Thiago Bassinello Burghi

Estudo do Problema de Rastreamento de Trajetórias de um Robô Móvel sujeito a Deslizamentos através da Teoria de Lyapunov para Sistemas Perturbados e Decomposições em Soma de Quadrados

90/2015

CAMPINAS

2015

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Thiago Bassinello Burghi

Estudo do Problema de Rastreamento de Trajetórias de um Robô Móvel sujeito a Deslizamentos através da Teoria de Lyapunov para Sistemas Perturbados e Decomposições em Soma de Quadrados

Dissertação de Mestrado apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Mecânica, na Área de Mecânica dos Sólidos e Projeto Mecânico.

Orientador: Prof. Dr. Juan Francisco Camino dos Santos

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO THIAGO BASSINELLO BURGHI, E ORIENTADA PELO PROF. DR. JUAN FRANCISCO CAMINO DOS SANTOS.

.....

ASSINATURA DO ORIENTADOR

CAMPINAS
2015

Agência de fomento: Capes
Nº processo: 33003017

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

B913e

Burghi, Thiago Bassinello, 1989-
Estudo do problema de rastreamento de trajetórias de um robô móvel sujeito a deslizamentos através da teoria de Lyapunov para sistemas perturbados e decomposições em soma de quadrados / Thiago Bassinello Burghi. – Campinas, SP: [s.n.], 2015.

Orientador: Juan Francisco Camino.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Navegação de robôs móveis. 2. Sistemas não lineares. 3. Teoria de controle. 4. Deslizamento. 5. Otimização não-linear. I. Camino, Juan Francisco, 1970-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Informações para Biblioteca Digital

Título em Inglês: Study of the trajectory tracking problem of a mobile robot subject to slip through Lyapunov theory for perturbed systems and sums of squares decompositions
Palavras-chave em Inglês:
Mobile robots
Nonlinear systems
Control theory
Skidding
Nonlinear programming
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca Examinadora:
Juan Francisco Camino [Orientador]
André Ricardo Fioravanti
Alim Pedro de Castro Gonçalves
Data da defesa: 27-08-2015
Programa de Pós Graduação: Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE SISTEMAS INTEGRADOS

DISSERTAÇÃO DE MESTRADO ACADÊMICO

Estudo do Problema de Rastreamento de Trajetórias de um Robô Móvel sujeito a Deslizamentos através da Teoria de Lyapunov para Sistemas Perturbados e Decomposições em Soma de Quadrados

Autor: Thiago Bassinello Burghi

Orientador: Prof. Dr. Juan Francisco Camino dos Santos

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. Juan Francisco Camino dos Santos, Presidente
DSI/FEM/UNICAMP

Prof. Dr. André Ricardo Fioravanti
DMC/FEM/UNICAMP

Prof. Dr. Alim Pedro de Castro Gonçalves
DCA/FEEC/UNICAMP

Campinas, 27 de Agosto de 2015.

Este trabalho é dedicado a meus pais, Nilton e Cláudia.

Agradecimentos

Agradeço primeiramente aos meus pais, Nilton e Cláudia, que me incentivaram a seguir o caminho da pesquisa.

Agradeço ao Professor Camino, meu orientador, que me ensinou os valores da precisão e do rigor.

Agradeço a Juliano Iossaqui pelo apoio durante o início de meu mestrado.

Agradeço aos meus colegas de laboratório, Ricardo, Luis, Guilherme e Marcos, que contribuíram para este trabalho em diversos momentos.

Agradeço a toda a comunidade da Faculdade de Engenharia Mecânica da Unicamp, em especial aos docentes, pelo aprendizado, e aos funcionários, pelo auxílio.

Por fim, agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro.

Resumo

O controle do movimento de robôs móveis em altas velocidades e sob condições adversas do solo é um problema difícil, pois as rodas do robô podem estar sujeitas a diferentes tipos de deslizamentos. O deslizamento lateral é um problema particularmente complicado quando se lida com robôs móveis de tração diferencial, já que suas rodas não podem produzir movimento na direção lateral. Este trabalho apresenta uma aplicação da Teoria de Lyapunov para sistemas não lineares perturbados. A principal aplicação é analisar a estabilidade de um robô móvel de tração diferencial controlado por uma lei adaptativa não linear cinemática capaz de estimar o deslizamento longitudinal. A abordagem proposta pode ser vista como um método de análise de controladores cinemáticos de robôs móveis sujeitos a deslizamentos laterais e longitudinais. É mostrado que sob condições apropriadas, as soluções da dinâmica do erro de postura do robô são uniformemente finalmente limitadas. Simulações numéricas são apresentadas para ilustrar esse exemplo. Para tratar um problema de otimização que surge durante a análise de estabilidade, técnicas de decomposição em soma de quadrados (SOS) para otimização polinomial são apresentadas e aplicadas. Um estudo minucioso dos recursos computacionais necessários para a resolução de problemas de decomposição SOS também é apresentado.

Palavras-chave: Robôs Móveis, Deslizamento Lateral, Rastreamento de Trajetórias, Sistemas Não Lineares, Decomposições SOS, Otimização Polinomial.

Abstract

The motion control of mobile robots at high speeds and under adverse ground conditions is a difficult problem because the robots' wheels may be subject to different kinds of slip. Lateral slip is a particularly complicated problem for differential drive mobile robots to deal with, since their wheels cannot directly produce movement in the lateral direction. This work presents an application of some results from Lyapunov Theory for nonlinear perturbed systems. The main application is the stability analysis of a differential drive mobile robot when it is controlled by an adaptive nonlinear kinematic controller capable of estimating longitudinal slip. The proposed approach can be seen as a method for analyzing kinematic controllers of mobile robots subject to longitudinal and lateral slip. It is shown that under the appropriate conditions, the solutions of the robot's posture error dynamics are uniformly ultimately bounded. Numerical simulations are presented to illustrate this example. To tackle an optimization problem which arises during the stability analysis, SOS techniques for polynomial optimization are presented and applied. A thorough study of the computational resources required for solving SOS problems is also presented.

Keywords: Mobile robots, Lateral Slip, Trajectory Tracking, Nonlinear Systems, SOS Decompositions, Polynomial Optization.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CA:	Controlador Adaptativo
EDO:	Equação Diferencial Ordinária
Gb:	Gigabyte
Mb:	Megabyte
MOM:	Momentos de Matrizes
NA:	Controlador não Adaptativo
NASA:	*National Aeronautics and Space Administration* (Agência espacial americana)
RAM:	*Random Access Memory* (Memória de Acesso Aleatório)
SOS:	*Sums of Squares* (Soma de Quadrados)
SDP:	*Semidefinite Programming* (Programação Semidefinida)

Lista de Símbolos

Letras latinas

a	Vetor de parâmetros de deslizamento longitudinal.
a_e	Parâmetro de deslizamento longitudinal da roda esquerda.
a_d	Parâmetro de deslizamento longitudinal da roda direita.
$\hat{a}$	Vetor de parâmetros de deslizamento longitudinal estimados.
$\hat{a}_e$	Parâmetro de deslizamento longitudinal estimado da roda esquerda.
$\hat{a}_d$	Parâmetro de deslizamento longitudinal estimado da roda direita.
$\tilde{a}_e$	Erro de estimação do parâmetro de deslizamento da roda esquerda.
$\tilde{a}_d$	Erro de estimação do parâmetro de deslizamento da roda direita.
A	Matriz de um sistema linear variante no tempo.
b	Distância entre o centro das rodas do robô móvel $[m]$.
c_i	Coeficientes de um polinômio ou função de Lyapunov polinomial.
d	Metade do grau de uma determinada forma polinomial.
e	Vetor de erro de postura do robô móvel.
e	Número de Euler.
e_1	Componente horizontal do erro de postura $[m]$.
e_2	Componente vertical do erro de postura $[m]$.
e_3	Componente angular do erro de postura [rad].
e_a	Vetor de erro aumentado.
f_a	Termo nominal da dinâmica do erro perturbado.
F_0	Referencial inercial.
F_1	Referencial fixo ao corpo do robô.

F_2 Referencial fixo ao robô móvel de referência.

g Termo evanescente da dinâmica do erro perturbado.

g_n Termo não evanescente da dinâmica do erro perturbado.

k_i Ganhos utilizados nos controladores.

ℓ_f Limitante final.

L_v Limitante da velocidade de referência de avanço.

L_ω Limitante da velocidade de referência angular.

L_σ Limitante do parâmetro de deslizamento lateral.

L_e Limitante do parâmetro de deslizamento longitudinal da roda esquerda.

L_d Limitante do parâmetro de deslizamento longitudinal da roda direita.

M Número de monômios numa determinada base monomial.

M_e Limitante da derivada do parâmetro de deslizamento longitudinal da roda esquerda.

M_d Limitante da derivada do parâmetro de deslizamento longitudinal da roda direita.

n Número de variáveis de um polinômio.

$\mathbb{N}_0$ Conjunto dos números naturais (incluindo zero).

p Variável utilizada para denotar polinômios diversos.

q Coordenada generalizada de um sistema de corpos rígidos.

q_p Vetor de postura do robô móvel.

q_r Vetor de postura do robô de referência.

Q Matriz Gramiana de uma decomposição SOS.

r Raio das rodas do robô móvel $[m]$.

$R(\cdot)$ Matriz de rotação entre dois referenciais no plano.

$\mathbb{R}$ Conjunto dos números reais.

$S(\cdot)$ Matriz que introduz o deslizamento lateral.

t Variável de tempo.

v_x Velocidade de avanço do robô móvel $[m/s]$.

v_y Velocidade de deslizamento lateral do robô móvel $[m/s]$.

v_r Velocidade de avanço de referência $[m/s]$.

v_c Entrada de controle auxiliar relativa à velocidade de avanço $[m/s]$.

v_e Velocidade de avanço do centro da roda esquerda $[m/s]$.

v_d Velocidade de avanço do centro da roda direita $[m/s]$.

V Função de Lyapunov.

w_p Vetor de velocidades locais do robô móvel.

x Vetor com as variáveis de um polinômio ou de um problema de otimização.

x_{p} Componente horizontal da postura do robô móvel $[m]$.

x_{r} Componente horizontal da postura de referência $[m]$.

y Vetor com as variáveis de um problema de otimização.

y_{p} Componente vertical da postura do robô móvel $[m]$.

y_{r} Componente vertical da postura de referência $[m]$.

z Base monomial de uma decomposição SOS.

Letras gregas

α_i Coeficientes de um polinômio característico.

$\bar{\alpha}_i$ Numeradores dos coeficientes de um polinômio característico.

$\bar{\alpha}_i$ Denominadores dos coeficientes de um polinômio característico.

β_i Polinômios auxiliares.

γ Constante utilizada na prova de estabilidade do sistema perturbado.

γ_i Ganhos da lei de adaptação.

δ Constante utilizada na prova de estabilidade do sistema nominal.

δ_f Constante utilizada na prova de estabilidade do sistema perturbado.

Δ_i Determinantes utilizados no Critério de Estabilidade de Liénard e Chipart.

$\bar{\Delta}_4$ Polinômio importante para a prova de estabilidade do sistema nominal.

ϵ Constante utilizada na prova de estabilidade do sistema nominal.

ε Constante pequena arbitrária.

ζ Constante pequena arbitrária.

η Vetor de velocidades de avanço e angular do robô móvel.

η_{r} Vetor de velocidades de referência.

η_c Vetor de entrada de controle auxiliar.

θ Constante utilizada na prova de estabilidade do sistema perturbado.

θ_{p} Orientação do robô móvel [rad].

θ_{r} Orientação do robô de referência [rad].

λ Autovalor.

ν Autovetor.

μ Limitante inferior da velocidade referência $[m/s]$.

ξ Vetor de velocidades angulares das rodas.

ρ Variável utilizada em problemas de otimização.

σ Parâmetro de deslizamento lateral.

ω Velocidade angular do robô móvel $[\text{rad}/s]$.

$\Phi(\cdot)$ Matriz que relaciona as velocidades de avanço com as velocidades angulares das rodas.

Ψ_i Funções utilizadas nas restrições de um sistema de corpos rígidos.

ω_{r} Velocidade angular de referência $[\text{rad}/s]$.

ω_c Entrada de controle auxiliar relativa à velocidade angular $[\text{rad}/s]$.

ω_e Velocidade angular da roda esquerda $[\text{rad}/s]$.

ω_d Velocidade angular da roda direita $[\text{rad}/s]$.

$\Omega(\cdot)$ Matriz antissimétrica de velocidades angulares do robô.

Capítulo 1

Introdução

Nas últimas duas décadas, o desenvolvimento de sistemas inovadores nas mais diversas áreas da robótica foi intenso. Sistemas como robôs humanoides, exoesqueletos e próteses robóticas, robôs móveis autônomos terrestres e aéreos têm exercido um papel cada vez mais de destaque tanto na comunidade científica quanto na sociedade em geral.

Dentre as áreas da robótica, a robótica móvel recebe grande atenção de pesquisadores devido à variedade de problemas a ela associados. Robôs móveis são extremamente diversos, podendo atuar em ambientes aéreos, terrestres e submarinos. Problemas ligados à locomoção, localização, mapeamento e inteligência artificial podem todos ser contextualizados na robótica móvel.

No que toca o problema da locomoção, é interessante estudar o caso dos robôs terrestres. Estes podem entrar em contato com o solo de diversas maneiras, cada uma com vantagens e desvantagens específicas. Seus meios de locomoção podem, por exemplo, se basear em pernas, patas, rodas, lagartas (esteiras) ou até mesmo numa mistura dessas tecnologias.

Quanto aos robôs terrestres com rodas, pode-se dizer que são utilizados com finalidades interessantes e importantes, dentre as quais é possível mencionar a exploração interplanetária, a manipulação de dispositivos explosivos improvisados e a locomoção de maquinário médico. Na Figura 1.1, pode-se ver dois exemplos de robôs móveis terrestres com rodas: o robô Curiosity, da agência espacial americana NASA, que está atualmente no solo de Marte, e o robô móvel Andros, da companhia do setor de defesa Northrop-Grumman, utilizado para manipular objetos periculosos

Figura 1.1: Exemplos de robôs móveis terrestres com rodas.

no lugar de pessoas. Nota-se que enquanto o Curiosity locomove-se com suas seis rodas tocando diretamente o solo, o Andros utiliza um sistema de lagartas acopladas às rodas.

As vantagens e desvantagens no uso de rodas ou lagartas está em parte ligada à questão do deslizamento, um problema comum entre os robôs móveis terrestres. Os tipos de deslizamentos que ocorrem nas rodas podem em geral ser divididos em deslizamentos longitudinais, que são responsáveis por fazer as rodas de um robô girarem em falso, e deslizamentos laterais, que ocorrem perpendicularmente à direção da trajetória do robô.

Os deslizamentos nas rodas dificultam o planejamento e o controle de trajetórias de um robô móvel. Para realizar tarefas que demandam uma certa precisão, é indispensável levar em conta o efeito dos deslizamentos no comportamento do robô. Assim, o problema do deslizamento tem sido o foco de diversos trabalhos recentes na área de robótica móvel e terá papel central neste trabalho.

1.1 Controle de sistemas robóticos

Assim como o projeto elétrico e mecânico dos diversos tipos de robôs existentes, os algoritmos matemáticos e computacionais utilizados no projeto de seus controladores têm evoluído de maneira rápida e dinâmica. Controladores robóticos, na prática, são algoritmos executados por microprocessadores conectados à rede eletrônica do robô. Essa rede pode ser constituída por inúmeros componentes, sendo que para fins de controle robótico os mais importantes são os sensores e atuadores.

A tarefa do controlador é, a partir de dados provenientes de sensores, bem como de dados externos chamados de referências, enviar comandos para os atuadores responsáveis pelo movimento do robô.

Um breve histórico sobre a integração entre a teoria de controle e as diferentes áreas da robótica, como o encontrado em Spong e Fujita (2011), mostra que a evolução e a diversificação dos robôs motivou o desenvolvimento e a aplicação prática de inúmeras técnicas de controle. Grande parte das técnicas de projeto de controlador requer que exista um modelo matemático representando o comportamento físico do robô. É comum, na robótica, fazer distinção entre modelos cinemáticos e modelos dinâmicos. Enquanto modelos cinemáticos relacionam diferentes velocidades do sistema robótico, modelos dinâmicos relacionam forças e momentos a acelerações do robô e de suas partes móveis. Neste trabalho, será dado enfoque ao controle de robôs móveis com base em modelos cinemáticos.

Os modelos cinemáticos e dinâmicos da maioria dos sistemas robóticos são representados em geral por sistemas de equações diferenciais não lineares. Em certos casos, a linearização do modelo em torno de um ponto de operação facilita o projeto de um controlador linear. Entretanto, quando se lida com sistemas robóticos, o uso de técnicas de controle linear torna-se limitado, dado que modelos de robôs possuem não linearidades importantes, dependendo muito da acoplagem entre seus ângulos, velocidades e posições (Spong et al., 2006). Assim, é comum que técnicas de controle não linear sejam utilizadas no projeto de controladores robóticos. Uma técnica clássica de controle não linear é o controle adaptativo, que permite atualizar determinados parâmetros do controlador de acordo com a evolução das condições de operação do sistema controlado. Neste trabalho, um controlador projetado com a técnica do controle adaptativo será utilizado para controlar um robô móvel, e o comportamento desse sistema controlado será analisado matematicamente.

Levando em conta que equações diferenciais ordinárias (EDOs) não lineares possuem soluções complicadas e em geral de difícil obtenção, as técnicas de projeto de controladores não lineares se baseiam em grande parte na Teoria de Lyapunov (Slotine e Li, 1991). A Teoria de Lyapunov trata do estudo de estabilidade de sistemas de EDOs, sem ser para isso necessário o conhecimento de suas soluções.

Ao longo deste trabalho, serão aplicados alguns teoremas ligados à Teoria de Lyapunov. Particularmente, será mostrado como a teoria de Lyapunov ajudou no desenvolvimento de uma lei

de controle adaptativa para um robô móvel capaz de estimar o deslizamento de suas rodas.

1.2 Robôs móveis holonômicos e não holonômicos

Os robôs móveis podem ser classificados em holonômicos e não holonômicos. A diferença entre essas duas classes se deve, principalmente, ao modo de construção das rodas do robô, ao número de rodas, e à geometria dos eixos das rodas em relação à carroceria do robô (Siegwart e Nourbakhsh, 2004). Esses três fatores fazem com que os modelos matemáticos de robôs holonômicos e não holonômicos apresentem características diferentes, que são devidas a certas relações geométricas e cinemáticas chamadas de restrições.

Um robô móvel pode ser representado por um sistema de múltiplos corpos rígidos, ligados fisicamente através de vínculos. A configuração desse conjunto de corpos rígidos no espaço pode ser descrita completamente através de variáveis chamadas coordenadas generalizadas. Os vínculos entre os corpos rígidos são responsáveis por introduzir relações matemáticas entre as coordenadas generalizadas que são chamadas de restrições. Alguns corpos rígidos podem ainda estar submetidos a condições específicas que também introduzem restrições, como por exemplo a condição de não deslizamento.

As restrições de um sistema podem ser representadas por relações geométricas, isto é, entre as coordenadas generalizadas, ou por relações cinemáticas, isto é, entre as coordenadas generalizadas e suas derivadas em relação ao tempo (Campion et al., 1991). Uma restrição é dita holonômica quando ela é representada por uma restrição geométrica ou por uma restrição cinemática que seja integrável. Quando uma restrição cinemática não é integrável, diz-se que ela é uma restrição não holonômica do sistema de corpos rígidos. A existência de restrições não holonômicas num sistema impede que seu número de coordenadas generalizadas possa ser reduzido a seu número de graus de liberdade.

Assim, formalmente, um robô móvel é dito holonômico quando ele pode ser representado por um sistema de corpos rígidos cujas restrições são todas restrições holonômicas. Se um robô móvel é representado por um sistema que possui pelo menos uma restrição não holonômica, então ele é chamado de robô não holonômico. Enquanto robôs terrestres holonômicos são sistemas de três graus de liberdade, o mesmo que o número de graus de liberdade de um corpo livre no plano,

robôs não holonômicos possuem menos de três graus de liberdade. A Figura 1.2 apresenta um robô móvel holonômico, e a Figura 1.3 apresenta um robô móvel não holonômico.

Figura 1.2: Robô móvel holonômico.

Figura 1.3: Robô móvel não holonômico.

Robôs holonômicos são muitas vezes chamados de "omnidirecionais" pois podem a qualquer dado momento se mover numa direção específica no plano, além de girar em torno de si mesmos. Devido a esse fato, as estratégias de controle de movimento para robôs móveis holonômicos e não holonômicos podem diferir substancialmente. Os trabalhos Liu et al. (2008), Kalmár-Nagy et al. (2004) e Jung et al. (1999) apresentam com detalhes os modelos cinemáticos de alguns robôs móveis holonômicos, e desenvolvem técnicas de controle de movimento para essa classe de robôs. Estratégias de controle de robôs móveis não holonômicos serão revisadas com mais detalhes adiante.

Um tipo de robô não holonômico muito utilizado devido à sua versatilidade, e que será foco deste trabalho, é o robô de tração diferencial. Esse robô possui duas rodas paralelas unidas por um eixo comum e acionadas independentemente. Uma terceira roda, não acionada, geralmente é utilizada para equilibrar o robô no plano, porém essa roda não tem influência alguma sobre a cinemática do robô. Um exemplo de robô de tração diferencial é mostrado na Figura 1.4. Um problema particularmente delicado que surge no controle de robôs de tração diferencial é a ocorrência de deslizamento lateral nas rodas, já que esse tipo de robô não possui acionamento na direção lateral.

Figura 1.4: Robô móvel de tração diferencial.

1.3 Revisão da literatura

A revisão apresentada nesta seção dará particular atenção ao problema do controle de movimentos de robôs móveis não holonômicos através de técnicas de controle cinemático, bem como às técnicas utilizadas para enfrentar o problema de deslizamento nas rodas.

De acordo com Morin e Samson (2008), duas classes de problemas de controle de movimento de robôs móveis são particularmente importantes: o problema de seguimento de caminho, e o problema de rastreamento de trajetórias. No problema de seguimento de caminho, o robô deve se mover com uma velocidade longitudinal constante, tentando se manter sobre uma determinada curva no plano. O desafio neste tipo de problema é manter em zero a distância entre o robô e o ponto mais próximo dessa curva.

No problema de rastreamento de trajetórias, a curva no plano sobre a qual o robô deve se mover é parametrizada pela variável de tempo, de modo que a velocidade do robô não pode mais ser uma constante predeterminada: o robô deve se mover com a velocidade necessária para se manter no ponto "atual" da curva. O desafio passa a ser manter em zero o erro entre a posição do robô e uma determinada posição de referência, parametrizada pelo tempo. Eventualmente, a orientação do robô também pode ser incluída no problema. Uma questão importante nesse caso é a factibilidade da trajetória. Só é possível rastrear perfeitamente uma trajetória caso esta seja factível para o tipo de robô considerado. Robôs não holonômicos, devido às suas restrições cinemáticas, não podem seguir qualquer tipo de trajetória. Para produzir trajetórias factíveis para esses robôs,

costuma-se utilizar um modelo de referência que possua as mesmas limitações cinemáticas que o modelo do robô.

Na década de 90, começaram a ser publicados trabalhos que abordavam o problema de rastreamento de trajetórias de robôs móveis não holonômicos a partir do uso de leis de controle não lineares com feedback. A premissa básica desses trabalhos era a medição da posição e da orientação do robô, e a elaboração de leis de controle que fizessem uso de um modelo de referência idêntico ao modelo cinemático do robô móvel. Um dos primeiros trabalhos nessa linha de pesquisa foi Kanayama et al. (1990), que propôs uma lei de controle não linear capaz de estabilizar a origem da dinâmica do erro de rastreamento, recorrendo a uma função de Lyapunov na prova de estabilidade. Esse resultado foi estendido por Kim e Oh (1998), que apresentou uma nova lei de controle e uma nova função de Lyapunov garantindo estabilidade assintótica do rastreamento. A estabilidade assintótica da origem da dinâmica do erro de rastreamento também foi obtida por Jiang e Nijmeijer (1997), que explorou a técnica de Backstepping, baseada no uso de sucessivas funções de Lyapunov.

Como notado por Fierro e Lewis (1995), que explorou o resultado de Kanayama et al. (1990) ao introduzir a dinâmica do robô no problema, o uso de leis de controle baseadas apenas no modelo cinemático do robô no plano tem suas limitações. Isso se deve ao fato que essas leis supõem que o robô produzirá perfeitamente as velocidades de translação e rotação no plano, e usam essas velocidades como a entrada de controle no modelo do robô. Na prática, isso não ocorre devido às inércias desconsideradas pelo modelo cinemático, e também devido a pequenos deslizamentos sofridos pelas rodas. Os trabalhos Yang e Kim (1999) e Kim et al. (2003) deram contribuições para a abordagem do problema de rastreamento de trajetórias com uso do modelo dinâmico, ao apresentar técnicas capazes de lidar com perturbações no torque aplicado nas rodas.

A abordagem com modelo cinemático, por sua versatilidade e pelo fato de não precisar de informações detalhadas sobre as inércias do robô, continuou sendo explorada. Para vencer as limitações impostas pelo deslizamento nas rodas, as técnicas cinemáticas de controle para robôs não holonômicos passaram a levar a cinemática das rodas em consideração. Nesse contexto, Tarokh e McDermott (2005) propôs modelos de deslizamento na cinemática de robôs do tipo *rover*, que têm várias rodas atuadas independentemente. No mesmo sentido, Wang e Low (2008) abordou a inclusão de diferentes tipos de deslizamento no modelo cinemático de vários tipos de robôs não holonômicos, incluindo o robô de tração diferencial.

De acordo com Wang e Low (2008), três tipos de perturbações ligadas ao deslizamento nas rodas podem ser incluídas no modelo cinemático de um robô móvel de tração diferencial: o deslizamento longitudinal (*slipping*), o deslizamento lateral (*skidding*) e uma perturbação na taxa de rotação (*yaw rate*) do robô. Nesse trabalho, foi mostrado que o nível de dificuldade de tratar cada uma dessas perturbações, do ponto de vista da teoria de controle, não é o mesmo. O deslizamento longitudinal e a perturbação na taxa de rotação se encontram em canais contendo entradas de controle, i.e., são perturbações do tipo *input additive*, e portanto podem ser compensadas e até mesmo eliminadas através de uma estratégia de controle adequada. Por outro lado, o deslizamento lateral não entra em nenhum canal que contenha entradas de controle, i.e., trata-se de uma perturbação do tipo *unmatched*, sendo portanto um problema muito mais difícil de ser tratado.

Alguns trabalhos obtiveram sucesso em resolver o problema de rastreamento de trajetórias de um robô de tração diferencial sujeito ao deslizamento longitudinal quando este ocorre isoladamente. Tal problema foi abordado por González et al. (2009) através do uso de um controlador baseado em *feedback linearization*. Esse controlador possui parâmetros ligados ao deslizamento longitudinal, que são estimados com base na medição discreta das velocidade das rodas e do robô. Em González et al. (2010), uma técnica adaptativa baseada em desigualdades matriciais lineares é utilizada para projetar um controlador robusto a variações no modelo do robô sujeito ao deslizamento longitudinal, porém o deslizamento longitudinal é assumido conhecido.

No caso em que o deslizamento longitudinal não é conhecido, Iossaqui et al. (2010) introduziu a ideia de estimar o deslizamento através de uma lei de controle adaptativa baseada na análise de estabilidade do robô móvel com o uso de uma função de Lyapunov. No entanto, nesse trabalho foi considerado que o deslizamento longitudinal é igual em ambas as rodas. Tal resultado foi estendido em Iossaqui et al. (2011), onde foi considerado que o deslizamento longitudinal é diferente em cada roda. Esse trabalho mostrou que, partindo do controlador cinemático de Kim e Oh (1998), é possível através do uso de funções de Lyapunov obter uma lei de controle adaptativa capaz de fazer com que o erro de rastreamento do robô móvel tenda assintoticamente a zero, mesmo sob efeito do deslizamento longitudinal desconhecido. Em Iossaqui et al. (2013), foi mostrado adicionalmente que o controlador proposto também faz com que os erros de estimação do deslizamento longitudinal tendam a zero, quando o deslizamento longitudinal é assumido constante. A tese de doutorado Iossaqui (2013) apresenta um estudo detalhado do problema de rastreamento de trajetórias de um robô móvel sujeito a deslizamentos longitudinais nas rodas, incluindo alguns dos resultados citados anteriormente.

O deslizamento longitudinal é apenas um dos tipos deslizamento que podem ocorrer durante a operação de um robô móvel. Apenas recentemente, porém, o deslizamento lateral passou a ser estudado no contexto de problemas de rastreamento de trajetórias com o uso de controladores cinemáticos. Em Ryu e Agrawal (2011), foi apresentado um controlador baseado na técnica de *differential flatness* para controlar um robô móvel sujeito a deslizamentos laterais. Em Zhou et al. (2007), foram considerados dois tipos de deslizamento, longitudinal e lateral, e foi proposto um método baseado em filtros de Kalman para estimá-los em tempo real. No entanto, não foi provada a estabilidade do sistema em malha fechada. O modelo de deslizamento lateral utilizado em Zhou et al. (2007) também foi utilizado em Burghi et al. (2015), em que foi apresentado um método para analisar o comportamento de um robô móvel não holonômico sujeito apenas a deslizamentos laterais.

O método em Burghi et al. (2015) utiliza resultados da teoria de Lyapunov para sistemas não lineares perturbados, e foi aplicado para analisar a robustez de um controlador relativamente simples. Essa aplicação é a base do método apresentado neste trabalho, cujo objetivo agora pode ser estabelecido.

1.4 Objetivo da dissertação

O objetivo principal deste trabalho é utilizar técnicas de análise de sistemas não lineares perturbados para investigar a robustez em relação ao deslizamento lateral do controlador adaptativo cinemático de Iossaqui et al. (2011) (também apresentado em Iossaqui et al. (2013) e Iossaqui (2013)), que é capaz de compensar o efeito isolado do deslizamento longitudinal. Esse controlador foi projetado para solucionar o problema de rastreamento de trajetórias de um robô móvel de tração diferencial sujeito a deslizamentos longitudinais, porém o deslizamento lateral foi ignorado por completo em seu projeto. A metodologia adotada para atingir esse objetivo será descrita a seguir.

1.4.1 Metodologia

Deve-se primeiramente derivar um modelo cinemático para um robô móvel não holonômico de tração diferencial que leve em conta o deslizamento lateral e longitudinal. O modelo obtido deve

ser versátil, ou seja, capaz de representar facilmente os casos em que os deslizamentos longitudinal e lateral são levados em conta simultaneamente, isoladamente, ou ignorados por completo. Esse modelo será obtido na Seção 2.1.

Para contextualizar de maneira precisa este trabalho, a técnica de controle de trajetórias de um robô móvel de tração diferencial desenvolvida por Kim e Oh (1998) e aprimorada em Iossaqui et al. (2011), Iossaqui et al. (2013) e Iossaqui et al. (2013) deve ser estudada detalhadamente. Essa técnica será revisada com profundidade na Seção 2.2, na qual também será formalizado o problema de rastreamento de trajetórias do robô móvel sujeito a deslizamentos laterais e longitudinais.

O deslizamento lateral será tratado como um termo de perturbação na dinâmica não linear do erro de rastreamento de trajetórias do robô, como feito em Wang e Low (2008) e Zhou et al. (2007). A ideia principal, que será desenvolvida na Seção 2.3, é mostrar que o controlador apresentado em Iossaqui et al. (2011), Iossaqui et al. (2013) e Iossaqui (2013) é capaz de fazer com que as soluções da dinâmica do erro de rastreamento sejam uniformemente finalmente limitadas. As ferramentas teóricas de análise de sistemas não lineares perturbados que serão utilizadas com esse fim, extraídas principalmente de Khalil (2002), estão apresentadas no Apêndice A.

Um dos passos da análise de estabilidade necessária para atingir o objetivo deste trabalho requer a resolução de um problema de otimização não linear consideravelmente grande. Para resolver esse problema, serão utilizadas técnicas de otimização polinomial baseadas em Somas de Quadrados, ou, abreviadamente, SOS (*Sums of Squares*). A Seção 4.1 revisará com detalhes os principais resultados ligados à otimização polinomial baseada em técnicas SOS. É importante notar que as técnica SOS vêm sendo cada vez mais utilizadas no contexto de controle não linear. Porém, suas aplicações no projeto de controladores foge do escopo deste trabalho.

Para complementar a análise teórica de estabilidade, uma simulação numérica do comportamento do robô móvel controlado pela lei de Iossaqui et al. (2011), e sujeito aos dois tipos de deslizamento, longitudinal e lateral, será apresentada no Capítulo 3.

É importante mencionar que a investigação de robustez de sistemas não lineares usando o conceito de soluções uniformemente finalmente limitadas já foi explorada no contexto de robótica manipulativa, como visto em Chen et al. (2001), Koo e Kim (1994) e Qu e Dorsey (1991), mas até hoje encontrou poucas aplicações no que concerne o deslizamento de robôs móveis não holonômicos.

1.4.2 Decomposições SOS

Uma ferramenta importante utilizada para atingir o objetivo principal deste trabalho será a teoria de otimização polinomial através de decomposições SOS. A dificuldade de se resolver numericamente um problema de otimização particularmente grande que surge durante a análise de estabilidade desenvolvida na Seção 2.3 exige uma investigação detalhada a respeito dos programas de otimização polinomial disponíveis atualmente, de suas funcionalidades, e dos recursos computacionais necessários para sua utilização.

Essa investigação computacional constitui um objetivo à parte deste trabalho, e será apresentada na Seção 4.3. A investigação passa primeiramente pela identificação de programas de computador capazes de resolver problemas de otimização polinomial, obtendo ótimos globais. A seguir, será necessário fazer um estudo dos recursos computacionais necessários para a resolução de problemas grandes de otimização polinomial com o uso desses programas. Será útil entender como os recursos computacionais, em especial a quantidade de memória do computador, estão relacionados com o tamanho dos problemas de otimização. Dado que os programas de otimização polinomial possuem diferentes funcionalidades opcionais, será importante entender quais são essas funcionalidades e quais são as vantagens trazidas por suas utilizações. Finalmente, é importante entender qual é a garantia numérica apresentada pelos programas que permite dizer que os ótimos encontrados são globais.

Capítulo 2

Controle de um Robô Móvel Sujeito a Deslizamentos Longitudinais e Laterais

2.1 Modelagem cinemática

Nesta seção, serão derivados modelos cinemáticos de um robô móvel de tração diferencial não holonômico, sujeito a deslizamentos laterais e longitudinais nas rodas. A modelagem do deslizamento longitudinal se baseia em Iossaqui et al. (2011) e Iossaqui (2013), enquanto o modelo de deslizamento lateral é extraído de Wang e Low (2008) e Zhou et al. (2007). A notação matricial utilizada na modelagem e nas próximas seções é própria deste trabalho.

2.1.1 Preliminares

Na Figura 2.1, estão representados três referenciais diferentes, sendo que o primeiro deles, o referencial F_0, é um referencial inercial. Os eixos x_0 e y_0 definem o plano no qual o robô pode se mover, e o eixo z_0 define o eixo de rotação no plano.

O robô é representado na Figura 2.1 por uma barra ligando duas rodas, vistas por cima. Apesar de, tecnicamente, esse conjunto ser composto por três corpos rígidos, neste trabalho não há interesse em tratar as rodas como tais. Assim, a posição e orientação do robô no plano podem ser

definidas com a ajuda do referencial local F_1, fixo ao corpo rígido dado pela barra. Tal consideração pode ser feita pois o corpo do robô é construído sobre a barra e não se move relativamente a esta. O eixo x_1 indica a direção frontal do robô, e o eixo y_1 indica a direção lateral do robô.

É importante estabelecer outro referencial, F_2, que será utilizado para posicionar e orientar um determinado robô móvel de referência, representado na Figura 2.1 por um triângulo. O robô de referência (ou simplesmente "a referência") será necessário para formular o problema de controle de trajetórias do robô principal (referido apenas como "o robô"). O robô de referência pode representar tanto um objetivo virtual quanto um robô "líder", que deve ser seguido pelo robô principal.

O vetor de postura do robô é definido por

$$q_\mathrm{p} = (x_\mathrm{p},\ y_\mathrm{p},\ \theta_\mathrm{p})^T \tag{2.1}$$

em que o ponto $(x_\mathrm{p}, y_\mathrm{p})$, chamado de posição do robô, é a posição da origem de F_1 em relação à origem de F_0, expressa na base de F_0. O ângulo θ_p, chamado de orientação do robô, é o ângulo entre x_0 e x_1.

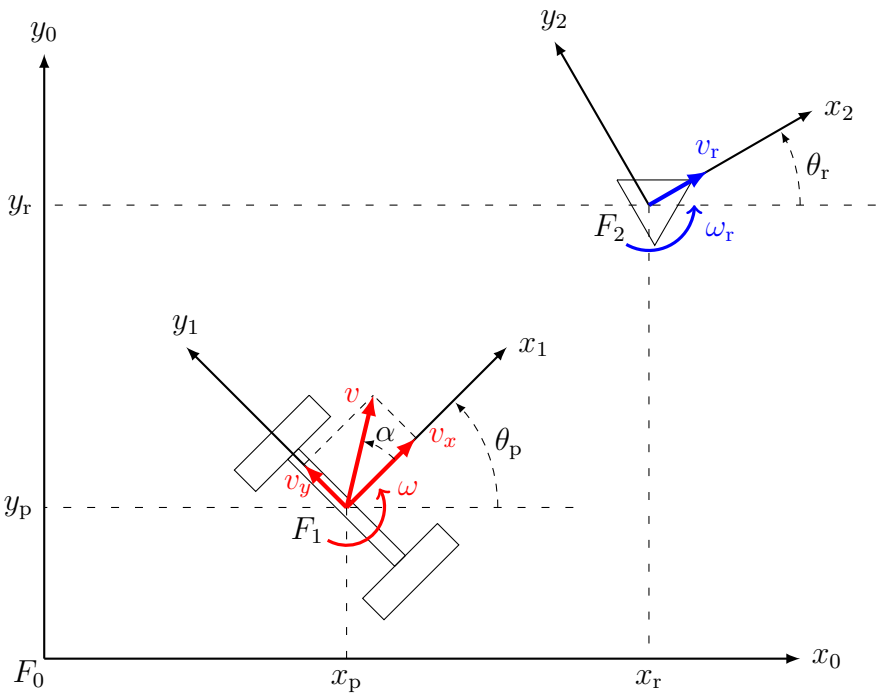

Figura 2.1: Representação bidimensional do robô móvel e da referência.

Analogamente, a posição e a orientação do robô de referência são agrupadas no vetor de

postura de referência,

$$q_{\mathrm{r}} = (x_{\mathrm{r}},\ y_{\mathrm{r}},\ \theta_{\mathrm{r}})^T \tag{2.2}$$

Ainda na Figura 2.1, as velocidades v e ω são, respectivamente, a velocidade de translação do robô em relação a F_0 e a velocidade angular do robô em torno de z_0. A velocidade v pode ser projetada nas direções x_1 e y_1, de modo a obter v_x e v_y. Tendo em vista as restrições de movimento impostas pelas rodas do robô da Figura 2.1, tem-se que, desconsiderando todo tipo de deslizamento, $v_y = 0$. Contudo, este trabalho visa estudar o caso em que há deslizamento lateral do robô, i.e., o caso em que $v_y \neq 0$. Portanto, chama-se a velocidade v_x de velocidade de avanço do robô, e a velocidade v_y de velocidade de deslizamento lateral do robô.

Na literatura, é comum que a análise de veículos com rodas sujeitos a deslizamentos seja feita levando em conta o ângulo de deslizamento lateral α (ver por exemplo Pacejka (2012)), ilustrado na Figura 2.1 e definido através da seguinte relação:

$$\tan(\alpha) = \frac{v_y}{v_x}$$

Neste trabalho, como em Wang e Low (2008) e Zhou et al. (2007), será utilizado um parâmetro de deslizamento lateral variante no tempo, $\sigma = \tan(\alpha)$. Assim, a velocidade de deslizamento lateral do robô pode ser dada em função da velocidade de avanço através da relação

$$v_y = \sigma v_x \tag{2.3}$$

Em relação ao robô móvel de referência, será considerado que seu movimento não é influenciado por deslizamentos. Logo, apenas duas velocidades caracterizam completamente esse movimento: v_{r} e ω_{r}, chamadas, respectivamente, de velocidade de avanço de referência e a velocidade angular de referência.

