspectivas interessantes. Como exemplos o chamado *Safe Reinforcement Learning*, que busca, primariamente, encontrar políticas seguras e o *Offline Reinforcement Learning*, no qual o agente deve aprender a agir a partir de um histórico de interações existente, sem efetuar diretamente suas ações no ambiente.

No mundo real, o custo de interagir com o ambiente é elevado, visto que ações não ótimas podem trazer consequências desastrosas. Desse modo, há um esforço constante no campo de RL no sentido de encontrar novas abordagens que façam o agente aprender o mais rápido possível com um número mínimo de interações com o ambiente.

Os três algoritmos discutidos nesse trabalho são todos do tipo *model-free*, isso é, não aprendem diretamente a dinâmica do ambiente, ainda que essa seja representada implicitamente pela função de valor $Q_\pi(s, a)$. Os métodos *model-based*, por outro lado, aprendem também a dinâmica do ambiente, o que confere uma capacidade teórica ao agente de planejar, isso é, simular a evolução do ambiente dentro de um horizonte futuro e assim selecionar suas ações ou simplesmente gerar um número maior de transições para o treinamento do ator e do crítico. O RL *model-based* ainda está menos desenvolvido, mas é possível que avanços futuros permitam abordagens mais eficientes ao problema de controle.

Essa visão geral delineia oportunidades e possíveis linhas de investigação para trabalhos futuros. Em particular, como sugestões estão a análise de estabilidade e propriedades fundamentais dos agentes obtidos, uso de redes neurais com arquiteturas alternativas e outras parametrizações da política e avaliação de algoritmos e abordagens do RL diferentes da apresentadas nesse trabalho. A capacidade do RL de resolver problemas desafiadores além do alcance de técnicas mais tradicionais deve encorajar mais investigações na área. Missões espaciais futuras terão que cumprir tarefas complexas e necessitarão ser mais autônomas ao longo de sua missão. As modernas técnicas de RL são uma alternativa viável para alcançar-se esse objetivo.

Deve-se observar que atualmente os recursos em termos de *hardware* muito certamente estão disponíveis para a implementação real das soluções apresentadas, especialmente visto o tamanho pequeno da rede. Não apenas isso, em razão da necessidade de confiabilidade e à austeridade do ambiente espacial, os computadores presentes nos veículos espaciais ainda estão algumas décadas atrás dos modelos comerciais acessíveis ao grande público. À medida que em anos vindouros esses avanços alcançarem o mercado espacial, essa potência computacional adicional poderá ser usada para a implementação de soluções mais complexas, tal como o controle a partir de redes neurais.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLISON, J.; WEST, M.; GHOSH, A.; VEDANT, F. Reinforcement learning for spacecraft attitude control. In: INTERNATIONAL ASTRONAUTICAL CONGRESS, 70. **Proceedings...** Washington DC, 2019. 13

BARTO, A. G.; SUTTON, R. S.; ANDERSON, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. **IEEE Transactions on Systems, Man, and Cybernetics**, n. 5, p. 834–846, 1983. 7, 44

BEALS, G. A.; CRUM, R. C.; DOUGHERTY, H. J.; HEGEL, D. K.; KELLEY, J. L.; RODDEN, J. J. Hubble space telescope precision pointing control system. **Journal of Guidance, Control, and Dynamics**, v. 11, n. 2, p. 119–123, 1988. Disponível em: <<https://doi.org/10.2514/3.20280>>. 15

BELLMAN, R. E. **Dynamic programming**. Princeton: Princeton University Press, 1957. 40

BOHN, E.; COATES, E. M.; REINHARDT, D.; JOHANSEN, T. A. Data-efficient deep reinforcement learning for attitude control of fixed-wing UAVs: field experiments. **IEEE Transactions on Neural Networks and Learning Systems**, 2021. 13

BUşONIU, L.; DE BRUIN, T.; TOLIć, D.; KOBER, J.; PALUNKO, I. Reinforcement learning for control: performance, stability, and deep approximators. **Annual Reviews in Control**, v. 46, p. 8–28, 2018. ISSN 1367-5788. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1367578818301184>>. 12

CARRARA, V. **Cinemática e dinâmica de satélites artificias**. São José dos Campos: INPE, 2012. Disponível em: <<http://urlib.net/8JMKD3MGP7W/3B96GD8>>. 18, 19, 21, 22, 27, 29

CARVALHO, T. A. M. de. **Controle de atitude: uma abordagem através de redes neurais artificias**. 157 p. Dissertação (Mestrado em Engenharia e Tecnologia Espacial) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2021. 13

FUJIMOTO, S.; HOOF, H. van; MEGER, D. Addressing function approximation error in actor-critic methods. **CoRR**, abs/1802.09477, 2018. Disponível em: <<http://arxiv.org/abs/1802.09477>>. 10, 50

GOERTZEL, B. Artificial general intelligence: concept, state of the art, and future prospects. **Journal of Artificial General Intelligence**, v. 5, p. 1–48, 2014. 4

