UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GUSTAVO PERIM COLA WEINKELLER

CONTROLE PREDITIVO DE SISTEMAS *PWA* VIA PROGRAMAÇÃO MULTIPARAMÉTRICA APLICADO EM UMA CADEIRA DE RODAS ROBÓTICA

VITÓRIA
2012

GUSTAVO PERIM COLA WEINKELLER

CONTROLE PREDITIVO DE SISTEMAS *PWA* VIA PROGRAMAÇÃO MULTIPARAMÉTRICA APLICADO EM UMA CADEIRA DE RODAS ROBÓTICA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica, na área de Controle Automático e robótica.
Orientador: Prof. Dr. José Leandro F. Salles
Co-orientador: Prof. Dr. Teodiano Freire Bastos Filho.

VITÓRIA
2012

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Weinkeller, Gustavo Perim Cola, 1979-
W423c Controle preditivo de sistemas pwa via programação multiparamétrica aplicado em uma cadeira de rodas robótica / Gustavo Perim Cola Weinkeller. – 2012.
121 f. : il.

Orientador: José Leandro F. Salles.
Coorientador: Teodiano Freire Bastos Filho.
Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Controle preditivo. 2. Otimização matemática 3. Robótica. 4. Cadeiras de rodas. I. Salles, José Leandro Félix. II. Bastos Filho, Teodiano Freire. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

A Dissertação **Controle Preditivo de Sistemas PWA Via Programação Multiparamétrica Aplicado em Uma Cadeira de Rodas Robótica**, elaborada por **Gustavo Perim Cola Weinkeller** orientado(a) pelo(s) Professor(es) José Leandro Félix Salles e Teodiano Freira Bastos Filho, aprovada por todos os membros da Banca Examinadora, foi aceita pelo Colegiado de Curso do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Espírito Santo, como requisito parcial à obtenção do título de:

MESTRE EM ENGENHARIA ELÉTRICA

Vitória, 24 de agosto de 2012.

BANCA EXAMINADORA

José Leandro Félix Salles, Dr.
Orientador

Teodiano Freire Bastos Filho, Dr.
Co-orientador

Wanderley Cardoso Celeste, Dr.
Membro Interno

André Gustavo Scolari Conceição
Membro Externo

“Os que sonham de dia são conscientes de muitas coisas que escapam aos que sonham apenas à noite”

Edgar Allan Poe

Dedico este trabalho à humanidade
como humilde contribuição às gerações vindouras.

Agradecimentos

Ao CNPq pelo apoio financeiro concedido, sem o qual esta pesquisa não teria sido possível.

Aos meus orientadores, Prof. Dr. José Leandro F. Salles e Prof. Dr Teodiano Freire Bastos Filho, por terem insentivado e apostado no estudo de uma abordagem nova e pouco difundida, interligando duas áreas até então distantes dentro do PPGEE na UFES. Sem seus esforços este trabalho não teria sido possível, em especial do Prof. Dr. José Leandro F. Salles, que praticamente voltou a ser aluno em prol do sucesso deste desafio. Muito obrigado!

À minha noiva, que sofreu com minhas ausências e sempre esteve ao meu lado me incentivando e apoiando.

À minha família, que sempre me deu o suporte necessário para atingir esta meta, em especial meu tio Toninho, grande incentivador do meu lado acadêmico.

Gostaria também de realizar um agradecimento póstumo ao meu tio, José Carlos Bento. Sem ele eu não estaria onde estou hoje e parte desta conquista é dele também.

Enfim, a todos que me aturaram, apoiaram e insentivaram, muito obrigado!

Resumo

Este trabalho propõe a modelagem de dois controladores preditivos híbridos, um para o sistema dinâmico e outro para o sistema cinemático de uma cadeira de rodas robótica. Esses controladores são calculados explicitamente (*off-line*) e são funções lineares por partes, do tipo PWA (*PieceWaise Affine*). Para gerar estes controladores *off-line*, foi utilizado um método de otimização conhecido como Programação Multiparamétrica (PM). A PM é parte integrante do *MPT Toolbox* para MatLab®, especialmente desenvolvido para sua aplicação em controle preditivo. Para se desenvolver os controladores foi preciso linearizar ambos os sistemas. A linearização do sistema dinâmico foi relativamente simples, precisando de apenas um ponto de operação e gerando uma resposta satisfatória. Contudo, a linearização do sistema cinemático foi mais problemática de se conseguir um resultado aceitável. De posse dos sistemas linearizados e representados por espaço de estados, deu-se a modelagem dos controladores. Foram realizadas simulações computacionais dos controladores obtidos, juntamente com os modelos não lineares dos sistemas, sendo verificado que ambos garantiram suas estabilidades. O controlador dinâmico apresentou excelentes resultados, respondendo de forma rápida e eficiente. Já o controlador cinemático apresentou um resultado inferior, se comparado ao controlador dinâmico, demandando maior tempo computacional e necessitando que fosse sintonizado a cada mudança de *setpoint*.

Palavras chaves: Controle Preditivo Hibrido, Robótica Móvel, Cadeira de Rodas Robótica, Programação Multiparamétrica, Sistemas Afins Por Partes.

Abstract

This dissertation proposes the modeling of two hybrid predictive controllers, one for the dynamic system and another for the kinematic system of a robotic wheelchair. These controllers are calculated explicitly and are piecewise linear functions, like PWA (PieceWaise Affine). To generate these controllers explicitly it was used an optimization method known as Multi-parametrical Programming (MP). The PM is part of the MPT Toolbox for Matlab ®, specially developed for its application in predictive control. To develop the controllers was necessary to linearize both systems. The linearization of the dynamic system was simple, needing only an operation point and generating a satisfactory answer. However, the linearization of the kinematic system was more problematic to achieve an acceptable result. From the linearized system and represented by state space, it was done the modeling of controllers. They were performed computer simulations of the controllers together with the nonlinear systems, and it was verified that both ensured their stability. The dynamic controller showed excellent results, responding quickly and efficiently. Since the kinematic controller presented an inferior result, as compared to the dynamic controller, requiring more computational time and needing to be tuned to each new setpoint.

Key words: Hybrid Predictive Control, Mobile Robotic, Robot Wheelchair, Multi-parametrical Programming, PieceWise Affine Systems.

Lista de Figuras

Lista de Tabelas

Siglas

CARIMA – Controlador Auto Regressivo Integrador com Média Móvel

COTFR – Controle Ótimo no Tempo Finito com Restrições

DMC – Dynamic Matrix Control

FLC – Fuzzy Logic Controller

GDA – Gradient Decent Algorithm

GPC – Generalized Predictive Control

LTI – Linear Time Invariant

MIMO – Multiple Input Multiple Output

MPC – Model Predictive Control

MPT – Multi-Parametric Toolbox

NMPC – Nonlinear Model Predictive Control

NNPC – Neural Network Predictive Control

OAV – Organic Air Vehicle

PL – Programação Linear

PM – Programação Multiparamétrica

PML – Programação Multiparamétrica Linear

PMLI – Programação Multiparamétrica Linear Inteira

PMQ – Programação Multiparamétrica Quadrática

PQ – Programação Quadrática

PRBS – Pseudo Random Binary Sequence

PSO-CREV – Particle Swarm Optimization With Controllable Random Exploration Velocity

PWA – PieceWaise Affine

SVM – Support Vector Machine

Notações

d – Atraso ou tempo morto

e_c – Distância entre dois pontos

$F_{rrx\prime}$, $F_{rlx\prime}$ – Forças longitudinais que atuam sobre as rodas de tração

$F_{rry\prime}$, $F_{rly\prime}$ – Forças laterais que atuam sobre as rodas de tração

$F_{crx\prime}$, $F_{clx\prime}$ – Forças longitudinais que atuam sobre as rodas livres

$F_{cry\prime}$, $F_{cly\prime}$ – Forças laterais que atuam sobre as rodas livres

$F_{ex\prime}$, $F_{ey\prime}$ – Forças externas

G – Centro de massa da cadeira de rodas robótica

h_c – Horizonte de controle

h_p – Horizonte de previsão

H – ponto de referência

I – Matriz identidade

I_z – Momento de inércia no ponto G

j – Quantidade de predições

J – Função custo

m – Massa total do sistema

p – Quantidade de entradas de um sistema

q – Quantidade de saídas de um sistema

Q – Centro do eixo virtual entre as rodas da cadeira de rodas robótica

u – Lei de controle

v – Velocidade linear

v' – Velocidade longitudinal do centro de massa da cadeira de rodas

$\overline{v}$ – Velocidade lateral do centro de massas da cadeira de rodas

$\overline{v}^s$ – Velocidade de deslizamento lateral das rodas de tração

x – Posição ao longo do eixo horizontal

x_g, y_g – Coordenada de referência do robô

y – Posição ao longo do eixo vertical

$\hat{y}_m$ – Saída prevista

β_c – Orientação de referência

β_u – Ângulo entre o vetor v e o vetor e_c

δ_v, δ_ω – Distúrbios ocasionados por fatores externos

δ_m e λ_i – Parâmetros de ajuste do controlador

ξ – Vetor postura do robô

τ_p – Torque externo

ψ – Orientação do robô

ω – Velocidade angular

Δu_l – Variação no sinal de controle

$\| \cdot \|$ - Norma euclidiana de um vetor

Sumário

Capítulo 1: INTRODUÇÃO

Ainda hoje, um grande número de pessoas que possuem algum tipo de deficiência são excluídas do mercado de trabalho e discriminadas em seus âmbitos sociais. Esse processo de exclusão social é tão antigo quanto o processo de socialização do homem. A estrutura social sempre marginalizou e privou os portadores de deficiência, fazendo com que essas pessoas sejam alvo de atitudes preconceituosas, pois é mais fácil prestar atenção aos impedimentos e às aparências do que aos potenciais e capacidades de tais pessoas.

Porém, diversos segmentos da ciência estudam uma melhor forma de viabilizar a inserção de pessoas com limitações físicas na sociedade nas quais estão inseridas, facilitando sua comunicação, sua locomoção e sua interação social como um todo. Uma dessas vertentes tem seu foco voltado para cadeiras de rodas especiais ou inteligentes. Essas cadeiras especiais podem se locomover sozinhas (CELESTE, 2009) ou sob comandos específicos, como por exemplo, comandos dados por posicionamento ocular, respiração, ondas cerebrais (SILVA, 2007), dentre outros.

Esse tipo de cadeira de rodas foi desenvolvida com foco em pessoas que apresentam um alto grau de deficiência motora, como por exemplo, paralisia permanente ou ausência de membros superiores e inferiores, mas que ainda sejam capazes de realizar algum movimento muscular voluntário e que tenham atividade cognitiva preservada (CELESTE, 2009).

A finalidade de uma cadeira de rodas inteligente é permitir que, pessoas com as características citadas, consigam locomover-se sem a ajuda de terceiros. Para isso, a pessoa informa à cadeira seu desejo de se locomover, seja para um local pré-determinado ou para uma direção específica, e ela provê os meios para que a solicitação seja atendida. Por se tratar de um veículo autônomo ou com navegação assistida, a cadeira de rodas inteligente pode ser definida como um robô móvel.

Devido à natureza de seu usuário, diversas questões, como segurança, conforto e confiabilidade, devem ser levadas em consideração no projeto desse tipo de cadeira, uma vez que toda a responsabilidade na execução do trajeto é transferida para ela. Para isso, o sistema embarcado na cadeira tem de identificar e interpretar com precisão o comando solicitado e possuir um controlador bem sintonizado. A Figura 1.1 mostra o diagrama de blocos de tal identificação e interpretação (CELESTE, 2009).

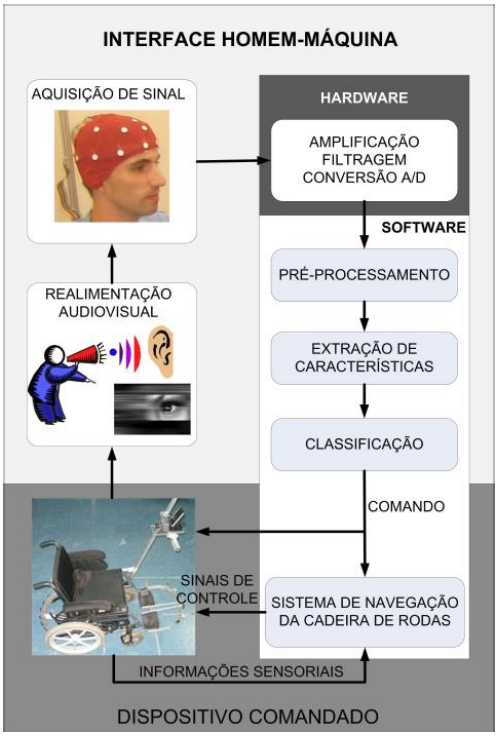


Figura 1.1: Diagrama de blocos do sistema de assistência a pessoas com deficiência. (CELESTE, 2009)

O controlador desse tipo de veículo tem de apresentar robustez em se tratando de perturbações, principalmente às externas ao sistema (cadeira de rodas mais usuário), pois, pessoas podem esbarrar na cadeira ou seu usuário pode se movimentar durante a execução de uma trajetória, alterando assim a dinâmica do sistema. Para isso o controlador deve responder rápido o suficiente para corrigir a trajetória, mas não tão rápido ao ponto de colocar em risco a integridade de seu usuário ou das pessoas ao seu redor. Outra característica inerente ao controlador é que ele seja multivariável e que suas variáveis possam ser limitadas. No caso da cadeira de rodas robótica, as variáveis a serem limitadas são as velocidades angular e linear, proporcionando assim um maior conforto e segurança ao cadeirante. Um controlador que possui as características adequadas para este tipo de sistema é o controlador ótimo de tempo finito ou preditivo (CELESTE, 2009).

O controle preditivo originou-se na década de 60, como um caso particular de controle ótimo. O controle ótimo calcula soluções ótimas em cada instante ao longo de um horizonte de tempo fixo, enquanto que o preditivo calcula as soluções subotimas para um horizonte de tempo móvel e finito. Este tipo de controlador é empregado com sucesso em sistemas restritos, multivariáveis e com atraso de tempo (CAMACHO e BORDONS, 2004).