Com o intuito de desenvolver equações cinemáticas na forma matricial, será utilizada a matriz de rotação,

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

que possui as seguintes propriedades:

- $R(\theta)$ é ortogonal, i.e., $R(\theta)^T = R(\theta)^{-1}$, implicando que $R(\theta)$ é sempre inversivel.
- $R(-\theta) = R(\theta)^T$
- $R(\theta_1)R(\theta_2) = R(\theta_1 + \theta_2)$

2.1.2 Modelo cinemático do robô móvel

Define-se o vetor de velocidades locais (expressas em F_1) do robô por

$$w_{\mathrm{p}} = (v_x, v_y, \omega)^T \tag{2.5}$$

Esse vetor pode ser reescrito como

$$w_{\mathrm{p}} = S(\sigma) \begin{bmatrix} v_x \\ \omega \end{bmatrix} \tag{2.6}$$

em que $S(\sigma)$ é dada por

$$S(\sigma) = \begin{bmatrix} 1 & 0 \\ \sigma & 0 \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

O vetor contendo apenas v_x e ω, em (2.6), será denotado por

$$\eta = (v_x, \omega)^T \tag{2.8}$$

Esse vetor será particularmente importante nas próximas seções deste capítulo, pois v_x e ω são as velocidades sobre as quais se têm controle através do acionamento das rodas do robô.

Para expressar o vetor de velocidades locais do robô w_{p} no referencial inercial, basta realizar uma troca de bases do referencial F_1 para o referencial F_0. Essa troca é feita através do uso da matriz de rotação (2.4). Observando que as velocidades do robô no referencial inercial F_0 nada mais são que a derivada de seu vetor de postura, $\dot{q}_{\mathrm{p}} = (\dot{x}_{\mathrm{p}}, \dot{y}_{\mathrm{p}}, \dot{\theta}_{\mathrm{p}})^T$, tem-se que

$$\dot{q}_{\mathrm{p}} = R(\theta_{\mathrm{p}})w_{\mathrm{p}} \tag{2.9}$$

$$= R(\theta_{\mathrm{p}})S(\sigma)\eta \tag{2.10}$$

De maneira totalmente análoga, definindo o vetor de velocidades de referência por

$$\eta_{\mathrm{r}} = (v_{\mathrm{r}}, \omega_{\mathrm{r}})^T$$

tem-se que as velocidades da referência no referencial inercial, $\dot{q}_{\mathrm{r}} = (\dot{x}_{\mathrm{r}}, \dot{y}_{\mathrm{r}}, \dot{\theta}_{\mathrm{r}})^T$, são dadas por

$$\dot{q}_{\mathrm{r}} = R(\theta_{\mathrm{r}})S(0)\eta_{\mathrm{r}} \tag{2.11}$$

em que $S(0)$ é a matriz da Equação (2.7) com $\sigma = 0$.

A Equação (2.10) é o modelo cinemático que relaciona as velocidades do robô, enquanto a Equação (2.11) é o modelo cinemático da referência.

2.1.3 Restrições não holonômicas

É interessante neste ponto fazer uma breve análise do significado do modelo (2.10) no contexto de restrições dinâmicas. Em termos de dinâmica analítica, os elementos do vetor de postura q_{p} da Equação (2.1) são coordenadas generalizadas do corpo rígido que representa o robô, ao qual é associado o referencial móvel F_1. Em outras palavras, com o vetor q_{p} é possível definir completamente a posição e a orientação do robô (sua configuração espacial) em um dado instante de tempo.

Os elementos da derivada do vetor de postura $\dot{q}_{\mathrm{p}}$ são velocidades generalizadas do robô, que definem completamente a variação temporal de sua configuração espacial em um dado instante de tempo. Contudo, existem outras velocidades, chamadas de quase-velocidades, que podem igualmente definir de maneira completa a variação temporal da configuração espacial de um corpo rígido (Greenwood, 2006). De forma geral, uma quase-velocidade w_j é definida em função do vetor de coordenadas generalizadas q e do vetor de velocidades generalizadas $\dot{q}$ de acordo com a seguinte relação:

$$w_j = \Psi_j(q,t)\dot{q} + \Psi_{j0}(q,t) \tag{2.12}$$

em que $\Psi_j : \mathbb{R}^{n_q} \times [0,\infty) \to \mathbb{R}^{1\times n_q}$ e $\Psi_{j0} : \mathbb{R}^{n_q} \times [0,\infty) \to \mathbb{R}$, com $n_q = \dim q$.

As quase-velocidades estão intimamente ligadas às restrições de um sistema de corpos rígidos. Quando uma dada quase-velocidade w_j é nula para todo tempo, a Equação (2.12) se torna uma relação entre coordenadas e velocidades generalizadas chamada de restrição do sistema,

$$0 = \Psi_j(q,t)\dot{q} + \Psi_{j0}(q,t) \tag{2.13}$$

Se $\Psi_j(q,t) = 0$, ou, alternativamente, se a Equação (2.13) é integrável, diz-se que ela representa uma restrição holonômica do sistema. Se $\Psi_j(q,t) \neq 0$ e Equação (2.13) não é integrável, diz-se que ela representa uma restrição não holonômica do sistema. O robô móvel de tração diferencial estudado neste trabalho é não holonômico. Para enxergar esse fato, nota-se, comparando as Equações (2.9) e (2.12), que o vetor de velocidades locais $w_\mathrm{p} = (v_x, v_y, \omega)^T$ pode ser visto como um vetor de quase-velocidades do robô móvel, definidas através das matrizes $\Psi(q_\mathrm{p}, t) = R(\theta_\mathrm{p})^T$ e $\Psi_0 = 0$, ou seja,

$$w_\mathrm{p} = R(\theta_\mathrm{p})^T \dot{q}_\mathrm{p} \tag{2.14}$$

Quando não há deslizamento lateral, tem-se $v_y = 0$, e a segunda linha da Equação matricial (2.14) resulta na Equação

$$0 = -\sin(\theta_\mathrm{p})\dot{x}_\mathrm{p} + \cos(\theta_\mathrm{p})\dot{y}_\mathrm{p}$$

que, por não ser integrável, é uma restrição não holonômica do corpo rígido que representa o robô. Portanto, o robô móvel de tração diferencial é dito não holonômico, adjetivo utilizado para caracterizar qualquer sistema que possua ao menos uma restrição não holonômica.

2.1.4 Modelo cinemático considerando as rodas

A Figura 2.2 representa as duas rodas do robô móvel, unidas por uma barra à qual é fixada o referencial local F_1. Esse referencial é o mesmo referencial F_1 ilustrado anteriormente na Figura 2.1. Sejam ω_e e ω_d as velocidades de rotação própria das rodas esquerda e direita do robô, respectivamente, como ilustrado na Figura 2.2. Elas são dadas em torno do eixo y_1. As velocidades ω_e e ω_d são importantes pois, na prática, são as velocidades geradas pelos motores do robô. É possível obter um modelo cinemático relacionando tais velocidades ao vetor de velocidades generalizadas $\dot{q}_\mathrm{p} = (\dot{x}_\mathrm{p}, \dot{y}_\mathrm{p}, \dot{\theta}_\mathrm{p})^T$, levando em conta possíveis deslizamentos longitudinais nas rodas.

Referindo-se novamente à Figura 2.2, tem-se que o deslizamento longitudinal ocorre em uma roda quando a componente no eixo x_1 da velocidade do ponto de contato da roda com o solo é não nula (a componente em y_1 dessa mesma velocidade é gerada pelo deslizamento lateral). Sejam v_e e v_d as componentes no eixo x_1 das velocidades de translação dos centros das rodas esquerda e direita, respectivamente. Se não há deslizamento longitudinal, e se r é o raio das rodas, então v_e e v_d são dadas pelas relações usuais $v_e = r\omega_e$ e $v_d = r\omega_d$.

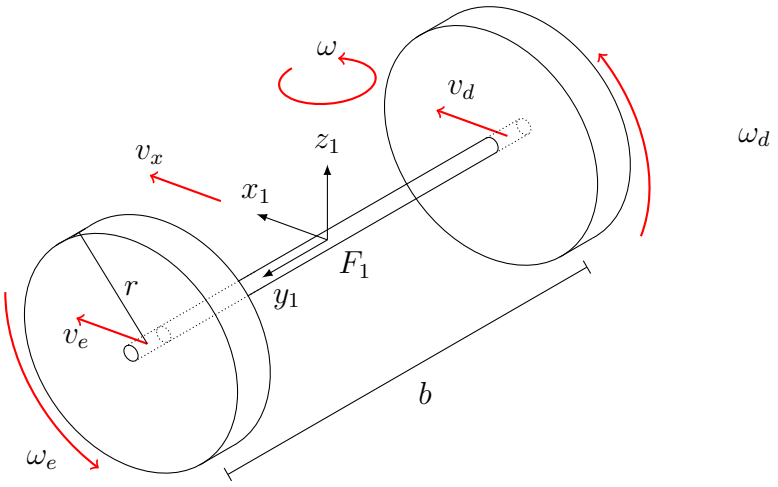

Figura 2.2: Velocidades do robô móvel e das rodas do robô.

Quando há deslizamento longitudinal nas rodas, entretanto, a falta de aderência ao solo faz com que as velocidades v_e e v_d sejam menores que aquelas esperadas caso as rodas girassem sem deslizar. Este efeito pode ser modelado, como em González et al. (2009), González et al. (2010), Iossaqui et al. (2011) e Iossaqui et al. (2013), através da introdução de parâmetros de deslizamento longitudinal. Sejam $a_e \in [1, \infty)$ e $a_d \in [1, \infty)$ os parâmetros de deslizamento longitudinal da roda esquerda e da roda direita, respectivamente. Então, o efeito do deslizamento longitudinal pode ser descrito pelas relações

$$v_e = r\frac{\omega_e}{a_e} \tag{2.15}$$

$$v_d = r\frac{\omega_d}{a_d} \tag{2.16}$$

De acordo com as relações (2.15) e (2.16), se $a_e = 1$ e $a_e = 1$, então não ocorre deslizamento longitudinal.

Nota-se que as velocidades $\eta = (v_x, \omega)^T$ são uma combinação linear de v_e e v_d. Seja b a distância entre o centro de cada roda. Então, como mostrado em Siegwart e Nourbakhsh (2004), tem-se que

$$v_x = \frac{1}{2}(v_d + v_e) \tag{2.17}$$

$$\omega = \frac{1}{b}(v_d - v_e) \tag{2.18}$$

Substituindo as Equações (2.15) e (2.16) nas Equações (2.17) e (2.18), obtém-se

$$v_x = \frac{r}{2}\left(\frac{\omega_d}{a_d} + \frac{\omega_e}{a_e}\right) \tag{2.19}$$

$$\omega = \frac{r}{b}\left(\frac{\omega_d}{a_d} - \frac{\omega_e}{a_e}\right) \tag{2.20}$$

Definindo o vetor de velocidades angulares das rodas por $\xi = (\omega_e, \omega_d)^T$, o vetor de parâmetros de deslizamento longitudinal por $a = (a_e, a_d)^T$ e lembrando que $\eta = (v_x, \omega)^T$, as Equações (2.19) e (2.20) podem ser expressas matricialmente por

$$\eta = \Phi(a)\xi \tag{2.21}$$

em que

$$\Phi(a) := \Phi(a_e, a_d) = \begin{bmatrix} \frac{r}{2a_e} & \frac{r}{2a_d} \\ \frac{-r}{ba_e} & \frac{r}{ba_d} \end{bmatrix} \tag{2.22}$$

A matriz $\Phi(a)$ acima representa a transformação entre o vetor de velocidades angulares das rodas ξ e o vetor de velocidades do robô η, considerando o deslizamento longitudinal nas rodas. Como $\det(\Phi(a)) \neq 0$ para quaisquer $a_e, a_d \in [1, \infty)$, a matriz $\Phi(a)$ é sempre inversível. Sua inversa é dada por

$$\Phi(a)^{-1} := \Phi(a_e, a_d)^{-1} = \begin{bmatrix} \frac{a_e}{r} & \frac{-ba_e}{2r} \\ \frac{a_d}{r} & \frac{ba_d}{2r} \end{bmatrix} \tag{2.23}$$

Finalmente, substituindo a Equação (2.21) na Equação (2.10), obtém-se o modelo cinemático do robô relacionando suas velocidades no referencial inercial, $\dot{q}_\mathrm{p} = (\dot{x}_\mathrm{p}, \dot{y}_\mathrm{p}, \dot{\theta}_\mathrm{p})^T$, com as velocidades de suas rodas, $\xi = (\omega_e, \omega_d)^T$, através de

$$\dot{q}_\mathrm{p} = R(\theta_\mathrm{p})S(\sigma)\Phi(a)\xi \tag{2.24}$$

em que $R(\theta_\mathrm{p})$ é a matriz de rotação (2.4), $S(\sigma)$ é a matriz da Equação (2.7), que introduz o deslizamento lateral, e $\Phi(a)$ é a matriz da Equação (2.22), que introduz a cinemática das rodas e o deslizamento longitudinal.

2.2 Controle do robô móvel sujeito a deslizamentos longitudinais

Esta seção apresenta uma lei de controle adaptativa não linear para o robô móvel da Figura 2.1 quando as rodas do robô sofrem deslizamentos longitudinais. Porém, o deslizamento lateral não é levado em conta. A lei que será apresentada foi desenvolvida em Iossaqui et al. (2011), Iossaqui et al. (2013) e Iossaqui (2013) utilizando técnicas de análise de sistemas não lineares, mais especificamente, a teoria de Lyapunov para sistemas não lineares. O trabalho Kim e Oh (1998) foi particularmente importante no desenvolvimento dessa lei de controle adaptativa.

2.2.1 Formulação do problema de controle

O problema de controle em questão é o problema de rastreamento de trajetórias por um robô móvel, como apresentado em Morin e Samson (2008). Para esse problema, o objetivo principal é projetar um controlador para o robô móvel que garanta

$$\lim_{t\to\infty}(q_{\mathrm{r}}(t) - q_{\mathrm{p}}(t)) = 0 \tag{2.25}$$

em que $q_{\mathrm{p}}(t) = (x_{\mathrm{p}}(t), y_{\mathrm{p}}(t), \theta_{\mathrm{p}}(t))^T$ é a postura do robô e $q_{\mathrm{r}}(t) = (x_{\mathrm{r}}(t), y_{\mathrm{r}}(t), \theta_{\mathrm{r}}(t))^T$ é a postura de referência, introduzidas na Seção 2.1. A postura de referência $q_{\mathrm{r}}(t)$ é regida pelo modelo cinemático da referência, Equação (2.11), e a postura do robô $q_{\mathrm{p}}(t)$ é regida pelo modelo cinemático do robô, que pode ser representado pela Equação (2.10), caso a cinemática das rodas seja ignorada, ou pela Equação (2.24), caso a cinemática das rodas seja levada em conta.

Para garantir que a postura do robô q_{p} siga a postura da referência q_{r}, começa-se por definir o erro de postura $e = (e_1, e_2, e_3)^T$ por

$$e = R(\theta_{\mathrm{p}})^T(q_{\mathrm{r}} - q_{\mathrm{p}}) \tag{2.26}$$

em que a matriz de rotação $R(\theta_{\mathrm{p}})$, dada pela Equação (2.4), é usada para expressar o erro de postura e no referencial local F_1.

Pode-se mostrar que a Equação (2.25) é satisfeita caso $e(t) \to 0$ com $t \to \infty$. Para entender o comportamento do erro de postura, é necessário levantar um sistema de equações diferenciais em

termos de $e(t)$. Nesse sistema, surgirá um termo chamado de entrada de controle, que dependerá da escolha do modelo cinemático do robô. A lei matemática escolhida para a entrada de controle é, na prática, o controlador que será responsável por garantir que a solução $e(t)$ tenda a zero, resolvendo o problema de rastreamento de trajetórias.

Derivando a Equação (2.26) em relação ao tempo, tem-se

$$\dot{e} = \dot{R}(\theta_\mathrm{p})^T(q_\mathrm{r} - q_\mathrm{p}) + R(\theta_\mathrm{p})^T(\dot{q}_\mathrm{r} - \dot{q}_\mathrm{p}) \tag{2.27}$$

A derivada da matriz de rotação $R(\theta_\mathrm{p})$ dada por (2.4) é

$$\begin{aligned}\dot{R}(\theta_\mathrm{p}) &= \begin{bmatrix} -\sin(\theta_\mathrm{p})\dot{\theta}_\mathrm{p} & -\cos(\theta_\mathrm{p})\dot{\theta}_\mathrm{p} & 0 \\ \cos(\theta_\mathrm{p})\dot{\theta}_\mathrm{p} & -\sin(\theta_\mathrm{p})\dot{\theta}_\mathrm{p} & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -\sin(\theta_\mathrm{p})\omega & -\cos(\theta_\mathrm{p})\omega & 0 \\ \cos(\theta_\mathrm{p})\omega & -\sin(\theta_\mathrm{p})\omega & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= R(\theta_\mathrm{p})\Omega(\omega)\end{aligned} \tag{2.28}$$

em que $\Omega(\omega)$ é a matriz antissimétrica

$$\Omega(\omega) = \begin{bmatrix} 0 & -\omega & 0 \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.29}$$

Substituindo a Equação (2.28) na Equação (2.27), tem-se

$$\dot{e} = \Omega(\omega)^T R(\theta_\mathrm{p})^T(q_\mathrm{r} - q_\mathrm{p}) + R(\theta_\mathrm{p})^T(\dot{q}_\mathrm{r} - \dot{q}_\mathrm{p})$$

Usando a definição de $e(t)$, da Equação (2.26), e observando que $\Omega(\omega)^T = -\Omega(\omega)$, tem-se

$$\dot{e} = -\Omega(\omega)e + R(\theta_\mathrm{p})^T(\dot{q}_\mathrm{r} - \dot{q}_\mathrm{p})$$

Substituindo $\dot{q}_\mathrm{r}$ dado pela Equação (2.11), o modelo cinemático de referência, tem-se

$$\dot{e} = -\Omega(\omega)e + R(\theta_\mathrm{p})^T\big(R(\theta_\mathrm{r})S(0)\eta_\mathrm{r} - \dot{q}_\mathrm{p}\big) \tag{2.30}$$

em que a transformação $S(\cdot)$ é dada por (2.7), e $\eta_\mathrm{r} = (v_\mathrm{r}, \omega_\mathrm{r})^T$.

Pelas propriedades da matriz de rotação, apresentadas na Seção 2.1.1, tem-se que

$$R(\theta_\mathrm{p})^T R(\theta_\mathrm{r}) = R(-\theta_\mathrm{p})R(\theta_\mathrm{r}) = R(\theta_\mathrm{r} - \theta_\mathrm{p}) = R(e_3)$$

Já que $e_3 = \theta_\mathrm{r} - \theta_\mathrm{p}$, como visto por (2.26), a Equação (2.30) pode ser reescrita como

$$\dot{e} = -\Omega(\omega)e + R(e_3)S(0)\eta_\mathrm{r} - R(\theta_\mathrm{p})^T\dot{q}_\mathrm{p} \tag{2.31}$$

Se a cinemática das rodas for desconsiderada, e negligenciando o deslizamento longitudinal, então o vetor $\dot{q}_\mathrm{p}$ em (2.31) deve ser substituído pela Equação (2.10), resultando em

$$\dot{e} = -\Omega(\omega)e + R(e_3)S(0)\eta_\mathrm{r} - S(\sigma)\eta \tag{2.32}$$

em que $\eta = (v_x, \omega)^T$. Caso a cinemática das rodas seja levada em conta, então o vetor $\eta = (v_x, \omega)^T$ em (2.32) deve ser substituído de acordo com (2.21), resultando em

$$\dot{e} = -\Omega(\omega)e + R(e_3)S(0)\eta_\mathrm{r} - S(\sigma)\Phi(a)\xi \tag{2.33}$$

em que $\xi = (\omega_e, \omega_d)^T$, a matriz $\Phi(a)$ é dada pela Equação (2.22) e ω é a segunda componente de $\eta = \Phi(a)\xi$.

Os sistemas de equações diferenciais ordinárias (2.32) e (2.33) representam a dinâmica do erro de postura do robô em malha aberta. A Equação (2.32) ignora a cinemática das rodas, e a Equação (2.33) a considera. A entrada de controle do sistema (2.32) é $\eta = (v_x, \omega)^T$, e a entrada de controle do sistema (2.33) é $\xi = (\omega_e, \omega_d)^T$. Em ambos os sistemas, $\eta_\mathrm{r} = (v_\mathrm{r}, \omega_\mathrm{r})^T$ é uma entrada de referência conhecida e o parâmetro de deslizamento lateral σ é uma perturbação desconhecida. No sistema (2.33), os parâmetros de deslizamento longitudinal $a = (a_e, a_d)^T$ também são perturbações desconhecidas.

Fica claro que o objetivo do problema de rastreamento de trajetórias, dado pela Equação (2.25), será satisfeito caso a entrada de controle, η no sistema (2.32) ou ξ no sistema (2.33), faça com que a solução $e(t)$ do respectivo sistema tenda a zero com $t \to \infty$. Nas duas próximas seções, será demonstrado que o projeto da lei de controle η para o sistema (2.32) pode ser usado no projeto da lei de controle ξ para o sistema (2.33).

2.2.2 Lei de controle não linear por realimentação de estados

Seja o sistema em malha aberta dado pela Equação (2.32), que desconsidera a cinemática das rodas e, consequentemente, o deslizamento longitudinal. Nesta seção, o deslizamento lateral também

é considerado nulo, $\sigma = 0$, e a entrada de referência $\eta_{\mathrm{r}} = (v_{\mathrm{r}}, \omega_{\mathrm{r}})^T$ é constante. Expandindo a Equação (2.32), tem-se

$$\begin{pmatrix} \dot{e}_1(t) \\ \dot{e}_2(t) \\ \dot{e}_3(t) \end{pmatrix} = \begin{pmatrix} \omega(t)e_2(t) + v_{\mathrm{r}} \cos e_3(t) - v_x(t) \\ -\omega(t)e_1(t) + v_{\mathrm{r}} \sin e_3(t) \\ \omega_{\mathrm{r}} - \omega(t) \end{pmatrix} \tag{2.34}$$

Para a dinâmica dada por (2.34), em que η_{r} é constante, os autores em Kim e Oh (1998) mostraram que a entrada de controle $\eta(t) = (v_x(t), \omega(t))^T$ dada pela lei de realimentação de estados

$$\begin{aligned} v_x(t) &= v_{\mathrm{r}} \cos e_3(t) - k_3 e_3(t)\omega(t) + k_1 e_1(t) \\ \omega(t) &= \omega_{\mathrm{r}} + \frac{v_{\mathrm{r}}}{2}\left[k_2\left(e_2(t) + k_3 e_3(t)\right) + \frac{1}{k_3}\sin e_3(t)\right] \end{aligned} \tag{2.35}$$

com constantes $v_{\mathrm{r}} > 0$ e $k_i > 0$, $i = 1, 2, 3$, leva o erro de postura $e(t)$ para zero com $t \to \infty$. Para mostrar tal resultado, é utilizado o Teorema de Lyapunov para sistemas autônomos, reproduzido neste trabalho no Teorema 3 do Apêndice A.

Substituindo a lei de controle dada pelas Equações (2.35) no sistema da Equação (2.34), obtém-se

$$\dot{e}(t) = f(e(t)) \tag{2.36}$$

em que

$$f(e(t)) = \begin{pmatrix} -k_1 e_1(t) + \frac{k_2 v_{\mathrm{r}}}{2}(e_2(t) + k_3 e_3(t))^2 + \omega_{\mathrm{r}}(e_2(t) + k_3 e_3(t)) + \frac{v_{\mathrm{r}}}{2k_3}(e_2(t) + k_3 e_3(t)) \sin e_3(t) \\ -\omega_{\mathrm{r}} e_1(t) - \frac{k_2 v_{\mathrm{r}}}{2} e_1(t)(e_2(t) + k_3 e_3(t)) - \frac{v_{\mathrm{r}}}{2k_3} e_1(t) \sin e_3(t) + v_{\mathrm{r}} \sin e_3(t) \\ -\frac{k_2 v_{\mathrm{r}}}{2}(e_2(t) + k_3 e_3(t)) - \frac{v_{\mathrm{r}}}{2k_3} \sin e_3(t) \end{pmatrix}$$

A Equação (2.36) representa a dinâmica do erro de postura do robô em malha fechada. Um sistema de equações diferenciais ordinárias como o da Equação (2.36) é dito autônomo, pois a função f não depende explicitamente do tempo. Observa-se que quando $e(t) = 0$, $f(e(t)) = 0$. Diz-se com isso que $e(t) = 0$ é um ponto de equilíbrio do sistema autônomo da Equação (2.36).

Para estudar a estabilidade da origem do sistema (2.36), os autores em Kim e Oh (1998) propuseram a função de Lyapunov $V(e(t))$ dada por

$$V(e(t)) = \frac{1}{2} e_1(t)^2 + \frac{1}{2}\left(e_2(t) + k_3 e_3(t)\right)^2 + \frac{(1 - \cos e_3(t))}{k_2} \tag{2.37}$$

Essa função de Lyapunov, bem como a lei de controle (2.35), foram também utilizadas em Iossaqui et al. (2011) e Iossaqui et al. (2013).

Após algumas simplificações, cujos detalhes podem ser encontrados em Iossaqui (2013), a derivada de $V(e(t))$ em relação ao tempo é dada por

$$\dot{V}(e(t)) = \frac{\partial V}{\partial e}\dot{e}(t) = \frac{\partial V}{\partial e}f(e(t)) = -k_1 e_1(t)^2 - \frac{v_\mathrm{r}}{2}k_2 k_3 (e_2(t) + k_3 e_3(t))^2 - \frac{v_\mathrm{r}}{2k_2 k_3}\sin^2 e_3(t)$$

Verifica-se que, no domínio $D = \{e(t) \in \mathbb{R}^3 \mid -\pi < e_3(t) < \pi\}$, a função $V(e(t))$ dada por (2.37) satisfaz $V(0) = 0$ e $V(e(t)) > 0$ em $D - \{0\}$. Diz-se com isso que $V(e(t))$ é definida positiva em D. Além disso, se $v_\mathrm{r} > 0$, a função $\dot{V}(e(t))$ acima satisfaz $\dot{V}(0) = 0$ e $\dot{V}(e(t)) < 0$ em $D - \{0\}$. Diz-se com isso que $\dot{V}(e(t))$ é definida negativa em D. Portanto, pelo Teorema 3 do Apêndice A, o ponto de equilíbrio $e(t) = 0$ é assintoticamente estável. De acordo com a Definição 7 do Apêndice A, isso significa que, para um erro de postura inicial suficientemente pequeno, o erro $e(t)$ convergirá para zero, ou seja, $e(t) \to 0$ com $t \to \infty$.

Logo, a entrada de controle dada pela lei de realimentação de estados (2.35) garante o rastreamento de trajetórias quando $v_\mathrm{r} > 0$ e ω_r são constantes e o robô não sofre deslizamento lateral nem longitudinal.

2.2.3 Lei de controle adaptativa não linear

Seja o sistema em malha aberta da Equação (2.33), que leva em conta a cinemática das rodas. Nesta seção, o deslizamento lateral é considerado nulo, $\sigma = 0$, a entrada de referência $\eta_\mathrm{r}(t) = (v_\mathrm{r}(t), \omega_\mathrm{r}(t))^T$ é variante no tempo, e o deslizamento longitudinal $a = (a_e, a_d)^T$ é considerado constante, mas desconhecido. Assim, tem-se

$$\dot{e}(t) = -\Omega(\omega(t))e(t) + R(e_3(t))S(0)\eta_\mathrm{r}(t) - S(0)\Phi(a)\xi(t) \tag{2.38}$$

em que $R(\cdot)$ é a matriz de rotação (2.4), $S(\cdot)$ é dada por (2.7), $\Phi(a)$ é dada por (2.22), $\Omega(\omega)$ é dada por (2.29), ω é a segunda componente de $\Phi(a)\xi(t)$ e $\xi(t) = (\omega_e(t), \omega_d(t))^T$ é a entrada de controle.

A introdução da cinemática das rodas e, consequentemente, do deslizamento longitudinal, torna o problema de rastreamento de trajetórias mais complicado do que o problema da Seção (2.2.2).

É interessante notar que se fosse possível medir com exatidão os parâmetros de deslizamento longitudinal, uma escolha óbvia para a entrada de controle do sistema (2.38) seria $\xi(t) = \Phi(a)^{-1}\eta_c(t)$, com $\eta_c(t) = (v_c(t), \omega_c(t))^T$ uma entrada de controle auxiliar e $\Phi(a)^{-1}$ dada pela Equação (2.23). Essa escolha na prática cancelaria totalmente o efeito do deslizamento longitudinal, deixando o sistema em malha aberta com a entrada de controle auxiliar na forma do sistema (2.32). Portanto, bastaria definir η_c como sendo a lei de controle (2.35) para que o problema de rastreamento de trajetórias estivesse resolvido.

Na prática, é muito difícil de se medir com precisão os parâmetros de deslizamento a_e e a_d. Como alternativa, a lei de controle adaptativa proposta em Iossaqui et al. (2011), Iossaqui et al. (2013) e Iossaqui (2013) resolve o problema de rastreamento de trajetórias utilizando estimadores dos parâmetros de deslizamento longitudinal. Isso é feito ao custo de ter que supor a_e e a_d constantes.

Seja $\hat{a} = (\hat{a}_e, \hat{a}_d)^T$, em que $\hat{a}_e$ e $\hat{a}_d$ são os parâmetros estimados de deslizamento longitudinal da roda esquerda e da roda direita, respectivamente. Então, motivando-se pela discussão sobre o cancelamento do efeito do deslizamento longitudinal, será utilizada a entrada de controle $\xi(t)$ dada por

$$\xi(t) = \Phi(\hat{a}(t))^{-1}\eta_c(t) \tag{2.39}$$

em que $\eta_c(t) = (v_c(t), \omega_c(t))^T$ é o vetor de entrada de controle auxiliar e $\Phi(\hat{a}(t))^{-1}$ é dada por

$$\Phi(\hat{a}(t))^{-1} = \begin{bmatrix} \dfrac{\hat{a}_e(t)}{r} & \dfrac{-b\hat{a}_e(t)}{2r} \\ \dfrac{\hat{a}_d(t)}{r} & \dfrac{b\hat{a}_d(t)}{2r} \end{bmatrix} \tag{2.40}$$

Substituindo a Equação (2.39) no sistema em malha aberta da Equação (2.38), obtém-se

$$\dot{e}(t) = -\Omega(\omega(t))e(t) + R(e_3(t))S(0)\eta_{\mathrm{r}}(t) - S(0)\Phi(a)\Phi(\hat{a}(t))^{-1}\eta_c(t) \tag{2.41}$$

Para o sistema (2.41), a entrada de controle auxiliar $\eta_c(t)$ é dada pela lei de controle da Equação (2.35), porém com $\eta_{\mathrm{r}}(t)$ variante no tempo, ou seja

$$\begin{aligned} v_c(t) &= v_{\mathrm{r}}(t)\cos e_3(t) - k_3 e_3(t)\omega_c(t) + k_1 e_1(t) \\ \omega_c(t) &= \omega_{\mathrm{r}}(t) + \frac{v_{\mathrm{r}}(t)}{2}\left[k_2\left(e_2(t) + k_3 e_3(t)\right) + \frac{1}{k_3}\sin e_3(t)\right] \end{aligned} \tag{2.42}$$

Os erros de estimação dos parâmetros de deslizamento longitudinal são dados por

$$\begin{aligned} \tilde{a}_e(t) &= \hat{a}_e(t) - a_e \\ \tilde{a}_d(t) &= \hat{a}_d(t) - a_d \end{aligned} \tag{2.43}$$

Enquanto a dinâmica de $e(t)$ é dada pela Equação (2.41), a dinâmica dos erros de estimação é dada pela derivada em relação ao tempo da Equação (2.43). Como a_e e a_d são constantes nesta seção, tem-se

$$\begin{aligned} \dot{\tilde{a}}_e(t) &= \dot{\hat{a}}_e(t) \\ \dot{\tilde{a}}_d(t) &= \dot{\hat{a}}_d(t) \end{aligned} \tag{2.44}$$

Observa-se que quando o erro de estimação é nulo, tem-se $\hat{a} = a$, e o efeito do deslizamento é cancelado na Equação (2.41). Com isso em mente, é definido um vetor de estados aumentado, $e_a(t) = (e(t)^T, \tilde{a}_e(t), \tilde{a}_d(t))^T = (e_1(t), e_2(t), e_3(t), \tilde{a}_e(t), \tilde{a}_d(t))^T$.

Como proposto por Iossaqui et al. (2011), Iossaqui et al. (2013) e Iossaqui (2013), é possível encontrar uma lei de adaptação para $\hat{a}_e(t)$ e $\hat{a}_d(t)$ através da análise da seguinte função de Lyapunov:

$$V_a(e_a(t)) = V(e(t)) + \frac{\tilde{a}_e(t)^2}{2\gamma_1 a_e} + \frac{\tilde{a}_d(t)^2}{2\gamma_2 a_d} \tag{2.45}$$

em que $V(e(t))$ é dada pela Equação (2.37) e os ganhos de adaptação $\gamma_1 > 0$ e $\gamma_2 > 0$ são constantes.

A derivada em relação ao tempo, $\dot{V}_a(e_a(t), t)$, é calculada ao longo das trajetórias da dinâmica do erro aumentado, Equações (2.41) e (2.44), em que $\eta_c(t)$ na Equação (2.41) é dado pela lei de controle (2.42) e $\hat{a}_e(t)$ e $\hat{a}_d(t)$ na matriz $\Phi(\hat{a}(t))^{-1}$ são dados pela Equação (2.43). O cálculo dessa derivada, que está detalhado em Iossaqui (2013), resulta em

$$\begin{aligned} \dot{V}_a(t, e_a(t)) = &-k_1 e_1(t)^2 - \frac{v_{\mathrm{r}}(t)}{2} k_2 k_3 (e_2(t) + k_3 e_3(t))^2 - \frac{v_{\mathrm{r}}(t)}{2k_2 k_3} \sin^2 e_3(t) \\ &+ \frac{\tilde{a}_e(t)}{a_e} \left\{ \frac{\dot{\hat{a}}_e(t)}{\gamma_1} - \left(v_c(t) - \frac{b}{2}\omega_c(t) \right) \left[\left(\frac{e_2(t)}{b} + \frac{1}{2} \right) e_1(t) - \left(\frac{e_1(t)}{b} + \frac{k_3}{b} \right) (e_2(t) + k_3 e_3(t)) - \frac{1}{bk_2} \sin e_3(t) \right] \right\} \\ &+ \frac{\tilde{a}_d(t)}{a_d} \left\{ \frac{\dot{\hat{a}}_d(t)}{\gamma_2} - \left(v_c(t) + \frac{b}{2}\omega_c(t) \right) \left[- \left(\frac{e_2(t)}{b} - \frac{1}{2} \right) e_1(t) + \left(\frac{e_1(t)}{b} + \frac{k_3}{b} \right) (e_2(t) + k_3 e_3(t)) + \frac{1}{bk_2} \sin e_3(t) \right] \right\} \end{aligned} \tag{2.46}$$

Observa-se que para a lei de adaptação proposta em Iossaqui et al. (2011) dada por

$$\begin{aligned}\dot{\hat{a}}_e(t) &= \gamma_1 \left(v_c(t) - \frac{b}{2}\omega_c(t)\right)\left[\left(\frac{e_2(t)}{b} + \frac{1}{2}\right)e_1(t) - \left(\frac{e_1(t)}{b} + \frac{k_3}{b}\right)(e_2(t) + k_3 e_3(t)) - \frac{1}{bk_2}\sin e_3(t)\right] \\ \dot{\hat{a}}_d(t) &= \gamma_2 \left(v_c(t) + \frac{b}{2}\omega_c(t)\right)\left[-\left(\frac{e_2(t)}{b} - \frac{1}{2}\right)e_1(t) + \left(\frac{e_1(t)}{b} + \frac{k_3}{b}\right)(e_2(t) + k_3 e_3(t)) + \frac{1}{bk_2}\sin e_3(t)\right]\end{aligned} \tag{2.47}$$

a derivada da Equação (2.46) se torna

$$\dot{V}_a(t, e_a(t)) = -k_1 e_1(t)^2 - \frac{v_\mathrm{r}(t)}{2} k_2 k_3 (e_2(t) + k_3 e_3(t))^2 - \frac{v_\mathrm{r}(t)}{2k_2 k_3}\sin^2 e_3(t) \tag{2.48}$$

É importante salientar que o controlador adaptativo fica sendo dado pelo conjunto das Equações (2.39), (2.40), (2.42) e (2.47). A Figura 2.3 ilustra o diagrama de blocos do controlador adaptativo.

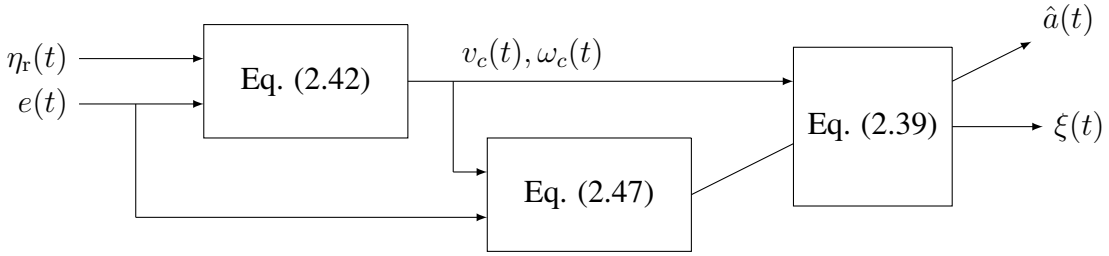

Figura 2.3: Esquema do controlador adaptativo não linear.

Com esse controlador, a dinâmica do erro em malha fechada, Equações (2.41) e (2.44), fica sendo dada por

$$\dot{e}_a(t) = f_a(t, e_a(t)) \tag{2.49}$$

em que

$$f_a(t, e_a(t)) = \begin{pmatrix} \left(1 + \frac{\tilde{a}_d(t)}{a_d}\right)\left(\frac{e_2(t)}{b} - \frac{1}{2}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right) + v_\mathrm{r}(t)\cos e_3(t) - \left(1 + \frac{\tilde{a}_e(t)}{a_e}\right)\left(\frac{e_2(t)}{b} + \frac{1}{2}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right) \\ \left(1 + \frac{\tilde{a}_e(t)}{a_e}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right)\frac{e_1(t)}{b} + v_\mathrm{r}(t)\sin e_3(t) - \left(1 + \frac{\tilde{a}_d(t)}{a_d}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right)\frac{e_1(t)}{b} \\ \omega_\mathrm{r}(t) - \frac{1}{b}\left(1 + \frac{\tilde{a}_d(t)}{a_d}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right) + \frac{1}{b}\left(1 + \frac{\tilde{a}_e(t)}{a_e}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right) \\ \gamma_1\left(v_c(t) - \frac{b}{2}\omega_c(t)\right)\left[\left(\frac{e_2(t)}{b} + \frac{1}{2}\right)e_1(t) - \left(\frac{e_1(t)}{b} + \frac{k_3}{b}\right)(e_2(t) + k_3 e_3(t)) - \frac{1}{bk_2}\sin e_3(t)\right] \\ \gamma_2\left(v_c(t) + \frac{b}{2}\omega_c(t)\right)\left[-\left(\frac{e_2(t)}{b} - \frac{1}{2}\right)e_1(t) + \left(\frac{e_1(t)}{b} + \frac{k_3}{b}\right)(e_2(t) + k_3 e_3(t)) + \frac{1}{bk_2}\sin e_3(t)\right] \end{pmatrix}$$

com $v_c(t)$ e $\omega_c(t)$ dados pela lei de controle (2.42). O desenvolvimento das três primeiras linhas da Equação (2.49), a partir da Equação (2.41), pode ser encontrado na Seção B.1 do Apêndice. Nota-se que as duas últimas linhas são dadas pela lei de adaptação (2.47).

