UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MIKAEL NEDEL HARTMANN

CONTROLE PREDITIVO DE TRAJETÓRIAS COM RESTRIÇÃO NO SINAL DE CONTROLE VIA REDES NEURAIS

CURITIBA

2023

MIKAEL NEDEL HARTMANN

CONTROLE PREDITIVO DE TRAJETÓRIAS COM RESTRIÇÃO NO SINAL DE CONTROLE VIA REDES NEURAIS

Predictive Control of Trajectories with Control Signal Constraint via Neural Networks

Dissertação de Mestrado apresentada como requisito para obtenção do título de Mestre em Engenharia de Automação e Sistemas do Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Flávio Neves Jr.

Coorientador: Profª. Drª. Lúcia Valéria Ramos de Arruda

CURITIBA

2023

4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

MIKAEL NEDEL HARTMANN

CONTROLE PREDITIVO DE TRAJETÓRIAS COM RESTRIÇÃO NO SINAL DE CONTROLE VIA REDES NEURAIS

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Ciências da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Automação E Sistemas.

Data de aprovação: 10 de Novembro de 2023

Dr. Flavio Neves Junior, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Joao Paulo Lima Silva De Almeida, Doutorado - Instituto Federal de Educação, Ciência e Tecnologia do Paraná (Ifpr)

Dra. Lucia Valeria Ramos De Arruda, Doutorado - Universidade Tecnológica Federal do Paraná

Vlademir Aparecido Freire Junior, - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 10/11/2023.

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. Flávio Neves Junior e minha coorientadora Profª Dr. Lúcia Valéria Ramos de Arruda por sua orientação valiosa ao longo deste processo.

Agradeço aos professores das disciplinas cursadas, pelo conhecimento passado.

Agradeço também aos meus pais pelo apoio inabalável.

E por fim agradeço a minha gata, Freyja, que foi uma presença constante e reconfortante durante a pesquisa e escrita deste trabalho.

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

RESUMO

Este trabalho propõe a aplicação de controle preditivo não linear baseado em modelo (NLMPC) com restrições no sinal de controle em sistemas robóticos caracterizados por uma arquitetura de controle em cascata. A estratégia de controle preditivo desenvolvida usa um modelo por redes neurais *feedforward* para inferir o valor das restrições no sinal de controle da malha secundária, que utiliza um controlador proporcional-derivativo (PD). Esses valores de restrições são incluídos no cálculo do sinal de controle da malha primária pelo controlador preditivo. As redes neurais foram treinadas a partir das condições iniciais e finais de posição e velocidade do sistema a ser controlado, e desta forma inferem como saída o pico da derivada dos torques ou forças futuras que devem ser respeitadas pelo PD a partir dos comandos da malha externa. Como estudo de caso, foi aplicado controle preditivo baseado em modelo (MPC) como a malha externa e controladores proporcional-derivativos como malha interna para um robô manipulador planar duplo rotativo (2R) e um drone quadrotor. A aplicação de controle com a arquitetura proposta foi testada em ambos os sistemas através de simulações computacionais no MATLAB. A abordagem resultou em trajetórias mais suaves para o robô manipulador durante o desvio de obstáculos. Já para os drones, em algumas situações experimentadas, o controlador permitiu em presença de obstáculos, a execução de grandes desvios na rota planejada. Em conclusão, a estratégia de controle desenvolvida apresenta um bom potencial para aplicações em sistemas robóticos com variáveis de estado "cinematicamente desacopladas".

Palavras-chave: controle preditivo; controle não linear; redes neurais; robótica; controle de trajetória.

ABSTRACT

This dissertation investigates the application of non-linear predictive control with constraints on the control signal for robotic systems characterized by a cascade control architecture. The proposed control strategy uses a feedforward neural network model to assist the controller when applying restrictions on the secondary loop control signal. This secondary loop has a proportional-derivative (PD) controller. The neural network model is trained based on the initial and final conditions of position and velocity for the system to be controlled. The trained model infers, as output, the peak derivative of torques or future forces that the PD controller must respect based on the commands from the outer loop. As a case study, model predictive control (MPC) is applied as the outer loop, and proportional-derivative (PD) controllers are used as the inner loop for a 2R planar manipulator robot and a quadrotor drone. The control application with the proposed architecture was tested on both systems through computational simulations in MATLAB. The approach resulted in smoother trajectories for the manipulator robot during obstacle avoidance. However, during some experiments with obstacles, the controller allowed the drone to realize significant deviations from the planned route. In conclusion, the developed control strategy presents good potential for applications in robotic systems with “kinematically decoupled” state variables.

Keywords: predictive control; nonlinear control; neural networks; robotics; trajectory control.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Letras Latinas

A	Coeficientes de atrito aerodinâmico	
a_i	Translação do i-ésimo eixo X do robô	
b	Vetor dos termos lineares da função custo	
C	Matriz de Coriolis	
d_i	Translação do i-ésimo eixo Z do robô	$[\mathrm{m}]$
F	Matriz usada na equação de predição	
G	Vetor de forças e torques gravitacionais	
g	Aceleração da gravidade	$[\mathrm{m/s^2}]$
H	Matriz dos termos quadráticos da função custo	
I	Momento de inércia	$[\mathrm{kg{\cdot}m^2}]$
J	Matriz jacobiana	
K	Energia Cinética	[Nm ou J]
L	Lagrangeano	[Nm ou J]
M	Matriz de inércia	
M'	Matriz de restrições	
m	Massa	$[\mathrm{kg}]$
N_c	Horizonte de controle	
N_p	Horizonte de predição	
o_i	Origem da base da i-ésima junta	$[\mathrm{m}]$
P	Energia Potencial	[Nm ou J]
Q_r	Matriz de ponderação do erro transiente	
Q_t	Matriz de ponderação do erro final	
Q_v	Matriz de ponderação das velocidades	

Q_u	Matriz de ponderação das acelerações	
q	Vetor de coordenadas de junta	[rad ou m]
$\bar{R}$	Matriz de regularização da saída	
R_s	Vetor de referências futuras	
R	Matriz de Rotação	
r	Posição cartesiana	[m]
r_c	Posição do centro de massa	[m]
S	Função custo do MPC	
${T_A}^B$	Matriz de transformação homogênea	
T	Thrust	[N]
t	Tempo	[s]
u	Sinal de controle	
U	Vetor de sinal de controle futuro	
W	Matriz de mudança de base para ângulos de Euler	
Y	Vetor de saídas preditas	
x, y, z	Coordenadas cartesianas	[m]
z_i	Vetor de rotação da i-ésima junta	

Letras Gregas

α_i	Ângulo do i-ésimo eixo X do robô	[°]
θ_i	Ângulo do i-ésimo eixo Z do robô	[°]
θ	Ângulo pitch do drone	[rad]
ϕ	Ângulo roll do drone	[rad]
ψ	Ângulo yaw do drone	[rad]
Φ	Matriz de transição de estados	
ξ	Posição cartesiana do drone	[m]
η	Orientação do drone	[rad]
λ	Multiplicador de Lagrange	
τ	Torques generalizados	[Nm ou N]
γ	Restrições	
ω	Velocidades angulares	[rad]

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações iniciais

O planejamento de trajetória ou percurso desempenha um papel importante na robótica, pois permite que sistemas robóticos como robôs manipuladores, robôs móveis ou drones se movam eficientemente de um ponto a outro em seu ambiente de trabalho. Esse planejamento busca proporcionar segurança e precisão nos movimentos do robô ao detalhar cada segmento do percurso, considerando obstáculos e limitações do ambiente, o planejamento de trajetória contribui para que os robôs executem tarefas complexas com maior exatidão. De modo que o percurso, as velocidades, as acelerações e restrições do sistema robótico são levadas em conta para atender as especificações do projeto e o cumprimento dos objetivos estipulados. Para realizar essa tarefa, é necessário um controlador que seja capaz de gerar um sinal de controle adequado para que o robô possa seguir as referências estabelecidas.

A técnica de controle conhecida como Controle Preditivo baseado em Modelo (MPC) pode ser utilizada nesse contexto. O MPC permite planejar a trajetória cinemática de sistemas autônomos de maneira dinâmica, levando em consideração as restrições tanto do sistema em si quanto do ambiente em que ele opera (LIU *et al.*, 2017). Com essa abordagem, o MPC é capaz de otimizar as trajetórias de robôs manipuladores ou robôs móveis, como quadricópteros, minimizando erros de trajetória e garantindo operações seguras e eficientes.

Uma prática em aplicações de robótica é o uso de técnicas de controle em cascata, em que um controlador é responsável por gerar trajetórias e um segundo controlador se encarrega do controle físico do robô propriamente dito, aplicando as forças e torques necessários para movimentar o sistema (KRÄMER *et al.*, 2020).

Uma configuração promissora é utilizar um controlador MPC na malha primária para geração de trajetórias e um controlador clássico do tipo Proporcional-Integral-Derivativo (PID) na malha secundária.

Entretanto, uma limitação dessa abordagem é que o MPC não tem controle direto sobre a saída da malha secundária, o que impede que o MPC aplique restrições a essa saída durante o processo de otimização da trajetória. Para superar essa limitação, uma abordagem baseada em dados, usando uma rede neural será utilizada neste trabalho para realizar estimativas sobre o sinal de controle da malha secundária. Essas estimativas fornecidas pela rede neural são incorporadas pelo MPC como restrições adicionais durante a otimização da trajetória, onde este precisa garantir que as restrições referentes aos valores máximos para o sinal de controle sejam respeitados.

Dessa forma, o MPC busca uma solução que esteja dentro da região viável definida conjuntamente pelas restrições originais do processo e pelas novas restrições estimadas pela rede neural. Isso permite contornar a limitação, característica das configurações em cascata, de não ter acesso direto à saída da malha secundária. A expectativa é que a incorporação das estimati-

vas oriundas do modelo neural como restrições adicionais ao problema de otimização resolvido pelo MPC, leve a trajetórias mais suaves e conforme os limites de operação do sistema.

Algumas aplicações de MPC com outros controladores em cascata, semelhantes à proposta deste trabalho são encontradas na literatura recente. Incremona, Ferrara e Magni (2017) apresentam um método para controlar o movimento de robôs manipuladores utilizando uma combinação de MPC, modos integrais de deslizamento (ISM) e controle por torque computado (CTC) em cascata. O ISM é uma técnica de controle que utiliza uma "superfície deslizante" para assegurar a estabilidade do sistema, sendo aplicado para corrigir as incertezas na malha de controle do torque computado. O método proposto nesse estudo integra essas diferentes técnicas com o objetivo de controlar o movimento de um robô manipulador em tempo real.

No estudo conduzido por Varga *et al.* (2019), foi empregado um MPC para regular um sistema veículo-manipulador. O robô manipulador acoplado ao veículo utilizou um controlador secundário composto por um PID com torque computado. O objetivo do MPC era coordenar o braço robótico juntamente com o veículo móvel para alcançar uma posição desejada.

Nubert *et al.* (2020) propuseram uma técnica baseada em redes neurais para aproximar o comportamento de um controlador MPC robusto aplicado em robôs manipuladores. O MPC apresentado no estudo garantia uma operação segura, mantendo a estabilidade e satisfazendo restrições, mesmo diante das incertezas inerentes ao sistema. Nessa abordagem, a otimização *online* feita pelo MPC foi substituída por uma avaliação realizada por uma rede neural previamente treinada e validada *offline*.

Elsisi *et al.* (2021) fizeram uso de redes neurais para otimizar os parâmetros de um MPC não linear (NLMPC) aplicado ao controle de um robô manipulador com dois graus de liberdade. Através desse método, os autores buscaram otimizar o horizonte de predição, o horizonte de controle e as ponderações da otimização do NLMPC.

Chiu *et al.* (2022) desenvolveram um MPC para controlar um robô quadrúpede com um manipulador acoplado. O MPC desempenhou um papel fundamental ao lidar com as restrições que evitavam colisões entre o robô e o ambiente, bem como colisões internas entre os componentes do robô. Essa abordagem permitiu ao robô realizar uma variedade de operações, como abrir portas e lançar objetos, sem que o braço robótico colidisse com o corpo principal do robô.

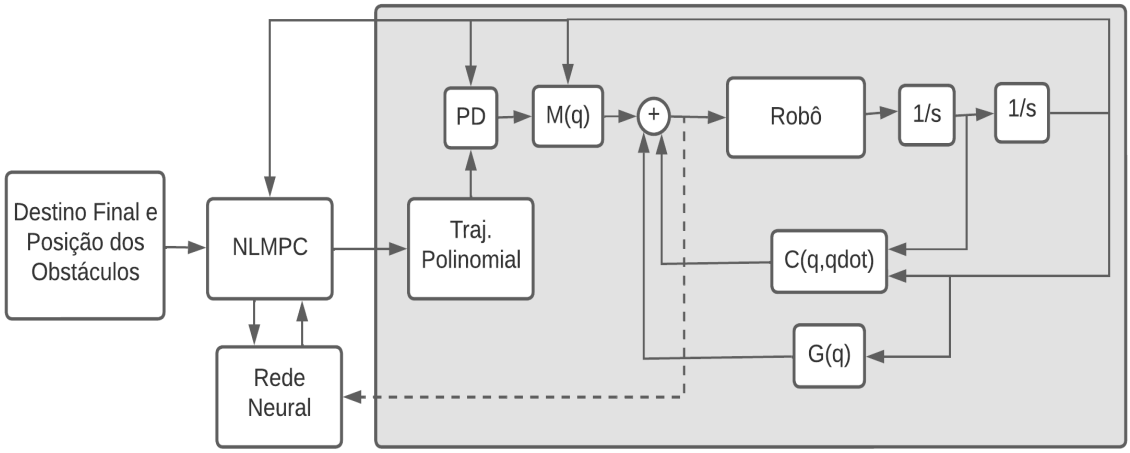
Em Eckhoff *et al.* (2022) foi apresentada uma abordagem na qual um MPC é integrado a um sistema de visão para permitir que um robô execute operações em colaboração com seres humanos. Nessa configuração, uma das componentes da função objetivo otimizada pelo MPC é a distância entre o robô e os operadores humanos presentes no ambiente. Esses operadores são identificados pelo sistema de visão. O objetivo é maximizar essa distância para garantir a segurança e evitar colisões durante a interação humano-robô.

Ding *et al.* (2022) realizaram o controle de um robô humanóide utilizando uma estratégia em cascata com um controlador NLMPC e um controlador MPC linear (LMPC). O controlador MPC linear operava com um tempo de amostragem de 10 ms, enquanto o controlador NLMPC possuía um tempo de amostragem de 100 ms devido ao seu maior custo computacional. Essa

abordagem em cascata permitiu combinar a eficiência do MPC linear com a maior capacidade do NLMPC para tratar não linearidades e incertezas.

A arquitetura para a malha de controle em cascata proposta neste trabalho é visualizada na Figura 1, sendo que o funcionamento da malha primária e secundária será discutido em maiores detalhes no Capítulo 2. Como destacado pelo diagrama, o controlador primário (NLMPC) age de forma indireta sobre o sistema controlado. Para que este consiga restringir os torques aplicados sobre o robô, é necessária alguma forma de inferência de como a ação de controle da malha secundária irá se comportar a partir dos comandos fornecidos pelo NLMPC. A importância do uso de uma abordagem não linear, como o NLMPC, se dá por conta de não linearidades trigonométricas oriundas de matrizes de transformação que impactam a função custo e as restrições do sistema, essas não linearidades serão melhor detalhadas ao decorrer do trabalho.

Figura 1 – Diagrama da Malha de Controle em Cascata NLMPC-CTC de um Robô Manipulador



Fonte: Autoria própria (2023).

Uma maneira viável de mapear os torques gerados pelo controlador proporcional-derivativo (PD) a partir das referências cinemáticas definidas pelo MPC é por meio do uso de redes neurais artificiais. A utilização de redes neurais para estimar os torques é uma prática comumente adotada no controle de força de robôs manipuladores, eliminando assim a necessidade de um sensor de força acoplado ao robô. Essa abordagem foi explorada por Kruzic *et al.* (2021).

Uma arquitetura análoga pode ser estabelecida para outros sistemas que necessitem de geração de trajetórias. No caso de quadricópteros por exemplo, também encontra-se na literatura o uso de MPC em cascata para o controle de posição.

Cheng e Yang (2017) propuseram a implementação de um controlador MPC para o controle de posição de um drone quadrotor, enquanto um controlador PD fica responsável pelo controle de orientação com uma resposta mais rápida para a rejeição de distúrbios.

Por sua vez, Santos, Ferramosca e Raffo (2018) utilizam de uma configuração distinta de drone, o tiltrotor, que possui apenas duas hélices. Da mesma forma foi usado o controle MPC para a malha externa, e para a malha interna, responsável pela dinâmica não linear do sistema, foi aplicada uma linearização por realimentação entrada-saída (IOFL).

Jiang *et al.* (2022) usam uma arquitetura de controle MPC-PID e um drone quadrotor assim como Cheng e Yang (2017), todavia utilizam uma rede neural *feedforward* para a previsão das acelerações lineares do drone, e com esta informação melhora-se o desempenho da malha de controle preditivo.

Para este trabalho, a partir da proposta de arquitetura para robôs manipuladores ilustrada na Figura 1, pretende-se usar de uma abordagem de controle cascata NLMPC-PD também para drones quadricópteros. Ou seja, o controlador NLMPC gerará as trajetórias livres de colisão enquanto o PD será responsável pela dinâmica rápida do sistema.

A diferença em relação ao trabalho de Jiang *et al.* (2022) é que, ao invés de usar a rede neural apenas para previsão, no presente trabalho um modelo por redes neurais será utilizado para inferir restrições no sinal de controle do drone. Mais especificamente, a rede neural estimará os valores máximos da derivada da força de *thrust* e dos torques de *roll* e *pitch* que serão gerados pelo controlador PD. Essas estimativas serão incorporadas como restrições adicionais pelo controlador NLMPC durante o planejamento da trajetória. Dessa forma, o NLMPC garantirá que as entradas de controle calculadas respeitem os limites inferidos pela rede neural para a saída do PD.

1.2 Objetivos

Neste contexto de aplicação de controle preditivo em cascata para robótica usando modelos baseados em redes neurais, esta dissertação de mestrado propõe-se a atender os seguintes objetivo geral e objetivos específicos.

1.2.1 Objetivo geral

Desenvolver um sistema de controle MPC em cascata com restrições no sinal de controle da malha secundária calculadas por redes neurais. O sistema de controle será aplicado na geração de trajetórias de um robô manipulador e de um drone quadricóptero.

1.2.2 Objetivos específicos

O objetivo geral pode ser subdividido em:

- Modelar a dinâmica de um robô de dois graus de liberdade e de um drone quadricóptero.

- Propor uma arquitetura de controle MPC não linear em cascata com restrições calculadas por redes neurais.
- Implementar um controle cascata MPC-PD para um robô de dois graus de liberdade e um drone quadricóptero com desvio de obstáculos.
- Treinar uma rede neural para a predição do sinal de controle da malha secundária de ambos os sistemas e usá-la como restrição do controlador preditivo.
- Validar a arquitetura proposta em modelos simulados da literatura.

1.3 Justificativa

O desenvolvimento de controladores que levam em consideração restrições de posição, torque ou potência é crucial em projetos que requerem medidas de segurança e manutenção. Conforme mencionado na Seção 1.1, estudos recentes têm explorado o uso do controle preditivo para evitar colisões. Por exemplo, os trabalhos de Chiu *et al.* (2022) e Eckhoff *et al.* (2022) abordam diferentes aspectos das colisões, incluindo colisões internas do robô, colisões com o ambiente e colisões com seres humanos. Esses estudos são relevantes especialmente em ambientes de trabalho nos quais humanos e máquinas operam simultaneamente.

Além das limitações de posição, que se concentram na prevenção de colisões, é igualmente importante considerar a intensidade de uma colisão caso esta ocorra, bem como o desgaste das peças mecânicas resultante de movimentos bruscos. Desse modo, tais questões colocam-se como pontos relevantes na implementação de protocolos de segurança e na manutenção preventiva de sistemas autônomos, como robôs e quadricópteros.

Portanto, o objetivo deste trabalho é delimitar a superfície de controle admissível para força e torque em sistemas controlados por MPC em cascata. Isso será feito integrando o controlador preditivo a uma rede neural que inferirá limites viáveis para essas grandezas, o MPC utilizará as estimativas da rede neural para restringir a região de força e torque durante a otimização.

1.4 Estrutura do trabalho

O presente trabalho está organizado em cinco capítulos, sendo este a introdução. O segundo Capítulo aborda a fundamentação teórica, com ênfase na modelagem de sistemas controlados e fornece uma breve explicação matemática sobre o funcionamento do controlador MPC.

O terceiro Capítulo se concentra nos métodos empregados para modelar especificamente um robô de dois graus de liberdade e um drone quadrotor. Também aborda a simulação utilizada para coletar dados e treinar as redes neurais.

O quarto Capítulo é dedicado à apresentação dos resultados obtidos, com a simulação dos sistemas robóticos citados controlados pela arquitetura de controle MPC com restrições inferidas pela rede neural implementada.

O quinto Capítulo apresenta as conclusões do trabalho e endereça sugestões de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este Capítulo é responsável pela fundamentação teórica das técnicas de controle e modelagem utilizadas neste trabalho. Dessa forma, cobre-se a modelagem cinemática e dinâmica de robôs manipuladores e drones quadricópteros, a aplicação de controladores PID nestes sistemas, e uma breve revisão de controladores MPC, NLMPC e redes neurais.

2.1 Robôs Manipuladores

Esta seção tem como objetivo revisar tópicos essenciais para a implementação e compreensão da malha de controle do robô manipulador, além de fundamentar, em mais detalhes, as equações geométricas e de movimento do robô. Os livros que foram utilizados como base para esta seção são Spong, Hutchinson e Vidyasagar (2006) e Craig (2009).

2.1.1 Cinemática e Trajetória

Na robótica, a cinemática é dividida em cinemática direta e inversa. A cinemática direta é de fácil compreensão e é derivada somente da multiplicação de matrizes de rotação e orientação, enquanto a cinemática inversa requer uma análise mais aprofundada, pois não há uma fórmula única para a sua resolução.

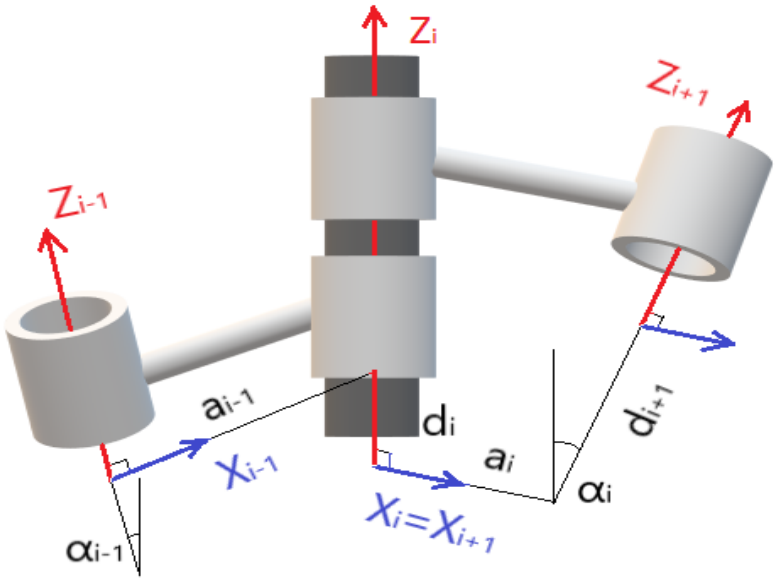
Para robôs manipuladores, a cinemática direta consiste na transformação da posição das juntas (rotativas ou prismáticas) do robô para as posições do espaço operacional. Essa transformação é composta por uma série de matrizes homogêneas (que realizam translações e rotações) e, dessa forma, as coordenadas de cada uma das juntas são referenciadas em relação à junta anterior até chegar ao efetuador final (posição da garra ou ferramenta) como visto na Equação 1. Por sua vez, cada matriz de transformação homogênea, segundo o método de Denavit-Hatemberg Modificado (DHM), descrito por Craig (2009),tem a forma da Equação 2.

$$\mathrm{T}_N^0 = \mathrm{T}_1^0 \mathrm{T}_2^1 \cdots \mathrm{T}_N^{N-1} \tag{1}$$

$$\mathrm{T}_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times 4} \tag{2}$$

onde α_i e a_i são a rotação e translação em relação ao i-ésimo eixo X, enquanto θ_i e d_i são a rotação e a translação em relação ao eixo Z. A notação de s e c representa senos e cossenos em uma forma mais enxuta. Esta transformação pode ser melhor visualizada na Figura 2.

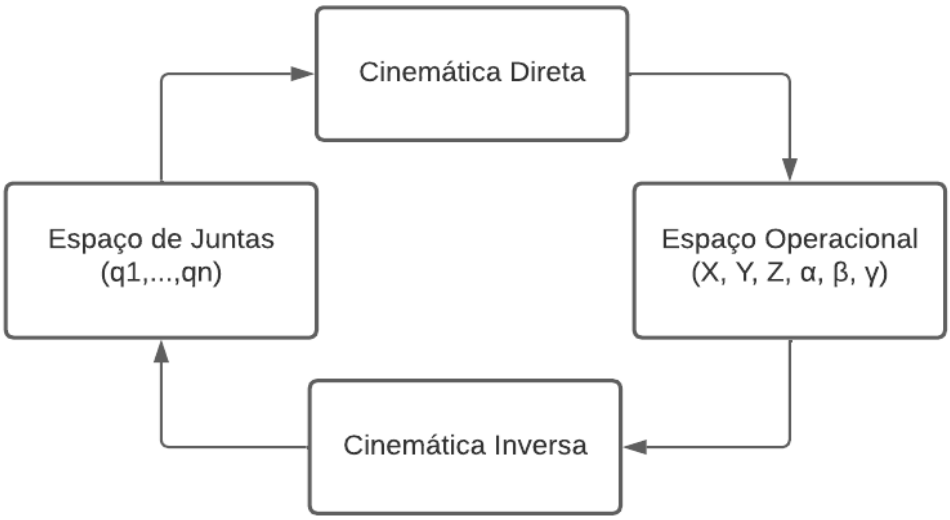
Figura 2 – Coordenadas das Juntas



Fonte: Autoria própria (2023).

A cinemática inversa faz o processo inverso, tendo como entradas as posições operacionais (cartesianas e de orientação do efetuador) e retornando as posições de juntas conforme ilustrado pela Figura 3.

Figura 3 – Cinemática Direta e Cinemática Inversa



Fonte: Autoria própria (2023).

A cinemática inversa pode ser resolvida analiticamente em alguns casos, usando funções trigonométricas como o arcotangente, com base nas equações obtidas pela cinemática direta. No entanto, em casos em que o robô apresenta assimetrias, aumentando sua complexi-

dade geométrica, não há uma solução analítica disponível e é necessário recorrer a soluções numéricas.

As relações entre a posição da junta e a posição cartesiana são complexas, enquanto as relações entre a velocidade angular e a velocidade operacional são mais simples, sendo calculadas através da matriz jacobiana como mostra a Equação 3.

$$\dot{X} = J\dot{q} \tag{3}$$

onde $\dot{X}$ é a velocidade operacional e cada coluna da matriz jacobiana J pode ser calculada através das Equações 4 e 5, sendo respectivamente a formulação para uma junta rotativa e uma junta prismática. A notação de z_i e o_i referem-se ao vetor z (rotação da junta) e a origem da base em questão.

$$J_i = \begin{bmatrix} z_i \times (o_n - o_i) \\ z_i \end{bmatrix} \tag{4}$$

$$J_i = \begin{bmatrix} z_i \\ 0 \end{bmatrix} \tag{5}$$

Invertendo a Equação 3 e integrando numericamente, é possível encontrar os ângulos das juntas necessários para alcançar a posição desejada.

Robôs que possuem mais juntas do que o espaço operacional são chamados de robôs redundantes e, neste caso, a matriz jacobiana não é quadrada, sendo necessário o uso de uma pseudo-inversa para calcular os ângulos das juntas. Se o robô tiver menos juntas do que o espaço operacional, a pseudo-inversa retornará os incrementos angulares com um erro que pode ser minimizado através do método dos mínimos quadrados. Caso o robô tenha juntas a mais do que o espaço operacional, a utilização da pseudo-inversa permite satisfazer as restrições cinemáticas enquanto minimiza as normas dos incrementos angulares (KLEIN; HUANG, 1983). Por fim, com as posições angulares iniciais e finais definidas, tem-se o planejamento de trajetória, que estabelece os percursos, as velocidades e as acelerações do robô para a realização de determinadas tarefas. Uma das formas mais simples de planejamento de trajetórias é o uso de velocidades trapezoidais, em que o robô tem uma pequena aceleração no início e no final do percurso, mas na maior parte do tempo se desloca com velocidade constante. No entanto, essas mudanças de velocidade resultam em acelerações e arrancadas (*jerks*) bruscas, por isso, trajetórias mais suaves como as trajetórias cúbicas ou polinomiais de quinta ordem podem ser mais indicadas. A equação de uma trajetória cúbica ou de quinta ordem é caracterizada através dos seus parâmetros k (coeficientes dos polinômios) extraídos das Equações matriciais 6 e 7, respectivamente.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix}_{4\times4} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}_{6\times6} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ a_0 \\ q_f \\ v_f \\ a_f \end{bmatrix} \tag{7}$$

onde t é o tempo (variável do polinômio), e q, v e a são as condições de contorno de posição, velocidade e aceleração. Os subíndices "0" e "f" denotam os pontos inicial e final da trajetória.

O problema dessas formulações é que elas pressupõem que o percurso do robô já está pré-definido, o que as tornam inadequadas para o controle cinemático de um robô em ambientes dinâmicos com eventos imprevisíveis ocorrendo em tempo real. Uma abordagem para esse problema, é o uso de controladores preditivos funcionando como planejadores de trajetória em tempo real, como aplicado neste trabalho.

2.1.2 Dinâmica e Controle

A modelagem dinâmica de um robô pode ser realizada por meio de diversas abordagens, sendo a abordagem de *Euler-Lagrange* uma das mais comuns. Esta abordagem permite obter as equações de movimento de um sistema com múltiplos elos de forma mais direta em comparação com a abordagem de *Newton-Euler*, que requerem um algoritmo recursivo para o cálculo das forças e torques nas juntas do robô.

As equações de *Euler-Lagrange* são baseadas no cálculo do Lagrangeano que é a diferença entre a energia cinética e potencial envolvidas na movimentação do robô, como visto nas Equações 8 e 9, o que torna o equacionamento mais simples, já que estas grandezas são escalares.

$$L = K - P \tag{8}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = 0 \tag{9}$$

em que L é o lagrangeano, K é a energia cinética e P é a energia potencial.

Essas equações de movimento são reformuladas separando os efeitos dinâmicos em matrizes e vetores, como a matriz de inércia do robô, matriz de Coriolis (que inclui efeitos como nutação e precessão) e a matriz de forças gravitacionais, denotados respectivamente por M, C e G como visto na Equação 10. É possível incluir matrizes adicionais para considerar atritos e outros efeitos.