De uma forma geral, o controlador preditivo é um algoritmo que calcula valores futuros de entrada e saída baseados em um Horizonte de Previsão e sinais de controle baseados em um Horizonte de Controle. Os sinais de controle são resultados da otimização de um critério, que é uma função do erro entre as saídas previstas e as referências previstas, chamadas função custo ou função objetivo. Essa otimização é um algoritmo que leva em consideração a função custo e todas as restrições impostas ao sistema. A partir do conjunto de sinais de controle obtidos, descartam-se os valores previstos e aplica-se ao sistema o valor referente ao sinal do instante atual (estratégia de controle com horizonte retrocedente). A Figura 1.2 ilustra a estratégia de um controlador preditivo (CAMACHO e BORDONS, 2004).

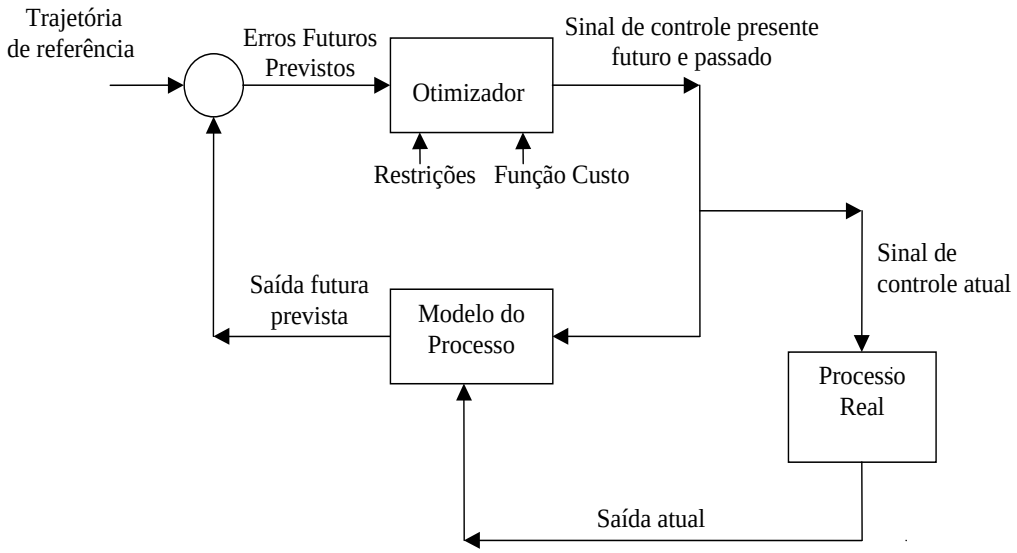


Figura 1.2: Estratégia de um controlador preditivo. Adaptado de (CAMACHO e BORDONS, 2004)

Os métodos de otimização usuais realizam todos os seus cálculos a cada iteração, o que, dependendo da quantidade de entradas, saídas e restrições, dificulta seu uso para sistemas com dinâmicas rápidas. O método de otimização por Programação Multiparamétrica, PM (BEMPORAD e MORARI, 1999), realiza todos os cálculos necessários para a obtenção das ações de controle explicitamente ou *off-line*, ou seja, antes do sistema estar em operação. Isso quer dizer que, antes do sistema entrar em operação, já serão conhecidos todos os valores factíveis de ações de controle, trazendo agilidade para o controle preditivo e um menor dispêndio de poder computacional (KVASNICA, 2009).

O conjunto factível de ações de controle gerado pela PM é dividido em subconjuntos de espaço de estados restritos, onde um sinal de controle específico é determinado para cada

estado do sistema. O modelo dinâmico usado para o cálculo das previsões será representado por uma equação de estados linear por partes, caracterizando-se, portanto, como um sistema dinâmico *PWA*. Sistemas *PWA* são equivalentes a sistemas dinâmicos lineares com variáveis contínuas e lógicas, denominados sistemas híbridos ou sistemas dinâmicos com variáveis contínuas e lógicas, (*MLD* - Mixed Logical Dynamical) (BEMPORAD e MORARI, 1999)

O controle da cadeira de rodas robótica para pessoas portadoras de necessidades especiais deve ser realizado de maneira eficiente e estável, de forma a garantir a integridade de seu usuário. É com base nestas premissas e considerando as vantagens do controle preditivo, que essa dissertação visa estudar a viabilidade da aplicação de dois controladores preditivos, um para o modelo dinâmico e outro para o modelo cinemático da cadeira de rodas robótica. Estes controladores serão desenvolvidos usando, como método de otimização, a Programação Multiparamétrica, PM.

Para se desenvolver o controlador preditivo de sistemas *PWA* serão utilizados os modelos matemáticos do sistema dinâmico e cinemático da cadeira de rodas robótica em espaço de estados. Os dados dos sistemas, assim como os parâmetros dos controladores, serão inseridos em um *toolbox* para MatLab® denominado, *Multi-Parametric Toolbox – MPT*, que foi desenvolvido por (KVASNICA, 2009). Este *toolbox* é utilizado para gerar o conjunto factível das ações de controle. Serão realizadas simulações para medir o tempo computacional do processo, assim como a eficiência do controlador.

1.1 CONTROLE PREDITIVO APLICADO À ROBÓTICA MÓVEL

Shinya *et al*. (2010), propõem um novo método de controle ótimo para rastreamento de robô móvel de duas rodas com restrição não-holonômicas[1]. O robô móvel de duas rodas foi modelado como um Sistema Dinâmico Híbrido, onde cada subsistema linearizado em um diferente ponto de equilíbrio é alterado de acordo com o estado do sistema. O problema de controle ótimo foi reduzido para um problema de programação quadrática inteira mista, que foi resolvido com a ajuda um controle preditivo baseado em modelo, *MPC* (*Model Predictive Control*).

[1]Restrição não holonômica é quando as variáveis de posição não são suficientes para se descrever um determinado estado.

No trabalho, dois robôs são colocados em uma área delimitada com uma câmera posicionada acima do chão. Conforme um robô segue o outro, a câmera filma a localização do robô seguidor e envia a coordenada em tempo real para um computador, onde os cálculos do *MPC* são realizados e transmitidos via *bluetooth* de volta ao robô seguidor.

Para resolver o problema das não linearidades do modelo do robô foram feitas algumas linearizações em pontos equidistantes e limitados entre o intervalo de $[0°, 180°]$, tornando assim o sistema *PWA*. Tanto as simulações quanto os experimentos apresentaram resultados que comprovam a eficácia do controlador proposto.

Uma interessante aplicação de controle preditivo hibrido foi apresentada por (CHIOU e WANG, 2008) em futebol de robôs. O controlador preditivo hibrido do tipo *GPC* (*Generalized Predictive Control*) proposto no trabalho inclui uma *SVM* (*Support Vector Machine*) e um *FLC* (*Fuzzy Logic Controller*), tendo como objetivo otimizar o movimento de um robô móvel. O *GPC* foi utilizado para prever a posição de um determinado alvo no instante de tempo seguinte, a *SVM* ficou responsável pela determinação do ângulo ótimo e da velocidade necessária para o robô atingir o alvo e, por fim, o *FLC*, responsável pelo envio dos sinais às rodas esquerda e direita.

Para demonstrar a eficácia do controlador hibrido, foram realizadas 3 práticas diferentes. Na primeira prática, foi realizada uma comparação entre as trajetórias realizadas por um robô com o *GPC* e outro com um controlador normal, sendo constatado que o robô com *GPC* chega mais rápido ao destino. A segunda prática mostrou um comparativo antes da implementação de *SVM*, e depois de sua implementação. Por fim, a terceira simulação foi realizada com o intuito de comparar a eficácia entre o *FLC* e o controlador preditivo hibrido. Observou-se que o controlador hibrido se mostrou mais eficiente que o *FLC*.

Kanjanawanishkul e Zell (2008), propõem a aplicação de um controlador preditivo para coordenarem um grupo de três robôs omnidirecionais por um dado caminho. O controlador em questão é um NMPC (*nonlinear model predictive control*). O *NMPC* apresentado é responsável pelo controle do caminho a ser seguido e pela coordenação dos robôs. Cada robô móvel ajusta sua própria velocidade ao longo do caminho e sua movimentação de translação e rotação é controlada individualmente. Para se realizar a coordenação dos robôs, as velocidades ao longo da trajetória de cada robô foram ajustadas de acordo com as informações de seus “vizinhos”. Essas informações compõem a função custo do *NMPC*.

Keviczky *et. al.* (2008), também apresentam uma aplicação de controle e coordenação de robôs móveis. Nesse trabalho o controle preditivo é responsável por sincronizar o vôo de robôs do tipo *OAV* (*Organic Air Vehicle*).

Em (KLANCAR e SKRJANC, 2007) é apresentado um modelo de um controle preditivo de rastreamento de trajetória aplicado em robótica móvel, cuja lei de controle é derivada de uma função custo quadrática. O controlador proposto possui restrições na velocidade e na aceleração, evitando que o robô sofra escorregamento. Um Preditor de Smith é usado para compensar o tempo morto do sistema de visão embarcado. Foram realizados experimentos em um robô móvel real, onde o objetivo era comparar o *MPC* modelado com um controlador por realimentação de estados variante no tempo. Ambos os controladores se mostraram eficientes quando o robô está perto da referência, porém o *MPC* apresentou melhores resultados de controle.

O trabalho de (CHEN e LI, 2007) propõe a otimização da técnica de Controle Preditivo por Redes Neurais (*NNPC - Neural Network Predictive Control*), objetivando sua aplicação em sistemas de dinâmica rápida, como robôs móveis. O controle *NNPC* normalmente demanda um alto poder computacional, o que inviabiliza sua utilização em sistemas do tipo citado. Assim, foi proposta a técnica de otimização com algoritmo evolucionário *PSO-CREV* (*Particle Swarm Optimization With Controllable Random Exploration Velocity*) para reduzir o tempo computacional do *NNPC*, aplicado no rastreamento de trajetórias de um robô móvel.

Os experimentos realizados compararam a velocidade de resposta e o tempo computacional utilizado entre o *NNPC* otimizado por *PSO-CREV* e pelo *GDA* (G*radient Decent Algorithm*). Segundo apresentado, os resultados se mostraram satisfatórios, onde a convergência do sistema se mostrou rápida e o consumo computacional menor.

Outra abordagem de *MPC* aplicado em controle de direção de um robô móvel é apresentada em (FALCONE, BORRELLI, *et al.*, 2007), onde é calculado o ângulo de orientação ao longo de um trajeto em uma estrada escorregadia. No trabalho são feitas três abordagens, a primeira com um controlador baseado no modelo não linear do robô e a segunda e terceira com controladores baseados em sucessivas linearizações *online* do modelo do robô.

O primeiro *MPC* desenvolvido resolve problemas de otimização não linear a cada instante de tempo, mostrando-se problemático, uma vez que o poder computacional exigido foi alto devido à natureza rápida do sistema. O segundo *MPC* visa amenizar esse problema realizando sucessivas linearizações em cada instante de tempo. Com isso, o controlador resolveria um

problema de otimização linear, e consequentemente, necessitando de menos tempo computacional. O terceiro *MPC* foi obtido de forma semelhante ao anterior, porém com uma ordem menor. Os experimentos dos controladores *online* foram conduzidos em um veículo de passageiros se deslocando em uma estrada com neve.

Yoo *et. al.* (2006) propõem em seu trabalho um *GPC* baseado em redes neurais para rastrear o caminho de um robô móvel. A rede neural foi utilizada como um identificador de modelo *online*, sendo responsável por identificar os estados do robô. O *GPC* foi modelado levando em consideração o modelo dinâmico e cinemático do robô. As simulações comprovaram uma rápida adaptabilidade e a habilidade de estabilização do sistema proposto.

Kühne *et. al.* (2005) apresentam um *MPC* aplicado ao problema de rastreamento de trajetórias de robôs móveis não-holonômicos. O problema é abordado em duas vertentes, a primeira com um *NMPC* e a segunda com um *MPC*. A aplicação linear foi otimizada com Programação Quadrática e teve como objetivo a diminuição do tempo computacional utilizado.

Verifíca-se em todas as aplicações comentadas nesta revisão bibliográfica que o tempo computacional para a realização dos cálculos do controlador preditivo é uma questão relevante na utilização do controle preditivo.

1.2 OBJETIVOS E ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação propõe a modelagem e o estudo de viabilidade de dois controladores preditivos de sistema *PWA* via Programação Multiparamétrica. Esses controladores serão responsáveis pelo controle do sistema cinemático e dinâmico de uma cadeira de rodas robótica. O controlador cinemático terá 3 entradas, coordenada (x, y) e um ângulo de referência φ, e duas saídas, a velocidade linear v e a velocidade angular ω. O controlador dinâmico terá 2 entradas e 2 saídas, ambas sendo as velocidades linear e angular. Os controladores possuirão restrições de entrada e de saída e deverão apresentar robustez a perturbações e resposta rápida. Todos esses quesitos deverão ser atendidos com um mínimo de uso computacional, a fim de viabilizar a aplicação dos controladores na cadeira de rodas robótica, sistema com dinâmica rápida.

Para tal, será empregada uma técnica de otimização que permite a obtenção das leis de controle antes da inicialização do sistema, a Programação Multiparamétrica. A Programação Multiparmétrica é parte integrante do *MPT*, desenvolvido por Kvasnica (2009) para o Matlab®. Todos os parâmetros inerentes à sintonia dos controladores, bem como os modelos dos sistemas dinâmico e cinemático em espaço de estados, são inseridos nesse *toobox*.

Após a modelagem dos controladores, serão realizadas simulações computacionais. Essas simulações visam verificar a aplicabilidade dos controladores em um sistema real.

