
# Relatório Final PIBIC

# Controladores Preditivos Não-Lineares Para Controle de Trajetória

Relatório submetido à Universidade Federal de Santa Catarina

**Orientador:** Prof. Eduardo Camponogara
**Coorientador:** Jean P. Jordanou

Gabriel Stockmann Lima

Florianópolis
Maio de 2026 – Agosto de 2026

# Resumo

O controle autônomo de trajetória para drones enfrenta grandes desafios devido à natureza não linear de sua dinâmica, além da presença de limites operacionais e obstáculos no ambiente. Para abordar essa problemática, este trabalho apresenta o desenvolvimento de um Controlador Preditivo Não Linear (NMPC) formulado via dualidade de Lagrange para garantir o desvio de obstáculos de forma eficiente. Investigou-se o comportamento do drone sob diferentes perspectivas: a inclusão de variáveis de estado adicionais (velocidades nos eixos $x$ e $y$) e a implementação da estratégia de horizonte rolante (*receding horizon*) em malha fechada. Avaliou-se o impacto do tamanho do horizonte no tempo de convergência, utilizando uma matriz de custo terminal simétrica e definida positiva $\boldsymbol{P}_{\text{mat}}$, ponderada pela solução de uma Equação Algébrica Discreta de Riccati (DARE), para garantir as condições assintóticas de uma janela de predição infinita. Analisou-se também a implementação de *warm-start* para a redução do tempo de convergência, discutindo-se os cenários em que essa técnica pode degradar o desempenho computacional em função do horizonte adotado, além da resposta do sistema em cenários de fase não mínima com alta proximidade inicial ao obstáculo. Diante da inviabilidade gerada por restrições rígidas em situações críticas, adotou-se uma estratégia de flexibilização por meio de restrições suaves (*soft constraints*) penalizadas quadraticamente. Os resultados demonstraram que o horizonte rolante proporciona uma navegação suave e computacionalmente viável, enquanto o uso de variáveis de folga assegura a viabilidade do solucionador sem comprometer a segurança. Por fim, as simulações geraram um banco de dados consistente para o aprendizado da lei de controle por redes neurais.

**Palavras-chave:** Controle Preditivo Não Linear (NMPC). Dualidade de Lagrange. Horizonte Rolante. Restrições Suaves.

# Abstract

Autonomous trajectory control for drones faces major challenges due to their nonlinear dynamics and the presence of operational limits and environmental obstacles. To address this problem, this work presents the development of a Nonlinear Model Predictive Control (NMPC) formulated via Lagrange duality to ensure efficient obstacle avoidance. The drone's behavior was investigated under different perspectives: the inclusion of additional state variables (velocities in the $x$ and $y$ axes) and the implementation of a closed-loop receding horizon strategy. The impact of the prediction horizon size on convergence time was evaluated by incorporating a symmetric and positive-definite terminal cost matrix $\boldsymbol{P}_{\text{mat}}$, weighted by the solution of a Discrete Algebraic Riccati Equation (DARE) to guarantee asymptotic conditions equivalent to an infinite prediction horizon. Furthermore, the implementation of warm-start techniques to reduce convergence time was analyzed, highlighting scenarios where performance may degrade depending on the horizon length, alongside the system response in non-minimum phase conditions with high initial proximity to the obstacle. To overcome the infeasibility caused by hard constraints in critical situations, a slack strategy based on quadratically penalized soft constraints was adopted. Results demonstrated that the receding horizon approach ensures smooth and computationally feasible navigation, while slack variables guarantee solver feasibility without compromising safety. Finally, the simulations generated a consistent dataset for learning the control law using neural networks.



**Keywords:** Non-linear Model Predictive Control (NMPC). Lagrangian Duality. Receding Horizon. Soft Constraints.

# Sumário

# 1 Introdução

Nos últimos anos, os Veículos Aéreos Não Tripulados (VANTs), popularmente conhecidos como drones, têm ganhado destaque na prestação de serviços agrários, comerciais e militares [1]. A transição para a operação totalmente autônoma desses drones revolucionou essas áreas, à medida que permitiu sua operação sem a necessidade de um piloto humano, viabilizando missões complexas de monitoramento, inspeção industrial e logística. Contudo, para que essa autonomia seja viável e segura, é necessário que o drone consiga operar de forma eficiente em ambientes dinâmicos e repletos de imprevistos [2]. O grande desafio reside em evitar obstáculos em tempo real [3], tendo em vista a natureza altamente não linear dos drones, bem como os limites operacionais do próprio veículo [4].

## 1.1 Caracterização do Problema

O controle da navegação de drones, ainda que em seu estágio mais atômico, é um problema significativamente complexo para que seja implementada uma estratégia de controle convencional, pois se trata de um problema:

1. Multivariável: No caso mais simples, em um espaço bidimensional sem considerar variáveis de estado adicionais como a velocidade e perturbações como o deslizamento ou uma carga adicionada ao drone, ainda haveria duas variáveis de estado e duas ações de controle.

2. Restrições: Existem diversas restrições considerando elementos do ambiente, limitações de sensores, etc., modeladas a partir de inequações matemáticas.

3. Otimização: Além de evitar colisões, o drone precisa seguir uma trajetória ótima, de modo a evitar desgastes sem necessidade, enquanto desvia de obstáculos e atinge o objetivo final. Tratando-se, portanto, de um problema de otimização.

Tendo em vista estas condições, tal problema teria sua eficiência restringida caso um controlador PID fosse implementado, pois não lida bem com sistemas MIMO (Multiple Inputs Multiple Outputs) [5]. Para a solução mais eficiente de tal problema, existe uma estratégia de controle que se encaixa perfeitamente com as condições enfrentadas: trata-se de um controlador preditivo não linear (NMPC) [6, 3]. O NMPC opera com as restrições exigidas pelo problema, com as diversas variáveis de estado e ações de controle envolvidas diretamente em seu modelo matemático [4].

Contudo, o NMPC ainda é um problema de otimização que, apesar de eficiente, pode ser computacionalmente caro, pois exige o cálculo das ações de controle

ótimas em cada iteração. Para isso, outra estratégia promissora é a utilização de uma rede neural com função de ativação ReLU, já que a avaliação da rede possui custo computacional desprezível em tempo de execução, transferindo a carga do cálculo para a fase prévia de treinamento e verificação offline. Assim, uma vez treinada, a rede passa a aproximar a lei de controle do NMPC com elevada eficiência computacional na operação em tempo real.

## 1.2 Objetivos

Com base no contexto fornecido, os objetivos deste projeto de iniciação científica são:

1. Projeto de controlador NMPC que atenda às especificações do problema;
2. Implementação e teste do NMPC projetado com variáveis adicionais de estado e condições adversas, adição do horizonte temporal rolante;
3. Geração de base de dados para treinamento de redes neurais.

O projeto da rede neural e seu teste ficarão a cargo de Victor Nishida Raposo (Bolsista PIBIC).

# 2 Materiais e Métodos

## 2.1 Definição do problema

*A priori*, define-se o conjunto de estados iniciais como $\mathscr{X}_{\text{init}} = \{\mathbf{x}_{\text{init}} \in \mathbb{R}^2 \mid \|\mathbf{x}_{\text{init}} - \mathbf{x}_c\|_\infty \leq 0{,}5\}$, em que $\mathbf{x}_c = [-5, 0]^\top$ e $\mathbf{x}_{\text{init}}$ trata-se da efetiva posição inicial ocupada pelo drone. O obstáculo é representado pelo poliedro $\mathscr{O} = \{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ (no caso específico de uma caixa centrada na origem, $\mathscr{O} = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_\infty \leq 1{,}0\}$). O objetivo da navegação é atingido quando o drone alcança o estado final $\mathbf{x}_{\text{goal}} = [5, 0]^\top$. Tais definições são ilustradas na Figura 1.

Conforme a formulação proposta por Zago et al. [7], para a modelagem do sistema dinâmico discreto e invariante no tempo (LTI), adota-se a seguinte dinâmica de transição:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k, \tag{1}$$

em que $\mathbf{x}_k \in \mathbb{R}^2$ representa o estado (posição) e $\mathbf{u}_k \in \mathbb{R}^2$ denota o vetor de ação de controle no instante $k$.

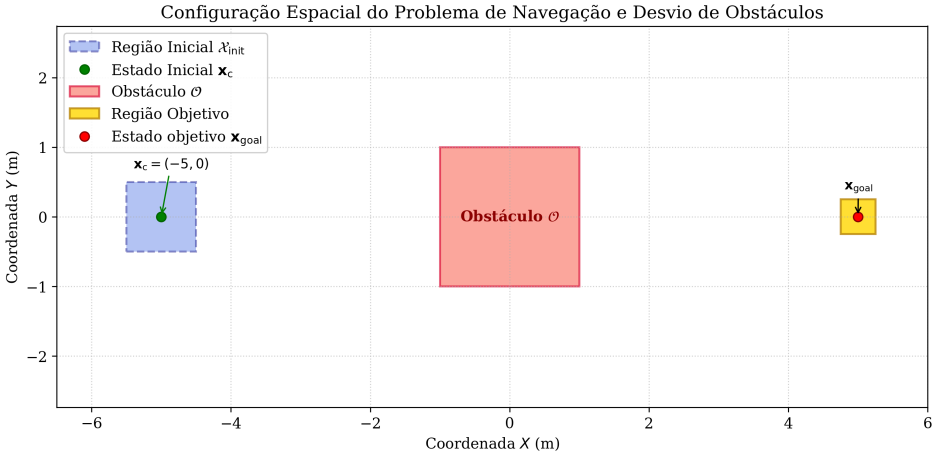


Figura 1: Configuração espacial do problema de navegação, contendo a região inicial $\mathscr{X}_{\text{init}}$, o obstáculo $\mathscr{O}$ e o estado objetivo $\mathbf{x}_{\text{goal}}$.