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. 3, 59

HAARNOJA, T.; ZHOU, A.; ABBEEL, P.; LEVINE, S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. **CoRR**, abs/1801.01290, 2018. Disponível em: <<http://arxiv.org/abs/1801.01290>>. 10

HAARNOJA, T.; ZHOU, A.; HARTIKAINEN, K.; TUCKER, G.; HA, S.; TAN, J.; KUMAR, V.; ZHU, H.; GUPTA, A.; ABBEEL, P.; LEVINE, S. Soft actor-critic algorithms and applications. **CoRR**, abs/1812.05905, 2018. Disponível em: <<http://arxiv.org/abs/1812.05905>>. 10, 51

HASSELT, H. v.; GUEZ, A.; SILVER, D. Deep reinforcement learning with double q-learning. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 30. **Proceedings...** Phoenix, 2016. p. 2094–2100. 9, 50

HAYKIN, S. **Neural networks and learning machines**. 3. ed. Upper Sadie River: Pearson, 2008. 45, 53, 55, 59

HORNIK, K. Approximation capabilities of multilayer feedforward networks. **Neural Networks**, v. 4, n. 2, p. 251–257, 1991. ISSN 0893-6080. Disponível em: <<https://www.sciencedirect.com/science/article/pii/089360809190009T>>. 61

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989. ISSN 0893-6080. Disponível em: <<https://www.sciencedirect.com/science/article/pii/0893608089900208>>. 61

Instituto Nacional de Pesquisas Espaciais. **Sobre o satélite**. 2021. Disponível em: <http://www.inpe.br/amazonia1/sobre_satelite/>. Acesso em: 18 set. 2023. 64, 65

KINGMA, D. P.; BA, J. **Adam: a method for stochastic optimization**. S.l.: s.n., 2017. 60

KOCH, W.; MANCUSO, R.; WEST, R.; BESTAVROS, A. Reinforcement learning for UAV attitude control. **CoRR**, abs/1804.04154, 2018. Disponível em: <<http://arxiv.org/abs/1804.04154>>. 13

LATHI, B. P. **Modern digital and analog communication systems**. 3. ed. New York: Oxford University Press, 1998. 52

LAY, D. C. **Álgebra linear e suas aplicações**. 4. ed. Rio de Janeiro: LTC, 2013. 26

LEGG, S.; HUTTER, M. **A collection of definitions of intelligence**. S.l.: s.n., 2007. 1

LILLICRAP, T. P.; HUNT, J. J.; PRITZEL, A.; HEESS, N.; EREZ, T.; TASSA, Y.; SILVER, D.; WIERSTRA, D. **Continuous control with deep reinforcement learning**. S.l.: s.n., 2015. 9, 48

LIN, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. **Machine Learning**, v. 8, n. 3–4, p. 293–321, May 1992. ISSN 0885-6125. Disponível em: <<https://doi.org/10.1007/BF00992699>>. 8, 43

LUO, F.-M.; XU, T.; LAI, H.; CHEN, X.-H.; ZHANG, W.; YU, Y. **A survey on model-based reinforcement learning**. S.l.: s.n., 2022. 12

MA, Z.; WANG, Y.; YANG, Y.; WANG, Z.; TANG, L.; ACKLAND, S. Reinforcement learning-based satellite attitude stabilization method for non-cooperative target capturing. **Sensors**, v. 18, n. 12, 2018. 13

MARKLEY, F. L.; CRASSIDIS, J. L. **Fundamentals of spacecraft attitude determination and control**. New York: Springer, 2014. 30, 31, 65, 66

MARQUES, W. J. de S. **Intelligent attitude control of satelittes via deep reinforcement learning**. 126 p. Dissertação (Mestrado em Engenharia e Tecnologia Espacial) — Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2021. 13, 71, 153

MCCARTHY, J. **What is Artificial Intelligence?** 2007. Disponível em: <https://www-formal.stanford.edu/jmc/whatisai.pdf>. Acesso em: 20 mar. 2023. 1

MCCARTHY, J.; MINSKY, M.; ROCHESTER, N.; SHANNON, C. E. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. **AI Magazine**, v. 27, p. 12–14, 2006. 1

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, 1943. 54

MICHIE, D.; CHAMBERS, R. A. BOXES: An experiment in adaptive control. In: DALE, E.; MICHIE, D. (Ed.). **Machine Intelligence**. Edinburgh, UK: Oliver and Boyd, 1968. 44

MINSKY, M. Steps toward artificial intelligence. **Proceedings of the IRE**, v. 49, n. 1, p. 8–30, 1961. 34

MITCHELL, T. M. **Machine learning**. New York: McGraw-Hll, 1997. 2

MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T. P.; HARLEY, T.; SILVER, D.; KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. **CoRR**, abs/1602.01783, 2016. Disponível em: <<http://arxiv.org/abs/1602.01783>>. 11

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M. A.; FIDJELAND, A.; OSTROVSKI, G.; PETERSEN, S.; BEATTIE, C.; SADIK, A.; ANTONOGLOU, I.; KING, H.; KUMARAN, D.; WIERSTRA, D.; LEGG, S.; HASSABIS, D. Human-level control through deep reinforcement learning. **Nature**, v. 518, p. 529–533, 2015. 9, 48, 58