A função f_a do sistema de Equações diferenciais ordinárias (2.49) depende explicitamente do tempo. Tais sistemas são ditos não autônomos. Além disso, vê-se que para $e_a(t) = 0$, tem-se $f(t, e_a(t)) = 0$, de modo que a origem $e_a(t) = 0$ é um ponto de equilíbrio do sistema não autônomo (2.49). O Teorema 4 do Apêndice A será utilizado para mostrar que o erro de postura $e(t) \to 0$ com $t \to \infty$, garantindo o rastreamento de trajetórias, através da escolha da lei de adaptação (2.47). Com esse intuito, definem-se as funções

$$W_1(e_a(t)) = W_2(e_a(t)) = V_a(e_a(t))$$

em que $V_a(e_a(t))$ é dada por (2.45). No domínio $D = \{e_a(t) \in \mathbb{R}^5 \mid -\pi < e_3(t) < \pi\}$, tem-se que W_1 e W_2 são definidas positivas. Para uma determinada constante $\mu > 0$, seja a função

$$W(e_a(t)) = k_1 e_1(t)^2 + \frac{k_2 k_3 \mu}{2}(e_2(t) + k_3 e_3(t))^2 + \frac{\mu}{2k_2 k_3}\sin^2 e_3(t) \tag{2.50}$$

Tem-se que $W(e_a(t)) \geq 0$ para todo $e_a \in D$. Diz-se com isso que W é semidefinida positiva em D.

Impondo sobre a referência variante no tempo a condição $0 < \mu \leq v_{\mathrm{r}}(t)$, tem-se que

$$\dot{V}_a(t, e_a(t)) \leq -W(e_a(t))$$

em que $\dot{V}_a(t, e_a(t))$ é dada pela Equação (2.48).

Com isso, as condições do Teorema 4 são satisfeitas. Conclui-se portanto que, para $e_a(t_0)$ suficientemente pequeno, todas as soluções do sistema em malha fechada (2.49) são limitadas e satisfazem $W(e_a(t)) \to 0$ com $t \to \infty$. Analisando $W(e_a(t))$, tem-se como consequência que o erro de postura $e(t) \to 0$ com $t \to \infty$. Observa-se que isso implica que apenas três dos estados de e_a convergem para zero. Porém, sobre os dois outros estados, $\tilde{a}_e$ e $\tilde{a}_d$, só se pode afirmar que são limitados.

Assim, o controlador adaptativo dado pelas Equações (2.39), (2.40), (2.42) e (2.47) resolve o problema de rastreamento de trajetórias formulado na Seção 2.2.1 quando o robô sofre apenas deslizamentos longitudinais nas rodas. No entanto, tal resultado é garantido apenas se os parâmetros de deslizamento longitudinais a_d e a_e forem constantes e se a velocidade de referência $v_{\mathrm{r}}(t)$ for limitada inferiormente por um valor constante positivo, ou seja, $0 < \mu \leq v_{\mathrm{r}}(t)$.

A seguir, será analisado o que acontece quando o deslizamento lateral deixa de ser nulo e o deslizamento longitudinal é variante no tempo.

2.3 Análise do robô móvel sujeito a deslizamentos laterais e longitudinais

Nesta seção, será considerado que o robô móvel de tração diferencial sofre um deslizamento lateral variante no tempo. Além disso, os parâmetros de deslizamento longitudinal também serão considerados variantes no tempo. Ambas essas considerações fazem com que o sistema em malha fechada se torne um sistema não linear perturbado. Assim, serão usados teoremas específicos para a análise de sistemas não lineares perturbados (Khalil, 2002, Rosenbrock, 1963).

2.3.1 Dinâmica não linear perturbada

Seja o sistema em malha aberta dado por (2.33). Os parâmetros de deslizamento lateral σ, de deslizamento longitudinal $a = (a_e, a_d)^T$ e a entrada de referência $\eta_{\mathrm{r}} = (v_{\mathrm{r}}, \omega_{\mathrm{r}})^T$ são considerados variantes no tempo. Utilizando a lei de controle $\xi = (\omega_e, \omega_d)^T$ dada pela Equação (2.39), a dinâmica do erro de postura fica sendo

$$\dot{e}(t) = -\Omega(\omega(t))e(t) + R(e_3(t))S(0)\eta_{\mathrm{r}}(t) - S(\sigma(t))\Phi(a(t))\Phi(\hat{a}(t))^{-1}\eta_c(t) \tag{2.51}$$

em que $R(\cdot)$ é a matriz de rotação (2.4), $S(\cdot)$ é dada por (2.7), $\Phi(a(t))$ é dada por (2.22), $\Omega(\omega)$ é dada por (2.29), $\Phi(\hat{a}(t))^{-1}$ é dada por (2.40), ω é a segunda componente de $\Phi(a(t))\Phi(\hat{a}(t))^{-1}\eta_c(t)$ e a lei $\eta_c(t) = (v_c(t), \omega_c(t))^T$ é dada por (2.42).

Se os parâmetros de deslizamento longitudinal são considerados variantes no tempo, ao derivar a Equação (2.43) em relação ao tempo, obtém-se

$$\begin{aligned} \dot{\tilde{a}}_e(t) &= \dot{\hat{a}}_e(t) - \dot{a}_e(t) \\ \dot{\tilde{a}}_d(t) &= \dot{\hat{a}}_d(t) - \dot{a}_d(t) \end{aligned} \tag{2.52}$$

Como na Seção 2.2.3, os termos $\dot{\hat{a}}_e(t)$ e $\dot{\hat{a}}_d(t)$ são dados pela lei de adaptação da Equação (2.47). Porém, desta vez, os termos desconhecidos $\dot{a}_e(t)$ e $\dot{a}_d(t)$ não são necessariamente nulos. Na

abordagem utilizada nesta seção, os termos $\dot{a}_e(t)$, $\dot{a}_d(t)$ e $\sigma(t)$ serão tratados como perturbações externas ao sistema.

Considerando (2.51) e (2.52), a dinâmica do erro aumentado $e_a = (e_1, e_2, e_3, \tilde{a}_e, \tilde{a}_d)^T$ pode ser escrita como o seguinte sistema não autônomo perturbado:

$$\dot{e}_a(t) = f_a(t, e_a(t)) + g(t, e_a(t)) + g_n(t, e_a(t)) \tag{2.53}$$

em que $f_a(t, e_a(t))$ é chamado de termo nominal,

$$f_a(t, e_a(t)) = \begin{pmatrix} \left(1 + \frac{\tilde{a}_d(t)}{a_d(t)}\right)\left(\frac{e_2(t)}{b} - \frac{1}{2}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right) + v_r(t)\cos e_3(t) - \left(1 + \frac{\tilde{a}_e(t)}{a_e(t)}\right)\left(\frac{e_2(t)}{b} + \frac{1}{2}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right) \\ \left(1 + \frac{\tilde{a}_e(t)}{a_e(t)}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right)\frac{e_1(t)}{b} + v_r(t)\sin e_3(t) - \left(1 + \frac{\tilde{a}_d(t)}{a_d(t)}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right)\frac{e_1(t)}{b} \\ \omega_r(t) - \frac{1}{b}\left(1 + \frac{\tilde{a}_d(t)}{a_d(t)}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right) + \frac{1}{b}\left(1 + \frac{\tilde{a}_e(t)}{a_e(t)}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right) \\ \gamma_1\left(v_c(t) - \frac{b}{2}\omega_c(t)\right)\left[\left(\frac{e_2(t)}{b} + \frac{1}{2}\right)e_1(t) - \left(\frac{e_1(t)}{b} + \frac{k_3}{b}\right)(e_2(t) + k_3 e_3(t)) - \frac{1}{bk_2}\sin e_3(t)\right] \\ \gamma_2\left(v_c(t) + \frac{b}{2}\omega_c(t)\right)\left[-\left(\frac{e_2(t)}{b} - \frac{1}{2}\right)e_1(t) + \left(\frac{e_1(t)}{b} + \frac{k_3}{b}\right)(e_2(t) + k_3 e_3(t)) + \frac{1}{bk_2}\sin e_3(t)\right] \end{pmatrix} \tag{2.54}$$

o termo $g(t, e_a(t))$ é chamado de perturbação evanescente, que é nula em $e_a = 0$,

$$g(t, e_a(t)) = \begin{pmatrix} 0 \\ -\sigma(t)\left[\frac{1}{4}\frac{\tilde{a}_e(t)}{a_e(t)}(2v_c(t) - b\omega_c(t)) + \frac{1}{4}\frac{\tilde{a}_d(t)}{a_d(t)}(2v_c(t) + b\omega_c(t)) - k_3 e_3(t)\omega_c(t) + k_1 e_1(t)\right] \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.55}$$

e o termo $g_n(t, e_a(t))$ é chamado de perturbação não evanescente, que pode não ser nula em $e_a = 0$,

$$g_n(t, e_a(t)) = \begin{pmatrix} 0 \\ -\sigma(t)v_r(t)\cos e_3(t) \\ 0 \\ -\dot{a}_e(t) \\ -\dot{a}_d(t) \end{pmatrix} \tag{2.56}$$

Nota-se que $v_c(t)$ e $\omega_c(t)$ são dados pelas leis de controle da Equação (2.42). O desenvolvimento dos termos acima, a partir das Equações (2.51) e (2.52), pode ser encontrado na Seção B.1 do Apêndice.

Observa-se que a perturbação não evanescente (2.56) pode não ser nula na origem. Isso implica que a origem $e_a(t) = 0$ pode não ser um ponto de equilíbrio do sistema perturbado da Equação (2.53). Consequentemente, não é possível analisar a origem $e_a(t)$ como um ponto de equilíbrio do sistema (2.53).

Quando a origem não pode ser estudada como ponto de equilíbrio, não se deve esperar que as soluções do sistema perturbado convirjam para a origem com $t \to \infty$. Uma alternativa razoável é tentar mostrar que as soluções do sistema se mantém limitadas a uma região em torno da origem. De forma específica, isso se resumiria a tentar mostrar que $e_a(t)$ satisfaz $\|e_a(t)\| < \ell$, para um certo ℓ independente de t_0, para todo $t \geq t_0 + T$, com T um período finito de tempo. Tais soluções são ditas uniformemente finalmente limitadas, como descrito na Definição 1.

Definição 1 (Khalil (2002), Definição 4.6)**.** *As soluções do sistema* (2.53) *são uniformemente finalmente limitadas com limitante final ℓ_f se existem constantes positivas ℓ_f e c, independentes de $t_0 \geq 0$, e para todo $r \in (0, c)$, existe um $T = T(r, \ell_f) \geq 0$, independente de t_0, tal que*

$$\|e_a(t_0)\| \leq r \implies \|e_a(t)\| \leq \ell_f, \quad \forall\, t \geq t_0 + T$$

Com o intuito de mostrar que a solução $e_a(t)$ do sistema perturbado (2.53) é uniformemente finalmente limitada, serão aplicados diversos teoremas de análise de estabilidade de sistemas perturbados. Esses teoremas serão aplicados em três passos. Primeiro, será mostrado que a origem do chamado sistema nominal, dado por

$$\dot{e}_a(t) = f_a(t, e_a(t)) \tag{2.57}$$

em que f_a é dada pela Equação (2.54), é exponencialmente estável. Note que este é um resultado mais forte que o resultado estudado na Seção 2.2.3, em que se mostrou apenas a convergência de parte dos estados à origem. Esse passo será desenvolvido na Seção 2.3.2. Em seguida, na Seção 2.3.3, será provado que a origem do sistema nominal (2.57), perturbado apenas pela perturbação evanescente (2.55), continua sendo exponencialmente estável, desde que certas condições sejam respeitadas. Finalmente, na Seção 2.3.4, será mostrado sob quais condições as soluções do sistema perturbado (2.53) são uniformemente finalmente limitadas.

2.3.2 Análise do sistema nominal

Nesta seção, será mostrado que a origem do sistema nominal (2.57) é exponencialmente estável. De acordo com a Definição 2 abaixo, isso implica que as soluções $e_a(t)$ do sistema nominal tenderão a zero com rapidez exponencial. Essa análise será feita seguindo ideias similares às usadas em Iossaqui et al. (2013).

Definição 2 (Khalil (2002), Definição 4.5)**.** *O ponto de equilíbrio* $e_a(t) = 0$ *de* (2.57) *é exponencialmente estável se existem constantes* c*,* k *e* r *tais que*

$$\|e_a(t)|| \leq c\|e_a(t_0)\| \, \mathrm{e}^{-k(t-t_0)}, \quad \forall \, \|e_a(t_0)\| \leq r$$

Para mostrar que a origem do sistema nominal (2.57) é exponencialmente estável, será aplicado o Teorema 5, do Apêndice A, que se baseia na linearização do sistema nominal. O Teorema 5 pode ser utilizado desde que a matriz Jacobiana $\partial f_a/\partial e_a$ seja limitada, uniformemente em t, para todo e_a contido numa bola de raio finito $B = \{e_a(t) \in \mathbb{R}^5 \mid \|e_a(t)\|_2 < d\}$. Como $f_a(t, e_a)$ é composta por somas e produtos de polinômios e senoides em e_a, suas derivadas parciais em relação a e_a também o são. Assim, se e_a estiver confinado a uma região limitada, a Jacobiana estará limitada como função de e_a. Porém, para que a Jacobiana seja limitada uniformemente em t, é necessário limitar as velocidades de referência v_r e ω_r. Portanto, será estabelecido daqui em diante que as velocidades de referência possuem um limitante superior

$$\begin{aligned} |v_\mathrm{r}(t)| &\leq L_v, \quad \forall t \geq t_0 \\ |\omega_\mathrm{r}(t)| &\leq L_\omega, \quad \forall t \geq t_0 \end{aligned} \tag{2.58}$$

em que L_v e L_ω são constantes. Notando que $f_a(t, e_a)$ é infinitamente diferenciável em e_a em $\mathbb{R}^5$, para garantir as condições de continuidade exigidas pelo Teorema 5, ou seja, que $f_a(t, e_a) : [0, \infty) \times B \rightarrow \mathbb{R}^5$ seja continuamente diferenciável e $\partial f_a/\partial e_a$ seja Lipschitz em uma bola de raio finito $B = \{e_a(t) \in \mathbb{R}^5 \mid \|e_a(t)\|_2 < d\}$, será assumido que $v_\mathrm{r}(t)$, $\omega_\mathrm{r}(t)$, $a_d(t)$ e $a_e(t)$ são continuamente diferenciáveis em t para $t \in [0, \infty)$.

Feitas as considerações acima, o Teorema 5 garante que $e_a(t) = 0$ é um ponto de equilíbrio exponencialmente estável do sistema nominal (2.57) se e somente se $e_a(t) = 0$ é um ponto de equilíbrio exponencialmente estável do sistema linear variante no tempo

$$\dot{e}_a(t) = A(t)e_a(t) \tag{2.59}$$

em que $A(t)$ é obtida através da linearização de (2.57) em torno da origem. A matriz $A(t)$ é dada por

$$A(t) = \begin{bmatrix} -k_1 & \omega_{\mathrm{r}}(t) & k_3\omega_{\mathrm{r}}(t) & \dfrac{v_2(t)}{4a_e(t)} & -\dfrac{v_1(t)}{4a_d(t)} \\ -\omega_{\mathrm{r}}(t) & 0 & v_{\mathrm{r}}(t) & 0 & 0 \\ 0 & -\dfrac{k_2 v_{\mathrm{r}}(t)}{2} & -\dfrac{k_4 v_{\mathrm{r}}(t)}{2k_3} & -\dfrac{v_2(t)}{2ba_e(t)} & -\dfrac{v_1(t)}{2ba_d(t)} \\ -\dfrac{\gamma_1 v_2(t)}{4} & \dfrac{\gamma_1 k_3 v_2(t)}{2b} & \dfrac{\gamma_1 k_4 v_2(t)}{2bk_2} & 0 & 0 \\ \dfrac{\gamma_2 v_1(t)}{4} & \dfrac{\gamma_2 k_3 v_1(t)}{2b} & \dfrac{\gamma_2 k_4 v_1(t)}{2bk_2} & 0 & 0 \end{bmatrix} \tag{2.60}$$

com

$$k_4 = 1 + k_2 k_3^2, \quad v_1(t) = b\omega_{\mathrm{r}}(t) + 2v_{\mathrm{r}}(t) \quad \text{e} \quad v_2(t) = b\omega_{\mathrm{r}}(t) - 2v_{\mathrm{r}}(t)$$

Para mostrar que a origem $e_a(t) = 0$ do sistema variante no tempo (2.59) é exponencialmente estável, pode-se aplicar o Teorema 6 do Apêndice A. Para aplicar o Teorema 6, é necessário que todo elemento $a_{ij}(t)$ de $A(t)$, dada em (2.60), seja diferenciável e satisfaça $|a_{ij}| \leq \rho$, para um ρ finito qualquer. Como foi assumido anteriormente que $v_{\mathrm{r}}(t)$, $\omega_{\mathrm{r}}(t)$, $a_d(t)$ e $a_e(t)$ são continuamente diferenciáveis para todo o tempo, todos os elementos de $A(t)$ são diferenciáveis. Além disso, tendo em vista os limitantes impostos sobre v_{r} e ω_{r} em (2.58), e lembrando que $a_e, a_d \in [1, \infty)$, fica claro que todo elemento de $A(t)$ é limitado, para todo o tempo, por um valor constante. Assim, existe uma constante ρ que satisfaz $|a_{ij}| \leq \rho$.

Continuando a aplicação do Teorema 6, resta mostrar que as partes reais dos autovalores de $A(t)$ satisfazem a Equação (A.8), i.e., que elas são limitadas por uma constante negativa. Isso será feito a partir do polinômio característico de $A(t)$, com o uso do Critério de Estabilidade de Liénard e Chipart (Gantmacher, 1959). Nota-se que nessa análise, o polinômio característico e os autovalores de $A(t)$ são utilizados para valores fixos de t. O polinômio característico da matriz $A(t)$ é dado por

$$p(s) = s^5 + \alpha_1 s^4 + \alpha_2 s^3 + \alpha_3 s^2 + \alpha_4 s + \alpha_5 \tag{2.61}$$

em que

$$\alpha_1 = \frac{2k_1k_3 + k_4v_\mathrm{r}}{2k_3} \tag{2.62}$$

$$\alpha_2 = \frac{1}{16a_ea_db^2k_2k_3}\big[k_3(v_1^2a_e\gamma_2 + v_2^2a_d\gamma_1)(b^2k_2 + 4k_4) + 8a_ea_db^2k_2(v_\mathrm{r}(k_1k_4 + k_2k_3v_\mathrm{r}) + 2k_3\omega_\mathrm{r}^2)\big] \tag{2.63}$$

$$\alpha_3 = \frac{1}{32a_ea_db_2^2k_2k_3}\big[(v_1^2a_e\gamma_2 + v_2^2a_d\gamma_1)(k_2v_\mathrm{r}(b^2k_4 + 8k_3^2) + 8k_1k_3k_4) + 16a_ea_db^2k_2v_\mathrm{r}(k_1k_2k_3v_\mathrm{r} + \omega_\mathrm{r}^2)\big] \tag{2.64}$$

$$\alpha_4 = \frac{1}{32a_ea_db^2k_2}\big[v_1^2\gamma_2a_e((bk_2v_\mathrm{r} + 2\omega_\mathrm{r})^2 + 4\omega_\mathrm{r}^2 + 8k_1k_2k_3v_\mathrm{r}) + v_2^2\gamma_1a_d((bk_2v_\mathrm{r} - 2\omega_\mathrm{r})^2 + 4\omega_\mathrm{r}^2 + 8k_1k_2k_3v_\mathrm{r}) + 2\gamma_1\gamma_2k_4v_1^2v_2^2\big] \tag{2.65}$$

$$\alpha_5 = \frac{\gamma_1\gamma_2k_3v_\mathrm{r}v_1^2v_2^2}{16a_ea_db^2} \tag{2.66}$$

com

$$k_4 = 1 + k_2k_3^2, \quad v_1 = b\omega_\mathrm{r} + 2v_\mathrm{r}, \quad \text{e} \quad v_2 = b\omega_\mathrm{r} - 2v_\mathrm{r}$$

Usando o Critério de Estabilidade de Liénard e Chipart, Teorema 8 do Apêndice A, é possível garantir que as partes reais dos autovalores de $A(t)$, para um dado t fixo, serão limitadas por um valor negativo caso exista um ε tal que as seguintes condições sejam satisfeitas para todo o tempo:

$$0 < \varepsilon \leq \alpha_i, \quad i = 1, \ldots, 5 \tag{2.67}$$

$$0 < \varepsilon \leq \Delta_2(\alpha) := \alpha_1\alpha_2 - \alpha_3 \tag{2.68}$$

$$0 < \varepsilon \leq \Delta_4(\alpha) := (\alpha_1\alpha_2 - \alpha_3)(\alpha_3\alpha_4 - \alpha_2\alpha_5) - (\alpha_1\alpha_4 - \alpha_5)^2 \tag{2.69}$$

Assume-se que a velocidade de referência obedeça

$$v_\mathrm{r}(t) \geq \mu > 0 \quad \text{e} \quad v_\mathrm{r}(t) \notin \left(\frac{b|\omega_\mathrm{r}(t)|}{2} - \zeta, \frac{b|\omega_\mathrm{r}(t)|}{2} + \zeta\right) \tag{2.70}$$

para todo o tempo, com μ e ζ constantes positivas arbitrariamente pequenas, e assume-se também que existem constantes L_e e L_d tais que os parâmetros de deslizamento longitudinal obedeçam

$$a_e(t) \leq L_e \quad \text{e} \quad a_d(t) \leq L_d \tag{2.71}$$

para todo o tempo.

Feitas essas suposições, lembrando que b, k_i e γ_i são constantes positivas e que $a_e \geq 1$, $a_d \geq 1$, e $v_{\mathrm{r}}(t) \geq \mu > 0$, tem-se que, por inspeção das Equações (2.62)-(2.66), pode-se escolher ε suficientemente pequeno tal que a condição (2.67) seja satisfeita. Nota-se, em particular, que devido à suposição (2.70), os termos $v_1(t)^2 = (b\omega_{\mathrm{r}} + 2v_{\mathrm{r}})^2$ e $v_2(t)^2 = (b\omega_{\mathrm{r}} - 2v_{\mathrm{r}})^2$ não se anulam nem tendem a zero. Semelhantemente, a condição (2.68) também é satisfeita para ε suficientemente pequeno, o que é constatado através da inspeção de

$$\Delta_2(\alpha) = \frac{1}{16a_e a_d b^2 k_2 k_3^2}\big[4a_e a_d b^2 k_2\big(k_1(k_4^2 v_{\mathrm{r}}^2 + 4k_3^2\omega_{\mathrm{r}}^2) + 2k_1^2 k_3 k_4 v_{\mathrm{r}} + k_2 k_3 v_{\mathrm{r}}(k_4 v_{\mathrm{r}}^2 + 2k_3^2\omega_{\mathrm{r}}^2)\big) + k_3(v_1^2 a_e\gamma_2 + v_2^2 a_d\gamma_1)\big(b^2 k_1 k_2 k_3 + 2(k_2^2 k_3^4 v_{\mathrm{r}} + v_{\mathrm{r}})\big)\big]$$

Mostrar que o termo $\Delta_4(\alpha)$ satisfaz a condição (2.69) constitui um desafio à parte. O tamanho da Equação que resulta da substituição de $\alpha_1, \ldots, \alpha_5$ em $\Delta_4(\alpha)$, e a presença de inúmeros termos negativos, torna impraticável tirar qualquer conclusão por mera inspeção. Na realidade, determinar a positividade de $\Delta_4(\alpha)$ é um problema de otimização extremamente difícil.

Alguns métodos numéricos foram utilizados para tentar resolver esse problema. Um dos métodos que se pode usar para testar a positividade de $\Delta_4(\alpha)$ é o método de força bruta, em que primeiramente se define um *grid* de valores para cada uma das variáveis do problema, respeitando as restrições $k_i > 0, \gamma_i > 0, b > 0$, $a_e \geq 1$, $a_d \geq 1$ e a Equação (2.70). Com o *grid*, pode-se testar o valor de $\Delta_4(\alpha)$ para todas as combinações possíveis de valores das variáveis. Esse método foi implementado numericamente utilizando uma discretização fina para cada uma das variáveis do problema. Verificou-se que o valor de $\Delta_4(\alpha)$ nunca se anula, desde que as condições mencionadas sobre as variáveis sejam impostas. A desvantagem desse método é que não há garantia que $\Delta_4(\alpha)$ seja positivo entre os pontos do *grid*. No entanto, como $\Delta_4(\alpha)$ é uma função racional em seus argumentos, para um *grid* que seja significativamente pequeno, há um indicativo forte de que $\Delta_4(\alpha)$ é de fato estritamente positivo. Por outro lado, em Iossaqui et al. (2013), o problema

$$\min_{b,k_i,\gamma_i,a_e,a_d,v_{\mathrm{r}},\omega_{\mathrm{r}}} \quad \Delta_4(\alpha)$$

sujeito a

$$b > 0, k_i > 0, \gamma_i > 0, a_e \geq 1, a_d \geq 1, v_{\mathrm{r}} > 0$$

foi abordado com o uso da função "NMinimize" do software Mathematica. Verificou-se que o mínimo atingido é sempre um número não negativo próximo de zero. Contudo, a função "NMinimize" pode não retornar um ótimo global.

A fim de fortalecer os resultados numéricos que apontam para a positividade de $\Delta_4(\alpha)$, este trabalho propõe utilizar uma abordagem de otimização polinomial através de relaxações SOS, que podem garantir resultados globais. Usando uma técnica SOS, será mostrado na Seção 4.2.3 que o termo $\Delta_4(\alpha)$ é limitado inferiormente por zero. Dado que $\det A(t) = -(k_3\gamma_1\gamma_2 v_\mathrm{r} v_1^2 v_2^2)/(16 a_e a_d b^2)$ não se anula nem tende a zero devido às suposições (2.70) e (2.71), pode-se concluir adicionalmente que nenhum autovalor da matriz $A(t)$ dada por (2.60) se encontra na origem ou tende a zero.

Logo, todas as condições dadas pelo Critério de Estabilidade de Liénard e Chipart são satisfeitas, e os autovalores de $A(t)$ satisfazem a condição (A.8) do Teorema 6. Portanto, existe um $\delta > 0$ tal que se $|\dot{a}_{ij}| \leq \delta$, a origem $e_a(t) = 0$ do sistema linear variante no tempo (2.59) é exponencialmente estável. Nota-se que o valor de δ não é conhecido, e pode ser difícil de ser calculado. Além disso, o valor de δ pode ser pequeno. Contudo, os $\dot{a}_{ij}$ de (2.60) dependem dos termos

$$\dot{v}_\mathrm{r}(t), \dot{\omega}_\mathrm{r}(t), \frac{\dot{v}_\mathrm{r}(t)}{a_d(t)}, \frac{\dot{v}_\mathrm{r}(t)}{a_e(t)}, \frac{\dot{\omega}_\mathrm{r}(t)}{a_d(t)}, \frac{\dot{\omega}_\mathrm{r}(t)}{a_e(t)}, \frac{v_\mathrm{r}(t)\dot{a}_e(t)}{a_e(t)^2}, \frac{v_\mathrm{r}(t)\dot{a}_d(t)}{a_d(t)^2}, \frac{\omega_\mathrm{r}(t)\dot{a}_e(t)}{a_e(t)^2} \quad \text{e} \quad \frac{\omega_\mathrm{r}(t)\dot{a}_d(t)}{a_d(t)^2}$$

multiplicados por constantes. Se existirem constantes M_e e M_d tais que

$$|\dot{a}_e(t)| \leq M_e \quad \text{e} \quad |\dot{a}_d(t)| \leq M_d \tag{2.72}$$

para todo o tempo, então, visto que tem-se liberdade para escolher $v_\mathrm{r}(t)$, $\omega_\mathrm{r}(t)$, $\dot{v}_\mathrm{r}(t)$ e $\dot{\omega}_\mathrm{r}(t)$, é possível garantir $|\dot{a}_{ij}| \leq \delta$, para um δ arbitrariamente pequeno, ao impor limitantes apropriados para as velocidades de referência e suas derivadas em relação ao tempo.

Nesse caso, a origem $e_a(t) = 0$ do sistema linear variante no tempo (2.59) será exponencialmente estável, e, pelo Teorema 5, a origem $e_a(t) = 0$ do sistema não linear (2.57) será também exponencialmente estável.

Nota-se que existe uma desvantagem neste método de análise, já que não se sabe o valor de δ nem dos limitantes M_e e M_d de $\dot{a}_e(t)$ e $\dot{a}_d(t)$. Assim, é difícil impor limitantes apropriados para $v_\mathrm{r}(t)$, $\omega_\mathrm{r}(t)$, $\dot{v}_\mathrm{r}(t)$ e $\dot{\omega}_\mathrm{r}(t)$. Porém, como vantagem, essa análise de estabilidade indica que é possível escolher as velocidades de referência de tal modo que a origem do erro de rastreamento seja exponencialmente estável.

2.3.3 Sistema com perturbação evanescente

Seja o sistema nominal perturbado pelo termo evanescente,

$$\dot{e}_a(t) = f_a(t, e_a(t)) + g(t, e_a(t)) \tag{2.73}$$

em que f_a é dada pela Equação (2.54) e g é dada pela Equação (2.55).

Deseja-se mostrar que a origem do sistema perturbado (2.73) é exponencialmente estável, sabendo de antemão que $e_a(t) = 0$ é um ponto de equilíbrio exponencialmente estável do sistema nominal (2.57). Com esse intuito, será utilizado o Lema 2 do Apêndice A.

O Lema 2 requer que exista uma função de Lyapunov $V(t, e_a(t))$ para o sistema nominal $\dot{e}_a(t) = f_a(t, e_a(t))$ que satisfaça as condições (A.11), (A.12) e (A.13). É possível afirmar que essa função existe através da aplicação do Teorema Converso 7 do Apêndice A. Como a origem do sistema nominal é exponencialmente estável, pela Definição 2, tem-se que as trajetórias do sistema nominal satisfazem a condição (A.9) do Teorema 7. Além disso, dada a imposição (2.58) feita anteriormente sobre as velocidades de referência, a Jacobiana $\partial f_a/\partial e_a$ é limitada, uniformemente em t, em $D = \{e_a(t) \in \mathbb{R}^5 \mid \|e_a(t)\| < d\}$, para qualquer d finito. A suposição feita anteriormente de que $v_{\mathrm{r}}(t)$, $\omega_{\mathrm{r}}(t)$, $a_d(t)$ e $a_e(t)$ são continuamente diferenciáveis em $t \in [0, \infty)$ garante que $f_a(t, e_a)$ seja continuamente diferenciável em $[0, \infty) \times D$, já que $f_a(t, e_a)$, que é composto por somas e produtos de polinômios e senoides em e_a, é continuamente diferenciável em e_a em $\mathbb{R}^5$. Conclui-se pelo Teorema 7 que existe uma $V(t, e_a(t))$ necessária para aplicação do Lema 2.

Para concluir a aplicação do Lema 2, deve-se mostrar que, para algum $\gamma > 0$ constante,

$$\|g(t, e_a(t))\| \leq \gamma \|e_a(t)\|, \quad \forall t \geq t_0, \quad \forall e_a(t) \in D \tag{2.74}$$

No Apêndice B.2, é mostrado que se $\sigma(t)$ for limitado,

$$|\sigma(t)| \leq L_\sigma$$

então garante-se que

$$\|g(t, e_a(t))\|_2 \leq L_\sigma L \|e_a(t)\|_2, \quad \forall e_a(t) \in D_1$$

com $D_1 = \{e_a(t) \in \mathbb{R}^5 \mid \|e_a(t)\| \leq 1\}$, e com $L := L(k_1, k_2, k_3, L_v, L_\omega)$ uma constante, dada por (B.9), que depende unicamente dos ganhos k_1, k_2, k_3, e dos limitantes L_v e L_ω. Logo, a condição

da Equação (A.14) é satisfeita com $\gamma = L_\sigma L$. Observa-se que γ é positivo e depende unicamente dos valores dos ganhos da lei de controle, k_1, k_2, k_3, e dos limitantes impostos a $v_\mathrm{r}(t), \omega_\mathrm{r}(t)$ e $\sigma(t)$, respectivamente, L_v, L_ω e L_σ.

Finalmente, nota-se que a função de Lyapunov, cuja existência é garantida pelo teorema converso 7, não é conhecida. Por essa razão, não é possível calcular o valor c_3/c_4 para a condição (A.15) do Lema 2. Entretanto, como $\gamma = L_\sigma L$, com L uma constante dependendo de parâmetros do controlador e das velocidades de referência, conclui-se que para um valor arbitrariamente pequeno de c_3/c_4, um limitante L_σ suficientemente pequeno para o parâmetro de deslizamento lateral $\sigma(t)$ garante que $\gamma < c_3/c_4$.

Assim, conclui-se do Lema 2 que a origem do sistema nominal perturbado pelo termo evanescente, Equação (2.73), também é exponencialmente estável, desde que a magnitude do parâmetro de deslizamento lateral $\sigma(t)$ seja limitada por um valor suficientemente pequeno.

2.3.4 Sistema com perturbações evanescente e não evanescente

Como foi mostrado que a origem $e_a(t) = 0$ do sistema com perturbação evanescente (2.73) é exponencialmente estável, pode-se agora aplicar o Lema 3 do Apêndice A para mostrar que as soluções $e_a(t)$ do sistema perturbado completo (2.53) são uniformemente finalmente limitadas. Para tanto, o sistema perturbado pode ser escrito como

$$\dot{e}_a(t) = f_n(t, e_a(t)) + g_n(t, e_a(t)) \tag{2.75}$$

com $f_n(t, e_a(t)) = f_a(t, e_a(t)) + g(t, e_a(t))$, com f_a e g dados pelas Equações (2.54) e (2.55), respectivamente, e g_n dada pela Equação (2.56).

Para mostrar que as soluções do sistema (2.75) são uniformemente finalmente limitadas, utiliza-se o Lema 3 do Apêndice A. Nota-se que, devido à análise anterior, sabe-se que a origem do sistema $\dot{e}_a(t) = f_n(t, e_a(t))$ é exponencialmente estável, desde que a magnitude do parâmetro de deslizamento lateral $\sigma(t)$ mantenha-se suficientemente pequena.

O Lema 3 requer que exista uma função de Lyapunov para o sistema (2.75) que satisfaça as condições (A.11), (A.12) e (A.13). Como feito anteriormente, pode-se aplicar o Teorema 7 do Apêndice A para afirmar que tal função existe, já que foi provado que a origem do sistema (2.73)

é exponencialmente estável. Observa-se que como já havia sido imposto que $\sigma(t)$, $v_{\mathrm{r}}(t)$ e $\omega_{\mathrm{r}}(t)$ sejam limitados para todo $t > t_0$, a Jacobiana $\partial f_n/\partial e_a$ é limitada, uniformemente em t, num domínio $D = \{e_a(t) \in \mathbb{R}^5 \mid \|e_a(t)\| < d\}$.

Então, dados os limitantes já assumidos anteriormente, a saber, $L_\sigma \geq |\sigma(t)|$, $L_v > |v_{\mathrm{r}}(t)|$, $M_e \geq |\dot{a}_e(t)|$ e $M_d \geq |\dot{a}_d(t)|$, a norma-infinito da perturbação não evanescente da Equação (2.56) resulta em

$$\begin{aligned}\|g_n(t, e_a(t))\|_\infty &= \max(|\sigma(t) v_{\mathrm{r}}(t) \cos e_3(t)|, |\dot{a}_e(t)|, |\dot{a}_d(t)|) \\ &\leq \delta_f := \max(L_\sigma L_v, M_e, M_d)\end{aligned}$$

Tem-se que para valores arbitrários de c_1, c_2, c_3, c_4, d e θ, é possível impor limitantes L_σ, L_v, M_e e M_d tais que (A.16) seja satisfeita para todo $t \geq 0$.

Logo, pelo Lema 3, conclui-se que, para um estado inicial $e_a(t_0)$ suficientemente próximo da origem, as soluções $e_a(t)$ do sistema perturbado (2.53) são uniformemente finalmente limitadas, com um limitante final dado por

$$\ell_f = \frac{c_4}{c_3}\sqrt{\frac{c_2}{c_1}}\,\frac{1}{\theta}\,\delta_f$$

Assim, se $L_\sigma L_v$, M_e e M_d forem pequenos o suficiente, então não apenas as soluções $e_a(t)$ serão uniformemente finalmente limitadas, mas seu limitante final será pequeno.

Nota-se ainda que sempre que a magnitude máxima do deslizamento lateral $\sigma(t)$ for grande, algo que pode ser esperado quando o solo no qual o robô opera é escorregadiço, a velocidade máxima de referência, L_v, deve ser pequena. Já em relação às taxas de variação do deslizamento longitudinal, $\dot{a}_e(t)$ e $\dot{a}_e(t)$, o máximo que se pode fazer é esperar que elas não sejam grandes demais.

Capítulo 3

Simulações Numéricas

Neste capítulo, serão apresentados resultados numéricos de simulações feitas em Matlab para verificar a robustez da lei de controle apresentada na Seção 2.2.3 em relação ao deslizamento lateral, que não havia sido previsto em seu projeto. Para fins de comparação, o desempenho do controlador adaptativo dado pelas Equações (2.39), (2.40), (2.42) e (2.47) será comparado ao desempenho do controlador não adaptativo dado pela lei de controle (2.35), sendo que será permitido à entrada de referência $\eta_{\mathrm{r}}(t) = (v_{\mathrm{r}}(t), \omega_{\mathrm{r}}(t))^T$ variar no tempo. O acrônimo CA será utilizado para designar o controlador adaptativo, enquanto o acrônimo NA se referirá ao controlador não adaptativo.

A trajetória de referência é gerada pela integração numérica do modelo cinemático da Equação (2.11), com a condição inicial $q_{\mathrm{r}}(0) = (0, 0, 0)^T$ e a entrada de referência $\eta_{\mathrm{r}} = (v_{\mathrm{r}}, \omega_{\mathrm{r}})^T$, adaptada de Kanayama et al. (1990):

$$
\begin{array}{rlll}
0\ \mathrm{s} \leq t < 5\ \mathrm{s}: & v_{\mathrm{r}}(t) = 0.25\,(1 - \cos(\pi t/5)) & \text{e} & \omega_{\mathrm{r}}(t) = 0 \\
5\ \mathrm{s} \leq t < 20\ \mathrm{s}: & v_{\mathrm{r}}(t) = 0.5 & \text{e} & \omega_{\mathrm{r}}(t) = 0 \\
20\ \mathrm{s} \leq t < 25\ \mathrm{s}: & v_{\mathrm{r}}(t) = 0.25\,(1 + \cos(\pi t/5)) & \text{e} & \omega_{\mathrm{r}}(t) = 0 \\
25\ \mathrm{s} \leq t < 30\ \mathrm{s}: & v_{\mathrm{r}}(t) = 0.15\pi\,(1 - \cos(2\pi t/5)) & \text{e} & \omega_{\mathrm{r}}(t) = -v_{\mathrm{r}}(t)/1.5 \\
30\ \mathrm{s} \leq t < 35\ \mathrm{s}: & v_{\mathrm{r}}(t) = 0.15\pi\,(1 - \cos(2\pi t/5)) & \text{e} & \omega_{\mathrm{r}}(t) = v_{\mathrm{r}}(t)/1.5 \\
35\ \mathrm{s} \leq t < 40\ \mathrm{s}: & v_{\mathrm{r}}(t) = 0.15\pi\,(1 - \cos(2\pi t/5)) & \text{e} & \omega_{\mathrm{r}}(t) = -v_{\mathrm{r}}(t)/1.5 \\
40\ \mathrm{s} \leq t < 45\ \mathrm{s}: & v_{\mathrm{r}}(t) = 0.15\pi\,(1 - \cos(2\pi t/5)) & \text{e} & \omega_{\mathrm{r}}(t) = v_{\mathrm{r}}(t)/1.5 \\
45\ \mathrm{s} \leq t < 50\ \mathrm{s}: & v_{\mathrm{r}}(t) = 0.25\,(1 + \cos(\pi t/5)) & \text{e} & \omega_{\mathrm{r}}(t) = 0 \\
50\ \mathrm{s} \leq t: & v_{\mathrm{r}}(t) = 0.5 & \text{e} & \omega_{\mathrm{r}}(t) = 0
\end{array}
$$

A Figura 3.1 mostra graficamente a entrada de referêcia $\eta_{\mathrm{r}}(t)$ dada pelas Equações acima até o tempo de 80 segundos, a Figura 3.2 mostra a parte de translação da trajetória de referência, $(x_{\mathrm{r}}(t), y_{\mathrm{r}}(t))$, gerada através da integração do modelo cinemático da Equação (2.11).