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{10}$$

em que τ é o vetor de torques aplicado às juntas que possuem valocidade dada por $\dot{q}$ e aceleração dada por $\ddot{q}$.

As matrizes são dadas nas equações 11, 12 e 13. Em que m é a massa de cada elo, r_c é a distância da junta ao centro de massa, I é o momento de inércia do elo, R é a matriz de rotação, J_v e J_ω são as componentes de velocidade linear e angular das matrizes jacobianas.

$$M(q) = \sum_{i=1}^{N} [m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T R_i I_i R_i^T J_{\omega_i}] \tag{11}$$

$$C(q,\dot{q}) = \sum_{i=1}^{N} \frac{1}{2} \left(\frac{\partial M_{kj}}{\partial q_i} + \frac{\partial M_{ki}}{\partial q_j} - \frac{\partial M_{ij}}{\partial q_k} \right) \tag{12}$$

$$G(q) = \frac{\partial}{\partial q_k} \sum_{i=1}^{N} g^T r_{c_i} m_i \tag{13}$$

Em geral, nos manipuladores robóticos cada junta do robô é controlada individualmente por um servomotor. Um controle independente usando um PID para cada servo pode ser suficiente para atender as especificações do projeto, mas um controle não linear mais elaborado pode ser necessário para operações com maior velocidade ou exatidão. Khosla e Kanade (1989) mostram a diferença de desempenho de um PID simples em comparação a uma abordagem de controle não linear para robôs manipuladores.

A técnica de torque computado utiliza as matrizes de modelagem acima mencionadas para desacoplar os efeitos das outras juntas e "normalizar" a saída do PID para a inércia atual do sistema, permitindo o cálculo do torque de entrada para cada junta do manipulador levando-se em conta as não linearidades do sistema. A formulação geral do sinal de controle do torque computado (com uso de PID) pode ser visualizada na Equação 14.

$$u = M(\ddot{q_{ref}} + k_p e + k_i \int_0^t e dt + k_d \dot{e}) + C\dot{q} + G \tag{14}$$

em que q_{ref} é a referência, e é o erro das variáveis de junta e u é o torque ou força aplicada pelo motor na junta.

O modelo dinâmico e o controle do robô manipulador planar implementado neste trabalho basearam-se nas equações de *Euler-Lagrange* no formato matricial e na técnica de torque

computado, respectivamente Equações 10 e 14. A configuração do robô utilizada é o robô planar 2R, que é mais detalhado no Capítulo 3.

2.2 Drones

Analogamente à seção anterior, esta seção faz uma breve revisão de controle e modelagem cinemática e dinâmica de drones quadrotores e tem como base Luukkonen (2011).

2.2.1 Cinemática

Assim, como discutido na seção de robôs manipuladores, a cinemática de drones também faz uso de mudança de bases de referencial. A Equação 15, mostra a matriz de rotação para descrever a orientação do drone a partir de três ângulos relativos ao espaço cartesiano, conhecidos como ângulos de Euler ou rolagem (*roll*), arfagem (*pitch*) e guinada (*yaw*), sendo estes representados por ϕ, θ e ψ.

$$R = R_z(\psi)R_y(\theta)R_x(\phi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}_{3\times 3} \tag{15}$$

O drone está sujeito ao movimento de translação e rotação sobre seus próprios eixos, e dessa forma, para manter as próximas Equações da seção mais enxutas, os vetores descritos pelas equações 16 (posição cartesiana) e 17 (posição angular) são responsáveis por descrever a posição e orientação do quadrotor.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{16}$$

$$\eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{17}$$

A transformação de mudança de base da referência inercial para a base móvel do quadrotor em relação às velocidades angulares se dá pela matriz descrita na Equação 18. A construção dessa matriz é realizada fixando dois ângulos e movendo um de cada vez e por fim soma-se as componentes referente a cada velocidade angular conforme mostra a Equação 19.

Mais detalhes em relação à descrição da velocidade angular a partir dos ângulos de Euler podem ser encontrados em Lemos (2007).

$$W_\eta = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix}_{3\times 3} \tag{18}$$

$$\omega = \omega_\phi + \omega_\theta + \omega_\psi = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \dot{\phi} + \begin{bmatrix} 0 \\ c_\phi \\ -s_\phi \end{bmatrix} \dot{\theta} + \begin{bmatrix} c_\phi s_\theta \\ -s_\phi \\ c_\phi c_\theta \end{bmatrix} \dot{\psi} = W_\eta \dot{\eta} \tag{19}$$

Por fim, a transformação se dá diretamente pela multiplicação da matriz pelo vetor de ângulos de Euler como mostrado em 20.

$$\dot{\eta}_B = W_\eta \dot{\eta}_A \tag{20}$$

2.2.2 Dinâmica

As equações de movimento para o drone podem ser deduzidas a partir das equações Euler-Lagrange, como mostra a Equação 21, em que se aloca no mesmo vetor, a força f e o torque τ resultantes agindo sobre o drone.

$$\begin{bmatrix} f \\ \tau \end{bmatrix}_{6\times 1} = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} \tag{21}$$

O vetor de força pode ser também descrito a partir da multiplicação da matriz de rotação R, com a configuração atual dos ângulos de Euler, com o *thrust* T_B aplicado ao sistema. *Thrust* é a força resultante das hélices do quadricóptero na direção do eixo Z do sistema de coordenadas móvel, e com a multiplicação pela matriz de rotação, o vetor de força f é descrito a partir do referencial inercial, sendo a soma das acelerações cartesianas com a ação da gravidade.

$$f = RT_B = m\ddot{\xi} + mg \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{22}$$

Remanejando-se a equação, pode-se isolar $\ddot{\xi}$ como visto na Equação 23. Os valores dos três ângulos de *Euler* da qual esta equação é dependente são calculados por outras equações diferenciais em paralelo a esta durante a integração numérica para a estimação de ξ. A matriz

com os elementos A_x, A_y e A_z corresponde aos coeficientes do atrito aerodinâmico que é proporcional à velocidade cartesiana do quadricóptero.

$$\ddot{\xi} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi s_\theta c_\phi - c_\psi s_\phi \\ c_\theta c_\phi \end{bmatrix} - \frac{1}{m} \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix} \dot{\xi} \tag{23}$$

O cálculo dos torques agindo sobre o drone seguem uma equação similar à da robótica de manipuladores com uma matriz de inércia e uma matriz de Coriolis (e efeitos giroscópicos) como mostra a Equação 24, sendo que o subíndice B reforça que os torques agem sobre o corpo do drone e estão descritos pelo referencial móvel.

$$\tau = \tau_B = M(\xi)\ddot{\xi} + C(\xi,\dot{\xi})\dot{\xi} \tag{24}$$

em que a matriz de inércia M e a matriz de Coriolis C são dadas pelas Equações 25 a 27. I_x, I_y e I_z são os momentos de inércia em relação a cada eixo móvel do drone.

$$M(\eta) = W_\eta^T I W_\eta = \begin{bmatrix} I_x & 0 & -I_x s_\theta \\ 0 & Iyc_\phi^2 + I_z s_\phi^2 & (I_y - I_z)c_\phi s_\phi c_\theta \\ -I_x s\theta & (I_y - I_z)c_\phi s_\phi c_\theta & I_x s_\theta^2 + I_y s_\phi^2 c_\theta^2 + I_z c_\phi^2 c_\theta^2 \end{bmatrix} \tag{25}$$

$$C(\eta,\dot{\eta}) = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{26}$$

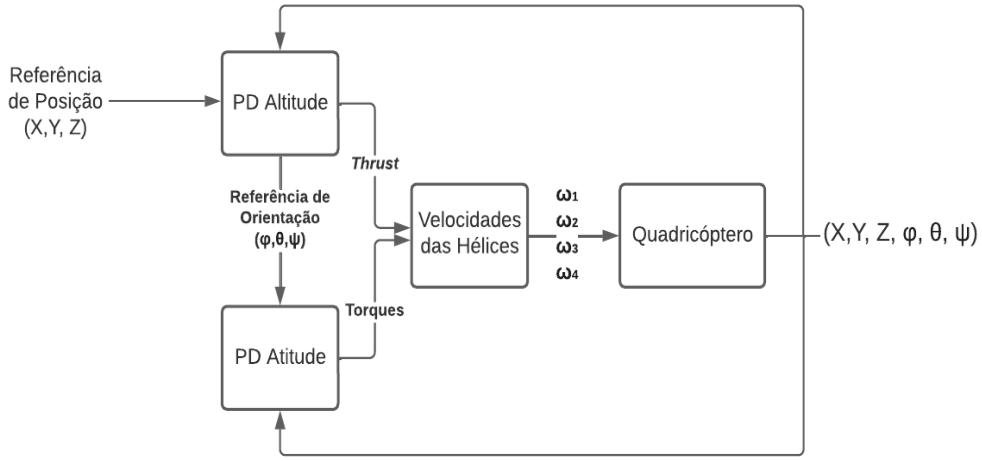
$$\begin{gathered} C_{11} = 0 \\ C_{12} = (I_y - I_z)(\dot{\theta}c\phi s\phi + \dot{\psi}s_\phi^2 c_\theta) + (I_z - I_y)\dot{\psi}c_\phi^2 c_\theta - I_x\dot{\psi}c_\theta \\ C_{13} = (I_z - I_y)\dot{\psi}c_\phi s_\phi c_\theta^2 \\ C_{21} = (I_z - I_y)(\dot{\theta}c\phi s\phi + \dot{\psi}s_\phi^2 c_\theta) + (I_y - I_z)\dot{\psi}c_\phi^2 c_\theta + I_x\dot{\psi}c_\theta \\ C_{22} = (I_z - I_y)\dot{\phi}c_\phi s_\phi \\ C_{23} = -I_x\dot{\psi}s_\theta c_\theta + I_y\dot{\psi}s_\phi^2 s_\theta c_\theta + I_z\dot{\psi}c_\phi^2 s_\theta c_\theta \\ C_{31} = (I_y - I_z)\dot{\psi}c_\phi s_\phi c_\theta^2 \\ C_{32} = (I_z - I_y)(\dot{\theta}c_\phi s_\phi s_\theta + \dot{\phi}s_\phi^2 c_\theta) + (I_y - I_z)\dot{\phi}c_\phi^2 c_\theta + I_x\dot{\psi}s_\theta c_\theta - I_y\dot{\psi}s_\phi^2 s_\theta c_\theta - I_z\dot{\psi}c_\phi^2 s_\theta c_\theta \\ C_{33} = (I_y - I_z)\dot{\phi}c_\phi s_\phi c_\theta^2 + I_x\dot{\theta}c_\theta s_\theta - I_y\dot{\theta}s_\phi^2 c_\theta s_\theta - I_z\dot{\theta}c_\phi^2 c_\theta s_\theta \end{gathered} \tag{27}$$

O modelo dinâmico do drone quadricóptero aplicado neste trabalho se fundamentou na equação de Euler-Lagrange generalizada, Equação 24, que relaciona forças e torques com as posições e velocidades lineares e angulares do sistema. As equações específicas implementadas serão apresentadas posteriormente na seção sobre o controle PD.

2.2.3 Controle PD

A arquitetura de controle PD para drones quadrotores divide-se em controle de altitude e atitude (orientação), como ilustrado pela Figura 4.

Figura 4 – Arquitetura de Controle PD de Drones



Fonte: Autoria própria (2023).

A ação do controlador PD de posição é dada pelo *thrust*, sendo calculadas pelas Equações 28 e 29, de modo similar ao torque computado visto na Equação 14 da seção de robótica que utiliza as acelerações desejadas da variável de processo. Bem como pode-se adicionar uma componente "*feedforward*" para compensar o atrito com o ar (Equação 28) e para a ação da gravidade (Equação 29). Como o controle de posição é dependente da orientação, o *thrust* é devidamente ponderado a partir dos ângulos de Euler atuais do sistema.

$$d = \ddot{\xi}_{ref} + K_p(\xi_{ref} - \xi) + K_d(\dot{\xi}_{ref} - \dot{\xi}) + \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix} \dot{\xi} \tag{28}$$

$$T = m(d_x(s_\theta c_\psi c_\phi + s_\psi s_\phi) + d_y(s_\theta s_\psi c_\phi - c_\psi s_\phi) + (d_z + g)c_\theta c_\phi) \tag{29}$$

As referências enviadas para a malha de controle de orientação são calculadas pelas Equações 30 e 31. O ângulo desejado de *roll* e *pitch* são estimados a partir da relação da ação

desejada no sistema cartesiano (ação do PD no eixo Z em comparação com o eixo X e Y do sistema de coordenadas inercial). [1]

$$\phi = \arcsin\left(\frac{d_x s_\psi - d_y c_\psi}{\sqrt{d_x^2 + d_y^2 + (d_z + g)^2}}\right) \tag{30}$$

$$\theta = \arctan\left(\frac{d_x c_\psi + d_y s_\psi}{d_z + g}\right) \tag{31}$$

A referência do ângulo *Yaw* convencionalmente é estabelecida em zero. Dessa forma tem-se uma malha simples de controle PD para a orientação do drone como visto na Equação 32 que usa apenas a matriz de inércia e os erros de posição e velocidade angular. Na ausência de referências de velocidade, pode-se adotá-las como nulas.

$$\tau_\eta = I(K_p(\eta_{ref} - \eta) + K_d(\dot{\eta}_{ref} - \dot{\eta})) \tag{32}$$

Em uma aplicação real, também é importante determinar o valor da velocidade angular da rotação das quatro hélices do drone. A magnitude da rotação média das hélices é diretamente responsável pela força do *thrust* e a diferença da velocidade angular entre as hélices ocasiona os torques que rotaciona o drone. As equações para a velocidade angular de cada uma das hélices são dadas pelas Equações 33.

$$\begin{aligned}
\omega_1 &= \frac{T}{4k} - \frac{\tau_\theta}{2kl} - \frac{\tau_\psi}{4b} \\
\omega_2 &= \frac{T}{4k} - \frac{\tau_\phi}{2kl} + \frac{\tau_\psi}{4b} \\
\omega_3 &= \frac{T}{4k} + \frac{\tau_\theta}{2kl} - \frac{\tau_\psi}{4b} \\
\omega_4 &= \frac{T}{4k} + \frac{\tau_\phi}{2kl} + \frac{\tau_\psi}{4b}
\end{aligned} \tag{33}$$

Por fim, o comportamento de um drone quadrotor pode ser descrito pelo sistema de Equações 34 a partir do *thrust* e dos torques correspondentes aos ângulos de Euler. Pelo sistema de equações é possivel observar que as acelerações angulares de *roll*, *pitch* e *yaw* estão ligadas com as simetrias do corpo rígido. Caso o drone fosse um sólido com os momentos de inércia iguais em todos os sentidos, não haveria distúrbios angulares. Dessa forma, a ação da malha de controle em yaw somente toma importância à medida que o drone não apresente simetria em relação aos momentos de inércia em X e Y.

[1] A referência Luukkonen (2011), usada para a fundamentação desta seção não apresenta uma raiz quadrada no denominador da Equação 30, o que provavelmente foi um erro de formatação do autor pois violaria a dimensionalidade da unidade de medida (o argumento do $\arcsin$ necessita ser adimensional). Em Zuo (2010), fonte citada por Luukkonen (2011), apresenta-se a raiz na equação.

$$
\begin{aligned}
\ddot{x} &= \tfrac{1}{m}((c_\psi s_\theta c_\phi + s_\psi s_\phi)T - A_x\dot{x}) \\
\ddot{y} &= \tfrac{1}{m}((s_\psi s_\theta c_\phi - c_\psi s_\phi)T - A_y\dot{y}) \\
\ddot{z} &= \tfrac{1}{m}(c_\theta c_\phi T - A_z\dot{z}) - g \\
\ddot{\phi} &= -\tfrac{1}{I_x}((I_y - I_z)\dot{\theta}\dot{\psi} + \tau_\phi) \\
\ddot{\theta} &= -\tfrac{1}{I_y}((I_z - I_x)\dot{\phi}\dot{\psi} + \tau_\theta) \\
\ddot{\psi} &= -\tfrac{1}{I_z}((I_x - I_y)\dot{\phi}\dot{\theta} + \tau_\psi)
\end{aligned}
\tag{34}
$$

Dessa forma, a Equação 34 representa o modelo dinâmico do drone e as Equações 28 a 32 apresentam os controladores PD de posição e orientação angular que serão usadas nas simulações deste trabalho.

2.3 Controle Preditivo baseado em Modelo

Esta seção tem como objetivo revisar os conceitos de controle preditivo baseado em modelo, bem como dar uma fundamentação matemática dos algoritmos usados no trabalho. Os livros base usados para esta seção são: Camacho e Bordons (2007) e Wang (2009).

2.3.1 Conceitos Básicos

Os controladores preditivos são uma classe de controladores que utilizam informações de referência e saídas futuras previstas para calcular a variável manipulada. Em contraste, os controladores tradicionais como o controlador PID usam informações atuais do erro, integral e derivada do erro do sistema para calcular a variável manipulada.

Embora a ação derivativa do controlador PID esteja relacionada com o erro futuro do sistema por meio da taxa de erro presente, este não se baseia em previsões dos estados futuros como um controlador preditivo.

Um controlador preditivo baseado em modelo (MPC) utiliza um modelo matemático do sistema, que pode ser derivado a partir de leis físicas (*white box*) ou identificado a partir dos dados (*black box*) de resposta ao degrau, ao impulso, ou outro sinal de teste.

No caso de uma planta linear modelada em espaço de estados discretos (OGATA, 2010), é possível utilizar as matrizes A, B, C das equações do sistema de forma recursiva para estabelecer uma relação entre uma saída futura y(k+n), a saída atual y(k), saídas passadas y(k-1), y(k-2), etc., sinais de controle passados u(k-1), u(k-2), etc., o sinal de controle atual e os sinais de controle futuros até o instante de predição. As Equações 35, 36 e 37 demonstram de modo recursivo como os estados futuros são calculados, a partir das matrizes A, B e C. No caso em questão usa-se Δu como sinal de entrada incremental, pois trata-se de um modelo de espaço de estados incremental.

$$x(k_i + 1|k_i) = Ax(k_i) + B\Delta u(k_i) \tag{35}$$

$$x(k_i + 2|k_i) = Ax(k_i + 1) + B\Delta u(k_i + 1) = A^2x(k_i) + AB\Delta u(k_i) + B\Delta u(k_i + 1) \tag{36}$$

$$x(k_i+N_p|k_i) = A^{N_p}x(k_i)+A^{N_p-1}B\Delta u(k_i)+A^{N_p-2}B\Delta u(k_i+1)+\cdots+AN_p - N_cB\Delta u(k_i+N_c-1) \tag{37}$$

O horizonte de predição (N_p) se refere ao número de intervalos de tempo que o controlador preditivo infere no futuro, enquanto o horizonte de controle (N_c) determina quantas saídas futuras serão projetadas.

É importante observar que $N_c \leq N_p$, pois se o horizonte de controle fosse maior, o controlador estaria tentando calcular o sinal de controle que minimiza o funcional além do período de tempo em que as previsões do sistema estão disponíveis.

Além disso, um horizonte de controle maior aumentaria a complexidade do problema de otimização, resultando em um aumento no tempo e esforço de computação. Isso é notável por conta do tamanho da matriz de transição de estado Φ, tratada adiante, estar ligado ao tamanho do horizonte de controle. Embora um horizonte de controle extenso possa oferecer uma análise mais abrangente, muitas vezes o aumento significativo no esforço computacional não resulta em melhorias proporcionais no desempenho do controle.

A matriz de transição de estados é construída a partir da Equação 37, e então transformada para a forma matricial 38 multiplicando cada elemento da matriz por C, para transformar os estados do sistema na resposta efetiva.

$$\Phi = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \cdots & CA^{N_p-N_c}B \end{bmatrix}_{N_p \times N_c} \tag{38}$$

O cálculo para a predição das saídas futuras pode então ser decomposto na contribuição dos estados atuais e dos incrementos futuros do sinal de controle, como mostra a Equação 39. Nesta equação, os estados atuais são elementos conhecidos e sua contribuição para a predição é calculada através da matriz F, presente na Equação 40. Por outro lado, a contribuição oriunda da matriz dinâmica necessita de um vetor de incrementos futuros no sinal de controle, sendo estes desconhecidos, e devem ser calculados como solução de um problema de otimização.

$$Y = Fx(k_i) + \Phi\Delta U \tag{39}$$

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}_{N_p \times 1} \tag{40}$$

2.3.2 Otimização

Com as equações do modelo de predição estabelecidas, o cálculo para os incrementos do sinal de controle advém da minimização da função custo descrita na Equação 41. Em que R_s é um vetor de referência futura e $\bar{R}$ é um peso para a regularização da saída. Caso não exista referências futuras, o vetor R_s torna-se inteiramente composto pela referência atual, assumindo que esta irá se manter pelo restante do horizonte de predição.

$$min\ S = (R_s - Y)^T(R_s - Y) + \Delta U^T \bar{R} \Delta U \tag{41}$$

Substituindo Y dado na Equação 39, a equação torna-se dependente apenas dos incrementos do sinal de controle. Então pode-se aplicar a derivada na função custo e igualar a zero para encontrar o mínimo como visto na Equação 42.

$$\frac{\partial S}{\partial \Delta U} = -2\phi^T(R_s - Fx(k_i)) + 2(\phi^T\phi + \bar{R})\Delta U = 0 \tag{42}$$

Por fim, têm-se a lei de controle preditivo na Equação 43, sendo ΔU um vetor de tamanho N_c com os valores do incremento do sinal de controle para N_c tempos de amostragem futuros. No entanto, apenas o primeiro o elemento (incremento para o instante atual) é efetivamente aplicado na planta a cada instante de amostragem.

$$\Delta U = (\phi^T\phi + \bar{R})^{-1}\phi^T(R_s - Fx(k_i)) \tag{43}$$

2.3.3 MPC com restrições

Quando existem restrições em um problema de controle MPC, pode não ser possível obter uma solução diretamente a partir da derivada parcial da função custo dada na Equação 42. Nesses casos, é necessário recorrer à programação quadrática (QP) para encontrar uma solução factível (vale-se ressaltar de que a QP também pode ser usada em problemas sem res-

trição, como mostrado adiante na Equação 50). A função custo quadrática, que pode ser escrita na forma apresentada na Equação 44, é composta por uma matriz H que pondera os termos quadráticos e cruzados e um vetor b que pondera os termos lineares. Embora uma constante f_0 possa ser inserida como um termo independente, ela não tem relação com as variáveis do problema e, portanto, não desempenha um papel significativo na minimização. Dessa forma, o termo f_0 pode ser descartado para os cálculos seguintes sem perda de informação.

$$minS = \frac{1}{2}\Delta U^T H \Delta U + b^T \Delta U + f_0 \tag{44}$$

Problemas com restrições de igualdade são substancialmente mais fáceis de resolver, uma vez que essas restrições podem eventualmente ser substituídas umas nas outras, simplificando assim o problema.

A solução de um problema de otimização com restrições pode ser obtida através da adição de multiplicadores de Lagrange à função custo, conforme apresentado na Equação 45. Nessa equação, M' é a matriz dos coeficientes das restrições e γ é o valor da igualdade. As restrições são expressas pela Equação 46. Esse método de otimização é conhecido como Método dos Multiplicadores de Lagrange e é usado para maximizar ou minimizar uma função sujeita a uma ou mais restrições de igualdade.

$$minJ = \frac{1}{2}\Delta U^T H \Delta U + b^T \Delta U + f_0 + \lambda^T (M' \Delta U - \gamma) \tag{45}$$

$$M'x = \gamma \tag{46}$$

Assim, como em problemas de minimização irrestritos, a solução analítica para problemas de minimização com restrições envolve a derivação da função custo.

Para obter a solução ótima, as Equações 45 e 46 são derivadas parcialmente em relação às variáveis ΔU e em relação aos multiplicadores de Lagrange, e então igualadas a zero conforme mostram as Equações 47 e 48. Isso resulta na Equação 49, que fornece os valores dos multiplicadores de Lagrange.

$$\frac{\partial S}{\partial \Delta U} = H\Delta U + b + M'^T \lambda = 0 \tag{47}$$

$$\frac{\partial S}{\partial \lambda} = M'\Delta U - \gamma = 0 \tag{48}$$

$$\lambda = -(M'H^-1M'^T)^-1(\gamma + M'H^-1b) \tag{49}$$

A solução ótima x^o para um problema de minimização sem restrições pode ser obtida a partir da Equação 50, que envolve a matriz H e o vetor b. No entanto, quando há restrições

de igualdade, a ação de controle ótima $u*$ é calculada de forma a minimizar a função custo S, respeitando as restrições do sistema, conforme demonstrado na Equação 51.

$$x^o = -H^{-1}b \tag{50}$$

$$x = x^o - H^{-1}M'^T\lambda \tag{51}$$

Problemas com restrições de igualdade podem ser simplificados através da combinação de equações, como por exemplo, quando a resolução está restrita a dois planos não paralelos, onde estas duas restrições geram uma única restrição de reta. No entanto, para problemas com restrições de desigualdade, não é possível simplificar as restrições diretamente. Uma abordagem comum para a resolução deste tipo de problema é através das condições de *Kuhn-Tucker*. Essas condições, representadas pelas Equações 52, 53, 54 e 55, são utilizadas para verificar se uma solução candidata é factível e, caso não seja, encontrar os multiplicadores de Lagrange correspondentes para movê-la para a região factível.

$$H\Delta U + b + M'^T\lambda = 0 \tag{52}$$

$$M'\Delta U - \gamma \leq 0 \tag{53}$$

$$\lambda(M'\Delta U - \gamma) = 0 \tag{54}$$

$$\lambda \geq 0 \tag{55}$$

O sinal do multiplicador λ determina se uma restrição é ativa ou inativa. Se λ é negativo, a restrição correspondente é redundante, ou seja, a solução ótima irrestrita já se encontra dentro da região delimitada pela restrição. Se λ é positivo, a solução ótima está fora do limite estabelecido pela restrição e, portanto, precisa ser recalculada. Se λ é nulo, a solução ótima está exatamente sobre a fronteira da restrição.

Ao utilizar o algoritmo de programação quadrática para lidar com restrições de desigualdade, as restrições correspondentes aos multiplicadores de Lagrange negativos devem ser "ignoradas” ao aplicar a atualização de x em 51. A condição 55, garante que apenas os multiplicadores de sinal positivo sejam considerados nesse processo, direcionando a solução ótima na direção da fronteira da restrição correspondente.

2.3.4 MPC não-linear

Algumas formulações de MPC foram desenvolvidas para lidar com problemas de controle mais específicos, como o MPC robusto, o MPC adaptativo e o MPC não-linear.

O *Nonlinear* MPC (NMPC ou NLMPC) é usado para plantas com dinâmicas não lineares. Nesse caso, é necessário usar um modelo não linear da planta para calcular as saídas futuras de forma recursiva, além de um otimizador de programação matemática não linear.

Quando a função custo não pode ser expressa na forma quadrática, isto é, esta função contém uma não-linearidade mais ampla, ou quando há a presença de restrições não lineares, uma forma comum de resolver o problema é usar *Sequential Quadratic Programming* (SQP).

Sejam f(x) e g(x) equações não-lineares quaisquer, onde f(x) é a função custo e g(x) é a função da restrição, estas podem ser minimizadas numericamente através do método de Newton em conjunto com os multiplicadores de Lagrange. A função custo é descrita de uma forma generalizada como mostra a Equação 56.

$$S(x,\lambda) = f(x) + g(x)^T\lambda \tag{56}$$

Assim como nas formulações anteriores, é aplicado o cálculo da derivada parcial em relação à variável x e à variável λ, e a igualdade é estabelecida com zero. As Equações 57 e 58 usam a notação de gradiente para uma representação mais concisa.

$$\nabla S(x,\lambda) = \nabla f + \nabla g^T\lambda = 0 \tag{57}$$

$$\nabla S(x,\lambda) = \begin{bmatrix} \nabla f + \nabla g^T\lambda \\ g \end{bmatrix} \tag{58}$$

Para aplicar o método de Newton, é necessário desenvolver uma aproximação de primeira ordem para o ponto de interesse, como mostrado na Equação 59. Como se trata de uma aproximação de um gradiente, a segunda ordem do gradiente aparece na forma de uma matriz Hessiana.

Por fim, o cálculo para o incremento do multiplicador de Lagrange e das variáveis de interesse é apresentado nas Equações 60 e 61.

$$\nabla S(x,\lambda) = \nabla S(x_0,\lambda_0) + \nabla^2 S(x_0,\lambda_0) \begin{bmatrix} \delta x \\ \delta\lambda \end{bmatrix} = 0 \tag{59}$$

$$\begin{bmatrix} \delta x_k \\ \delta\lambda_k \end{bmatrix} = -(\nabla^2 S(x_k,\lambda_k))^{-1}\nabla S(x_k,\lambda_k)) \tag{60}$$

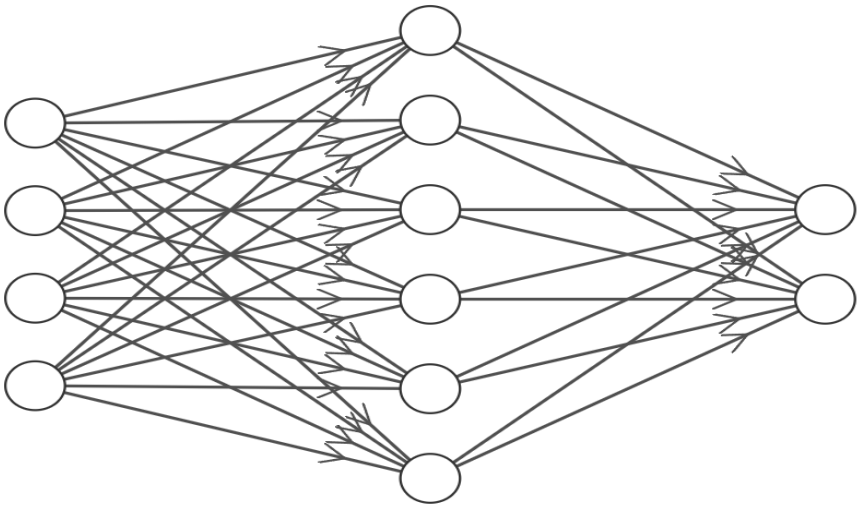
$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} + \begin{bmatrix} \delta x_k \\ \delta \lambda_k \end{bmatrix} \tag{61}$$

2.4 Redes Neurais Artificiais

Esta seção oferece uma breve introdução às redes neurais artificiais do tipo *feedforward*, ou redes *Multi-Layer Perceptron* (MLP). Esta descrição é fundamentada por Raschka e Mirjalili (2017) e pela documentação oficial do MATLAB.

As redes MLP, conforme o próprio nome sugere, empregam múltiplos *perceptrons* organizados em camadas. A estrutura típica de uma rede MLP é exemplificada pela Figura 5.

Figura 5 – Diagrama de uma Rede Neural MLP



Fonte: Autoria própria (2023).

Uma das principais vantagens das MLPs é a sua capacidade de resolver problemas de classificação que não são linearmente separáveis, exemplificado pelo clássico problema do "XOR", onde perceptrons simples não são capazes de solucioná-lo. Além disso, as MLPs demonstram uma propriedade de atuar como aproximadores universais em problemas de regressão, de modo que conseguem "mapear" uma superfície não linear.