OGATA, K. **Engenharia de controle moderno**. 5. ed. São Paulo: Pearson, 2010. 30

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, p. 533–536, 1986. Disponível em: <<https://doi.org/10.1038/323533a0>>. 60

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. 3. ed. New Jersey: Pearson, 2010. 1

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959. 2

SCHULMAN, J.; LEVINE, S.; MORITZ, P.; JORDAN, M. I.; ABBEEL, P. Trust region policy optimization. **CoRR**, abs/1502.05477, 2015. Disponível em: <<http://arxiv.org/abs/1502.05477>>. 11

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. **CoRR**, abs/1707.06347, 2017. Disponível em: <<http://arxiv.org/abs/1707.06347>>. 11

SHOEMAKE, K. Iii.6 - uniform random rotations. In: KIRK, D. (Ed.). **Graphics Gems III (IBM Version)**. San Francisco: Morgan Kaufmann, 1992. p. 124–132. ISBN 978-0-12-409673-8. Disponível em: <<https://www.sciencedirect.com/science/article/pii/B9780080507552500361>>. 67

SHUSTER, M. D. Survey of attitude representations. **Journal of the Astronautical Sciences**, v. 41, n. 4, p. 439–517, 1993. 21

SILVER, D.; LEVER, G.; HEESS, N.; DEGRIS, T.; WIERSTRA, D.; RIEDMILLER, M. Deterministic policy gradient algorithms. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 31. **Proceedings...** Beijing, 2014. p. 387–395. 8, 45, 46

SILVER, D.; SCHRITTWIESER, J.; SIMONYAN, K.; ANTONOGLOU, I.; HUANG, A.; GUEZ, A.; HUBERT, T.; BAKER, L.; LAI, M.; BOLTON, A.; CHEN, Y.; LILLICRAP, T. P.; HUI, F.; SIFRE, L.; DRIESSCHE, G. van den; GRAEPEL, T.; HASSABIS, D. Mastering the game of go without human knowledge. **Nature**, v. 550, n. 7676, p. 354–359, 2017. 12

SU, R.; WU, F.; ZHAO, J. Deep reinforcement learning method based on ddpg with simulated annealing for satellite attitude control system. In: CHINESE AUTOMATION CONGRESS (CAC). **Proceedings...** [S.l.], 2019. p. 390–395. 13

SUTTON, R. S. Learning to predict by the methods of temporal differences. **Machine Learning**, v. 3, p. 9–44, 1988. 7

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: an introduction**. 2. ed. Massachusetts: The MIT Press, 2017. 7, 9, 11, 38, 39, 42

SUTTON, R. S.; MCALLESTER, D.; SINGH, S.; MANSOUR, Y. Policy gradient methods for reinforcement learning with function approximation. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 12. **Proceedings...** Denver, 1999. p. 1057–1063. 8

TAYLOR, J. **Mecânica clássica**. Porto Alegre: Bookman, 2013. 25, 26, 27

TIAN, J.; CONTRIBUTORS other. **ReinforcementLearning.jl: a reinforcement learning package for the Julia programming language**. 2020. Disponível em: <<https://github.com/JuliaReinforcementLearning/ReinforcementLearning.jl>>. 63

TIPALDI, M.; IERVOLINO, R.; MASSENIO, P. R. Reinforcement learning in spacecraft control applications: advances, prospects, and challenges. **Annual Reviews in Control**, v. 54, p. 1–23, 2022. ISSN 1367-5788. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S136757882200089X>>. 13

WATKINS, C. **Learning from delayed rewards**. 241 p. Tese (Doutorado em Ciência de Computação) — University of Cambridge, 1989. 8, 40

WATKINS, C.; DAYAN, P. Q-learning. **Machine Learning**, v. 8, p. 279–292, 1992. 41

WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. **Machine Learning**, v. 8, n. 3–4, p. 229–256, May 1992. ISSN 0885-6125. Disponível em: <<https://doi.org/10.1007/BF00992696>>. 8

ZHANG, T.; MO, H. Reinforcement learning for robot research: a comprehensive review and open issues. **International Journal of Advanced Robotic Systems**, v. 18, n. 3, p. 1–22, 2021. Disponível em: <<https://doi.org/10.1177/17298814211007305>>. 12