Seguindo a estrutura desenvolvida em [7], a função custo e as restrições do problema de controle ótimo por horizonte preditivo são formuladas como:

$$V(\mathbf{x}_{\text{init}}) = \min_{\mathbf{x}_k, \mathbf{u}_k} \sum_{k=1}^{T} \bar{\mathbf{x}}_k^\top \mathbf{P} \bar{\mathbf{x}}_k + \sum_{k=1}^{T} \mathbf{u}_k^\top \mathbf{Q} \mathbf{u}_k \tag{2a}$$

$$\text{s.t.} \begin{cases} \bar{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_{\text{goal}}, \\ \mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{u}_k, \quad \forall k \in \{1, \dots, T\}, \\ \text{dist}(\mathbf{x}_k, \mathscr{O}) \geq 1, \quad \forall k \in \{1, \dots, T\}. \end{cases} \tag{2b}$$

Note que a condição de desvio $\text{dist}(\mathbf{x}_k, \mathscr{O}) \geq 1$ envolve uma restrição não convexa no espaço de estados. A distância euclidiana de um ponto $\mathbf{x}_k$ ao conjunto poliedral $\mathscr{O}$ é definida por um problema de minimização interno:

$$\text{dist}(\mathbf{x}_k, \mathscr{O}) = \min_{\mathbf{x}_{\text{init}}, \mathbf{z}} \|\mathbf{z}\|_2 \tag{3a}$$

$$\text{s.t.} \begin{cases} \mathbf{z} = \mathbf{x}_k - \mathbf{x}_{\text{init}}, \\ \mathbf{A}\mathbf{x}_{\text{init}} \leq \mathbf{b}. \end{cases} \tag{3b}$$

Inserir essa minimização diretamente na função custo resultaria em um problema de otimização aninhado (*bilevel optimization*), cuja elevada complexidade computacional inviabiliza a sua resolução em tempo real devido ao alto custo de otimização hierárquica [8]. Para contornar essa limitação, adota-se a abordagem do Lagrangiano dual apresentada em [7]. Como o problema interno é estritamente convexo, a dualidade forte se mantém, permitindo reescrever a restrição de distân-

cia de forma explícita através do seu problema dual e incorporar os multiplicadores de Lagrange diretamente no problema principal de NMPC.

## 2.2 Fundamentos de Otimização Convexa

Um problema geral de otimização matemática possui a forma padrão [9, 10]:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m, \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p, \end{aligned} \tag{4}$$

em que $\mathbf{x} \in \mathbb{R}^n$ é o vetor de variáveis de decisão, $f : \mathbb{R}^n \to \mathbb{R}$ é a função objetivo, $g_i : \mathbb{R}^n \to \mathbb{R}$ representam as restrições de desigualdade e $h_j : \mathbb{R}^n \to \mathbb{R}$ representam as restrições de igualdade.

### Conjuntos e Funções Convexas

**Definição 1** (Conjunto Convexo)**.** *Um conjunto $\mathscr{C} \subseteq \mathbb{R}^n$ é dito **convexo** se, para quaisquer pontos $\mathbf{x}, \mathbf{y} \in \mathscr{C}$ e qualquer escalar $\gamma \in [0, 1]$, tem-se:*

$$\gamma\mathbf{x} + (1 - \gamma)\mathbf{y} \in \mathscr{C}. \tag{5}$$

*Geometricamente, isso significa que o segmento de reta que conecta qualquer par de pontos pertencentes a $\mathscr{C}$ está totalmente contido em $\mathscr{C}$.*

**Definição 2** (Função Convexa)**.** *Uma função $f : \mathscr{C} \to \mathbb{R}$ definida sobre um conjunto convexo $\mathscr{C}$ é dita **convexa** se, para quaisquer $\mathbf{x}, \mathbf{y} \in \mathscr{C}$ e $\gamma \in [0, 1]$:*

$$f(\gamma\mathbf{x} + (1 - \gamma)\mathbf{y}) \leq \gamma f(\mathbf{x}) + (1 - \gamma) f(\mathbf{y}). \tag{6}$$

*A função é dita **estritamente convexa** se a desigualdade for estrita para $\mathbf{x} \neq \mathbf{y}$ e $\gamma \in (0, 1)$.*

Para funções de classe $\mathscr{C}^2$ (duas vezes continuamente diferenciáveis), a convexidade pode ser verificada analiticamente através do estudo da matriz Hessiana $\nabla^2 f(\mathbf{x})$:

$$f \text{ é convexa} \iff \nabla^2 f(\mathbf{x}) \succeq \mathbf{0}, \quad \forall \mathbf{x} \in \mathscr{C}, \tag{7}$$

isto é, a matriz Hessiana deve ser semidefinida positiva em todo o domínio.

### Propriedades dos Problemas de Otimização Convexos

Um problema na forma (4) é denominado **convexo** se:

1. A função objetivo $f(\mathbf{x})$ for convexa;
2. As funções de restrição de desigualdade $g_i(\mathbf{x})$ forem convexas para todo $i = 1, \ldots, m$;
3. As funções de restrição de igualdade $h_j(\mathbf{x})$ forem afins (ou seja, da forma $h_j(\mathbf{x}) = \mathbf{a}_j^\top \mathbf{x} - b_j$).

Uma propriedade fundamental dos problemas de otimização convexos é que **todo ponto de mínimo local é garantidamente um mínimo global**. Além disso, caso a função objetivo $f(\mathbf{x})$ seja estritamente convexa, o ponto de mínimo global é **único**.

## 2.3 Dualidade de Lagrange

### A Função Lagrangiana e a Função Dual

Para incorporar e relaxar as restrições do problema (4), define-se a **função Lagrangiana** $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ [9, 10]:

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \mu_j h_j(\mathbf{x}), \tag{8}$$

em que $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_m]^\top \geq \mathbf{0}$ e $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_p]^\top$ são os **multiplicadores de Lagrange** (ou variáveis duais).

A **função dual de Lagrange** $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ é definida como o ínfimo da função Lagrangiana em relação às variáveis primais $\mathbf{x}$:

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}). \tag{9}$$

Uma propriedade crucial é que a função dual $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$ é **sempre côncava** (mesmo que o problema primal não seja convexo) e fornece um limite inferior (*lower bound*) para o valor ótimo primal $p^*$:

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq p^*, \quad \forall \boldsymbol{\lambda} \geq \mathbf{0},\ \boldsymbol{\mu}. \tag{10}$$

O **Problema Dual de Lagrange** busca encontrar a melhor limitação inferior possível:

$$\begin{aligned} \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \quad & g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \\ \text{s.t.} \quad & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{11}$$

A relação $d^* \leq p^*$ é chamada de **dualidade fraca**. Quando $d^* = p^*$, tem-se **dualidade forte**, a qual é garantida para problemas convexos sob condições de regularidade (como a condição de Slater).

## 2.4 Condições de Karush-Kuhn-Tucker (KKT)

Se o problema for convexo e a dualidade forte for atendida, as condições de Karush-Kuhn-Tucker (KKT) são necessárias e suficientes para a otimalidade primal-dual $(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ [9, 10]:

1. **Estacionariedade:** $\nabla_{\mathbf{x}} L(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \mathbf{0}$;
2. **Factibilidade Primal:** $g_i(\mathbf{x}^*) \leq 0,\ \forall i = 1, \ldots, m \quad \text{e} \quad h_j(\mathbf{x}^*) = 0,\ \forall j = 1, \ldots, p$;
3. **Factibilidade Dual:** $\lambda_i^* \geq 0,\ \forall i = 1, \ldots, m$;
4. **Folga Complementar:** $\lambda_i^* g_i(\mathbf{x}^*) = 0,\ \forall i = 1, \ldots, m$.

## 2.5 Aplicação da Dualidade Lagrangeana ao Problema de Distância Mínima

Considerando o problema de calcular a distância entre o estado $\mathbf{x}_k$ e um obstáculo politópico $\mathscr{O} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$, a função dual Lagrangeana do problema de distância é expressa por:

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) &= \inf_{\mathbf{x}_o, \mathbf{z}} \|\mathbf{z}\|_2 + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x}_o - \mathbf{b}) + \boldsymbol{\mu}^\top (\mathbf{x}_k - \mathbf{x}_o - \mathbf{z}) \\ &= -\boldsymbol{\lambda}^\top \mathbf{b} + \boldsymbol{\mu}^\top \mathbf{x}_k + \inf_{\mathbf{x}_o} (\boldsymbol{\lambda}^\top \mathbf{A} - \boldsymbol{\mu}^\top)\mathbf{x}_o + \inf_{\mathbf{z}} \left( \|\mathbf{z}\|_2 - \boldsymbol{\mu}^\top \mathbf{z} \right), \end{aligned}$$

em que $\boldsymbol{\lambda} \geq \mathbf{0}$ e $\boldsymbol{\mu}$ são as variáveis duais.

Observe que $g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = -\infty$ se $\boldsymbol{\lambda}^\top \mathbf{A} - \boldsymbol{\mu}^\top \neq \mathbf{0}$ ou $\|\boldsymbol{\mu}\|_2 > 1$. Considerando apenas os multiplicadores duais viáveis, o problema de distância dual é formulado como:

$$\begin{aligned} \operatorname{dist}(\mathbf{x}_k, \mathscr{O}) = \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}} \quad & -\boldsymbol{\lambda}^\top \mathbf{b} + \boldsymbol{\mu}^\top \mathbf{x}_k, && \text{(12a)} \\ \text{s.t} \quad & \boldsymbol{\lambda}^\top \mathbf{A} - \boldsymbol{\mu}^\top = \mathbf{0}, && \text{(12b)} \\ & \boldsymbol{\lambda} \geq \mathbf{0}, && \text{(12c)} \\ & \|\boldsymbol{\mu}\|_2 \leq 1. && \text{(12d)} \end{aligned}$$

## 2.6 Formulação da Função Custo Explicita no Controle Preditivo (MPC)

Substituindo o problema de distância em termos das variáveis duais diretamente como uma restrição de desvio de obstáculos no problema de otimização do controlador, obtém-se a seguinte função custo implícita:

$$V(\mathbf{x}_{\text{init}}) = \min_{\mathbf{x}_k, \mathbf{u}_k} \sum_{k=1}^{T} \bar{\mathbf{x}}_k^\top \mathbf{P} \bar{\mathbf{x}}_k + \sum_{k=1}^{T} \mathbf{u}_k^\top \mathbf{Q} \mathbf{u}_k, \tag{13a}$$

$$\text{s.t.} \begin{cases} \bar{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_{\text{goal}}, \\ \mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{u}_k, \quad \forall k \in \{1, \ldots, T\}, \\ -\boldsymbol{\lambda}_k^\top \mathbf{b} + \boldsymbol{\mu}_k^\top \mathbf{x}_k \geq 1, \\ \boldsymbol{\lambda}_k^\top \mathbf{A} - \boldsymbol{\mu}_k^\top = \mathbf{0}, \\ \boldsymbol{\lambda}_k \geq \mathbf{0}, \\ \|\boldsymbol{\mu}_k\|_2 \leq 1, \quad \forall k \in \{1, \ldots, T\}. \end{cases} \tag{13b}$$

Com esta reformulação, as variáveis duais tornam-se variáveis de decisão adicionais no próprio problema principal, eliminando a formulação aninhada original. Note que, apesar de a função custo ser quadrática e a maioria das restrições serem lineares ou convexas, o problema global é não convexo devido à restrição $-\boldsymbol{\lambda}_k^\top \mathbf{b} + \boldsymbol{\mu}_k^\top \mathbf{x}_k \geq 1$, a qual contém o produto de duas variáveis de decisão, caracterizando um termo bilinear.