Esta dissertação está dividida em 5 capítulos: O capítulo 2 apresenta uma visão geral de robótica móvel e suas aplicações e detalha a modelagem do sistema cinemático e dinâmico, tando de forma genérica quanto de uma cadeira de rodas robótica. O controle não linear da cadeira de rodas usando Lyapunov, proposto por (CELESTE, 2009), também é discutido; o capítulo 3 apresenta um breve histórico do controle preditivo e descreve sobre as técnicas de controle preditivo GPC e no espaço de estados para sistemas lineares invariantes no tempo (*LTI – Linear Time Invariant*), bem como para sistemas PWA utilizando Programação Multiparamétrica; no capítulo 4 são obtidos os modelos *PWA* do sistema dinâmico e cinemático, além de mostrar os procedimentos realizados no *MPT* para obtenção de seus respectivos controladores. Também é realizada a simulação dos modelos não lineares com os controladores implementados. Os estudos sobre o controlador dinâmico gerou um artigo para o comgresso SBR/LARS 2012 (WEINKELLER, SALLES e BASTOS, 2012); por fim o capítulo 5 mostra uma síntese dos resultados obtidos, assim como suas conclusões. Novas idéias e perspectivas de trabalhos futuros também são abordadas.

Capítulo 2: ROBÓTICA MÓVEL

Desde os egípcios que a humanidade demostra o desejo de reproduzir os movimentos e ações dos seres humanos em máquinas, como estátuas com articulações móveis e ajustáveis e marionetes com polias e pesos. No século XVIII, a construção de autômatos[2] teve um crescimento considerável, tornando-se, alguns deles, famosos como o pato mecânico de Jacques de Vaucanson, que imitava os movimentos de um pato real, comia e evacuava.

O termo robô foi utilizado pela primeira vez no ano de 1921 pelo checo Karel Capeck em seu livro *Rossum's Universal Robots* e tem origem da palavra tcheca *robota*, tendo como significado, trabalho escravo.

Os robôs como conhecemos hoje só passaram a ser viáveis com a invenção do transistor, podendo então ser controlados por computador. O primeiro robô com tais características foi patenteado por George Devol em 1954. Ele era fixo e possuía um braço com uma garra, sendo comercializado sete anos depois para as linhas de montagem da *General Motors*, dando início assim a era da automação industrial. Este tipo de robô ficou popularizado como robô industrial (TEIXEIRA, 2000).

Alguns anos depois surgiram os robôs que não eram fixos em uma base, eram controlados por computador e que possuíam diversos tipos de sensores e atuadores para que o mesmo fosse capaz de interagir com o ambiente no qual estava inserido. Este tipo de robô ficou popularizado como robô móvel (MARCHI, 2001).

2.1 DEFINIÇÕES DOS ROBÔS MÓVEIS

Os robôs móveis podem ser divididos segundo diversas definições, dependendo do autor ou do tipo de aplicação do mesmo. Uma dessas definições os separa pelo tipo de ambiente no qual estão inseridos como, aéreos, aquáticos e terrestres, sendo o último subdividido em locomoção por meio de rodas, de esteiras e de pernas (NOURBAKHSH e SIEGWART, 2004).

[2] Autômatos: bonecos mecânicos que imitavam seres humanos e animais.

- Robôs aéreos:

Possui diversas aplicações como, vigilância e reconhecimento, combate aéreo, buscas e salvamento, comunicações e radar, espionagem, etc. Seus projetos são diversificados, podendo ser encontrados como dirigíveis, aviões, helicópteros, mísseis, no formato de insetos, dentre outros (AGOSTINO, MAMMONE, *et al.*).

- Robôs aquáticos:

Amplamente utilizado no segmento petrolífero para mapeamento oceanográfico de possíveis locais que contenham óleo e gás para exploração, realização de reparos, soldas, inspeções e apoio em atividades no fundo do mar. Sua utilização não se restringe ao setor petrolífero, sendo aplicado também em situações de busca e salvamento, no estudo da vida marinha e da topografia do oceano, etc (NIÑO, 2009).

- Robôs terrestres:

Possui uma gama diversificada de aplicações, como domésticas, auxiliando em limpezas ou como "animais" de estimação; militares, procurando e desarmando bombas e minas; esportivas, como o futebol e a batalha de robôs; exploratórias, como em expedições espaciais e em vulcões; de acessibilidade, auxiliando portadores de deficiências motoras a se locomoverem, como é o caso da cadeira de rodas robotizada; etc (PIERI, 2002) (NOURBAKHSH e SIEGWART, 2004).

Outra definição usual classifica os robôs quanto ao seu tipo de controle, podendo ser tele operado, o robô realiza os movimentos estipulados por um operador; semi-autônomos, o robô tem a autonomia de seus movimentos, porém recebe o objetivo de um operador; e autônomos, onde o robô tem total autonomia de seus movimentos e objetivos, realizando suas decisões com base nas coletas de dados do ambiente no qual está inserido (FLYNN, JONES e SEIGER, 1998) (PIERI, 2002).

2.2 ROBÔS MÓVEIS TERRESTRES

Como dito anteriormente, os robôs móveis terrestres podem ser subdivididos em 3 classes, baseadas no seu tipo de locomoção: com pernas ou patas, com esteiras ou com rodas (NOURBAKHSH e SIEGWART, 2004).

- Locomoção com pernas ou patas:

Este tipo de configuração é muito utilizada quando o robô tem de se locomover em topografias acidentadas, com subidas e descidas íngremes e em situações particulares, como ambientes com escadas. As configurações comumente utilizadas são com 2,4 e 6 pernas, inspiradas nos animais como mostra a Figura 2.1. Existem robôs com uma perna só, chamados de "saltadores", mas têm pouca utilidade. A Figura 2.2 mostra alguns tipos de robôs com patas (PIERI, 2002) (NOURBAKHSH e SIEGWART, 2004).

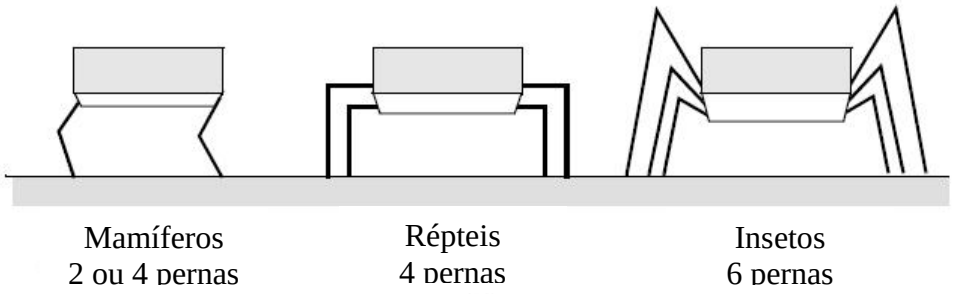


Figura 2.1: Arranjo das pernas de animais. Adaptado de (NOURBAKHSH e SIEGWART, 2004).

Figura 2.2: Exemplos de robôs com pernas (NOURBAKHSH e SIEGWART, 2004).

Apesar da versatilidade que esse tipo de robô tem para com o ambiente, a complexidade em controlar e sincronizar de maneira adequada as pernas, além dos custos e dos *hardwares* têm limitado projetos deste tipo. Isso porque cada perna possui pelo menos dois graus de liberdade[3], aumentando assim a quantidade de motores necessários para gerar o movimento e, consequentemente, aumentando a complexidade dos cálculos envolvidos para o mesmo (PIERI, 2002) (HUTCHINSON, SPONG e VIDYASAGAR, 2005).

- Locomoção com esteiras:

Esta forma de locomoção é comumente utilizada em solos arenosos e/ou com muitas pedras ou objetos espalhados aleatoriamente pelo caminho. Porém, como no caso dos robôs com pernas, a sua complexidade matemática e o seu custo são significantes devido ao fato da esteira ter de estar apoiada sob diversas rodas. Outro agravante é o problema do atrito das rodas com a esteira e da mesma com o solo, causando uma considerável dissipação de energia, diminuindo assim a eficiência do robô. Exemplo de robô com esteiras na Figura 2.3.

Figura 2.3: Exemplo de robôs com esteiras (NOURBAKHSH e SIEGWART, 2004).

[3] graus de liberdade: Em robótica, graus de liberdade referem-se as parte de um robô que são unidas por juntas. Um braço humano tem dois segmentos unidos por uma junta, o cotovelo, capaz de permitir o movimento translacional e rotacional do antebraço, logo possui dois graus de liberdade. Incluindo a mão tem-se um novo segmento unido por outra junta, o pulso, passando então a possuir três graus de liberdade (HUTCHINSON, SPONG e VIDYASAGAR, 2005).

- Locomoção com rodas:

Este tipo de robô é o que mais se popularizou e também é considerado menos complexo do que os com pernas ou esteiras. Isso se dá pelo fato dele não precisar de um *hardware* tão robusto e por possuir uma mecânica mais simples. Porém, seu uso em terrenos irregulares é uma tarefa complicada e até impossível de se realizar em determinadas situações, pelo fato da roda do robô, em geral, ter de possuir um raio maior ou igual ao obstáculo que deverá ser transposto.

Os quatro tipos de rodas comumente utilizadas, Figura 2.4, são; a roda padrão e a roda castor, que são clássicas e mais difundidas, podendo ser fixas ou com liberdade de giro em torno de seu próprio eixo vertical; a roda sueca, que possui roletes passivos ao longo de todo o perímetro da roda, proporcionando uma maior mobilidade com menos atrito; e a roda esférica, sendo este tipo de roda, considerada onidirecional[4], pois proporciona movimento em todas as direções, porém sua complexa utilização a torna pouco usual (PIERI, 2002) (NOURBAKHSH e SIEGWART, 2004).

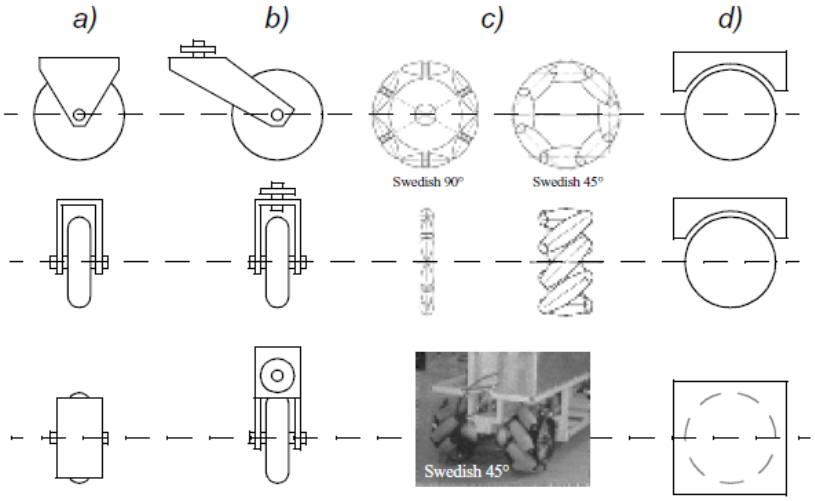


Figura 2.4: a) roda padrão, b) roda castor, c) roda sueca e d) roda esférica (NOURBAKHSH e SIEGWART, 2004).

Existem diversas configurações de disposição das rodas, conforme mostrado na Tabela 2.1, e cada uma delas influencia na estabilidade, mobilidade e na forma de controle do robô, sendo então sua escolha de primordial importância para o projeto do robô.

[4] Onidirecional: que envolve todas as direções; que se move ou funciona em todas as direções. Ref: Dicionário eletrônico Houaiss 3.0.

Tabela 2.1: Possíveis configurações de rodas para robôs móveis. Continua.

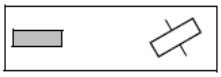
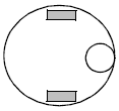
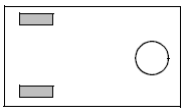
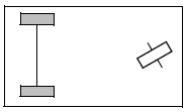
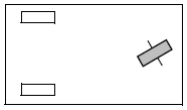
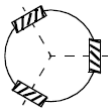
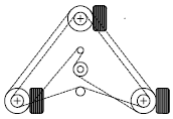
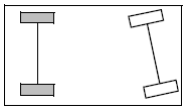
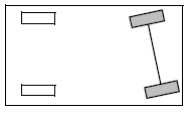
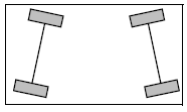
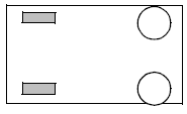
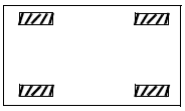
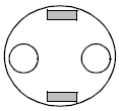
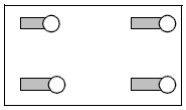
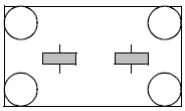
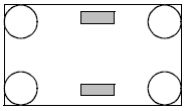
Quantidade de rodas	Configuração	Descrição
2		Uma roda padrão de tração atrás e uma roda direcional à frente.
		Duas rodas do tipo padrão de tração diferencial com o centro de massa abaixo do eixo.
3		Duas rodas do tipo padrão de tração diferencial, centralizadas com direção diferencial e uma roda onidirecional.
		Duas rodas do tipo padrão, de tração diferencial na traseira/dianteira e uma roda onidirecional no lado oposto.
		Duas rodas do tipo padrão de tração conectadas na traseira e uma roda padrão direcionada na frente.
		Duas rodas do tipo padrão, livres na traseira e uma roda padrão, tracionada e direcionada na dianteira.
		Três rodas suecas tracionadas dispostas em 120° uma da outra.
		Três rodas do tipo padrão direcionais com sincronia motorizada. Não é possível o controle da orientação.
4		Duas rodas do tipo padrão de tração conectadas na traseira e duas rodas do tipo padrão direcionais na dianteira.
		Duas rodas do tipo padrão de tração e direcionais conectadas na dianteira e duas rodas do tipo padrão livres na traseira.
		Quatro rodas do tipo padrão tracionadas e direcionais.
		Duas rodas do tipo padrão, de tração diferencial na traseira/dianteira e duas rodas onidirecionais no lado oposto.

Tabela 2.1: Possíveis configurações de rodas para robôs móveis. Conclusão.

Quantidade de rodas	Configuração	Descrição
4		Quatro rodas suecas tracionadas
		Duas rodas do tipo padrão, de tração diferencial e duas rodas onidirecionais.
		Quatro rodas do tipo castor de tração e direcionadas.
6		Duas rodas do tipo padrão de tração, direcionais e alinhadas no centro horizontal e quatro onidirecionais nas extremidades.
		Duas rodas do tipo padrão de tração diferencial alinhadas no centro e quatro onidirecionais nas extremidades.