ANEXO A - ATOR COM UMA CAMADA OCULTA

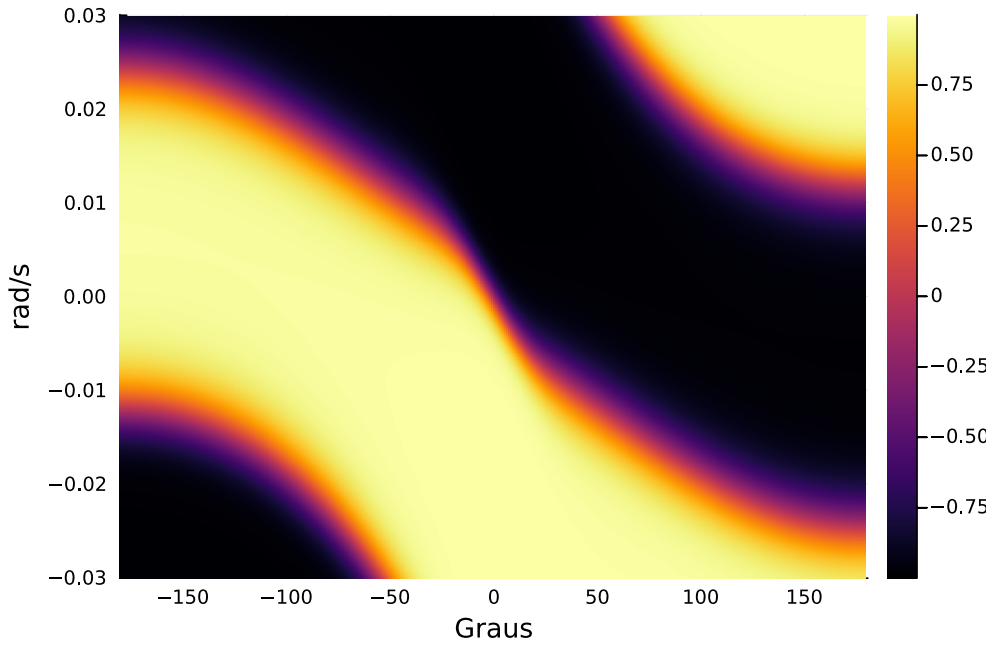
Após o sucesso dos algoritmos aplicados ao problema unidimensional, como discutido na Seção 6.1, buscou-se reduzir o número de parâmetros a serem ajustados na rede do ator. Os hiperparâmetros possuem os mesmos valores da Tabela 6.1, exceto que agora há apenas uma camada oculta. Com isso, o ator possui 96 parâmetros na forma de seus pesos, ao invés de 1120 para o caso com duas ocultas.

Os resultados do SAC para os três eixos separadamente são apresentados nas Figuras A.1, A.2 e A.3. Notadamente, os três mapas de calor possuem um formato muito semelhante.

É curioso notar que as zonas de escape - regiões em que o torque muda de sinal - apresentam tamanhos diferentes. O parâmetro importante é o valor T_{max}/I, que determina a capacidade do atuador de frear ou acelerar o satélite. Como T_{max} é igual para todos os eixos, quanto maior é a inércia maior a área ocupada por essa região.

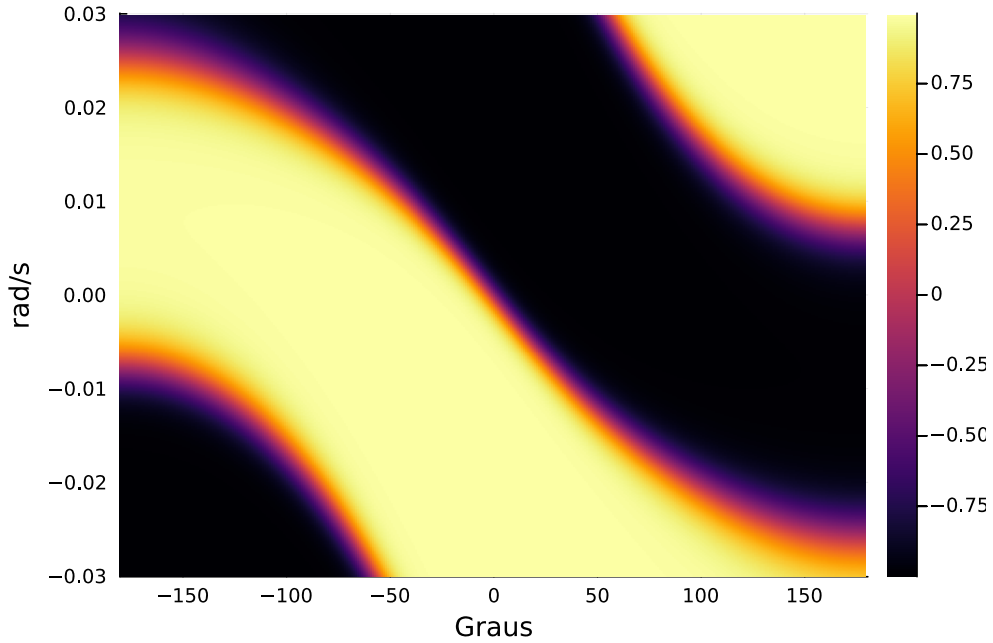
Comparada com a Figura 6.4, um aspecto surpreendente dessa abordagem é que o agente tendeu a um comportamento muito determinístico, o que pode ser observado pela evolução da temperatura na Figura A.4, cujo valor médio foi de 0,0067.

Figura A.1 - Fração do torque nominal comandado pelo SAC, eixo x ($I = 310\ kg\,m^2$).



Fonte: Autor.