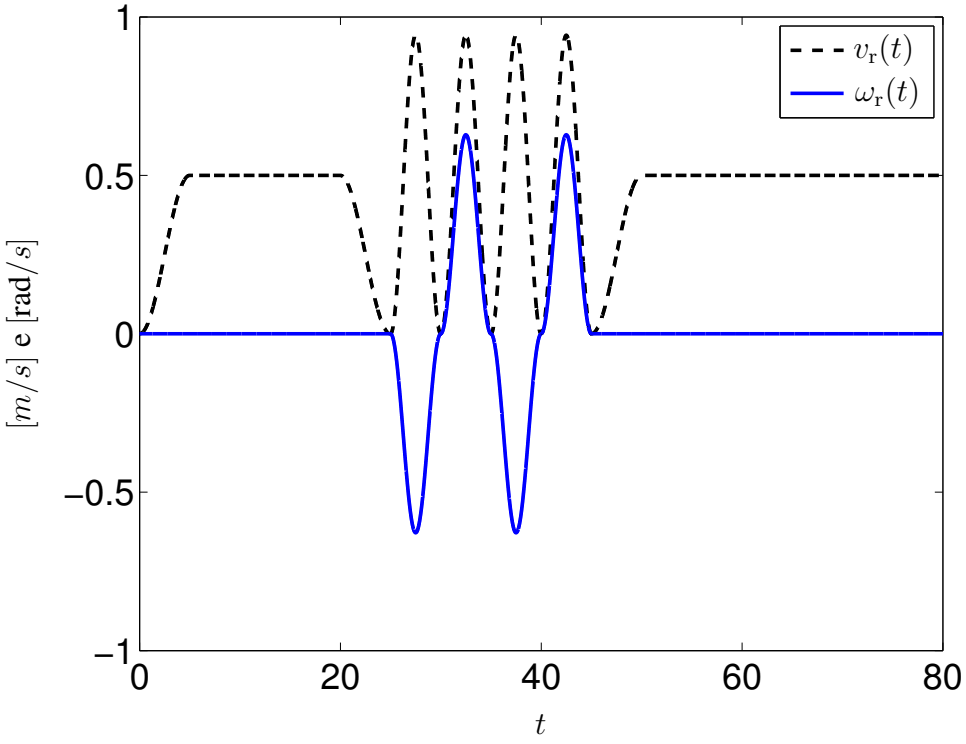

Figura 3.1: Entrada de referência $\eta_{\mathrm{r}}(t) = (v_{\mathrm{r}}(t), \omega_{\mathrm{r}}(t))^T$ utilizada nas simulações.

Observa-se que a trajetória de referência pode ser dividida em três partes. A primeira parte consiste em uma linha reta que ocorre no intervalo 0 s $\leq t \leq 25$ s , a segunda parte consiste numa "curva em S" que ocorre no intervalo 25 s $\leq t \leq 45$ s, e a terceira parte consiste em outra linha reta que ocorre a partir de $t \geq 45$ s.

Para incluir o efeito dos deslizamentos na simulação, os parâmetros de deslizamento longitudinal $a_e(t)$ e $a_d(t)$ e o parâmetro de deslizamento lateral $\sigma(t)$ são escolhidos arbitrariamente de acordo com os sinais variantes no tempo não lineares dados por

$$
\begin{aligned}
a_e(t) &= [0.7 + 0.3\,\mathrm{e}^{-0.1t}\cos(1.1t)]^{-1} \\
a_d(t) &= [0.4 + 0.6\,\mathrm{e}^{-0.08t}\sin^2(0.02t^2)]^{-1} \\
\sigma(t) &= 3.0\,\mathrm{e}^{-0.03t}\sin(\pi(t-35))/(\pi(t-35))
\end{aligned}
$$

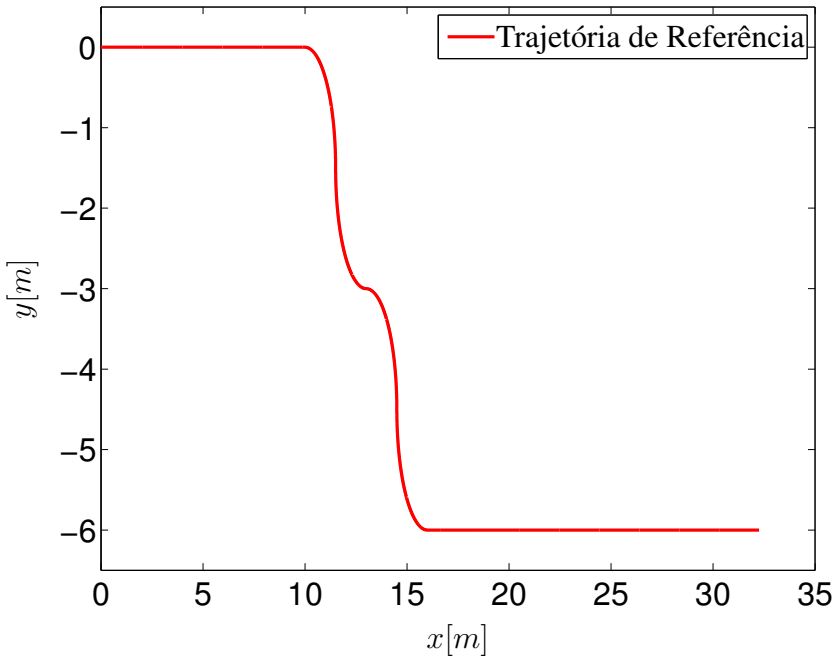

Figura 3.2: Trajetória de referência $(x_\mathrm{r}, y_\mathrm{r})$ gerada pela entrada de referência da Figura 3.1.

As condições iniciais para a lei de adaptação da Equação (2.47) serão $\hat{a}_e(0) = 1.6$ e $\hat{a}_d(0) = 1.2$, que diferem dos valores iniciais $a_e(0) = 1$ e $a_d(0) = 2.5$.

A Figura 3.3 mostra graficamente os parâmetros de deslizamento longitudinal $a_e(t)$ e $a_d(t)$, e a Figura 3.4 mostra graficamente o parâmetro de deslizamento lateral $\sigma(t)$. Esses parâmetros foram escolhidos de modo a satisfazer as suposições feitas ao longo da análise de estabilidade desenvolvida na Seção 2.3. Assim, foram escolhidos $a_e(t)$ e $a_d(t)$ variantes no tempo, continuamente diferenciáveis e limitados para todo $t \geq 0$, com derivadas $\dot{a}_e(t)$ e $\dot{a}_d(t)$ também limitadas em $t \geq 0$. O parâmetro de deslizamento lateral $\sigma(t)$ foi escolhido variante no tempo e limitado, com um um alto valor de $\sigma(t)$ ocorrendo no meio do intervalo $25 \text{ s} \leq t \leq 45$ s, quando a trajetória de referência é um caminho curvo. Fisicamente, durante o movimento do robô em linha reta, seria esperado que $\sigma(t)$ fosse nulo. Entretanto, considera-se que valores diferentes de $a_e(t)$ e $a_d(t)$ possam induzir um pequeno deslizamento lateral. Assim, $\sigma(t)$ foi escolhido não nulo mesmo em partes retas da trajetória de referência.

Os valores utilizados para os ganhos dos controladores, que foram tirados em parte de Iossaqui (2013), se encontram na Tabela 3.1. Os parâmetros físicos do sistema, extraídos de Kim e Oh

(1998), são $b = 0.1624$ m (distância entre as rodas) e $r = 0.0825$ m (raio das rodas).

Tabela 3.1: Valores escolhidos para os ganhos dos controladores adaptativo e não adaptativo.

	k_1	k_2	k_3	γ_1	γ_2
Controlador Adaptativo (CA)	1	49	1	7	7
Controlador Não Adaptativo (NA)	1	1	3	-	-

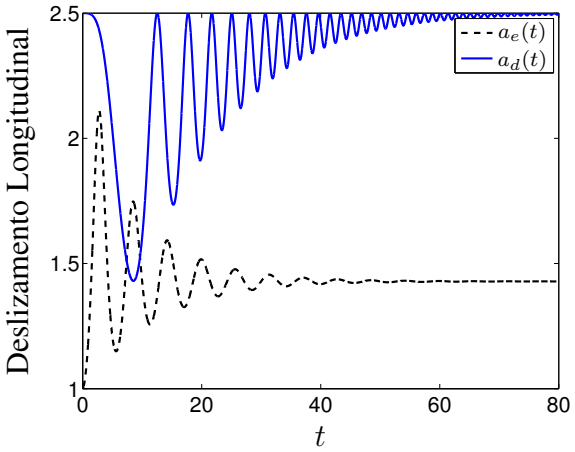

Figura 3.3: Parâmetros de deslizamento longitudinal utilizados nas simulações.

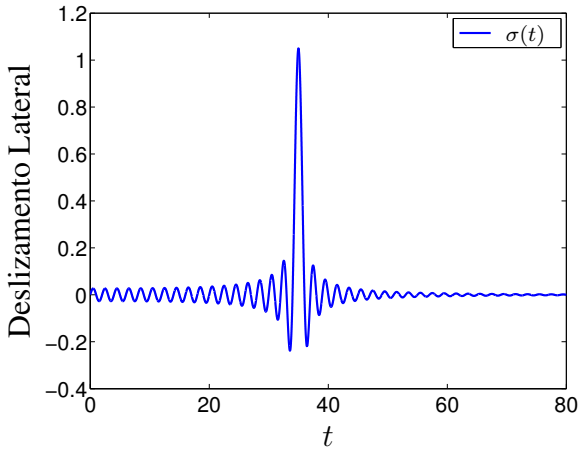

Figura 3.4: Parâmetro de deslizamento lateral utilizado nas simulações.

As Figuras 3.5 a 3.10 mostram o resultado das simulações do robô sujeito a deslizamentos longitudinais e laterais controlado com as leis de controle adaptativa (CA) e não adaptativa (NA). As simulações foram realizadas no programa Matlab utilizando a função de integração numérica *ODE45*, com um passo mínimo de 0.001 segundos, tolerância absoluta de 10^{-12} e tolerância relativa de 10^{-9}.

A Figura 3.5 mostra as trajetórias do robô móvel utilizando as leis de controle CA e NA e a Figura 3.8 mostra uma parte da trajetória ampliada graficamente. A linha pontilhada, a linha tracejada e a linha sólida representam, respectivamente, a trajetória de referência, a trajetória utilizando a lei NA, e a trajetória utilizando a lei CA. A postura inicial do robô, representada por um círculo preto, é dada por $q_{pos}(0) = (1/2, -3/4, -\pi/6)^T$. Observa-se que o robô com a lei CA é capaz de seguir a trajetória de referência, se recuperando facilmente do deslizamento lateral maior que ocorre no meio da curva em S. O deslizamento lateral persistente ao longo de toda a trajetória não desestabiliza o robô quando este é controlado pela lei CA, algo que havia sido investigado teoricamente.

Isso significa que os parâmetros de deslizamento escolhidos são suficientemente pequenos para que a estabilidade seja mantida. O deslizamento lateral não afeta consideravelmente o desempenho do controlador CA. É importante enfatizar, entretanto, que não há garantia que o desempenho será adequado para outros valores de deslizamentos. Em relação ao controle do robô com a lei NA, pode-se ver que o controlador não é capaz de compensar as perturbações dos deslizamentos e consequentemente a trajetória do robô diverge em relação à trajetória de referência.

As Figuras 3.6, 3.7 e 3.9 mostram o erro de postura do robô $e(t) = (e_1(t), e_2(t), e_3(t))^T$. As linhas tracejadas representam o erro de postura usando a lei NA, enquanto as linhas sólidas representam o erro de postura usando a lei CA. Como esperado, a lei CA apresenta um desempenho muito superior à lei NA. Dado que o deslizamento lateral $\sigma(t)$ tende a zero e os parâmetros $a_e(t)$ e $a_d(t)$ tendem a valores constantes quando $t \to \infty$, é de se esperar que o erro de postura usando a lei CA tenda a zero com $t \to \infty$, dados os resultados teóricos da Seção 2.2.3. Esse fato foi verificado numericamente para valores grandes de t (os erros ficaram abaixo da tolerância absoluta).

A Figura 3.10 mostra os erros de estimação dos parâmetros de deslizamento longitudinais. A linha tracejada corresponde aos valores de $\tilde{a}_e(t)$, e a linha sólida corresponde aos valores de $\tilde{a}_d(t)$. Verificou-se numericamente que os erros de estimação tendem a zero para valores grandes de t, quando $a_e(t)$ e $a_d(t)$ tendem a valores constantes e $\sigma(t)$ tende a zero. Esse resultado está de acordo com análise da Seção 2.3.2.

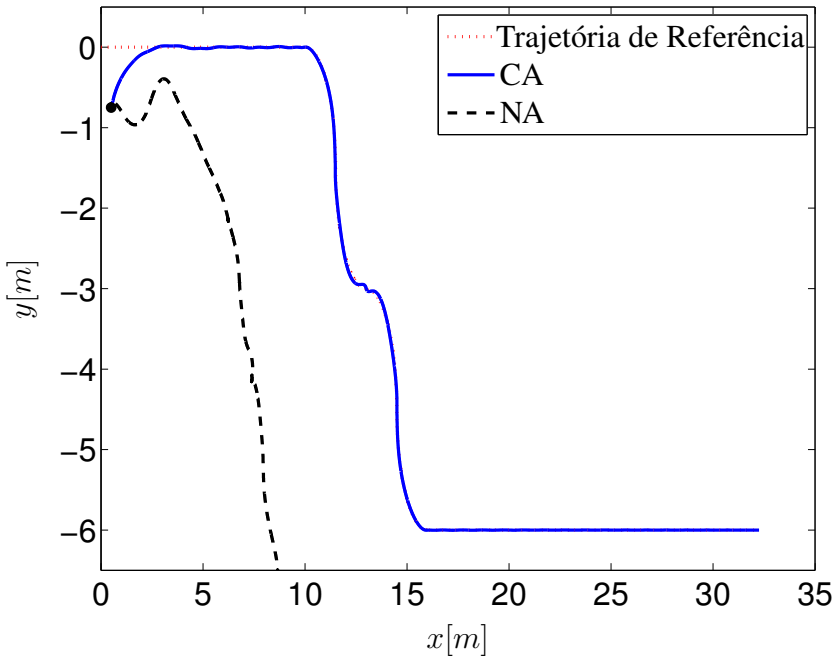

Figura 3.5: Trajetórias do robô sujeito a deslizamentos longitudinais e laterais, usando as leis de controle CA e NA.

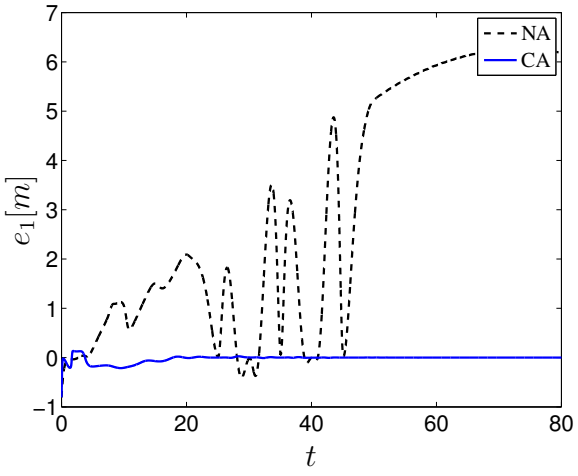

Figura 3.6: Erro de postura e_1 durante a trajetória do robô sujeito a deslizamentos laterais e longitudinais.

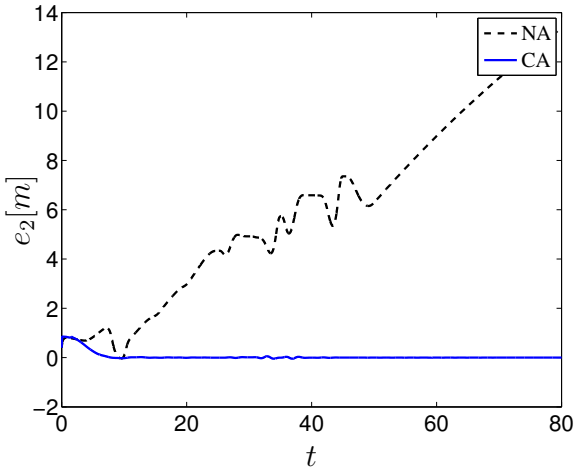

Figura 3.7: Erro de postura e_2 durante a trajetória do robô sujeito a deslizamentos laterais e longitudinais.

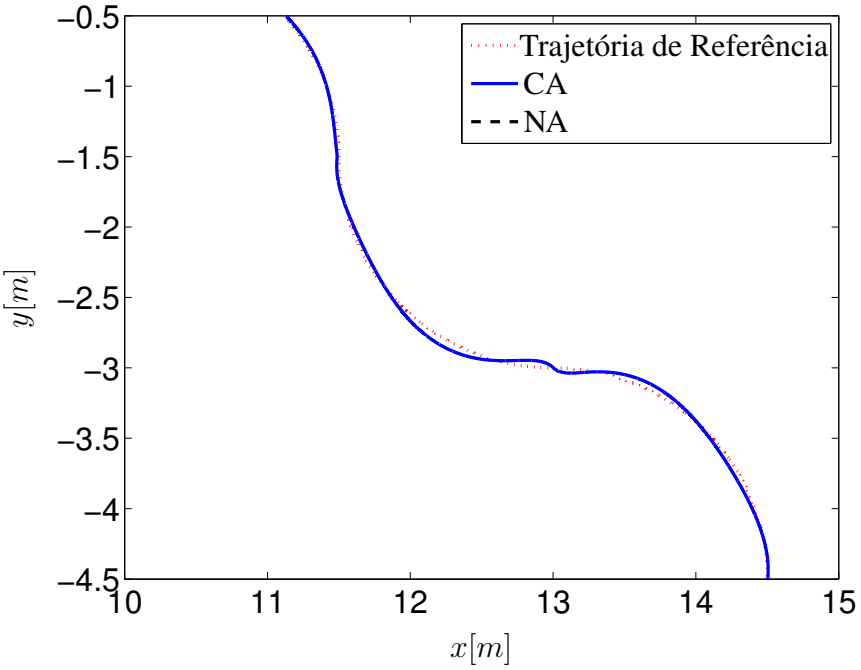

Figura 3.8: Zoom no intervalo $x = [10, 15]$ e $y = [-4.5, -0.5]$ das trajetórias do robô sujeito a deslizamentos laterais e longitudinais, usando as leis de controle CA e NA.

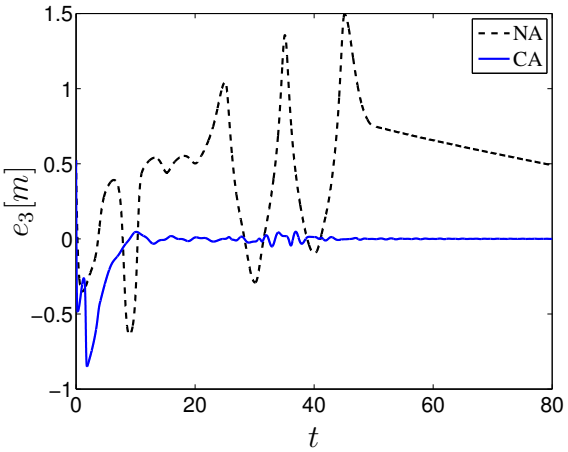

Figura 3.9: Erro de postura e_3 durante a trajetória do robô sujeito a deslizamentos laterais e longitudinais.

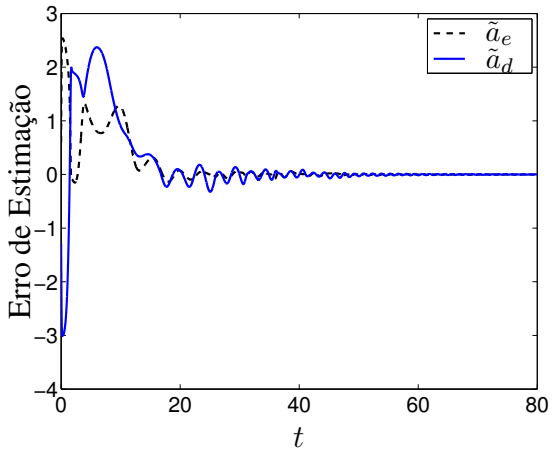

Figura 3.10: Erro de estimação $\tilde{a}_e(t)$ e $\tilde{a}_d(t)$ durante a trajetória do robô sujeito a deslizamentos laterais e longitudinais.

Capítulo 4

Otimização Polinomial Baseada em Decomposições SOS

4.1 Desenvolvimento teórico

Esta seção apresenta uma breve introdução à teoria de otimização polinomial baseada em decomposições polinomiais em somas de quadrados, ou, simplesmente, SOS (do inglês *sums of squares*). Os resultados aqui apresentados serão usados ao longo deste trabalho como apoio para a resolução de determinados problemas de análise não linear de estabilidade. A teoria apresentada nesta seção se apoia principalmente na referência Laurent (2009). A Definição 3 apresenta a terminologia utilizada neste trabalho.

Definição 3. *Um polinômio $p(x)$ em n variáveis é uma função $p(x) : \mathbb{R}^n \to \mathbb{R}$ definida por um vetor de coeficientes $c \in \mathbb{R}^m$ e uma matriz de expoentes $P \in \mathbb{N}_0^{m \times n}$, de acordo com*

$$p(x) := p(x_1, \ldots, x_n) = \sum_{i=1}^{m} c_i \prod_{j=1}^{n} x_j^{P_{ij}} \tag{4.1}$$

Os termos $c_i x_1^{P_{i1}} x_2^{P_{i2}} \ldots x_n^{P_{in}}$ são chamados de monômios. Além disso, denota-se o grau de $p(x)$ por

$$\deg(p) = \|P\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |P_{ij}|$$

em que $|P_{ij}| = P_{ij}$.

4.1.1 Não negatividade e existência de decomposições SOS

Nos algoritmos de otimização por decomposições SOS, uma ideia central é verificar se um dado polinômio pode ser colocado na forma apresentada pela Definição 4 abaixo.

Definição 4. *Um polinômio $p(x)$ em n variáveis é uma soma de quadrados de polinômios (ou simplesmente $p(x)$ é SOS) se $p(x)$ puder ser escrito nas formas equivalentes*

$$\begin{aligned} p(x) &= \sum_{i=1}^{M} u_i(x)^2 \\ &= z(x)^T Q z(x) \end{aligned} \tag{4.2}$$

em que $u_i(x)$, $i = 1, \ldots, M$, são polinômios, $z(x) : \mathbb{R}^n \to \mathbb{R}^M$ é um vetor de monômios chamado de base monomial, e $Q = Q^T \in \mathbb{R}^{M \times M}$ é uma matriz semidefinida positiva (os autovalores de Q são todos não negativos).

Para mostrar como um polinômio pode ser colocado na forma (4.2), será apresentado um pequeno exemplo.

Exemplo 1 (Parrilo (2003), Exemplo 3.5)**.** *Seja o polinômio em duas variáveis $x = (x_1, x_2)^T$ dado por*

$$f(x) = 2x_1^4 + 2x_1^3 x_2 - x_1^2 x_2^2 + 5x_2^4 \tag{4.3}$$

e seja um vetor de monômios dado por

$$z(x) = [\ x_1^2\ ,\ x_2^2\ ,\ x_1 x_2\ ]^T$$

Deseja-se testar se $f(x)$ pode ser colocado em uma das formas SOS da Equação (4.2)*. Usando a forma quadrática, tem-se*

$$\begin{aligned} f(x) &= z(x)^T Q z(x) \\ &= [\ x_1^2\ ,\ x_2^2\ ,\ x_1 x_2\ ] \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix} \\ &= q_{11}x_1^4 + q_{22}x_2^4 + (q_{33} + 2q_{12})x_1^2 x_2^2 + 2q_{13}x_1^3 x_2 + 2q_{23}x_1 x_2^3 \end{aligned} \tag{4.4}$$

em que se ressalta que $Q = Q^T$. Para que $f(x)$ seja SOS, é necessário que Q seja semidefinida positiva, e além disso, comparando (4.4) *com* (4.3)*, é necessário respeitar as restrições*

$$q_{11} = 2, \quad q_{22} = 5, \quad q_{33} + 2q_{12} = -1, \quad 2q_{13} = 2, \quad 2q_{23} = 0 \tag{4.5}$$

Uma matriz Q semidefinida positiva que respeita as restrições (4.5) *é dada por*

$$Q = \begin{bmatrix} 2 & -3 & 1 \\ -3 & 5 & 0 \\ 1 & 0 & 5 \end{bmatrix}$$

Logo, existe uma decomposição SOS para $f(x)$. Nota-se que a matriz Q não é necessariamente única, já que a equação $q_{33} + 2q_{12} = -1$ em (4.5) *possui infinitas soluções.*

A existência de uma decomposição SOS do tipo (4.2) está ligada ao conceito de não negatividade polinomial, apresentado pela Definição 5.

Definição 5. *Dado um conjunto $S \subseteq \mathbb{R}^n$, um polinômio $p(x)$ em n variáveis é dito não negativo em S (ou $p(x) \geq 0$ em S), caso $p(x) \geq 0$ para todo $x \in S$.*

É evidente, pela Equação (4.2), que um polinômio SOS em n variáveis é não negativo em $\mathbb{R}^n$. Portanto, procurar uma decomposição SOS para um polinômio é uma maneira de se testar sua não negatividade. O recíproco, porém, não é em geral verdadeiro, como explicado em Laurent (2009). Essa referência descreve um teorema devido a Hilbert, que foi um dos primeiros a mostrar que o fato de um polinômio ser não negativo em $\mathbb{R}^n$ não implica que esse polinômio possua uma decomposição SOS. Dito de outra forma, nem todo polinômio não negativo é SOS. De fato, Hilbert determinou que a equivalência entre a existência de uma decomposição SOS para um polinômio qualquer e a não negatividade desse polinômio em $\mathbb{R}^n$ só ocorre quando o número de variáveis do polinômio é 1, ou quando o grau do polinômio é 2, ou quando o número de variáveis do polinômio é 2 e seu grau é 4. O Exemplo 2 ilustra esse fato.

Exemplo 2 (Laurent (2009), Exemplo 3.7)**.** *O polinômio em duas variáveis $x = (x_1, x_2)^T$ dado por*

$$p(x) = x_1^2 x_2^2 (x_1^2 + x_2^2 - 3) + 1$$

conhecido como Polinômio de Motzkin, obedece $p(x) \geq 0$ para todo $x \in \mathbb{R}^n$. Porém, $p(x)$ não possui nenhuma decomposição SOS.

Como consequência do resultado descoberto por Hilbert, tem-se que o problema de procurar uma decomposição SOS para um polinômio qualquer $p(x)$ é menos geral que o problema de determinar a não negatividade de $p(x)$ em $\mathbb{R}^n$. Entretanto, determinar a não negatividade de um polinômio globalmente, i.e., mostrar que seu mínimo global é não negativo, é um problema de otimização não linear para o qual, em geral, não há garantia de resolução em tempo polinomial (Boyd e Vandenberghe, 2004). Por outro lado, nos últimos anos, foi mostrado que a busca por decomposições SOS pode ser formulada como um problema de programação semidefinida. Essa é uma classe de problemas de otimização cuja solução leva a ótimos globais em tempo polinomial, e para a qual existem algoritmos numéricos muito eficientes (Vandenberghe e Boyd, 1996).

Assim, a busca por decomposições SOS é uma *relaxação* do problema de determinação de não negatividade de polinômios.

Para entender como uma busca por decomposições SOS pode ser realizada através de um problema de programação semidefinida (abreviado SDP, do inglês *semidefinite programming*), pode-se ver que, no Exemplo 1 dado anteriormente, encontrar uma matriz $Q = Q^T$ semidefinida positiva sujeito às restrições (4.5) é um problema que pode ser colocado na forma padrão de problemas SDP (Boyd e Vandenberghe, 2004), dada por

$$\begin{aligned} \min \quad & tr(CQ) \\ \text{sujeito a} \quad & tr(A_i Q) = b_i, \quad i = 1, ..., r \\ & Q \geq 0 \end{aligned} \tag{4.6}$$

em que $Q \geq 0$ deve ser entendido como "Q é semidefinida positiva", e $tr(\cdot)$ denota o traço. No caso do Exemplo 1, a matriz Q poderia ter sido encontrada resolvendo um problema da forma (4.6), com C uma matriz de ponderação qualquer (a escolha $C = 0$ fornece um problema de factibilidade pura) e A_i e b_i escolhidos de acordo com as igualdades (4.5).

4.1.2 Geração da base monomial e esparsidade

No Exemplo 1, uma base monomial $z(x)$ foi escolhida arbitrariamente para que o teste de decomposição SOS pudesse ser estabelecido. Para polinômios em poucas variáveis e com poucos monômios, como o do Exemplo 1, a escolha de $z(x)$ pode ser feita por tentativa e erro, a partir da inspeção dos expoentes das variáveis em seus monômios.

No entanto, quando o número de variáveis, o número de monômios e o grau de um polinômio são significativamente altos, a escolha por tentativa e erro da base monomial $z(x)$ passa a ser muito difícil. Quando este é o caso, uma estratégia exaustiva para gerar $z(x)$ pode ser usada. Para testar se existe uma decomposição SOS para um polinômio $p(x)$ em n variáveis e de grau $\deg(p) = 2d$, pode-se incluir em $z(x)$ todos os monômios existentes em n variáveis cujo grau seja no máximo d. O problema dessa abordagem é que ela resulta em um vetor $z(x)$ com $\binom{n+d}{d}$ elementos, e portanto em uma matriz quadrada Q com $\binom{n+d}{d}$ linhas (Laurent, 2009). Para valores relativamente altos de n e d, esse tamanho de Q torna proibitiva a resolução do problema SDP descrito em (4.6).

Assim, a inclusão exaustiva de monômios em $z(x)$ é muito pouco eficiente. Além de gerar um programa SDP grande, há a possibilidade de que a decomposição SOS, se esta existir, não faça uso da maioria dos monômios da base monomial escolhida dessa forma. Seria interessante portanto que a geração de $z(x)$ para o teste de decomposição SOS fosse feita de forma mais eficiente. Nesse contexto, uma propriedade importante da estrutura polinomial que pode ser aproveitada para reduzir o tamanho de $z(x)$ e Q é a esparsidade. A esparsidade está ligada ao número de monômios que um polinômio possui, em relação ao número de monômios máximo que um polinômio de mesmo grau e mesmo número de variáveis poderia ter. O Exemplo 3 ilustra essa noção.

Exemplo 3 (Laurent (2009), Exemplo 8.2)**.** *Seja o polinômio em 4 variáveis de grau 16 dado por*

$$f(x) = (x_1^4 + 1)(x_2^4 + 1)(x_3^4 + 1)(x_4^4 + 1) + 2x_1 + 3x_2 + 4x_3 + 5x_4$$

O polinômio acima é muito esparso, pois possui apenas 20 monômios. Este é um número muito menor que o total de $4845 = \binom{4+16}{16}$ *monômios que um polinômio em 4 variáveis de grau 16 poderia ter.*

A noção de esparsidade polinomial não leva em conta exclusivamente o número de monômios nulos. O padrão com o qual os monômios de um polinômio esparso se anulam também é importante. Um dos algoritmos que melhor aproveita a esparsidade polinomial, ajudando a reduzir o tamanho da base monomial $z(x)$ associada ao teste de decomposição SOS, é baseado nos chamados Politopos de Newton (Reznick, 1978, Laurent, 2009, Löfberg, 2009b).

Para entender o conceito do Politopo de Newton, é útil referir-se à Definição 3. Considerando cada linha da matriz $P \in \mathbb{N}_0^{m \times n}$ como um vetor em $\mathbb{R}^n$, obtém-se um conjunto de m pontos em $\mathbb{R}^n$, com m o número de monômios não nulos do polinômio. Cada ponto desse conjunto representa os

expoentes de um determinado monômio (os expoentes de cada variável do monômio). O politopo de Newton é definido como a casca convexa desse conjunto de pontos em $\mathbb{R}^n$ (Sturmfels, 1998). Um resultado descoberto por Reznick (1978) garante que $z(x)$ precisa conter apenas monômios que, elevados ao quadrado, possuam expoentes que estejam contidos no Politopo de Newton. O Exemplo 4 ilustra esse conceito.

Exemplo 4 (Papachristodoulou et al. (2013), Seção 3.4.3)**.** *Seja o polinômio*

$$f(x) = 4x_1^4x_2^6 + x_1^2 - x_1x_2^2 + x_2^2 \tag{4.7}$$

Com os expoentes do monômio $4x_1^4x_2^6$, obtém-se o ponto $(4, 6)$, em que 4 é o expoente da variável x_1 e 6 é o expoente da variável x_2. Semelhantemente, com os expoentes de x_1^2, $-x_1x_2^2$ e x_2^2, são obtidos os pontos $(2, 0)$, $(1, 2)$ e $(0, 2)$, respectivamente. Nota-se que cada um desses pontos é uma linha da matriz P da Definição 3, ou seja

$$P = \begin{bmatrix} 4 & 6 \\ 2 & 0 \\ 1 & 2 \\ 0 & 2 \end{bmatrix}$$

O Politopo de Newton de $f(x)$ é obtido calculando a casca convexa do conjunto de pontos $(4, 6)$, $(2, 0)$, $(1, 2)$, $(0, 2)$ em $\mathbb{R}^2$. A Figura 4.1 ilustra essa casca convexa, bem como os pontos representando os expoentes das variáveis x_1 e x_2 de $f(x)$.

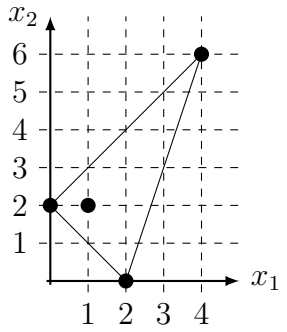

Figura 4.1: Politopo de Newton de (4.7).

Para escolher uma base monomial para a decomposição SOS de $f(x)$, é suficiente incluir na base todos os monômios que, elevados ao quadrado, possuam expoentes que estejam contidos no

Politopo de Newton. Ou seja, monômios cujos expoentes dobrados, e representados como pontos em $\mathbb{R}^n$, estejam no interior do Politopo de Newton. No caso, os pontos inteiros do plano x_1x_2 da Figura 4.1 que, quando multiplicados por dois, se encontram dentro da casca convexa da Figura 4.1, são $(1,0)$, $(0,1)$, $(1,1)$, $(1,2)$ e $(2,3)$. Portanto, uma base monomial para a decomposição SOS de $f(x)$ é dada por

$$z(x) = [\ x_1\ ,\ x_2\ ,\ x_1x_2\ ,\ x_1x_2^2\ ,\ x_1^2x_2^3\ ]^T \tag{4.8}$$

Assim, um algoritmo que escolha $z(x)$ com base no cálculo do Politopo de Newton é capaz de gerar eficientemente uma base monomial $z(x)$ para o teste de decomposição SOS. É importante frisar que para que um algoritmo por Politopo de Newton seja eficiente na geração de um $z(x)$ reduzido, não basta que o polinômio que se deseja decompor em SOS tenha poucos monômios. A estrutura dos monômios, i.e., a localização dos pontos gerados por seus expoentes em $\mathbb{R}^n$, é extremamente importante.

4.1.3 Otimização polinomial com restrições

A procura por decomposições SOS é útil como um meio de testar a não negatividade global de um polinômio, como visto na Seção 4.1.1. Sua utilidade, porém, vai além disso. Nesta Seção, será visto como as decomposições SOS ajudam a resolver problemas de otimização polinomial do tipo

$$\begin{aligned} \inf_{x\in\mathbb{R}^n} \quad & p(x) \\ \text{sujeito a} \quad & g_1(x) \geq 0 \\ & \vdots \\ & g_m(x) \geq 0 \end{aligned} \tag{4.9}$$

em que $p(x)$ e $g_1(x), \ldots, g_m(x)$ são polinômios em n variáveis.

Para entender como as decomposições SOS entram no contexto do problema (4.9), convém escrevê-lo na forma equivalente dada por

$$\begin{aligned} \sup_{\rho\in\mathbb{R}} \quad & \rho \\ \text{sujeito a} \quad & p(x) - \rho \geq 0 \quad \text{em } K \end{aligned} \tag{4.10}$$

em que

$$K = \{x \in \mathbb{R}^n \mid g_1(x) \geq 0, \ldots, g_m(x) \geq 0\} \tag{4.11}$$

De acordo com Laurent (2009), se p^* é o valor ótimo de (4.9) e ρ^* é o valor ótimo de (4.10), então $p^* = \rho^*$.

Para um problema irrestrito, i.e., $K = \mathbb{R}^n$, a condição de não negatividade do problema de otimização (4.10) pode ser relaxada, de acordo com a Seção 4.1.1, por uma condição de existência de decomposição SOS. O problema irrestrito relaxado é dado por

$$\begin{aligned} \sup_{\rho \in \mathbb{R}} \quad & \rho \\ \text{sujeito a} \quad & p(x) - \rho \text{ é SOS} \\ & x \in \mathbb{R}^n \end{aligned} \tag{4.12}$$

Devido à não equivalência em geral entre existência de decomposições SOS e não negatividade em $\mathbb{R}^n$, o problema irrestrito (4.12) pode ser infactível mesmo se houver algum ρ finito tal que $p(x) - \rho \geq 0, \forall x \in \mathbb{R}^n$. Quando o Problema (4.10) é restrito, no entanto, e quando as restrições que definem o conjunto K satisfazem uma determinada condição, há garantia teórica de que é possível, resolvendo uma sequência de problemas com relaxações SOS, obter soluções que convergem ao ótimo do Problema (4.10). Essa garantia teórica é baseada numa classe de teoremas chamados *Positivstellensatz*. Os trabalhos Parrilo (2003) e Lasserre (2001) foram os primeiros que exploraram o potencial de tais teoremas para a resolução de problemas de otimização polinomial com restrições. Neste trabalho, será utilizado um resultado que envolve o *Positivstellensatz* de Putinar, apresentado no Teorema 1.

Teorema 1 (Laurent (2009), Teorema 3.20)**.** *Seja $f(x)$ um polinômio em n variáveis, e K um conjunto construído com os polinômios das restrições, $g_1(x), \ldots, g_m(x)$, de acordo com*

$$K = \{x \in \mathbb{R}^n \mid g_1(x) \geq 0, \ldots, g_m(x) \geq 0\}$$

Suponha que os polinômios das restrições satisfaçam a seguinte condição[1]*:*

$$\begin{aligned} &\exists N \in \mathbb{N},\ \exists s_j(x) \textit{ SOS},\ j = 0, 1, \ldots, m, \quad \textit{tais que} \\ &N - \|x\|_2^2 = s_0 + \sum_{j=1}^{m} s_j(x) g_j(x) \end{aligned} \tag{4.13}$$

[1] A condição (4.13) pode ser escrita de várias outras formas (cf. Laurent (2009), Lema 3.17).