Fonte: Adaptado de (NOURBAKHSH e SIEGWART, 2004).

2.3 CINEMÁTICA DE UM ROBÔ MÓVEL COM RODAS

O termo cinemática de um robô representa o estudo de seu movimento em um dado sistema de referência. Ele tem por interesse a descrição deste movimento espacial na forma de uma função do tempo como, por exemplo, um sistema de equações que forneça valores de X e Y no plano cartesiano, representando a posição, o ângulo referente à sua orientação.

Cabe ressaltar que, em geral, em vez de medir instantaneamente a posição de um robô móvel, integra-se a velocidade do robô ao longo do tempo. Porém todas as incertezas referentes às medições de velocidades são refletidas no cálculo de seu posicionamento (BARRIENTOS, PEÑÍN e BALAGUER, 1997).

Para se entender o movimento de um robô, deve-se entender a contribuição de cada roda no movimento, pois são elas as responsáveis pelo deslocamento do robô como um todo. Da mesma forma que a roda é responsável por gerar o movimento,ela também gera restrições como, por exemplo, o fator de escorregamento lateral e as próprias limitações impostas pelo seu tipo e tamanho (NOURBAKHSH e SIEGWART, 2004).

Partindo do princípio que o robô é um corpo rígido sobre rodas não deformáveis e que a superfície na qual está inserido é um plano horizontal, assume-se que a postura[5] do robô tem três dimensões, duas coordenadas que representam sua posição no referido plano e um ângulo que determina sua orientação. Sendo assim, pode-se representar a postura de um robô móvel pelo vetor

$$\xi = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix},$$

onde x representa a posição ao longo do eixo das abicissas, y representa a posição ao longo do eixo das ordenadas e ψ seu ângulo de orientação em relação ao eixo das abcissas.

Um modelo cinemático generalista que rege um veículo móvel uniciclo é dado pela equação (2.1), descrevendo assim a postura destes tipos de robôs. A postura do robô é mostrada na Figura 2.5 (WIT, SICILIANO e BASTIN, 1997).

$$\begin{cases} \dot{x} = v\cos\psi - a\omega\,\mathrm{sen}\,\psi \\ \dot{y} = v\,\mathrm{sen}\,\psi + a\omega\cos\psi \\ \dot{\psi} = \omega \end{cases}, \tag{2.1}$$

onde as variáveis de estados e as saídas, que determinam a pose do robô, são x, y e ψ e as entradas de controle são v e ω, velocidade linear e velocidade angular, respectivamente.

Adotando-se $a = 0$, ou seja, fazendo com que os graus de liberdade do robô sejam a translação e a rotação e que não dependam um do outro, reduz-se consideravelmente a complexidade da matemática envolvida na implementação de um controlador cinemático. A equação (2.2) mostra o sistema cinemático, dito reduzido.

$$\begin{cases} \dot{x} = v\cos\psi \\ \dot{y} = v\,\mathrm{sen}\,\psi \\ \dot{\psi} = \omega \end{cases} \tag{2.2}$$

[5] Postura: Nome dado ao conjunto das informações das coordenadas no qual está situado um robô e do ângulo que o mesmo faz com um eixo referencial.

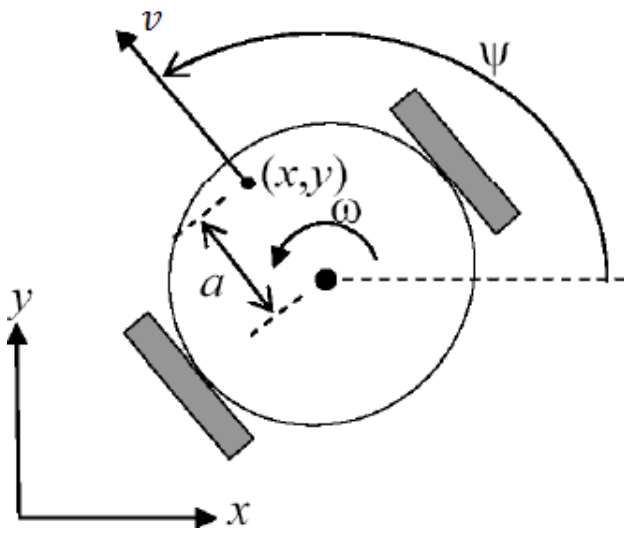


Figura 2.5: Postura da cadeira de rodas robótica. Adaptado de (CRUZ, 2009).

Este modelo cinemático é de coordenadas retangulares, podendo também ser na forma de coordenadas polares. A Figura 2.6 mostra os estados do robô em coordenadas polares e, a partir dela, pode-se chegar ao modelo cinemático polar

$$\begin{cases} \dot{e}_c = -v\cos\beta_u \\ \dot{\beta}_u = -\omega + v\frac{\operatorname{sen}\beta_u}{e_c} \\ \dot{\beta}_c = v\frac{\operatorname{sen}\beta_u}{e_c} \end{cases}, \tag{2.3}$$

onde x_g, y_g e β_c são as coordenadas e a orientação de referência, e_c é a distância entre a coordenada de origem e a coordenada de referência e β_u é o ângulo entre o vetor v e o vetor e_c.

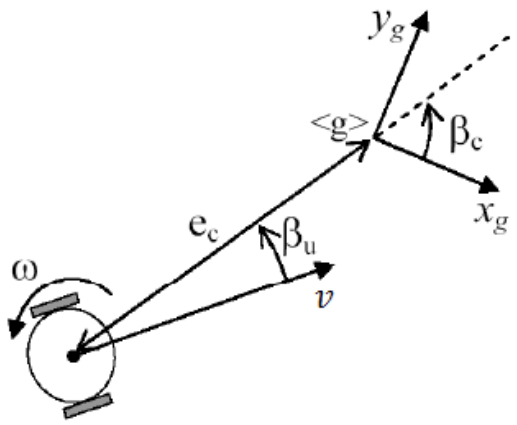


Figura 2.6: Coordenadas polares de um robô móvel. Adaptado de (CRUZ, 2009).

2.3.1 Cinemática da cadeira de rodas robótica

A Figura 2.7 aponta os principais efeitos cinemáticos e dinâmicos que atuam sobre o conjunto cadeira de rodas robótica mais seu usuário, podendo-se extrair então seu modelo cinemático e o seu modelo dinâmico, capítulo 2.4.1.

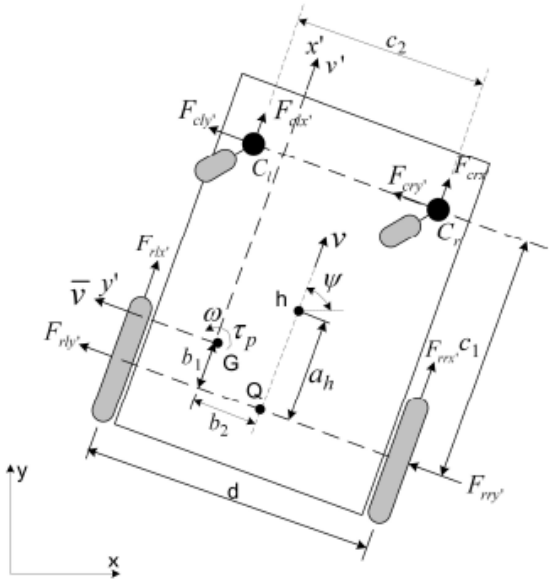


Figura 2.7: Efeitos cinemáticos e dinâmicos de uma cadeira de rodas robótica (CELESTE, 2009).

Na figura, G representa o centro de massa, deslocado lateralmente pelo fato de seu usuário inclinar-se sobre o encosto da cadeira de rodas, Q é o centro do eixo virtual entre as duas rodas, h é o ponto de referência, v' e $\overline{v}$ representam as velocidades longitudinal e lateral do centro de massa do sistema, v e ω são as velocidades linear e angular do sistema e ψ o ângulo da orientação de referência. As definições e explicações sobre as forças indicadas serão mencionadas no capítulo 2.4.1 (CELESTE, 2009) (CRUZ, 2009).

Segundo (CELESTE, 2009, p. 85) o comportamento cinemático do sistema cadeira de rodas robótica mais seu usuário é dado por

$$\begin{cases} \dot{x} = v\cos\psi - a_h\omega\,\mathrm{sen}\,\psi + \delta_x \\ \dot{y} = v\,\mathrm{sen}\,\psi + a_h\omega\cos\psi + \delta_y \\ \dot{\psi} = \omega \end{cases}, \tag{2.4}$$

sendo $\delta_x = -\overline{v}^s \operatorname{sen}\psi$, $\delta_y = \overline{v}^s \cos\psi$ componentes do distúrbio ocasionado pelo deslizamento lateral das rodas de tração, onde $\overline{v}^s$ é a velocidade de deslizamento lateral de tais rodas.

2.3.2 Controle de postura usando Lyapunov

Este tipo de controlador visa controlar a posição e a postura de um robô com rodas baseado em seu modelo cinemático polar. Quando o modelo cinemático é baseado em coordenadas retangulares e $a = 0$, este controlador se torna inviável com uma lei de controle de estados invariantes no tempo e, caso $a > 0$, seu projeto se torna demasiadamente complicado. Por conta disso, optou-se em mostrar o projeto de controle baseado no modelo cinemático com coordenadas polares (CRUZ, 2009, p. 4).

Este controlador tem como ponto de partida a mais simples estrutura da função candidata de Lyapunov:

$$V(e_c, \beta_u, \beta_c) = \frac{1}{2}\lambda_1 {e_c}^2 + \frac{1}{2}\lambda_2 {\beta_u}^2 + \frac{1}{2}\lambda_3 {\beta_c}^2; \quad \lambda_i > 0, \ \ i = 1,2,3$$

tendo como derivada,

$$\dot{V} = \lambda_1 e_c \dot{e_c} + \lambda_2 \beta_u \dot{\beta_u} + \lambda_3 \beta_c \dot{\beta_c} \, . \tag{2.5}$$

Substituindo as linhas de (2.3) em (2.5), tem-se:

$$\dot{V} = -\lambda_1 e_c v \cos\beta_u + \lambda_2 \beta_u \left(-\omega + v\frac{\operatorname{sen}\beta_u}{e_c}\right) + \lambda_3 \beta_c v \frac{\operatorname{sen}\beta_u}{e_c} \, . \tag{2.6}$$

Segundo (CRUZ, 2009, p. 5), escolhendo v e ω como sendo

$$\begin{cases} v = k_1 e_c \cos\beta_u, \quad k_1 > 0 \\ \omega = k_2\beta_u + k_1 \cos\beta_u \operatorname{sen}\beta_u \left(1 + \frac{\lambda_3\beta_c}{\lambda_2\beta_u}\right), \quad k_2 > 0 \end{cases} \tag{2.7}$$

e substituindo (2.7) em (2.6), tem-se

$$\dot{V} = -\lambda_1 k_1 {e_c}^2 \cos^2\beta_u - \lambda_2 k_2 {\beta_u}^2 \leq 0. \tag{2.8}$$

Com isso a equação (2.8) se mostra semi-definida negativa, implicando em um sistema com uma lei de controle assintoticamente estável em malha fechada quando $t \to \infty$.

Porém há um inconveniente em utilizar esta lei de controle, pois se faz necessária uma medida dos ângulos β_u e β_c, sendo os mesmos indefinidos quando $e_c = 0$. Com isso o robô nunca atingiria sua postura final exata, tendo de parar com uma postura próxima a verdadeira.

2.4 DINÂMICA DE UM ROBÔ MÓVEL COM RODAS

A dinâmica relaciona as forças de contato de um corpo, a aceleração e a trajetória na qual está submetido, assim, o modelo dinâmico de um robô proporciona o conhecimento da relação entre seu movimento e da atuação das forças aplicadas sobre ele (SICILIANO e KHATIB, 2008).

Pode-se chegar a um modelo dinâmico generalista de um robô com rodas, considerando que o centro de massa está localizado no centro do eixo virtual das rodas, e considerando ainda que não há deslizamento lateral das rodas de tração e nem existam forças externas atuando sobre o robô. A Figura 2.8 mostra a força atuante sobre o robô que segue as considerações supracitadas (CRUZ, 2009).

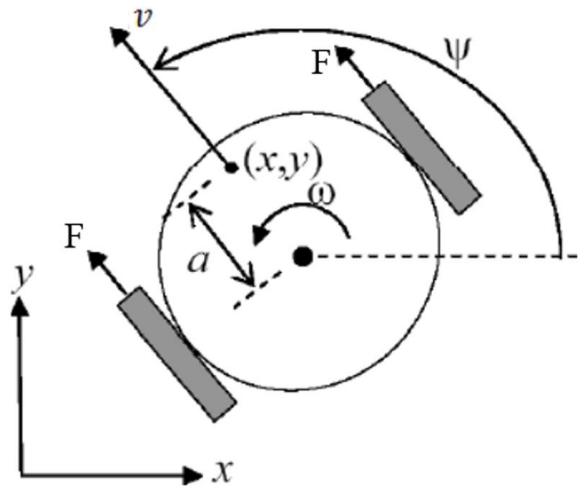


Figura 2.8: Força atuante em robô móvel. Adaptado de (CRUZ, 2009).

Neste caso, tem-se que

$$\begin{cases} F = m\ddot{x} \\ \tau = I_z\ddot{\psi} \end{cases},$$

sendo F e τ a força longitudinal e o torque proporcionados pelas rodas de tração, respectivamente, m é a massa do robô e I_z o momento de inércia do robô sobre o centro do eixo virtual das rodas de tração. Desta forma, chega-se no modelo dinâmico simplificado

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{m} & 0 \\ 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} F \\ \tau \end{bmatrix},$$

que utiliza entradas de força e torque (CRUZ, 2009).