Figura A.2 - Fração do torque nominal comandado pelo SAC, eixo y ($I = 360\ kg\,m^2$).



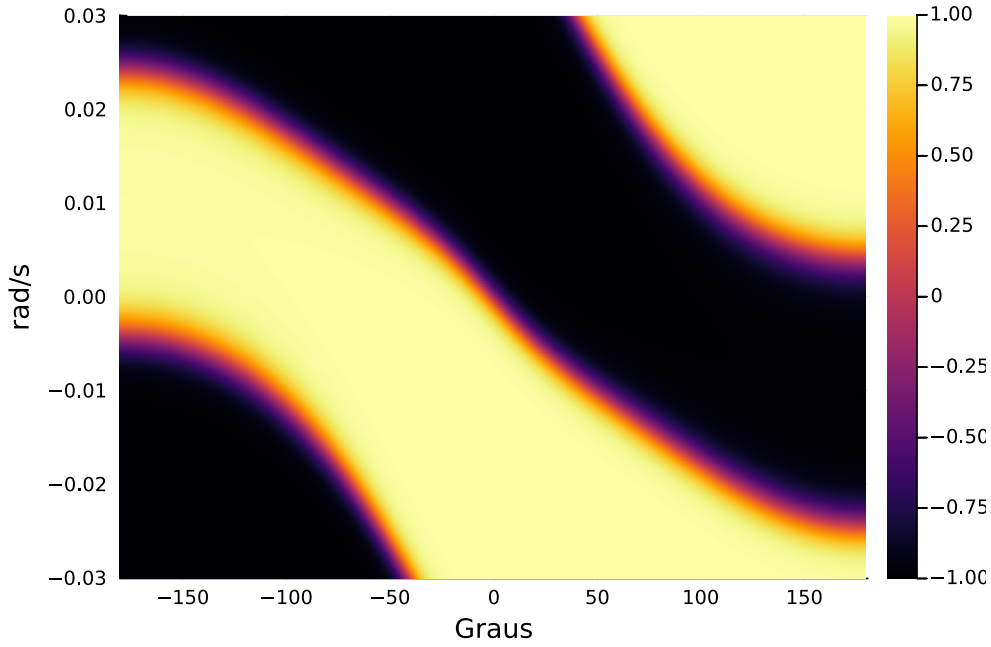
Fonte: Autor.

O retorno $E[G_0]$ para cada eixo é mostrado na Tabela A.1 e comparado com o controlador PD equivalente. Como esperado, tanto para o SAC como para o PD, o retorno é maior o quanto menor for o momento de inércia do eixo em questão.

Tabela A.1 - Retorno ajustado médio $\mathbb{E}[G_0]$ (1 milhão de episódios).

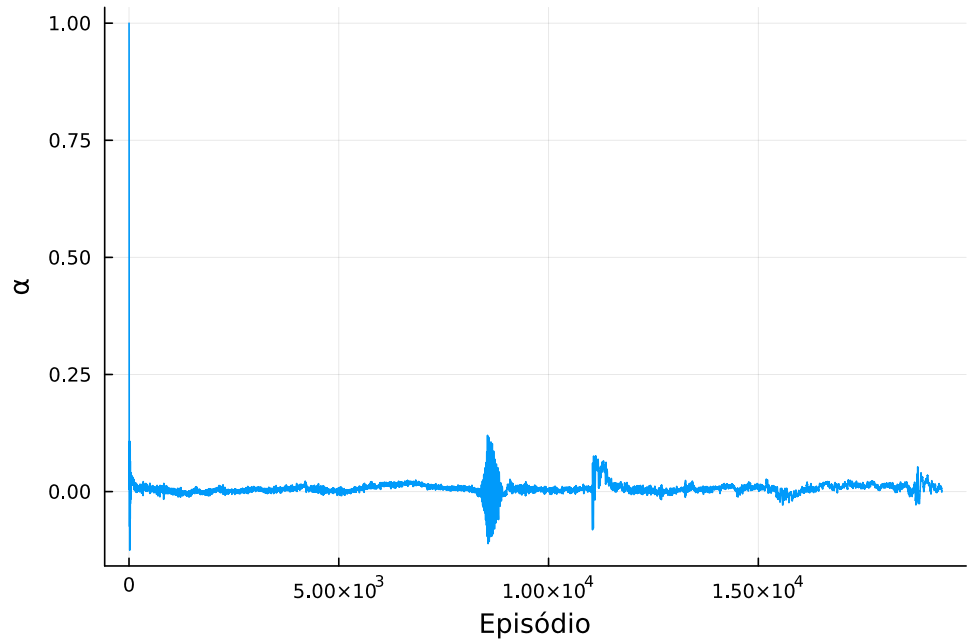
	PD	SAC
Eixo x	-34,26	-32,66
Eixo y	-35,28	-33,23
Eixo z	-38,18	-36,59

Figura A.3 - Fração do torque nominal comandado pelo SAC, eixo z ($I = 530,7\ kg\,m^2$).



Fonte: Autor.