A função matemática que representa uma rede neural envolve uma série de operações de multiplicação matricial e a aplicação de funções de ativação não lineares. Esta dinâmica é expressa pela Equação 62.

$$\mathbf{y} = f(\mathbf{W}_n f(\mathbf{W}_{n-1} f(\ldots f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) \ldots) + \mathbf{b}_{n-1}) + \mathbf{b}_n) \tag{62}$$

onde $\mathbf{W}_i$ é a matriz de pesos da i-ésima camada e $f()$ é a função de ativação.

Entre as funções de ativação mais populares estão a ReLU, sigmoidal, tangente hiperbólica e a função de base radial. Neste trabalho, a função de tangente hiperbólica foi a escolhida.

Os pesos das redes são ajustados utilizando a técnica de *Backpropagation* que tem como base o método do gradiente descendente. Esta técnica atualiza os pesos das camadas anteriores "retro-propagando" o erro da rede neural através de derivadas parciais e a aplicação da regra da cadeia. Assim aplicando a atualização de pesos adequado a cada conexão, consegue-se minimizar a função custo de erro da rede, conforme a Equação 63.

$$\begin{aligned} \mathbf{E}(\mathbf{y}, \hat{\mathbf{y}}) &= \tfrac{1}{2} \textstyle\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ \Delta \mathbf{w} &= -\eta \nabla \mathbf{E}(\mathbf{x}, \mathbf{w}) \end{aligned} \tag{63}$$

onde ∇ representa o gradiente, **w** os pesos da rede, **E** a função custo do erro, y_i e $\hat{y}_i$ são os valores real e a estimativa da rede, e η a taxa de aprendizado (*learning rate*).

Foram desenvolvidas variantes mais eficientes do gradiente descendente, incluindo a implementação de um η adaptativo, que se ajusta ao longo do treinamento, e a introdução de "momento". Este adiciona uma espécie de "inércia" aos ajustes de peso, contribuindo para que o processo de otimização não fique preso em mínimos locais.

Dentre os métodos de treinamento disponíveis no MATLAB, destacam-se duas abordagens mais sofisticadas: o método de Levenberg-Marquardt (LMM) e o método de regularização bayesiana (BRM). O LMM é uma adaptação mais eficiente do método de *Gauss-Newton*, o qual substitui a matriz hessiana por aproximações via jacobiana, conforme descrito por Yu e Wilamowski (2011). Por outro lado, o BRM integra princípios de inferência bayesiana ao processo de treinamento, uma técnica detalhada por Burden e Winkler (2009). Este último foi o método escolhido para uso neste trabalho, devido a sua eficiência em evitar sobreajuste no treinamento.

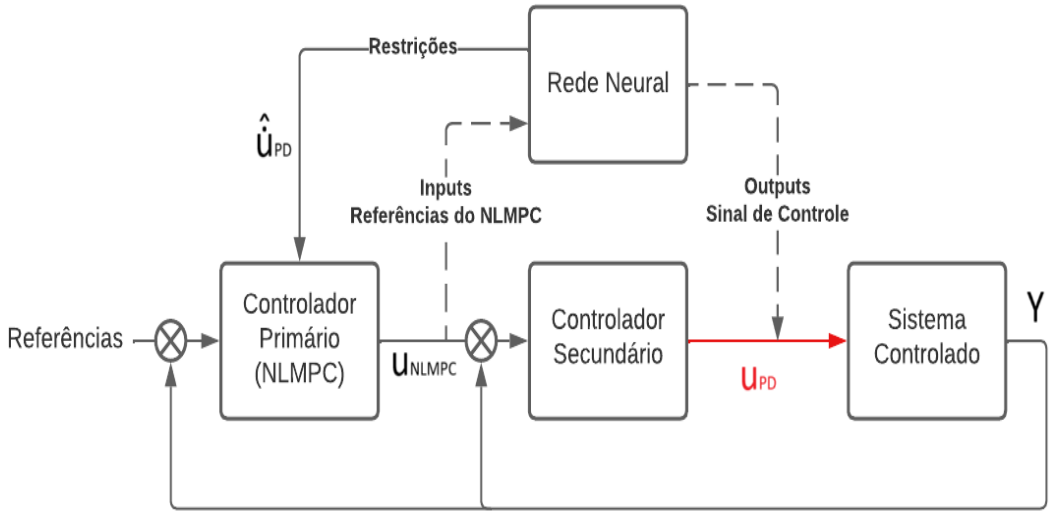
Em resumo, este capítulo apresentou os conceitos teóricos que fundamentam o desenvolvimento deste trabalho. Foram abordadas as técnicas de modelagem de robôs manipuladores, drones quadricópteros e seus respectivos controladores, os quais representarão a malha secundária. O controlador NLMPC será utilizado como malha primária planejando trajetórias livres de colisão e a rede neural servirá para fazer o mapeamento da saída da malha primária para a secundária e então incorporada às restrições do NLMPC. Dessa forma, os tópicos abordados neste capítulo fornecem a base teórica para o desenvolvimento da abordagem de controle em cascata aplicados a sistemas robóticos, proposta nesta dissertação.

3 MATERIAIS E MÉTODOS

3.1 Arquitetura Proposta

Neste trabalho, propõe-se o uso de redes neurais para inferir restrições no sinal de controle de sistemas controlados por NLMPC em cascata. Em uma configuração típica de cascata, o NLMPC na malha primária calcula referências cinemáticas que são enviadas para controladores na malha secundária, como PDs. A Figura 6, ilustra a configuração, onde a rede neural é treinada para fazer o mapeamento entre a saída do NLMPC e a saída do controlador secundário (destacado em vermelho).

Figura 6 – Redes Neurais em Sistemas em Cascata



Fonte: Autoria própria (2023).

Como o NLMPC não tem acesso à saída do controlador secundário, não há como fazer a aplicação de restrições nessa saída diretamente. Assim, o NLMPC passa a incorporar as restrições inferidas pela rede neural durante o planejamento da trajetória, garantindo que os sinais de controle da malha secundária (derivada de torques e forças no caso deste trabalho) não ultrapassem certos limites.

O presente trabalho se baseou no exemplo de controlador NLMPC do MATLAB intitulado *Plan and Execute Collision-Free Trajectories Using KINOVA Gen3 Manipulator*. Nesse exemplo, o controlador é responsável por gerar pontos de referência intermediários entre a posição inicial e a posição desejada do efetuador final do manipulador. Dessa forma, o controlador resolve implicitamente o problema da cinemática inversa enquanto cumpre o percurso respeitando as restrições não lineares do ambiente.

Todavia, no exemplo do MATLAB, o NLMPC calculava toda a trajetória *offline* para depois enviá-la a um controle de torque computado. Enquanto neste trabalho, o NLMPC opera *online* em conjunto com a malha de torque computado em uma arquitetura em cascata.

Em contrapartida, as malhas secundárias usadas neste trabalho (robô manipulador planar 2R e drone quadricóptero) foram programadas a partir das equações de Euler-Lagrange e das leis de controle por torque computado vistas na sessão de fundamentação teórica.

3.2 Robô Manipulador

O primeiro sistema usado como estudo de caso da arquitetura proposta é o de um robô manipulador, que além de apresentarem restrições no sinal de controle calculadas via redes neurais, tem a necessidade do uso de um controlador NLMPC por conta da cinemática apresentar não linearidades. Isso ocorre porque as posições desejadas no planejamento das trajetórias estão no espaço operacional (cartesiano), enquanto os estados fornecidos para a malha de controle secundária precisam estar no espaço de junta (posições e velocidades angulares). Essa relação é não linear pois há a presença de funções trigonométricas nas matrizes de transformação como visto na Equação 2 na fundamentação teórica.

Existem ainda as restrições de percurso que estão ligadas à necessidade de evitar colisões entre o manipulador e os obstáculos. Assim restrições são impostas na norma da distância euclidiana entre as juntas e os obstáculos, o que também introduz não linearidades no problema.

3.2.1 Propriedade do Robô

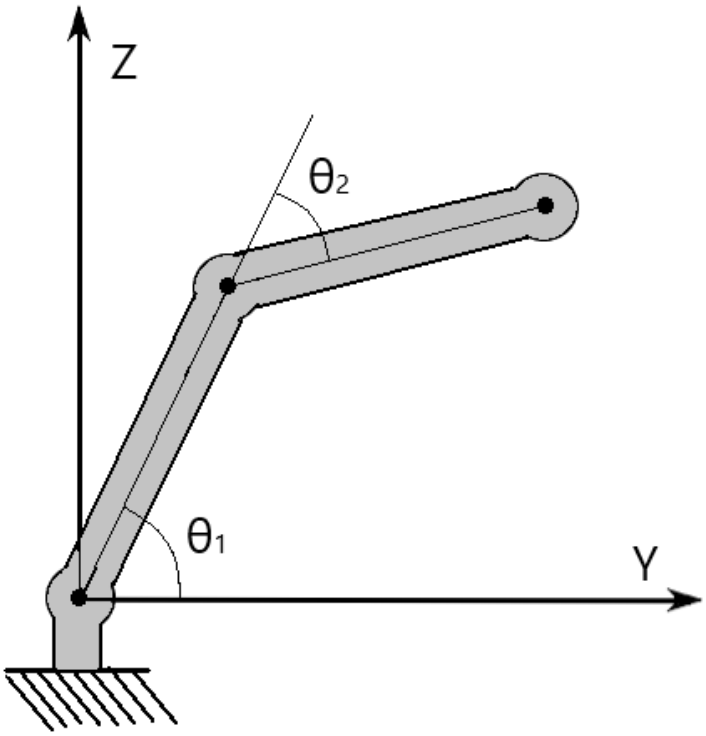
Para fins de prova de conceito, foi utilizada uma configuração de robô planar duplo rotativo 2R com ambos os elos de comprimento unitário como mostrado na Figura 7, sendo essencialmente um pêndulo duplo com controladores de posição acoplados em cada elo. Por conter apenas quatro variáveis (posição e velocidade angular das duas juntas), essa configuração foi escolhida por ser de mais fácil implementação e treinamento das redes neurais usadas na abordagem proposta por este trabalho.

A matriz de transformação homogênea da base para o efetuador final é apresentada na Equação 64.

$$T_F^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{12} & -s_{12} & c_1 + c_{12} \\ 0 & s_{12} & c_{12} & s_1 + s_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times4} \tag{64}$$

Nesta modelagem, o robô planar tem sua área de trabalho restrita ao plano YZ, o que permite que a resolução da cinemática inversa seja facilmente deduzida através de arco cosseno e arco tangente. Devido a essa relação trigonométrica, há duas soluções possíveis para uma mesma posição cartesiana expressas nas Equações 65 e 66.

Figura 7 – Robô Planar 2R



Fonte: Autoria própria (2023).

$$q2 = acos((y^2 + z^2 - 2)/2) \tag{65}$$

$$q1 = atan2(z(1 + c_2) - ys_2, y(1 + c_2) + zs_2) \tag{66}$$

As matrizes de inércia, coriolis e gravidade do robô em questão, utilizadas tanto na equação do movimento quanto na lei de controle, estão expressas nas Equações 67, 68 e 69.

$$M = \begin{bmatrix} c_2 + \frac{1003}{600} & \frac{c_2}{2} + \frac{403}{1200} \\ \frac{c_2}{2} + \frac{403}{1200} & \frac{403}{1200} \end{bmatrix}_{2\times2} \tag{67}$$

$$C = \begin{bmatrix} \frac{-\dot{q_2}s_2}{2} & -s_2\frac{\dot{q_1}+2\dot{q_2}}{4} \\ \frac{\dot{q_1}s_2}{4} & 0 \end{bmatrix}_{2\times2} \tag{68}$$

$$G = \begin{bmatrix} \frac{981}{200}c_{12} + \frac{2943}{200}c_1 \\ \frac{981}{200}c_{12} \end{bmatrix}_{2\times1} \tag{69}$$

Para cada etapa do percurso, são utilizadas trajetórias de 0,25 segundos com os parâmetros K_p = 188 e K_d = 198 (obtidos empiricamente) no controlador CTC-PD presente na malha secundária. No entanto, a aplicação de um degrau neste sistema pode resultar numa

ação muito agressiva para o controlador, pois acarretaria em uma mudança abrupta na referência, especialmente devido à ação derivativa quando se encontra na ausência de filtros. Por isso, são estipulados perfis de trajetória polinomiais com base na posição, velocidade e aceleração iniciais e finais, os quais fazem uma transição suave na referência antes de enviá-la para o controlador PD, como discutido na Seção 2.1.

Para criar o percurso, a malha primária envia apenas as posições e velocidades angulares para a malha secundária, o que condiz com o perfil de trajetória cúbico por usar somente posição, velocidade e instante de tempo iniciais e finais (Equação 6).

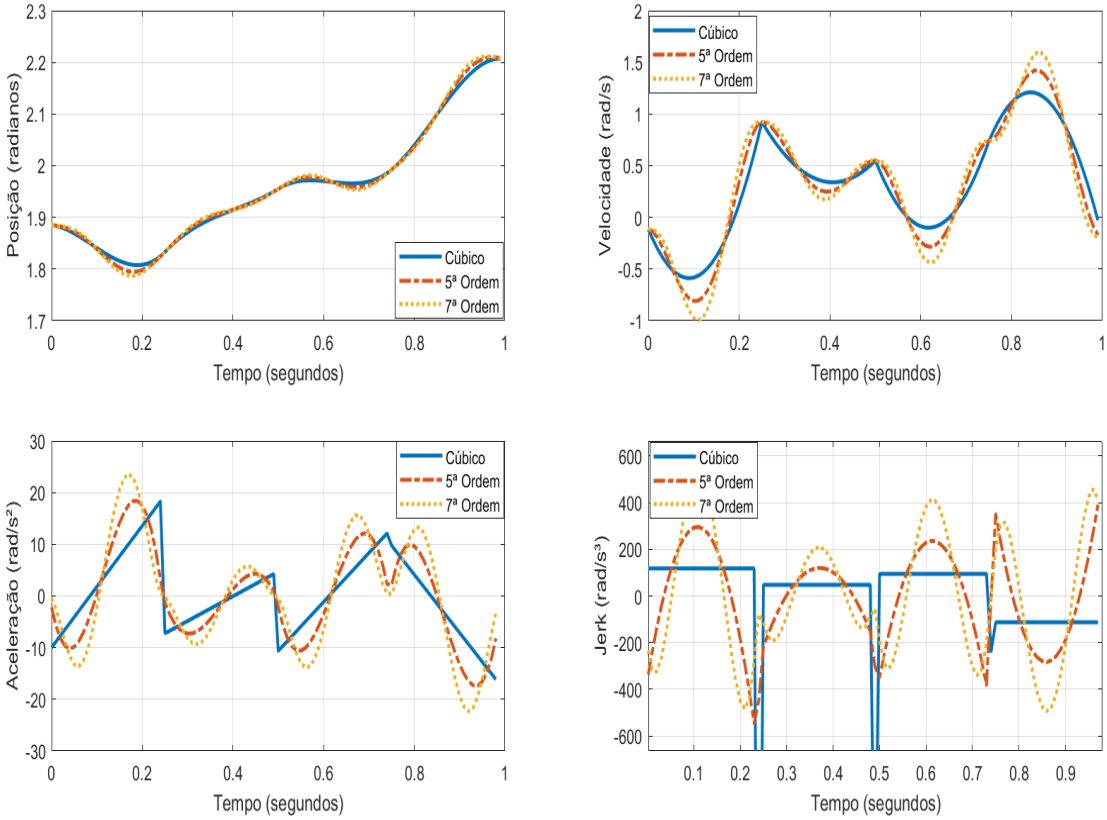
No entanto, ao sequenciar as trajetórias cúbicas de cada tempo de amostragem (t_0 a t_f), ocorre uma descontinuidade na aceleração angular, o que afeta diretamente o torque, conforme mostrado na Equação 10 no Capítulo de fundamentação teórica. Uma das componentes da equação é o produto da matriz de inércia M pela aceleração $\ddot{q}$, desse modo a mudança brusca de velocidades angulares enviadas pelo NLMPC ocasionam uma aceleração infinita e por conseguinte um torque infinito.

O problema da descontinuidade do torque foi resolvido ao adotar um polinômio de quinta ordem e impondo uma aceleração angular nula como condições de contorno para cada etapa do percurso, conforme a Equação 7. No entanto, o problema persistiu para a derivada do torque, por conta da derivada do torque necessitar de uma suavidade na transição do *jerk*.

Para solucionar esse problema, aumentou-se a dimensão da matriz e foram impostos *jerks* angulares nulos como condições de contorno, resultando em um planejamento de trajetória de sétima ordem, como visto na Equação 70. A Figura 8, compara o comportamento da posição angular, velocidade angular, aceleração angular e *jerk* angular para polinômios de ordem 3, 5 e 7. É possível observar que o *jerk* do polinômio cúbico "explode" ao derivar descontinuidades da aceleração angular, o polinômio de quinta ordem mantém uma ordem de grandeza consistente, mas sofre de descontinuidades a cada 0,25 s (tempo de amostragem do MPC e duração de cada intervalo de trajetória), enquanto o polinômio de sétima ordem fornece um comportamento mais próximo do desejado.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 & t_0^6 & t_0^7 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 & 6t_0^5 & 7t_0^6 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 & 30t_0^4 & 42t_0^5 \\ 0 & 0 & 0 & 6 & 24t_0 & 60t_0^2 & 120t_0^3 & 210t_0^4 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 & t_f^6 & t_f^7 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 & 6t_0^5 & 7t_0^6 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 & 30t_0^4 & 42t_0^5 \\ 0 & 0 & 0 & 6 & 24t_f & 60t_f^2 & 120t_f^3 & 210t_f^4 \end{bmatrix}_{8\times 8} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ k_6 \\ k_7 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ a_0 \\ j_0 \\ q_f \\ v_f \\ a_f \\ j_f \end{bmatrix} \tag{70}$$

Figura 8 – Comparativo da Ordem Polinomial das Trajetórias



Fonte: Autoria própria (2023).

3.2.2 Linearidade de Robôs Manipuladores

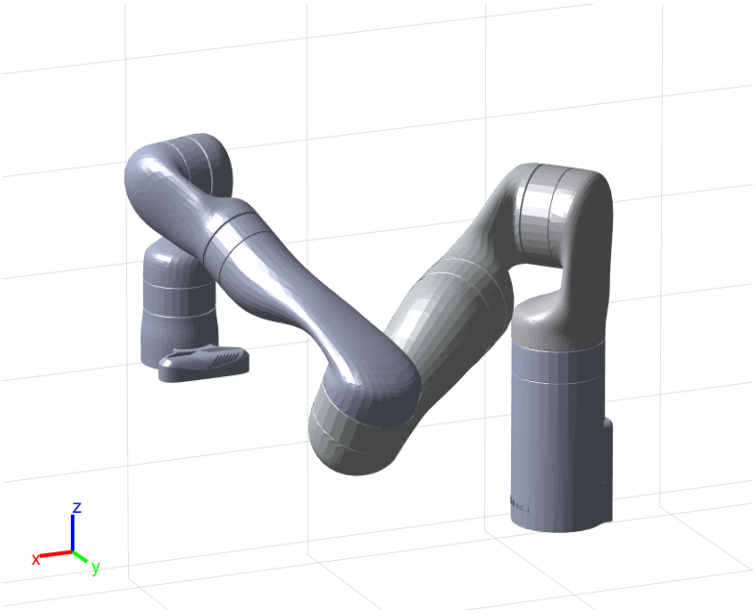
A técnica de torque computado ou dinâmica inversa é uma abordagem utilizada na malha secundária que tem a capacidade de "linearizar" a planta do ponto de vista do MPC. Quando as matrizes M, C e G do modelo coincidem com as matrizes do robô real, o que significa que há uma modelagem perfeita, as Equações 10 e 14 praticamente se anulam, restando apenas uma relação de igualdade entre o sinal de controle e a aceleração angular.

Assim, em um caso ideal, o robô manipulador operaria exatamente como um duplo integrador, tendo torques "equivalentes" a acelerações angulares como entrada, e posições angulares como saída.

Para avaliar o desempenho da técnica de torque computado, foram realizados testes no robô Kinova de sete graus de liberdade (por se tratar de um modelo mais complexo do que o planar 2R), ilustrado na Figura 9. Aplicaram-se então degraus de diferentes amplitudes para uma lei de controle com matrizes do modelo com um erro de 20% em relação à matriz verdadeira. As Figuras 10 e 11 mostram as respostas normalizadas do sistema. É possível perceber que o torque computado é relativamente robusto, pois mesmo com imprecisões no modelo e com um robô complexo como o Kinova, o sistema se comporta de modo muito similar

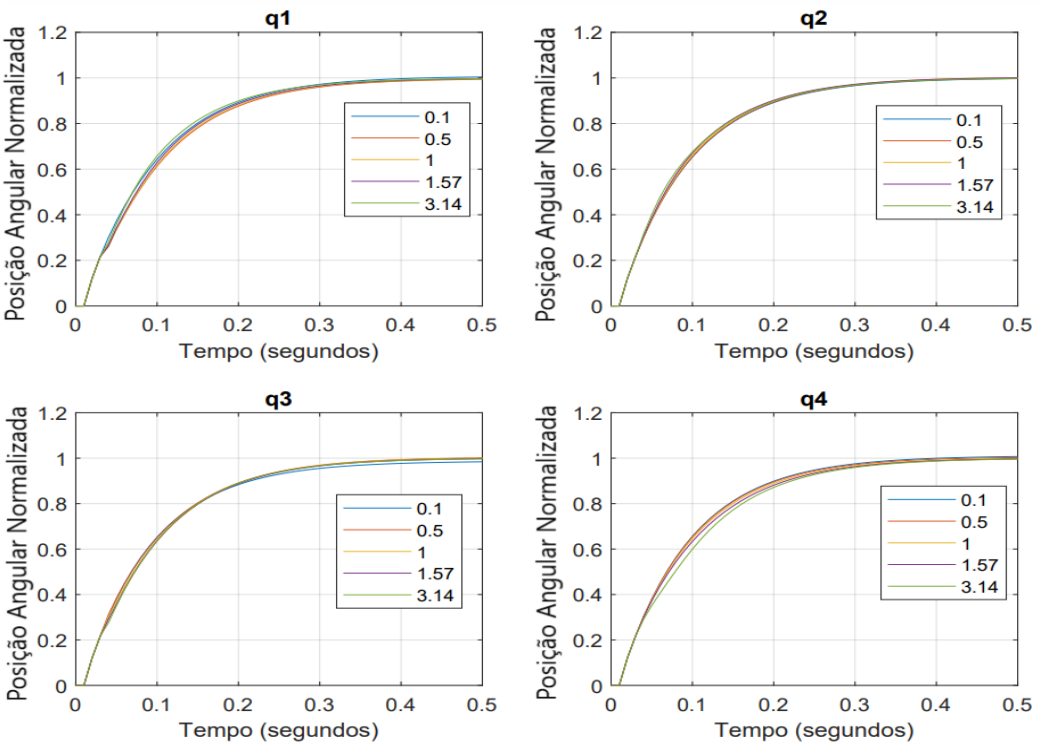
a uma planta linear. Isso quer dizer que ele obedece à propriedade de superposição, típica de sistemas lineares onde para uma determinada variação na entrada (por exemplo, um incremento ΔU no sinal de controle), ocorrerá uma variação proporcional na saída (um incremento ΔY).

Figura 9 – Robô Kinova



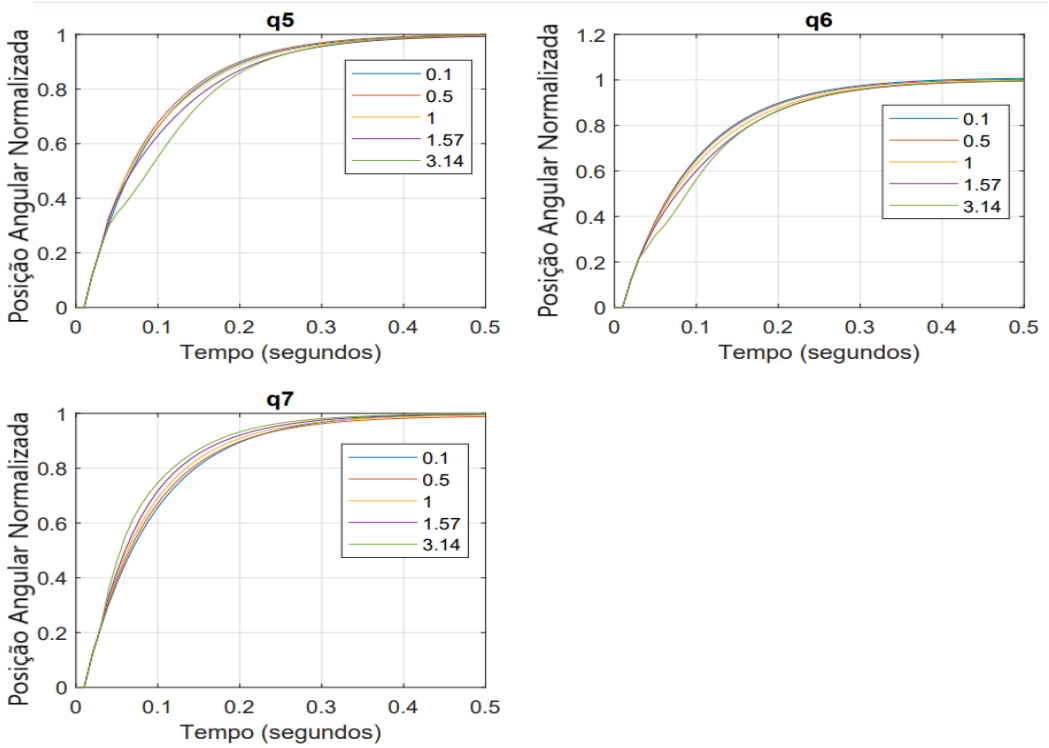
Fonte: Autoria própria (2023).

Figura 10 – Resposta ao Degrau: Juntas 1 a 4



Fonte: Autoria própria (2023).

Figura 11 – Resposta ao Degrau: Juntas 5 a 7



Fonte: Autoria própria (2023).

3.2.3 Configuração do MPC

Os algoritmos de controle do tipo MPC e NLMPC usados nesta dissertação estão disponíveis na *toolbox* MPC do software MATLAB.

A *toolbox* de MPC cria um objeto para o controlador NLMPC com os seguintes parâmetros:

- nlobj.Ts
- nlobj.PredictionHorizon
- nlobj.ControlHorizon
- nlobj.Model.StateFcn
- nlobj.Optimization.CustomCostFcn
- nlobj.Optimization.CustomIneqConFcn
- nlobj.States

Nesta dissertação, os parâmetros do MPC tais que tempo de amostragem, horizontes de predição, controle e matrizes de ponderação (mostradas adiante) foram estabelecidas empiricamente.

O parâmetro *nlobj.Ts* refere-se ao tempo de amostragem, o qual foi estipulado como 0,25 segundos. Se o tempo de amostragem for muito alto, o robô dará passos grandes demais, o que pode dificultar a geração de pontos intermediários suficientes para contornar um obstáculo. Restrições nas velocidades geradas pelo MPC podem ajudar a resolver esse problema, mas isso pode tornar a movimentação do robô mais lenta.

Por outro lado, um tempo de amostragem muito baixo pode exigir maior custo computacional e resultar em torques mais agressivos. Isso tornaria a relação mapeada pela rede neural mais complexa e irregular. Isso dificulta o aprendizado da rede neural, pois pequenas variações nas entradas geram grandes variações não lineares nas saídas.

Os parâmetros *nlobj.PredictionHorizon* e *nlobj.ControlHorizon* são os horizontes de predição e controle, os quais foram definidos como N_p = 6 e N_c = 2, respectivamente. Esses horizontes foram os menores encontrados para permitir que o robô desvie da maioria dos obstáculos com sucesso. Um horizonte de predição muito curto pode impedir que o MPC antecipe o movimento necessário a tempo de evitar o obstáculo, enquanto horizontes de predição muito longos podem tornar a resposta do robô excessivamente lenta.

O elemento *nlobj.Model.StateFcn* é o modelo de predição do NLMPC, o qual é aproximadamente um duplo integrador, como discutido na subseção anterior.

O item *nlobj.Optimization.CustomCostFcn* recebe um arquivo $.m$ contendo a função custo que será otimizada pelo NLMPC seguindo o formato da Equação 71.

As duas primeiras componentes da Equação 71 são responsáveis por minimizar a distância entre a posição do efetuador final, como indicado pela Equação 72, e a posição desejada no espaço cartesiano.

Para garantir que a aproximação ao *setpoint* seja mais suave, o NLMPC incorpora uma "regularização" com os estados de velocidades generalizadas e acelerações (componentes $\dot{q}$ e U), mas não há impacto de forma direta no sinal de controle de torque (que de fato realiza um esforço físico).

$$S = \sum_{k=1}^{N_p-1} \left((r_d - \hat{r}_k)^T Q_r (r_d - \hat{r}_k)\right) + (r_d - \hat{r}_{N_p})^T Q_t (r_d - \hat{r}_{N_p}) + \sum_{k=1}^{N_p} \left(\dot{q}_k^T Q_v \dot{q}_k\right) + \sum_{k=1}^{N_c} \left(U_k^T Q_u U_k\right) \tag{71}$$

$$\hat{r} = \begin{bmatrix} \cos(\hat{q}_1) + \cos(\hat{q}_1 + \hat{q}_2) \\ \sin(\hat{q}_1) + \sin(\hat{q}_1 + \hat{q}_2) \end{bmatrix} \tag{72}$$

As matrizes de ponderação utilizadas no cálculo da função custo são Q_r, Q_t, Q_v, e Q_u, que correspondem respectivamente ao peso do erro em regime transiente do horizonte de predição, ao erro ao final do horizonte de predição, à regularização das velocidades generalizadas e à regularização das entradas (acelerações) fornecidas pelo NLMPC ao modelo de predição (duplo integrador). Os valores de cada matriz são os seguintes:

- $Q_r = 3I$
- $Q_t = 5I$
- $Q_v = 0{,}25I$
- $Q_u = 0{,}025I$

onde a matriz identidade I é de tamanho 2, por se tratar de uma espaço de trabalho cartesiano bidimensional e também porque robô contem dois atuadores.

Por fim, os elementos *nlobj.States* e *nlobj.Optimization.CustomIneqConFcn* cumprem o papel de estabelecer as restrições do controlador. O primeiro item é responsável pelas restrições de mínimo e máximo das varáveis de posições e velocidades angulares enviadas à malha secundária, e o segundo item recebe um arquivo $.m$ para as restrições não lineares.

As saídas do NLMPC (entradas da malha secundária) estão restritas por:

- $|q_2| \leq \frac{7\pi}{8}$ rad
- $|\dot{q}| \leq 1$ rad/s
- Restrição não linear referente a colisão de cada elo com o obstáculo: usa-se a função *checkCollision* do MATLAB.