2.4.1 Dinâmica da cadeira de rodas robótica

No caso de uma cadeira de rodas robótica, a dinâmica de seu comportamento é peculiar. Isso se dá pelo fato de haver uma carga pesada e móvel, uma pessoa, que pode vir a representar até 60% da massa do sistema cadeira de rodas robótica mais usuário. Devido este fato, é fundamental levar em consideração a presença do usuário no sistema dinâmico do robô. A Figura 2.7 mostra o esquema da cadeira de rodas, onde é levado em consideração seu usuário, explicitando as forças atuantes sobre ela. (CELESTE, 2009)

A figura mostra as forças longitudinais, $F_{rrx\prime}$ e $F_{rlx\prime}$, e as forças laterais, $F_{rry\prime}$ e $F_{rly\prime}$, que atuam sobre as rodas de tração, as forças longitudinais, $F_{crx\prime}$ e $F_{clx\prime}$, e as forças laterais, $F_{cry\prime}$ e $F_{cly\prime}$, que atuam sobre as rodas livres e forças e torque externos, $F_{ex\prime}$, $F_{ey\prime}$ e τ_p.

Segundo (CELESTE, 2009, p. 85), as equações de força e momento que descrevem o comportamento dinâmico do sistema são dadas por

$$\sum F_{x'} = 0 \rightarrow m(\dot{v}' - \overline{v}\omega) = F_{rrx'} + F_{rlx'} + F_{crx'} + F_{clx'} + F_{ex'},$$

$$\sum F_{y'} = 0 \rightarrow m\left(\dot{\overline{v}} + v'\omega\right) = F_{rry'} + F_{rly'} + F_{cry'} + F_{cly'} + F_{ey'},$$

$$\sum M_z = 0 \rightarrow I_z\dot{\omega} = \frac{d}{2}(F_{rrx'} - F_{rlx'}) + b_2(F_{rrx'} + F_{rlx'}) - b_1\left(F_{rry'} + F_{rly'}\right) + \tau_p,$$

onde m é a massa total do sistema e I_z o momento de inércia localizado em G.

Celeste (2009, p. 88) com base nesse equacionamento e levando em consideração o modelo dos motores, a dinâmica entre os motores e as rodas e os modelos dos controladores de baixo nível, descreve o modelo dinâmico completo da cadeira de rodas robótica como sendo

$$\begin{cases} v_{ref} = \theta_1\dot{v} + \theta_3(-\omega^2) + \theta_4 v + \theta_7(-\dot{\omega}) - \delta_v \\ \omega_{ref} = \theta_2\dot{\omega} + \theta_5 v\omega + \theta_6\omega + \theta_8(-\dot{v}) - \delta_\omega \end{cases}, \tag{2.9}$$

onde $\theta_i,\ i = 1, 2, \ldots, 8$ são parâmetros identificáveis que dependem unicamente de características físicas do sistema, identificáveis, e δ_v e δ_ω são perturbações inerentes a atuação de forças e torques externos e do deslizamento lateral das rodas de tração, não identificáveis.

O modelo dinâmico pode ser representado na forma matricial da seguinte forma,

$$\boldsymbol{T}(u, \dot{u})\boldsymbol{\theta} + \boldsymbol{\delta} = \boldsymbol{u_{ref}},$$

sendo

$$\boldsymbol{T} = \begin{bmatrix} \dot{v} & 0 & \omega^2 & v & 0 & 0 & -\dot{\omega} & 0 \\ 0 & \dot{\omega} & 0 & 0 & v\omega & \omega & 0 & -\dot{v} \end{bmatrix},\ \boldsymbol{\theta} = [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_8]^T,$$

$$\boldsymbol{\delta} = [-\delta_v \quad -\delta_\omega]^T,\ \boldsymbol{u} = [v \quad \omega]^T \text{ e } \boldsymbol{u_{ref}} = [v_{ref} \quad \omega_{ref}]^T.$$

Celeste (2009, p. 89) afirma que o modelo dinâmico matricial pode ser reduzido a

$$\boldsymbol{T\theta} = \boldsymbol{u_{ref}},$$

uma vez que o distúrbio δ é ignorado por não obedecer a uma relação conhecida de causa e efeito. Tal modelo assume então uma estrutura conhecida como modelo de regressão, podendo-se assim estimar os parâmetros da matriz $\boldsymbol{\theta}$.

Para se chegar à estimativa do parâmetro $\boldsymbol{\theta}$, (CELESTE, 2009) aplicou um filtro linear de primeira ordem e utilizou o método de estimação por mínimos quadrados, obtendo os dados da Tabela 2.2, representando o modelo sem usuário e com um usuário de aproximadamente 100 Kg.

Tabela 2.2: Parâmetros identificados

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8
Sem usuário	0,4042	0,1877	-0,0069	1,0261	0,0405	0,9239	-0,0304	-0,1162
Usuário pesado	0,3241	0,0120	0,0092	0,9969	0,0634	0,9898	-0,0562	-0,0002

Fonte: (CELESTE, 2009)

2.4.2 Controle dinâmico

Existem diversos tipos de leis de controle para o modelo dinâmico supracitado. Um exemplo relevante é proposto por (CELESTE, 2009, p. 96), que segue o diagrama mostrado na Figura 2.9, sendo definindo pela lei de controle

$$\boldsymbol{u}_{ref} = \boldsymbol{T}(\boldsymbol{u},\boldsymbol{\sigma})\widehat{\boldsymbol{\theta}} + \boldsymbol{K_r} sign(\tilde{\boldsymbol{u}}) \,,$$

onde

$$\boldsymbol{T}(\boldsymbol{u},\boldsymbol{\sigma}) = \begin{bmatrix} \sigma_v & 0 & \omega^2 & v & 0 & 0 & -\sigma_\omega & 0 \\ 0 & \sigma_\omega & 0 & 0 & v\omega & \omega & 0 & -\sigma_v \end{bmatrix} ,$$

σ_v e σ_ω são ações de controle que atuam no erro $\tilde{\boldsymbol{u}} = \boldsymbol{u}^c_{ref} - u$, $\boldsymbol{K}_r = diag(k_v, k_\omega) > 0$ é uma matriz de ganho proporcional e $\widehat{\boldsymbol{\theta}}$ é o vetor de parâmetros estimados considerando o vetor de erros paramétricos.

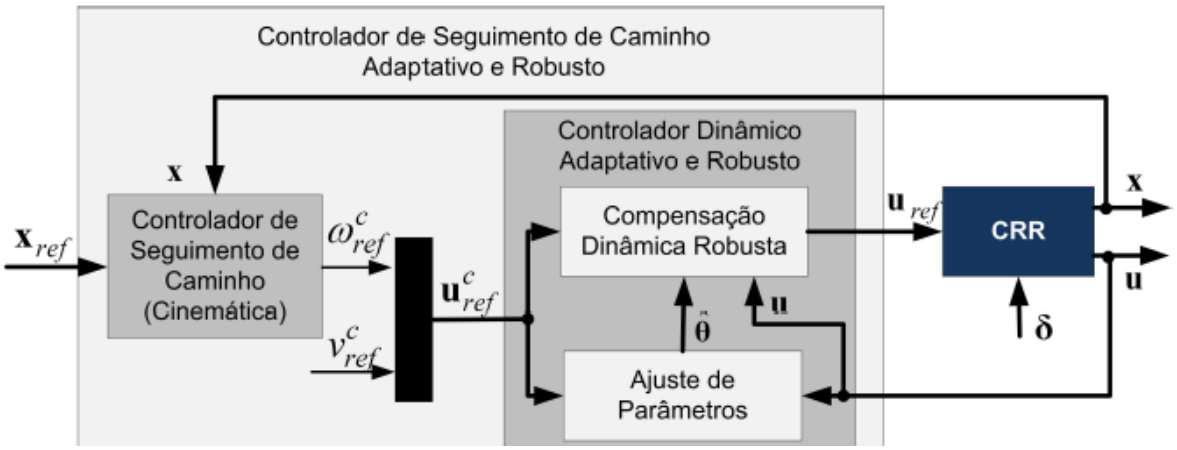


Figura 2.9: Diagrama de blocos da estrutura de controle (CELESTE, 2009).

Vale ressaltar que o sistema é estável, porem não garante que os erros paramétricos tendam a zero quando $t \to \infty$, fazendo com que os termos estimados convirjam para valores diferentes dos reais com a finalidade de fazer com que $\tilde{\boldsymbol{u}}$ convirja em um valor limitado conforme proposição 3.2 apresentada em (CELESTE, 2009, p. 98).

2.5 CONSIDERAÇÕES

Este capítulo mostrou, de forma superficial, diversos tipos de robôs com as mais variadas aplicabilidades, e de forma detalhada, o robô com rodas, mais especificamente uma cadeira de rodas robótica, foco do presente trabalho. Foi detalhado o modelo cinemático e o modelo dinâmico da cadeira de rodas robótica. Esses modelos agrupam as grandezas físicas que atuam interna e externamente na mesma, gerando as equações que descrevem seu comportamento no ambiente que o cerca.

O equacionamento dos modelos supracitados é imprescindível para o devido desenvolvimento de seus respectivos controladores, pois a criação dos mesmos parte da modelagem pressuposta, assunto explorado no capítulo 4.

Além da modelagem, foram mostradas algumas formas de controle comumente utilizados em robôs a rodas, como é o caso da lei de controle proposta por Lyapunov, que se baseia na redução contínua do erro e possui estabilização assintótica.

Capítulo 3: CONTROLE PREDITIVO

O controle preditivo data da década de 60, quando se deu o surgimento do controle ótimo. Nesse tipo de controle, visa-se a solução ótima de um determinado sistema que evolue no tempo com um horizonte temporal infinito. A solução de um controle ótimo dentro de um horizonte de previsão finito foi chamada de Controle Preditivo Baseado em Modelo, ou *MPC*, dando origem, assim, ao controle preditivo moderno. Outra diferença entre o controle ótimo e o preditivo é que no primeiro as ações de controle futuras são completamente aplicadas no processo, enquanto que no segundo somente a ação de controle referente ao instante atual é aplicada, sendo chamada também de controle por horizonte retrocedente (MAYNE, RAWLINGS, *et al.*, 2000).

Pouco tempo depois, no final da década de 70, surgiu o Controle por Matriz Dinâmica, ou *DMC*, ele foi desenvolvido por engenheiros da *Shell* (CUTLER e RAMAKER, 1979). Se baseia na resposta ao degrau, tendo como objetivo levar a saída para um valor o mais próximo possível de uma faixa de operação. O controle preditivo *DMC* é uma das técnicas de controle avançada mais utilizadas em aplicações industriais, principalmente o multivariável ou *MIMO* (múltiplas entradas e múltiplas saídas) (MIURA, 2003).

A partir de então, diversos controladores preditivos têm sido desenvolvidos, como o DMC quadrático (GARCIA e MORSHEDI, 1986), o Controle Preditivo Generalizado ou GPC (CLARKE, MOHTADI e TUFFS, 1987), o *GPC* baseado em espaço de estados (CLARKE e ORDYS, 1993), dentre outros (WANG, 2009).

Independente do algoritmo, os controladores preditivos têm em comum uma função custo pré-determinada, cujo objetivo é calcular valores para as variáveis manipuladas com o intuito de levar ou manter a saída do sistema o mais próximo possível do desejado. A função custo que se deseja minimizar para p entradas e q saídas é dada por:

$$J(k) = \frac{1}{2}\sum_{m=1}^{q}\sum_{j=1}^{h_p}\delta_m[\hat{y}_m(k+j) - w_m(k+j)]^2 + \sum_{i=1}^{p}\sum_{j=1}^{h_c}\lambda_i[\Delta u_i(k+j-1)]^2 \quad (3.1)$$

onde $\delta_m,\ m = 1 \cdots q$, $\lambda_i,\ i = 1 \cdots p$ são parâmetros de ajustes do controlador.

Outros itens em comum são o Horizonte de previsão, h_p, intervalo de tempo futuro onde será considerado o comportamento da saída do sistema e o Horizonte de Controle, h_c, intervalo de

tempo futuro, normalmente menor do que o do horizonte de previsão, que representa as ações de controle consideradas. A Figura 3.1 ilustra tais conceitos (PALÚ, 2001).

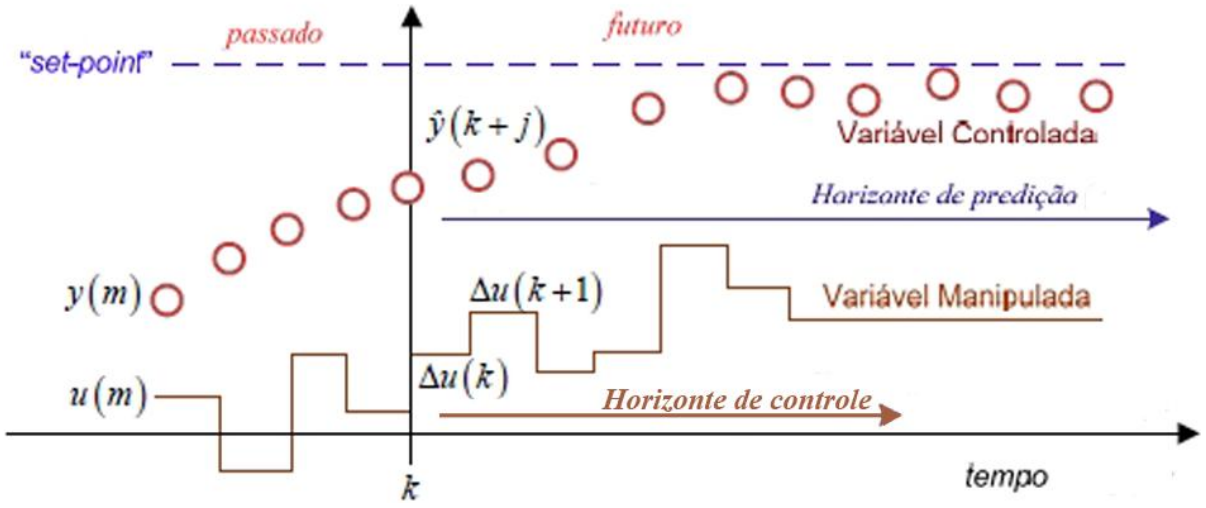


Figura 3.1: Esquema da representação da ação de um controlador preditivo. Adaptado de (PALÚ, 2001).