Problemas dessa natureza podem ser resolvidos por solvers globais (como Gurobi, SCIP, Couenne e BARON) ou por solvers locais (como o IPOPT). No entanto, o custo computacional para a otimização global pode ser proibitivo em tempo real, ao passo que a solução local depende fortemente do chute inicial e pode resultar em estagnação em mínimos locais ruins ou em falhas de viabilidade. Sob essa perspectiva, destaca-se a importância de resolver o problema offline para diversos estados iniciais, gerando um conjunto de trajetórias ótimas que podem ser utilizadas para o treinamento de redes neurais.

## 2.7 Horizonte Rolante, Custo Terminal e Garantia de Estabilidade via DARE

Como já explicitado, o controlador NMPC calcula uma sequência de ações de controle ótimas ao longo de um determinado horizonte de predição. Em formulações teóricas de controle preditivo em tempo discreto, a adoção de um horizonte de predição infinito ($k_{\text{pred}} \to \infty$) assegura de forma implícita a estabilidade assintótica do sistema em malha fechada [11]. Contudo, do ponto de vista de otimização em

tempo real, resolver um problema de programação quadrática ou não linear com horizonte infinito é computacionalmente inviável [4].

Para conciliar a viabilidade de execução em tempo real com a capacidade de antecipação do controlador, adota-se a estratégia do **horizonte rolante** (*receding horizon*) [12, 4]. Sob essa abordagem, a janela de otimização de tamanho finito $k_{\text{pred}}$ não permanece estática, mas desloca-se continuamente a cada instante de amostragem $k$, acompanhando o estado atual do drone [13]. O algoritmo resolve o problema de otimização para toda a trajetória futura no intervalo $[k, k + k_{\text{pred}}]$, aplica estritamente a primeira ação de controle $\mathbf{u}(k)$ e descarta as subsequentes, recalculando todo o processo no passo $k + 1$.

A dimensão desse horizonte finito $k_{\text{pred}}$ exige um dimensionamento criterioso:

- Um horizonte demasiadamente curto torna a navegação "míope", fazendo com que o drone reaja tardiamente às regiões de restrição e comprometendo a segurança do trajeto [4];
- Um horizonte excessivamente longo introduz uma antecipação desnecessária a perturbações distantes, além de elevar drasticamente o custo computacional por iteração [13].

Todavia, ao truncar o horizonte para um valor finito ($k_{\text{pred}} < \infty$), elimina-se a propriedade implícita de estabilidade assintótica inerente à formulação infinita [14, 11]. Sem um mecanismo de compensação, o sistema otimizado para um horizonte curto pode apresentar comportamentos agressivos ao final da janela de predição ou mesmo induzir a malha fechada à instabilidade no longo prazo [15].

### Custo Terminal Ponderado via DARE

Para restabelecer as garantias formais de estabilidade em sentido de Lyapunov sem requerer o aumento do horizonte $k_{\text{pred}}$ (o que inviabilizaria a execução em tempo real), insere-se na função objetivo um **custo terminal** mapeado por uma matriz de ponderação simétrica e definida positiva $\mathbf{P} \succ \mathbf{0}$.

Considere a função de custo do MPC quadrático discretizado dada por:

$$J(\mathbf{x}_t, \mathbf{u}) = \sum_{k=0}^{k_{\text{pred}}-1} \left( \mathbf{x}_k^\top \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k \right) + \mathbf{x}_{k_{\text{pred}}}^\top \mathbf{P} \mathbf{x}_{k_{\text{pred}}}, \tag{14}$$

em que $\mathbf{Q} \succeq \mathbf{0}$ e $\mathbf{R} \succ \mathbf{0}$ são as matrizes de ponderação dos estados e das ações de controle ao longo do horizonte, respectivamente, e o termo $\mathbf{x}_{k_{\text{pred}}}^\top \mathbf{P} \mathbf{x}_{k_{\text{pred}}}$ penaliza o estado no final do horizonte de predição.

A matriz $\mathbf{P}$ atua como a aproximação do "custo para ir" (*cost-to-go*) do estado terminal $\mathbf{x}_{k_{\text{pred}}}$ até o infinito, assumindo que, após $k_{\text{pred}}$, o sistema operará sob

uma lei de controle linear assintoticamente estável $\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k$ [11]. Para que o custo terminal corresponda exatamente à energia acumulada no horizonte infinito restante, a matriz $\mathbf{P}$ deve satisfazer a **Equação Algébrica de Riccati Discreta (DARE)**:

$$\mathbf{A}^\top\mathbf{P}\mathbf{A} - \mathbf{P} + \mathbf{Q} - \mathbf{A}^\top\mathbf{P}\mathbf{B}\left(\mathbf{R} + \mathbf{B}^\top\mathbf{P}\mathbf{B}\right)^{-1}\mathbf{B}^\top\mathbf{P}\mathbf{A} = \mathbf{0}, \tag{15}$$

em que $\mathbf{A}$ e $\mathbf{B}$ representam as matrizes do sistema linearizado em torno do ponto de operação. Quando $\mathbf{P}$ satisfaz a Eq. (15), obtém-se concomitantemente o ganho ótimo de realimentação terminal $\mathbf{K} = (\mathbf{R} + \mathbf{B}^\top\mathbf{P}\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{P}\mathbf{A}$.

### Garantia de Estabilidade de Lyapunov em Malha Fechada

A estabilidade assintótica do sistema sob o paradigma do horizonte rolante com custo terminal pode ser provada definindo a própria função de custo ótimo $V(\mathbf{x}_k) = J^*(\mathbf{x}_k)$ como uma função candidata de Lyapunov $V : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$.

Para que $V(\mathbf{x}_k)$ seja uma função de Lyapunov estrita, satisfazem-se as seguintes propriedades:

1. $V(\mathbf{x}_k) > 0, \quad \forall \mathbf{x}_k \neq \mathbf{0} \quad \text{e} \quad V(\mathbf{0}) = 0;$
2. $\Delta V(\mathbf{x}_k) = V(\mathbf{x}_{k+1}) - V(\mathbf{x}_k) < 0, \quad \forall \mathbf{x}_k \neq \mathbf{0}$ (Decrescimento monótono ao longo das trajetórias em malha fechada).

Com a incorporação da matriz $\mathbf{P}$ proveniente da DARE no termo terminal, a variação do custo ótimo entre instantes amostrais sucessivos satisfaz a desigualdade:

$$V(\mathbf{x}_{k+1}) - V(\mathbf{x}_k) \leq -\left(\mathbf{x}_k^\top\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^\top\mathbf{R}\mathbf{u}_k\right). \tag{16}$$

Sendo $\mathbf{Q} \succ \mathbf{0}$ e $\mathbf{R} \succ \mathbf{0}$, o termo $-\left(\mathbf{x}_k^\top\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^\top\mathbf{R}\mathbf{u}_k\right)$ é estritamente negativo para qualquer $\mathbf{x}_k \neq \mathbf{0}$. Isso assegura que a energia do sistema decresce continuamente a cada iteração da estratégia de horizonte rolante, garantindo formalmente a estabilidade assintótica em sentido de Lyapunov em relação ao ponto de equilíbrio desejado.

### Desafios Numéricos na Resolução da DARE em Tempo Real

Em aplicações práticas com drones e plataformas NMPC adaptativas — nas quais as matrizes de linearização $\mathbf{A}$ e $\mathbf{B}$ variam com a trajetória e com a dinâmica do veículo —, a DARE precisa ser resolvida com baixíssimo tempo de processamento [16].

Metodologias clássicas de solução baseadas em decomposição de Schur ou autovetores possuem complexidade $\mathcal{O}(n^3)$, o que pode sobrecarregar a rotina em tempo

real [17]. Estruturas adaptativas e estimadores iterativos associados ao condicionamento numérico via **Pseudo-Inversa de Moore-Penrose** ($\mathbf{Y}^{+}$) garantem a estabilidade da inversão do termo $(\mathbf{R} + \mathbf{B}^{\top}\mathbf{P}\mathbf{B})^{+}$ mesmo diante de matrizes mal condicionadas, assegurando a rápida convergência e a atualização eficiente da matriz $\mathbf{P}$ a cada passo do horizonte rolante [18, 19].