Então, se $f(x) > 0$ em K, existem polinômios SOS $\sigma_0(x), \sigma_1(x), \ldots, \sigma_m(x)$ tais que

$$f(x) = \sigma_0 + \sum_{j=1}^{m} \sigma_j(x) g_j(x) \tag{4.14}$$

Observação 1 (Nie et al. (2008), Observação 4.3)**.** *Para garantir que a Condição* (4.13) *seja satisfeita, basta adicionar às restrições que definem o conjunto K a restrição*

$$R - \|x\|_2^2 \geq 0$$

Na prática, $R > 0$ deve ser grande o suficiente para que a bola de raio R contenha a região onde se espera encontrar o ótimo global (ou os ótimos globais) do problema de otimização[2]*.*

Para entender como o Teorema 1 ajuda a resolver o Problema (4.10), nota-se primeiramente que, como o enunciado de (4.10) é feito com o uso supremo, a restrição de desigualdade não estrita ($\geq$) de (4.10) pode ser equivalentemente substituída por uma restrição estrita ($>$). Assim, tem-se

$$\begin{aligned} &\sup_{\rho \in \mathbb{R}} \quad \rho \\ &\text{sujeito a} \quad p(x) - \rho > 0 \quad \text{em } K \end{aligned} \tag{4.15}$$

com K o conjunto dado por (4.11). Posto isso, o Teorema 1 implica diretamente o resultado a seguir.

Lema 1. *Seja ρ^* o ótimo de ρ no Problema* (4.15)*, e seja ρ^{sos} o ótimo de ρ no problema*

$$\begin{aligned} &\sup_{\rho \in \mathbb{R}} \quad \rho \\ &\textit{sujeito a} \quad p(x) - \rho - \sum_{j=1}^{m} \sigma_j(x) g_j(x) \quad \textit{é SOS} \\ &\qquad\qquad \sigma_j(x) \ \textit{é SOS}, \quad j = 1, \ldots, m \end{aligned} \tag{4.16}$$

Então, se $g_1, \ldots, g_m$ satisfazem (4.13)*, tem-se que $\rho^* = \rho^{sos}$.*

Demonstração. Se ρ^* é o ótimo do Problema (4.15), então para qualquer $\rho < \rho^*$ tem-se que $p(x) - \rho > 0$ em K. Assim, se $g_1, \ldots, g_m$ satisfazem (4.13), o Teorema 1 garante que existem polinômios SOS $\sigma_0(x), \sigma_1(x), \ldots, \sigma_m(x)$ tais que

$$p(x) - \rho = \sigma_0(x) + \sum_{j=1}^{m} \sigma_j(x) g_j(x), \quad \forall \rho < \rho^*$$

[2]Uma estratégia para determinar R é dada em Nie et al. (2008), Observação 4.3.

de onde se conclue que ρ^* também é o ótimo do Problema (4.16) e logo $\rho^* = \rho^{\text{sos}}$. □

No Problema (4.16), os σ_j são polinômios que podem ser encontrados considerando seus coeficientes como variáveis do problema de programação semidefinida que é gerado pelo problema de procura de decomposição SOS para as restrições. O número dessas variáveis naturalmente aumenta de acordo com o grau dos σ_j. Porém, não fica claro qual grau deve-se escolher para os σ_j. Para que se possa implementar o problema (4.16) computacionalmente, esse grau deve ser finito. Podendo haver cancelamento de termos no somatório de (4.16), essa questão se torna ainda mais complicada. O Teorema a seguir propõe uma maneira de limitar os graus dos polinômios SOS σ_j, e portanto garante um resultado importante para implementações numéricas.

Teorema 2 (Laurent (2009), Teorema 6.8)**.** *Seja o problema de otimização* (4.9)*, cujo ótimo é denotado por* p^**. Seja* $T \in \mathbb{N}$ *tal que*

$$2T \geq \max[\deg(p), \deg(g_1), \ldots, \deg(g_m)]$$

Para um dado T*, é definido o problema de otimização*

$$\begin{aligned} \sup_{\rho \in \mathbb{R}} \quad & \rho \\ \textit{sujeito a} \quad & p(x) - \rho - \sum_{j=1}^{m} \sigma_j(x) g_j(x) \quad \textit{é SOS} \\ & \sigma_j(x) \ \textit{é SOS}, \quad j = 1, \ldots, m \\ & \deg(\sigma_j g_j) \leq 2T, \quad j = 1, \ldots, m \end{aligned} \tag{4.17}$$

Denota-se o ótimo do problema acima por ρ_T^{sos}*. Se os* g_j *satisfazem a Condição* (4.13)*, então tem-se*

$$\lim_{T \to \infty} \rho_T^{sos} = p^*$$

Portanto, para se resolver o Problema (4.9), pode-se começar escolhendo o menor T possível tal que $2T \geq \max[\deg(p), \deg(g_1), \ldots, \deg(g_m)]$, e então resolver uma sequência de problemas do tipo (4.17) impondo $\deg(\sigma_j g_j) \leq 2T$. Aumentando T a cada nova resolução, o valor de ρ_T^{sos} eventualmente convergirá para a solução de (4.9), p^*.

4.2 Aplicação de relaxações SOS em um problema de otimização

O objetivo desta seção é formalizar e apresentar os resultados de um problema de otimização polinomial capaz de mostrar pelo menos a não negatividade do termo Δ_4 da Condição (2.69), algo necessário para as provas de estabilidade da Seção 2.3. Com essa finalidade, será utilizada a teoria de relaxações SOS para problemas de otimização polinomial, introduzida na Seção 4.1.

4.2.1 Estruturação do problema de otimização

Os coeficientes α_i dados pelas equações (2.62)-(2.66) são funções racionais que dependem de dez parâmetros do sistema em malha fechada: a_e, a_d, b, k_1, k_2, k_3, v_r, ω_r, γ_1 e γ_2. Esses parâmetros serão agrupados no seguinte vetor de variáveis:

$$y = (a_e, a_d, b, k_1, k_2, k_3, v_\mathrm{r}, \omega_\mathrm{r}, \gamma_1, \gamma_2)^T$$

Tem-se da Condição (2.69) que $\Delta_4(\alpha)$ é dado por

$$\Delta_4(\alpha) = (\alpha_1\alpha_2 - \alpha_3)(\alpha_3\alpha_4 - \alpha_2\alpha_5) - (\alpha_1\alpha_4 - \alpha_5)^2 \tag{4.18}$$

Substituindo os $\alpha_i(y)$ das equações (2.62)-(2.66) na Equação acima, denota-se

$$\Delta_4(y) := \Delta_4(\alpha(y))$$

que também é uma função racional. Nesta seção, serão estabelecidos problemas de otimização que levam à determinação da não negatividade de $\Delta_4(y)$.

Alguns dos parâmetros de y são variantes no tempo, como as velocidades de referência, v_r e ω_r, e os parâmetros de deslizamento longitudinal, a_e e a_d, enquanto os ganhos do controlador, k_i e γ_i, e a largura entre as rodas do robô, b, são parâmetros constantes, invariantes no tempo. No que concerne os problemas de otimização que serão formalizados a seguir, a questão da dependência do tempo de um dado parâmetro não será central.

Ao longo do Capítulo 2, estabeleceu-se que as variáveis b, k_i e γ_i são todas positivas. Além disso, foi imposta sobre v_r a condição $v_\mathrm{r}(t) \geq \mu > 0$. Por fim, tem-se pelo modelo de deslizamento

que os parâmetros a_e, a_d estão contidos no intervalo $[1, \infty)$. Essas condições podem ser utilizadas como restrições do problema de otimização dado por

$$\begin{aligned} \min_{y \in \mathbb{R}^{10}} \quad & \Delta_4(y) \\ \text{sujeito a} \quad & a_e \geq 1 \\ & a_d \geq 1 \\ & b > 0 \\ & k_i > 0, \quad i = 1, 2, 3 \\ & \gamma_i > 0, \quad i = 1, 2 \\ & v_{\mathrm{r}} \geq \mu > 0, \quad \mu \in \mathbb{R} \end{aligned} \tag{4.19}$$

A resolução do Problema (4.19) traria uma resposta sobre a não negatividade de $\Delta_4(y)$ no subconjunto de $\mathbb{R}^{10}$ dado pelas restrições acima. Nota-se que (4.19) não é um problema de otimização polinomial, já que $\Delta_4(y)$ é uma função racional. Porém, pode-se obter um problema polinomial equivalente a partir do problema (4.19).

Para cada um dos $\alpha_i(y)$ das Equações (2.62)-(2.66), denotam-se seus numeradores por $\bar{\alpha}_i(y)$ e seus denominadores por $\underline{\alpha}_i(y)$, i.e.,

$$\alpha_i(y) = \frac{\bar{\alpha}_i(y)}{\underline{\alpha}_i(y)}, \quad i = 1, 2, 3, 4, 5 \tag{4.20}$$

Assim, de acordo com as equações (2.62)-(2.66), os numeradores $\bar{\alpha}_i$ são dados por

$$\begin{aligned} \bar{\alpha}_1(y) &= 2k_1k_3 + k_4v_{\mathrm{r}} \\ \bar{\alpha}_2(y) &= k_3(v_1^2a_e\gamma_2 + v_2^2a_d\gamma_1)(b^2k_2 + 4k_4) + 8a_ea_db^2k_2(v_{\mathrm{r}}(k_1k_4 + k_2k_3v_{\mathrm{r}}) + 2k_3\omega_{\mathrm{r}}^2) \\ \bar{\alpha}_3(y) &= (v_1^2a_e\gamma_2 + v_2^2a_d\gamma_1)(k_2v_{\mathrm{r}}(b^2k_4 + 8k_3^2) + 8k_1k_3k_4) + 16a_ea_db^2k_2v_{\mathrm{r}}(k_1k_2k_3v_{\mathrm{r}} + \omega_{\mathrm{r}}^2) \\ \bar{\alpha}_4(y) &= v_1^2\gamma_2a_e((bk_2v_{\mathrm{r}} + 2\omega_{\mathrm{r}})^2 + 4\omega_{\mathrm{r}}^2 + 8k_1k_2k_3v_{\mathrm{r}}) \\ &\qquad + v_2^2\gamma_1a_d((bk_2v_{\mathrm{r}} - 2\omega_{\mathrm{r}})^2 + 4\omega_{\mathrm{r}}^2 + 8k_1k_2k_3v_{\mathrm{r}}) + 2\gamma_1\gamma_2k_4v_1^2v_2^2 \\ \bar{\alpha}_5(y) &= \gamma_1\gamma_2k_3v_{\mathrm{r}}v_1^2v_2^2 \end{aligned} \tag{4.21}$$

com $k_4 = 1 + k_2 k_3^2$, $v_1 = b\omega_\mathrm{r} + 2v_\mathrm{r}$ e $v_2 = b\omega_\mathrm{r} - 2v_\mathrm{r}$. Os denominadores $\underline{\alpha}_i$ são dados por

$$\begin{aligned}
\underline{\alpha}_1(y) &= 2k_3 \\
\underline{\alpha}_2(y) &= 16a_e a_d b^2 k_2 k_3 \\
\underline{\alpha}_3(y) &= 32a_e a_d b_2^2 k_2 k_3 \\
\underline{\alpha}_4(y) &= 32a_e a_d b^2 k_2 \\
\underline{\alpha}_5(y) &= 16a_e a_d b^2
\end{aligned} \tag{4.22}$$

Substituindo $\alpha_i(y) = \bar{\alpha}_i(y)/\underline{\alpha}_i(y)$ na Equação (4.18), Δ_4 é dado por

$$\Delta_4(y) = \frac{(\bar{\alpha}_1\bar{\alpha}_2 - \beta_1\bar{\alpha}_3)(\bar{\alpha}_3\bar{\alpha}_4 - \beta_2\bar{\alpha}_2\bar{\alpha}_5) - \beta_3(\bar{\alpha}_1\bar{\alpha}_4 - \beta_4\bar{\alpha}_5)^2}{2^{15}a_e^3 a_d^3 b^6 k_2^3 k_3^3} \tag{4.23}$$

em que os $\bar{\alpha}_i = \bar{\alpha}_i(y)$ são dados por (4.21) e

$$\begin{aligned}
\beta_1(y) &= k_3 \\
\beta_2(y) &= 4k_2 \\
\beta_3(y) &= 8a_e a_d b^2 k_2 k_3 \\
\beta_4(y) &= 4k_2 k_3
\end{aligned} \tag{4.24}$$

Nota-se que, de acordo com as restrições do problema (4.19), o denominador de Δ_4 na Equação (4.23), dado por $2^{15}a_e^3 a_d^3 b^6 k_2^3 k_3^3$, sempre assume valores positivos. Além disso, dado o fato que b, k_2 e k_3 são constantes positivas, o termo $2^{15}a_e^3 a_d^3 b^6 k_2^3 k_3^3$ não tende a zero. Assim, testar a positividade de Δ_4 na Equação (4.23) resume-se a testar a positividade de seu numerador, que será denotado por

$$\bar{\Delta}_4(y) = (\bar{\alpha}_1\bar{\alpha}_2 - \beta_1\bar{\alpha}_3)(\bar{\alpha}_3\bar{\alpha}_4 - \beta_2\bar{\alpha}_2\bar{\alpha}_5) - \beta_3(\bar{\alpha}_1\bar{\alpha}_4 - \beta_4\bar{\alpha}_5)^2 \tag{4.25}$$

em que os $\alpha_i = \alpha_i(y)$ são dados por (4.21) e os $\beta_i = \beta_i(y)$ são dados por (4.24).

Do exposto acima, o teste de não negatividade do termo Δ_4 se resume ao seguinte problema

de otimização

$$\begin{aligned} \min_{y\in\mathbb{R}^{10}} \quad & \bar{\Delta}_4(y) \\ \text{sujeito a} \quad & a_e \geq 1 \\ & a_d \geq 1 \\ & b > 0 \\ & k_i > 0, \quad i = 1,2,3 \\ & \gamma_i > 0, \quad i = 1,2 \\ & v_{\mathrm{r}} \geq \mu > 0, \quad \mu \in \mathbb{R} \end{aligned} \tag{4.26}$$

A resolução do Problema (4.26) traria uma resposta sobre a não negatividade de $\bar{\Delta}_4(y)$ no subconjunto de $\mathbb{R}^{10}$ dado pelas restrições acima. Por conseguinte, traria uma resposta sobre a não negatividade de Δ_4 no mesmo subconjunto (pois o denominador de Δ_4 em (4.23) é positivo).

É uma tarefa extremamente difícil obter uma solução fechada para o problema de otimização polinomial restrito dado em (4.26). Uma opção é recorrer a técnicas numéricas. No entanto, nem sempre é possível resolver problemas de otimização não linear (em particular, otimização polinomial) de modo a obter ótimos globais em tempo polinomial. De acordo com Boyd e Vandenberghe (2004), não existe um método infalível para se resolver, de modo geral, problemas de otimização não linear. Os métodos existentes abordam o problema de várias maneiras diferentes, cada qual com suas vantagens e desvantagens.

A abordagem escolhida para tratar o problema de otimização polinomial (4.26) são as relaxações SOS, introduzidas na Seção 4.1. A vantagem deste método é a obtenção de resultados globais a partir da resolução de problemas de programação semidefinida, para os quais existem algoritmos eficientes disponíveis. A desvantagem é a dependência de cálculos numéricos, algo que na prática dificulta, entre outros fatores, a obtenção de ótimos localizados na fronteira de regiões estabelecidas com restrições de desigualdade estritas (como é o caso do Problema (4.26)). Por essa razão, a distinção entre positividade e não negatividade é muitas vezes difícil de ser determinada a partir de métodos numéricos de otimização.

No restante desta seção, será utilizada a teoria da Seção 4.1 para elaborar problemas de otimização utilizando decomposições SOS capazes de abordar o problema (4.26).

4.2.2 Minimização com restrições usando o Positivstellensatz

Tendo em vista que o Problema (4.26) é um problema de otimização polinomial com restrições, é natural investigar se a teoria de relaxações SOS baseadas no Positivstellensatz, introduzidas na Seção 4.1.3, pode ser aplicada para ajudar a resolver o problema (4.26).

Comparando o problema de otimização polinomial genérico (4.9) com o Problema (4.26), nota-se uma diferença relevante: algumas das restrições de desigualdade do Problema (4.26) são restrições estritas, e.g., $b > 0$, enquanto as restrições do problema 4.9 são restrições de desigualdade não estritas. Para poder aplicar as relaxações SOS usando o Positivstellensatz, que usam como base o Problema (4.9), é portanto necessário colocar o Problema (4.26) de alguma maneira na forma do Problema (4.9). Duas abordagens para se fazer isso são:

1. Substituir as restrições estritas $b > 0$, $k_i > 0$ e $\gamma_i > 0$ por $b - \mu_b \geq 0$, $k_i - \mu_{k_i} \geq 0$ e $\gamma_i - \mu_{\gamma_i} \geq 0$, escolhendo valores reais arbitrários para μ_b, μ_{k_i} e μ_{γ_i}. Também deve-se escolher um valor arbitrário para μ na restrição $v_{\mathrm{r}} - \mu \geq 0$, que já está na forma apropriada. A vantagem desta abordagem é que a determinação de positividade estrita da função objetivo do Problema (4.26) seria em teoria possível. A desvantagem é que essa positividade dependeria fortemente das escolhas arbitrárias para os limitantes μ, μ_b, μ_{k_i} e μ_{γ_i}, algo indesejado.

2. Substituir as restrições estritas de (4.26) por restrições não estritas. Nesse caso, abdica-se da intenção de determinar a positividade estrita da função objetivo (4.26), já que zero se torna um valor factível. A vantagem seria englobar todas as escolhas possíveis para os limitantes propostos pela abordagem alternativa acima.

O dilema da escolha entre a abordagem (1.) ou (2.) acima evidencia a dificuldade que se encontra quando tenta-se resolver problemas originalmente analíticos através de métodos numéricos.

Será considerada neste trabalho a abordagem (2.), ou seja, substitui-se as restrições estritas de (4.26) por restrições não estritas.

Essa abordagem leva ao problema

$$\begin{aligned} \min_{y\in\mathbb{R}^{10}} \quad & \bar{\Delta}_4(y) \\ \text{sujeito a} \quad & g_1(y) = a_e - 1 \geq 0 \\ & g_2(y) = a_d - 1 \geq 0 \\ & g_3(y) = b \geq 0 \\ & g_{i+3}(y) = k_i \geq 0, \quad i = 1,2,3 \\ & g_{i+6}(y) = \gamma_i \geq 0, \quad i = 1,2 \\ & g_9(y) = v_{\mathrm{r}} \geq 0 \end{aligned} \tag{4.27}$$

É importante notar que a resolução do Problema (4.27) gera um limitante inferior para o ótimo do Problema (4.26). O Problema (4.27) pode ser resolvido computacionalmente através da aplicação do Teorema 2 da Seção 4.1. Para aplicar o Teorema 2, entretanto, é necessário que o conjunto de restrições g_j do problema (4.27) satisfaça a Condição (4.13). Para garantir que o conjunto de restrições satisfaça a Condição (4.13), lembrando da Observação 1, impõe-se uma restrição adicional para o problema (4.27), dada por

$$g_{10}(y) = R - \|y\|_2^2 \geq 0 \tag{4.28}$$

em que o número $R > 0$ deve ser escolhido suficientemente grande para que a região definida por $g_{10}(y) \geq 0$ contenha o ponto de ótimo do Problema (4.27).

O Teorema 2 garante que se pode convergir ao ótimo do Problema (4.27) através da resolução de uma sequência de relaxações do do tipo

$$\begin{aligned} \sup_{\rho\in\mathbb{R}} \quad & \rho \\ \text{sujeito a} \quad & \bar{\Delta}_4(y) - \rho - \sum_{j=1}^{10} \sigma_j(y) g_j(y) \quad \text{é SOS} \\ & \sigma_j(y) \ \text{é SOS}, \quad j = 1,\ldots,10 \\ & \deg(\sigma_j g_j) \leq 2T, \quad j = 1,\ldots,10 \end{aligned} \tag{4.29}$$

com

$$2T \geq \max[\deg(p), \deg(g_1), \ldots, \deg(g_m)]$$

em que os g_j são dados nas equações (4.27) e (4.28), e o polinômio $\bar{\Delta}_4(y)$ é dado por (4.25).

Tentou-se implementar o Problema (4.29) computacionalmente numa máquina com uma quantidade considerável de memória (32 Gb), usando inicialmente $R = 100$, e verificou-se que não havia memória suficiente para resolver o problema. Isso motivou a elaboração de uma série de testes com decomposições SOS cujo intuito era analisar o quanto de memória seria necessário para resolver um problema como o problema (4.29). Esses testes serão descritos em detalhes na Seção 4.3.2, já que constituem resultados interessantes por si sós. Verificou-se a partir dos testes mencionados que uma quantidade exorbitante de memória seria necessária para resolver o problema (4.29), tornando sua resolução impossível em termos práticos.

Vale a pena elucidar a razão pela qual a quantidade de memória necessária para resolver o Problema (4.29) é grande. Ela primeiramente tem a ver com o tamanho do polinômio $\bar{\Delta}_4(y)$, que é um polinômio em dez variáveis de grau quarenta. Mas essa razão também está relacionada com a perda de esparsidade do problema devido à existência dos polinômios desconhecidos σ_j, cujos coeficientes devem ser encontrados numericamente pelo programa de computador utilizado para resolver o problema (4.29). Dado o grau de $\bar{\Delta}_4(y)$, o valor de $2T$ que deve ser usado no Problema (4.29) é no mínimo $2T = 40$. Além disso, o maior grau apresentado por uma restrição é $\deg(g_{10}) = 2$. Assim, de acordo com o Teorema 2, deve-se buscar por polinômios SOS $\sigma_j(y)$ que tenham grau até no máximo 38. Essa busca, na prática, resume-se a encontrar matrizes Q_{σ_j} associadas às decomposições SOS $z_{\sigma_j}(y)^T Q_{\sigma_j} z_{\sigma_j}(y)$ de cada um dos $\sigma_j(y)$. Mas visto que os σ_j são totalmente desconhecidos a priori, para garantir que a busca pelos σ_j inclua todos os polinômios SOS de grau até 38, deve-se incluir na base $z_\sigma(y)$ todos os monômios existentes em dez variáveis e de grau até 19, como explicado na Seção (4.1.2). Essa inclusão exaustiva de monômios nas bases $z_{\sigma_j}(y)$ tira completamente a esparsidade do problema, que, por envolver muitas variáveis e graus muito altos, requer uma quantidade exagerada de memória para ser resolvido.

4.2.3 Existência de decomposição SOS: problema equivalente Irrestrito

O problema de otimização polinomial com restrições (4.26) pode ser relaxado por um problema de otimização polinomial irrestrito através da imposição de uma mudança de variáveis:

$$\begin{aligned}
a_e &= 1 + x_1^2, & a_e &\geq 1 \quad \forall x_1 \in \mathbb{R} \\
a_d &= 1 + x_2^2, & a_d &\geq 1 \quad \forall x_2 \in \mathbb{R} \\
b &= x_3^2, & b &\geq 0 \quad \forall x_3 \in \mathbb{R} \\
k_i &= x_{i+3}^2, & k_i &\geq 0 \quad \forall x_{i+3} \in \mathbb{R}, \quad i = 1,2,3 \\
v_{\mathrm{r}} &= x_7^2, & v_{\mathrm{r}} &\geq 0 \quad \forall x_7 \in \mathbb{R} \\
\omega_{\mathrm{r}} &= x_8, & \omega_{\mathrm{r}} &\in \mathbb{R} \quad \forall x_8 \in \mathbb{R} \\
\gamma_i &= x_{i+8}^2, & \gamma_i &\geq 0 \quad \forall x_{i+8} \in \mathbb{R}, \quad i = 1,2
\end{aligned} \tag{4.30}$$

em que x_i, $i = 1, \ldots, 10$ são as novas variáveis do problema.

Definindo o vetor de variáveis livres $x = (x_1, \ldots, x_{10})^T$, denota-se a substituição por

$$y(x) = (a_e(x_1), a_d(x_2), b(x_3), k_1(x_4), k_2(x_5), k_3(x_6), v_{\mathrm{r}}(x_7), \omega_{\mathrm{r}}(x_8), \gamma_1(x_9), \gamma_2(x_{10}))$$

em que as relações entre as variáveis antigas e as novas são dadas por (4.30).

Através da substituição de variáveis (4.30) em $\bar{\Delta}_4(y)$, dado pela Equação (4.25), é formulado o problema de otimização polinomial irrestrito

$$\min_{x \in \mathbb{R}^{10}} \quad \bar{\Delta}_4(y(x)) := \bar{\Delta}_4(x) \tag{4.31}$$

que, devido à natureza da substituição de variáveis (4.30), é equivalente ao problema de otimização restrito

$$\begin{aligned}
\min_{y \in \mathbb{R}^{10}} \quad & \bar{\Delta}_4(y) \\
\text{sujeito a} \quad & a_e \geq 1 \\
& a_d \geq 1 \\
& b \geq 0 \\
& k_i \geq 0, \quad i = 1,2,3 \\
& \gamma_i \geq 0, \quad i = 1,2 \\
& v_{\mathrm{r}} \geq 0
\end{aligned} \tag{4.32}$$

O Problema (4.32) é equivalente ao problema (4.27), que havia sido obtido simplesmente substituindo as restrições estritas de (4.26) por restrições não estritas. Portanto, fica claro que uma solução alternativa para esse problema de otimização restrita pode ser obtida resolvendo o problema irrestrito (4.31), em vez de recorrer ao Teorema 2 e ao Positivstellensatz, como feito na Seção 4.2.2.

Assim, o ótimo do problema irrestrito (4.31), que é igual ao ótimo do problema restrito (4.32), é um limitante inferior para o ótimo do problema original (4.26).

Realizando as substituições de variáveis (4.30) nas equações (4.21), tem-se que os polinômios $\bar{\alpha}_i(y(x)) := \bar{\alpha}_i(x)$ são dados por

$$\begin{aligned}
\bar{\alpha}_1(x) &= 2x_4^2x_6^2 + (1 + x_5^2x_6^4)x_7^2 \\
\bar{\alpha}_2(x) &= x_6^2[(x_3^2x_8 + 2x_7^2)^2(1 + x_1^2)x_{10}^2 + (x_3^2x_8 - 2x_7^2)^2(1 + x_2^2)x_9^2][x_3^4x_5^2 + 4(1 + x_5^2x_6^4)] \\
&\qquad + 8(1 + x_1^2)(1 + x_2^2)x_3^4x_5^2[x_7^2(x_4^2(1 + x_5^2x_6^4) + x_5^2x_6^2x_7^2) + 2x_6^2x_8^2] \\
\bar{\alpha}_3(x) &= [(x_3^2x_8 + 2x_7^2)^2(1 + x_1^2)x_{10}^2 + (x_3^2x_8 - 2x_7^2)^2(1 + x_2^2)x_9^2] \\
&\qquad \times [x_5^2x_7^2(x_3^4(1 + x_5^2x_6^4) + 8x_6^4) + 8x_4^2x_6^2(1 + x_5^2x_6^4)] \\
&\qquad\qquad + 16(1 + x_1^2)(1 + x_2^2)x_3^4x_5^2x_7^2(x_4^2x_5^2x_6^2x_7^2 + x_8^2) \\
\bar{\alpha}_4(x) &= (x_3^2x_8 + 2x_7^2)^2x_{10}^2(1 + x_1^2)[(x_3^2x_5^2x_7^2 + 2x_8)^2 + 4x_8^2 + 8x_4^2x_5^2x_6^2x_7^2] \\
&\qquad + (x_3^2x_8 - 2x_7^2)^2x_9^2(1 + x_2^2)[(x_3^2x_5^2x_7^2 - 2x_8)^2 + 4x_8^2 + 8x_4^2x_5^2x_6^2x_7^2] \\
&\qquad\qquad + 2x_9^2x_{10}^2(1 + x_5^2x_6^4)(x_3^2x_8 + 2x_7^2)^2(x_3^2x_8 - 2x_7^2)^2 \\
\bar{\alpha}_5(x) &= x_9^2x_{10}^2x_6^2x_7^2(x_3^2x_8 + 2x_7^2)^2(x_3^2x_8 - 2x_7^2)^2
\end{aligned} \tag{4.33}$$

e os polinômios $\beta_j(y(x)) := \beta_j(x)$ são dados por

$$\begin{aligned}
\beta_1(x) &= x_6^2 \\
\beta_2(x) &= 4x_5^2 \\
\beta_3(x) &= 8(1 + x_1^2)(1 + x_2^2)x_3^4x_5^2x_6^2 \\
\beta_4(x) &= 4x_5^2x_6^2
\end{aligned} \tag{4.34}$$

Como preparação para a resolução do problema irrestrito (4.31), foram procuradas, numericamente, decomposições SOS para os polinômios (4.33). Verificou-se que todos os $\bar{\alpha}_i(x)$ das equações (4.33) são SOS. Isso também pode ser verificado por inspeção dos $\bar{\alpha}_i(x)$. Além disso, os $\beta_i(x)$ das equações (4.34) são claramente SOS.

Adicionalmente, durante os testes preparatórios para a resolução de (4.31), foi constatado numericamente que todos os seguintes polinômios são SOS:

$$\begin{aligned}
p_1(x) &= \bar{\alpha}_1(x)\bar{\alpha}_2(x) - \bar{\alpha}_3(x)\beta_1(x) \\
p_2(x) &= \bar{\alpha}_3(x)\bar{\alpha}_4(x) - \bar{\alpha}_2(x)\bar{\alpha}_5(x)\beta_2(x) \\
p_3(x) &= \bar{\alpha}_1(x)\bar{\alpha}_4(x) - \bar{\alpha}_5(x)\beta_4(x) \\
p_4(x) &= \bar{\alpha}_2(x)\bar{\alpha}_3(x) - \bar{\alpha}_1(x)\bar{\alpha}_4(x)\beta_3(x) \\
p_5(x) &= \bar{\alpha}_3(x)\bar{\alpha}_3(x) - \bar{\alpha}_1(x)\bar{\alpha}_5(x)\beta_5(x) \\
p_6(x) &= \bar{\alpha}_2(x)\bar{\alpha}_2(x) - \bar{\alpha}_4(x)\beta_6(x) \\
p_7(x) &= \bar{\alpha}_2(x)\bar{\alpha}_3(x) - \bar{\alpha}_5(x)\beta_7(x)
\end{aligned} \tag{4.35}$$

em que os $\bar{\alpha}_i(x)$ são dados por (4.33), os $\beta_i(x), i = 1, 2, 3, 4$ são dados por (4.34) e

$$\begin{aligned}
\beta_5(x) &= 32(1 + x_1^2)(1 + x_2^2)x_3^4x_5^4x_6^2 \\
\beta_6(x) &= 8(1 + x_1^2)(1 + x_2^2)x_3^4x_5^2x_6^4 \\
\beta_7(x) &= 32(1 + x_1^2)(1 + x_2^2)x_3^4x_5^4x_6^4
\end{aligned} \tag{4.36}$$

Os polinômios (4.35) constituem um conjunto de polinômios extremamente útil para realizar um *benchmark* de decomposições SOS, já que vários deles não são obviamente SOS à primeira vista, e portanto mereceram uma análise à parte, que será feita na Seção 4.3. Mas o importante no momento é ver que, utilizando a substituição de variáveis (4.30) na Equação (4.25), tem-se

$$\bar{\Delta}_4(x) = p_1(x)p_2(x) - \beta_3(x)p_3(x)^2 \tag{4.37}$$

em que p_1, p_2 e p_3 são dados por (4.35) e são verificadamente SOS, e $\beta_3(x)$ é SOS. Esse fato trás uma esperança grande de que $\bar{\Delta}_4(x)$ seja também SOS. Ora, caso seja possível encontrar uma decomposição SOS para $\bar{\Delta}_4(x)$, então será imediatamente verificada a não negatividade de $\bar{\Delta}_4(x)$.

Foi implementado computacionalmente um teste de decomposição SOS para o polinômio $\bar{\Delta}_4(x)$ da Equação (4.37) através do módulo SOS do *toolbox* Yalmip (Löfberg, 2009b), com o *solver* de programação semidefinida CSDP (Borchers, 1999). Foram utilizadas as opções de aproveitamento de esparsidade e de exploração de simetrias do Yalmip, e foi selecionado o modelo de problema SDP primal[3]. As tolerâncias do *solver* CSDP[4] foram colocadas em 10^{-12}.

[3]Os *flags* dessas opções são: `newton=1`, `congruence=2` e `model=1`.

[4]As variáveis de tolerância são: `axtol`, `atytol`, `objtol`,`minstepp` e `minstepd`.

Foi obtida uma decomposição SOS $z_\Delta(x)^T Q_\Delta z_\Delta(x)$ tal que o coeficiente de maior valor absoluto do polinômio obtido pela diferença

$$\bar{\Delta}_4(x) - z_\Delta(x)^T Q_\Delta z_\Delta(x)$$

foi 9.4×10^{-5}. Na prática, as decomposições numéricas SOS raramente são exatas, de modo que esse valor, chamado de resíduo da decomposição, representa a adequação entre a decomposição obtida e o polinômio original. Uma análise numérica sobre resíduos de decomposições SOS será apresentada na Seção 4.3.4. Dado o tamanho do resíduo obtido, que é várias ordens de grandeza menor que qualquer um dos coeficientes de (4.37), o polinômio $z_\Delta(x)^T Q_\Delta z_\Delta(x)$ é considerado uma perturbação muito pequena do polinômio (4.37), levando a crer que (4.37) seja de fato SOS. Devido ao tamanho da base monomial $z_\Delta(x)$, que contém 17440 monômios, o polinômio $z_\Delta(x)^T Q_\Delta z_\Delta(x)$ não será reproduzido aqui. Muitos desse monômios são residuais, devido ao caráter numérico da resolução.

Portanto, foi mostrado que o ótimo global dos problemas de otimização 4.31 e 4.32 é não negativo. Por conseguinte, o máximo limitante inferior do ótimo do problema (4.26) é não negativo.

4.3 Estudo sobre a implementação numérica de problemas de otimização polinomial

Esta seção visa apresentar uma série de experimentos numéricos que foram realizados com o intuito de testar programas de computador especializados em otimização polinomial. A motivação por trás desses experimentos foi a necessidade de resolver o problema de otimização abordado na Seção 4.2, que é um problema particularmente grande. Apesar de haver relaxações para problemas de otimização polinomial que exploram algoritmos eficientes de programação semidefinida, como é o caso da relaxação baseada em decomposições SOS da Seção 4.1, o tamanho dos problemas ainda é um fator limitante para sua implementação computacional. A questão da geração da base monomial para decomposições SOS, Seção 4.1.2, ilustra bem esse fato. Assim, os testes apresentados nesta seção visaram em primeiro lugar analisar se o problema abordado na Seção 4.2 poderia ser implementado computacionalmente por algum dos vários programas de otimização polinomial desenvolvidos e disponibilizados pela comunidade científica.

4.3.1 Programas de otimização polinomial

Existem vários programas de computador especializados em resolver problemas de otimização polinomial através de métodos de relaxação que resultam em problemas SDP. Os dois principais métodos de relaxação para problemas de otimização polinomial são o método SOS (Parrilo, 2003) e o método de Momentos de Matrizes (Lasserre, 2001). Os dois métodos são considerados duais, como explicado em Laurent (2009).

Tais programas permitem em geral que o problema de otimização seja formulado de várias maneiras diferentes. Por exemplo, podem ou não ser incluídas restrições no problema, pode ou não ser incluída uma função objetivo, etc. Os programas possuem em geral uma estrutura que envolve dois módulos distintos, cada qual com uma função específica:

- Um *toolbox* do programa Matlab é utilizado para estruturar o problema de otimização polinomial, apoiando-se em uma teoria de relaxação específica (SOS ou Momentos de Matrizes). Esses *toolboxes* possuem funções responsáveis por interpretar o problema de otimização polinomial e transformá-lo num problema de programação semidefinida (SDP). Os *toolboxes* podem conter funcionalidades para reduzir o tamanho e a complexidade do problema SDP, que, uma vez formulado, é resolvido internamente por *solvers* especializados.

- Um *solver* SDP especializado é responsável por receber o problema de programação semidefinida, construído pelo *toolbox* de otimização polinomial, e resolvê-lo. Existem muitos *solvers* especializados na resolução de problemas SDP, sendo que os algoritmos utilizados podem diferir de um *solver* para o outro (de acordo com Boyd e Vandenberghe (2004), ainda não houve convergência na área de otimização para um único algoritmo para a resolução de problemas SDP).

Os programas que foram testados neste trabalho são apresentados na Tabela 4.1. Além de programas de otimização polinomial baseados em relaxações SOS, foram também testados programas baseados em relaxações por Momentos de Matrizes.

A Tabela 4.1 também informa a disponibilidade, em cada *toolbox*, de duas funcionalidades opcionais importantes para a simplificação do problema SDP associado à relaxação do problema polinomial: aproveitamento de esparsidade e exploração de simetrias.

No caso dos dois *toolboxes* baseados em relaxações SOS, o aproveitamento de esparsidade ajuda a reduzir o tamanho do problema SDP, e é baseado nos Politopos de Newton, introduzidos na Seção 4.1.2. Os algoritmos utilizados para o cálculo das cascas convexas no SOSTools e no Yalmip/SOS são diferentes. Já o *toolbox* SparsePOP possui um método específico para o aproveitamento de esparsidade, baseado na teoria de Momento de Matrizes, que é detalhado em Waki et al. (2006).

O único *toolbox* que apresenta suporte para exploração de simetrias é o Yalmip/SOS. Essa funcionalidade consiste em simplificar o problema SDP através da bloco-diagonalização da matriz positiva semidefinida na restrição do problema. A funcionalidade de exploração de simetrias, que também é chamada de "congruência" pelo Yalmip/SOS, utiliza um algoritmo que explora a existência de simetrias nos sinais dos polinômios envolvidos no problema de otimização (Löfberg, 2009b).

É importante mencionar que além das funcionalidades desenvolvidas para simplificar o problema SDP associado ao problema de decomposição SOS, o Yalmip/SOS também permite a escolha de dois tipos de modelo para a elaboração do problema SDP: modelo primal (que é a forma padrão (4.6)) e modelo dual. O modelo primal é o mesmo modelo utilizado pelo SOSTools.

Tabela 4.1: Funcionalidades dos *Toolboxes* de otimização polinomial do Matlab.

Toolbox	Teoria	Aproveitamento de Esparsidade	Exploração de Simetrias
SOSTools (Papachristodoulou et al., 2013)	SOS	Sim	Não
Yalmip/SOS (Löfberg, 2009b)	SOS	Sim	Sim
Yalmip/MOM (Löfberg, 2004)	Momentos	Não	Não
GloptiPoly (Henrion et al., 2009)	Momentos	Não	Não
SparsePOP (Waki et al., 2008)	Momentos	Sim	Não

O *toolbox* Yalmip contém um módulo de otimização por decomposições SOS (abreviado

Yalmip/SOS) e um módulo distinto de otimização por Momentos de Matrizes (Yalmip/MOM), que serão tratados por simplicidade como *toolboxes* diferentes.

As principais limitações físicas de um sistema computacional são sua capacidade de processamento e a sua quantidade de memória disponível. A capacidade de processamento influencia no tempo que será necessário para a resolução de um dado problema, enquanto a quantidade de memória disponível limita o tamanho dos problemas que podem ser implementados no sistema. Assim, duas variáveis importantes que serão utilizadas na comparação dos *toolboxes* da Tabela 4.1 são o tempo e a memória necessários para resolver um dado problema.

Os testes utilizando os toolboxes da Tabela 4.1 serão apresentados nas próximas seções. Na Seção 4.3.2, será analisada a performance dos programas no que concerne a quantidade de memória necessária para a resolução de problemas de otimização com polinômios pouco esparsos. Na Seção 4.3.3, será analisada a capacidade de cada programa de simplificar os problemas SDP associados a problemas de otimização com polinômios esparsos.

A máquina utilizada para realizar os testes numéricos possui um processador Intel® Core™ i7-2600K de quatro núcleos e oito threads, com frequência de 3.40 GHz e 64 bits no conjunto de instruções. A quantidade de memória RAM total da máquina é 32 Gb, sendo que um espaço SWAP de 250 Gb pode ser liberado caso necessário. A máquina opera com a distribuição Slackware versão 13.37.0 do sistema operacional Linux versão 2.6.37.6.

4.3.2 Testes para análise de consumo de memória

Nesta seção, será investigado de que maneira a quantidade de memória exigida pelos programas de otimização polinomial é afetada pelo tamanho dos polinômios envolvidos nos problemas de otimização. Por tamanho de um polinômio, refere-se a dois parâmetros independentes: o grau do polinômio, e o número de variáveis do polinômio. Em especial, essa investigação será feita utilizando polinômios pouco esparsos, visando adquirir uma ideia da quantidade de memória exigida, no pior caso, para a resolução do problema de otimização da Seção 4.2.