Na Figura 3.1, $y(m)$ e $u(m)$ representam as saídas e entradas passadas, $\Delta u(k)$ representa a entrada no instante atual e, $\hat{y}(k+j)$ e $\Delta u(k+1)$ representam a saída e a entrada futura.

3.1 DMC MULTIVARIÁVEL

O *DMC-MIMO* é considerado o algoritmo de controle mais utilizado na indústria por possuir uma dinâmica de convolução simples, relacionando as entradas com as saídas, podendo-se assim considerar a influência de uma entrada específica sobre todas as saídas do sistema (CAMACHO e BORDONS, 2004).

O modelo de convolução de um sistema *DMC-MIMO* para p entradas e q saídas é dado por:

$$y_m(k) = \sum_{l=1}^{p} \sum_{i=1}^{\infty} g_{ml}(i) \Delta u_l(k-i)$$

sendo,

$$l = 1, \cdots, p \text{ e } m = 1, \cdots, q$$

A resposta ao degrau da saída m relacionada à l é representada por g_{ml} e a variação no sinal de controle representada por $\Delta u_l(k) = u_l(k) - u_l(k-1)$.

Considerando que a predição de j passos a frente da saída m a partir do instante atual k é dada por $\hat{y}_m(k+j)$ e que $\Delta u_l(k+j) = 0$, $1 \leq j < h_c$ para $\Delta u_l(k) = 0$ onde $k > N_m^l$ e N_m^l um número inteiro, tem-se a expressão (CAMACHO e BORDONS, 2004)

$$\hat{y}_m(k+j) = \sum_{l=1}^{p} \sum_{i=\max\{1, j-h_c-1\}}^{N_m^l} g_{ml}(i)\Delta u_l(j+k-i)$$

Definindo:

$$G_m(k) = [g_{m1}(k) \quad g_{m2}(k) \quad \cdots \quad g_{mp}(k)]_{1xp} ,$$

$$\Delta U(k) = [\Delta u_1(k) \quad \Delta u_2(k) \quad \cdots \quad \Delta u_p(k)]_{px1}^T ,$$

$$f_m(k+j) = \sum_{i=j+1}^{N} G_m(i)\Delta U(j+k-i) ,$$

$$\hat{Y}(k) = [\hat{y}_1 \quad \hat{y}_2 \quad \cdots \quad \hat{y}_q]_{qx1}^T ,$$

$$F(k) = [f_1(k) \quad f_2(k) \quad \cdots \quad f_q(k)]_{qx1}^T ,$$

$$G(k) = [G_1(k) \quad G_2(k) \quad \cdots \quad G_q(k)]_{qxp}^T .$$

Obtêm-se a expressão com as ações de controle passadas separadas das ações de controle presente e futuras definida por:

$$\hat{Y}(k+j) = \sum_{i=\max\{1, j-h_c-1\}}^{j-h_c} G(i)\Delta U(k+h_c-i) + \sum_{i=j-h_c+1}^{j} G(i)\Delta U(k+j-i) + F(k+j)$$

O termo $\sum_{i=\max\{1, j-h_c-1\}}^{j-h_c} G(i)\Delta U(k+h_c-i)$ é a resposta forçada, contendo as ações de controle presente e futuras e o termo $\sum_{i=j-h_c+1}^{j} G(i)\Delta U(k+j-i) + F(k+j)$ representa a resposta livre com as ações de controle passadas.

3.2 *GPC* MULTIVARIÁVEL

O *GPC-MIMO* é um controlador preditivo baseado em modelo, mais especificamente o modelo CARIMA, (Controlador Auto Regressivo Integrador com Média Móvel). Esse controlador foi concebido com o intuito de suprir algumas deficiências dos controladores preditivos anteriores, como o controle de plantas instáveis, variantes no tempo, com polos na origem e com fase não mínima (CAMACHO e BORDONS, 2004).

Para Camacho e Bordons (2004), um sistema *MIMO* para p entradas e q saídas é dado por:

$$A_i(Z^{-1})y_i(k) = Z^{-d}B_{i1}(Z^{-1})u_1(k+1) + Z^{-d}B_{i2}(Z^{-1})u_2(k+1) + \cdots + Z^{-d}B_{ip}(Z^{-1})u_3(k+1)\,,$$

sendo $i = 1, \cdots, q$ e d o atraso.

Sejam os polinômios $E_j^i(Z^{-1})$ e $F_j^i(Z^{-1})$, unicamente definidos com graus $j-1$ e na respectivamente, pertencentes à equação Diophantina

$$1 = E_j^i(Z^{-1})\tilde{A}_i(Z^{-1}) + Z^{-1}F_j^i(Z^{-1})\,.$$

Definem-se os seguintes polinômios:

$$G_j^{i,m}(Z^{-1}) = E_j^i(Z^{-1})B_m(Z^{-1}) = g_0^{i,m} + g_1^{i,m}Z^{-1} + g_2^{i,m}Z^{-2} + \cdots + g_{nb+j-1}^{i,m}Z^{-(nb+j-1)}$$

$$H_j^{i,m}(Z^{-1}) = g_0^{i,m} + g_1^{i,m}Z^{-1} + g_2^{i,m}Z^{-2} + \cdots + g_{j-d-1}^{i,m}Z^{-(j-d-1)} \quad ,$$

$$\overline{H}_j^{i,m}(Z^{-1}) = G_j^{i,m}(Z^{-1}) - H_j^{i,m}(Z^{-1}) = g_{j-d}^{i,m}Z^{-(j-d)} + \cdots + g_{nb+j-1}^{i,m}Z^{-(nb+j-1)}$$

onde $i = 1 \cdots q$ e $m = 1 \cdots p$. Sendo assim, as saídas previstas j passos a frente são representadas pelo seguinte equacionamento:

$$\begin{aligned}\hat{y}_i(k+j) = \; & H_j^{i,1}\Delta u_1(k+j-d-1) + H_j^{i,2}\Delta u_2(k+j-d-1) + \cdots + \\ & H_j^{i,p}\Delta u_p(k+j-d-1) + \overline{H}_j^{i,1}\Delta u_1(k-1) + \cdots + \\ & \overline{H}_j^{i,1}\Delta u_p(k-1) + F_j^i y_i(k)\end{aligned}$$

Definindo os vetores:

$$\Delta U(k) = [\Delta u_1 \quad \Delta u_2 \quad \cdots \quad \Delta u_p]^T_{px1},$$

$$H_j^i = \left[H_j^{i,1} \quad H_j^{i,2} \quad \cdots \quad H_j^{i,p}\right]_{1xp},$$

$$H_j = \left[H_j^1 \quad H_j^2 \quad \cdots \quad H_j^p\right]^T_{1xp},$$

$$\overline{H}_j^i = \left[\overline{H}_j^{i,1} \quad \overline{H}_j^{i,2} \quad \cdots \quad \overline{H}_j^{i,p}\right]_{1xp},$$

$$\overline{H}_j = \left[\overline{H}_j^1 \quad \overline{H}_j^2 \quad \cdots \quad \overline{H}_j^p\right]^T_{1xp},$$

$$F_j y(k) = \left[F_j^1 y_1 \quad F_j^2 y_2 \quad \cdots \quad F_j^q y_q\right]^T_{qx1},$$

$$\hat{Y}(k) = [y_1(k) \quad y_2(k) \quad \cdots \quad y_q(k)]^T_{qx1}.$$

Pode-se escrever as saídas previstas j passos à frente na forma matricial representada por:

$$\hat{Y}(k+j) = H_j \Delta U(k+j-d-1) + \overline{H}_j \Delta U(k-1) + F_j y(k)$$

onde o termo $H_j \Delta U(k+j-d-1)$ representa a resposta forçada e $\overline{H}_j \Delta U(k-1) + F_j y(k)$ representa a resposta livre.

3.3 CONTROLADOR PREDITIVO BASEADO EM ESPAÇO DE ESTADOS

Este tipo de controlador preditivo utiliza o modelo em espaço de estados para realizar as previsões para p entradas e q saídas onde $q \leq p$, (WANG, 2009).

A formulação do modelo *MIMO* em espaço de estados é

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) + B_m w(k) \\ y(k) &= C_m x_m(k) \end{aligned},$$

onde $u(k)$ é o vetor de entradas $(p \times 1)$, $y(k)$ é o vetor de saída $(q \times 1)$, $x_m(k)$ são os estados e $w(k)$ é uma perturbação que afeta a entrada do processo. Essa entrada é assumida como uma sequência de ruído branco integrada, ou seja, satisfaz a equação

$$w(k) - w(k-1) = \varepsilon(k)\,,$$

onde $\varepsilon(k)$ é um ruído branco com média zero (WANG, 2009).

Sendo $\Delta x_m = x_m(k) - x_m(k-1)$ e $\Delta u(k) = u(k) - u(k-1)$, chega-se a seguinte expressão:

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k) + B_m \varepsilon(k)\,.$$

Fazendo $y(k)$ em função de Δx_m chega-se em

$$\Delta y(k+1) = C_m \Delta x_m(k) = C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k) + C_m B_m \varepsilon(k)\,,$$

onde $\Delta y(k+1) \triangleq y(k+1) - y(k)$.

Usando como novas variáveis de estados o vetor $x(k) = [\Delta x_m(k)^T \quad y(k)^T]^T$ obtém-se o modelo em espaço de estados para saídas e estados futuros dado por:

$$\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} A_m & o_m^T \\ C_m A_m & I_{qxq} \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} \Delta u(k) + \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} \varepsilon(k)$$

$$y(k) = [o_m \quad I_{qxq}] \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}$$

,

onde o_m é uma matriz de zeros.

Segundo Wang (2009), a expressão simplificada das previsões de estados e saídas e a solução do controlador *j* passos a frente são dados por:

$$\hat{Y} = \mathrm{G}\Delta u + f,$$

onde,

$$f = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{hp} \end{bmatrix} \mathrm{x(k)}, \quad \mathrm{G} = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{hp-1}B & CA^{hp-2}B & CA^{hp-3}B & \cdots & CA^{hp-hc}B \end{bmatrix},$$

$$\hat{Y} = \begin{bmatrix} \hat{y}(k) \\ \hat{y}(k+1) \\ \vdots \\ \hat{y}(k+h_p) \end{bmatrix} \; e \; \Delta u = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+h_c-1) \end{bmatrix}.$$

A expressão da função custo (3.1) na forma matricial é igual a

$$J(\Delta u) = (G\Delta u + f - w)^T Q_y (G\Delta u + f - w) + \Delta u^T Q_u \Delta u \;\; ,$$

onde

$$Q_y = \begin{bmatrix} \delta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \delta_q \end{bmatrix} \quad Q_u = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix}.$$

Após a multiplicação dos termos da expressão acima obtém-se

$$J(\Delta u) = \frac{1}{2} \Delta u^T \Phi \Delta u + c^T \Delta u + d \;\; ,$$

onde

$$\Phi = \left(G^T Q_y G + Q_u\right),$$

$$C^T = -2(w-f)^T G \;\; ,$$

$$d = (f-w)^T Q_y (f-w) \, .$$

Como processos reais estão submetidos às limitações de equipamentos, instrumentos, sinais, dentre outros, restrições no sinal de controle $u(t)$, na saída $y(t)$ e na variação do sinal de controle $\Delta u(t)$ devem ser introduzidas no processo de sintonia do controlador. Com isso, a função custo deve ser minimizada levando em consideração essas restrições (CAMACHO e BORDONS, 2004).

O novo problema deve ser solucionado a partir de

$$\min_{\Delta u} = J(\Delta u) \quad \boldsymbol{s.a.} \; \{S\Delta u \leq c \;\; ,$$

onde $\{S\Delta u \leq c$ corresponde ao conjunto de todas as restrições envolvidas no sistema, representadas por

$$\Delta u_{min} \leq \Delta u(\mathrm{k}+j) \leq \Delta u_{max}\ , j = 1 \cdots h_c - 1;$$

$$u_{min} \leq u(\mathrm{k}+j) \leq u_{max}\ , j = 1 \cdots h_c - 1,$$

$$y_{min} \leq \hat{y}(\mathrm{k}+j) \leq y_{max}\ , j = 1 \cdots h_p.$$

Para que seja possível realizar a minimização da função custo, todas as restrições devem ser em função de Δu, isto é

$$\underbrace{\begin{bmatrix} I_{phc} \\ -I_{phc} \end{bmatrix}_{2\mathrm{p}h_c \times ph_c} \Delta\mathrm{U} \leq \begin{bmatrix} \Gamma_{phc}\Delta\mathrm{U}_{max} \\ -\Gamma_{phc}\Delta\mathrm{U}_{min} \end{bmatrix}_{2\mathrm{p}h_c \times 1}}_{Restrições\ em\ \Delta u},$$

$$\underbrace{\begin{bmatrix} T \\ -T \end{bmatrix}_{2\mathrm{p}h_c \times ph_c} \Delta\mathrm{U} \leq \begin{bmatrix} \Gamma_{ph_c}\big(U_{max} - \mathrm{U}(t-1)\big) \\ -\Gamma_{ph_c}\big(U_{min} + U(t-1)\big) \end{bmatrix}_{2\mathrm{p}h_c \times 1}}_{Restrições\ em\ u},$$

$$\underbrace{\begin{bmatrix} G \\ -G \end{bmatrix}_{2\mathrm{m}h_p \times ph_c} \Delta\mathrm{U} \leq \begin{bmatrix} \Gamma_{h_p}Y_{max} - f \\ -\Gamma_{h_p}Y_{min} + f \end{bmatrix}_{2\mathrm{m}h_p X1}}_{Restrições\ em\ y},$$

onde

$$\Gamma_{phc} = [I_p \quad I_p \quad \cdots \quad I_p]^T_{ph_c \times 1}\ ,$$

$$T = \begin{bmatrix} I_p & O & O & \cdots & O \\ I_p & I_p & O & \cdots & O \\ I_p & I_p & I_p & \cdots & O \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_p & I_p & I_p & \cdots & I_p \end{bmatrix}_{ph_c \times ph_c},$$

$$\Gamma_{mh_p} = [I_p \quad I_p \quad \cdots \quad I_p]^T_{mh_p \times \mathrm{m}}\ .$$

Nas equações acima, I_p é uma matriz identidade com dimensão $p \times p$; I_{phc} é uma matriz identidade com dimensão $phc \times phc$; O é uma matriz de zeros com dimensão $p \times p$; $\mathrm{U}(t-1)$, $\Delta\mathrm{U}_{min}$ e $\Delta\mathrm{U}_{max}$ são vetores com dimensão $p \times 1$ e; Y_{max}, Y_{min} e f são vetores com dimensão $h_p \times 1$.