A restrição sobre q_2 serve para que o segundo elo não vire 180º e colida com o elo da junta anterior. A função *checkCollision* é uma função da *toolbox* de Robótica do MATLAB que retorna a distância euclidiana mínima entre a superfície de dois sólidos. Neste trabalho trata-se da distância entre a superfície dos cilindros em relação a uma esfera (problema ilustrado no Capítulo 4).

3.2.4 Redes Neurais

O objetivo da rede neural desenvolvida nesta seção é inferir os picos da derivada do sinal de controle gerados pelos controladores PD da malha secundária, fornecendo essa informação ao NLMPC. Dessa forma, o NLMPC pode incluir as restrições na taxa de variação do torque dentro do problema de otimização e gerar trajetórias mais suaves e que respeitem os limites dinâmicos dos atuadores.

Para calcular a curva de torque, sua derivada e o pico máximo em um intervalo de trajetória, é necessário conhecer as condições de contorno, incluindo as posições angulares iniciais e finais e as velocidades angulares iniciais e finais. Como discutido anteriormente, as acelerações e os *jerks* foram definidos como nulos para suavizar a resposta.

Uma abordagem alternativa, sem perda de informação, é utilizar coordenadas generalizadas incrementais em vez de coordenadas absolutas para o ponto final. Dessa forma, torna-se possível calcular a curva de torque, sua derivada e o pico máximo sem depender explicitamente

da posição e da velocidade final. Essa formulação tornou o mapeamento da rede neural mais eficiente para a inferência do torque de pequenas movimentações do robô, pois cria-se uma relação direta entre pequenas variações de posição e velocidade com pequenas variações no sinal de controle.

Desta forma, estabeleceu-se que as redes neurais teriam as seguintes entradas e saídas:

- Entrada 1: q_1 inicial
- Entrada 2: q_2 inicial
- Entrada 3: $\dot{q}_1$ inicial
- Entrada 4: $\dot{q}_2$ inicial
- Entrada 5: Δq_1
- Entrada 6: Δq_2
- Entrada 7: $\Delta \dot{q}_1$
- Entrada 8: $\Delta \dot{q}_2$
- Saída 1: max($|\dot{u}_1|$)
- Saída 2: max($|\dot{u}_2|$)

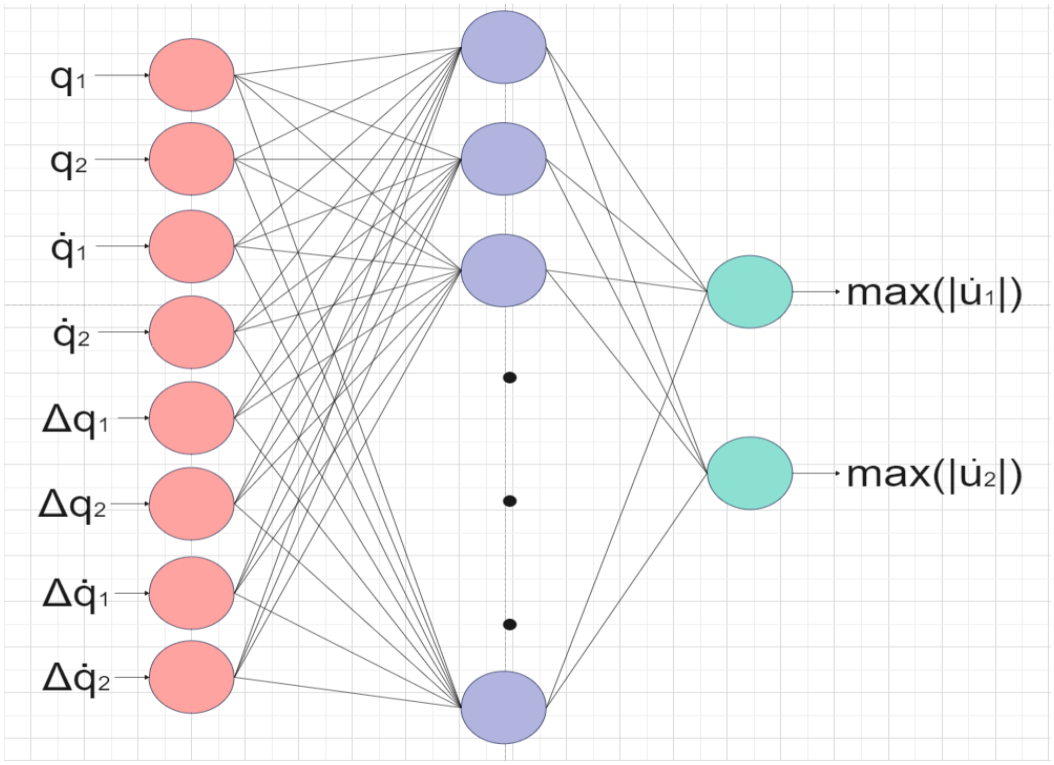
A Figura 12 ilustra, com uma quantidade de neurônios e camadas escondidas ainda não definida, as entradas e saídas da rede neural *feedforward*. A função de ativação para as camadas escondidas foi adotada como a tangente hiperbólica e manteve-se a camada de saída como linear.

Os dados de entrada de velocidade angular foram gerados aleatoriamente com função de distribuição uniforme (através da função *rand*) e independentes entre si. No entanto, a geração dos dados de posição exigiu um pouco mais de cuidado.

Se os dados de posição angular fossem gerados aleatoriamente de forma independente dos dados de velocidade angular, poderia haver inconsistências cinemáticas na base de dados. Por exemplo, uma amostra poderia ter um grande incremento positivo na posição, mas uma velocidade negativa. Embora esse tipo de situação não seja fisicamente impossível, seria incomum para o contexto do MPC, que tipicamente planeja movimentos contínuos em uma única direção.

Uma base de dados com tais inconsistências cinemáticas, apesar de matematicamente válida, seria majoritariamente composta por situações pouco plausíveis para o problema em questão. Além disso, variações muito grandes e aleatórias na posição angular dificilmente representariam movimentos suaves desejados na prática.

Figura 12 – Arquitetura da Rede Neural



Fonte: Autoria própria (2023).

Para evitar esses problemas, optou-se por vincular o incremento da posição angular às velocidades angulares, conforme a Equação 73. Dessa forma, as variações de posição geradas são mais verossímeis e consistentes cinematicamente para o contexto do controle de trajetória. Isso permite gerar uma base de dados mais realística e adequada para o treinamento da rede neural.

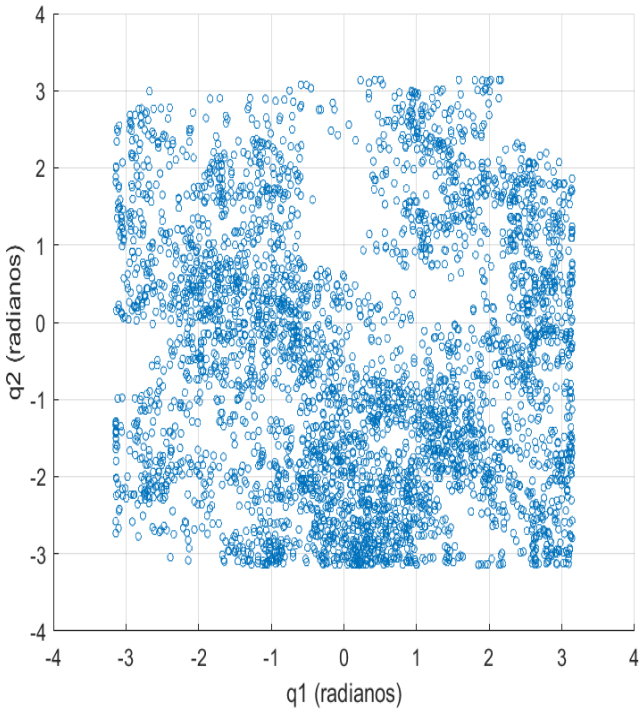
$$q_i = q_{i-1} + \frac{a}{2} \cdot \dot{q}_{i-1} + b \tag{73}$$

em que a e b são variáveis para adicionar um pouco de variância nos dados, respectivamente nos intervalos de [0,1] e [-0,1; 0,1].

Assim, a geração dos dados de posição desejada foi baseada na ideia do ”zig-zag", com restrições para garantir que as posições estivessem dentro do intervalo [-π, π]. No entanto, para um conjunto pequeno de dados, não há garantia de que o "andar do bêbado” cubra todo o espaço de junta, o que pode gerar concentração de dados em determinadas regiões, como mostrado na Figura 13, que apresenta 4000 dados. À medida que o número de dados aumenta, a distribuição das posições tende a se tornar mais homogênea, como demonstrado na figura 14 com 40.000 dados.

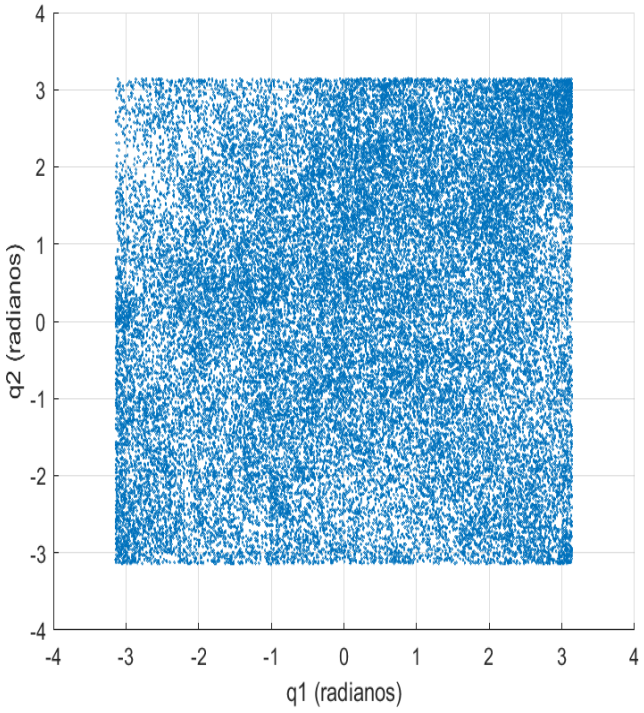
O método de validação empregado foi o K-fold conforme elaborado em (RASCHKA; MIRJALILI, 2017), com k = 5, visando testar diferentes arquiteturas de rede neural. Inicialmente, os dados de entrada e saída foram normalizados pelo valor máximo de cada *feature* e *label*, de

Figura 13 – Dados Polarizados



Fonte: Autoria própria (2023).

Figura 14 – Dados Adequados



Fonte: Autoria própria (2023).

modo que todos os valores RMSE apresentados correspondam à saída da rede neural operando no intervalo de [-1,1]. Posteriormente, 20% dos dados foram selecionados aleatoriamente do conjunto de dados para serem usados como dados de teste, após a escolha da arquitetura. Foi adotado como padrão a regularização bayesiana como método de treinamento.

As Tabelas 1 e 2 apresentam, respectivamente, os valores médios de desempenho de RMSE para os 5 folds do treinamento e validação de cada arquitetura. As linhas das tabelas indicam a quantidade de neurônios em cada camada, enquanto as colunas representam o número de camadas escondidas. Os valores obtidos nas duas tabelas são bastante próximos, indicando que o modelo não sofreu *overfitting* durante o treinamento, considerando que foram utilizados 40.000 dados para o treinamento.

Após a análise das tabelas, optou-se por utilizar a configuração de 3 camadas com 20 neurônios em cada camada, totalizando 60 neurônios nas camadas escondidas.

Tabela 1 – Desempenho Médio do Treinamento das Arquiteturas

Neurônios/Camadas	1	2	3
10	0,028039	0,010489	0,009454
20	0,016914	0,00871	**0,008196**
30	0,013682	0,008206	-
50	0,011148	-	-

Fonte: Autoria própria (2023).

Tabela 2 – Desempenho Médio da Validação das Arquiteturas

Neurônios/Camadas	1	2	3
10	0,027963	0,01050	0,009438
20	0,016864	0,008713	**0,008206**
30	0,013718	0,008208	-
50	0,011147	-	-

Fonte: Autoria própria (2023).

A rede com melhor desempenho de validação foi escolhida dentre as 5 redes treinadas para a arquitetura selecionada, e seus valores de RMSE de treinamento, validação e teste estão apresentados na Tabela 3. Observa-se que há pouco indício de sobreajuste, uma vez que os dados de teste, que não foram utilizados durante o treinamento com 5-fold, apresentaram um valor muito próximo do desempenho de treinamento. A Figura 15 mostra graficamente a predição da rede neural para alguns dados de teste, com os dados já ajustados para a escala da derivada do torque.

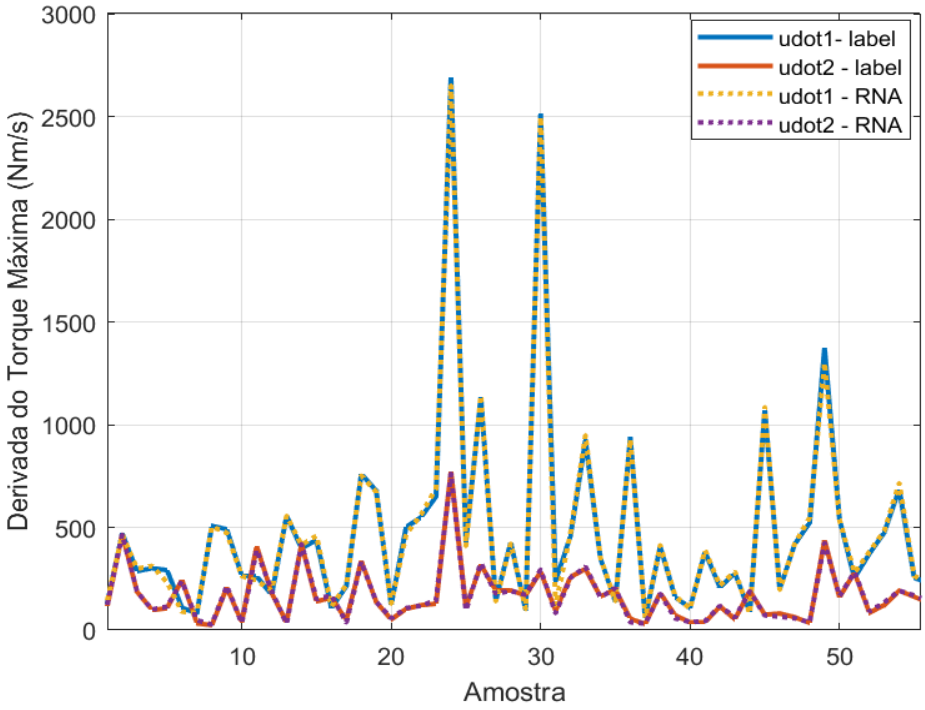
Tabela 3 – Desempenho da Melhor Rede Encontrada

Treinamento	Validação	Teste
0,008220	0,007965	0,008278

Fonte: Autoria própria (2023).

A Figura 15 apresenta resultados aparentemente próximos do real, mas ainda pode haver valores atípicos em regiões do espaço de dados que não foram bem cobertas, as linhas

Figura 15 – Desempenho dos Dados de Teste



Fonte: Autoria própria (2023).

sólidas do gráfico referem-se ao pico real da derivada do torque provida pelos controladores PD, enquanto as linhas tracejadas apontam a inferência da rede neural. A Tabela 4 mostra que o maior erro encontrado no conjunto de testes foi de 223,03 Nm/s para a junta 1 e 71,59 Nm/s para a junta 2. Embora esses erros absolutos sejam consideráveis, eles representam percentualmente um erro relativamente baixo, dada a escala dos rótulos.

Tabela 4 – Margem de Erro

	$\dot{u}_1$	$\dot{u}_2$
Erro Máximo	223,03	71,59
Escala do Label	5840,6	1897,7
Erro Relativo a Escala	3,8186%	3,7722%

Fonte: Autoria própria (2023).

A rede neural desenvolvida acrescenta uma carga computacional que é proporcional ao número de iterações do *solver* de otimização. Esse custo adicional foi reduzido significativamente ao implementar a rede neural de forma direta, multiplicando manualmente as entradas pelas matrizes de pesos previamente calculadas, ao invés de utilizar a *toolbox* de redes neurais do MATLAB para aplicação. Essa abordagem permitiu reduzir o tempo de uma inferência neural de 33,4 ms para apenas 3,4 ms na máquina utilizada, viabilizando melhor a aplicação em tempo real.

3.3 Drone Quadrotor

3.3.1 Modelo e Controladores PD

O modelo de quadricóptero usado neste trabalho se baseia nos parâmetros de massa, momento de inércia e coeficiente de arrasto descritos por Luukkonen (2011). Apenas o momento de inércia em relação ao eixo móvel X que foi reduzido pela metade, pois dessa forma cria-se uma assimetria responsável por distúrbios no ângulo de Euler ψ responsável pela rotação em *yaw*.

- $m = 0{,}468kg$
- $A = 0{,}25\frac{kg}{s}$
- $I_x = 2{,}428 \times 10^{-3}kgm^2$
- $I_y = 4{,}856 \times 10^{-3}kgm^2$
- $I_z = 8{,}801 \times 10^{-3}kgm^2$

Os controladores PD foram desenvolvidos para o controle de posição e orientação conforme as Equações 28 e 32. Todavia, como não são calculadas referências de velocidade, foi estabelecido $\dot{\eta}_{ref}$ como nulo. Desta forma, foram testados empiricamente diferentes ganhos para o PD e estabeleceu-se os ganhos K_p e K_d para as seis variáveis de processo na Tabela 5. A malha secundária com os ganhos empíricos apresentou desempenho suficientemente adequado para as simulações tratadas neste trabalho.

Tabela 5 – Ganhos dos Controladores PD do Drone

	X	Y	Z	ϕ	θ	ψ
K_p	330	450	450	4500	4200	2400
K_d	15	16,5	12	72	84	60

Fonte: Autoria própria (2023).

Percebe-se que os ganhos para a malha de orientação estão em uma ordem de grandeza mais elevada. Isso se dá porque o controle de orientação deve ser mais rápido, pois é necessário que a orientação esteja posicionada na direção correta quando o *thrust* calculado pelo PD de posição atue. Caso contrário, o drone iria mover-se para uma direção distinta da desejada.

3.3.2 Controlador NLMPC

A função custo do NLMPC foi a mesma aplicada no robô manipulador, presente na Equação 71, com a diferença de que os estados de posição e velocidade do drone já estão expressos

em coordenadas cartesianas e desta forma não há a não linearidade vista na Equação 72. Por outro lado, um controle NLMPC ainda se faz necessário por conta das não linearidades das restrições geométricas (distância euclidiana aos obstáculos) e restrições no sinal de controle por meio da rede neural desenvolvida a seguir.

Por sua vez, os parâmetros do NLMPC para o drone foram estabelecidos de forma empírica e distinguem-se dos usados para o robô manipulador, e encontram-se listados a seguir, onde apenas o peso da matriz Q_r se manteve.

- $Q_r = 3I$
- $Q_t = 10I$
- $Q_v = 0{,}1I$
- $Q_u = I$

em que as matrizes I são identidades de tamanho 3, pois o drone opera nas três dimensões do espaço cartesiano.

Como o ambiente de trabalho do drone é maior do que o do robô de dois graus de liberdade, pode-se optar por um tempo de amostragem maior na malha primária, visto que o drone é um sistema móvel e pode se mover em distâncias maiores do que o manipulador que estava limitado ao comprimento dos elos. Assim foi escolhido o tempo de amostragem de 1 segundo.

De forma similar, as restrições de velocidade cartesiana e angular foram estabelecidas de forma que o "passo" do controle cinemático seja preciso o suficiente para desviar dos obstáculos usados nas simulações de teste.

- $|\dot{X}| \leq 0{,}5\frac{m}{s}$
- $|\dot{Y}| \leq 0{,}5\frac{m}{s}$
- $|\dot{Z}| \leq 0{,}5\frac{m}{s}$
- $|\dot{\phi}| \leq 5\frac{rad}{s}$
- $|\dot{\theta}| \leq 5\frac{rad}{s}$
- Restrição não linear da colisão do drone com o obstáculo

A restrição não linear para o caso do drone é substancialmente mais simples do que as de manipuladores, visto que no caso dos robôs é necessário calcular a distância euclidiana entre cada elo do robô com cada obstáculo da trajetória. Como o drone não é um corpo articulado, somente uma distância por obstáculo é suficiente no decorrer da simulação.

3.3.3 Rede Neural

A abordagem aplicada no robô foi reaplicada para a construção e treinamento do drone, sendo que a maior diferença entre ambos os problemas é a quantidade de variáveis de estado.

No caso do robô manipulador, haviam duas variáveis de junta (posições ângulares) e suas respectivas velocidades. No caso dos drones quadricópteros, há as posições cartesianas, os ângulos de Euler e suas respectivas velocidades, o que totalizaria 12 variáveis de estado. Além disso, o incremento em cada uma das variáveis precisaria ser considerado para dessa forma cobrir todas as condições de contorno.

Algumas dessas variáveis podem ser desconsideradas a priori, como as posições cartesianas do drone, pois a dinâmica do drone não irá mudar dada a sua posição em X, Y e Z.

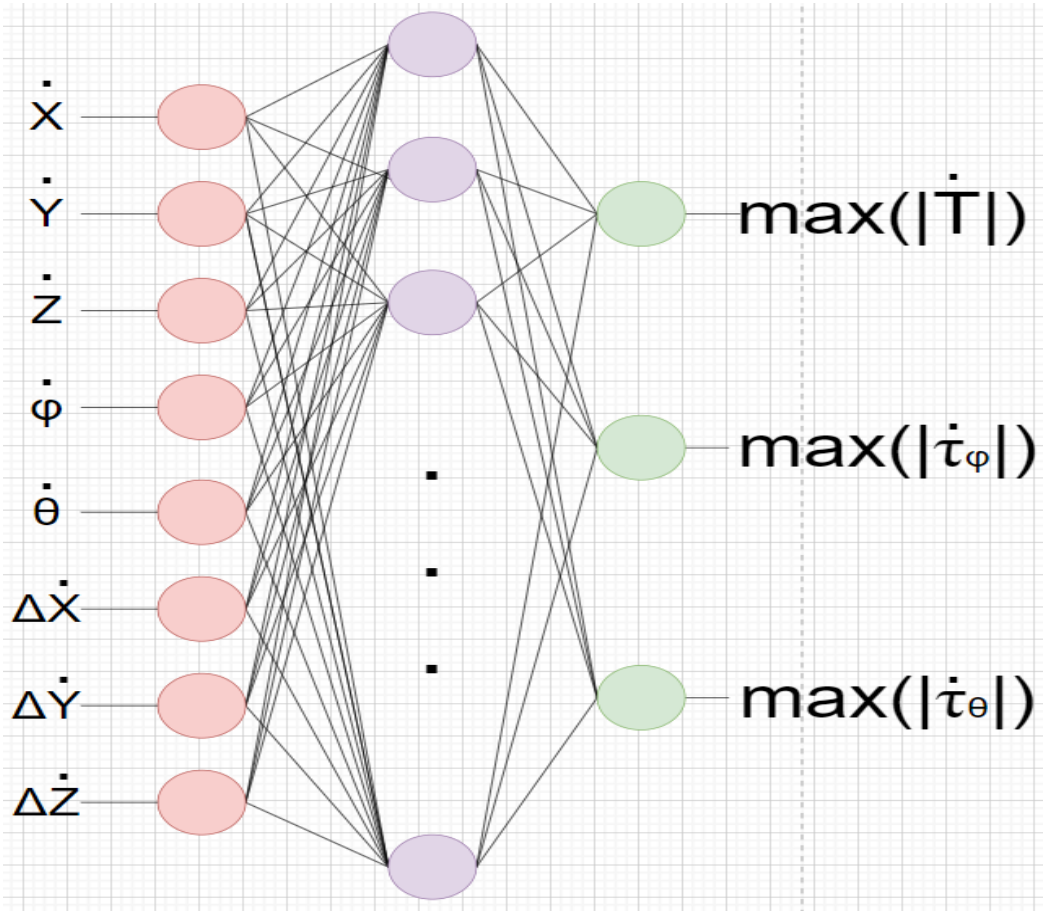
Outras variáveis como o *yaw* ("ψ"), por manter um valor muito próximo de zero durante toda a simulação, podem ser descartadas por terem um impacto desprezável nas forças e torques resultantes sobre o sistema.

Os ângulos de Euler ϕ e θ possuem maior variação, todavia algumas de suas condições de contorno não demonstraram impacto significativo durante o treinamento da rede neural, e desta forma também foram descartados.

Assim sendo, manteve-se somente as velocidades angulares iniciais de *roll* e *pitch*, as velocidades cartesianas iniciais e os seus incrementos. As entradas e saídas das arquiteturas de rede neural usadas estão listadas a seguir e ilustradas pela Figura 17.

- Entrada 1: $\dot{X}$ inicial
- Entrada 2: $\dot{Y}$ inicial
- Entrada 3: $\dot{Z}$ inicial
- Entrada 4: $\dot{\phi}$
- Entrada 5: $\dot{\theta}$
- Entrada 6: $\Delta\ \dot{X}$
- Entrada 7: $\Delta\ \dot{Y}$
- Entrada 8: $\Delta\ \dot{Z}$
- Saída 1: max($|\dot{T}|$)
- Saída 2: max($|\dot{\tau}_{\phi}|$)
- Saída 3: max($|\dot{\tau}_{\theta}|$)

Figura 16 – Arquitetura da Rede Neural para o Drone



Fonte: Autoria própria (2023).

Foram treinadas redes neurais de diferentes arquiteturas, Tabelas 6 e 7, onde observou-se que a rede com três camadas e 20 neurônios foi a melhor arquitetura encontrada. Todavia, a melhoria (redução do RMSE) é relativamente menor ao aumentar os números de neurônios e camadas quando comparadas com as Tabelas 1 e 2 do robô manipulador.

Tabela 6 – Desempenho Médio do Treinamento das Arquiteturas para o Drone

Neurônios/Camadas	1	2	3
10	0,048595	0,043100	0,042189
20	0,042215	0,041231	**0,040771**
30	0,041711	0,041118	-
50	0,041505	-	-

Fonte: Autoria própria (2023).

O desempenho da melhor arquitetura ao aplicar os dados de teste sofreu apenas um pequeno aumento no RMSE, conforme mostra a Tabela 8, o que não apresenta indícios de sobreajuste.

O erro máximo da rede neural com a base de dados teste para as três saídas pode ser verificado na Tabela 9, onde os valores da primeira coluna estão na unidade de derivada de força (N/s) e as demais em derivada de torque (Nm/s). Percebe-se pelo erro relativo, de que o

Tabela 7 – Desempenho Médio da Validação das Arquiteturas para o Drone

Neurônios/Camadas	1	2	3
10	0,048544	0,043095	0,042155
20	0,042052	0,040873	**0,040731**
30	0,041698	0,040771	-
50	0,041481	-	-

Fonte: Autoria própria (2023).

Tabela 8 – Desempenho da Melhor Rede Encontrada para o Drone

Treinamento	Validação	Teste
0,040817	0,040434	0,042194

Fonte: Autoria própria (2023).

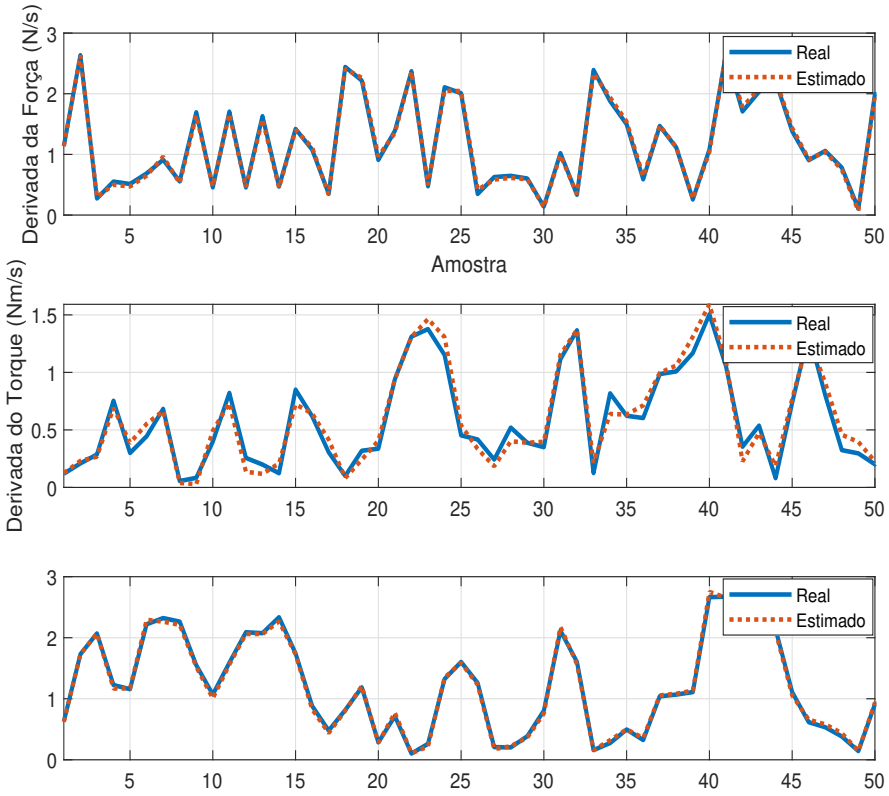
torque no sentido de rotação *roll* registrou um valor considerável, este erro pode ser visualizado melhor na Figura 17.

Tabela 9 – Margem de Erro para o Drone

	$\dot{T}$	$\dot{\tau}_\phi$	$\dot{\tau}_\theta$
Erro Máximo	0,1212	0,2977	0,0968
Escala do Label	3,0500	1,8768	3,0692
Erro Relativo a Escala	3,9738%	15,8621%	3,1539%

Fonte: Autoria própria (2023).

Figura 17 – Comparativo do Desempenho da Rede Neural: (a) - Thrust, (b) - Roll, (c) - Pitch



Fonte: Autoria própria (2023).

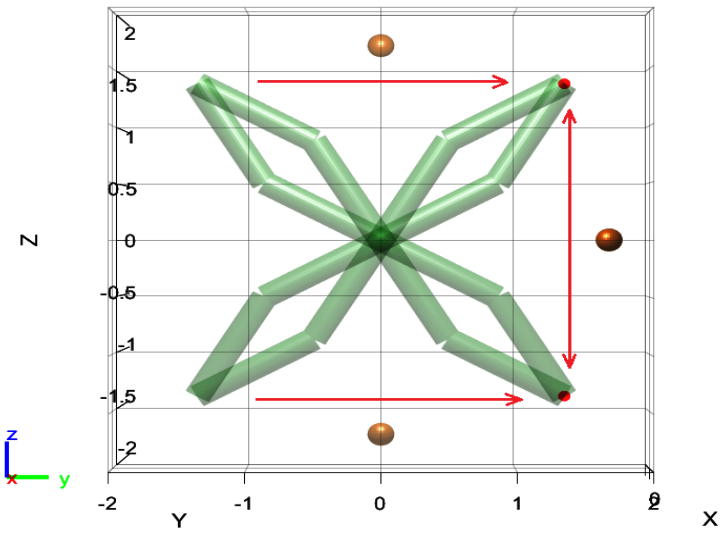
4 RESULTADOS

Este capítulo apresenta os resultados das simulações realizadas para validar a estratégia de controle proposta. Todos os resultados apresentados foram obtidos por meio de simulações computacionais utilizando o software MATLAB. O problema de otimização do controlador NLMPC foi resolvido pelo pelo *solver Find minimum of constrained nonlinear multivariable function* (FMINCON), o qual tem como um dos seus métodos de resolução o SQP apresentado no Capítulo 2. As simulações foram executadas em um laptop com processador AMD Ryzen 5 PRO 4650U 2.10 GHz. O objetivo das simulações é demonstrar o desempenho do controlador proposto em diferentes cenários, avaliando sua capacidade de gerar trajetórias que respeitem as restrições nos sinais de controle. Os estudos de caso envolvem a aplicação da arquitetura proposta para um robô manipulador planar de 2 juntas e um drone quadrotor.