Figura A.4 - Evolução da temperatura ao longo da simulação para o eixo x.



Fonte: Autor.

ANEXO B - ESTADOS INICIAIS UTILIZADOS PARA AVALIAR O DESEMPENHO DO AGENTE

Tabela B.1 - Conjunto de estados iniciais utilizadas para a análise do agente.

Estado	$[q_1\, q_2\, q_3\, \omega_x\, \omega_y\, \omega_z]^T$
s_1	$[-0,8517\ \ 0,2326\ \ -0,2505\ \ -0,0001\ \ -0,0008\ \ 0,0020]$
s_2	$[0,0954\ \ -0,6152\ \ -0,1445\ \ -0,0012\ \ 0,0025\ \ 0,0023]$
s_3	$[0,3074\ \ -0,0707\ \ -0,8855\ \ -0,0019\ \ -0,0019\ \ -0,0008]$
s_4	$[0,3438\ \ 0,5343\ \ -0,7057\ \ 0,0146\ \ -0,0102\ \ -0,0023]$
s_5	$[0,0804\ \ 0,5822\ \ -0,1258\ \ -0,0110\ \ -0,0115\ \ 0,0089]$
s_6	$[0,0440\ \ 0,5125\ \ -0,8140\ \ 0,0024\ \ 0,0042\ \ -0,0054]$
s_7	$[-0,9521\ \ 0,2121\ \ 0,2168\ \ -0,0015\ \ 0,0085\ \ 0,0007]$
s_8	$[0,0095\ \ -0,8377\ \ 0,0566\ \ -0,0106\ \ -0,0017\ \ -0,0008]$
s_9	$[-0,6213\ \ -0,3828\ \ -0,3476\ \ -0,0018\ \ -0,0067\ \ 0,0043]$
s_{10}	$[-0,6173\ \ -0,3816\ \ 0,0697\ \ 0,0033\ \ 0,0033\ 0,0061]$
s_{11}	$[0,7865\ \ -0,2781\ \ 0,4821\ \ -0,0104\ \ 0,0114\ \ 0,0111]$
s_{12}	$[-0,2844\ \ -0,8628\ \ 0,3489\ \ 0,0083\ \ 0,0052\ \ -0,0026]$
s_{13}	$[0,8023\ \ -0,3298\ \ 0,4900\ \ 0,0062\ \ 0,0046\ \ 0,0209]$
s_{14}	$[0,3322\ \ 0,6138\ \ 0,5077\ \ -0,0121\ \ -0,0074\ \ 0,0031]$
s_{15}	$[0,3039\ \ -0,3908\ \ -0,6691\ \ 0,0072\ \ -0,0024\ \ 0,0098]$
s_{16}	$[0,1863\ \ -0,3387\ \ -0,8243\ \ 0,0022\ \ 0,0016\ \ -0,0026]$
s_{17}	$[0,0437\ \ 0,0336\ \ 0,4440\ \ -0,0109\ \ 0,0044\ \ 0,0015]$
s_{18}	$[-0,1563\ \ -0,8520\ \ 0,4667\ \ -0,0003\ \ -0,0007\ \ -0,0018]$
s_{19}	$[-0,8860\ \ 0,4513\ \ 0,0794\ \ -0,0080\ \ 0,0038\ \ 0,0017]$
s_{20}	$[-0,5150\ \ 0,5863\ \ -0,2668\ \ -0,0091\ \ -0,0161\ \ 0,0107]$
s_{21}	$[0,4829\ \ -0,0739\ \ 0,6192\ \ -0,0053\ \ 0,0012\ \ 0,0117]$
s_{22}	$[-0,4033\ \ 0,7897\ \ 0,4498\ \ -0,0144\ \ 0,0084\ \ 0,0082]$
s_{23}	$[0,6374\ \ 0,5987\ \ -0,0443\ \ 0,0126\ \ -0,0111\ \ -0,0037]$
s_{24}	$[0,3747\ \ 0,6879\ \ -0,4742\ \ -0,0021\ \ 0,0067\ \ 0,0076]$
s_{25}	$[0,0599\ \ -0,5273\ \ 0,2438\ \ -0,0089\ \ 0,0051\ \ 0,0130]$
s_{26}	$[-0,6350\ \ -0,6088\ \ 0,2390\ \ 0,0181\ \ 0,0108\ \ -0,0054]$
s_{27}	$[-0,5982\ \ 0,3565\ \ -0,0169\ \ -0,0038\ \ 0,0075\ \ -0,0090]$
s_{28}	$[-0,2769\ \ -0,3638\ \ -0,2269\ \ -0,0090\ \ 0,0208\ \ 0,0051]$
s_{29}	$[-0,7760\ \ -0,4689\ \ 0,1452\ \ -0,0001\ \ -0,0010\ \ -0,0054]$
s_{30}	$[-0,1624\ \ 0,3515\ \ 0,1721\ \ 0,0087\ \ 0,0069\ \ -0,0071]$

ANEXO C - LINGUAGEM JULIA PARA O APRENDIZADO POR REFORÇO

O objetivo desse Anexo é descrever a experiência do autor com a linguagem de programação Julia e alguns de seus pacotes para a obtenção dos resultados apresentados anteriormente. Sendo em essência um relato de experiências, então um aspecto particular e individual deve ser atribuído a essas observações, razão por qual optou-se por apresentá-las à parte. Ainda assim, tais observações podem ser importantes para permitir a reprodução de resultados semelhantes e investigações futuras na linha de RL.