Com isso, é possível montar o conjunto de restrições $\{S\Delta u \leq c$ como sendo:

$$\begin{bmatrix} I_{phc} \\ -I_{phc} \\ T \\ -T \\ G \\ -G \end{bmatrix}_{(4\mathrm{p}h_c+2\mathrm{m}h_p)\times ph_c} \Delta\mathrm{U} \leq \begin{bmatrix} \Gamma_{phc}\Delta\mathrm{U}_{max} \\ -\Gamma_{phc}\Delta\mathrm{U}_{min} \\ \Gamma_{ph_c}\left(U_{max} - \mathrm{U}(t-1)\right) \\ -\Gamma_{ph_c}\left(U_{min} + U(t-1)\right) \\ \Gamma_{h_p}Y_{max} - f \\ -\Gamma_{h_p}Y_{min} + f \end{bmatrix}_{(4\mathrm{p}h_c+2\mathrm{m}h_p)\times 1}$$

A utilização da programação quadrática como algoritmo de otimização para o problema de controle é aplicada a sistemas com dinâmica lenta. Isso ocorre devido a grande quantidade de cálculos necessário para realizar a otimização e devido ao fato desses cálculos terem de ser repetidos em todas as iterações, tornando-se inviável sua aplicação em sistemas com dinâmica rápida (CAMACHO e BORDONS, 2004).

Uma alternativa para realizar a otimização desse tipo de sistema é o método de Programação Multiparamétrica, PM, onde todos os cálculos da solução do controlador são realizados *off-line*, ou seja, antes de se inicializar o sistema.

3.4 OTIMIZIÇÃO POR PROGRAMAÇÃO MULTIPARAMÉTRICA

A PM pode ser subdivida em 3 problemas de otimização, a Programação Multiparamétrica Linear, PML; a Programação Multiparamétrica Quadrática, PMQ; e a Programação Multiparamétrica Linear Inteira, PMLI, definidas por:

$$PML \quad \begin{cases} \min_v J(x) = c^T v + d^T x \\ s.a.: Av \leq b + Ex \end{cases}, \tag{3.2a}$$

$$PMQ \quad \begin{cases} \min_v J(x) = \frac{1}{2} v^T G v + v^T F v \\ s.a.: Av \leq b + Ex \end{cases}, \tag{3.2b}$$

$$PMLI \begin{cases} \min_v J(x) = \frac{1}{2} v^T G v + v^T F v \\ s.a.: A_r v_r + A_b v_b \leq b + Ex \\ v_r \in \mathbb{R} \ , \ \ v_b \in \{0,1\}^{n_b} \end{cases}, \tag{3.2c}$$

onde $x \in \mathbb{R}^p$, $v \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $E \in \mathbb{R}^{m\times p}$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}^p$ (KVASNICA, 2009).

O objetivo da PM é determinar uma solução ótima, definida por $v^*(x)$, como sendo uma função explícita em relação ao parâmetro x. Para aplicação em controle, o parâmetro x representa as variáveis de estados do sistema dinâmico, calculada a cada instante, e $v^*(x)$ representa as ações de controle. Assim, partindo da premissa que as soluções ótimas e a equação de estados do sistema já são conhecidas, o procedimento *online* a ser feito se resumiria em localizar a ação de controle específica para um dado estado naquele instante, reduzindo significativamente o tempo computacional (KVASNICA, 2009).

3.4.1 Solução do Problema de Programação Multiparamétrica

Para se obter uma melhor compreensão de como a solução explícita é obtida, algumas definições e teoremas devem ser abordados.

Definição 3.1 **(Conjunto Factível**) (KVASNICA, 2009): Define-se um conjunto factível $x \subset \mathbb{R}^p$como o conjunto de parâmetros $x \in \mathbb{R}^p$ para o qual o problema de otimização (3.2) é factível, ou seja:

$$\mathcal{X} = \{x \in \mathbb{R}^p | Av \leq b + Ex\}$$

Definição 3.2 **(Partição Politópica)** (KVASNICA, 2009): Uma coleção de conjuntos politópicos $\{\mathcal{P}_r\}_{r=1}^R = \{\mathcal{P}_1, \cdots, \mathcal{P}_R\}$ é uma partição politópica de um conjunto politópico $\mathcal{X}$ se,

$$\begin{cases} (i)\ U_{r=1}^R \mathcal{P}_r = x \\ (ii)\ (\mathcal{P}_r / \partial\mathcal{P}_r) \cap \left(\mathcal{P}_q / \partial\mathcal{P}_q\right) = \emptyset,\ \ \forall r \neq q \end{cases},$$

onde ∂ significa a fronteira.

Definição 3.3 **(Função Afim Por Partes (*PWA*))** (KVASNICA, 2009): Uma função $f: \mathcal{X} \to \mathbb{R}^d$com $d \in \mathbb{N}_+$ é afim por partes (*PWA*), se a partição $\{\mathcal{P}_r\}_{r=1}^R$ de um conjunto $\mathcal{X}$ existir, tal que $f(x) = L_r x + C_r$ se $x \in \mathcal{P}_r$.

Definição 3.4 (**Função Quadrática Por Partes (*PWQ*)**) (KVASNICA, 2009): Uma função $f: \mathcal{X} \to \mathbb{R}$ é quadrática por partes (PWQ), se a partição $\{\mathcal{P}_r\}_{r=1}^R$ de um conjunto $\mathcal{X}$ existir, tal que $f(x) = x^T Q_r x + L_r x + C_r$ se $x \in \mathcal{P}_r$.

Teorema 3.1 (BEMPORAD, MORARI, *et al.*, 2002), (BORRELLI, 2003): Considere o Problema de Programação Multiparamétrica Quadrática (PMQ) (3.2). Então:

1 – O conjunto de parâmetros factíveis $\mathcal{X}$ é um poliedro convexo;

2 – O conjunto $\mathcal{X}$ é particionado em $\mathcal{R} \in \mathbb{N}_+$ regiões poliedrais, i.e.:

$$\mathcal{P}_r = \{x \in \mathbb{R}^p | H_r x \leq K_r\}, \;\; r = 1, \cdots R$$

3 – A solução ótima $v^*: \mathcal{X} \rightarrow \mathbb{R}^n$ é uma função contínua e afim por partes (PWA) em relação ao parâmetro $\mathcal{X}$, i.e.

$$v^*(x) = L_r x + C_r \text{ se } x \in \mathcal{P}_r$$

4– A função custo ótima $J^*: \mathcal{X} \rightarrow \mathbb{R}$ é contínua, convexa e quadrática por partes, i.e.

$$J^*(x) = x^T Q_r x + L_r x + C_r \text{ se } x \in \mathcal{P}_r$$

Teorema 3.2 (BORRELLI, 2003): Considere o Problema de Programação Multiparamétrica Linear e Inteira (PMLI) (3.2c). Então o conjunto de parâmetros factíveis $\mathcal{X}$ é um poliedro particionado em $\mathcal{R} \in \mathbb{N}_+$ regiões poliedrais, onde existe uma solução ótima $v^*: \mathcal{X} \rightarrow \mathbb{R}^n$ afim por partes (PWA) em relação ao parâmetro $\mathcal{X}$ e, a função custo ótima $J^*: \mathcal{X} \rightarrow \mathbb{R}$ é *PWA*.

A grande vantagem das propriedades apresentadas nos Teoremas acima é que a solução ótima é definida avaliando a função *PWA* para um valor conhecido de x, o que envolve os seguintes passos:

1 – Identificar a região $\mathcal{P}_r$ que contém x;

2 – Avaliar a respectiva função afim $L_r x + C_r$.

O total de cálculos necessários para realizar as operações nos passos 1 e 2 é de no máximo igual a R, onde R é o total de partições de um conjunto $\mathcal{X}$. Portanto, a complexidade computacional do algoritmo PM está relacionada com os cálculos *off-line* da solução ótima v^* e da capacidade de armazenamento da solução. A complexidade *off-line* é igual a $\mathcal{O}\left(R\mathcal{O}(PQ_{n,m}) + R(n_f + m)\mathcal{O}(PL_{p,m})\right)$, onde $\mathcal{O}(PL_{n,m})$ é o máximo de operações numéricas para resolver o algoritmo de Programação Quadrática PQ com n variáveis e m restrições, $\mathcal{O}(PL_{p,m})$ o máximo de operações numéricas para resolver o algoritmo de Programação Linear PL com p variáveis e m restrições, e $n_f = (\sum_r f_r)/R$, sendo que f_r é o número de restrições da r-ésima região (KVASNICA, 2009).

3.4.2 Sistemas dinâmicos afins por partes

Seja um conjunto de funções lineares $f_{LIN,i}(x,u), i = 1,\cdots,n_L$, onde n_L é o total de pontos de linearização, obtidas através da expansão da série de Taylor de uma função não linear f em torno do ponto $x^{s,i},\ u^{s,i}$, isto é:

$$f_{LIN,i}(x,u) = f\left(x^{s,i},u^{s,i}\right) + \left(\frac{\partial f(x,u)}{\partial x}\right)_{x^{s,i},u^{s,i}}\left(x - x^{s,i}\right)\left(\frac{\partial f(x,u)}{\partial u}\right)_{x^{s,i},u^{s,i}}\left(u - u^{s,i}\right),$$

$$\mathrm{i} = 1,\cdots,\mathrm{n}_L\ .$$

Seja $x(t) \in \mathbb{R}^n$ o vetor de estado no instante t, $u(t) \in \mathbb{R}^m$ o vetor de controle. Assume-se que os estados e as entradas são sujeitos a restrições: $x(t) \in \mathbb{X} \subset \mathbb{R}^n, u(t) \in \mathbb{U} \subset \mathbb{R}^m$. O sistema dinâmico afim por partes (*PWA*) é definido pela equação de estados:

$$x(t+1) = f_{LIN,i}(x,u)\ se\ \begin{bmatrix}x\\u\end{bmatrix} \in \mathcal{D}_i, i = 1,\cdots,n_L\ , \qquad (3.3)$$

onde, $\mathcal{D}_i \subseteq \mathbb{X} \times \mathbb{U}$ (KVASNICA, 2009).

Definição 3.5 (BEMPORAD e MORARI, 1999): O sistema dinâmico *PWA* (3.3) está bem estruturado se o seu domínio $\mathcal{D} \subseteq \mathbb{X} \times \mathbb{U}$ pode ser particionado em $\mathcal{D}_i$ regiões tais que, $\mathcal{D} = \cup_i \mathcal{D}_i$ e as regiões não se cruzam, i.e, $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset, \forall j \neq i$

Um sistema *PWA* bem estruturado, quer dizer que a sua dinâmica é determinada unicamente para um dado par (x,u). Tais sistemas pertencem à classe de sistemas lineares híbridos, constituídos de variáveis lógicas e de dinâmicas lineares no tempo.

3.4.3 Controle Ótimo no Tempo Finito com Restrições (COTFR)

Considere o sistema dinâmico discreto no tempo representado pela equação de estados:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned}\ .$$

Assuma que este sistema seja controlável e observável, e que existam os conjuntos limitados $\mathbb{X} = \{x | H^x x \leq K^x\}, \mathbb{U} = \{u | H^u u \leq K^u\}$ e $\mathcal{T}_{set} = \{x | Lx \leq M\}$. Sejam h_p e h_c números inteiros positivos, então o problema COTFR é definido através da seguinte formulação:

$$J_{h_p}^*(x_0) = \min_{U_{h_c}} \ell_{h_p}\left(x_{h_p}\right) + \sum_{k=0}^{h_p-1} \ell(x_k, u_k) \,, \tag{3.4a}$$

$sa.$

$$x_{k+1} = Ax_k + Bu_k \,, \tag{3.4b}$$

$$x_k \in \mathbb{X}, \ \ \forall k = 0, \cdots, h_p \,, \tag{3.4c}$$

$$u_k \in \mathbb{U}, \ \ \forall k = 0, \cdots, h_c - 1 \,, \tag{3.4d}$$

$$x_{h_p} \in \mathcal{T}_{set} \ . \tag{3.4e}$$

Onde J^* é o custo ótimo que depende da condição inicial x_0, $U_{h_c} = \left(u_0, \cdots, u_{h_c}\right)^T$ é a sequência de controle ótimo a ser determinada, x_{h_p} é o estado final da trajetória, $\ell_{h_p}\left(x_{h_p}\right)$ é a função custo final, $\ell(x_k, u_k)$ é o custo de cada estágio k, e $\mathcal{T}_{set}$ representa a restrição do estado final. Para qualquer sistema *PWA*, a estabilidade é garantida se um conjunto invariante é imposto como uma restrição de estado final. Quando a norma é quadrática, é possível garantir a estabilidade desde que $\ell_{h_p}\left(x_{h_p}\right) = {x_{h_p}}^T P x_{h_p}$, $u_k = Kx_k$, $h_c \leq K \leq h_p$. Onde K representa o ganho de realimentação estabilizante e P é a solução da equação algébrica de Riccati do problema de controle ótimo com horizonte infinito. Neste caso, a solução do problema COTFR se aproxima do controle ótimo com horizonte de tempo infinito, garantindo estabilidade conforme mostra (BEMPORAD, MORARI, *et al.*, 2002).

O problema COTFR pode ser classificado como:

Problema do Regulador - quando o objetivo é manter a trajetória de estado x_k o mais próximo possível da origem, ou seja, minimizar a função custo dada por

$$\ell_{h_p}\left(x_{h_p}\right) = \left\| Q_{h_p} x_{h_p} \right\|_P \,,$$
$$\ell(x_k, u_k) = \|Q_x x_k\|_P + \|Q_u u_k\|_P \,,$$

onde $Q_{h_p} \in \mathbb{R}^{n \times n}, Q_x \in \mathbb{R}^{n \times n}$ e $Q_u \in \mathbb{R}^{m \times m}$ são matrizes de ponderações usadas para sintonizar o desempenho do controlador. Além disto $\|\cdot\|_P$ é tal que $\|Pz\|_1 = \sum_i |P_i z_i| \,, \|z\|_2 = z^T P z$ e $\|Pz\|_\infty = max_i |P_i z_i|$ para algum vetor z e matriz P.