# 3 Resultados e Discussão

## 3.1 1º Cenário: Otimização com Horizonte Estendido

Para a resolução do problema de otimização NLP não convexo derivado, o *solver* comercial Gurobi destaca-se como uma das alternativas mais adequadas. O principal motivo é a sua capacidade de resolver problemas com não convexidades bilineares ($\boldsymbol{\mu}_k^{\top}\mathbf{x}_k$), garantindo otimalidade global por meio do algoritmo *Spatial Branch-and-Bound* e de técnicas avançadas de relaxamento convexo (*Presolve*). Em comparação a *solvers* locais, o Gurobi evita a estagnação em mínimos locais inadequados; em comparação a outros *solvers* globais, oferece um desempenho computacional superior e alta eficiência na execução massiva necessária para a geração do conjunto de dados *offline*. A simulação é implementada através do seguinte script:

```
import gurobipy as gp
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import patches
import random
import pandas as pd

# Parametros de simulacao
T = 10
nxu = 2
nlam = 4

# Matrizes de ponderacao do custo
P = 50
Q = 1

# Geometria do obstaculo (A * x <= b)
A = np.array([
    [1, 0],
    [-1, 0],
    [0, 1],
    [0, -1]
])
b = np.array([1, 1, 1, 1])
```

```
25
26 # Pontos de interesse
27 x_init = [-5, 0]
28 x_goal = [5, 0]
29
30 # Criacao do Modelo
31 m = gp.Model("NMPC")
32 m.setParam("NonConvex", 2) # Permite restricoes quadraticas nao
       convexas
33
34 # Variaveis de Decisao
35 x = m.addVars(T, nxu, lb=-6, ub=6, name="pos")
36 x_bar = m.addVars(T, nxu, lb=-20, ub=20, name="x_bar")
37 u = m.addVars(T, nxu, lb=-1, ub=1, name="u")
38 lam = m.addVars(T, nlam, lb=0, name='Lambda')
39 mu = m.addVars(T, nxu, name='mu')
40
41 # Funcao Objetivo
42 obj = gp.quicksum(P * x_bar[k, i]**2 for k in range(T) for i in
       range(nxu)) + \
43       gp.quicksum(Q * u[k, i]**2 for k in range(T) for i in range(
       nxu))
44 m.setObjective(obj, sense=gp.GRB.MINIMIZE)
45
46 # Restricoes
47 for i in range(nxu):
48     m.addConstr(x[0, i] == x_init[i] + u[0, i])
49
50 for k in range(T):
51     for i in range(nxu):
52         m.addConstr(x_bar[k, i] == x[k, i] - x_goal[i])
53
54 for k in range(1, T):
55     for i in range(nxu):
56         m.addConstr(x[k, i] == x[k-1, i] + u[k, i])
57
58     # Restricao de evitacao de obstaculo via dualidade
59     m.addConstr(-gp.quicksum(lam[k, l] * b[l] for l in range(nlam)
       ) + \
60                 gp.quicksum(mu[k, i] * x[k, i] for i in range(nxu)
       ) >= 1)
61
62     for i in range(nxu):
63         m.addConstr(gp.quicksum(lam[k, l] * A[l, i] for l in range
       (nlam)) - mu[k, i] == 0)
64
65     m.addQConstr(gp.quicksum(mu[k, i] * mu[k, i] for i in range(
       nxu)) <= 1)
66
```

```
# Resolucao
m.optimize()

if m.status == gp.GRB.OPTIMAL:
    print("Otimo encontrado")
    todos_pontos = [x_init] + [[x[k, 0].X, x[k, 1].X] for k in range(T)]

    x_traj = np.array(todos_pontos)
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.plot(x_traj[:, 0], x_traj[:, 1], 'bo-', linewidth=2, label='Trajet ria')

    square_init = patches.Rectangle((-5.25, -0.25), 0.5, 0.5, edgecolor='black', facecolor='green', label='Estado inicial')
    ax.add_patch(square_init)

    square_goal = patches.Rectangle((4.75, -0.25), 0.5, 0.5, edgecolor='black', facecolor='yellow', label='Estado objetivo')
    ax.add_patch(square_goal)

    square = patches.Rectangle((-1, -1), 2, 2, edgecolor='black', facecolor='red', label='Obst culo')
    ax.add_patch(square)

    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_title('Trajet ria  tima  do NMPC')
    ax.grid(True)
    ax.axis('equal')
    ax.legend()
    plt.show()
else:
    print("Status do Solver:", m.status)
```

Listing 1: Algoritmo NMPC usando Gurobi em Python

A Figura 2 apresenta a trajetória gerada a partir da solução desse primeiro cenário.

Como explicitado na Seção 2.7, observa-se uma forte antecipação do drone em relação à região de restrição ao adotar um horizonte de predição excessivamente longo (neste caso, de $T = 10$ passos). Nota-se que o primeiro movimento executado pelo veículo já busca realizar o desvio. Conforme discutido anteriormente, essa abordagem antecipatória resulta em um elevado custo computacional, o qual escala significativamente à medida que o problema ganha complexidade com o acréscimo de dimensões e variáveis de estado.

Ademais, a ação de controle $\mathbf{u}$ está limitada ao intervalo $[-1, 1]$. Esse limite

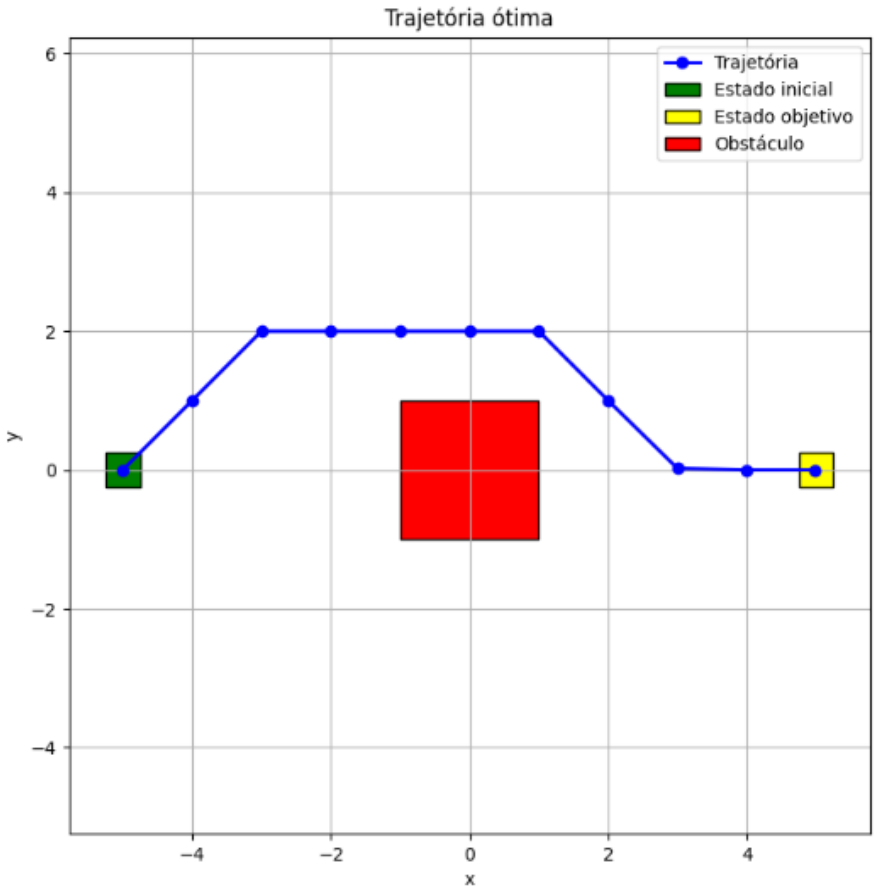


Figura 2: Trajetória ótima gerada pelo NMPC para um horizonte de predição $T = 10$.

seria suficiente para fazer com que o drone atingisse a posição objetivo exatamente no horizonte final de 10 passos em uma trajetória retilínea, caso não houvesse o elemento obstrutivo. Contudo, a trajetória final é fortemente ditada pelo compromisso entre as matrizes de ponderação $\mathbf{P}$ e $\mathbf{Q}$. Na presente simulação, o erro de posição é punido de forma agressiva ($P = 50$), enquanto a ação de controle possui menor peso relativo ($Q = 1$), conferindo ao *solver* liberdade para aplicar magnitudes elevadas de controle a fim de alcançar o estado alvo o mais rápido possível dentro do envelope seguro.

## 3.2 2º Cenário: Otimização em Horizonte Rolante e Dinâmica Estendida

A posteriori, adicionaram-se duas novas variáveis de estado ao modelo: as velocidades nos eixos das abscissas ($\dot{x}$) e das ordenadas ($\dot{y}$), totalizando um vetor de estado quadridimensional ($n_x = 4$). Além disso, implementou-se a estratégia de *Receding Horizon* (Horizonte Rolante) em malha fechada com o intuito de mitigar o efeito de forte antecipação observado no cenário anterior. Em paralelo a essa abordagem, incorporou-se um termo de custo terminal à função de custo, elaborado segundo o dispositivo da Seção 2.7, ponderado pela matriz $\mathbf{P}_{\text{mat}}$, adicionando uma camada adicional de penalização ao estado final da predição. Esse custo penaliza o desvio do estado no instante final do horizonte de predição ($k = k_{\text{pred}}$); assim, caso o objetivo esteja localizado logo após o término da janela de predição, o otimizador tende a reduzir os erros de posição e velocidade ao final do horizonte, favorecendo a aproximação ao objetivo e o cumprimento das condições de parada. A nova função de custo assume a seguinte forma, sujeita às mesmas restrições:

$$V(\mathbf{x}_{\text{init}}) = \min_{\mathbf{x}_k, \mathbf{u}_k} \left\{ \sum_{k=0}^{k_{\text{pred}}-1} \left( \bar{\mathbf{x}}_k^\top \mathbf{P} \bar{\mathbf{x}}_k + \mathbf{u}_k^\top \mathbf{Q} \mathbf{u}_k \right) + \bar{\mathbf{x}}_{k_{\text{pred}}}^\top \mathbf{P}_{\text{mat}} \bar{\mathbf{x}}_{k_{\text{pred}}} \right\}. \tag{17}$$