A estratégia adotada para fazer a investigação de consumo de memória se baseou em criar uma forma polinomial conhecida a priori, construída a partir de um grau par $2d$, $d \in \mathbb{N}$, e de um

número de variáveis $n \in \mathbb{N}$. A forma escolhida foi

$$p_{n,d}(x) = \sum_{i=1}^{d} (x^T D_i x)^i, \quad x \in \mathbb{R}^n, \quad \deg(p_{n,d}) = 2d \tag{4.38}$$

em que $D_i \in \mathbb{R}^{n \times n}$, $i = 1, \ldots, d$, é uma matriz diagonal com elementos reais positivos. A forma (4.38) é SOS. Por exemplo, para $n = 2$, $d = 3$, $A_1 = A_2 = A_3 = I$,

$$\begin{aligned} p_{2,3}(x) &= (x_1^2 + x_2^2)^3 + (x_1^2 + x_2^2)^2 + x_1^2 + x_2^2 \\ &= (x_1^2 + x_2^2)(x_1^2 + x_2^2)^2 + (x_1^2 + x_2^2)^2 + x_1^2 + x_2^2 \\ &= (x_1(x_1^2 + x_2^2))^2 + (x_2(x_1^2 + x_2^2))^2 + (x_1^2 + x_2^2)^2 + x_1^2 + x_2^2 \end{aligned}$$

Usando a forma polinomial (4.38), a investigação do consumo de memória foi feita com base na resolução do problema

$$\begin{aligned} \max_{\rho \in \mathbb{R}} \quad & \rho \\ \text{sujeito a} \quad & p_{n,d}(x) - \rho \geq 0 \end{aligned} \tag{4.39}$$

É importante mencionar que, para quaisquer n e d, o polinômio $p_{n,d}(x) - \rho$ no Problema (4.39), com $p_{n,d}(x)$ dado por (4.38), não possui uma estrutura que possa ser explorada por algoritmos de Politopos de Newton, introduzidos na Seção 4.1.2. Isso se dá devido à existência simultânea do monômio de grau zero ρ e dos monômios de grau $2d$ dados por x_i^{2d}, que fazem com que a casca convexa dos expoentes dos monômios de $p_{n,d}(x) - \rho$ em $\mathbb{R}^n$ englobe todos os pontos dos expoentes de monômios de grau até $2d$. Assim, não é possível excluir com base nos Politopos de Newton nenhum monômio da base monomial $z(x)$ da eventual decomposição SOS de $p_{n,d}(x) - \rho$. Por essa razão, o polinômio $p_{n,d}(x) - \rho$ (e também $p_{n,d}$) é dito pouco esparso. Esse fato faz com que os *toolboxes* de otimização por relaxações SOS (Tabela 4.1), em teoria, não possam se aproveitar da vantagem oferecida pelo aproveitamento de esparsidade por Politopos de Newton.

O procedimento desenvolvido para testar o consumo de memória exigido por cada *toolbox* na resolução de problemas de otimização polinomial pouco esparsos envolveu resolver o problema (4.39) numericamente, usando diversos valores de n e d. A forma escolhida de $p_{n,d}$ possui ótimo global em $x = 0$, tornando fácil averiguar se as resoluções numéricas foram bem-sucedidas. Os *toolboxes* que utilizam a teoria SOS e os que utilizam a teoria de Momentos de Matrizes diferem essencialmente na maneira que a restrição $(\geq)$ do problema (4.39) é relaxada. Por exemplo, para o

SOSTools e o Yalmip/SOS, essa restrição é substituída por um teste de existência de decomposição SOS. Visto que $p_{n,d}$ é SOS, isso garante que pelo menos para o SOSTools e o Yalmip/SOS, o Problema (4.39) seja factível.

O procedimento de testes de memória é descrito abaixo:

- Para cada um dos *toolboxes* da Tabela 4.1, foi escrita uma função Matlab para resolver o Problema (4.39). Essa função aceita n e d como argumentos, monta $p_{n,d}$ de acordo com (4.38), e resolve o problema (4.39) usando as funções apropriadas de cada *toolbox*. Os elementos diagonais das matrizes D_i foram escolhidos com base numa lista de números reais gerados aleatoriamente entre 1 e 3. Essa lista foi gerada previamente aos testes e armazenada, para que todos os programas de otimização utilizassem exatamente os mesmos polinômios.

- Usando as funções de Matlab descritas acima, tentou-se resolver o Problema (4.39) utilizando diferentes valores de n e d para a forma polinomial 4.38. Foram utilizadas todas as as combinações entre o número de variáveis n e o grau $2d$ dadas por

  $$n = 5, 6, 7, \ldots, 14, 15, 16$$

  e

  $$2d = 2, 4, 6, 8, 10, 12, 14, 16$$

- As funções Matlab foram programadas de modo que a resolução do Problema (4.39) foi feita utilizando as configurações padrão de cada *toolbox*. Em relação aos *toolboxes* que possuem funcionalidades especiais, as configurações padrão são:

  - SOSTools: Aproveitamento de esparsidade desativado.
  - Yalmip/SOS: Aproveitamento de esparsidade e exploração de simetrias[5] ativados. O modelo SDP utilizado foi o modelo primal.
  - SparsePOP: Aproveitamento de esparsidade ativado.

- O *solver* SDP utilizado internamente pelos *toolboxes* foi o SeDuMi (Sturm, 1999).

- Um script de Linux foi criado para lançar as funções Matlab mencionadas nos itens anteriores. Uma instância do programa Matlab era aberta a cada chamada de função, e fechada

[5] O flag utilizado foi `congruence = 2`.

ao término da execução desta. Isso foi feito para garantir que toda a memória associada à resolução de um dado problema fosse esvaziada do sistema após sua resolução.

- Paralelamente à chamada das funções Matlab, o script de Linux armazenava em um arquivo texto, a cada 0.5 segundos, o consumo de memória do processo Matlab.
- A memória permitida para a resolução do problema (4.39) foi limitada a 32 Gb (a quantidade de memória RAM da máquina utilizada), e o número de núcleos do processador foi limitado a um, para maior controle dos dados.

Os resultados dos testes realizados utilizando o procedimento descrito acima forneceram um conjunto de dados relacionando o consumo máximo de memória necessário para a resolução do Problema (4.39) por cada um dos *toolboxes*, para cada par de variáveis n e $2d$.

Os dados obtidos podem ser apresentados de duas formas. Para um dado grau $2d$ fixo, pode-se mostrar a evolução do consumo máximo de memória em relação ao aumento do número de variáveis n. Para um dado número de variáveis fixo n, pode-se mostrar a evolução do consumo máximo de memória em relação ao aumento do grau $2d$. Os dados serão apresentados levando em conta o número de variável fixo n. Para cada conjunto de dados, fixando o número de variáveis n, foi feito um ajuste polinomial quadrático. O objetivo desse ajuste é poder extrapolar os dados de consumo máximo de memória para além dos graus e dos números de variáveis que foram levados em consideração nos testes. Isso será útil, em particular, para obter um valor para a quantidade de memória exigida no "pior caso" para a resolução do problema de otimização da Seção 4.2.

As Figuras 4.2 a 4.6 mostram o consumo máximo de memória em função do grau $2d$, para $n = 6, 7, 8, 9, 10$ fixos, usando o *toolbox* SOSTools. São mostrados os dados e os ajustes quadráticos feitos com base nos dados. Por brevidade, as fórmulas dos ajustes obtidos são apresentadas na Tabela C.2 do Apêndice C. Cada ponto na curva de dados representa uma resolução bem sucedida do problema (4.39) com o n da Figura e o grau $2d$ do gráfico. Como a memória do sistema foi limitada a 32 Gb, que é aproximadamente $10^{4,5}$ Mb, nenhum ponto foi obtido acima desse valor de memória. Pode-se ver que quando o número de variáveis cresce, menos problemas podem ser resolvidos com a memória limitada a 32 Gb: para $n = 6$, o grau máximo que permite uma resolução com menos de 32 Gb de memória é $2d = 12$, para $n = 7$ é $2d = 10$, para $n = 8$ é $2d = 8$, etc.

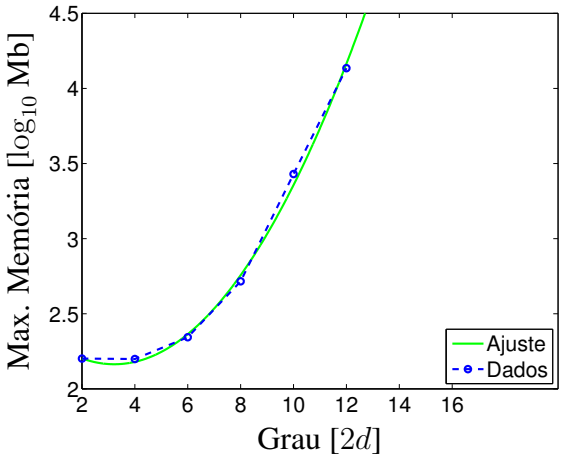

Figura 4.2: Consumo de memória pelo SOSTools na resolução de (4.39): $n = 6$.

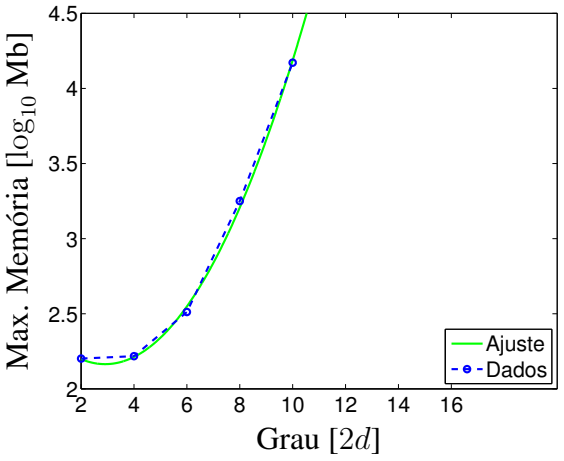

Figura 4.3: Consumo de memória pelo SOSTools na resolução de (4.39): $n = 7$.

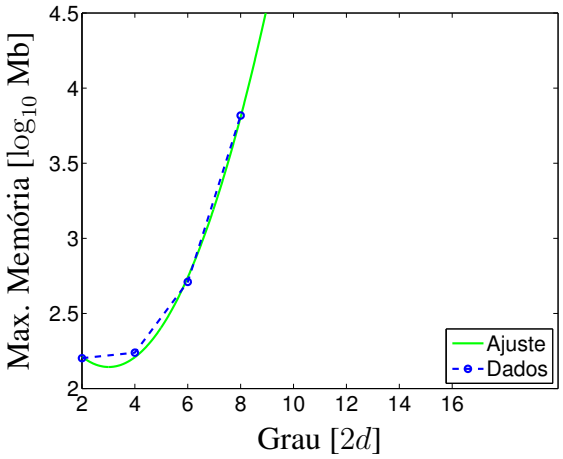

Figura 4.4: Consumo de memória pelo SOSTools na resolução de (4.39): $n = 8$.

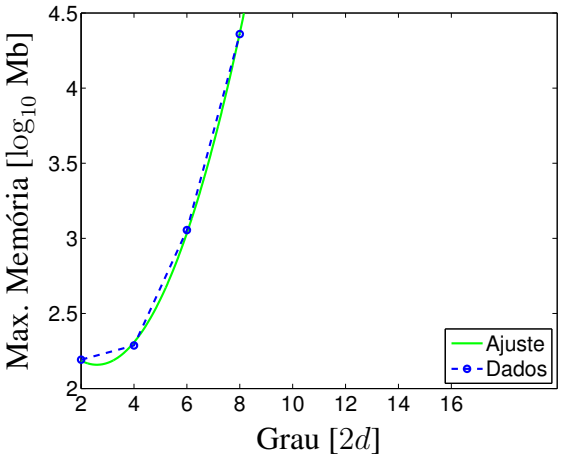

Figura 4.5: Consumo de memória pelo SOSTools na resolução de (4.39): $n = 9$.

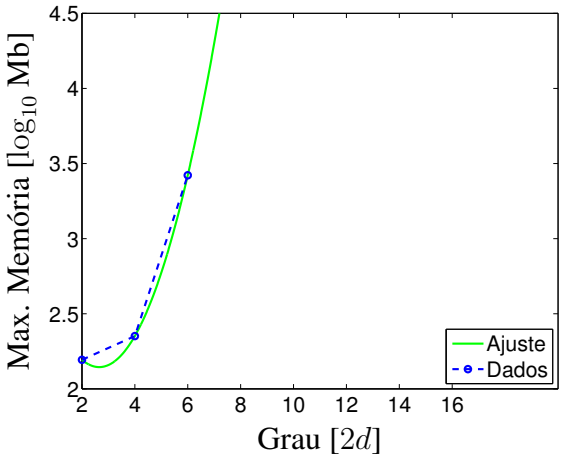

Figura 4.6: Consumo de memória pelo SOSTools na resolução de (4.39): $n = 10$.

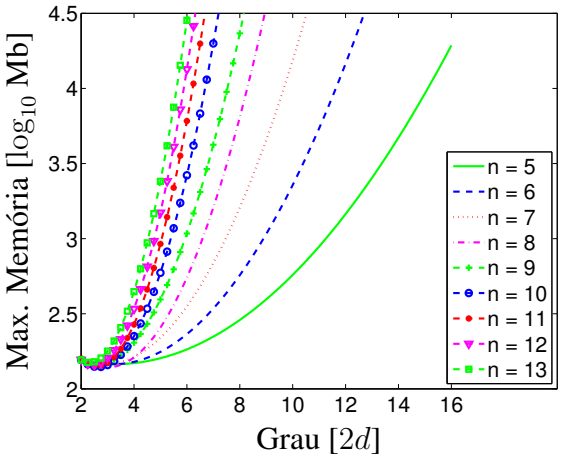

Figura 4.7: Consumo de memória pelo SOSTools na resolução de (4.39).

Ainda nas Figuras 4.2 a 4.6, pode-se ver que curvas quadráticas ajustam muito bem os dados numéricos. Na Figura 4.7 são mostrados todos os ajustes obtidos para $n = 5, \ldots, 13$ (para $n > 13$ apenas dois pontos de dados foram obtidos, não possibilitando ajustes). Pode-se ver claramente que aumentar o número de variáveis fixo n tem o efeito de aumentar a inclinação da parábola convexa que ajusta o máximo de memória em função do grau $2d$.

As Figuras 4.8 a 4.12 mostram o consumo máximo de memória em função de $n = 6$ a $n = 10$ fixos, usando o *toolbox* Yalmip/SOS. Por brevidade, as fórmulas dos ajustes obtidos são apresentadas na Tabela C.1 do Apêndice C. Na Figura 4.13 são mostrados todos os ajustes obtidos para $n = 5, \ldots, 16$. Observa-se que há uma diferença grande em relação às curvas obtidas com o SOSTools. O Yalmip/SOS consegue resolver, para um dado n fixo, mais problemas que o SOSTools, pois há uma menor inclinação da parábola que ajusta o máximo de memória em função do grau $2d$. Como ambos os toolboxes trabalham com a teoria de decomposições SOS, a razão da maior eficiência apresentada pelo Yalmip deve estar em suas funcionalidades adicionais, que são ativadas por padrão. Mas os Politopos de Newton não são explorados na resolução do Problema (4.39), devido à estrutura escolhida para $p_{n,d}$ na Equação (4.38). Isso foi confirmado analisando o diagnóstico que os *toolboxes* fornecem: nem o SOSTools nem o Yalmip/SOS reduziram o problema com base nos Politopos de Newton. Assim, a maior eficiência do Yalmip/SOS pode ser explicada pela sua funcionalidade de exploração de simetrias. Isso mostra que a exploração de simetrias realizada pelo Yalmip/SOS é eficiente, mesmo quando o problema de otimização polinomial é muito pouco esparso.

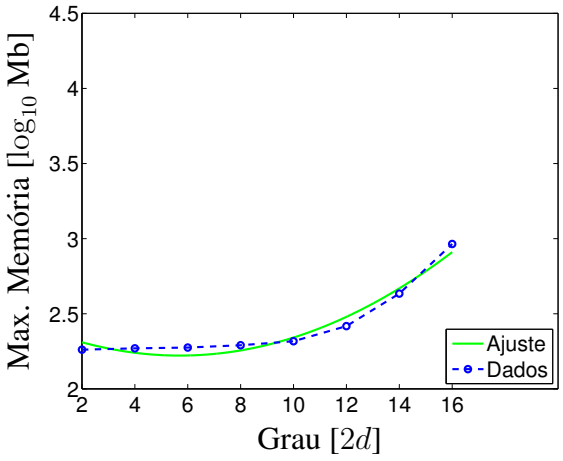

Figura 4.8: Consumo de memória pelo Yalmip/SOS na resolução de (4.39): $n = 6$.

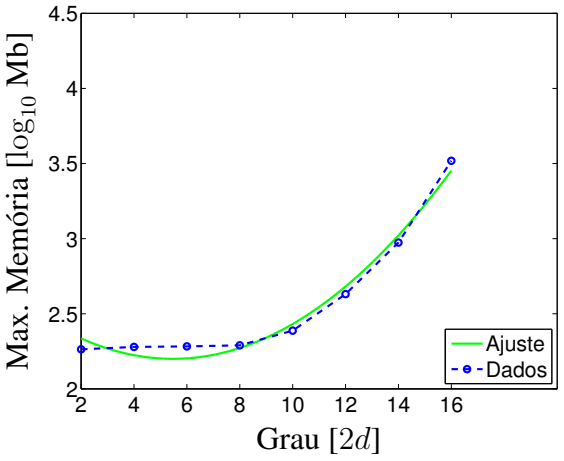

Figura 4.9: Consumo de memória pelo Yalmip/SOS na resolução de (4.39): $n = 7$.

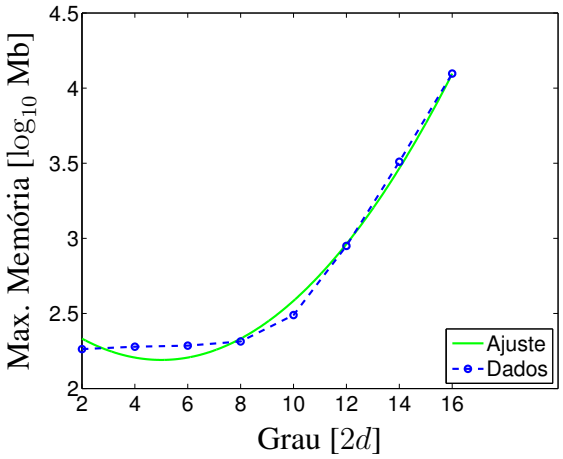

Figura 4.10: Consumo de memória pelo Yalmip/SOS na resolução de (4.39): $n = 8$.

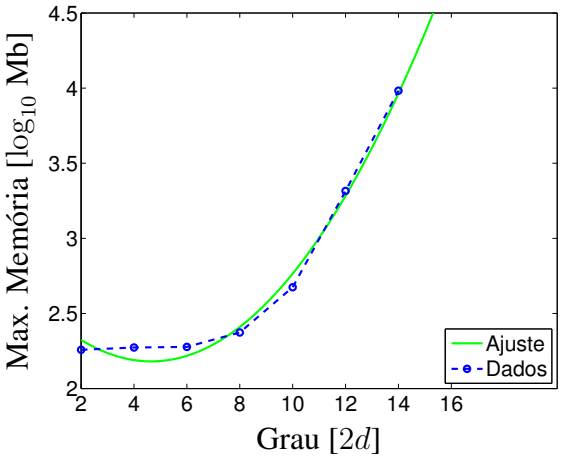

Figura 4.11: Consumo de memória pelo Yalmip/SOS na resolução de (4.39): $n = 9$.

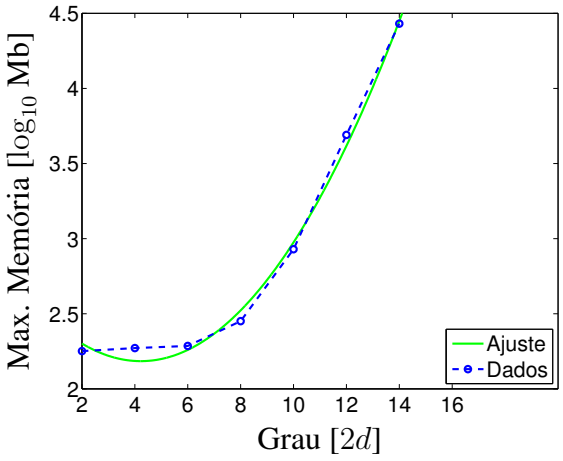

Figura 4.12: Consumo de memória pelo Yalmip/SOS na resolução de (4.39): $n = 10$.

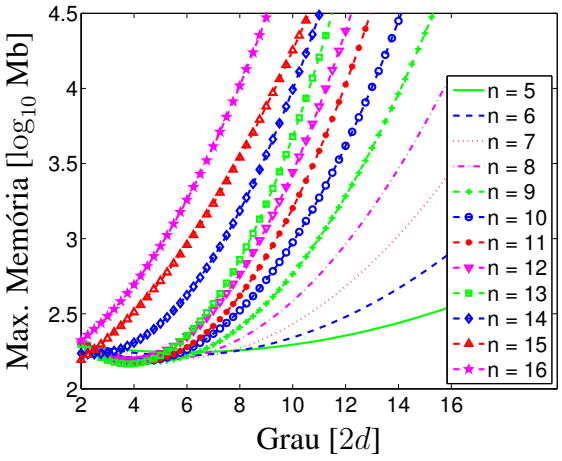

Figura 4.13: Consumo de memória pelo Yalmip/SOS na resolução de (4.39).

No Apêndice C, as Figuras C.1 a C.5 mostram o consumo máximo de memória em função de $n = 6, 7, 8, 9, 10$ fixos, usando o *toolbox* Yalmip/MOM. Na Figura C.6 são mostrados todos os ajustes obtidos para $n = 5, \ldots, 13$. As Figuras C.7 a C.11 mostram o consumo máximo de memória em função de $n = 6, 7, 8, 9, 10$ fixos, usando o *toolbox* Gloptipoly. Na Figura C.12 são mostrados todos os ajustes obtidos para $n = 5, \ldots, 13$. As Figuras C.13 a C.17 mostram o consumo máximo de memória em função de $n = 6, 7, 8, 9, 10$ fixos, usando o *toolbox* SparsePOP. Na Figura C.18 são mostrados todos os ajustes obtidos para $n = 5, \ldots, 10$.

Pôde-se constatar que os resultados dos *toolboxes* Gloptipoly, Yalmip/MOM e SparsePOP são muito parecidos com os resultados do SOSTools. Isso fica mais evidente vendo as Figuras 4.14 e 4.15, que mostram como exemplos os ajustes obtidos para $n = 6$ e $n = 10$ com todos os *toolboxes*. Em primeiro lugar, isso mostra que não houve aproveitamento relevante de esparsidade do problema por parte do SparsePOP, que por padrão tenta aproveitá-la. O Gloptipoly e o Yalmip/MOM, que trabalham com o mesmo tipo de relaxação que o SparsePOP (Momentos de Matrizes), não possuem essa funcionalidade, e não obstante tiveram um desempenho muito similar em termos de consumo de memória. Em segundo lugar, é notável que o SOSTools, que trabalha com o método de relaxação SOS, consuma praticamente a mesma quantidade de memória que os três *toolboxes* que trabalham com o método de relaxação por Momentos de Matrizes. Por fim, destaca-se pela eficiência o Yalmip/SOS, que consumiu menos memória para resolver os mesmos problemas que os outros *toolboxes*. Como já foi mencionado, isso pode se dever em grande parte à sua funcionalidade de exploração de simetrias, que consegue economizar uma grande quantidade

de memória mesmo quando o polinômio envolvido no problema de otimização é pouco esparso.

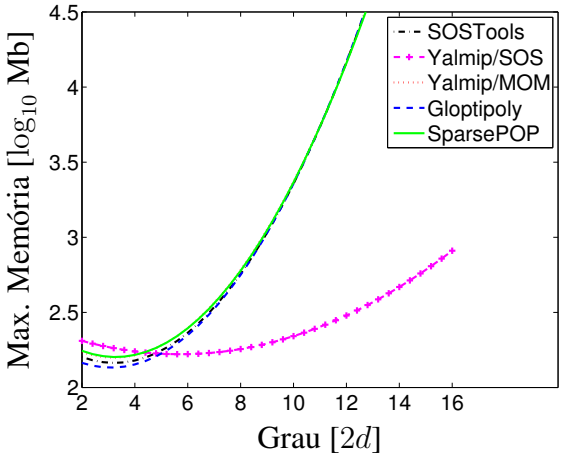

Figura 4.14: Comparação do consumo de memória pelos *toolboxes*: $n = 6$.

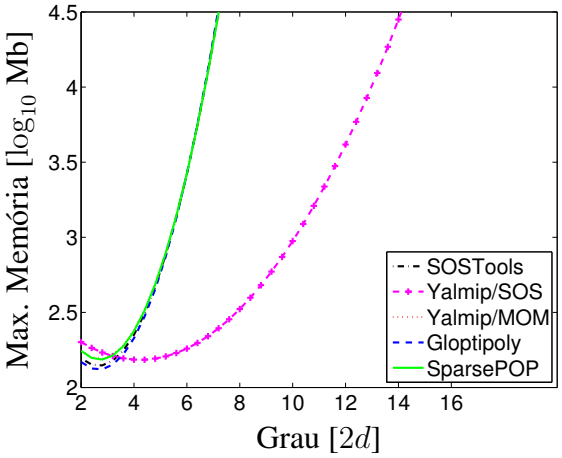

Figura 4.15: Comparação do consumo de memória pelos *toolboxes*: $n = 10$.

Como apoio à Seção 4.2, será estimado o quanto de memória é necessário para a resolução da relaxação SOS 4.29. O Problema (4.29) é muito pouco esparso, como explicado anteriormente, devido aos polinômios σ_j. Pode-se utilizar as curvas ajustadas nesta seção, obtidas resolvendo problemas com um polinômio pouco esparso, para realizar uma estimativa do quanto de memória seria necessária para resolver o Problema (4.29). O número de variáveis envolvido no problema é $n = 10$. A função quadrática da curva de ajuste para $n = 10$ do Yalmip/SOS, que foi o *toolbox* com o melhor desempenho em termos de memória consumida, é dada por

$$\log_{10} \text{Memória } = 0.023705(2d)^2 - 0.20033(2d) + 2.6077 \text{ Mb} \tag{4.40}$$

Substituindo $2d = 40$, que é o menor valor de $2T$ possível para resolver o Problema (4.29), tem-se

$$\text{Memória } = 10^{32.52} \text{ Mb} \tag{4.41}$$

Esse resultado mostra de maneira inequívoca que é impossível de se resolver o Problema (4.29) com uma quantidade razoável de memória.

4.3.3 Testes para simplificação do problema SDP

Como visto na Seção 4.3.2, a quantidade de memória necessária para resolver um problema de otimização polinomial se torna proibitiva quando o grau e o número de variáveis do polinômio a

ser otimizado é grande. Alguns dos *toolboxes* introduzidos nas seções anteriores possuem certas opções que exploram propriedades do problema, acarretando a redução no custo de memória e tempo necessários para sua resolução. Os *toolboxes* que possuem algum tipo de opção desse tipo são o SOSTools, o Yalmip/SOS, e o SparsePOP.

Esses três *toolboxes* possuem funcionalidades para aproveitar a esparsidade do problema de otimização polinomial. O aproveitamento de esparsidade baseado no conceito de politopos de Newton, muito útil na busca por decomposições SOS, foi apresentado na Seção 4.1.2. Tanto o SOSTools e o Yalmip/SOS possuem suporte para redução do problema SDP através de Politopos de Newton. Já o SparsePOP, que trabalha com a teoria de momentos de matrizes, explora a esparsidade utilizando o conceito de esparsidade correlativa, cuja teoria (Waki et al., 2008) não será apresentada neste trabalho. Em todos os *toolboxes*, uma simples *flag* booleana é utilizada para ativar ou desativar a capacidade de redução do problema por esparsidade.

O Yalmip/SOS possui uma funcionalidade adicional, que é a exploração das simetrias de sinais no polinômio. Essa opção melhora o condicionamento do problema SDP, e pode reduzir a quantidade de memória e esforço computacional exigidos para resolver o problema de otimização polinomial. O Yalmip/SOS permite que se escolha os valores 0, 1 ou 2 para o *flag* que controla a opção de exploração de simetrias, sendo que o valor 0 desabilita essa funcionalidade por completo. O Yalmip/SOS também permite a escolha do modelo primal ou dual para o problema SDP, sendo que a escolha automática feita pelo próprio programa quase sempre é pelo primal. Visto que o SOSTools utiliza o modelo primal, este foi o modelo escolhido para o funcionamento do Yalmip/SOS.

Com o intuito de explorar as opções relevantes dos *toolboxes* SOSTools, Yalmip/SOS e SparsePOP, foram testados quatro problemas de otimização polinomial, e foram gravados a quantidade de memória e o tempo necessário para sua resolução. Foram testados dois problemas utilizando polinômios esparsos, e dois problemas utilizando dois polinômios pouco esparsos tirados dos testes de memória da Seção anterior. Os problemas testados a seguir possuem a forma básica

$$\begin{aligned} \max_{\rho \in \mathbb{R}} \quad & \rho \\ \text{sujeito a} \quad & p(x) - \rho \geq 0 \end{aligned} \tag{4.42}$$

e todos eles possuem mínimo global em zero para facilitar a verificação do sucesso da resolução do problema.

O polinômio $p(x)$ e as variáveis x serão especificados para cada problema. Os resultados apresentarão o consumo máximo de memória (Max. Memo), o tempo de pré-processamento (t Pre), associado aos algoritmos que exploram esparsidade e simetrias, o tempo de resolução do problema SDP (foi utilizado o SeDuMi como *solver* em todos os casos), o tempo de pós-processamento (t Pos), e o tempo total. Apenas um núcleo de processador estava habilitado para os testes desta seção.

Polinômio muito esparso em 3 variáveis de grau 42

- Variáveis: $x = (x_1, x_2, x_3)^T$.
- Polinômio: $p(x) = x_1^2 + x_2^4 + x_3^6 + x_1^2x_2^4x_3^2 + x_1^8x_2^{16}x_3^4 + x_2^{20}x_3^2 + x_3^{30}x_1^4 + x_1^{40}x_2^2$

Para este caso, pode-se ver na Tabela 4.2 que quando apenas a funcionalidade de aproveitamento de esparsidade está ativa, o Yalmip/SOS, o SOSTools e o SparsePOP consomem aproximadamente a mesma quantidade de memória para a resolução do problema (em torno de 1 Gb). Isso mostra que os algoritmos de exploração de esparsidade nos três programas têm eficiências similares. Quando todas as funcionalidades estão inativas, o SOSTools e o Yalmip/SOS consomem quantidades similares de memória. Isso era de se esperar, já que ambos trabalham com o mesmo tipo de teoria de relaxação. Não houve mudança significativa nos dados do SparsePOP para quando o aproveitamento de esparsidade estava ativado e para quando ele estava desativado. Isso leva a crer que o SparsePOP sempre trabalha aproveitando a esparsidade (talvez a *flag* relevante não esteja funcionando adequadamente).

Tabela 4.2: Resultados: polinômio esparso em 3 variáveis de grau 42.

Programa	Espars.	Simet.	Max Memo	t Pre	t Sedumi	t Pos	t Total
SOStools	1	-	0.997 Gb	46s	20m 9s	0.17s	20m 55s
SOStools	0	-	6.266 Gb	1h 8m 27s	8h 44m 50s	0.18s	9h 53m 18s
SparsePop	1	-	1.008 Gb	34m 34s	19m 36s	0.12s	54m 10s
SparsePop	0	-	1.006 Gb	34m 31s	19m 37s	0.16s	54m 9s
Yalmip/sos	1	2	224.45 Mb	4.73s	9.02s	0.45s	14.21s
Yalmip/sos	1	1	221.11 Mb	4.71s	8.71s	0.37s	13.79s
Yalmip/sos	1	0	1.058 Gb	10.4s	15m 53s	1.34s	16m 5s
Yalmip/sos	0	2	508.11 Mb	6.16s	2m 30s	5.94s	2m 42s
Yalmip/sos	0	1	543.66 Mb	6.09s	2m 29s	5.88s	2m 41s
Yalmip/sos	0	0	6.655 Gb	2m 48s	8h 15m 26s	16.5s	8h 18m 30s

O que mais chama a atenção, no entanto, é a eficiência da funcionalidade de simetrias do Yalmip/SOS. Usada individualmente, a funcionalidade de exploração de simetrias aproveita 500 Mb a mais de memória que a funcionalidade de exploração de esparsidade usada individualmente. Quando combinadas, as duas funcionalidades do Yalmip/SOS fazem com que este exija cinco vezes menos memória que o SOSTools com sua funcionalidade única de exploração de esparsidade.

Polinômio pouco esparso em 5 variáveis de grau 16

- Variáveis: $x = (x_1, x_2, x_3, x_4, x_5)^T$.
- Polinômio: $p(x) = \sum_{i=1}^{8} (x^T x)^i$

Este caso apresenta um polinômio tirado dos testes da Seção 4.3.2. Havia sido observado na Seção 4.3.2 que os polinômios testados usando a forma polinomial (4.38) não tinham estrutura aproveitável pelos algoritmos de Politopo de Newton. A Tabela (4.3) comprova este fato: o consumo máximo de memória do SOSTools e do Yalmip/SOS, que usam algoritmos de Politopo de Newton para aproveitar esparsidade, é praticamente o mesmo (em torno de 16.5 Gb) quando o aproveitamento de esparsidade está ligado isoladamente e quando todas as funcionalidades estão desligadas. Adicionalmente, a memória consumida pelo SparsePOP também é praticamente a

mesma. Isso leva a crer que o método de aproveitamento de esparsidade do SparsePOP, que usa teoria de Momentos de Matrizes, tem um desempenho similar ao método dos Politopos de Newton da teoria SOS.

Tabela 4.3: Resultados: polinômio pouco esparso em 5 variáveis de grau 16.

Programa	Espars.	Simet.	Max Memo	t Pre	t Sedumi	t Pos	t Total
SOStools	1	-	16.19 Gb	24m 34s	11h 04m 53s	0.08s	11h 29m 27s
SOStools	0	-	16.16 Gb	24m 26s	10h 24m 01s	0.16s	10h 46m 38s
SparsePop	1	-	16.22 Gb	13.55s	11h 10m 01s	0.05s	11h 10m 13s
SparsePop	0	-	16.21 Gb	13.68s	11h 07m 54s	0.05s	11h 08m 08s
Yalmip/sos	1	2	340.3 Mb	2.55s	21s	2.61s	26.16s
Yalmip/sos	1	1	339.4 Mb	2.51s	21s	2.64s	26.16s
Yalmip/sos	1	0	16.48 Gb	2m 34s	13h 24m 34s	8.46s	13h 27m 16s
Yalmip/sos	0	2	325.6 Mb	2.42s	21.02s	2.63s	26.07s
Yalmip/sos	0	1	339.29 Mb	2.45s	21.01s	2.61s	26.07s
Yalmip/sos	0	0	16.45 Gb	2m 34s	13h 32m 43s	8.42s	13h 35m 26s

Novamente a exploração de simetrias do Yalmip/SOS mostra que faz muita diferença no consumo de memória: foram economizados 16 Gb de memória apenas usando essa funcionalidade. Observa-se, por fim que o tempo de resolução total é imensamente reduzido quando a exploração de simetrias é ativada.

Polinômio pouco esparso em 16 variáveis de grau 4

- Variáveis: $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16})^T$.
- Polinômio: $p(x) = \sum_{i=1}^{2} (x^T x)^i$

Novamente foi usado um polinômio proveniente da Seção 4.3.2. Comparando as Tabelas 4.3 e 4.2, pode-se ver que a análise é praticamente a mesma que aquela feita para o caso anterior. O polinômio acima também não permitiu aproveitamento de esparsidade por parte de nenhum dos *toolboxes*. Porém, é digno de nota a diferença na memória consumida pelo Yalmip/SOS quando

se utiliza a opção 1 ou 2 da funcionalidade de exploração de simetrias. Isso não havia sido notado nos resultados anteriores, em que a quantidade de memória usando a opção 1 ou 2 era praticamente a mesma. Pode-se ver que a maior parte do tempo exigido pelo Yalmip/SOS para resolver esse problema foi gasta no pré-processamento.

Tabela 4.4: Resultados: polinômio pouco esparso em 16 variáveis de grau 4.

Programa	Espars.	Simet.	Max Memo	t Pre	t Sedumi	t Pos	t Total
SOStools	1	-	1.039 Gb	3s	10m 8s	0.12s	10m 11s
SOStools	0	-	1.038 Gb	2.79s	10m 6s	0.13s	10m 9s
SparsePop	1	-	1.054	0.89s	9m 6s	0.14s	9m 7s
SparsePop	0	-	1.03 Gb	0.88s	9m 12s	0.15s	9m 13s
Yalmip/sos	1	2	411.9 Mb	3s	0.4s	0.24s	3.62s
Yalmip/sos	1	1	192.6 Mb	1.67s	0.5	0.27	2.44s
Yalmip/sos	1	0	1.07 Gb	1.49s	10m 22s	0.28s	10m 24s
Yalmip/sos	0	2	411.61 Mb	2.82s	0.45s	0.23s	3.5s
Yalmip/sos	0	1	186.4 Mb	1.55s	0.5s	0.25s	2.3s
Yalmip/sos	0	0	1.068 Gb	0.88s	9m 12s	0.15s	9m 13s

Polinômio esparso em 10 variáveis de grau 42

- Variáveis: $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10})^T$.
- Polinômio: $p(x) = x_1^2 + x_2^4 + x_3^6 + x_4^8 + x_6^{10} + x_7^{12} + x_8^{14} + x_9^4 + x_{10}^{10} + x_1^2x_2^4x_3^2x_7^4x_{10}^2 + x_2^{14}x_1^8x_3^4 + x_2^{14}x_3^{10} + x_1^4x_3^{14}x_6^8 + x_1^{14}x_2^2 + x_2^{14}x_9^{10}x_5^{10} + x_4^2x_5^4x_6^4x_7^6 + x_8^2x_5^8x_9^6x_{10}^4 + x_4^2x_5^4x_6^4x_7^6 + x_1^6x_3^8x_5^4x_7^8x_9^4x_4^{10}x_8^2 + x_1^8x_7^{10}x_3^6x_8^2x_{10}^{12} + x_1^2x_2^4x_3^2x_4^4x_5^2x_6^4x_7^2x_8^4x_9^2x_{10}^4$

O único *toolbox* capaz de resolver este caso utilizando menos de 32 Gb de memória foi o Yalmip/SOS, como se vê pela Tabela 4.5. Mesmo com ambas as suas funcionalidades ativas, o consumo de memória foi elevado: 16 Gb. Não se sabe quanto seria necessário de memória para o SOSTools e o SparsePOP resolverem este problema, já que os programas foram interrompidos quando a memória demandada foi maior do que a disponível no sistema.

Tabela 4.5: Resultados: polinômio esparso em 10 variáveis de grau 42.

Programa	Espars.	Simet.	Max Memo	t Pre	t Sedumi	t Pos	t Total
SOStools	1	-	> 32 Gb	46m 51s*	—	—	—
SparsePop	1	-	> 32 Gb	12d 06h 21m 18s*	—	—	—
Yalmip/sos	1	2	15.62 Gb	23m 27s	3m 10s	19s	26m 57s
Yalmip/sos	1	1	16.13 Gb	23m 23s	3m 14s	18s	26m 54s

*No momento da quebra por falta de memória.

Conclusão dos testes

O Yalmip/SOS apresentou um desempenho superior em todos os testes, devido à sua funcionalidade de exploração de simetrias. Exceto no teste da Tabela 4.4, não se verificou muita diferença entre as opções 1 e 2 do exploração de simetrias. O fato de que o Yalmip/SOS conseguiu resolver um problema envolvendo um polinômio com muitas variáveis e de grau muito alto levou a crer que ele seria capaz de resolver o problema da Seção 4.2.3 com uma quantidade realista de memória, e isso de fato foi verificado, como apresentado na Seção 4.2.3.