4.1 Resultados do Manipulador

Foram realizadas 8 trajetórias que percorreram os quatro pontos (-1,366,1,366), (1,366,1,366), (1,366,-1,366) e (-1,366,-1,366), cobrindo todos os quadrantes do plano euclidiano. A Figura 18 mostra os pontos de partida e chegada utilizados no experimento.

Figura 18 – Pontos de Partida e Chegada - (metros)



Fonte: Autoria própria (2023).

As trajetórias foram agrupadas em pares, uma vez que há duas configurações possíveis de juntas para uma mesma posição inicial.

- Trajetórias 1 e 2: partem do segundo quadrante para o primeiro quadrante.
- Trajetórias 3 e 4: partem do quarto quadrante para o primeiro quadrante.

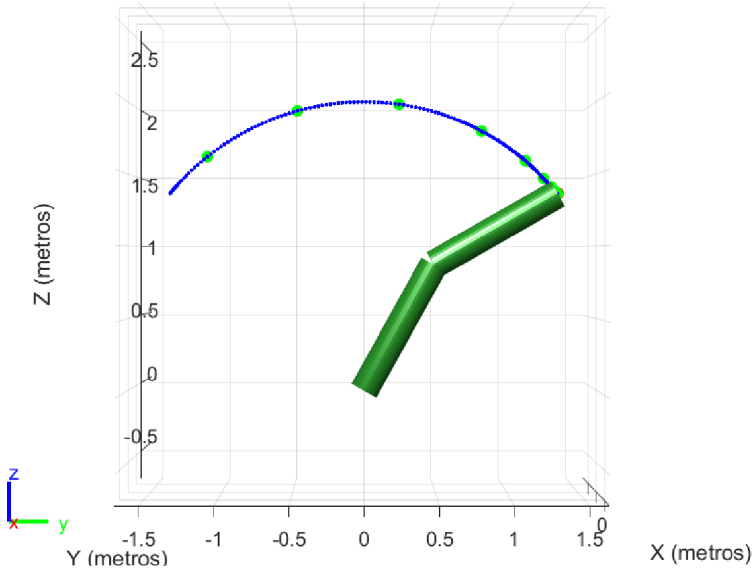
- Trajetórias 5 e 6: partem do terceiro quadrante para o quarto quadrante.
- Trajetórias 7 e 8: partem do primeiro quadrante para o quarto quadrante.

É importante destacar que não foram realizados testes de trajetórias entre o segundo e o terceiro quadrante, visto que esse problema é equivalente ao teste de trajetórias entre o primeiro e o quarto quadrante. Da mesma forma, as transições entre os quadrantes superiores e inferiores foram realizadas em apenas um sentido, já que o problema é equivalente em ambos os sentidos.

Por outro lado, a transição entre o primeiro e o quarto quadrante tem uma diferença fundamental: em um o robô manipulador está abaixando-se e no outro sentido está levantando-se.

Os obstáculos foram posicionados em locais estratégicos: (0;1,7), (1,7;0) e (0;-1,7), com o objetivo de forçar o robô a seguir uma rota diferente daquela que seria percorrida em uma situação livre de obstáculos como ilustrado na Figura 19.

Figura 19 – Trajetória sem Obstáculo



Fonte: Autoria própria (2023).

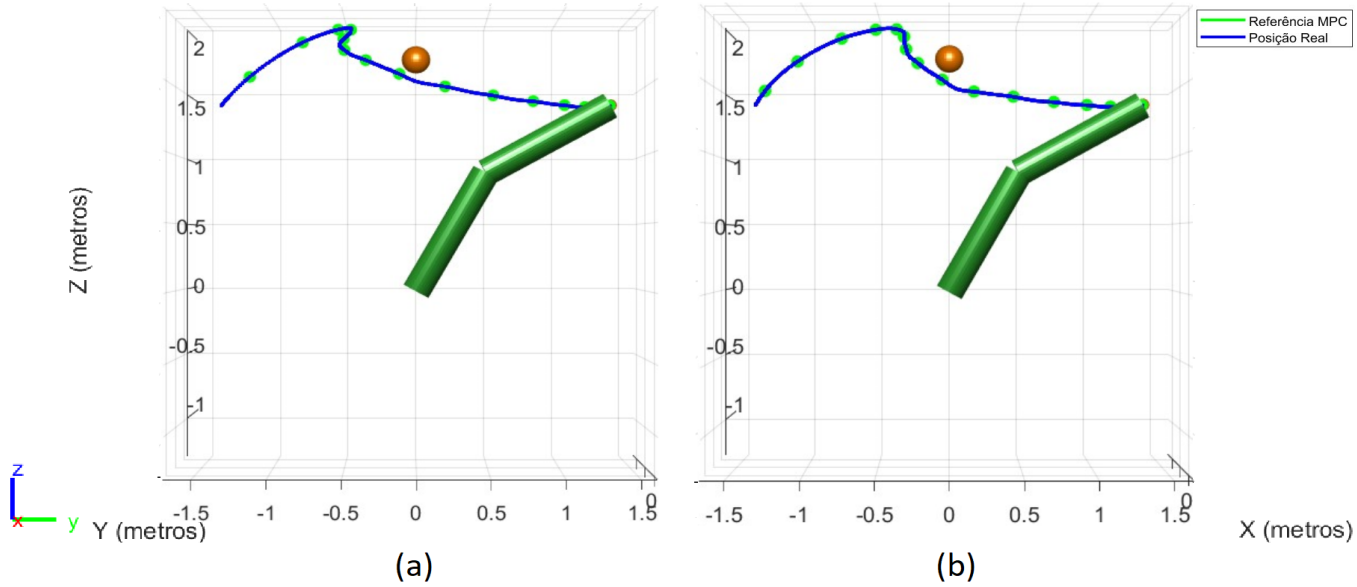
4.1.1 *Hard Constraints*

Como uma primeira abordagem para aplicação de restrições, foram adotadas *hard constraints*, as quais não admitem folga na busca de uma solução factível.

Em contrapartida a Figura 19, observa-se na Figura 20 que o robô, partindo do estado de juntas (-120º,-30º), necessita realizar um desvio brusco ao se aproximar do obstáculo. No entanto, é perceptível que a curva realizada na presença da restrição da derivada do torque de $[\pm 100, \pm 50]$ Nm/s foi mais suave em comparação com a trajetória irrestrita.

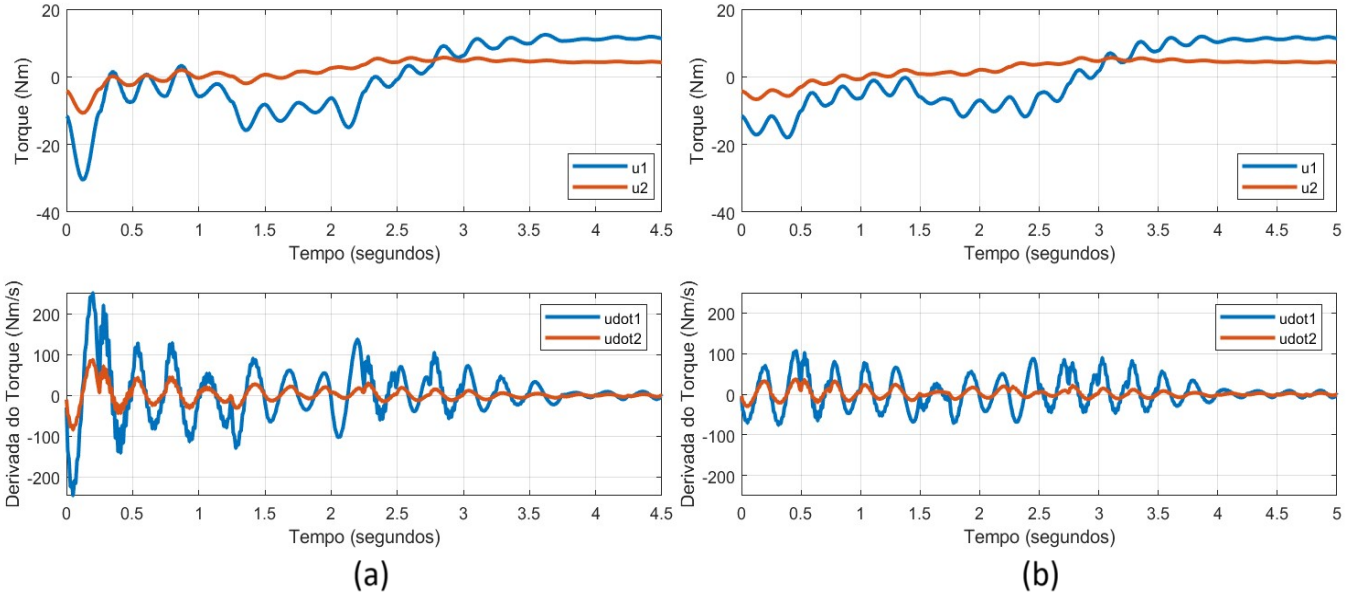
A Figura 21, apresenta os gráficos do torque e da derivada do torque de cada junta (calculados pelo PD), onde é possível observar que a limitação na derivada do torque também reduziu consideravelmente o torque máximo de cada junta, que ultrapassava em valor absoluto os 30 Nm e 10 Nm, respectivamente. Por outro lado, a trajetória restrita levou 5 segundos (meio segundo a mais) para atingir a referência desejada.

Figura 20 – Trajetória 1: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

Figura 21 – Sinal de Controle da Trajetória 1: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque

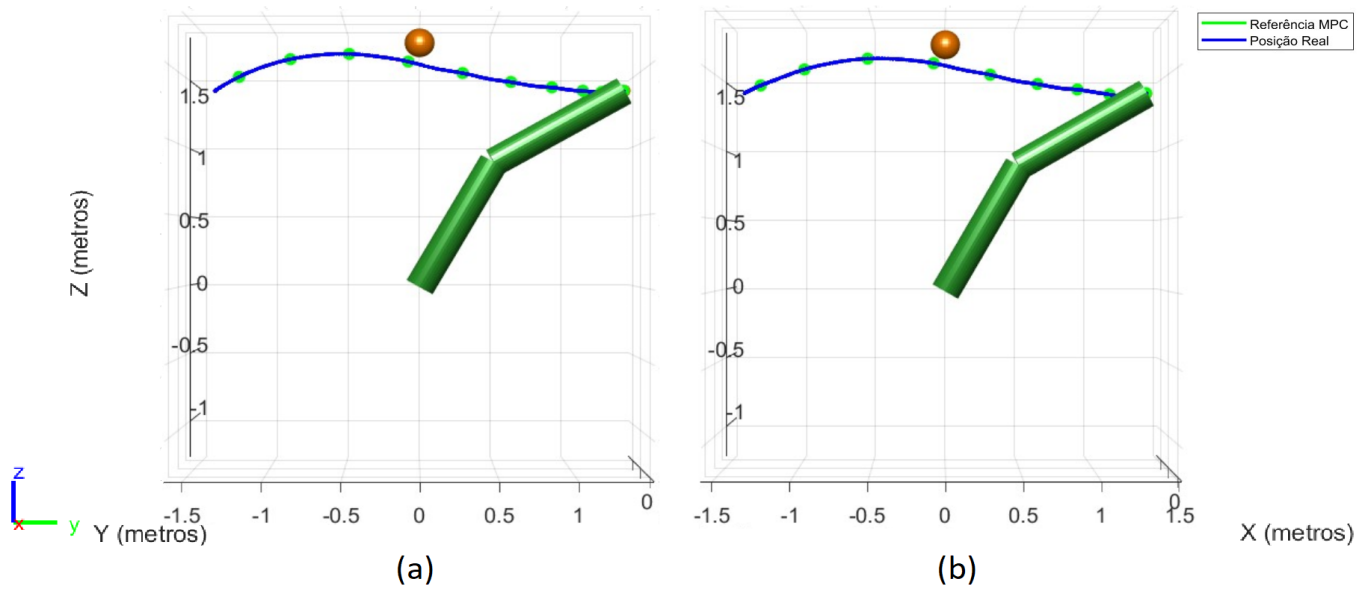


Fonte: Autoria própria (2023).

A segunda trajetória teve origem na mesma posição cartesiana da primeira, mas desta vez no espaço de junta (-150º,30º). Essa configuração tornou o desvio do obstáculo consideravelmente mais fácil, de modo que não houve nenhum desvio brusco na rota do efetuador final, conforme visto na Figura 22. A diferença mais notável entre as trajetórias livre e restrita foi a distância da primeira referência intermediária em relação ao ponto de partida, sendo que a trajetória restrita teve uma arrancada mais lenta. Nos gráficos de torque, Figura 23, o impacto da restrição também foi menor, pois a trajetória original já era naturalmente mais suave que a anterior.

Ainda na segunda trajetória, observou-se uma situação incomum em que a trajetória restrita alcançou o ponto desejado 0,25 segundos mais rápido que a trajetória livre. Isso pode ter ocorrido pois, devido à natureza não linear do problema, o MPC pode ter ficado preso em mínimos locais durante o processo de otimização. Ao impor a restrição da rede neural, o MPC pode ter descartado essa alternativa e, portanto, foi compelido a buscar outra resposta, o que coincidentemente conduziu a otimização restrita a uma melhor solução.

Figura 22 – Trajetória 2: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque

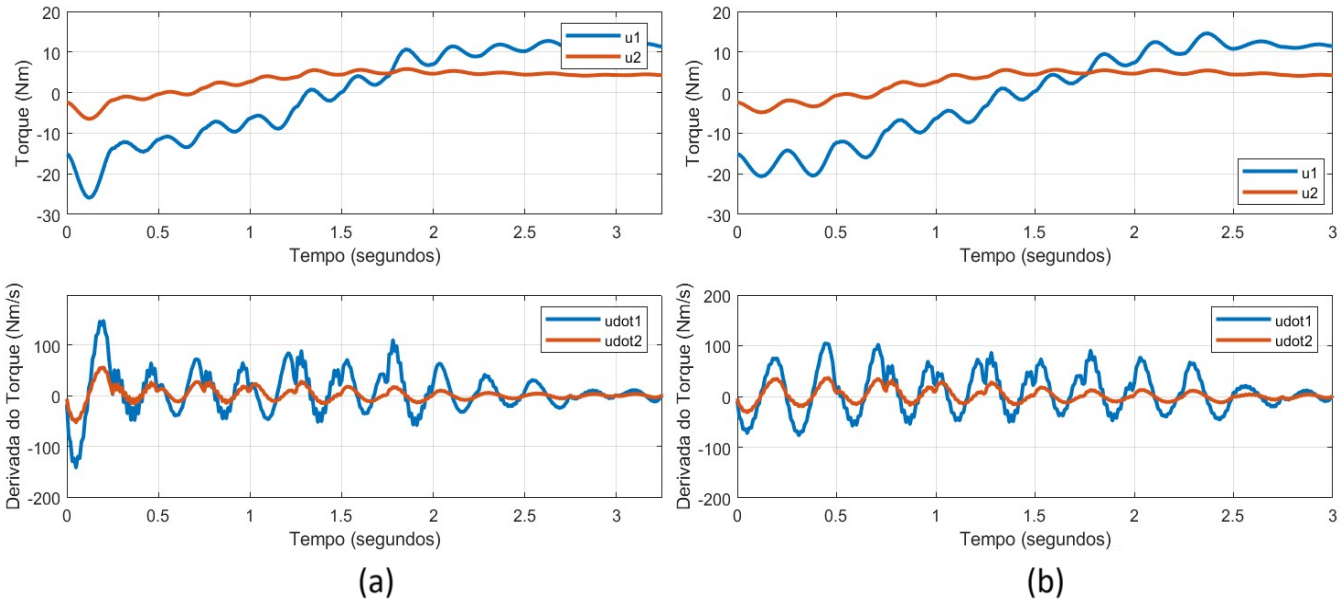


Fonte: Autoria própria (2023).

A terceira trajetória teve origem no espaço de junta (-30º,-30º) e seu objetivo era alcançar o ponto cartesiano (1,366; 1,366). Diferentemente das outras trajetórias, o MPC não conseguiu satisfazer a restrição da derivada do torque com os limites estabelecidos em [± 100, ± 50] Nm/s, sendo necessário aumentá-los para [± 150,± 75] Nm/s. A Figura 24 mostra a trajetória realizada e a Figura 25 apresenta os gráficos de torque.

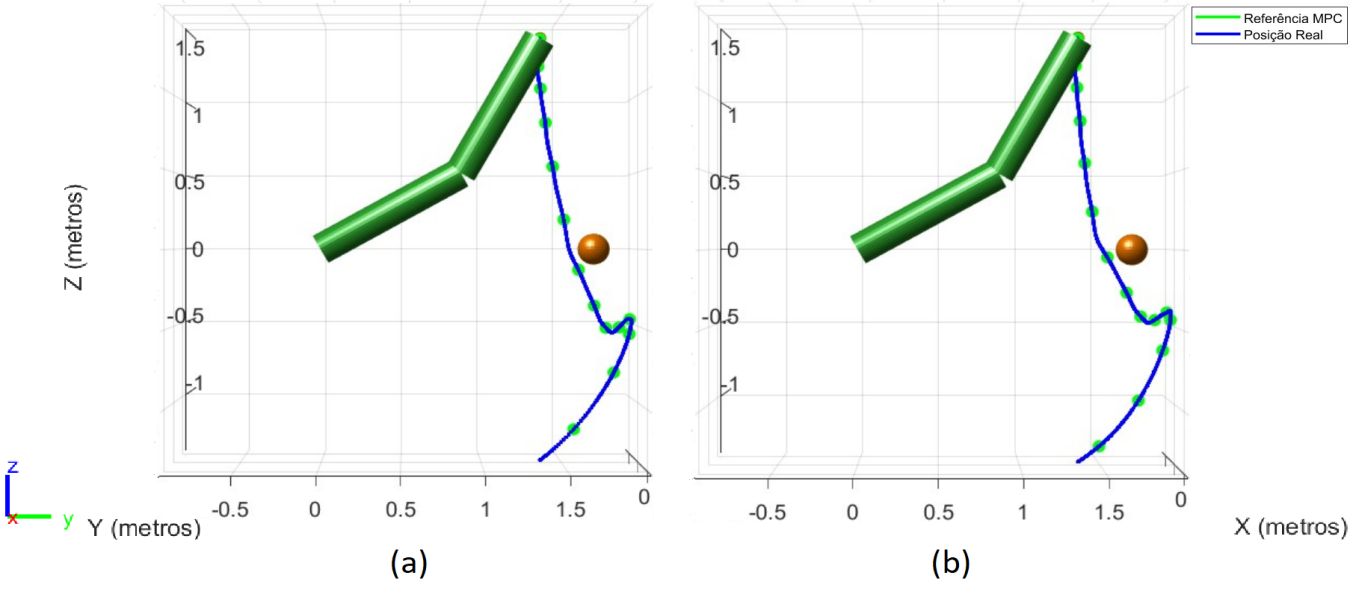
A quarta simulação, apresentada nas Figuras 26 e 27, tem início na posição de junta (-60º,30º). Assim como na segunda simulação, a trajetória restrita também foi 0,25 segundos mais rápida do que a trajetória livre.

Figura 23 – Sinal de Controle da Trajetória 2: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



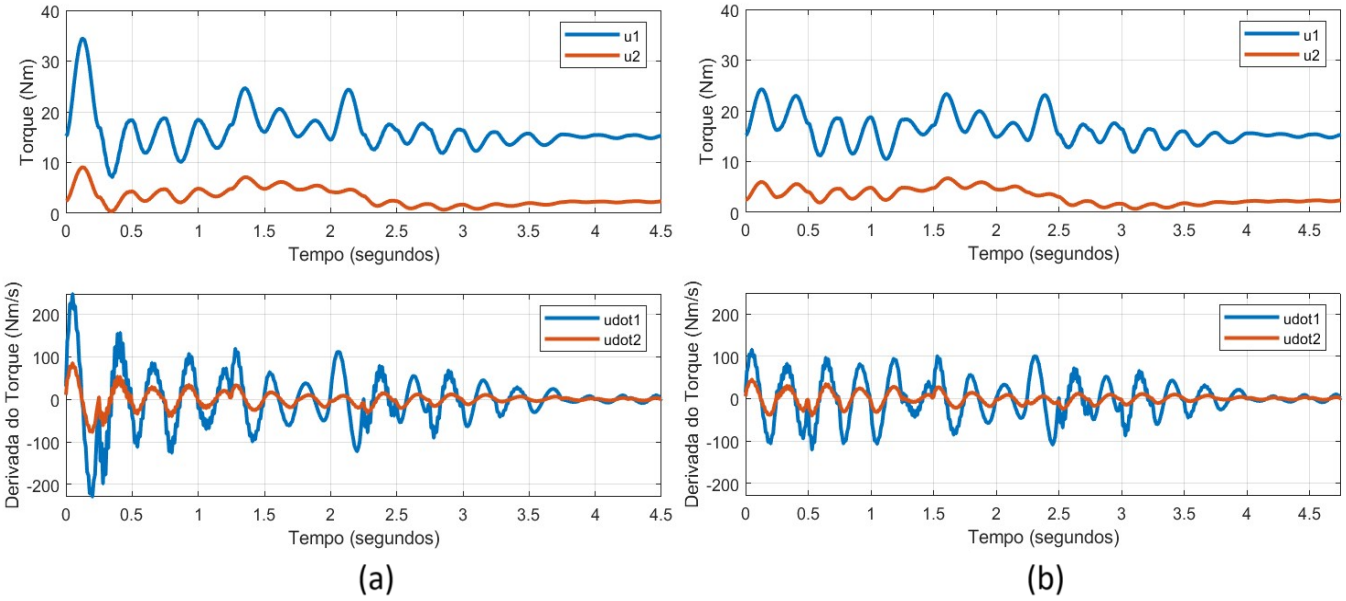
Fonte: Autoria própria (2023).

Figura 24 – Trajetória 3: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



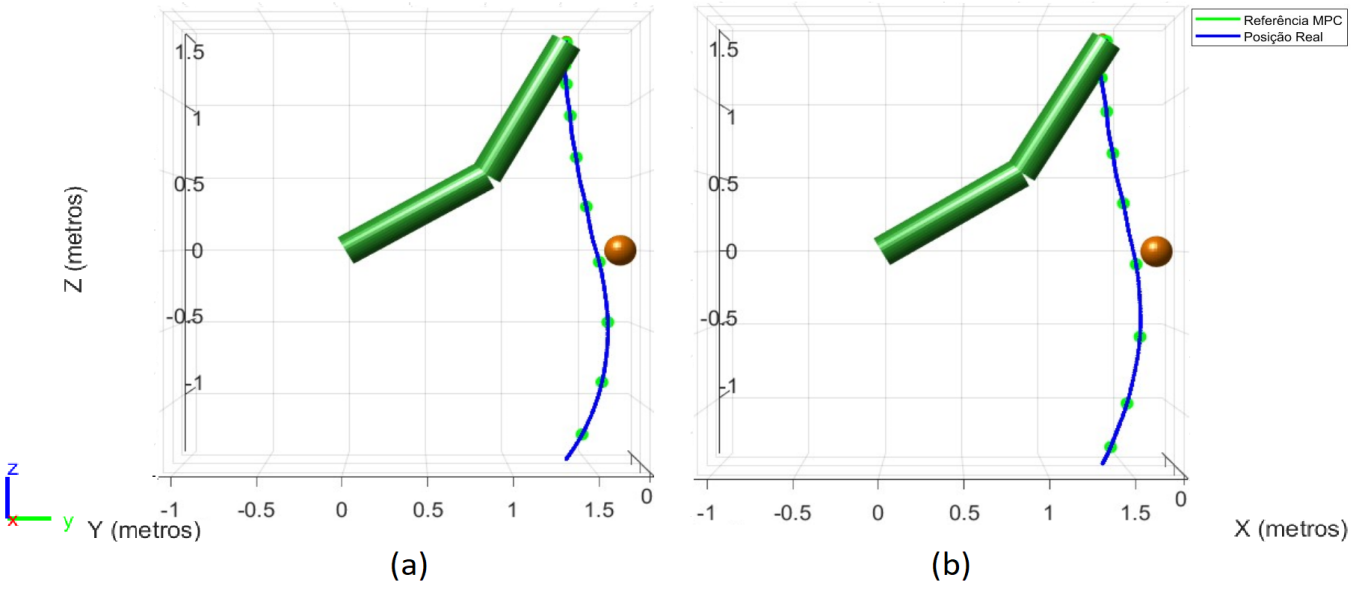
Fonte: Autoria própria (2023).

Figura 25 – Sinal de Controle da Trajetória 3: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



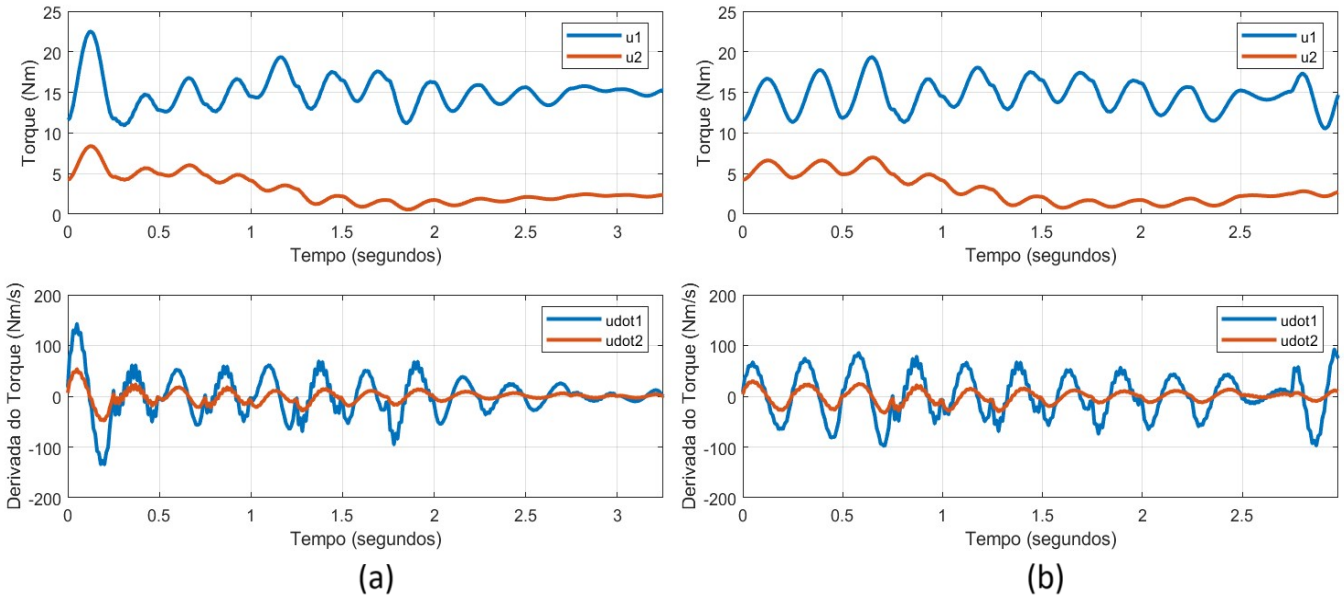
Fonte: Autoria própria (2023).

Figura 26 – Trajetória 4: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

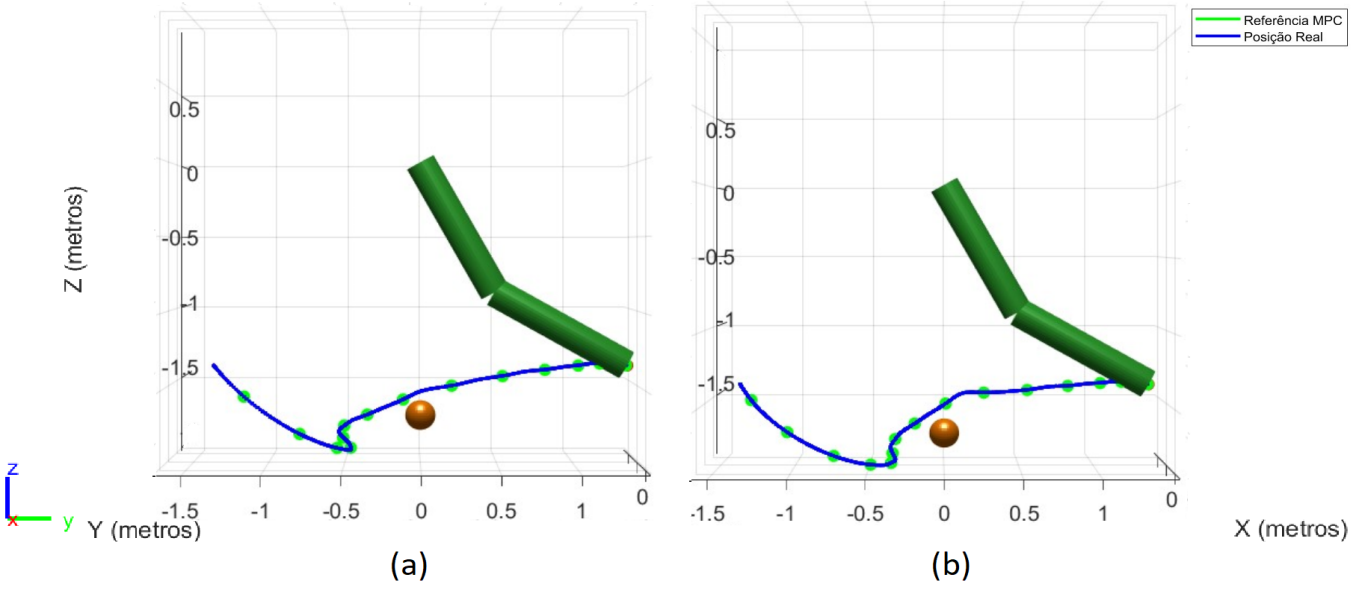
Figura 27 – Sinal de Controle da Trajetória 4: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

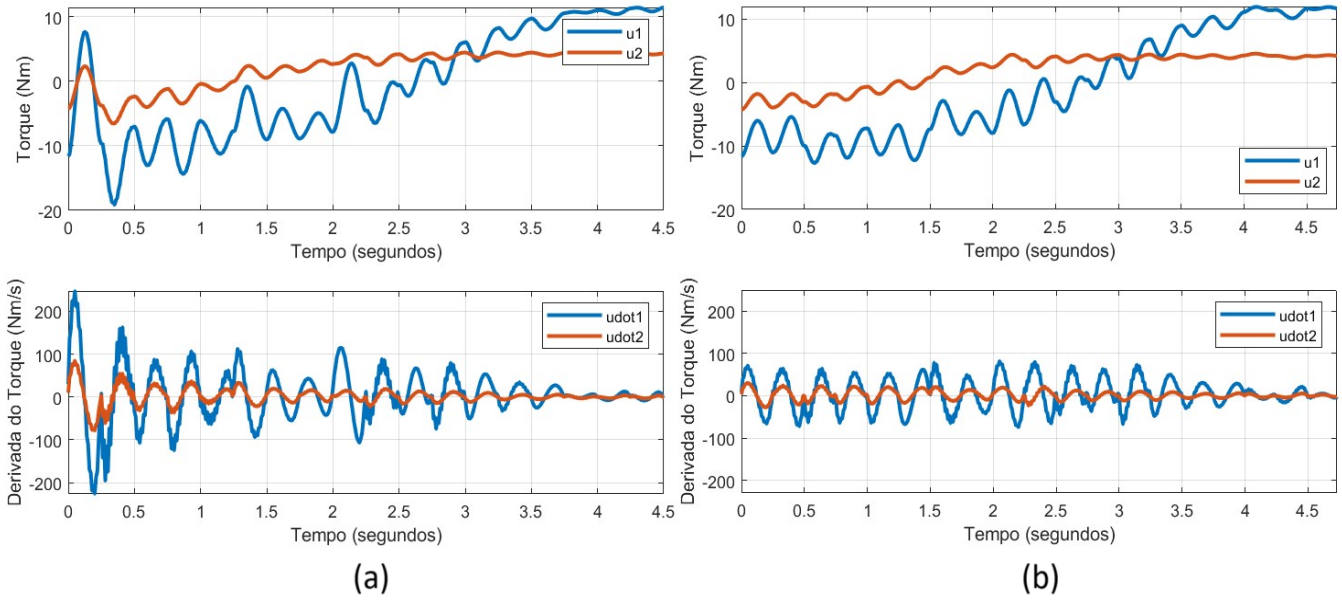
A quinta simulação inicia-se na posição de junta (-120º,-30º) e se destaca pela significativa redução na derivada do torque, que caiu de um pico de 247 Nm/s para 81,95 Nm/s com o uso da rede neural. Os gráficos de posição e torque podem ser observados nas Figuras 28 e 29. Os percursos tomados são similares, com a única diferença notável sendo o número de referências intermediárias e tempo necessários para chegar ao destino final.