Talvez a primeira pergunta seja: por quê escolher a linguagem Julia e não uma outra qualquer? Para responder essa pergunta, convém realçar algumas características ideais que cevem ser atendidas pela linguagem adotada:

- sintaxe amigável e propícia à programação de alto nível;
- elevado desempenho numérico;
- existência de pacotes próprios voltados ao RL;
- comunidade ampla de usuários;
- preferências pessoais do autor.

Em maior ou menor grau, esses requerimentos são atendidos tanto por Julia como por Python. Esse último em particular conta com o *framework* do PyTorch e a biblioteca TensorFlow voltada para ML, bem como uma comunidade muito mais extensa que Julia. Tendo em vista que Python se tornou em anos recentes uma espécie de linguagem padrão para ML - de fato, a maioria dos algoritmos apresentados aqui foram implementados em seus trabalhos originais em Python -, então escolher Julia pode parecer uma escolha pouco intuitiva e essa decisão deve ser melhor justificada.

Primeiro, o autor possuía maior familiaridade e afinidade com Julia. Segundo, em relação ao desempenho Julia é em geral mais rápido que Python, que é uma linguagem interpretada. A sintaxe de Julia assemelha-se a linguagens de alto nível como MATLAB e Octave ao mesmo tempo que sua velocidade está próxima de linguagens como C e C++. Simulações anteriores dos algoritmos de RL em Python por Marques (2021) tinham fornecido tempos de execução muito proibitivos, da ordem de

20 horas. Esses dois pontos tiveram maior peso que os demais, de forma que Julia foi escolhida.

Definindo-se Julia como a linguagem, então a escolha seguinte dizia respeito aos pacotes a serem utilizados. No que se segue, uma visão geral deles será apresentada, bem como suas capacidades e possíveis limitações.

Sem dúvida alguma, o pacote principal foi o *ReinforcementLearning*, dentro do qual estão contidas as implementações dos algoritmos e as abstrações necessárias para realizar as simulações, sendo escrito inteiramente em Julia. Sua escolha foi imediata, visto que é o pacote mais completo e robusto para RL disponível em Julia.

Conforme discutido, suas abstrações básicas são *AbstractEnv* e *AbstractPolicy*. A lógica do ambiente em si é bastante flexível, sendo deixada a cargo do usuário, enquanto a interface deve possuir o seguinte formato:

```
mutable struct MeuAmbiente <: AbstractEnv
        <campos do ambiente>
end

function RLBase.reset!(env::MeuAmbiente)
        <reinicializa o ambiente>
end

function (env::MeuAmbiente)(action::Vector)
        <altera os campos do ambiente de acordo com a entrada>
end

RLBase.state(env::MeuAmbiente) = <vetor estado atual do ambiente>
RLBase.reward(env::MeuAmbiente) = <termo da recompensa>
RLBase.is_terminated(env::MeuAmbiente) = <episodio chegou ao fim?>
```

A abstração *AbstractPolicy* contém todos algoritmos apresentados, assim como alguns outros do RL. Seu tipo concreto mais importante, contudo, é o tipo *Agent*, composto de dois campos: uma política propriamente falando do mesmo tipo *AbstractPolicy* e um replay de memória do tipo *AbstractTrajectory*. Novos algoritmos podem ser facilmente adicionados, definindo-se uma função de interação com o ambiente e a atualização correspondente da política.

Como pode ser concluído a partir dessa descrição, o pacote *ReinforcementLearning*

é bastante poderoso e flexível. Sua instabilidade, entretanto, pode ser demonstrada a partir de dois problemas enfrentados, que exigiram certas modificações na implementação dos algoritmos para serem corrigidos.

O primeiro diz respeito ao TD3 para ambientes com espaço de ação maior que 1, isso é, nos quais a ação produzida é um vetor com 2 ou mais elementos. A implementação do algoritmo assumia que certas variáveis eram escalares e não estava generalizada para operações com vetores, isso é, não as realizava elemento por elemento. Sua descrição mais precisa e solução podem ser encontradas no link Issue 951.

O segundo problema encontrado foi no SAC, que não convergia para uma política minimamente razoável. De fato, pelo conhecimento do autor, esse problema estava presente em todas as versões do pacote até então, de forma que o SAC não era um algoritmo utilizável. Curiosamente, sua causa não estava ligado à implementação do algoritmo em si, mas sim da rede gaussiana para gerar a distribuição das ações. Para uma discussão mais profunda ver o link Issue 970.