O Código 2 apresenta a implementação do algoritmo NMPC em malha fechada considerando a dinâmica de duplo integrador discretizada.

```
import gurobipy as gp
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import patches
import pandas as pd
import time

def checa(A, b, x_pos):
    return np.all(A @ x_pos <= b)

def MPC(x_init, k_pred):
    x_traj_real = np.zeros((T + 1, nx))
    u_traj_real = np.zeros((T, nu))
    x_traj_real[0, :] = x_init

    passos_executados = T
    telemetria_passos = []

    for t in range(T):
        x_atual = x_traj_real[t, :]

        # Criterio de parada ao atingir a meta
```

```
23          if np.linalg.norm(x_atual[0:2] - x_goal[0:2]) < 0.25 and
    np.linalg.norm(x_atual[2:4]) < 0.40:
24              passos_executados = t
25              x_traj_real = x_traj_real[:passos_executados + 1, :]
26              u_traj_real = u_traj_real[:passos_executados, :]
27              break
28
29          m = gp.Model(f"NMPC_Step_{t}")
30          m.setParam("OutputFlag", 0)
31          m.setParam("NonConvex", 2)
32
33          x = m.addVars(k_pred + 1, nx, lb=-10, ub=10, name="pos")
34          x_bar = m.addVars(k_pred + 1, nx, lb=-20, ub=20, name="
    x_bar")
35          u = m.addVars(k_pred, nu, lb=-1.5, ub=1.5, name="u")
36          lam = m.addVars(k_pred, nlam, lb=0, name='Lambda')
37          mu = m.addVars(k_pred, nu, lb=-gp.GRB.INFINITY, name='mu')
38
39          obj = (gp.quicksum(P * x_bar[k, i]**2 for k in range(1,
    k_pred + 1) for i in range(nx)) +
40                 gp.quicksum(P_mat[i,j] * x_bar[k_pred, i] * x_bar[
    k_pred, j] for i in range(nx) for j in range(nx)) +
41                 gp.quicksum(Q * u[k, i]**2 for k in range(k_pred)
    for i in range(nu)))
42          m.setObjective(obj, sense=gp.GRB.MINIMIZE)
43
44          # Condicao inicial do passo
45          for i in range(nx):
46              m.addConstr(x[0, i] == x_atual[i])
47
48          # Dinamica e restricoes de obstaculos
49          for k in range(k_pred):
50              m.addConstr(x[k + 1, 0] == x[k, 0] + x[k, 2] * dt)
51              m.addConstr(x[k + 1, 1] == x[k, 1] + x[k, 3] * dt)
52              m.addConstr(x[k + 1, 2] == x[k, 2] + u[k, 0] * dt)
53              m.addConstr(x[k + 1, 3] == x[k, 3] + u[k, 1] * dt)
54
55              m.addConstr(gp.quicksum(-lam[k, j] * b[j] for j in
    range(nlam)) +
56                          mu[k, 0] * x[k, 0] + mu[k, 1] * x[k, 1] >=
     1)
57
58              for i in range(nu):
59                  m.addConstr(gp.quicksum(lam[k, j] * A[j, i] for j
    in range(nlam)) - mu[k, i] == 0)
60
61              m.addQConstr(gp.quicksum(mu[k, i] * mu[k, i] for i in
    range(nu)) <= 1)
62
```

```
63          # Variavel de erro (tracking)
64          for k in range(k_pred + 1):
65              for i in range(nx):
66                  m.addConstr(x_bar[k, i] == x[k, i] - x_goal[i])
67
68          # Restricao terminal quando proximo do objetivo
69          distancia_atual = np.linalg.norm(x_atual[0:2] - x_goal
    [0:2])
70          if distancia_atual < 1.5:
71              m.addQConstr(x_bar[k_pred, 0]**2 + x_bar[k_pred, 1]**2
     <= 0.25**2)
72              m.addQConstr(x_bar[k_pred, 2]**2 + x_bar[k_pred, 3]**2
     <= 0.15**2)
73
74
75          m.optimize()
76
77          # Telemetria do Gurobi
78          tempo_solver = m.Runtime
79          nos_explorados = int(m.NodeCount)
80
81          try:
82              gap = m.MIPGap
83          except AttributeError:
84              gap = 0.0
85
86          telemetria_passos.append({
87              'Passo': t,
88              'Tempo_Solver_s': tempo_solver,
89              'Nos_Explorados': nos_explorados,
90              'Gap': gap
91          })
92
93          if m.status == gp.GRB.OPTIMAL:
94              u_aplicar = np.array([u[0, 0].X, u[0, 1].X])
95              u_traj_real[t, :] = u_aplicar
96
97              # Atualiza dinamica do sistema
98              x_traj_real[t + 1, 0] = x_traj_real[t, 0] +
    x_traj_real[t, 2] * dt
99              x_traj_real[t + 1, 1] = x_traj_real[t, 1] +
    x_traj_real[t, 3] * dt
100             x_traj_real[t + 1, 2] = x_traj_real[t, 2] + u_aplicar
    [0] * dt
101             x_traj_real[t + 1, 3] = x_traj_real[t, 3] + u_aplicar
    [1] * dt
102
103             # Salva variaveis para o proximo warm-start
104             prev_x = np.array([[x[k, i].X for i in range(nx)] for
```

```
    k in range(k_pred + 1)])
105             prev_x_bar = np.array([[x_bar[k, i].X for i in range(
    nx)] for k in range(k_pred + 1)])
106             prev_u = np.array([[u[k, i].X for i in range(nu)] for
    k in range(k_pred)])
107             prev_lam = np.array([[lam[k, j].X for j in range(nlam)
    ] for k in range(k_pred)])
108             prev_mu = np.array([[mu[k, i].X for i in range(nu)]
    for k in range(k_pred)])
109         else:
110             print(f"Falhou para k_pred={k_pred} no passo {t}")
111             return None, None, t, pd.DataFrame(telemetria_passos)
112 
113     df_telemetria = pd.DataFrame(telemetria_passos)
114     return x_traj_real, u_traj_real, passos_executados,
    df_telemetria
115 
116 # --- Parametros ---
117 T = 25
118 dt = 0.5
119 nx = 4
120 nu = 2
121 nlam = 4
122 
123 P = 10
124 Q = 10
125 
126 P_mat = np.array([
127     [45.7230,  0.0000, 30.8341,  0.0000],
128     [ 0.0000, 45.7230,  0.0000, 30.8341],
129     [30.8341,  0.0000, 55.0745,  0.0000],
130     [ 0.0000, 30.8341,  0.0000, 55.0745]
131 ])
132 
133 A = np.array([[1, 0], [-1, 0], [0, 1], [0, -1]])
134 b = np.array([1, 1, 1, 1])
135 
136 x_init = np.array([-5.0, 0.0, 0.0, 0.0])
137 x_goal = np.array([5.0, 0.0, 0.0, 0.0])
138 
139 # Execucao da simulacao e exportacao para CSV
140 lista_k_pred = [3,5,7,10,12]
141 trajetorias = {}
142 
143 for kp in lista_k_pred:
144     print(f"Simulando NMPC para k_pred = {kp}...")
145     estados, acoes, passos, df_telemetria = MPC(x_init, k_pred=kp)
146 
147     if estados is not None:
```

```
148         trajetorias[kp] = (estados, acoes)
149
150         # --- EXPORTACAO TELEMETRIA PARA CSV ---
151         nome_arquivo = f"K_pred_{kp}_warm-start.csv"
152         df_telemetria.to_csv(nome_arquivo, index=False)
153         print(f"\nTelemetria salva em: {nome_arquivo}")
154
155         # Impressao no terminal
156         print("\n--- Resumo de Telemetria do Gurobi ---")
157         print(df_telemetria.to_string(index=False))
158         print(f"\nTempo Total Solver: {df_telemetria['
    Tempo_Solver_s'].sum():.4f} s")
159         print(f"Total de Nos Explorados: {df_telemetria['
    Nos_Explorados'].sum()}")
```

Listing 2: Algoritmo NMPC em horizonte rolante com estado 4D em Python

Com a inclusão das velocidades no vetor de estado, verifica-se uma suavização significativa na trajetória percorrida. Uma vez que as ações de controle atuam diretamente nas acelerações, o drone exibe um perfil dinâmico fisicamente coerente: acelera gradativamente em direção ao objetivo até atingir a velocidade máxima permitida e reduz a velocidade conforme se aproxima da posição alvo, garantindo uma parada estável.

Para avaliar o impacto do tamanho do horizonte de predição na resposta do sistema e no tempo de processamento, foram realizadas simulações comparativas configurando dois tamanhos distintos de horizonte: $k_{\text{pred1}} = 5$ e $k_{\text{pred2}} = 7$, representados nas Figuras 3 e 4, respectivamente.

Na simulação com $k_{\text{pred}} = 5$ (Figura 3), nota-se um comportamento míope por parte do controlador: o drone inicia sua trajetória deslocando-se em linha reta em direção ao objetivo e reage à região de restrição apenas quando esta passa a interceptar a janela do horizonte de predição. Essa reação tardia exige correções mais severas na trajetória ao se aproximar da barreira.

Por outro lado, ao estender o horizonte para $k_{\text{pred}} = 7$ (Figura 4), o veículo passa a "enxergar" a presença do elemento obstrutivo com antecedência suficiente para planejar uma manobra de contorno mais suave e progressiva. Diferente do 1º Cenário (no qual a otimização em malha aberta com $T = 10$ gerava um custo computacional desnecessariamente elevado para todo o percurso), a abordagem em horizonte rolante com $k_{\text{pred}} = 7$ atinge um compromisso ideal: reduz substancialmente o esforço computacional a cada passo de amostragem sem comprometer a suavidade do desvio e a estabilidade na chegada ao estado objetivo.

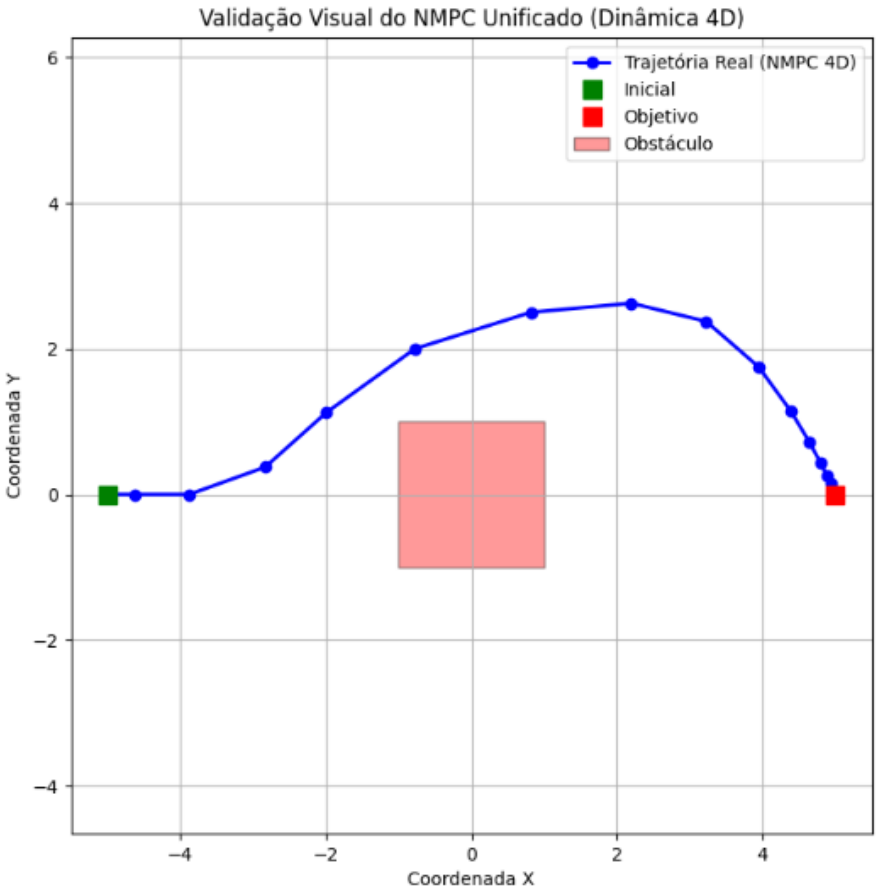


Figura 3: Trajetória executada em malha fechada considerando um horizonte de predição curto ($k_{\text{pred}} = 5$).