Figura 28 – Trajetória 5: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

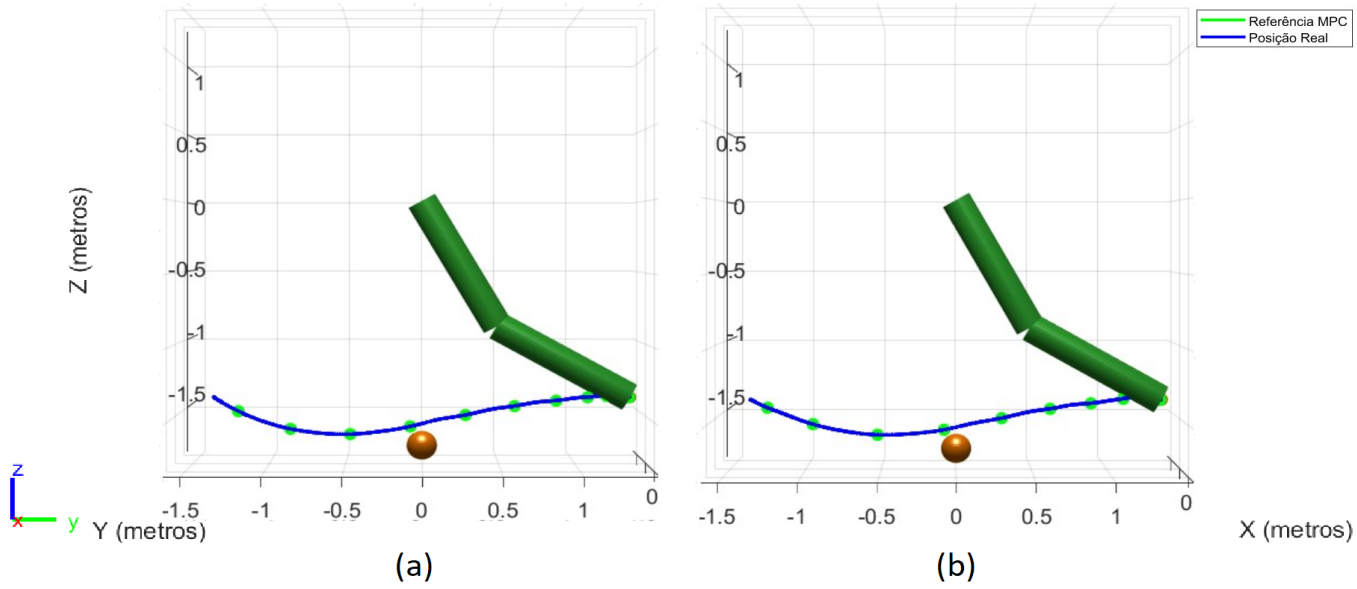
Figura 29 – Sinal de Controle da Trajetória 5: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

A sexta simulação iniciou a partir da posição de junta (-150º,30º). Foi a única simulação em que a restrição neural atrasou a trajetória em 0,75 segundos em relação à trajetória livre. Os picos de torque também não apresentaram uma redução tão significativa quanto nas outras simulações. As Figuras 30 e 31 mostram o percurso e os sinais de controle correspondentes.

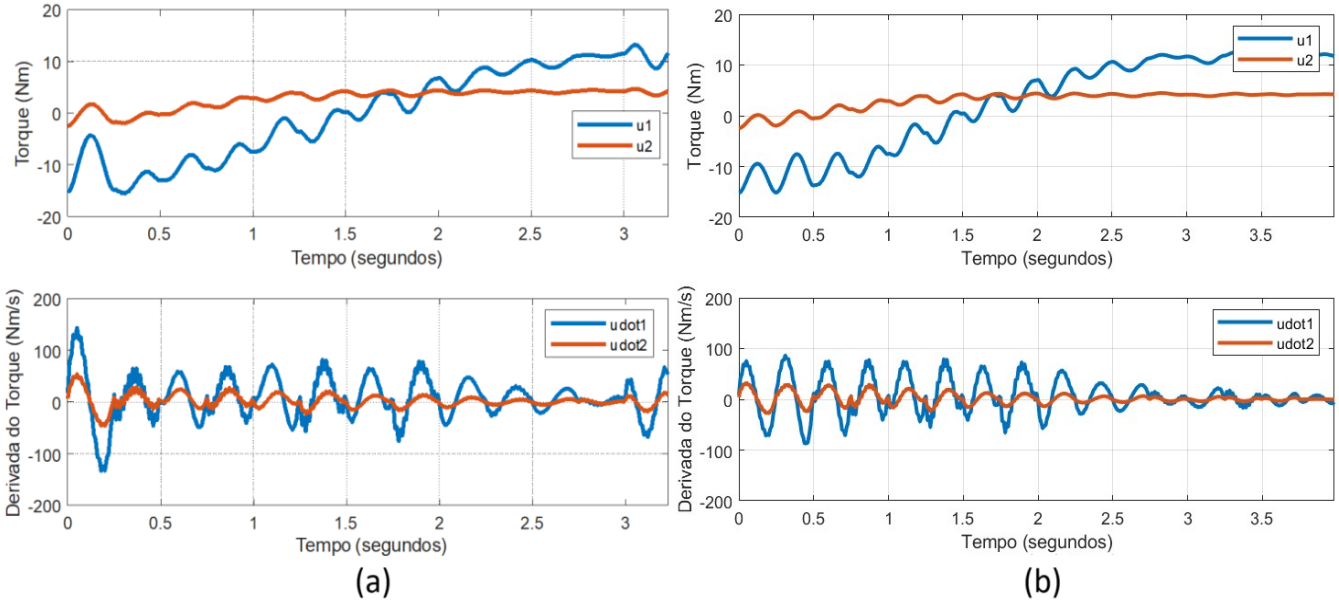
Figura 30 – Trajetória 6: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

A sétima simulação, Figuras 32 e 33, começa da configuração de juntas (30º, 30º) e deve descer para a posição cartesiana (1,366; -1,366). Os resultados da restrição neural foram

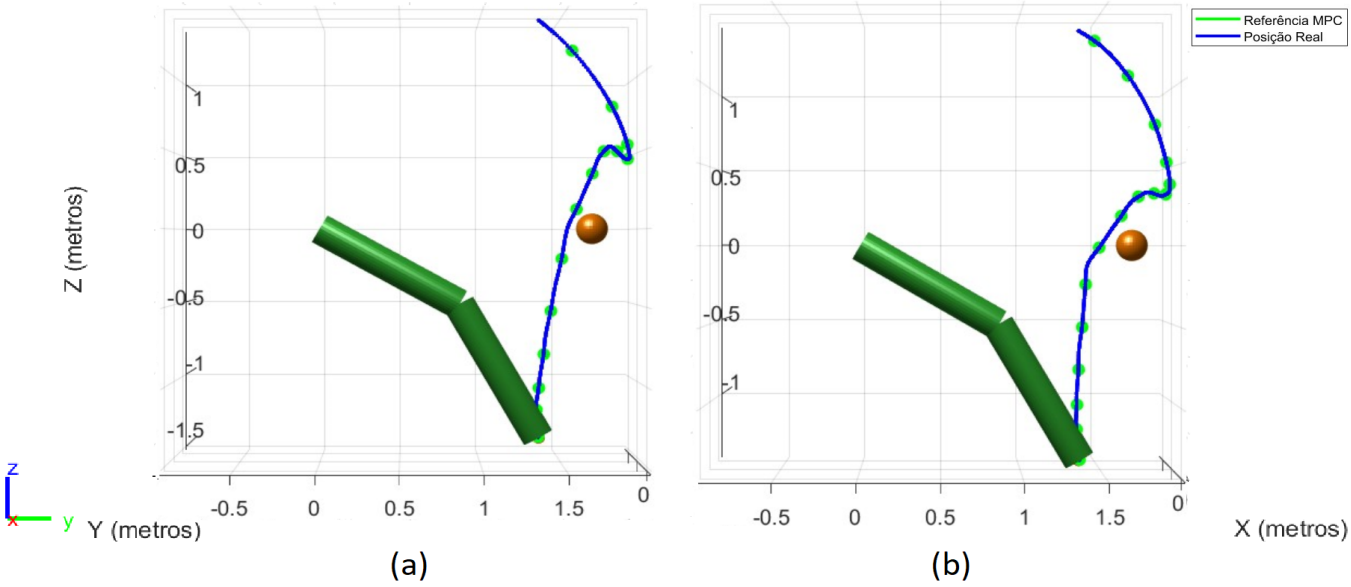
Figura 31 – Sinal de Controle da Trajetória 6: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

similares aos das trajetórias 1, 3 e 5, com uma redução significativa no sinal de controle e uma curva mais suave no desvio do obstáculo.

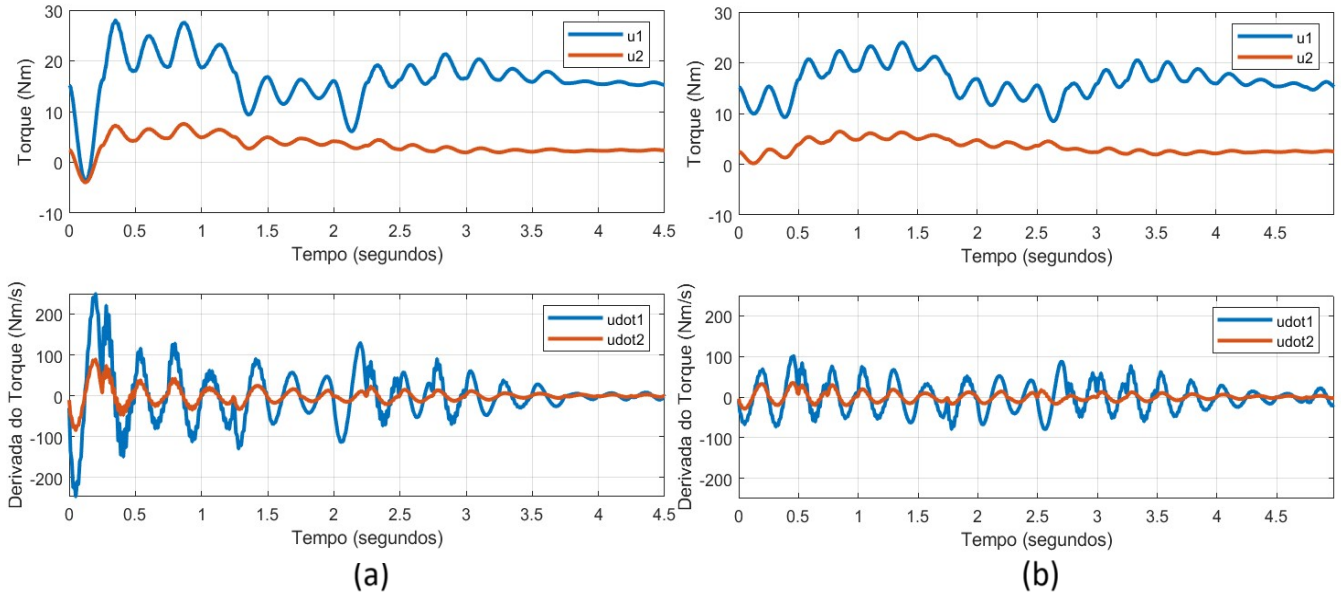
Figura 32 – Trajetória 7: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

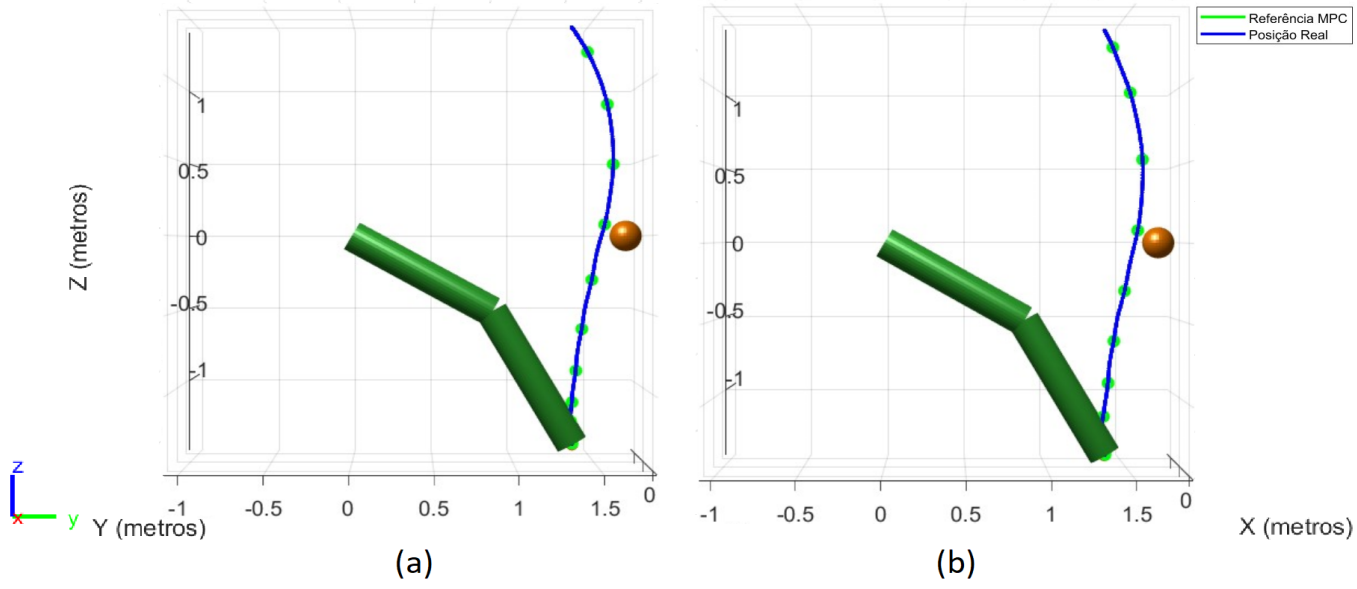
Na oitava e última simulação, que parte da posição (60º,-30º), as diferenças em relação à trajetória 7 são menos notáveis. No entanto, houve um ponto incomum, pois esta foi a única simulação em que registrou-se um aumento no pico de torque da segunda junta (de 5,22 Nm para 5,5 Nm). As Figuras podem ser vistas em 34 e 35.

Figura 33 – Sinal de Controle da Trajetória 7: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



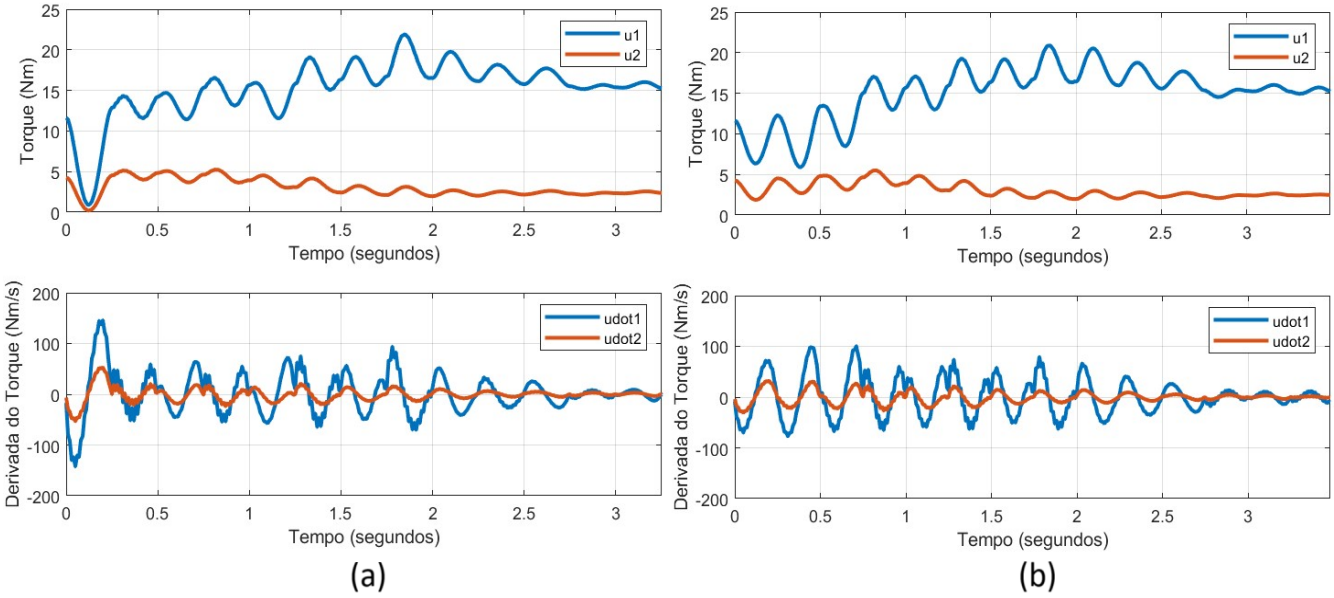
Fonte: Autoria própria (2023).

Figura 34 – Trajetória 8: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

Figura 35 – Sinal de Controle da Trajetória 8: (a) Sem Restrição de Derivada de Torque, (b) Com Restrição de Derivada de Torque



Fonte: Autoria própria (2023).

Para facilitar a comparação do desempenho das trajetórias com e sem restrição neural, uma tabela com todas as características mencionadas está disponível na Tabela 10. As colunas de 2 a 5 apresentam o valor máximo absoluto de cada variável. Em geral, o uso da restrição neural resultou em redução dos picos de torque e suas derivadas na maioria das trajetórias, sendo que as reduções mais significativas ocorreram nas trajetórias 1, 3, 5 e 7.

Tabela 10 – Comparativo das Trajetórias

	u_1 (Nm)	u_2 (Nm)	$\dot{u_1}$ (Nm/s)	$\dot{u_2}$ (Nm/s)	Tempo Total (s)
Trajetória 1	30,4911	10,6952	252,3800	87,8148	4,50
Trajetória 1 Restrita	17,9402	6,6387	107,2952	37,9672	5,00
Trajetória 2	25,9127	6,4689	147,1131	56,3289	3,25
Trajetória 2 Restrita	20,6261	5,6739	104,7275	37,5269	3,00
Trajetória 3	34,4019	9,0443	247,6025	85,1843	4,50
Trajetória 3 Restrita	24,2376	6,8062	121,3604	45,3406	4,75
Trajetória 4	22,4991	8,3724	142,7668	53,4815	3,25
Trajetória 4 Restrita	19,3574	6,9902	98,8874	32,8784	3,00
Trajetória 5	18,8984	6,5563	247,6317	85,0556	4,50
Trajetória 5 Restrita	12,7314	4,5654	81,9537	31,8136	4,75
Trajetória 6	15,5467	4,7079	142,8023	53,5788	3,25
Trajetória 6 Restrita	15,1961	4,4843	88,0742	33,2655	4,00
Trajetória 7	27,6962	7,5647	249,2034	89,1446	4,50
Trajetória 7 Restrita	24,0110	6,4510	101,2640	36,1676	5,00
Trajetória 8	21,6386	5,2237	145,5459	53,5124	3,25
Trajetória 8 Restrita	20,8783	5,5000	100,5641	32,4946	3,50

Fonte: Autoria própria (2023).

Desse modo, os resultados demonstram o efeito positivo da restrição neural em suavizar as trajetórias e reduzir esforços nos atuadores, apesar de pequenos aumentos no tempo de movimento. A redução percentual dos torques e derivadas do torque estão expressos na Tabela 11, em destaque estão as maiores reduções percentuais.

Tabela 11 – Redução Percentual do Torque e da Derivada do Torque

	u_1	u_2	$\dot{u_1}$	$\dot{u_2}$
Redução Traj. 1	**41,16%**	**37,93%**	57,49%	56,76%
Redução Traj. 2	20,40%	12,29%	28,81%	33,38%
Redução Traj. 3	29,55%	24,75%	50,99%	46,77%
Redução Traj. 4	13,96%	16,51%	30,74%	38,52%
Redução Traj. 5	32,63%	30,37%	**66,91%**	**62,60%**
Redução Traj. 6	2,26%	4,75%	38,32%	37,91%
Redução Traj. 7	13,31%	14,72%	59,36%	59,43%
Redução Traj. 8	3,51%	-5,29%	30,91%	39,28%

Fonte: Autoria própria (2023).

4.1.2 *Soft Constraints*

Um dos problemas oriundos das *hard constraints* é a possibilidade de não encontrar-se uma solução factível durante a otimização. Dado o problema que ocorreu com a trajetória 3, as mesmas oito trajetórias foram simuladas novamente, todavia com o uso de *soft constraints* nas restrições de derivada do torque.

As *soft constraints* admitem o uso de uma folga nas restrições. Neste caso, o NLMPC não descarta as soluções que violam as restrições, somente penaliza a função custo de acordo com o valor excedente sob as restrições.

O uso desta técnica permitiu o NLMPC executar a trajetória 3 com a restrição de derivada do torque em ($\pm$100, $\pm$50), onde a trajetória registrou picos de 106,9500 Nm e 30,3082 Nm para cada junta. Os gráficos correspondentes a essas simulações encontram-se no Apêndice A.

Os demais resultados obtidos com uma folga nas restrições não demonstraram um desempenho melhor do que as simulações anteriores, onde o robô levou mais tempo para se deslocar até a referência e na maioria das trajetórias registrou um maior pico na derivada do torque do primeiro motor. A Tabela 12 mostra os novos resultados obtidos.

Todavia, apesar de um desempenho "inferior", a aplicação com *soft constraints* apresenta-se como uma alternativa mais viável em uma aplicação prática, como evitar um travamento na operação devido a impossibilidade do NLMPC respeitar as restrições sem folga.

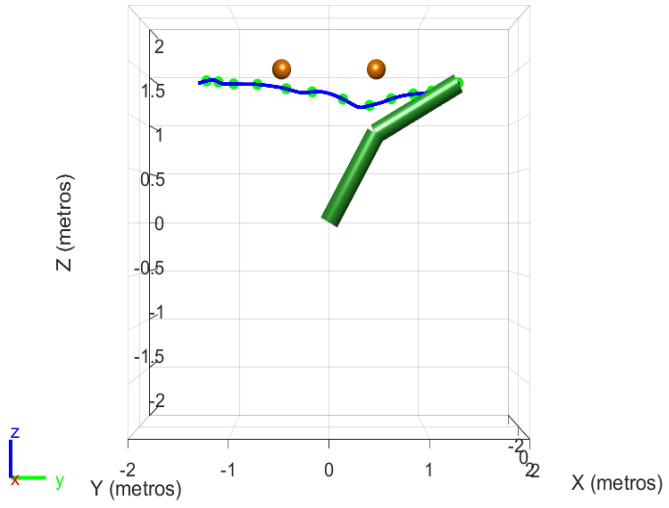
Operações com a presença de mais de um obstáculo também podem ser simuladas, como na Figura 36, onde o NLMPC leva em consideração a distância mínima entre os dois elos do robô à superfície das duas esferas, incorporando todas as distâncias às restrições não lineares.

Tabela 12 – Comparativo das Trajetórias com *Soft Constraints*

	u_1 (Nm)	u_2 (Nm)	$\dot{u_1}$ (Nm/s)	$\dot{u_2}$ (Nm/s)	Tempo Total (s)
Trajetória 1	30,4911	10,6952	252,3800	87,8148	4,50
Trajetória 1 Soft	19,8759	6,7886	110,5734	49,4738	5,25
Trajetória 2	25,9127	6,4689	147,1131	56,3289	3,25
Trajetória 2 Soft	22,6543	7,4546	114,4645	43,3108	4,00
Trajetória 3	34,4019	9,0443	247,6025	85,1843	4,50
Trajetória 3 Soft	23,0986	6,1680	106,9500	30,3082	5,00
Trajetória 4	22,4991	8,3724	142,7668	53,4815	3,25
Trajetória 4 Soft	18,9029	6,9919	100,0841	40,5338	4,00
Trajetória 5	18,8984	6,5563	247,6317	85,0556	4,50
Trajetória 5 Soft	13,5720	4,4774	109,7735	33,0601	5,25
Trajetória 6	15,5467	4,7079	142,8023	53,5788	3,25
Trajetória 6 Soft	15,2446	4,4973	105,0668	35,1615	4,00
Trajetória 7	27,6962	7,5647	249,2034	89,1446	4,50
Trajetória 7 Soft	24,6070	6,8835	110,5785	37,7292	5,25
Trajetória 8	21,6386	5,2237	145,5459	53,5124	3,25
Trajetória 8 Soft	20,4693	5,7145	109,2813	43,1763	3,75

Fonte: Autoria própria (2023).

Figura 36 – Trajetória do Manipulador com Dois Obstáculos



Fonte: Autoria própria (2023).

Em suma, os resultados da seção indicam que a abordagem proposta permitiu suavizar as trajetórias executadas na maioria dos casos analisados. Isso ficou evidenciado pela redução nos picos da derivada do torque observada na grande parte das simulações. Além disso, nota-se uma diminuição nos valores absolutos máximos dos torques em diversos testes, o que é benéfico por implicar em menor esforço nos atuadores. O tempo de movimento aumentou ligeiramente com as restrições devido ao planejamento mais conservador, porém esse *trade-off* parece aceitável em aplicações onde necessita-se de limitações na ação dos atuadores. A redução percentual no sinal de controle com o uso de *Soft Constraints* pode ser verificados na Tabela 13, onde a segunda trajetória neste caso também registrou um aumento no pico de torque da segunda junta.

Tabela 13 – Redução Percentual do Torque e da Derivada do Torque com *Soft Constraints*

	u_1	u_2	$\dot{u}_1$	$\dot{u}_2$
Redução Traj. 1	**34,81%**	**36,53%**	56,19%	43,66%
Redução Traj. 2	12,57%	-15,24%	22,19%	23,11%
Redução Traj. 3	32,86%	31,80%	**56,81%**	**64,42%**
Redução Traj. 4	15,98%	16,49%	29,90%	24,21%
Redução Traj. 5	28,18%	31,71%	55,67%	61,13%
Redução Traj. 6	1,94%	4,47%	26,42%	34,37%
Redução Traj. 7	11,15%	9,00%	55,63%	57,68%
Redução Traj. 8	5,40%	-9,40%	24,92%	19,32%

Fonte: Autoria própria (2023).

4.2 Resultados do Drone

As trajetórias escolhidas para o drone quadricóptero dividiram-se em: movimento no eixo X, movimento no eixo Y, movimento no eixo Z (sentido positivo e negativo), e movimento nos três eixos simultaneamente. O eixo Z foi testado em ambos os sentidos, pois seria possível que o sistema apresentasse diferença ao se mover no sentido favorável ou contrário a força gravitacional.

Em todas as simulações o drone quadricóptero parte da origem (0,0,0) e tem como destino as seguintes coordenadas:

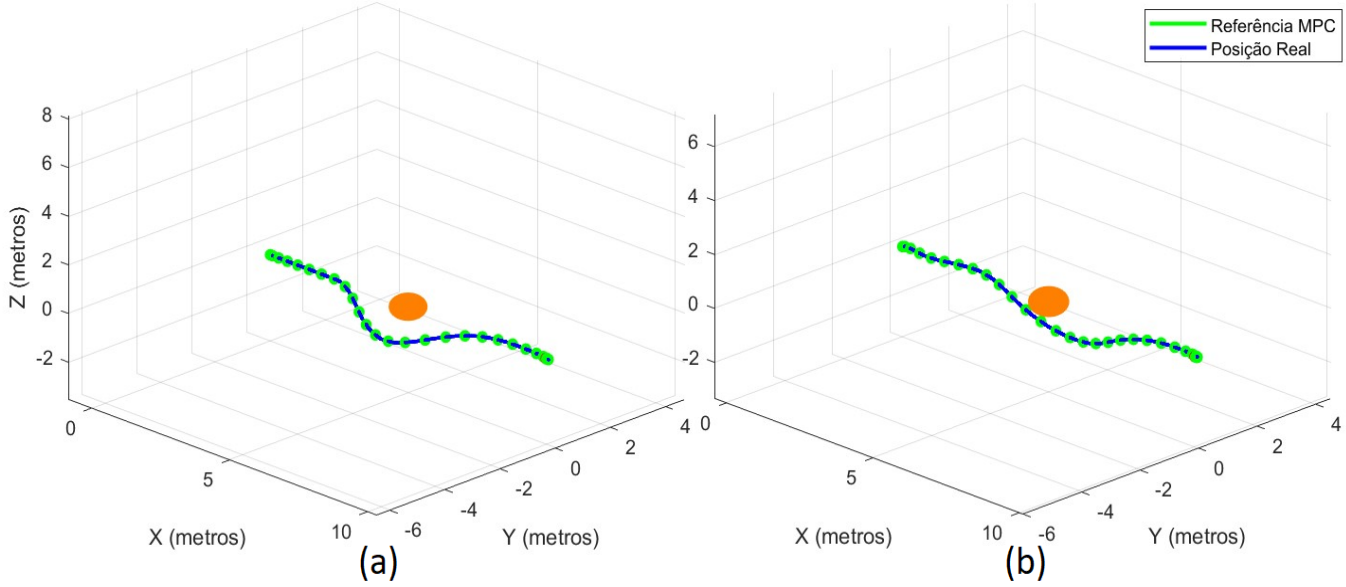
- Trajetória em X: (10,0,0)
- Trajetória em Y: (0,10,0)
- Trajetória em Z: (0,0,10)
- Trajetória em -Z: (0,0,-10)
- Trajetória em XYZ: (10,10,10)

onde consta um obstáculo esférico posicionado no meio do percurso.