O pacote *Zygote* realiza a diferenciação automática de funções, sendo necessário na implementação de todos os algoritmos para a atualização do ator e do crítico por meio do cálculo de gradientes. Seu funcionamento interno é extremamente complexo e envolve uma análise minuciosa do comportamento do código para a determinação das derivadas. No problema descrito acima com o SAC, um bloco mal posicionado do *Zygote* impedia o cálculo do gradiente da Equação 4.34 e, consequentemente, a atualização correta da política.

O código a seguir, por exemplo, calcula a derivada da função $f(x) = x^2 - 5\,x + 10$ no ponto 1, retornando o valor -3, uma vez que $f'(1) = -3$.

```
gs = gradient(1) do x
        x ^ 2 - 5 * x + 10
end
```

De maneira semelhante, as derivadas parciais também podem ser encontradas. Para a função $f(x, y) = x^3 - 2\,x\,y^2 + 5\,x$, o seguinte código retorna os valores $\frac{\partial f}{\partial x}$ e $\frac{\partial f}{\partial y}$ no ponto (1,2), isso é, o gradiente de f, como uma tupla de dois componentes.

```
gs = gradient(1, 2) do x, y
        x ^ 3 - 2 * x * y ^ 2 + 5 * x
end
```

O pacote *Flux* possui as implementações básicas para a criação de redes neurais de variados tipos, otimizadores para a atualização de parâmetros e algumas outras utilidades, como funções *loss* específicas e certas funções de ativação. De maneira geral, o estado atual do pacote é bastante estável, o que junto com sua sintaxe clara e fácil customização o fazem uma alternativa muito segura para ML.

Uma camada totalmente conectada de uma rede neural pode ser criada pela função *Dense*; camadas podem ser conectadas usando-se o comando *Chain*. Arquiteturas personalizadas podem ser facilmente criadas. Por exemplo, a rede parcialmente conectada para o controle das 4 rodas pode ser criada a partir da seguinte lógica:

```
struct Controlador
        eixo_x::Chain
        eixo_y::Chain
        eixo_z::Chain
        eixo_w::Chain
end

function (m::Controlador)(input::Vector)
        < ... >
        return output
end

Flux.@functor Controlador
```

A macro *Flux.@functor* define que a estrutura Controlador é um functor e seus campos correspondem a parâmetros que podem ser modificados.

Os pacotes *Zygote* e *Flux* podem ser facilmente combinados para o treinamento de redes neurais, com o primeiro calculando o gradiente dos parâmetros em relação a uma dada função *loss* e o segundo pacote realizando sua atualização, a partir da função *Flux.update!*.

Como nas máquinas em que os códigos foram executados não havia nenhuma placa de vídeo (GPU) dedicada, o pacote *CUDA* não foi empregado. As operações envolvendo rede neurais, como inferência e atualização dos pesos, foram realizadas, portanto, inteiramente na CPU. O paralelismo inato a GPU favorece seu amplo uso no campo de ML, permitindo um treinamento mais rápido das redes neurais. Ainda assim, é discutível se a presença de uma GPU traria ganhos notáveis: tanto o ator

como o crítico foram representados, em todas as simulações, por redes relativamente pequenas para os padrões do *Deep Learning*. Nessas condições, as vantagens de usar uma GPU frente a CPU tendem a ser menores. Ainda assim, é evidente que apenas uma implementação real poderia esclarecer esse ponto de forma definitiva.

Outros pacotes que foram usados de maneira pontual merecem ser mencionados. O pacote *ReferenceFrameRotations* disponibiliza estruturas e funções para operar com representações básicas de atitude, como quatérnions ou ângulos de Euler. A diferenciação, como dado pelas Equações 3.26 e 3.27, também encontra-se disponível. *IntervalSets* define intervalos abertos e fechados, o que é útil para definir o espaço de ação do problema. *LinearAlgebra* define alguma operações básicas de álgebra linear, como produto interno, vetorial e norma. Sendo o *Core* de Julia relativamente enxuto, essas funcionalidades adicionais facilitam a programação. O pacote *StableRNGs* fornece geradores de números aleatórios, o que é importante para garantir uma inicialização suficiente aleatória para os parâmetros das redes.

Em resumo, embora a comunidade de Julia e, consequentemente, o número de pacotes seja consideravelmente menor que Python, as funcionalidades elementares para seu uso no campo de RL, bem como ML de uma forma mais geral, estão disponíveis. É a visão do autor que os complexos pacotes *Flux* e *Zygote* estão maduros e estáveis o bastante para servirem de base para trabalhos do ML em Julia.

ReinforcementLearning ainda apresenta algumas instabilidades visíveis, possivelmente em razão da amplitude do pacote, do número relativamente baixo de desenvolvedores e de se tratarem de algoritmos muito recentes. Essas instabilidades demandam do usuário um bom conhecimento teórico acerca do RL e dos algoritmos em questão, de modo que eventuais alterações no código fonte podem ser necessárias. Ainda assim seu elevado desempenho, a boa estruturação de suas abstrações e seu escopo abrangente, englobando uma vasta quantidade de algoritmos do RL, tornam o pacote uma excelente escolha para investigações futuras.