4.3.4 Qualidade das decomposições e *solvers* SDP

Na prática, raramente as decomposições SOS encontradas por programas de computador são exatamente iguais ao polinômio original que desejava-se decompor. Isso pode acontecer devido aos critérios de parada dos *solvers* SDP, responsáveis pela resolução dos problemas de programação semidefinida associados aos testes de decomposição SOS, ou devido à limitação fundamental inerente a implementações com números em ponto-flutuante (Löfberg, 2009b).

Nesta seção, serão apresentados resultados de testes de decomposições SOS visando avaliar a qualidade das decomposições encontradas numericamente usando diferentes *solvers* SDP. Os *solvers* SDP são programados em geral com algoritmos diferentes, de modo que uma análise comparativa é interessante. A qualidade da decomposição SOS pode ser avaliada de diferentes maneiras. Neste trabalho, será usado o conceito de resíduo utilizado pelo *toolbox* Yalmip/SOS, dado

na Definição (6).

Definição 6. *Seja $p(x)$ um polinômio para o qual se deseja encontrar uma decomposição SOS. Seja $z_p(x)^T Q_p z_p(x)$ um polinômio SOS encontrado numericamente resolvendo o problema SDP associado ao problema de decomposição SOS de $p(x)$. O resíduo r_p é definido como o maior coeficiente do polinômio $p(x) - z_p(x)^T Q_p z_p(x)$, em valor absoluto.*

Serão comparados os resíduos obtidos resolvendo-se certos problemas de decomposição SOS implementados através do Yalmip/SOS. O Yalmip/SOS possui suporte para a escolha de vários *solvers* SDP, dentre os quais cinco foram escolhidos para os testes. Os cinco *solvers* SDP que serão testados são apresentados na Tabela 4.6.

Tabela 4.6: Solvers SDP testados na Seção 4.3.4.

Solver SDP	Opções
SeDuMi (Sturm, 1999)	eps$= 10^{-12}$ bigeps $= 10^{-6}$ sdp $= 0$ stepdif $= 1$ free $= 1$
SDPT3 (Toh et al., 1999)	gaptol$= 10^{-12}$ inftol $= 10^{-12}$ steptol $= 10^{-12}$
CSDP (Borchers, 1999)	axtol $= 10^{-12}$ atytol $= 10^{-12}$ objtol $= 10^{-12}$ minstepp $= 10^{-12}$ minstepd $= 10^{-12}$
MOSEK (MOSEK-ApS, 2015)	INTPNT_CO_TOL_DFEAS $= 10^{-12}$ INTPNT_CO_TOL_NEAR_REL $= 10^{6}$ INTPNT_CO_TOL_REL GAP $= 10^{-12}$
SDPA (Fujisawa et al., 2008)	epsilonStar $= 10^{-12}$ epsilonDash $= 10^{-12}$

Na Tabela 4.6 também são informadas as opções relevantes que foram selecionadas para cada

solver. Apesar do fato que os *solvers* podem apresentar tolerâncias específicas a seus algoritmos, tentou-se manter uma tolerância geral de 10^{-12}. As opções não informadas foram as opções padrão de cada *solver*.

Os parâmetros do Yalmip/SOS que serão utilizados nesta seção foram fixos da seguinte maneira: exploração de simetrias[6] ativada, aproveitamento de esparsidade ativado, e normalização dos polinômios ativada. Além desses parâmetros que foram fixos, decidiu-se por testar a funcionalidade "model" do Yalmip/SOS, que permite a escolha do modelo de problema SDP associado ao problema SOS. Nos testes das seções anteriores, foi utilizado o modelo primal, que coloca o problema SDP na forma da Equação (4.6). Porém, o Yalmip/SOS permite que a formulação do problema SDP seja feita na forma dual. A ideia é verificar a influência da escolha de um modelo ou outro no resíduo das decomposições SOS.

Os polinômios utilizados para realizar os testes de qualidade de decomposição SOS são os polinômios $p_i(x), i = 1, 2, 3, 4, 5, 6, 7$ da Equação (4.35). Esses polinômios SOS surgiram do estudo do problema de otimização abordado na Seção 4.2, e constituem um conjunto de resultados interessante para a elaboração de testes numéricos.

O número de monômios que o Yalmip/SOS mantém na base monomial associada ao teste de decomposição SOS de cada um dos polinômios da Equação (4.35), após a redução feita pelo algoritmo de Politopos de Newton, é mostrado na Tabela 4.7. O tamanho da base monomial mantida no teste de decomposição SOS, como visto na Seção 4.1, está diretamente ligado ao tamanho do problema SDP associado.

Tabela 4.7: Dimensão das bases monomiais $z(x)$ das decomposições SOS de $p_i(x)$ dados em (4.35).

	$p_1(x)$	$p_2(x)$	$p_3(x)$	$p_4(x)$	$p_5(x)$	$p_6(x)$	$p_7(x)$
$\dim z(x)$	68	800	115	1011	909	708	1119

Além da influência de cada *solver* na qualidade da decomposição SOS, será também testada a influência da existência ou não de funções objetivo no teste de decomposição SOS. Três tipos de problema foram levados em consideração.

[6] O flag utilizado foi `congruence = 2`.

1. Teste de Factibilidade de Decomposição SOS:

$$\begin{aligned} \text{Encontre} \quad & Q_i \\ \text{sujeito a} \quad & p_i(x) = z_i(x)^T Q_i z_i(x) \Leftrightarrow p_i(x) \text{ é SOS} \end{aligned}$$

2. Minimização do Traço da Matriz Q

$$\begin{aligned} \min \quad & tr(Q_i) \\ \text{sujeito a} \quad & p_i(x) = z_i(x)^T Q_i z_i(x) \Leftrightarrow p_i(x) \text{ é SOS} \end{aligned}$$

3. Maximização do Limitante Inferior

$$\begin{aligned} \max \quad & \rho \\ \text{sujeito a} \quad & p_i(x) - \rho = z_i(x)^T Q_i z_i(x) \Leftrightarrow p_i(x) - \rho \text{ é SOS} \end{aligned}$$

As Tabelas 4.8-4.14 mostram os resíduos obtidos pelo Yalmip/SOS após a resolução de cada um dos três problemas acima, para cada um dos $p_i(x)$ da Equação (4.35), utilizando cada um dos *solvers* da Tabela 4.6. Foi também variado entre o modelo primal (1) e dual (2) do Yalmip/SOS. Os resultados indicados por "-" não puderam ser obtidos por falhas variadas na resolução.

A primeira observação relevante a se fazer é que, analisando a Tabela 4.7 e os resíduos das decomposições de cada um dos polinômios $p_i(x)$, é possível concluir que quanto maior a base monomial utilizada no teste SOS, maior é a ordem de grandeza do resíduo. Esse fato mostra uma característica importante das decomposições SOS numéricas: a qualidade das decomposições é melhor para problemas menores. Por essa razão, é crucial a utilização das técnicas de Politopo de Newton na redução da base monomial utilizada no problema.

Em segundo lugar, é interessante analisar pelas Tabelas 4.8-4.14 que a utilização do modelo dual na elaboração do problema SDP, que é uma funcionalidade exclusiva do Yalmip/SOS, faz com que o resíduo obtido seja em geral menor que a utilização do modelo primal. A desvantagem da utilização do modelo dual é uma menor eficiência computacional (Löfberg, 2009a). Essa menor eficiência foi constatada pelo maior consumo de memória dos solvers durante a execução com o modelo dual do Yalmip/SOS. Para problemas utilizando polinômios maiores que os $p_i(x)$, com muito mais monômios na base monomial, isso pode se tornar um problema grande.

Em relação à comparação entre os *solvers*, a diferença mais relevante observada nas Tabelas 4.8-4.14 é o melhor desempenho do CSDP para o teste 1, utilizando o modelo 1. O CSDP foi

capaz de resolver o problema SDP gerando um resíduo da decomposição SOS de ordem duas a três vezes menor que os resíduos obtidos utilizando os outros *solvers*. Esse fato foi determinante na escolha do CSDP para resolver o problema de decomposição SOS da Seção 4.2.3. Infelizmente, não foi possível obter decomposições utilizando o CSDP na resolução dos outros problemas. As dificuldades foram duas. Primeiro, quando o modelo selecionado era o modelo dual (2), o CSDP não retornava uma solução útil do problema SDP, e o Yalmip/SOS não era capaz de terminar a resolução do problema. Segundo, quando os testes eram o teste 2 ou o teste 3, que envolvem funções objetivo, o programa Matlab era interrompido devido a um erro de execução no código do CSDP.

Em relação aos outros *solvers* SDP, o SeDuMi e o SDPA geraram resíduos de decomposição SOS similares. Para esses dois *solvers*, é difícil de analisar a influência da inclusão de uma função objetivo (testes 2 e 3) na qualidade do resíduo, em relação ao teste de factibilidade pura de decomposição SOS (teste 1). Por exemplo, na Tabela 4.13, o menor resíduo obtido pelo SDPA foi na resolução do teste 1, sem função objetivo. Já na Tabela 4.11 o SDPA apresentou um resíduo pior na resolução do teste 1, de modo que a inclusão de funções objetivo foi benéfica. O SeDuMi apresenta um comportamento semelhante.

Já o SDPT3 em geral gera resíduos melhores quando é incluída uma função objetivo no problema. Essa diferença pode ser muito grande, como constatado na Tabela 4.9. O MOSEK apresenta resultados similares ao SDPA e ao SeDuMi, porém não é tão consistente na resolução dos problemas, não conseguindo retornar uma solução em alguns casos.

A conclusão destes testes é que o CSDP é o *solver* que gera uma decomposição SOS de maior qualidade quando o modelo primal (1) do Yalmip/SOS é selecionado. O uso desse modelo tem a vantagem de ser mais computacionalmente eficiente. Quando o modelo (2) é escolhido, o CSDP falha na obtenção de uma solução para o problema SDP, de modo que a escolha alternativa do SeDuMi ou do SDPA pode ser feita, nesse caso, para que melhores resíduos sejam obtidos.

Por fim, é importante mencionar que a existência de qualquer resíduo na decomposição SOS encontrada numericamente para um determinado polinômio faz com que a decomposição seja na realidade uma perturbação muito pequena do polinômio. A garantia de que o polinômio original é de fato SOS não é teórica, nesse caso, mas a existência de um polinômio SOS muitíssimo parecido com o polinômio original é uma indicação numérica forte de que o este seja de fato não negativo

(Löfberg, 2009b).

Tabela 4.8: Resíduos da decomposição SOS de $p_1(x)$ da Equação (4.35).

Teste	Model	SeDuMi	SDPT3	CSDP	MOSEK	SDPA
1	1	3.8×10^{-13}	1.1×10^{-12}	4.5×10^{-12}	7.8×10^{0}	6.5×10^{-12}
1	2	5.3×10^{-15}	2.1×10^{-14}	-	3.5×10^{-15}	1.4×10^{-14}
2	1	1.0×10^{-12}	2.7×10^{-11}	-	$2.\times10^{-7}$	7.9×10^{-12}
2	2	3.4×10^{-13}	1.1×10^{-12}	-	2.1×10^{-14}	1.7×10^{-14}
3	1	2.2×10^{-12}	1.2×10^{-11}	-	1.0×10^{-7}	-
3	2	3.1×10^{-14}	5.3×10^{-14}	-	2.1×10^{-14}	4.2×10^{-14}

Tabela 4.9: Resíduos da decomposição SOS de $p_2(x)$ da Equação (4.35).

Teste	Model	SeDuMi	SDPT3	CSDP	MOSEK	SDPA
1	1	6.1×10^{-5}	2.1×10^{-2}	3.8×10^{-8}	3.5×10^{-4}	3.4×10^{-4}
1	2	3.6×10^{-12}	6.3×10^{-12}	-	-	7.1×10^{-12}
2	1	1.0×10^{-4}	3.2×10^{-4}	-	-	6.3×10^{-4}
2	2	3.4×10^{-12}	5.4×10^{-12}	-	-	3.6×10^{-12}
3	1	6.2×10^{-5}	8.8×10^{-7}	-	9.9×10^{-5}	2.7×10^{-4}
3	2	2.5×10^{-12}	7.0×10^{-12}	-	-	6.5×10^{-12}

Tabela 4.10: Resíduos da decomposição SOS de $p_3(x)$ da Equação (4.35).

Teste	Model	SeDuMi	SDPT3	CSDP	MOSEK	SDPA
1	1	7.2×10^{-7}	1.6×10^{-9}	1.0×10^{-10}	1.0×10^{-6}	6.4×10^{-6}
1	2	2.1×10^{-14}	2.8×10^{-14}	-	2.1×10^{-14}	1.4×10^{-14}
2	1	9.9×10^{-6}	9.8×10^{-6}	-	1.7×10^{-6}	7.5×10^{-7}
2	2	1.4×10^{-14}	2.1×10^{-14}	-	-	1.4×10^{-14}
3	1	5.0×10^{-8}	4.0×10^{-11}	-	1.2×10^{-6}	2.9×10^{-5}
3	2	1.4×10^{-14}	1.4×10^{-14}	-	-	1.4×10^{-14}

Tabela 4.11: Resíduos da decomposição SOS de $p_4(x)$ da Equação (4.35).

Teste	Model	SeDuMi	SDPT3	CSDP	MOSEK	SDPA
1	1	7.6×10^{-5}	2.9×10^{-3}	3.2×10^{-8}	8.8×10^{-5}	1.6×10^{-4}
1	2	5.4×10^{-12}	6.8×10^{-12}	-	-	6.8×10^{-12}
2	1	5.2×10^{-5}	3.9×10^{-4}	-	2.6×10^{-4}	4.5×10^{-5}
2	2	5.9×10^{-12}	5.4×10^{-12}	-	-	5.9×10^{-12}
3	1	2.5×10^{-5}	2.5×10^{-6}	-	1.0×10^{-4}	6.5×10^{-5}
3	2	3.6×10^{-12}	3.1×10^{-12}	-	-	3.9×10^{-12}

Tabela 4.12: Resíduos da decomposição SOS de $p_5(x)$ da Equação (4.35).

Teste	Model	SeDuMi	SDPT3	CSDP	MOSEK	SDPA
1	1	1.5×10^{-5}	1.4×10^{-3}	7.1×10^{-8}	1.5×10^{-4}	1.1×10^{-4}
1	2	7.0×10^{-12}	6.1×10^{-12}	-	6.5×10^{-12}	7.2×10^{-12}
2	1	3.4×10^{-4}	5.9×10^{-4}	-	9.5×10^{-4}	5.3×10^{-4}
2	2	7.5×10^{-12}	6.8×10^{-12}	-	-	6.1×10^{-12}
3	1	3.4×10^{-5}	8.2×10^{-7}	-	8.1×10^{-5}	-
3	2	3.6×10^{-12}	5.5×10^{-12}	-	-	3.6×10^{-12}

Tabela 4.13: Resíduos da decomposição SOS de $p_6(x)$ da Equação (4.35).

Teste	Model	SeDuMi	SDPT3	CSDP	MOSEK	SDPA
1	1	4.7×10^{-6}	2.2×10^{-4}	6.2×10^{-8}	2.1×10^{-5}	1.1×10^{-5}
1	2	9.0×10^{-13}	1.0×10^{-12}	-	6.8×10^{-13}	9.0×10^{-13}
2	1	1.0×10^{-5}	7.7×10^{-5}	-	4.4×10^{-5}	1.3×10^{-4}
2	2	1.7×10^{-12}	6.8×10^{-13}	-	-	1.3×10^{-12}
3	1	1.7×10^{-5}	4.2×10^{-7}	-	1.6×10^{-5}	1.8×10^{-5}
3	2	1.3×10^{-12}	8.8×10^{-13}	-	-	1.3×10^{-12}

Tabela 4.14: Resíduos da decomposição SOS de $p_7(x)$ da Equação (4.35).

Teste	Model	SeDuMi	SDPT3	CSDP	MOSEK	SDPA
1	1	3.1×10^{-5}	6.1×10^{-4}	5.1×10^{-8}	4.7×10^{-5}	3.0×10^{-5}
1	2	6.1×10^{-12}	7.7×10^{-12}	-	-	6.8×10^{-12}
2	1	4.4×10^{-5}	1.9×10^{-4}	-	8.1×10^{-5}	3.9×10^{-4}
2	2	5.6×10^{-12}	5.9×10^{-12}	-	-	5.2×10^{-12}
3	1	5.7×10^{-5}	5.7×10^{-6}	-	7.5×10^{-5}	3.1×10^{-4}
3	2	2.7×10^{-12}	4.0×10^{-12}	-	-	3.1×10^{-12}

Capítulo 5

Conclusão

Este trabalho apresentou um método para a análise não linear da robustez de controladores cinemáticos de robôs móveis não holonômicos em relação à perturbação introduzida pelo deslizamento lateral do robô. Esse método foi apresentado através de um exemplo de aplicação para analisar a robustez de um controlador adaptativo capaz de compensar um deslizamento longitudinal constante. As técnicas de controle não linear utilizadas para o projeto desse controlador também foram estudadas neste trabalho.

Foi mostrado que apesar do fato que o projeto do controlador não levou em conta o deslizamento lateral e a variação temporal do deslizamento longitudinal, quando esses fatores estão presentes, o controlador é capaz de garantir que o erro de postura do robô seja uniformemente finalmente limitado, desde que a velocidade máxima de referência e os valores máximos do deslizamento lateral e da taxa de variação do deslizamento longitudinal sejam suficientemente pequenos.

Na prática, como apresentado através de uma simulação numérica, isso significa que o robô, sob deslizamento lateral e deslizamentos longitudinais variantes no tempo, ainda é capaz de rastrear a trajetória, de um certo modo. Esse rastreamento, quando o deslizamento lateral se faz presente, não é perfeito. Porém, como mencionado na revisão da literatura, o deslizamento lateral é uma perturbação muito difícil de ser combatida por um robô móvel de tração diferencial, já que essa perturbação não entra no sistema pelo mesmo canal que as entradas de controle. Portanto, garantir que o robô móvel, quando sob efeito do deslizamento lateral, se mantenha numa vizinhança próxima da trajetória a ser rastreada, já é um resultado satisfatório.

É importante mencionar que o mesmo método apresentado no exemplo de aplicação deste trabalho pode ser utilizado para analisar a robustez de outros tipos de controladores cinemáticos em relação ao deslizamento lateral.

A análise de estabilidade apresentada evidenciou a importância do estabelecimento de limitantes para os parâmetros variantes no tempo de um sistema de controle não linear. Os limitantes impostos sobre as velocidades de referência e sobre as perturbações relacionadas aos deslizamentos foram particularmente importantes, já que as condições que garantem a robustez do controlador cinemático perante o deslizamento lateral dependem fortemente desses limitantes. Foi visto que essas condições também dependem de certos coeficientes de funções de Lyapunov específicas para os sistemas analisados. Neste trabalho, foi utilizado um Teorema Converso de Lyapunov para garantir a existência de tais funções, e portanto esses coeficientes não foram apresentados numericamente.

Em aplicações em que é desejável conhecer um valor numérico máximo para os limitantes das velocidades de referência e dos deslizamentos, é necessário encontrar as funções de Lyapunov cujas existências são garantidas pelo Teorema Converso. A obtenção numérica dessas funções não fazia parte do escopo deste projeto. Porém, vale a pena mencionar que existem técnicas numéricas surgindo recentemente que ajudam na procura por funções de Lyapunov polinomiais para sistemas não lineares, como as exigidas pelos teoremas de análise de sistemas não lineares perturbados. Uma dessas técnicas utiliza otimização polinomial baseada em decomposições SOS. Neste trabalho, as técnicas SOS foram utilizadas com outro objetivo: a minimização restrita de um polinômio em muitas variáveis e de grau extremamente alto.

Foi mostrado que a resolução de problemas de otimização polinomial através de técnicas de relaxações, como as decomposições SOS, encontra limitações de ordem computacional, especialmente no que concerne a quantidade de memória disponível no sistema. Essa limitação está relacionada a basicamente três características dos polinômios envolvidos nos problemas de otimização: seu grau, seu número de variáveis, e sua esparsidade.

Foram explorados cinco *toolboxes* de otimização polinomial disponibilizados pela comunidade científica, e constatou-se que suas funcionalidades de simplificação dos problemas SDP influenciam muito em suas capacidades de resolver problemas de otimização com quantidades razoáveis de memória. Em especial, constatou-se particular eficiência do módulo SOS do *toolbox* Yalmip. Uma funcionalidade exclusiva desse *toolbox*, que é a exploração de simetrias, foi crucial

para a obtenção de uma solução para o problema de otimização que deu origem à investigação de técnicas SOS apresentada neste trabalho.

As curvas de extrapolação da memória necessária para a resolução de problemas de otimização polinomial sem esparsidade, obtidas a partir dos testes desenvolvidos na Seção (4.3), podem ser utilizadas na prática por qualquer um que esteja interessado em saber aproximadamente o quanto de memória seria necessário para resolver problemas similarmente pouco esparsos. Foi visto que um caso em que a esparsidade do problema é muito prejudicada ocorre quando utiliza-se o Positivstellensatz na resolução de problemas restritos.

Vários dos polinômios apresentados na Seção (4.2) constituem um conjunto interessante para a realização de testes de decomposição SOS. Foi evidenciado que a qualidade das decomposições SOS depende do tamanho da base monomial utilizada no problema. Isso evidencia a importância das técnicas de redução da base monomial para a procura de decomposições SOS, como os Politopos de Newton. Não apenas essas técnicas ajudam a reduzir a quantidade de memória necessária para a procura numérica de decomposições SOS, mas elas também ajudam a garantir que a qualidade das decomposições encontradas seja alta.

Referências Bibliográficas

Borchers, B. (1999). CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 11(1-4):613–623.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Burghi, T. B., Iossaqui, J. G., and Camino, J. F. (2015). Stability analysis of perturbed nonlinear systems applied to the tracking control of a wheeled robot under lateral slip. In *Proceedings of the XVII International Symposium on Dynamic Problems of Mechanics - DINAME*, pages 1–10, Natal, Brazil.

Campion, G., d'Andrea Novel, B., and Bastin, G. (1991). Modelling and state feedback control of nonholonomic mechanical systems. In *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1184–1189, Brighton, England.

Chen, W., Mills, J. K., Chu, J., and Sun, D. (2001). Brief communication: Uniform ultimate boundedness of a fuzzy logic controlled industrial robot. *Journal of Robotic Systems*, 18(9):553–561.

Desoer, C. A. (1969). Slowly varying system $\dot{x} = A(t)x$. *IEEE Transactions on Automatic Control*, 14(6):780–781.

Fierro, R. and Lewis, F. L. (1995). Control of a nonholonomic mobile robot: backstepping kinematics into dynamics. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 3805–3810.

Fujisawa, K., Fukuda, M., Kobayashi, K., Kojima, M., Nakata, K., Nakata, M., and Yamashita, M. (2008). SDPA (SemiDefinite Programming Algorithm) user's manual – version 7.0.5. Technical report, Tokyo Institute of Technology.

Gantmacher, F. R. (1959). *The Theory of Matrices*. AMS Chelsea Publishing.

González, R., , Fiacchini, M., Alamo, T., Guzman, J. L., and Rodriguez, F. (2010). Adaptive control for a mobile robot under slip conditions using an LMI-based approach. *European Journal of Control*, 16(2):144–155.

González, R., Guzman, J. L., Rodriguez, F., and Berenguel, M. (2009). Localization and control of tracked mobile robots under slip conditions. In *Proceedings of the IEEE International Conference on Mechatronics*, pages 1–6, Malaga, Spain.

Greenwood, D. T. (2006). *Advanced Dynamics*. Cambridge University Press.

Henrion, D., Lasserre, J.-B., and Löfberg, J. (2009). Gloptipoly 3: moments, optimization and semidefinite programming. *Optimization Methods and Software*, 24(4-5):761–779.

Iossaqui, J. G. (2013). *Técnicas não lineares de controle e filtragem aplicadas ao problema de rastreamento de trajetórias de robôs móveis com deslizamento longitudinal das rodas*. PhD thesis, Universidade Estadual de Campinas.

Iossaqui, J. G., Camino, J. F., and Zampieri, D. E. (2010). Adaptive tracking control of tracked mobile robots with unknown slip parameter. In *Anais do XVIII Congresso Brasileiro de Automática*, pages 1846–1851, Bonito, Brasil.

Iossaqui, J. G., Camino, J. F., and Zampieri, D. E. (2011). A nonlinear control design for tracked robots with longitudinal slip. In *Proceedings of the 18th IFAC World Congress*, pages 5932–5937, Milano, Italy.

Iossaqui, J. G., Camino, J. F., and Zampieri, D. E. (2013). Wheeled robot slip compensation for trajectory tracking control problem with time-varying reference input. In *Proceedings of the 9th Workshop on Robot Motion and Control*, pages 167–173, Wasowo, Poland.

Jiang, Z.-P. and Nijmeijer, H. (1997). Tracking control of mobile robots: A case study in backstepping. *Automatica*, 33(7):1393–1399.

Jung, M.-J., Shim, H.-S., Kim, H.-S., and Kim, J.-H. (1999). The miniature omni-directional mobile robot omnikity-I (OK-I). In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2686–2691.

Kalmár-Nagy, T., D'Andrea, R., and Ganguly, P. (2004). Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46(1):47–64.

Kanayama, Y., Kimura, Y., Miyazaki, F., and Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 384–389, Cincinnati, USA.

Khalil, H. K. (2002). *Nonlinear Systems*. Prentice-Hall, 3rd edition.

Kim, D.-H. and Oh, J.-H. (1998). Globally asymptotically stable tracking control of mobile robots. In *Proceedings of the IEEE International Conference on Control Applications*, pages 1297–1301, Trieste, Italy.

Kim, M.-S., Shin, J.-H., Hong, S.-G., and Lee, J.-J. (2003). Designing a robust adaptive dynamic controller for nonholonomic mobile robots under modeling uncertainty and disturbances. *Mechatronics*, 13(5):507–519.

Koo, K.-M. and Kim, J.-H. (1994). Robust control of robot manipulators with parametric uncertainty. *IEEE Transactions on Automatic Control*, 39(6):1230–1233.

Lasserre, J.-B. (2001). Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817.

Laurent, M. (2009). Sums of squares, moment matrices and optimization over polynomials. In Putinar, M. and Sullivant, S., editors, *Emerging Applications of Algebraic Geometry*, volume 149 of *The IMA Volumes in Mathematics and its Applications*, pages 157–270. Springer New York.

Liu, Y., Zhu, J. J., Williams, R. L., and Wu, J. (2008). Omni-directional mobile robot controller based on trajectory linearization. *Robotics and Autonomous Systems*, 56(5):461–479.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in Matlab. In *Proceedings of the IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, Taipei, Taiwan.

Löfberg, J. (2009a). Dualize it: software for automatic primal and dual conversions of conic programs. *Optimization Methods and Software*, 24(3):313–325.

Löfberg, J. (2009b). Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions on Automatic Control*, 54(5):1007–1011.

Morin, P. and Samson, C. (2008). Motion control of wheeled mobile robots. In Siciliano, B. and Khatib, O., editors, *Springer Handbook of Robotics*, pages 799–826. Springer Berlin Heidelberg.

MOSEK-ApS (2015). *The MOSEK optimization toolbox for Matlab, version 7.1.* Manual available at `http://docs.mosek.com/7.1/toolbox/index.html`.

Nie, J., Demmel, J., and Gu, M. (2008). Global minimization of rational functions and the nearest GCDs. *Journal of Global Optimization*, 40(4):697–718.

Pacejka, H. (2012). *Tire and Vehicle Dynamics*. Butterworth-Heinemann, 3rd edition.

Papachristodoulou, A., Anderson, J., Valmorbida, G., Prajna, S., Seiler, P., and Parrilo, P. A. (2013). *SOSTOOLS: Sum of squares optimization toolbox for Matlab*. Manual available at `http://arxiv.org/abs/1310.4716`.

Parrilo, P. A. (2003). Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320.

Qu, Z. and Dorsey, J. (1991). Robust tracking control of robots by a linear feedback law. *IEEE Transactions on Automatic Control*, 36(9):1081–1084.

Reznick, B. (1978). Extremal PSD forms with few terms. *Duke Mathematical Journal*, 45(2):363–374.

Rosenbrock, H. H. (1963). On the stability of linear time-dependent control systems. *Journal of Electronics and Control*, 15(1):73–80.

Ryu, J.-C. and Agrawal, S. K. (2011). Differential flatness-based robust control of mobile robots in the presence of slip. *International Journal of Robotics Research*, 30(4):463–475.

Siegwart, R. and Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*. Bradford Company.

Slotine, J.-J. and Li, W. (1991). *Applied Nonlinear Control*. Prentice-Hall.

Spong, M. W. and Fujita, M. (2011). Control in robotics. In Samad, T. and Annaswamy, A., editors, *The Impact of Control Technology*. IEEE Control Systems Society.

Spong, M. W., Hutchinson, S., and Vidyasagar, M. (2006). *Robot Modeling and Control*. John Wiley and Sons.

Sturm, J. F. (1999). Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1–4):625–653.

Sturmfels, B. (1998). Polynomial equations and convex polytopes. *American Mathematical Monthly*, 105:907–922.

Tarokh, M. and McDermott, G. (2005). Kinematics modeling and analyses of articulated rovers. *IEEE Transactions on Robotics*, 21(4):539–553.

Toh, K. C., Todd, M. J., and Tütüncü, R. H. (1999). SDPT3 – a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11(1–4):545–581.

Vandenberghe, L. and Boyd, S. (1996). Semidefinite programming. *SIAM Review*, 38(1):49–95.

Waki, H., Kim, S., Kojima, M., and Muramatsu, M. (2006). Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17(1):218–242.

Waki, H., Kim, S., Kojima, M., Muramatsu, M., and Sugimoto, H. (2008). SparsePOP - A sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Transactions on Mathematical Software*, 35(2):1–13.

Wang, D. and Low, C. B. (2008). Modeling and analysis of skidding and slipping in wheeled mobile robots: Control design perspective. *IEEE Transactions on Robotics*, 24(3):676–687.

Yang, J.-M. and Kim, J.-H. (1999). Sliding mode motion control of nonholonomic mobile robots. *Control Systems, IEEE*, 19(2):15–23.

Zhou, B., Peng, Y., and Han, J. (2007). UKF based estimation and tracking control of nonholonomic mobile robots with slipping. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pages 2058–2063, Sanya, China.

Apêndice A

Teoremas, Lemas e Definições

A.1 Estabilidade de sistemas autônomos

Considere o sistema autônomo

$$\dot{e}(t) = f(e(t)) \tag{A.1}$$

onde $f(e(t)) : D \to \mathbb{R}^n$ é um mapeamento localmente Lipschitz de um domínio $D \subset R^n$ para R^n.

Definição 7 (Khalil (2002), Definição 4.1)**.** *O ponto de equilíbrio* $e(t) = 0$ *do sistema* (A.1) *é*

- *estável se, para cada* $\epsilon > 0$*, existe* $\delta = \delta(\epsilon) > 0$ *tal que*

$$\|e(0)\| < \delta \implies \|e(t)\| < \epsilon, \quad \forall\, t \geq 0$$

- *assintoticamente estável se* $e(t) = 0$ *é estável e se* δ *pode ser escolhido tal que*

$$\|e(0)\| < \delta \implies \lim_{t \to \infty} e(t) = 0$$

Teorema 3 (Khalil (2002), Teorema 4.1)**.** *Seja* $e(t) = 0$ *um ponto de equilíbrio para* (A.1) *e* $D \subset R^n$ *um domínio contendo* $e(t) = 0$*. Seja* $V(e(t)) : D \to R$ *uma função continuamente diferenciável tal que*

$$V(0) = 0 \quad e \quad V(e(t)) > 0 \quad em \quad D - \{0\} \tag{A.2}$$

$$\dot{V}(e(t)) \leq 0 \quad em \quad D$$

Então, $e(t) = 0$ *é estável. Além disso, se*

$$\dot{V}(e(t)) < 0 \quad in \quad D - \{0\} \tag{A.3}$$

então $e(t) = 0$ *é assintoticamente estável.*

A.2 Estabilidade de sistemas não autônomos

Considere o sistema não autônomo

$$\dot{e}_a = f(t, e_a(t)) \tag{A.4}$$

em que $f(t, e_a(t)) : [0, \infty) \times D \to \mathbb{R}^n$ é contínua por partes em t e localmente Lipschitz em e_a em $[0, \infty) \times D$, e $D \subset \mathbb{R}^n$ é um domínio contendo a origem $e_a(t) = 0$.

Teorema 4 (Khalil (2002), Teorema 8.4)**.** *Seja* $D \subset \mathbb{R}^n$ *um domínio contendo* $e_a(t) = 0$ *e suponha que* $f(t, e_a(t))$ *é contínua por partes em* t *e localmente Lipschitz em* e_a*, uniformemente em* t*, em* $[0, \infty) \times D$*. Além disso, suponha que* $f(t, 0)$ *é uniformemente limitada para todo* $t \geq 0$*. Seja* $V(t, e_a(t)) : [0, \infty) \times D \to R$ *uma função continuamente diferenciável tal que*

$$W_1(e_a(t)) \leq V(t, e_a(t)) \leq W_2(e_a(t)) \tag{A.5}$$

$$\dot{V}(t, e_a(t)) = \frac{\partial V(t, e_a(t))}{\partial t} + \frac{\partial V(t, e_a(t))}{\partial e_a(t)} f(t, e_a(t)) \leq -W(e_a(t)) \tag{A.6}$$

$\forall\ t \geq 0$, $\forall\ e_a(t) \in D$*, onde* $W_1(e_a(t))$ *e* $W_2(e_a(t))$ *são funções contínuas positivas definidas e* $W(e_a(t))$ *é uma função contínua semidefinida positiva em* D*. Escolha* $d > 0$ *tal que* $B_d \subset D$ *e seja* $c < \min_{\|e_a\|=d} W_1(e_a(t))$*. Então, todas as soluções de* $\dot{e}_a(t) = f(t, e_a(t))$ *com* $e_a(t_0) \in \{e_a \in B_d \mid W_2(e_a(t)) \leq c\}$ *são limitadas e satisfazem*

$$W(e_a(t)) \to 0 \quad com \quad t \to \infty$$

Teorema 5 (Khalil (2002), Teorema 4.15)**.** *Seja* $e_a(t) = 0$ *um ponto de equilíbrio do sistema não linear*

$$\dot{e}_a(t) = f(t, e_a(t))$$

em que $f : [0, \infty) \times D \to \mathbb{R}^n$ *é continuamente diferenciável,* $D = \{e_a(t) \in \mathbb{R}^n \mid \|e_a(t)\|_2 < d\}$*, e a matriz Jacobiana* $[\partial f(t, e_a(t))/\partial e_a(t)]$ *é limitada e Lipschitz em* D*, uniformemente in* t*. Seja*

$$A(t) = \left.\frac{\partial f(t, e_a(t))}{\partial e_a(t)}\right|_{e_a(t)=0}$$

Então, $e_a(t) = 0$ é um ponto de equilíbrio exponencialmente estável do sistema não linear se e somente se ele é um ponto de equilíbrio exponencialmente estável do sistema linear

$$\dot{e}_a(t) = A(t)e_a(t) \tag{A.7}$$

Teorema 6 (Rosenbrock (1963), Teorema 1)**.** *Seja $\dot{e}_a(t) = A(t)e_a(t)$, em que todo elemento $a_{ij}(t)$ de $A(t)$ é diferenciável e satisfaz $|a_{ij}(t)| \leq \rho$ para algum $\rho > 0$, e todo autovalor de $A(t)$ satisfaz*

$$Re[\lambda(A(t))] \leq -\epsilon < 0 \tag{A.8}$$

Então, existe um $\delta > 0$ (independente de t) tal que se todo $|\dot{a}_{ij}| \leq \delta$, o ponto de equilíbrio $e_a(t) = 0$ é assintoticamente estável. Em Desoer (1969), essa conclusão foi estendida e se mostrou que o ponto de equilíbrio $e_a(t) = 0$ é exponencialmente estável.

Teorema 7 (Khalil (2002), Teorema 4.14)**.** *Seja $e_a(t) = 0$ um ponto de equilíbrio para o sistema não linear*

$$\dot{e}_a = f(t, e_a(t))$$

em que $f(t, e_a(t)) : [0, \infty) \times D \rightarrow \mathbb{R}^n$ é continuamente diferenciável,

$$D = \{e_a(t) \in \mathbb{R}^n \mid \|e_a(t)\| < d\}$$

e a matriz Jacobiana $\partial f / \partial e_a$ é limitada em D, uniformemente em t. Sejam k, λ, e d_0 constantes positivas tais que $d_0 < d/k$. Seja $D_0 = \{e_a \in \mathbb{R}^n \mid \|e_a(t)\| < d_0\}$. Assuma que as trajetórias do sistema satisfaçam

$$\|e_a(t)\| \leq k\|e_a(t_0)\| \, \mathrm{e}^{-\lambda(t-t_0)}, \quad \forall \, e_a(t_0) \in D_0, \quad \forall \, t \geq t_0 \geq 0 \tag{A.9}$$

Então, existe uma função $V(t, e_a(t)) : [0, \infty) \times D_0 \rightarrow R$ que satisfaz as seguintes desigualdades

$$c_1\|e_a(t)\|^2 \leq V(t, e_a(t)) \leq c_2\|e_a(t)\|^2$$

$$\frac{\partial V(t, e_a(t))}{\partial t} + \frac{\partial V(t, e_a(t))}{\partial e_a(t)} f(t, e_a(t)) \leq -c_3\|e_a(t)\|^2$$

$$\left\| \frac{\partial V(t, e_a(t))}{\partial e_a(t)} \right\| \leq c_4\|e_a(t)\|$$

com constantes positivas c_1, c_2, c_3, e c_4.

A.3 Estabilidade de sistemas perturbados

Considere o sistema

$$\dot{e}_a(t) = f(t, e_a(t)) + g(t, e_a(t)) \tag{A.10}$$

onde $f(t, e_a(t)) : [0, \infty) \times D \rightarrow \mathbb{R}^n$ e $g(t, e_a(t)) : [0, \infty) \times D \rightarrow \mathbb{R}^n$ são contínuas por partes em t e localmente Lipschitz em e_a em $[0, \infty) \times D$, e $D \subset \mathbb{R}^n$ é um domínio que contém a origem $e_a(t) = 0$. O termo $g(t, e_a(t))$ é considerado uma perturbação do sistema nominal da equação (A.4).

Considere uma função de Lyapunov $V(t, e_a(t))$ satisfazendo

$$c_1\|e_a(t)\|^2 \leq V(t, e_a(t)) \leq c_2\|e_a(t)\|^2 \tag{A.11}$$

$$\frac{\partial V(t, e_a(t))}{\partial t} + \frac{\partial V(t, e_a(t))}{\partial e_a(t)} f(t, e_a(t)) \leq -c_3\|e_a(t)\|^2 \tag{A.12}$$

$$\left\| \frac{\partial V(t, e_a(t))}{\partial e_a(t)} \right\| \leq c_4\|e_a(t)\| \tag{A.13}$$

para todo $(t, e_a(t)) \in [0, \infty) \times D$ com constantes positivas c_1,c_2, c_3, e c_4.