A forma das restrições utilizadas foram de *soft constraints*, pois o NLMPC teve muitas dificuldades de encontrar soluções factíveis com *hard constraints* no sinal de controle, onde restringiu-se o sinal em $(\pm 0{,}4, \pm 0{,}1, \pm 0{,}15)$.

O movimento no eixo X, ilustrado pela Figura 37, apresenta um percurso distinto entre a trajetória livre e a com restrição com folgas. O caminho percorrido pelo drone é notado por uma curva mais aberta ao desviar do obstáculo. Apesar do percurso visualmente parecer maior, o tempo gasto em ambos os percursos foram respectivamente 30 e 31 segundos, o que implica num atraso relativamente pequeno.

Figura 37 – Trajetória do Drone em X: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque



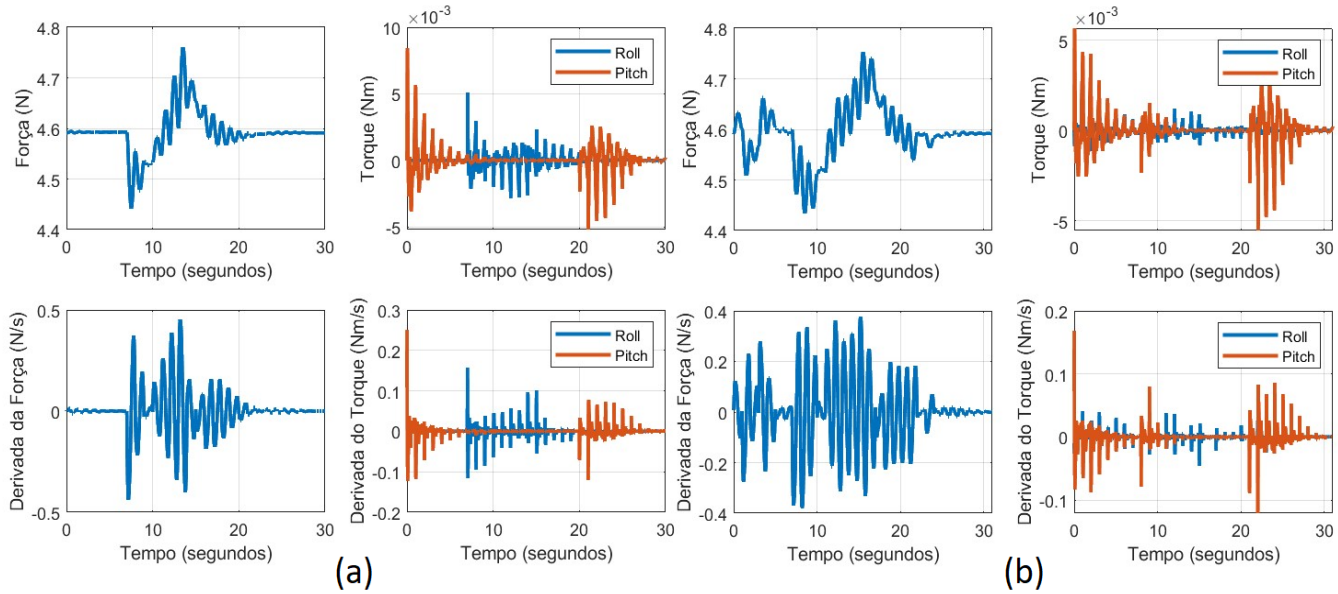
Fonte: Autoria própria (2023).

Mesmo com o uso de *soft constraints* que permitem uma folga nas restrições, o NLMPC teve dificuldades em calcular as trajetórias com restrições de derivada de força e torque muito baixas, e em muitos casos não conseguindo desviar do obstáculo.

A Figura 38, apresenta quatro gráficos para a trajetória livre e quatro para a trajetória restrita: *thrust* (N), Derivada do Thrust (N/s), torques de roll e pitch, e suas derivadas. Nesta figura, pode-se ver no gráfico de derivada da força (a) que o pico é de 0,5 N/s, enquanto no de (b) como 0,4 N/s, sendo uma redução relativamente pequena comparada às reduções efetuadas na seção do robô manipulador.

Desse modo, somente pequenas reduções na derivada do sinal de controle foram possíveis de se efetuar. Ainda assim, percebe-se de que os gráficos a direita (b), apesar de terem seus picos absolutos menores, apresentam mais picos do que as trajetórias livres (a), o que implica em resultados não tão satisfatórios em contraste com a seção anterior.

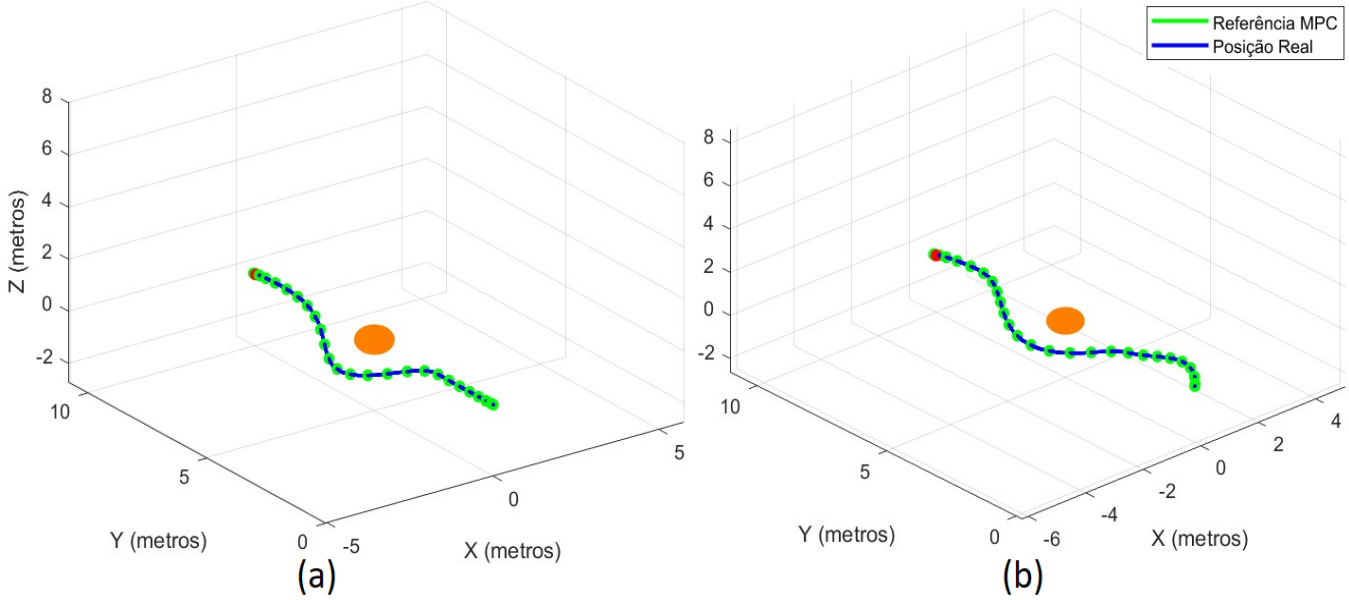
Figura 38 – Sinal de Controle do Drone em X: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque



Fonte: Autoria própria (2023).

O movimento em Y apresenta um comportamento similar ao em X. Como há pouca variação de altitude em ambos os casos, estes apresentam um comportamento similar, como visto na Figura 39.

Figura 39 – Trajetória do Drone em Y: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque

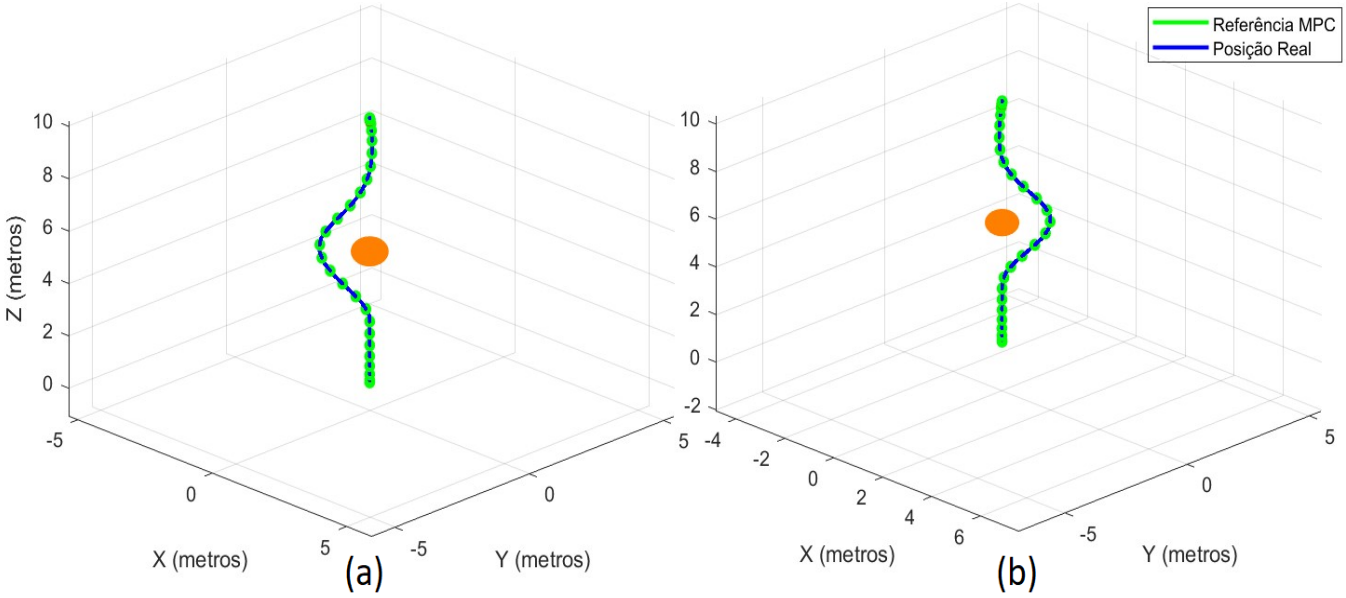


Fonte: Autoria própria (2023).

As Figuras 40 e 41 tem um enfoque maior no controle de altitude, sendo que a primeira atua no sentido contrário a força gravitacional e a segunda no sentido favorável. A diferença mais notável entre as trajetórias restritas e livres para ambos os casos é de que o NLMPC

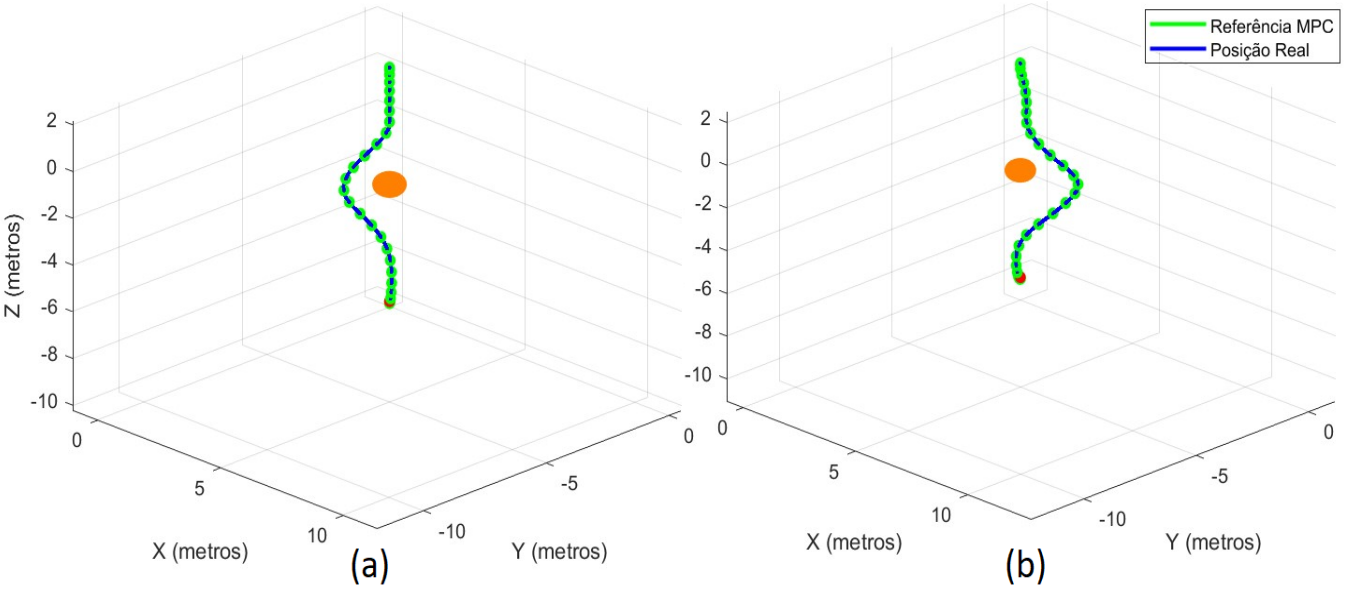
acabou desviando para lados distintos partindo da mesma configuração inicial. A redução na derivada do torque foi maior no ângulo de *pitch* em comparação com as trajetórias em X e Y que obtiveram maior redução no ângulo de *roll*.

Figura 40 – Trajetória do Drone em Z: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque



Fonte: Autoria própria (2023).

Figura 41 – Trajetória do Drone em -Z: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque

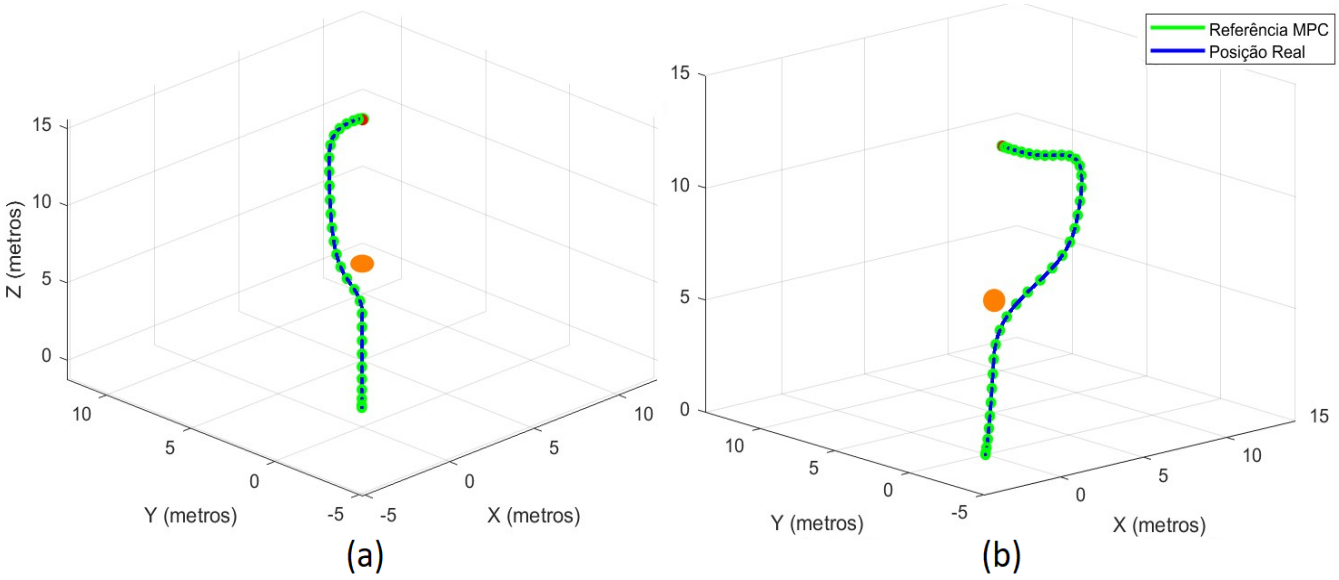


Fonte: Autoria própria (2023).

Por último, foi aplicada uma trajetória com movimento simultâneo nos três eixos, sendo uma distância maior, o tempo de execução também aumentou. A Figura 42, apresenta a maior

distinção de comportamento entre o percurso livre e o restrito, onde a trajetória restrita chegou a altitude desejada muito mais rapidamente do que a sua coordenada XY. Isso resultou em um atraso de 35 para 42 segundos.

Figura 42 – Trajetória do Drone em XYZ: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque



Fonte: Autoria própria (2023).

Por apresentarem os mesmos problemas discutidos na Figura 38, os gráficos do sinal de controle das demais trajetórias constam no Apêndice B. Todavia, uma tabela comparando os picos de cada variável manipulada e suas derivadas constam na Tabela 10, onde mostra-se uma redução relativamente menor na maioria dos valores, quando comparada com as Tabelas 10 e 12.

Tabela 14 – Comparativo das Trajetórias dos Drones

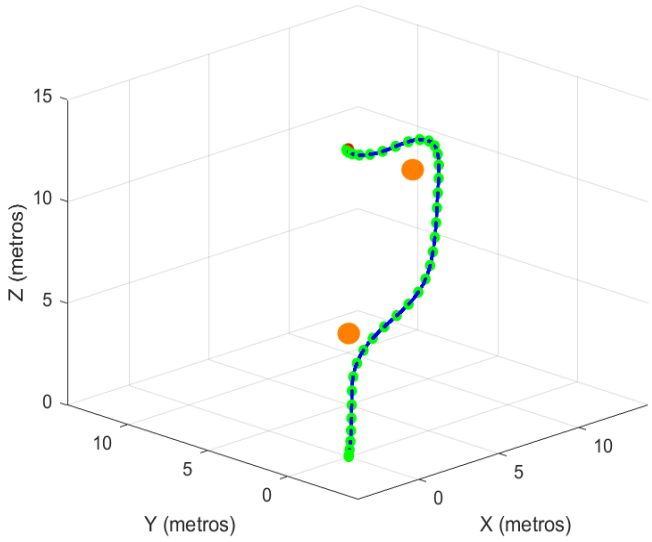
	T (N)	τ_ϕ (Nm)	τ_θ (Nm)	$\dot{T}$ (N/s)	$\dot{\tau_\phi}$ (Nm/s)	$\dot{\tau_\theta}$ (Nm/s)	Tempo (s)
Traj. X	4,7609	0,0051	0,0084	0,4501	0,1565	0,2501	30,00
Traj. X Soft	4,7521	0,0012	0,0057	0,3787	0,0463	0,1689	31,00
Traj. Y	4,7614	0,0051	0,0063	0,4625	0,1564	0,1862	30,00
Traj. Y Soft	4,7511	0,0030	0,0072	0,3879	0,0856	0,1493	31,00
Traj. Z	4,7597	0,0042	0,0069	0,4906	0,1293	0,2067	30,00
Traj. Z Soft	4,7478	0,0034	0,0059	0,3943	0,1230	0,1559	31,00
Traj. -Z	4,6465	0,0044	0,0073	0,4906	0,1361	0,2177	30,00
Traj. -Z Soft	4,6490	0,0042	0,0073	0,3948	0,1319	0,1528	31,00
Traj. XYZ	4,7657	0,0051	0,0112	0,5024	0,1562	0,3125	35,00
Traj. XYZ Soft	4,7522	0,0042	0,0059	0,3840	0,1324	0,1563	42,00

Fonte: Autoria própria (2023).

Assim como o robô manipulador, pode-se incorporar múltiplos obstáculos ao drone, todavia este demonstrou maior dificuldade de desviar dos obstáculos. A Figura 43 mostra uma

trajetória com dois obstáculos, onde o NLMPC necessitou realizar grandes desvios de sua rota principal para atender as restrições do sinal de controle.

Figura 43 – Trajetória do Drone com Dois Obstáculos



Fonte: Autoria própria (2023).

Portanto, os resultados do uso da abordagem proposta em drones quadricópteros não foram tão satisfatórios quanto para robôs manipuladores. As reduções na derivada do sinal de controle alcançadas foram pequenas na maioria dos casos. Além disso, observou-se a presença de mais picos nas trajetórias restritas. A redução percentual dos picos de força, torque e suas derivadas constam na Tabela 15.

Tabela 15 – Redução Percentual da Derivada do Sinal de Controle nos Drones

	T	τ_ϕ	τ_θ	$\dot{T}$	$\dot{\tau_\phi}$	$\dot{\tau_\theta}$
Redução Traj. X	0,18%	76,47%	32,14%	15,86%	70,42%	32,47%
Redução Traj. Y	0,22%	41,18%	-14,29%	16,13%	45,27%	19,82%
Redução Traj. Z	0,25%	19,05%	14,49%	19,63%	4,87%	24,58%
Redução Traj. -Z	-0,05%	4,55%	0,00%	19,53%	3,09%	29,81%
Redução Traj. XYZ	0,28%	17,65%	47,32%	23,57%	15,24%	49,98%

Fonte: Autoria própria (2023).

Uma hipótese para explicar esse desempenho inferior é a maior dependência entre as variáveis de estado no drone, no caso o "acoplamento" entre a orientação e o movimento na direção do eixo Z móvel. Já no robô manipulador, a independência cinemática entre as juntas permite mais flexibilidade na busca por soluções suaves.

Dessa forma, os resultados parecem indicar que a abordagem proposta se mostrou mais adequada para sistemas com variáveis de estado "desacopladas" do que para sistemas com maior dependência entre seus estados, como drones quadricópteros e sistemas móveis.

5 CONCLUSÃO

5.1 Considerações Finais

Este trabalho teve como objetivo desenvolver um sistema de controle preditivo não linear (NLMPC) em cascata com restrições no sinal de controle da malha secundária, calculadas por redes neurais artificiais. A técnica foi aplicada na geração de trajetórias de um robô manipulador planar de dois graus de liberdade e um drone quadricóptero.

O referencial teórico abordou os conceitos de modelagem e controle de robôs manipuladores e drones, incluindo técnicas como torque computado e controle PD, controle MPC e NLMPC.

Os resultados mostraram que a abordagem proposta se mostrou eficaz na maioria dos casos simulados para reduzir picos de torque e derivada de torque do robô manipulador, resultando em movimentos mais suaves durante desvios de obstáculos. Isso evidencia o potencial da técnica para sistemas robóticos com variáveis de estado "cinematicamente desacopladas".

Por outro lado, a aplicação em drones quadricópteros apresentou desempenho menos satisfatório, com pouca redução na derivada do sinal de controle. Uma hipótese para esse resultado é a maior dependência entre os estados de orientação e posição do drone, diferentemente da independência entre as juntas do manipulador.

O trabalho apresentou algumas dificuldades na implementação de *hard constraints* no NLMPC, o qual obteve problemas de infactibilidade durante a otimização (principalmente com o drone quadricóptero). Essas questões foram contornadas com o uso de *soft constraints*.

Como contribuições, este trabalho propôs uma integração original entre NLMPC e redes neurais para controle em cascata, além de avaliar o desempenho dessa abordagem em dois tipos distintos de sistemas robóticos simulados. O método desenvolvido tem potencial para aumentar a segurança e suavidade de operação ao restringir indiretamente o esforço dos atuadores.

Em aplicações em tempo real da arquitetura proposta, os custos computacionais precisam ser levados em conta, especialmente o tempo de otimização do NLMPC via SQP. Este tenderá a aumentar em sistemas com maior número de graus de liberdade devido à complexidade do problema de otimização.

5.2 Sugestões para Trabalhos Futuros

Para trabalhos futuros seria interessante explorar a aplicação da abordagem em cenários mais complexos e em robôs com mais graus de liberdade, o que aumentaria o espaço de estados e tornaria a geração e coleta de dados para o treinamento das redes neurais mais custoso. Dependendo da complexidade do problema, redes mais sofisticadas do que redes MLP

feedforward poderiam ser adotadas, e outras formas de restrições como *chance constraints* poderiam ser exploradas.

Da mesma forma, a arquitetura poderia ser testada em robôs móveis terrestres e veículos autônomos, os quais possivelmente apresentariam um resultado mais satisfatório do que os drones por operarem com apenas duas coordenadas de posição e uma de orientação.

Melhorias no custo computacional relativo ao uso do otimizador seriam interessantes por possibilitar uma diminuição no tempo de amostragem do NLMPC, o que possivelmente facilitaria os desvios mais complexos, como múltiplos obstáculos em movimento.

O atual trabalho depende do uso da *toolbox* de MPC do MATLAB, dessa forma uma possível implementação do NLMPC e da otimização via SQP em outros softwares e linguagens de programação tornariam a aplicação da arquitetura proposta mais viável em outros projetos sem a necessidade de depender exclusivamente do MATLAB.

Explorar o uso de softwares com modelos e ambientes mais realistas para a simulação dos sistemas, como o CoppeliaSim ou o Webots.

Por fim, a implementação da técnica proposta em sistemas físicos colocaria o controlador diante de atrasos e incertezas não modeladas na simulação, isso permitiria uma melhor validação dos resultados obtidos.

REFERÊNCIAS

BURDEN, F.; WINKLER, D. Bayesian regularization of neural networks. **Artificial neural networks: methods and applications**, Springer, p. 23–42, 2009.

CAMACHO, E. F.; BORDONS, C. **Model Predictive Control**. London: Springer, 2007.

CHENG, H.; YANG, Y. Model predictive control and PID for path following of an unmanned quadrotor helicopter. *In*: **12th IEEE conference on industrial electronics and applications (ICIEA)**. [*S.l.*]: IEEE, 2017. p. 768–773.

CHIU, J.-R. *et al.* A collision-free MPC for whole-body dynamic locomotion and manipulation. *In*: **2022 International Conference on Robotics and Automation (ICRA)**. [*S.l.*]: IEEE, 2022.

CRAIG, J. J. **Introduction To Robotics**: Mechanics and control. New Jersey: Pearson Education, 2009.

DING, J. *et al.* Robust Locomotion Exploiting Multiple Balance Strategies: An observer-based cascaded model predictive control approach. **IEEE Transactions on Mechatronics**, v. 27, n. 4, p. 2089–2097, jun. 2022.

ECKHOFF, M. *et al.* An MPC Framework For Planning Safe Trustworthy Robot Motions. *In*: **2022 International Conference on Robotics and Automation (ICRA)**. [*S.l.*]: IEEE, 2022.

ELSISI, M. *et al.* Effective nonlinear model predictive control scheme tuned by improved NN for robotic manipulators. **IEEE Access**, v. 9, p. 64278–64290, abr. 2021.

INCREMONA, G. P.; FERRARA, A.; MAGNI, L. MPC for robot manipulators with integral sliding modes generation. **IEEE Transactions on Mechatronics**, v. 22, n. 3, p. 1299–1307, jun. 2017.

JIANG, B. *et al.* Neural network based model predictive control for a quadrotor UAV. **Aerospace**, v. 9, n. 8, p. 460, 2022.

KHOSLA, P. K.; KANADE, T. Real-time implementation and evaluation of computed-torque scheme. **IEEE Transactions on Robotics and Automation**, v. 5, n. 2, p. 245–253, 1989.

KLEIN, C. A.; HUANG, C.-H. Review of pseudoinverse control for use with kinematically redundant manipulators. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-13, n. 2, p. 245–250, 1983.

KRÄMER, M. *et al.* Model predictive control of a collaborative manipulator considering dynamic obstacles. **Optimal Control Applications and Methods**, v. 41, n. 4, p. 1211–1232, 2020.

KRUZIC, S. *et al.* End-effector force and joint torque estimation of a 7-DoF robotic manipulator using deep learning. **Eletronics**, v. 10, n. 23, p. 2963, 2021.

LEMOS, N. **Mecânica Analítica**. São Paulo: Editora Livraria da Física, 2007.

LIU, C. *et al.* Path planning for autonomous vehicles using model predictive control. *In*: **2017 IEEE Intelligent Vehicles Symposium (IV)**. [*S.l.*]: IEEE, 2017.

LUUKKONEN, T. Modelling and control of quadcopter. **Independent research project in applied mathematics**, v. 22, n. 22, 2011.

NUBERT, J. *et al.* Safe and fast tracking on a robot manipulator: Robust mpc and neural network control. **IEEE Robotics and Automation Letters**, v. 5, n. 2, p. 3050–3057, abr. 2020.

OGATA, K. **Modern Control Engineering**. Upper Saddle River: Prentice Hall, 2010.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow**. Birmingham: Packt Publishing Ltd, 2017.

SANTOS, M. A.; FERRAMOSCA, A.; RAFFO, G. V. Tube-based MPC with Nonlinear Control for Load Transportation using a UAV. **IFAC-PapersOnLine**, v. 51, n. 25, p. 459–465, 2018.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. **Robot modeling and control**. New York: Wiley, 2006.

VARGA, B. *et al.* Model predictive control and trajectory optimization of large vehicle-manipulators. *In*: **2019 IEEE International Conference on Mechatronics (ICM)**. [*S.l.*]: IEEE, 2019.

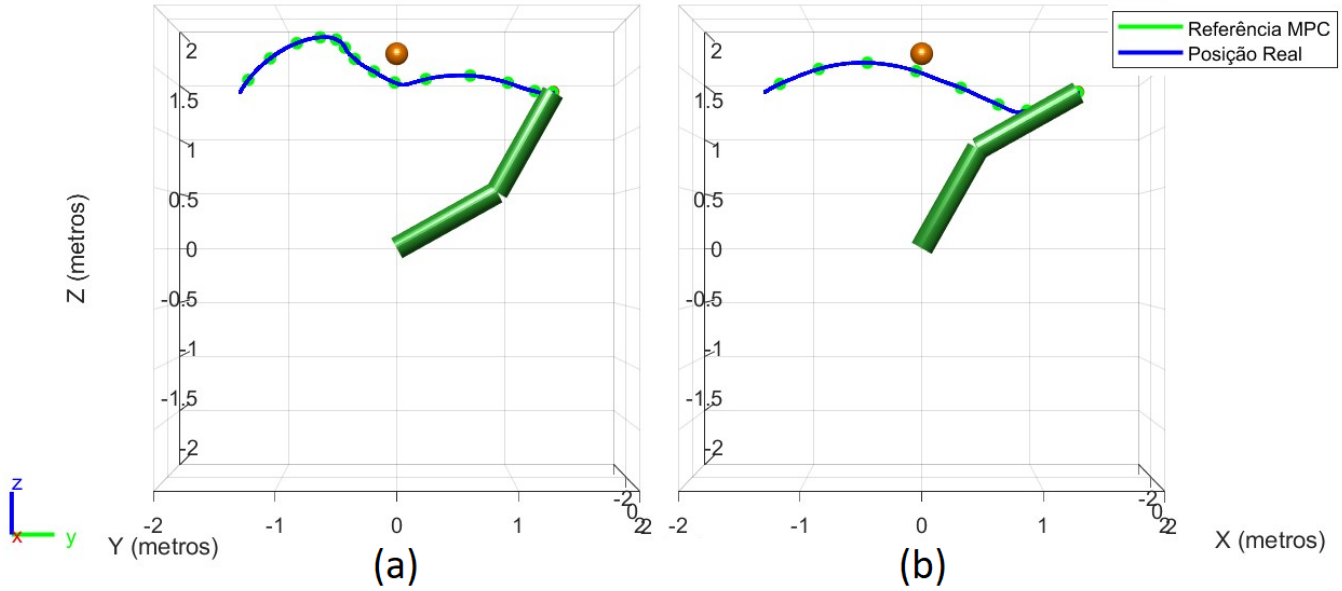
WANG, L. **Model Predictive Control System Design and Implementation Using MATLAB**. London: Springer, 2009.