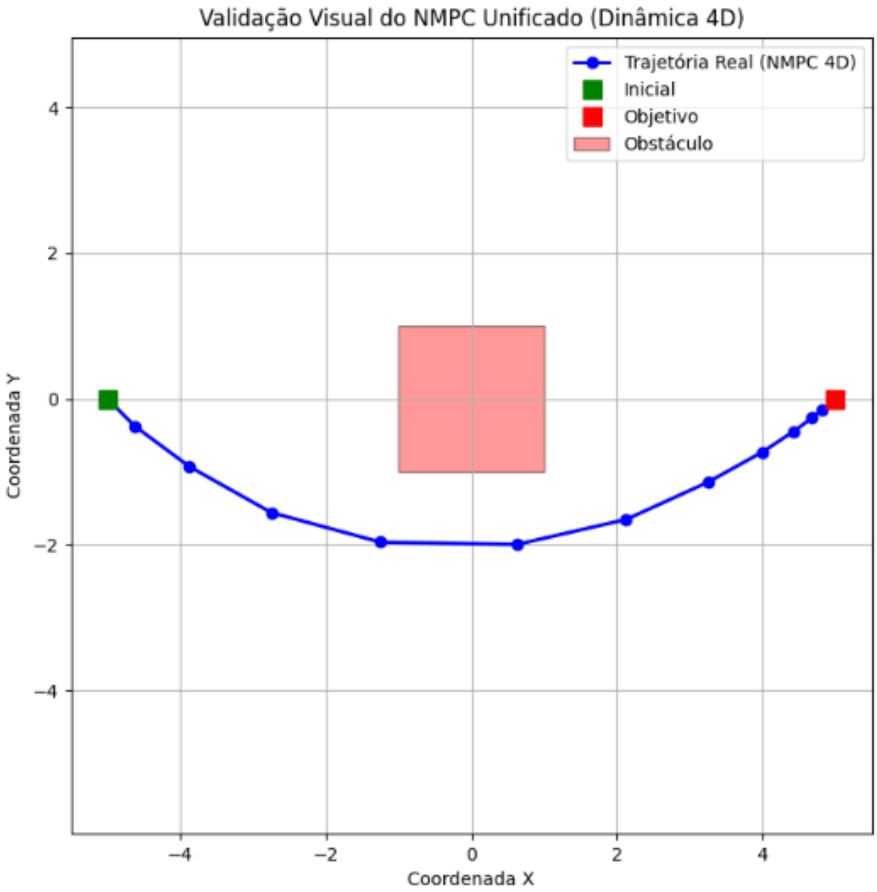


Figura 4: Trajetória executada em malha fechada considerando um horizonte de predição estendido ($k_{\text{pred}} = 7$).

### Análise e Comparação do Tempo de Execução

A seguir, apresenta-se uma medição mais acurada a respeito do impacto computacional decorrente do tamanho do horizonte de predição. Para tal, além dos horizontes explorados anteriormente, adicionaram-se as configurações $k_{\text{pred}} \in \{3, 10, 12\}$.

Na Figura 5, avaliam-se o tempo médio por passo de amostragem e o tempo total de otimização para cada horizonte. Como esperado, quanto maior a janela de predição, maior é a complexidade do problema e o tempo exigido pelo *solver*, alcançando médias superiores a 1,3 segundos por passo no caso $k_{\text{pred}} = 10$. Por outro lado, um horizonte demasiadamente curto ($k_{\text{pred}} = 3$) inviabilizou a convergência do problema, fazendo com que o *solver* classificasse a otimização como infactível devido à impossibilidade de antecipar o desvio a tempo.

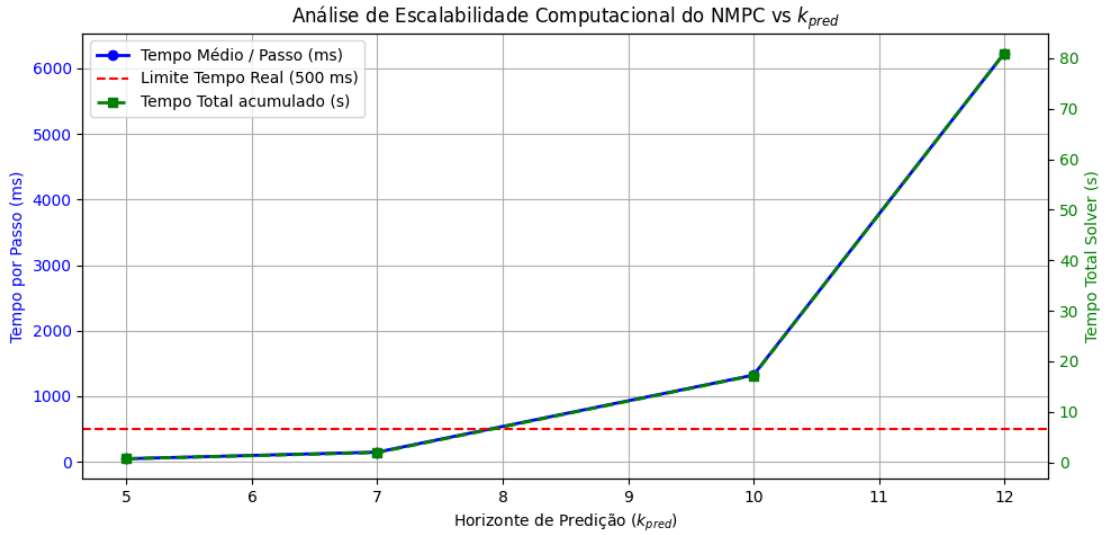


Figura 5: Comparativo do tempo médio por passo e tempo total do *solver* para diferentes tamanhos de horizonte.

A Figura 6 ilustra que a expansão excessiva do horizonte de predição além de $k_{\text{pred}} = 7$ traz benefícios irrisórios no desvio do obstáculo. Contudo, o aumento acentuado no tempo de cálculo compromete o desempenho em tempo real, o que pode acarretar perda de segurança em aplicações práticas.

A Tabela 1 sintetiza os tempos de execução, bem como a quantidade de passos necessária para alcançar o objetivo. Nota-se que, embora o horizonte $k_{\text{pred}} = 5$ apresente um tempo de computação por passo significativamente menor, ele exige mais iterações (15 passos) para concluir o trajeto. Nesse sentido, a escolha entre $k_{\text{pred}} = 5$ e $k_{\text{pred}} = 7$ depende da perspectiva adotada para a análise do controlador. Sob a ótica do custo computacional, $k_{\text{pred}} = 5$ apresenta uma vantagem

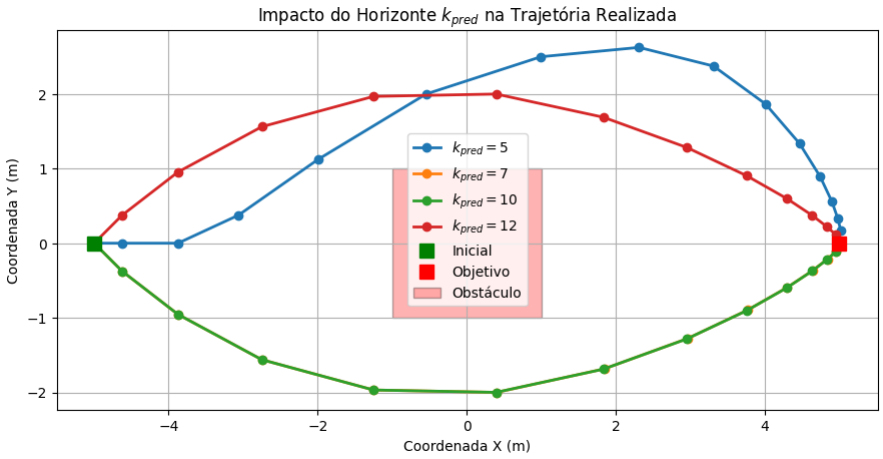


Figura 6: Trajetórias resultantes em malha fechada para múltiplos horizontes de predição.

significativa, com tempo de processamento aproximadamente três vezes menor por passo. Por outro lado, sob a ótica do desempenho da trajetória, $k_{\text{pred}} = 7$ mostra-se mais vantajoso, pois atinge o objetivo em menos passos (13 passos) com uma resposta dinâmica mais fluida, evitando atuações bruscas e desgastes desnecessários nos atuadores, além de fornecer um sinal de controle mais suave e seguro para o treinamento de um modelo em tempo real (como uma rede neural).

Tabela 1: Análise do Custo Computacional e Desempenho do NMPC para Diferentes Horizontes de Predição.

| **Horizonte ($k_{\text{pred}}$)** | **Tempo Médio/Passo (ms)** | **Tempo Total (s)** | **Passos Executados** |
|---|---|---|---|
| 5 | 47,66 | 0,715 | 15 |
| 7 | 148,69 | 1,933 | 13 |
| 10 | 1322,95 | 17,198 | 13 |
| 12 | 6229,58 | 80,985 | 13 |

## 3.3 3º Cenário: Implementação de Warm-start

O livro **Predictive Control with Constraints** [12] aborda o tema de *warm-start* (inicialização quente) no contexto da solução de problemas de otimização em tempo real, em que a velocidade de convergência torna-se crucial para a aplicabilidade do controlador. A técnica de *warm-start* baseia-se no conceito discutido na Seção 2.7: assumindo condições de baixo ruído e pequenas perturbações, a sequência ótima de controle calculada no instante $k$ estará muito próxima daquela requerida no

instante $k+1$. Assim, em vez de reiniciar o algoritmo de otimização a partir de um ponto genérico ou nulo (*cold-start*), desloca-se a solução anterior no tempo para usá-la como estimativa inicial do ciclo subsequente, o que tende a acelerar a convergência do solver.

A Tabela 2 sintetiza como a implementação do *warm-start* afetou o tempo de computação para os mesmos horizontes analisados anteriormente.

Tabela 2: Análise do Custo Computacional e Desempenho do NMPC com Aplicação de Warm-start.

| **Horizonte ($k_{\text{pred}}$)** | **Tempo Médio/Passo (ms)** | **Tempo Total (s)** | **Passos Executados** |
|---|---|---|---|
| 5 | 41,87 | 0,6280 | 15 |
| 7 | 132,70 | 1,7256 | 13 |
| 10 | 1327,30 | 17,2543 | 13 |
| 12 | 6553,90 | 85,2003 | 13 |

Nota-se que, para os horizontes menores ($k_{\text{pred}} = 5$ e 7), houve uma melhora notável tanto no tempo total de otimização quanto no tempo médio por passo (reduções de aproximadamente 12,2% e 10,8%, respectivamente). Contudo, para os horizontes mais longos ($k_{\text{pred}} = 10$ e 12), observou-se um comportamento contraintuitivo: a inclusão do *warm-start* acarretou um aumento no tempo computacional exigido pelo algoritmo.