Lema 2 (Khalil (2002), Lema 9.1)**.** *Seja $e_a(t) = 0$ é um ponto de equilíbrio exponencialmente estável do sistema nominal* (A.4)*. Seja $V(t, e_a(t))$ uma função de Lyapunov do sistema nominal que satisfaça de* (A.11) *até* (A.13) *em $[0, \infty) \times D$. Suponha que o termo de perturbação $g(t, e_a(t))$ satisfaça*

$$\|g(t, e_a(t))\| \leq \gamma\|e_a(t)\|, \quad \forall\, t \geq 0,\ \forall\, e_a(t) \in D \tag{A.14}$$

e

$$\gamma < \frac{c_3}{c_4} \tag{A.15}$$

Então, a origem é um ponto de equilíbrio exponencialmente estável do sistema perturbado (A.10)*.*

Lema 3 (Khalil (2002), Lema 9.2)**.** *Seja $e_a(t) = 0$ um ponto de equilíbrio exponencialmente estável do sistema nominal* (A.4)*. Seja $V(t, e_a(t))$ uma função de Lyapunov do sistema nominal que satisfaça* (A.11) *até* (A.13) *em $[0, \infty) \times D$, onde $D = \{e_a(t) \in \mathbb{R}^n \mid \|e_a(t)\| < d\}$. Suponha que o termo de perturbação $g(t, e_a(t))$ satisfaça*

$$\|g(t, e_a(t))\| \leq \delta_f < \frac{c_3}{c_4}\sqrt{\frac{c_1}{c_2}}\theta d \tag{A.16}$$

para todo $t \geq 0$, todo $e_a(t) \in D$, com alguma constante positiva $\theta < 1$. Então, para todo $\|e_a(t_0)\| < \sqrt{c_1/c_2}d$, a solução $e_a(t)$ do sistema perturbado (A.10) *satisfaz*

$$\|e_a(t)\| \leq \alpha \, \mathrm{e}^{-\gamma(t-t_0)} \, \|e_a(t_0)\|, \quad \forall \, t_0 \leq t < t_0 + T$$

e

$$\|e_a(t)\| \leq \ell_f, \quad \forall \, t \geq t_0 + T$$

para algum T finito, onde

$$\alpha = \sqrt{\frac{c_2}{c_1}}, \quad \gamma = \frac{(1-\theta)c_3}{2c_2}, \quad \ell_f = \frac{c_4}{c_3}\sqrt{\frac{c_2}{c_1}}\frac{\delta_f}{\theta}$$

A.4 Critério de estabilidade de Liénard e Chipart

Teorema 8 (Gantmacher (1959), Teorema 11)**.** *Condições necessárias e suficientes para que todas as raízes do polinômio real*

$$f(x) = a_0 x^n + a_1 x^{n-1} + ... + a_n \quad (a_0 > 0)$$

tenham partes reais negativas podem ser dadas por qualquer uma das quatro seguintes formas:

1. $a_n > 0, a_{n-2} > 0, \ldots$; $\Delta_1 > 0, \Delta_3 > 0, \ldots$,
2. $a_n > 0, a_{n-2} > 0, \ldots$; $\Delta_2 > 0, \Delta_4 > 0, \ldots$,
3. $a_n > 0, a_{n-1} > 0, a_{n-3} > 0, \ldots$; $\Delta_1 > 0, \Delta_3 > 0, \ldots$,
4. $a_n > 0, a_{n-1} > 0, a_{n-3} > 0, \ldots$; $\Delta_2 > 0, \Delta_4 > 0, \ldots$,

com

$$\Delta_i = \det \begin{bmatrix} a_1 & a_3 & a_5 & \cdots & & \\ a_0 & a_2 & a_4 & \cdots & & \\ 0 & a_1 & a_3 & \cdots & & \\ 0 & a_0 & a_2 & a_4 & & \\ & & & & \ddots & \\ & & & & & a_i \end{bmatrix} \quad (a_k = 0 \textit{ para } k > n)$$

Apêndice B

Cálculos Adicionais

B.1 Dinâmica do erro aumentado em malha fechada

Seja a matriz $\Phi(a(t))$ dada por (2.22), e a matriz $\Phi(\hat{a}(t))^{-1}$ dada por (2.40). Então,

$$
\begin{aligned}
\Phi(a)\Phi(\hat{a}(t))^{-1}\eta_c(t) &= \begin{bmatrix} \frac{r}{2a_e(t)} & \frac{r}{2a_d(t)} \\ \frac{-r}{ba_e(t)} & \frac{r}{ba_d(t)} \end{bmatrix} \begin{bmatrix} \frac{\hat{a}_e(t)}{r} & \frac{-b\hat{a}_e(t)}{2r} \\ \frac{\hat{a}_d(t)}{r} & \frac{b\hat{a}_d(t)}{2r} \end{bmatrix} \begin{bmatrix} v_c(t) \\ \omega_c(t) \end{bmatrix} \\
&= \begin{bmatrix} \left(\frac{\hat{a}_d(t)}{2a_d(t)} + \frac{\hat{a}_e(t)}{2a_e(t)}\right) & \left(\frac{b\hat{a}_d(t)}{4a_d(t)} - \frac{b\hat{a}_e(t)}{4a_e(t)}\right) \\ \left(\frac{\hat{a}_d(t)}{ba_d(t)} - \frac{\hat{a}_e(t)}{ba_e(t)}\right) & \left(\frac{\hat{a}_d(t)}{2a_d(t)} + \frac{\hat{a}_e(t)}{2a_e(t)}\right) \end{bmatrix} \begin{bmatrix} v_c(t) \\ \omega_c(t) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\hat{a}_d(t)}{2a_d(t)} & \frac{b\hat{a}_d(t)}{4a_d(t)} \\ \frac{\hat{a}_d(t)}{ba_d(t)} & \frac{\hat{a}_d(t)}{2a_d(t)} \end{bmatrix} \begin{bmatrix} v_c(t) \\ \omega_c(t) \end{bmatrix} + \begin{bmatrix} \frac{\hat{a}_e(t)}{2a_e(t)} & -\frac{b\hat{a}_e(t)}{4a_e(t)} \\ -\frac{\hat{a}_e(t)}{ba_e(t)} & \frac{\hat{a}_e(t)}{2a_e(t)} \end{bmatrix} \begin{bmatrix} v_c(t) \\ \omega_c(t) \end{bmatrix} \\
&= \frac{\hat{a}_d(t)}{a_d(t)} \begin{bmatrix} \frac{1}{2} & \frac{b}{4} \\ \frac{1}{b} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} v_c(t) \\ \omega_c(t) \end{bmatrix} + \frac{\hat{a}_e(t)}{a_e(t)} \begin{bmatrix} \frac{1}{2} & -\frac{b}{4} \\ -\frac{1}{b} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} v_c(t) \\ \omega_c(t) \end{bmatrix} \\
&= \frac{\hat{a}_d(t)}{a_d(t)} \left(v_c(t) + \frac{b}{2}\omega_c(t)\right) \begin{bmatrix} \frac{1}{2} \\ \frac{1}{b} \end{bmatrix} + \frac{\hat{a}_e(t)}{a_e(t)} \left(v_c(t) - \frac{b}{2}\omega_c(t)\right) \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{b} \end{bmatrix}
\end{aligned}
$$

Portanto, usando a Equação (2.43), tem-se que

$$\Phi(a)\Phi(\hat{a}(t))^{-1}\eta_c(t) = \left(1 + \frac{\tilde{a}_d(t)}{a_d(t)}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right)\begin{bmatrix} \frac{1}{2} \\ \frac{1}{b} \end{bmatrix} + \left(1 + \frac{\tilde{a}_e(t)}{a_e(t)}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right)\begin{bmatrix} \frac{1}{2} \\ -\frac{1}{b} \end{bmatrix} \quad \text{(B.1)}$$

Agora, tendo em vista que $\omega(t)$ é a segunda componente de $\eta(t) = \Phi(a)\Phi(\hat{a}(t))^{-1}\eta_c(t)$, é possível fazer o seguinte desenvolvimento:

$$\begin{aligned} -\Omega(\omega(t))e(t) &= -\begin{bmatrix} 0 & -\omega(t) & 0 \\ \omega(t) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} e_1(t) \\ e_2(t) \\ e_3(t) \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} e_1(t) \\ e_2(t) \\ e_3(t) \end{bmatrix}\omega(t) \\ &= \begin{bmatrix} e_2(t) \\ -e_1(t) \\ 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \end{bmatrix}\Phi(a)\Phi(\hat{a}(t))^{-1}\eta_c(t) \\ &= \begin{bmatrix} 0 & e_2(t) \\ 0 & -e_1(t) \\ 0 & 0 \end{bmatrix}\Phi(a)\Phi(\hat{a}(t))^{-1}\eta_c(t) \end{aligned} \quad \text{(B.2)}$$

Substituindo a Equação (B.2), a matriz $S(\cdot)$ dada por (2.7) e a matriz $R(\cdot)$ dada por (2.4) na dinâmica do erro de postura com deslizamento longitudinal e lateral, Equação (2.51), tem-se

$$\begin{aligned} \dot{e}(t) &= -\Omega(\omega(t))e(t) + R(e_3(t))S(0)\eta_{\mathrm{r}}(t) - S(\sigma(t))\Phi(a(t))\Phi(\hat{a}(t))^{-1}\eta_c(t) \\ &= \begin{bmatrix} -1 & e_2(t) \\ -\sigma(t) & -e_1(t) \\ 0 & -1 \end{bmatrix}\Phi(a)\Phi(\hat{a}(t))^{-1}\eta_c(t) + \begin{bmatrix} \cos e_3(t) & -\sin e_3(t) & 0 \\ \sin e_3(t) & \cos e_3(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}\eta_{\mathrm{r}}(t) \end{aligned}$$

Substituindo a Equação (B.1) na equação acima, e lembrando que $\eta_{\mathrm{r}}(t) = (v_{\mathrm{r}}(t), \omega_{\mathrm{r}}(t))^T$, tem-se que

$$\dot{e}(t) = f_e(t, e_a(t)) + g_e(t, e_a(t)) \tag{B.3}$$

em que a função

$$f_e(t, e_a(t)) = \begin{pmatrix} \left(1 + \frac{\tilde{a}_d(t)}{a_d(t)}\right)\left(\frac{e_2(t)}{b} - \frac{1}{2}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right) + v_r(t)\cos e_3(t) - \left(1 + \frac{\tilde{a}_e(t)}{a_e(t)}\right)\left(\frac{e_2(t)}{b} + \frac{1}{2}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right) \\ \left(1 + \frac{\tilde{a}_e(t)}{a_e(t)}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right)\frac{e_1(t)}{b} + v_r(t)\sin e_3(t) - \left(1 + \frac{\tilde{a}_d(t)}{a_d(t)}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right)\frac{e_1(t)}{b} \\ \omega_r(t) - \frac{1}{b}\left(1 + \frac{\tilde{a}_d(t)}{a_d(t)}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right) + \frac{1}{b}\left(1 + \frac{\tilde{a}_e(t)}{a_e(t)}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right) \end{pmatrix}$$

representa as três primeiras linhas do termo nominal $f_a(t, e_a(t))$ dado por (2.54), bem como as três primeiras linhas do lado direito da Equação (2.49), e a função

$$\begin{aligned} g_e(t, e_a(t)) &= \begin{pmatrix} 0 \\ -\sigma(t)\left[\frac{1}{2}\left(1 + \frac{\tilde{a}_e(t)}{a_e(t)}\right)\left(v_c(t) - \frac{b}{2}\omega_c(t)\right) + \frac{1}{2}\left(1 + \frac{\tilde{a}_d(t)}{a_d(t)}\right)\left(v_c(t) + \frac{b}{2}\omega_c(t)\right)\right] \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ -\sigma(t)\left[\frac{1}{4}\frac{\tilde{a}_e(t)}{a_e(t)}\left(2v_c(t) - b\omega_c(t)\right) + \frac{1}{4}\frac{\tilde{a}_d(t)}{a_d(t)}\left(2v_c(t) + b\omega_c(t)\right) + v_c(t)\right] \\ 0 \end{pmatrix} \end{aligned}$$

representa as três primeiras linhas da soma dos termos evanescente e não evanescente, $g(t, e_a(t)) + g_n(t, e_a(t))$, dados por (2.55) e (2.56). Nota-se que como $v_c(t)$ é dado pela lei de controle (2.42), a função $g_e(t, e_a(t))$ acima também pode ser escrita como

$$g_e(t, e_a(t)) = \begin{pmatrix} 0 \\ -\sigma(t)\left[\frac{1}{4}\frac{\tilde{a}_e(t)}{a_e(t)}\left(2v_c(t) - b\omega_c(t)\right) + \frac{1}{4}\frac{\tilde{a}_d(t)}{a_d(t)}\left(2v_c(t) + b\omega_c(t)\right) - k_3 e_3(t)\omega_c(t) + k_1 e_1(t)\right] \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -\sigma(t)v_r(t)\cos e_3(t) \\ 0 \end{pmatrix}$$

As duas últimas linhas dos termos $f(t, e_a(t))$, $g(t, e_a(t)$ e $g_n(t, e_a(t))$ das Equações (2.54)-(2.56) são dadas pela derivada do erro de estimação (2.52), junto com a lei de adaptação (2.47).

B.2 Condição sobre a perturbação evanescente

Na Seção 2.3.3, é necessário mostrar que para algum $\gamma > 0$ constante e algum

$$D = \{e_a(t) \in \mathbb{R}^5 \mid \|e_a(t)\| < d\}$$

a perturbação evanescente $g(t, e_a(t))$ dada por (2.55) obedece a seguinte condição:

$$\|g(t, e_a(t))\| \leq \gamma \|e_a(t)\|, \quad \forall t \geq t_0, \quad \forall e_a(t) \in D$$

Usando a equação (2.42) da lei de controle e os limitantes da equação (2.58), tem-se que

$$\begin{aligned} |\omega_c(t)| &\leq |\omega_{\mathrm{r}}(t)| + \frac{|v_{\mathrm{r}}(t)|}{2} \left| k_2 \left(e_2(t) + k_3 e_3(t) \right) + \frac{1}{k_3} \sin e_3(t) \right| \\ &\leq \underbrace{L_\omega + \frac{L_v}{2k_3}}_{L_1} + \underbrace{\frac{L_v k_2}{2}}_{L_2} \left(|e_2(t)| + k_3 |e_3(t)| \right) \ . \end{aligned} \tag{B.4}$$

Com base na equação (B.4) e na equação (2.42) da lei de controle , a seguinte desigualdade pode ser desenvolvida:

$$\begin{aligned} \left| v_c(t) - \frac{b}{2}\omega_c(t) \right| &\leq \left| v_{\mathrm{r}}(t) \cos e_3(t) - \left(k_3 e_3(t) + \frac{b}{2} \right) \omega_c(t) + k_1 e_1(t) \right| \\ &\leq |v_{\mathrm{r}}(t) \cos e_3(t)| + \left| \left(k_3 e_3(t) + \frac{b}{2} \right) \omega_c(t) \right| + k_1 |e_1(t)| \\ &\leq |v_{\mathrm{r}}(t)| + \left(k_3 |e_3(t)| + \frac{b}{2} \right) |\omega_c(t)| + k_1 |e_1(t)| \\ &\leq L_v + \left(k_3 |e_3(t)| + \frac{b}{2} \right) \left(L_1 + L_2 (|e_2(t)| + k_3 |e_3(t)|) \right) + k_1 |e_1(t)| \end{aligned}$$

resultando em

$$\begin{aligned} \left| v_c(t) - \frac{b}{2}\omega_c(t) \right| \leq\ & L_v + \frac{b}{2} L_1 + k_1 |e_1(t)| + \frac{b}{2} L_2 |e_2(t)| + \underbrace{\left(k_3 L_1 + \frac{b}{2} L_2 k_3 \right)}_{L_3} |e_3(t)| \\ &+\ k_3 L_2 |e_2(t)||e_3(t)| + k_3^2 L_2 |e_3(t)|^2 \end{aligned} \tag{B.5}$$

Com um desenvolvimento idêntico ao desenvolvimento acima, pode-se mostrar que

$$\left| v_c(t) + \frac{b}{2}\omega_c(t) \right|$$

também obedece à desigualdade (B.5).

Agora, aplicando a norma 2 no termo de perturbação evanescente (2.55), tem-se

$$\|g(t, e_a(t))\|_2 = |\sigma(t)| \left| \frac{1}{2} \frac{\tilde{a}_e(t)}{a_e(t)} \left(v_c(t) - \frac{b}{2}\omega_c(t) \right) + \frac{1}{2} \frac{\tilde{a}_d(t)}{a_d(t)} \left(v_c(t) + \frac{b}{2}\omega_c(t) \right) - k_3 e_3(t) \omega_c(t) + k_1 e_1(t) \right| \tag{B.6}$$

Manipulando o valor absoluto do lado direito da equação (B.6) e lembrando que, por definição, $a_e(t) \in [1, \infty)$ e $a_d(t) \in [1, \infty)$, tem-se

$$\|g(t, e_a(t))\|_2 \leq |\sigma(t)| \left(\frac{1}{2}|\tilde{a}_e(t)| \left| v_c(t) - \frac{b}{2}\omega_c(t) \right| + \frac{1}{2}|\tilde{a}_d(t)| \left| v_c(t) + \frac{b}{2}\omega_c(t) \right| + k_3|e_3(t)||\omega_c(t)| + k_1|e_1(t)| \right)$$

Usando a desigualdade (B.5) para $|v_c(t) - b\omega_c(t)/2|$ e também para $|v_c(t) + b\omega_c(t)/2|$, usando a desigualdade (B.4) para $|\omega_c(t)|$, obtém-se

$$\begin{aligned} \|g(t, e_a(t))\|_2 \leq \quad |\sigma(t)| \quad & \left[\frac{1}{2}\big(|\tilde{a}_e(t)| + |\tilde{a}_d(t)|\big) \left(L_v + \frac{b}{2}L_1 + k_1|e_1(t)| + \frac{b}{2}L_2|e_2(t)| \right. \right. \\ & \left. + L_3|e_3(t)| + k_3L_2|e_2(t)||e_3(t)| + k_3^2L_2|e_3(t)|^2 \right) \\ & \left. + k_3|e_3(t)|\big(L_1 + L_2(|e_2(t)| + k_3|e_3(t)|)\big) + k_1|e_1(t)| \right] \end{aligned} \tag{B.7}$$

Usando o fato que $|e_1(t)|, |e_2(t)|, |e_3(t)|, |\tilde{a}_d(t)|, |\tilde{a}_e(t)| \leq \|e_a(t)\|_2$, desenvolvendo a desigualdade (B.7) e organizando os coeficientes resulta em

$$\begin{aligned} \|g(t, e_a(t))\|_2 \quad \leq \quad |\sigma(t)| \Bigg[& \underbrace{\left(L_v + \frac{b}{2}L_1 + k_1 + k_3L_1 \right)}_{L_4} \|e_a(t)\|_2 \\ & + \underbrace{\left(k_1 + \frac{b}{2}L_2 + L_3 + k_3L_2 + k_3^2L_2 \right)}_{L_5} \|e_a(t)\|_2^2 + \underbrace{\left(k_3L_2 + k_3^2L_2 \right)}_{L_6} \|e_a(t)\|_2^3 \Bigg] \end{aligned}$$

Juntando os termos constantes, tem-se

$$\|g(t, e_a(t))\|_2 \leq |\sigma(t)|\big(L_4\|e_a(t)\|_2 + L_5\|e_a(t)\|_2^2 + L_6\|e_a(t)\|_2^3\big)$$

Finalmente, no domínio $D_1 = \{e_a(t) \in \mathbb{R}^5 \mid \|e_a(t)\| \leq 1\}$ a seguinte desigualdade valerá:

$$\|g(t, e_a(t))\|_2 \leq |\sigma(t)|\big(L_4 + L_5 + L_6\big)\|e_a(t)\|_2 \tag{B.8}$$

Se $\sigma(t)$ for limitado,

$$|\sigma(t)| \leq L_\sigma$$

então tem-se

$$\|g(t, e_a(t))\|_2 \leq L_\sigma L\|e_a(t)\|_2, \quad \forall e_a(t) \in D_1$$

em que

$$L = L_4 + L_5 + L_6 \tag{B.9}$$

com

$$L_4 = L_v + k_1 + \left(\frac{b}{2} + k_3\right)\left(L_\omega + \frac{L_v}{2k_3}\right)$$
$$L_5 = k_1 + \left(\frac{b}{2}(1 + k_3) + k_3 + k_3^2\right)\left(\frac{L_v k_2}{2}\right) + k_3\left(L_\omega + \frac{L_v}{2k_3}\right)$$
$$L_6 = \left(k_3 + k_3^2\right)\left(\frac{L_v k_2}{2}\right)$$

Apêndice C

Curvas de Consumo de Memória

Neste Apêndice, serão apresentados resultados do Capítulo 4.3 que não haviam sido apresentados anteriormente por brevidade.

A Tabela C.1 apresenta as fórmulas de ajuste do consumo máximo de memória utilizado pelo Yalmip/SOS na resolução do problema (4.39), para cada número de variáveis n fixo. O teste utilizado para a obtenção dessas fórmulas é descrito no Capítulo 4.3, e os ajustes estão representados graficamente nas Figuras 4.8-4.13.

Para fins de comparação, os ajustes obtidos com o uso do SOSTools são apresentados na tabela C.2. Eles estão representados graficamente nas figuras 4.2-4.7. Os ajustes dos outros três *toolboxes*, Yalmip/MOM, Gloptipoly e SparsePOP, são consideravelmente parecidos com os ajustes do SOSTools (como evidenciado pelas Figuras 4.14 e 4.15), e não serão reproduzidos aqui por brevidade. Os gráficos representando os ajustes desses três *toolboxes* são apresentadas nas Figuras C.1-C.18.

Tabela C.1: Ajuste do consumo máximo de memória: Yalmip/SOS.

Número de variáveis	Consumo máx. de memória em função do grau $2d$ (em $[\log_{10}$Mb $]$)
5	$0.0029761(2d)^2 - 0.035118(2d) + 2.3473$
6	$0.0064905(2d)^2 - 0.073961(2d) + 2.4321$
7	$0.011301(2d)^2 - 0.12365(2d) + 2.538$
8	$0.015754(2d)^2 - 0.15759(2d) + 2.5846$
9	$0.020404(2d)^2 - 0.18977(2d) + 2.6221$
10	$0.023705(2d)^2 - 0.20033(2d) + 2.6077$
11	$0.030092(2d)^2 - 0.25032(2d) + 2.6966$
12	$0.032632(2d)^2 - 0.24852(2d) + 2.665$
13	$0.039485(2d)^2 - 0.30068(2d) + 2.7377$
14	$0.030687(2d)^2 - 0.1487(2d) + 2.411$
15	$0.016559(2d)^2 + 0.058753(2d) + 2.0101$
16	$0.024378(2d)^2 + 0.03883(2d) + 2.1473$

Tabela C.2: Ajuste do consumo máximo de memória: SOSTools.

Número de variáveis	Consumo máx. de memória em função do grau $2d$ (em $[\log_{10}$Mb $]$)
5	$0.013059(2d)^2 - 0.084918(2d) + 2.3019$
6	$0.025971(2d)^2 - 0.1674(2d) + 2.4335$
7	$0.040248(2d)^2 - 0.23434(2d) + 2.5055$
8	$0.066951(2d)^2 - 0.40363(2d) + 2.7517$
9	$0.075662(2d)^2 - 0.39311(2d) + 2.669$
10	$0.11407(2d)^2 - 0.60544(2d) + 2.9481$
11	$0.1409(2d)^2 - 0.73168(2d) + 3.1005$
12	$0.15697(2d)^2 - 0.77228(2d) + 3.1097$
13	$0.16898(2d)^2 - 0.7869(2d) + 3.0913$

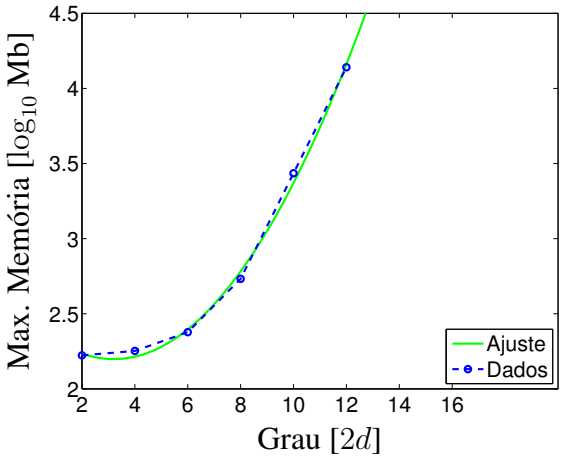

Figura C.1: Consumo de memória pelo Yalmip/MOM na resolução de (4.39): $n = 6$.

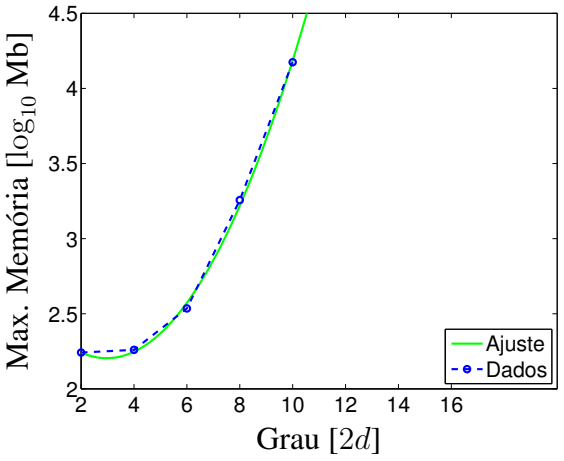

Figura C.2: Consumo de memória pelo Yalmip/MOM na resolução de (4.39): $n = 7$.

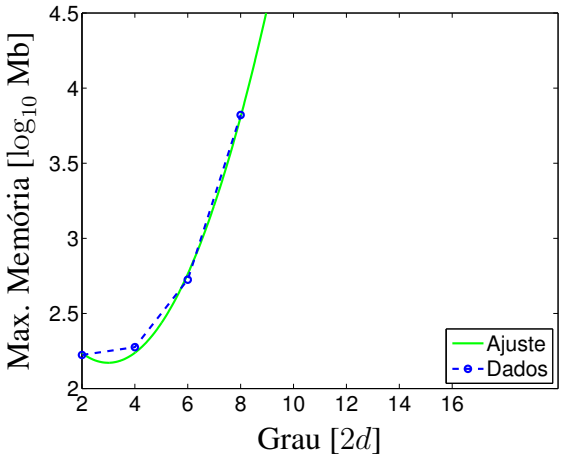

Figura C.3: Consumo de memória pelo Yalmip/MOM na resolução de (4.39): $n = 8$.

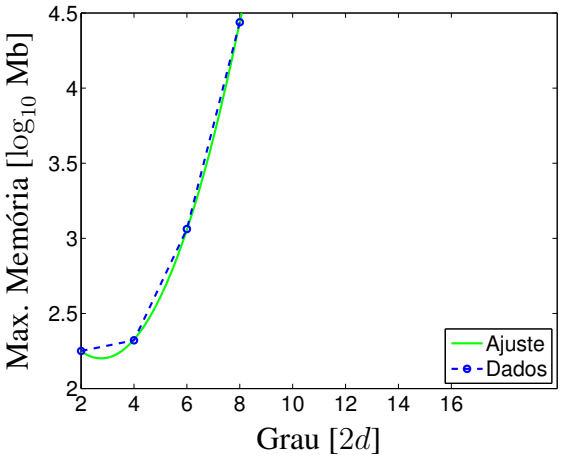

Figura C.4: Consumo de memória pelo Yalmip/MOM na resolução de (4.39): $n = 9$.

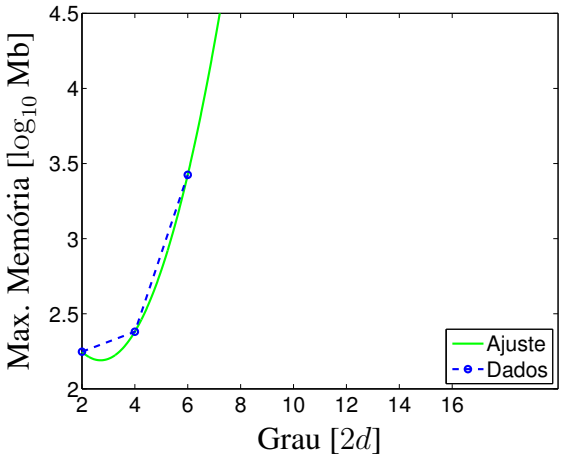

Figura C.5: Consumo de memória pelo Yalmip/MOM na resolução de (4.39): $n = 10$.

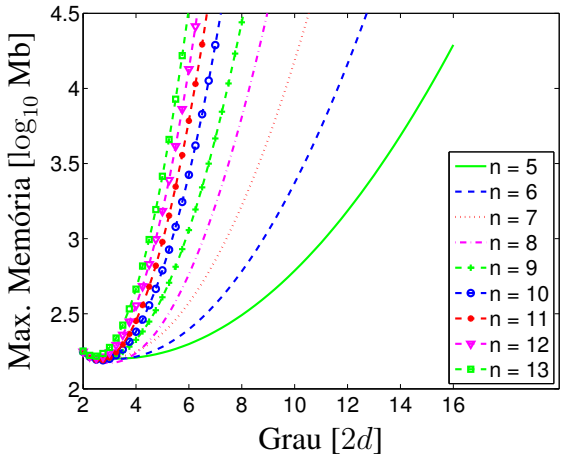

Figura C.6: Consumo de memória pelo Yalmip/MOM na resolução de (4.39).

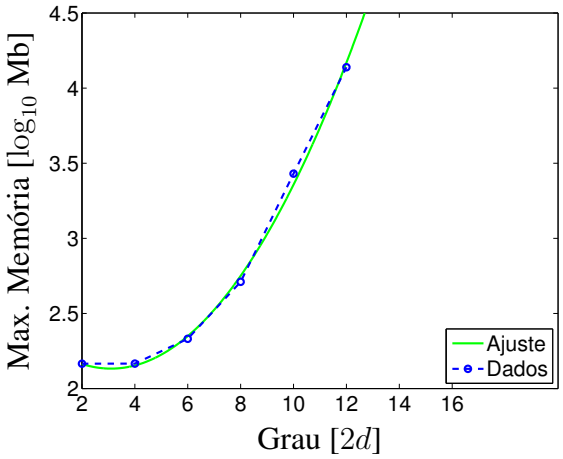

Figura C.7: Consumo de memória pelo Gloptipoly na resolução de (4.39): $n = 6$.

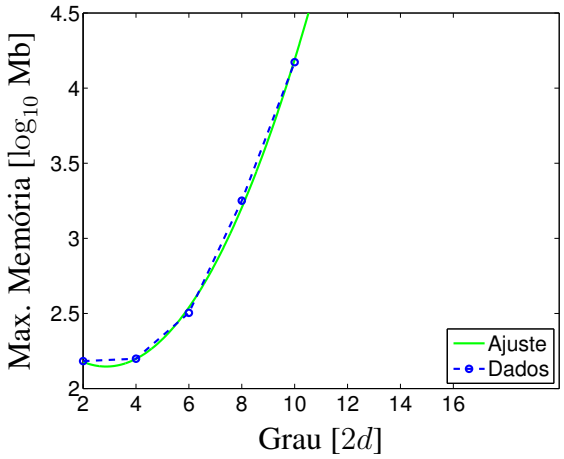

Figura C.8: Consumo de memória pelo Gloptipoly na resolução de (4.39): $n = 7$.

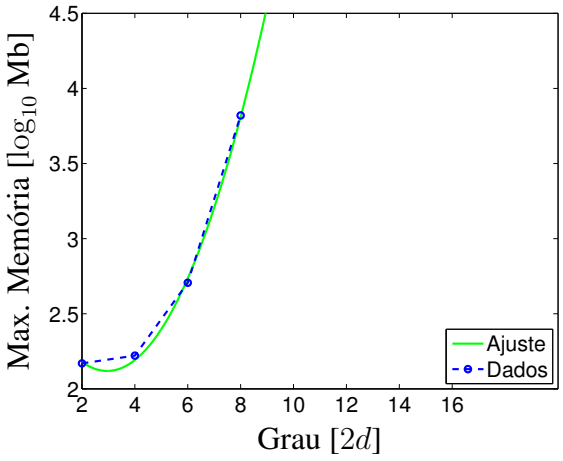

Figura C.9: Consumo de memória pelo Gloptipoly na resolução de (4.39): $n = 8$.

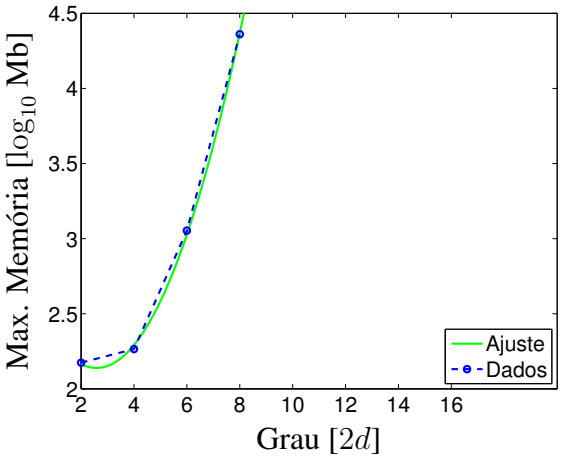

Figura C.10: Consumo de memória pelo Gloptipoly na resolução de (4.39): $n = 9$.

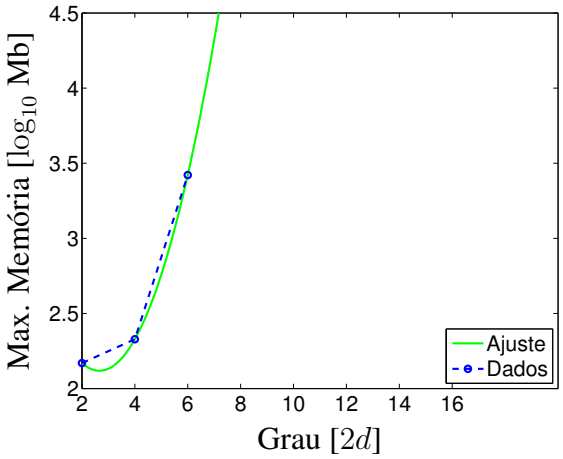

Figura C.11: Consumo de memória pelo Gloptipoly na resolução de (4.39): $n = 10$.

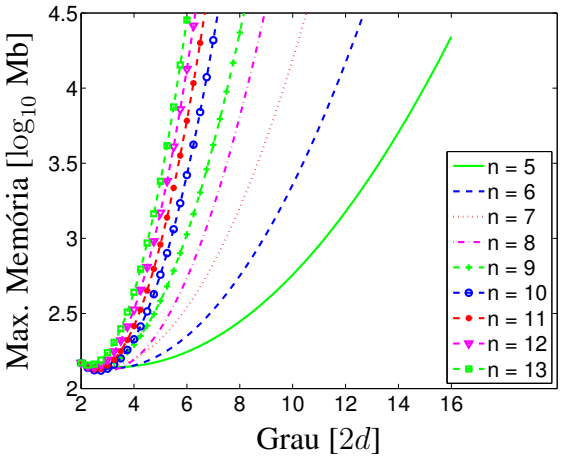

Figura C.12: Consumo de memória pelo Gloptipoly na resolução de (4.39).

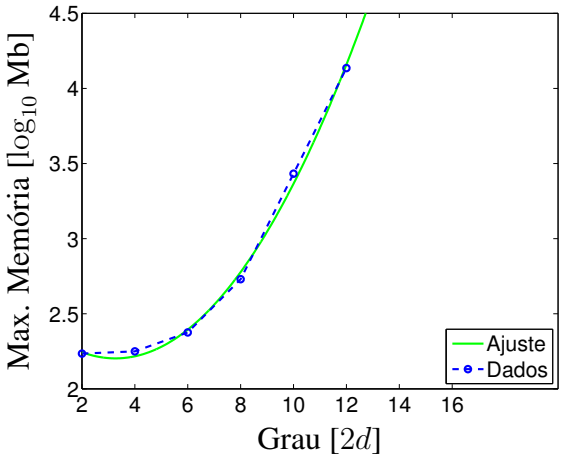

Figura C.13: Consumo de memória pelo SparsePOP na resolução de (4.39): $n = 6$.

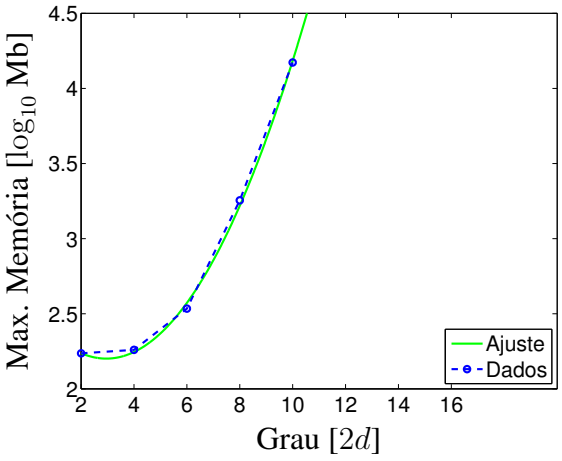

Figura C.14: Consumo de memória pelo SparsePOP na resolução de (4.39): $n = 7$.

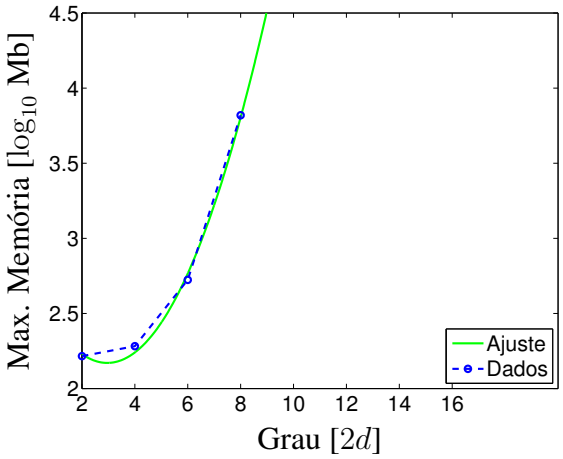

Figura C.15: Consumo de memória pelo SparsePOP na resolução de (4.39): $n = 8$.

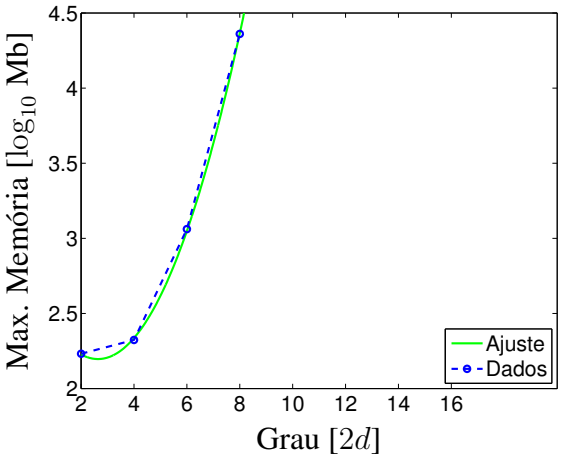

Figura C.16: Consumo de memória pelo SparsePOP na resolução de (4.39): $n = 9$.

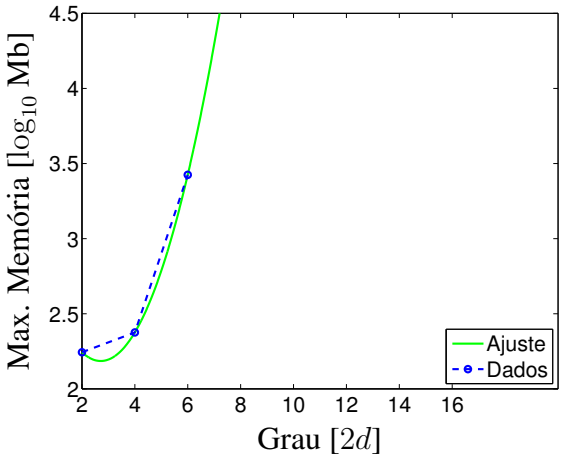

Figura C.17: Consumo de memória pelo SparsePOP na resolução de (4.39): $n = 10$.

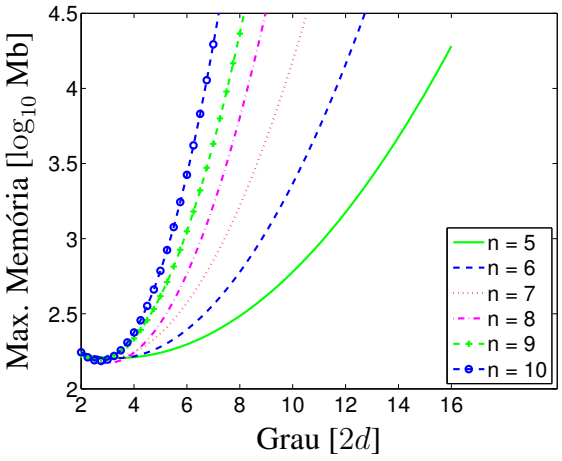

Figura C.18: Consumo de memória pelo SparsePOP na resolução de (4.39).