YU, H.; WILAMOWSKI, B. M. Levenberg-marquardt training. **Industrial electronics handbook**, CRC Press Boca Raton, FL, USA, v. 5, n. 12, p. 1, 2011.

ZUO, Z. Trajectory tracking control design with command-filtered compensation for a quadrotor. **IET control theory applications**, v. 4, n. 11, p. 2343–2355, 2010.

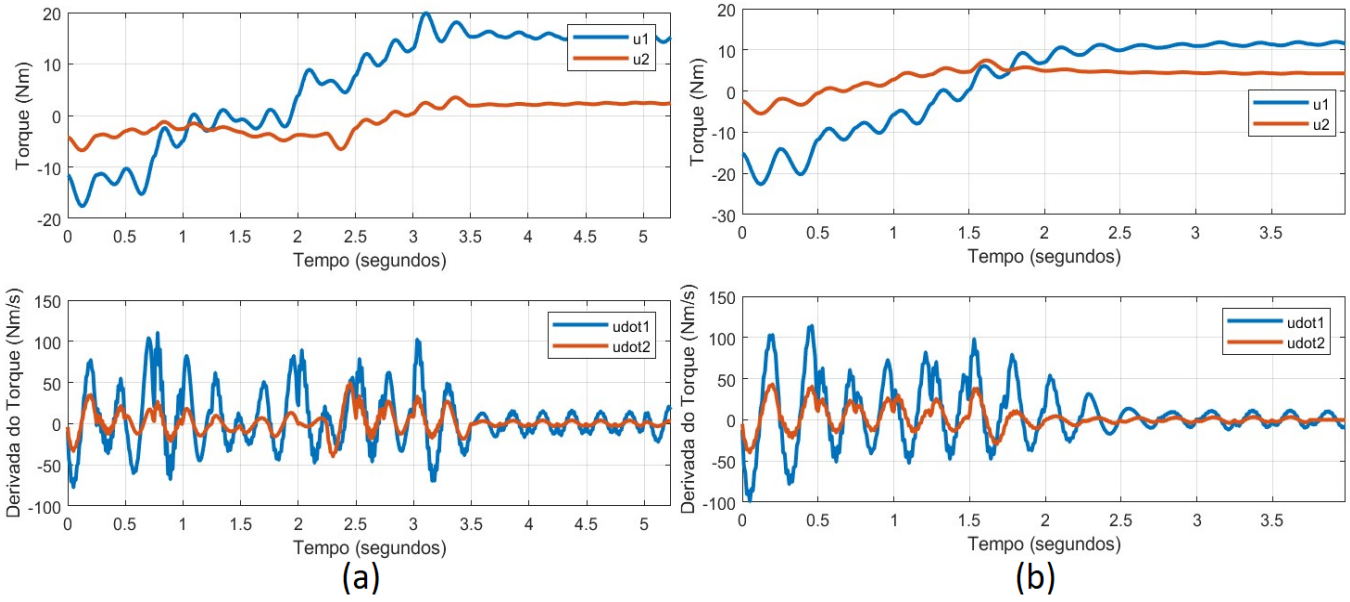
APÊNDICE A – Simulações do Robô com Soft Constraints

Figura 44 – Trajetórias com *Soft Constraints*: (a) Trajetória 1, (b) Trajetória 2



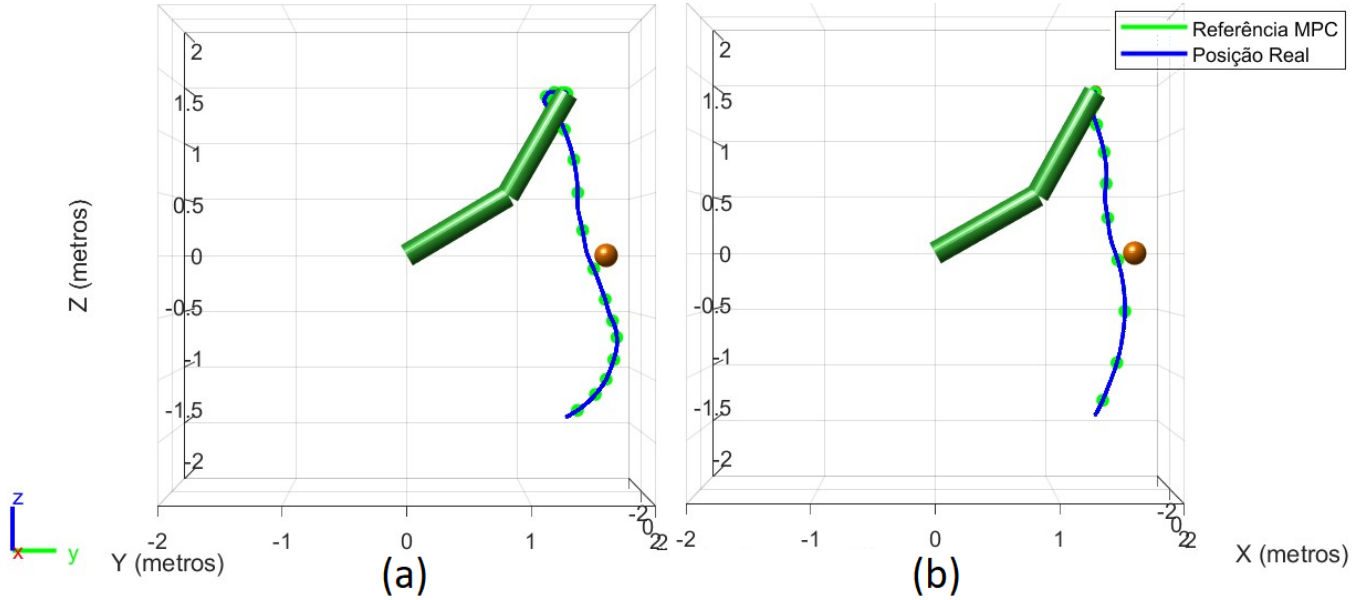
Fonte: Autoria própria (2023).

Figura 45 – Sinal de Controle com *Soft Constraints*: (a) Trajetória 1, (b) Trajetória 2



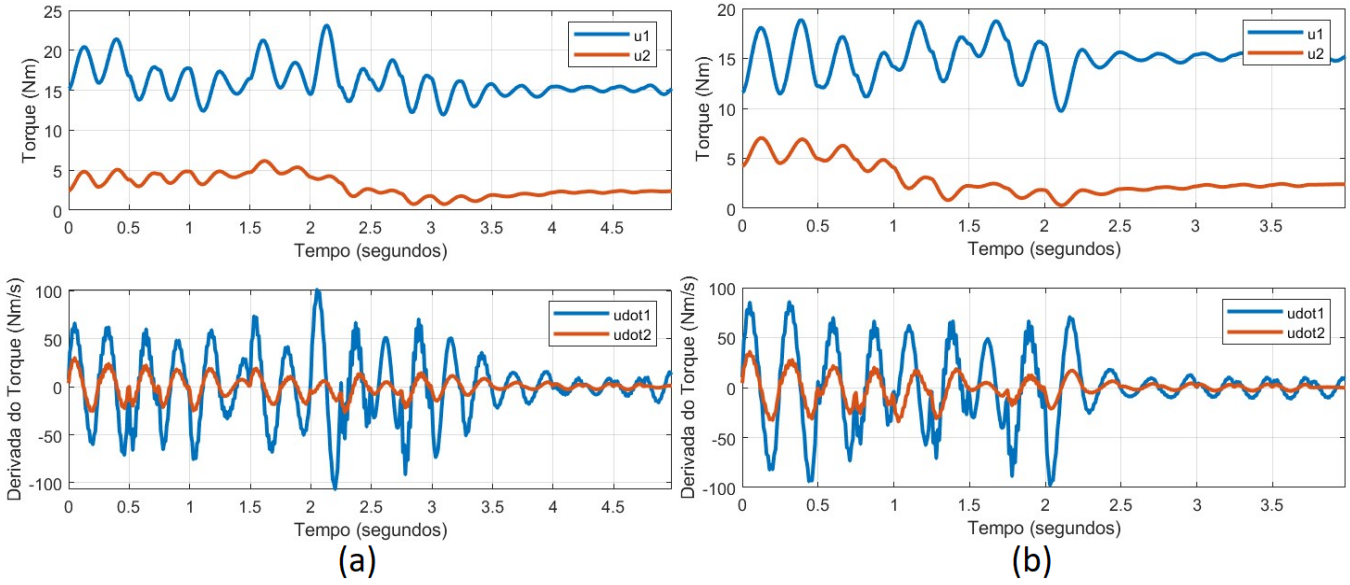
Fonte: Autoria própria (2023).

Figura 46 – Trajetórias com *Soft Constraints*: (a) Trajetória 3, (b) Trajetória 4



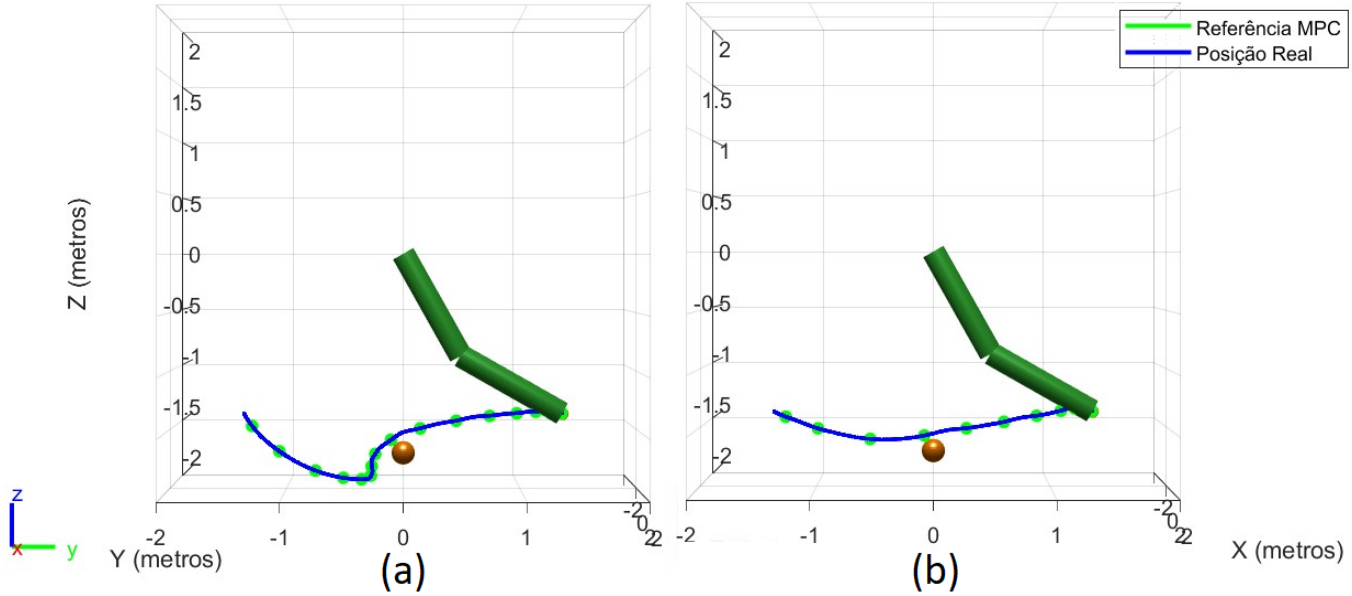
Fonte: Autoria própria (2023).

Figura 47 – Sinal de Controle com *Soft Constraints*: (a) Trajetória 3, (b) Trajetória 4



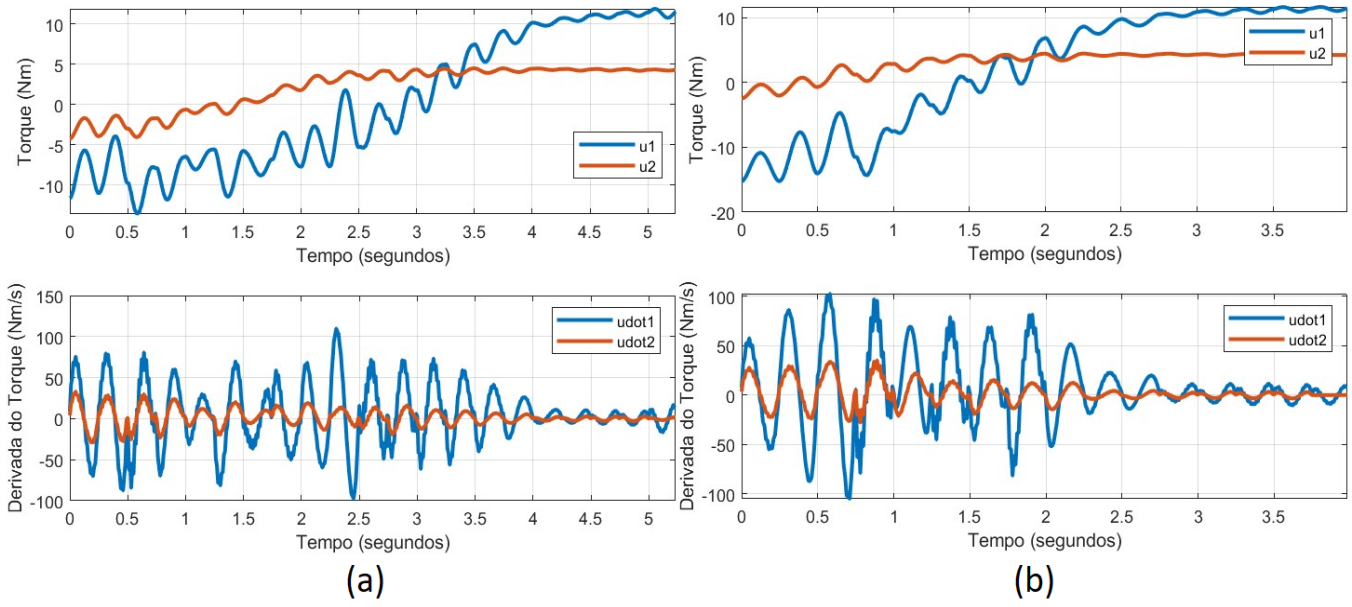
Fonte: Autoria própria (2023).

Figura 48 – Trajetórias com *Soft Constraints*: (a) Trajetória 5, (b) Trajetória 6



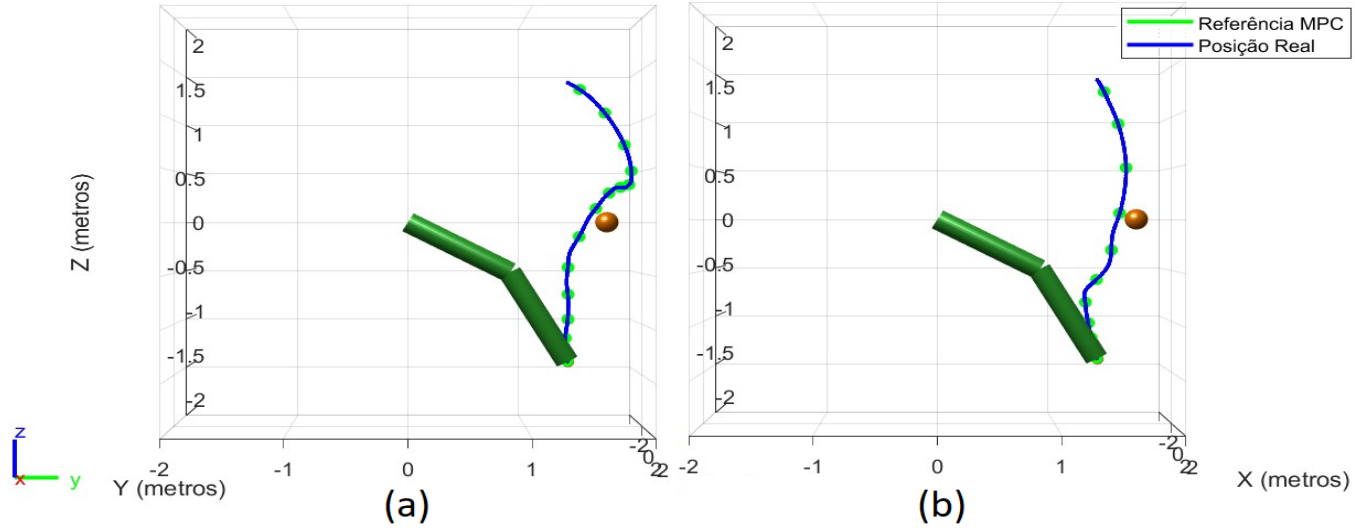
Fonte: Autoria própria (2023).

Figura 49 – Sinal de Controle com *Soft Constraints*: (a) Trajetória 5, (b) Trajetória 6



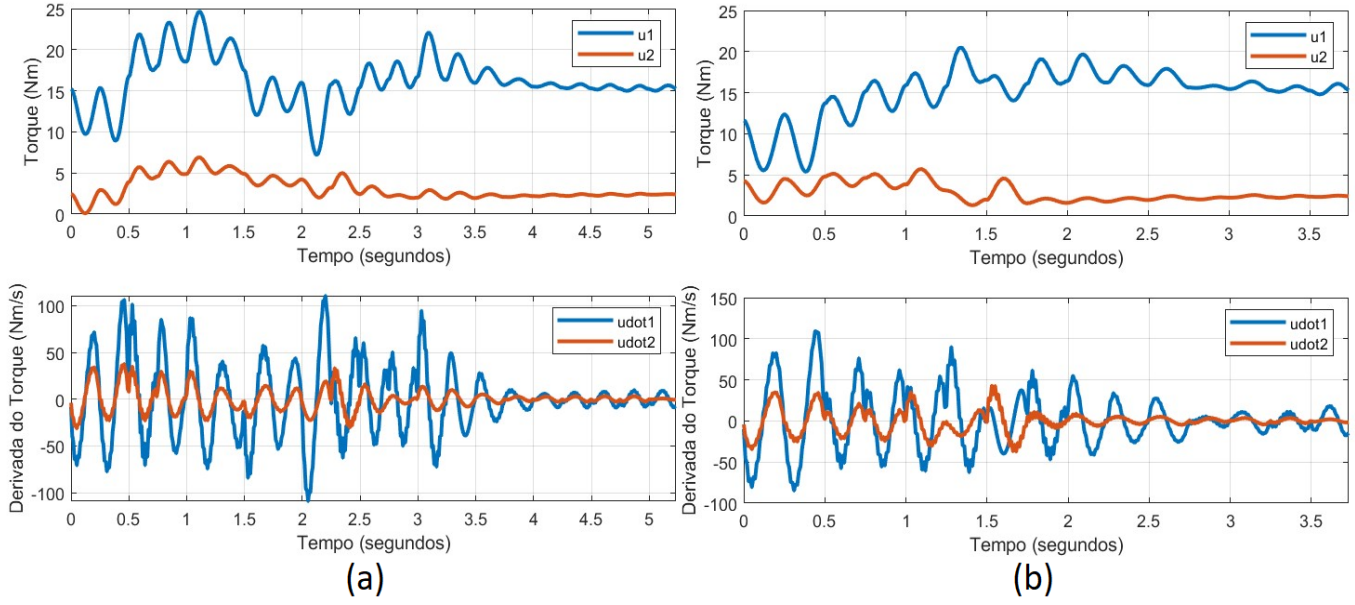
Fonte: Autoria própria (2023).

Figura 50 – Trajetórias com *Soft Constraints*: (a) Trajetória 7, (b) Trajetória 8



Fonte: Autoria própria (2023).

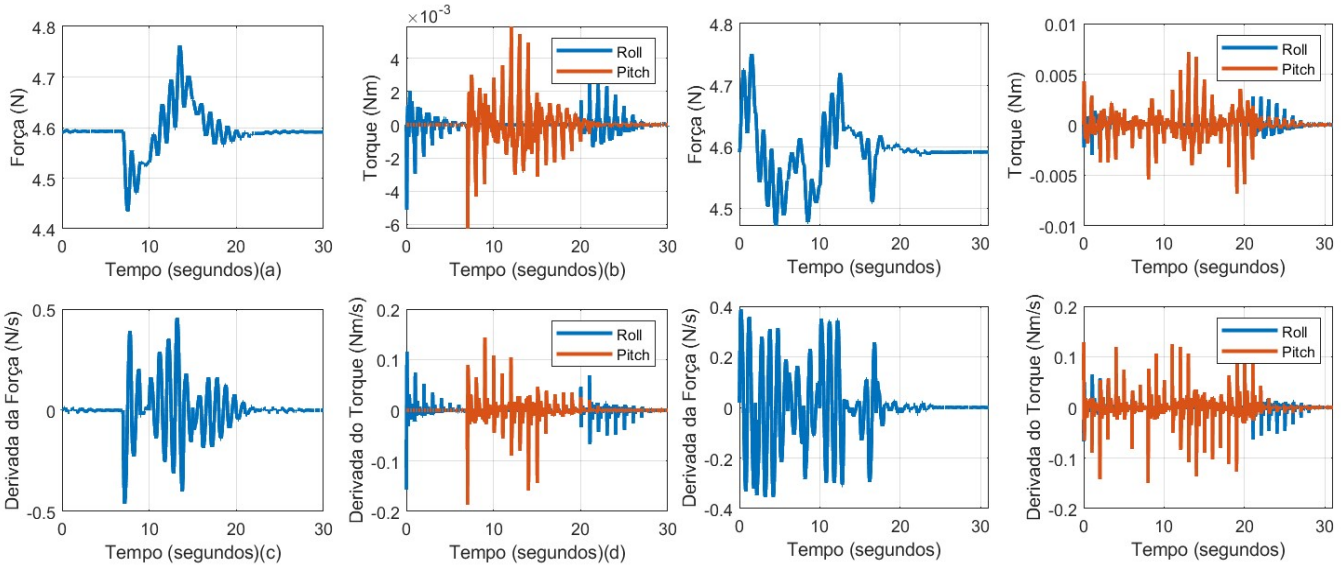
Figura 51 – Sinal de Controle com *Soft Constraints*: (a) Trajetória 7, (b) Trajetória 8



Fonte: Autoria própria (2023).

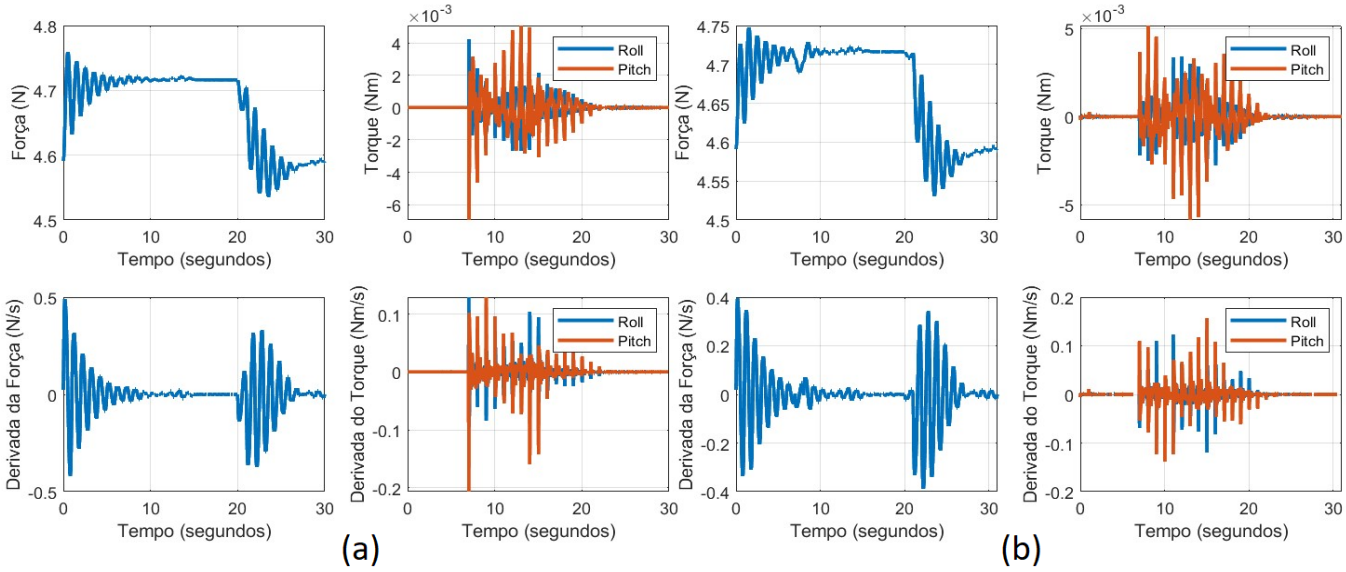
APÊNDICE B – Sinais de Controle dos Drones Quadricópteros

Figura 52 – Sinal de Controle do Drone em Y: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque



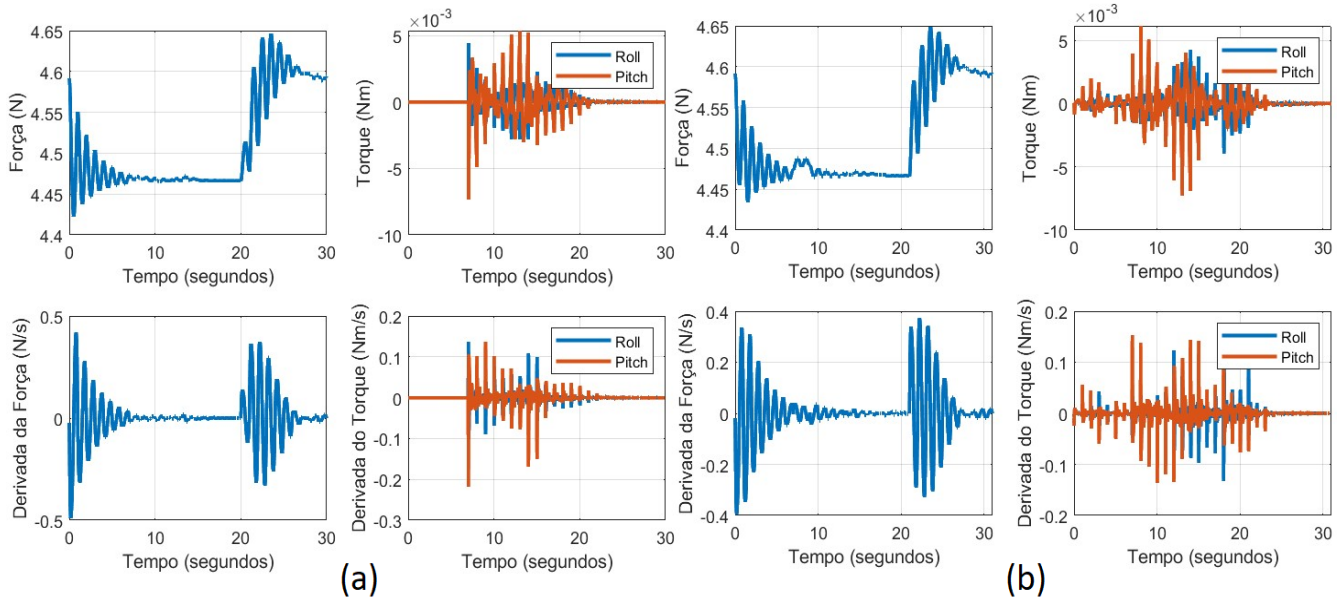
Fonte: Autoria própria (2023).

Figura 53 – Sinal de Controle do Drone em Z: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque



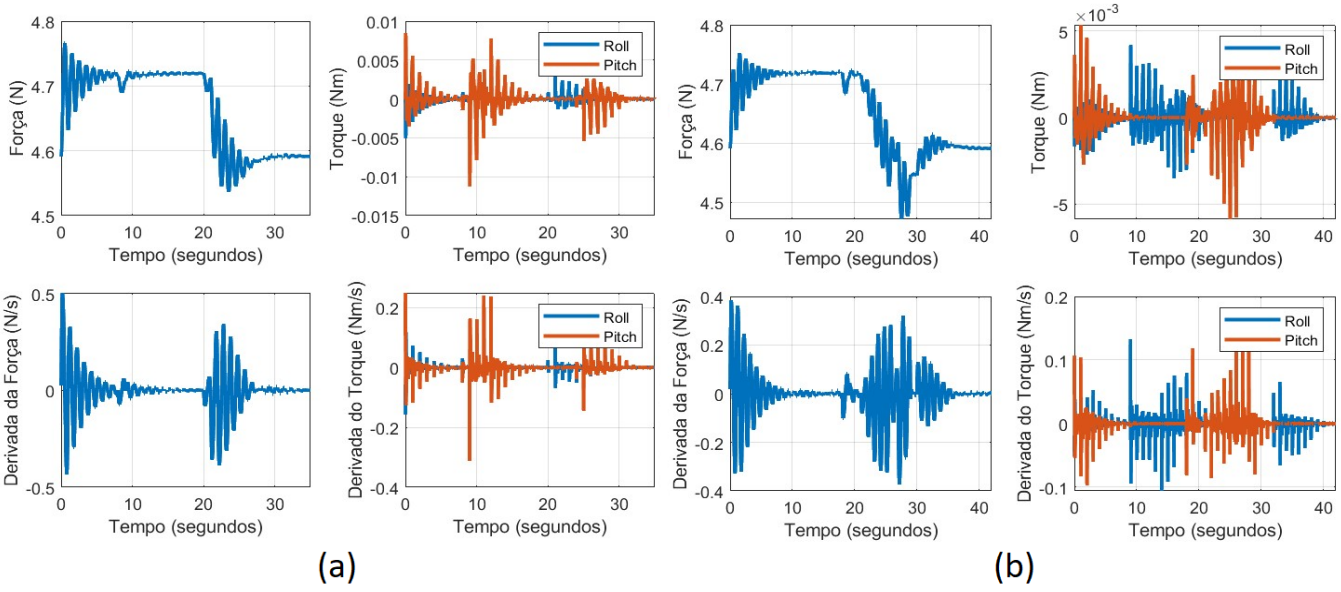
Fonte: Autoria própria (2023).

Figura 54 – Sinal de Controle do Drone em -Z: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque



Fonte: Autoria própria (2023).

Figura 55 – Sinal de Controle do Drone em XYZ: (a) Sem Restrição de Derivada de Força e Torque, (b) Com Restrição de Derivada de Força e Torque



Fonte: Autoria própria (2023).