Esse fenômeno, embora à primeira vista inesperado, é amplamente documentado na literatura de otimização não linear e controle preditivo não linear (NMPC) [11, 10]. A eficácia do *warm-start* em reduzir o tempo de solução depende criticamente de o ponto inicial fornecido pertencer à bacia de atração do ótimo local correto ou estar suficientemente próximo de uma trajetória factível. Três causas principais explicam a degradação do tempo computacional para horizontes mais longos:

1. **Não Convexidade e pontos de sela/mínimos locais:** Diferente do MPC linear (cuja otimização resulta em um programa quadrático convexo), o NMPC lida com superfícies de custo altamente não convexas e com multiplicidade de mínimos locais [11]. À medida que o horizonte de predição $k_{\text{pred}}$ aumenta, o espaço de busca cresce exponencialmente em dimensão e a topologia da função de custo torna-se significativamente mais complexa. O deslocamento simples do horizonte (*shift*) pode fornecer um palpite inicial que cai próximo de uma região de má condicionabilidade (ex.: ponto de sela) ou em uma bacia de atração local desfavorável, exigindo que métodos baseados em Gradientes Conjugados ou Programação Quadrática Sucessiva (SQP) executem passos de *line-search* ou *trust-region* muito conservadores para garantir a descida, aumentando o número de iterações do *solver* [10].

2. **Incompatibilidade na Ponta do Horizonte (Tail Initialization):** Na estratégia padrão de *warm-start*, os estados e entradas de $k+1$ a $k+k_{\text{pred}}-1$ são reaproveitados da otimização anterior, enquanto o último elemento do horizonte ($k+k_{\text{pred}}$) precisa ser extrapolado (geralmente replicando o último valor ou utilizando uma lei de controle local pré-definida) [12]. Em horizontes longos sobre sistemas não lineares rígidos ou instáveis, pequenos erros acumulados na extremidade final da trajetória extrapolada geram grande infactibilidade nas restrições dinâmicas do modelo, forçando os métodos de Ponto Interior (IPM) ou SQP a dispender um elevado esforço computacional para restaurar a viabilidade das restrições antes de otimizar a função objetivo.

3. **Efeito de Marcas Ativas (*Active-Set Changes*):** Em solvers de Otimização Não Linear baseados em conjuntos ativos ou pontos interiores, a estimativa do *warm-start* inclui também os multiplicadores de Lagrange associados às restrições do sistema [10]. Se a mudança de estado entre o instante $k$ e $k+1$ provocar a alteração do conjunto de restrições ativas (por exemplo, um atuador atingindo um limite de saturação que não estava ativo antes), a estimativa prévia dos multiplicadores torna-se incoerente. Para horizontes longos, a probabilidade de ocorrer variação na sequência de restrições ativas ao longo da trajetória é substancialmente maior, o que exige reinicializações internas da matriz Hessiana/Jacobiana pelo solver.

Portanto, conclui-se que o *warm-start* é extremamente benéfico para manter a operabilidade em tempo real de controladores com horizontes moderados, mas sua implementação em horizontes longos requer estratégias avançadas de inicialização da extremidade do horizonte (*tail-strategy*) ou regularização do solver para evitar o aprisionamento em regiões computacionalmente custosas.

## 3.4 4º Cenário: Fase Não Mínima

Diz-se que um modelo possui comportamento de fase não mínima quando, ao aplicar um degrau para seguimento de referência, a resposta do sistema vai inicialmente na direção contrária à desejada (fenômeno conhecido como resposta inversa).

Sob essa óptica, o objetivo desta quarta etapa de simulação é observar o comportamento do drone quando este inicia demasiadamente próximo ao poliedro $\mathscr{O}$. Diferente dos primeiros cenários, nos quais o drone sempre atua de forma a diminuir a distância entre ele e o objetivo, aqui o veículo precisa inicialmente afastar-se para conseguir contorná-lo, caracterizando um comportamento análogo ao de fase não mínima.

Em estratégias de controle preditivo, tradicionalmente formulam-se restrições rígidas (*hard constraints*) para o sistema. Matematicamente, isso significa exigir que o estado do sistema pertença obrigatoriamente ao conjunto factível $\mathscr{X}$.

Embora esse tipo de restrição garanta, teoricamente, a segurança do sistema, ele apresenta uma vulnerabilidade prática: a factibilidade do problema de otimização. Se o drone for atingido por uma perturbação não planejada ou iniciar excessivamente próximo ao obstáculo, a dinâmica do sistema combinada com os limites de aceleração impostos podem impossibilitar o encontro de uma sequência ótima de ações $\mathbf{u}_k$, fazendo com que o *solver* classifique o problema como infactível.

Para contornar essa limitação, adota-se a técnica consagrada por Maciejowski [12] e Borrelli et al. [13], na qual as restrições rígidas são convertidas em restrições suaves (*soft constraints*) por meio da introdução de uma variável de folga (*slack variable*) $s_k \geq 0$. A variável de folga atua como um "amortecimento", permitindo que o sistema viole ligeiramente a restrição rígida apenas quando estritamente necessário. Como demonstrado experimentalmente por Zhang et al. [20] em problemas de navegação de veículos autônomos, essa estratégia evita a inviabilidade computacional do *solver* em tempo de execução na malha fechada.

Para o modelo desenvolvido neste projeto, baseado na dualidade forte e nas condições de KKT formuladas por Zago et al. [7], a restrição rígida de não-interseção para que o veículo permaneça estritamente fora do polítopo $\mathscr{O} = \{\mathbf{x} \in \mathbb{R}^2 \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ é originalmente expressa por:

$$\sum_{j=1}^{n_\lambda} (-\lambda_{k,j} \cdot b_j) + \boldsymbol{\mu}_k^\top \mathbf{x}_k \geq 1.$$

Seguindo os preceitos de flexibilização de restrições, insere-se uma variável de folga não negativa $s_k \geq 0$ para flexibilizar a fronteira em cenários críticos sem alterar a estrutura dual do problema. A restrição suave passa a ser formulada como:

$$\sum_{j=1}^{n_\lambda} (-\lambda_{k,j} \cdot b_j) + \boldsymbol{\mu}_k^\top \mathbf{x}_k + s_k \geq 1, \quad \text{com } s_k \geq 0. \tag{19}$$

- Se $s_k = 0$: a trajetória do drone respeita integralmente a margem de segurança.
- Se $s_k > 0$: o otimizador absorve a impossibilidade física momentânea através de $s_k$, mantendo o problema matematicamente factível.

### Modificação da Função Objetivo e Penalização

Para garantir que o otimizador utilize $s_k > 0$ exclusivamente em situações de extrema necessidade (quando o problema seria, de outro modo, inviável), insere-se um termo de penalização na função objetivo do NMPC:

$$V(\mathbf{x}_{\text{init}}) = \min_{\mathbf{x}_k, \mathbf{u}_k, s_k} \left\{ \sum_{k=0}^{k_{\text{pred}}-1} \left( \bar{\mathbf{x}}_k^\top \mathbf{P} \bar{\mathbf{x}}_k + \mathbf{u}_k^\top \mathbf{Q} \mathbf{u}_k \right) + \sum_{k=1}^{k_{\text{pred}}} W_{\text{slack}} s_k^2 + \bar{\mathbf{x}}_{k_{\text{pred}}}^\top \mathbf{P}_{\text{term}} \bar{\mathbf{x}}_{k_{\text{pred}}} \right\}, \tag{20}$$

em que $W_{\text{slack}} \in \mathbb{R}^+$ representa um peso escalar positivo elevado (por exemplo, $W_{\text{slack}} \approx 10^5$).

Neste cenário de simulação, as matrizes de ponderação são definidas como $\mathbf{P} = \mathbf{I}$ e $\mathbf{Q} = \mathbf{I}$ (diferente da sintonia $\mathbf{P} = \mathbf{Q} = 10 \cdot \mathbf{I}$ empregada anteriormente). Essa alteração maximiza a razão relativa entre a penalização de folga e o custo de estado ($W_{\text{slack}}/\|\mathbf{P}\| = 10^5$), garantindo que a variável de folga atue puramente como uma salvaguarda de emergência diante de cenários limítrofes.

### O Papel do Peso $W_{\text{slack}}$

1. **Prioridade Absoluta à Evasão:** Como $W_{\text{slack}} \gg P, Q$, o custo associado a utilizar $s_k > 0$ é ordens de grandeza superior ao custo de desviar da rota de referência ou aplicar esforço de controle máximo. Consequentemente, o *solver* não utilizará a folga para atalhar a trajetória.

2. **Penalização Quadrática ($s_k^2$):** A formulação quadrática penaliza desvios elevados de forma severa, promovendo o retorno imediato do sistema à região estritamente segura ($s_k = 0$) no menor número possível de passos de amostragem.

### Resultados e Implementação

As Figuras 7 e 8 apresentam os resultados da abordagem proposta para as condições iniciais $\mathbf{x}_{\text{init},1} = (-2{,}0; 0{,}0)$ e $\mathbf{x}_{\text{init},2} = (-1{,}2; 0{,}0)$, respectivamente.

Observa-se, em ambos os casos, o comportamento de afastamento inicial do drone em relação ao objetivo, confirmando a dinâmica análoga ao comportamento de fase não mínima discutida nesta seção.

Durante a resolução do problema de otimização em tempo real, a variável de folga $s_0$ foi acionada apenas quando a restrição de segurança se tornou fisicamente inalcançável de forma rígida. Os momentos e magnitudes de violação da restrição suave são detalhados na Tabela 3.

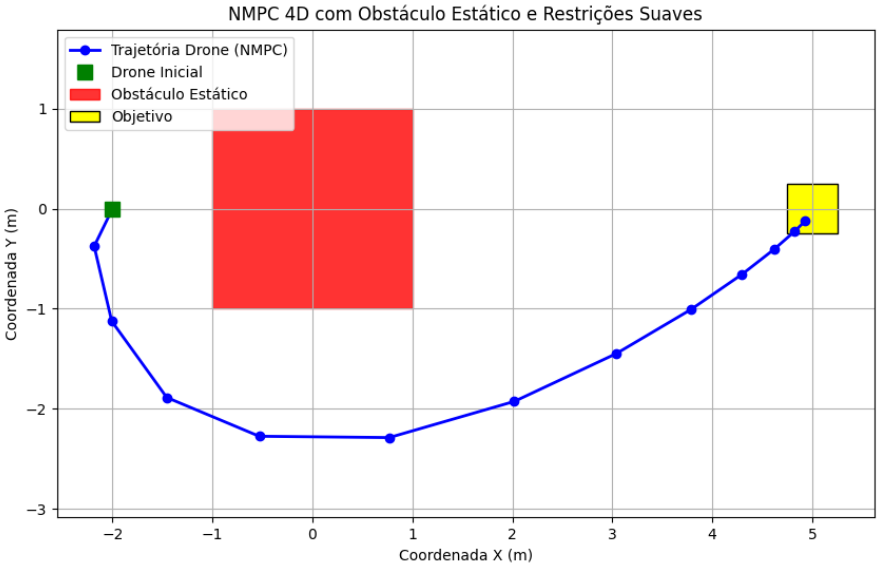


Figura 7: Trajetória do drone para a condição inicial $\mathbf{x}_{\text{init},1} = (-2{,}0; 0{,}0)$.

Tabela 3: Ativação da variável de folga ($s_0$) ao longo do horizonte de controle.

| **Condição Inicial** | **Passo de Tempo ($t$)** | **Slack ($s_0$)** | **Status de Convergência** |
|---|---|---|---|
| $\mathbf{x}_{\text{init},1} = (-2{,}0;\ 0{,}0)$ | $t = 4$ | 0,0010 | Alvo alcançado em $t = 13$ |
| $\mathbf{x}_{\text{init},2} = (-1{,}2;\ 0{,}0)$ | $t = 0$ | 0,8000 | Alvo alcançado em $t = 14$ |
| | $t = 1$ | 0,8000 | |
| | $t = 2$ | 0,4250 | |
| | $t = 3$ | 0,0059 | |

Nota-se que o otimizador não recorreu ao uso da variável de folga de forma irrestrita. O valor elevado atribuído ao peso de penalização $W_{\text{slack}}$ garantiu que a folga fosse utilizada exclusivamente para absorver a inviabilidade pontual. Além disso, a penalização quadrática atuou de maneira efetiva, reduzindo severamente os desvios de maior magnitude e reconduzindo o sistema à zona estritamente segura em poucos passos de amostragem.

### Análise de Desempenho e Degradação Computacional

Os resultados demonstram que a inclusão das *soft constraints* cumpriu seu papel de salvaguarda numérica, impedindo que o *solver* classificasse o problema como infactível diante de condições iniciais críticas. Contudo, a análise temporal de convergência revela um custo computacional significativo associado a essa flexibi-

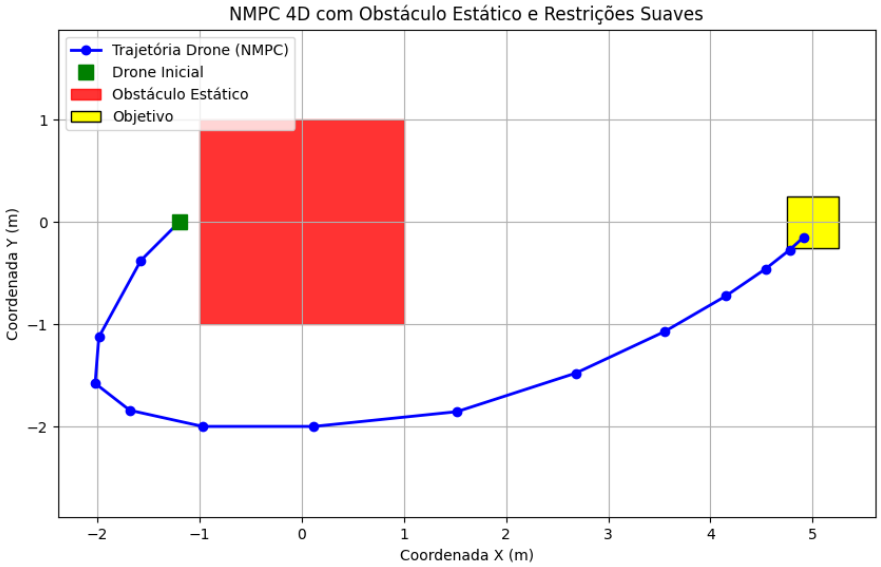


Figura 8: Trajetória do drone para a condição inicial $\mathbf{x}_{\text{init},2} = (-1{,}2; 0{,}0)$.

lização.

Para a condição inicial $\mathbf{x}_{\text{init},1} = (-2{,}0;\ 0{,}0)$ — na qual o sistema é fisicamente capaz de contornar o obstáculo utilizando restrições rígidas —, a inclusão do *slack* elevou o tempo total de convergência da simulação de 4,0150 s (formulação rígida) para 6,0181 s (formulação suave), representando um acréscimo computacional superior a 49,8%. Fenômeno análogo foi registrado para a posição inicial $\mathbf{x}_{\text{init}} = (-5{,}0;\ 0{,}0)$, cujo tempo aumentou de 0,7467 s para 1,0706 s (uma degradação de 43,3%).

Esse overhead computacional ocorre por dois fatores principais:

1. **Aumento de Dimensionalidade:** A adição das variáveis de folga $s_k$ expande o vetor de decisão do problema de Otimização Quadrática Não Convexa (NQP), aumentando o espaço de busca a ser percorrido.

2. **Mau Condicionamento Numérico:** O peso elevado $W_{\text{slack}} = 10^5$ insere uma grande disparidade de escala na matriz Hessiana da função objetivo em relação aos pesos $P$ e $Q$. Essa rigidez numérica exige iterações adicionais do algoritmo de pontos interiores do *Gurobi* a cada passo de amostragem.

Na condição limítrofe $\mathbf{x}_{\text{init},1} = (-2{,}0;\ 0{,}0)$, o *solver* registrou uma ativação residual de $s_0 = 0{,}0010$ no passo $t = 4$. Essa microflexibilização ocorre por margens superficiais de otimização na fronteira de decisão, redundando em perda de desempenho temporal em um regime no qual a trajetória rígida já era totalmente viável.

Em suma, embora as variáveis de folga sejam indispensáveis como mecanismo de contingência para evitar a falha da malha em cenários severos (como $\mathbf{x}_{\text{init},2} = (-1{,}2;\ 0{,}0)$), seu uso contínuo no regime nominal compromete o desempenho em tempo real do NMPC.

**Estratégias de Mitigação**

Para evitar a degradação computacional no regime nominal sem abdicar da segurança do sistema, duas abordagens podem ser adotadas em implementações embarcadas:

1. **Ativação Híbrida/Condicional:** Manter a formulação rígida (*hard constraints*) como padrão de operação rápida e ativar o módulo com *slack variables* exclusivamente como salvaguarda quando o *solver* retornar um *status* de infactibilidade (*Infeasible*).

2. **Penalização com Zona Morta (*Deadzone*):** Substituir ou combinar a penalização quadrática por uma função de penalidade exata $L_1$ dotada de uma zona morta ($\epsilon > 0$), impedindo que resíduos numéricos infinitesimais ativem o gradiente de busca quando o veículo estiver fora da zona de perigo do obstáculo.

# 4 Conclusão

Este trabalho apresentou o projeto, a modelagem e a simulação de estratégias de Controle Preditivo Não Linear (NMPC) aplicadas ao rastreamento de trajetória e desvio de obstáculos para um drone. A formulação matemática fundamentou-se no uso da dualidade de Lagrange para converter um problema de otimização de dois níveis em um NLP não convexo explícito, contornando a alta complexidade de otimização aninhada.

A partir da transição de um modelo atômico em malha aberta para uma formulação em malha fechada sob a estratégia de horizonte rolante (*receding horizon*) com dinâmica estendida (4D), observou-se uma significativa melhoria no perfil de navegação do drone. A inclusão das velocidades no vetor de estado conferiu suavidade às ações de controle, enquanto o horizonte rolante estabeleceu um compromisso ideal entre a capacidade de antecipação do veículo e o custo computacional por passo de amostragem.

Além disso, a análise do cenário com proximidade crítica ao obstáculo (fase não mínima) evidenciou as limitações práticas das restrições rígidas (*hard constraints*) diante de problemas de infactibilidade computacional. A implementação de restrições suaves (*soft constraints*) com variáveis de folga (*slack variables*) e alta

penalização quadrática mostrou-se eficaz, garantindo a viabilidade do solucionador Gurobi em condições severas sem comprometer a segurança da operação.

Os dados obtidos ao longo das simulações consolidaram uma base sólida e fisicamente coerente, viabilizando o treinamento futuro de redes neurais (como arquiteturas com função de ativação ReLU) para o aprendizado da lei de controle do NMPC com baixo custo computacional.

Como trabalhos futuros, sugere-se a expansão das seguintes frentes de pesquisa:

1. O projeto, a implementação e o teste de metodologias de programação neurodinâmica e aprendizado por reforço (*Reinforcement Learning*) para a síntese de políticas de controle ótimo;

2. A extensão do modelo para ambientes tridimensionais ($\mathbb{R}^3$) com múltiplos obstáculos dinâmicos;

3. A incorporação de modelos aerodinâmicos mais complexos, considerando efeitos de escorregamento e perturbações externas de vento em tempo de execução.

# Referências

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[2] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2520–2525, IEEE, 2011.

[3] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear mpc for collision avoidance and control of uavs with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6001–6008, 2020.

[4] E. F. Camacho and C. B. Alba, *Model Predictive Control*. London, UK: Springer Science & Business Media, 2nd ed., 2013.

[5] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2451–2456, IEEE, 2004.

[6] M. Bangura and R. Mahony, "Real-time nonlinear model predictive control for quadrotor uavs," in *In Control Conference (ECC), 2014 European*, pp. 1173–1178, IEEE, 2014.

[7] J. G. Zago, G. Notarstefano, and E. Camponogara, "A guided retraining strategy for safe ReLU neural network controllers of linear systems," in *IEEE Conference on Decision and Control (CDC)*, pp. 1–6, 2023. Artigo de conferência.

[8] E. Camponogara, H. Scherer, L. T. Biegler, and I. E. Grossmann, "Hierarchical decompositions for MPC of resource constrained control systems: applications to building energy management," *Optimization and Engineering*, vol. 22, no. 1, pp. 187–215, 2021.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[10] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2nd ed., 2006.

[11] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Madison, WI, USA: Nob Hill Publishing, 2nd ed., 2017.

[12] J. M. Maciejowski, *Predictive Control with Constraints*. Harlow, UK: Pearson Education / Prentice Hall, 2002.

[13] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge, UK: Cambridge University Press, 2017.

[14] S. S. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback laws for general non-linear systems including optimal control, energy management and constrained problems," *Journal of Optimization Theory and Applications*, vol. 57, no. 2, pp. 265–293, 1988.

[15] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[16] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for sequence of quadratic programming problems," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[17] A. Laub, "A schur technique for solving algebraic riccati equations," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 913–921, 1979.

[18] G. H. Golub and C. F. Van Loan, *Matrix Computations*. JHU Press, 2013.

[19] A. Ben-Israel and T. T. Greville, *Generalized Inverses: Theory and Applications*. John Wiley & Sons, 1974.

[20] X. Zhang, A. Lin, and F. Borrelli, "Optimization-based autonomous racing of 1/10th-scale rc cars," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2080–2093, 2017.