Problema do Regulador ou Seguidor - quando o objetivo é manter a trajetória de estado x_k o mais próximo possível da referência x_{REF}, ou seja, minimizar a função custo dada por:

$$\begin{aligned} \ell_{h_p}\left(x_{h_p}\right) &= \left\| Q_{h_p}\left(x_{h_p} - x_{REF}\right)\right\|_P , \\ \ell(x_k, u_k) &= \left\| Q_x\left(x_{h_p} - x_{REF}\right)\right\|_P + \|Q_u \Delta u_k\|_P , \end{aligned} \tag{3.5}$$

onde, $\Delta u_k = u_k - u_{k-1}$.

Problema do Tempo Mínimo: quando o objetivo é obter a sequência de controles ótimos U_N^* que leva o estado de uma condição inicial x_0 para o estado terminal $\mathcal{T}_{set}$ no menor tempo possível, enquanto que as restrições do sistema são atendidas.

O problema COTFR pode ser formulado como um problema de Programação Multiparamétrica Quadrática (PMQ) ou linear (PML), dependendo do valor da norma *P* em (3.4). Na formulação do PMQ, temos $P = 2, \ell_{h_p}\left(x_{h_p}\right) = x_{h_p}^T Q_{h_p} x_{h_p}$ e $\ell(x_k, u_k) = x_k^T Q_x x_k + u_k^T Q_u u_k$. Assim, definindo as matrizes

$$\tilde{A} = \begin{bmatrix} I \\ A \\ \vdots \\ A^{h_p} \end{bmatrix} \quad \tilde{B} = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A^{h_p-1}B & A^{h_p-2}B & \cdots & 0 \end{bmatrix}.$$

O problema (3.4) para $P = 2$ pode ser escrito como:

$$\begin{aligned} & J_{h_p}^*(x_0) = x_0^T Y x_0 + \min_{U_{h_c}} \left\{ U_{h_c}^T H U_{h_c} + x_0^T F U_{h_c} \right\} \\ & sa. \\ & G U_{h_c} \leq W + E x_0 \\ & H \succ 0 \end{aligned} \quad , \tag{3.6}$$

onde $H = \tilde{B}^T \tilde{Q} \tilde{B} + \tilde{Q}_u,\ F = \tilde{A}^T \tilde{Q} \tilde{B},\ Y = \tilde{A}^T \tilde{Q} \tilde{A},\ \tilde{Q} = diag(Q_x, \cdots, Q_x, Q_N)$ e $\tilde{Q}_u = diag(Q_u, \cdots, Q_u)$. Considerando H semi-definida positiva, a solução será dada através do Teorema 3.1.

Na formulação PML, tem-se $P = 1,\ \ell_{h_p}\left(x_{h_p}\right) = Q_{h_p}\left|x_{h_p}\right|$ e $\ell(x_k, u_k) = Q_x|x_k| + Q_u|u_k|$. Usando variáveis de folga $\left(\epsilon_0, \epsilon_1, \cdots, \epsilon_{h_p-1}\right)$, $\left(\delta_0, \delta_1, \cdots, \delta_{h_p-1}\right)$ e γ, pode-se, então,

transformar a função não linear $J_{h_p}^*(x_0) = \min_{U_{h_c}} Q_{h_p} \left|x_{h_p}\right| + \sum_{k=0}^{h_p-1} Q_x|x_k| + Q_u|u_k|$ em uma função custo linear dada por:

$$
\begin{aligned}
& J_{h_p}^*(x_0) = \min_{\substack{U_{h_c},\epsilon_0,\cdots,\epsilon_{h_p-1},\\ \delta_0,\cdots,\delta_{h_p-1},\gamma}} \sum_{k=0}^{h_p-1} (\epsilon_k + \delta_k) + \gamma \\
& sa. \\
& GU_{h_c} \leq W + Ex_0 \\
& Q_u u_k \leq \mathbf{1}\epsilon_k, \qquad -Q_u u_k \leq \mathbf{1}\epsilon_k, \qquad k = 0, \cdots h_p - 1 \\
& Q_x x_k \leq \mathbf{1}\delta_k, \qquad -Q_x x_k \leq \mathbf{1}\delta_k, \qquad k = 0, \cdots h_p - 1 \\
& Q_{h_p} x_{h_p} \leq \mathbf{1}\gamma, \qquad -Q_{h_p} x_{h_p} \leq \mathbf{1}\gamma.
\end{aligned}
$$

No caso da norma $P = \infty$, a função custo será:

$$
\begin{aligned}
& J_{h_p}^*(x_0) = \min_{U_{h_c},\epsilon,\delta,\gamma} \epsilon + \delta + \gamma \\
& sa. \\
& GU_{h_p} \leq W + Ex_0 \\
& Q_u u_k \leq \mathbf{1}\epsilon, \qquad -Q_u u_k \leq \mathbf{1}\epsilon \\
& Q_x x_k \leq \mathbf{1}\delta, \qquad -Q_x x_k \leq \mathbf{1}\delta \\
& Q_{h_p} x_{h_p} \leq \mathbf{1}\gamma, \qquad -Q_{h_p} x_{h_p} \leq \mathbf{1}\gamma.
\end{aligned}
$$

Inserindo em (3.6) as restrições em (3.5) é possível resolver o problema da PMP através do Teorema 3.2 (KVASNICA, 2009).

3.4.4 COTFR para Sistemas Afins Por Partes

A formulação do problema de Controle Ótimo para Sistemas Afins por Partes requer a transformação do conectivo lógico Se-Então em uma representação matemática equivalente. Conforme sugerido por (BEMPORAD e MORARI, 1999), deve-se definir os seletores binários $\delta_i = \{0,1\}, i = 1, \cdots, n_L$ tais que

$$
(\delta_i = 1) \Leftrightarrow \left(\begin{bmatrix} x_k \\ u_k \end{bmatrix} \in \mathcal{D}_i \right).
$$

Assim, as restrições $H_i^x x \leq K_i^x, H_i^u u \leq K_i^u$ e $x_{k+1} = A_i x_k + B_i u_k$ para $i = 1, \cdots, n_L$, serão dadas por:

$$
\begin{aligned}
& H_i \begin{bmatrix} x_k \\ u_k \end{bmatrix} - K_i \leq M_i\left(1 - \delta_{k,i}\right), \\
& \sum_{i=1}^{n_L} \delta_i = 1, \\
& x_{k+1} - (A_i x_k + B_i u_k + c_i) \leq \widetilde{M}_i(1 - \delta_i), \\
& x_{k+1} - (A_i x_k + B_i u_k + c_i) \geq \widetilde{m}_i(1 - \delta_i).
\end{aligned}
$$

Nota-se que os seletores $\delta_i = \{0,1\}, i = 1, \cdots, n_L$ devem ser introduzidos para cada instante $k = 0, \cdots N$, o que implica em um total de variáveis binárias igual a $n_L \times N$. Como o problema está formulado com variáveis inteiras e reais, a minimização da função custo (3.4) será obtida usando a norma um ou infinita através da Programação Multiparamétrica Linear e Inteira (PMLI).

No caso do Algoritmo de Controle Preditivo, seja um sistema dinâmico Afins por Partes ou não, a solução do problema COTFR $U_N^* = \left\{u_0^*, u_1^*, \cdots u_{h_c}^*\right\}$ deve ser obtida a cada iteração, através da Programação Multiparamétrica, sendo que somente o sinal de controle no instante atual (u_0^*) é aplicado à planta.

Para qualquer sistema *PWA*, a estabilidade é garantida se um conjunto invariante, (definido em 3.6), é imposto como uma restrição de estado final, (definida em 3.4e) e a função custo final, (definida em 3.4a), corresponde a uma função de Lyapunov para este conjunto (KVASNICA, 2009).

Definição 3.6 (KVASNICA, 2009): O conjunto invariante $\mathcal{O}_\infty^{PWA}$, para um sistema discreto no tempo $x_{k+1} = f_{PWA}(x_k)$ e sujeita às restrições $x_k \in \mathbb{X}, \forall k \geq 0$ é definido por

$$\mathcal{O}_\infty^{PWA} \triangleq \{x_0 \in \mathbb{X} | \phi(k; x) \in \mathbb{X}, k \in \mathbb{N}_+\},$$

onde $\phi(k; x)$ é a solução de $x_{k+1} = f_{PWA}(x_k)$ no tempo k se o estado inicial é x_0.

Teorema 3.3 (GRIEDER, KVASNICA, *et al.*, 2004) e (GRIEDER, KVASNICA, *et al.*, 2005): Assumindo que o problema de otimização (3.4) é dado com a norma $P = 2$, isto é, o conjunto final é $\mathcal{T}_{set} = \mathcal{O}_\infty^{PWA}$ e o custo final é $\mathcal{Q}_{xh_p} = \mathcal{P}$. Se este problema é resolvido em cada intervalo de tempo para o sistem *PWA* e somente a primeira entrada é aplicada (Controle por Horizonte Retrocedente), então o sistema em malha fechada é assintoticamente estável.

3.5 CONSIDERAÇÕES

Este capítulo mostrou um breve histórico do controle preditivo, bem comos suas principais características e aplicações. Seu foco foi no controle preditivo multivariável por espaço de estados com restrições sendo otimizado pelo método da Programação Multiparamétrica. O controlador otimizado por este método é do tipo *PWA*, com variáveis contínuas e lógicas.

Foi definido o que são sistemas do tipo *PWA* e descrito como a PM gera as leis de controle *off-line*, separando o conjunto factível de estados do sistema em subconjuntos ou regiões politópicas.

Também foi demonstrado que o controlador gerado será estável, tanto para sistemas invariantes no tempo, *LTI*, quanto para sistemas *PWA*.

Capítulo 4: RESULTADOS

O desenvolvimento dos controladores dinâmico e cinemáticos da cadeira de rodas robótica foi realizado através de simulações computacionais utilizando o *MPT*, desenvolvido por M. Kvasnica *et. al*, (2010). Fazendo uso de ambos os modelos apresentados no capítulo 2, nos quais representam a dinâmica e cinemática do sistema constituído pela cadeira de rodas com ou sem usuário, foram feitas as suas linearizações através da série de Taylor.

Os testes foram realizados considerando: que os parâmetros estimados θ são obtidos na Tabela 2.2, tanto para a cadeira de rodas vazia quanto com passageiro; que os limites para a velocidade linear são de $-0{,}4m/s \leq v \leq 0{,}4m/s$ e para a velocidade angular de $-0{,}5rad/s \leq \omega \leq 0{,}5rads/s$; e que as perturbações externas δ_v e δ_ω, quando consideradas, tivessem seus limites iguais a 5% dos valores máximos e mínimos das velocidades nas quais influenciam, inseridas na forma de degrau. Assim, a perturbação na velocidade linear é igual a $-0{,}02 \leq \delta_v \leq 0{,}02$ e a perturbação na velocidade angular igual a $-0{,}025 \leq \delta_\omega \leq 0{,}025$.

Os parâmetros dos controladores, h_p e h_c, foram escolhidos por tentaiva e erro, de tal forma a fazer com que o sistema gastasse o menor tempo de simulação possível, utilizando assim, menos poder computacional. Os ganhos do controlador, referentes à entrada, estados e saída, foram escolhidos por tentativa e erro, visando uma melhor resposta do sistema.

Ambos os sistemas tiveram de ser linearizados pelo método que se utiliza da série de Taylor, conforme o seguinte procedimento: seja a equação não linear

$$\dot{X} = f(x,u), onde\ x = [x_1, \cdots, x_n]^T \ \text{ e } \ u = \left[u_1, \cdots, u_p\right]^T.$$

Considerando x e u pertencentes a regiões próximas dos pontos de operação $\overline{x}$ e $\overline{u}$, pode-se aproximar f para uma função linear $\overline{f}$ dada por

$$\overline{f}(x,u) = f(\overline{x},\overline{u}) + \frac{\partial f(\overline{x},\overline{u})}{\partial x_1}(x_1 - \overline{x_1}) + \cdots + \frac{\partial f(\overline{x},\overline{u})}{\partial x_n}(x_n - \overline{x_n}) + \frac{\partial f(\overline{x},\overline{u})}{\partial u_1}(u_1 - \overline{u_1}) + \cdots + \frac{\partial f(\overline{x},\overline{u})}{\partial u_p}\left(u_p - \overline{u_p}\right).$$

4.1 DETERMINAÇÃO DOS MODELOS LINEARES POR FUNÇÕES *PWA*

Ambos os modelos, dinâmico e cinemático, descritos por (2.4) e (2.9), respectivamente, possuem não linearidades e devem ser tratados a fim de representá-los por um sistema linear por funções *PWA*.

4.1.1 Sistema dinâmico

Para ser feita a linearização do modelo dinâmico representado pela equação (2.9), deve-se isolar $\dot{v}$ e $\dot{\omega}$ a fim de separar as saídas das entradas. Ficando assim,

$$\begin{cases} \dot{v} = \frac{v_{ref}}{\theta_1} + \frac{\delta_v}{\theta_1} + \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}v + \frac{\theta_7}{\theta_1}\dot{\omega} \\ \dot{\omega} = \frac{\omega_{ref}}{\theta_2} + \frac{\delta_\omega}{\theta_2} - \frac{\theta_5}{\theta_2}v\omega - \frac{\theta_6}{\theta_2}\omega + \frac{\theta_8}{\theta_2}\dot{v} \end{cases}, \tag{4.1}$$

fazendo,

$$\begin{cases} \frac{\theta_3}{\theta_1} = \theta_9 \\ \frac{\theta_4}{\theta_1} = \theta_{10} \\ \frac{\theta_7}{\theta_1} = \theta_{11} \end{cases} \quad e \quad \begin{cases} \frac{\theta_5}{\theta_2} = \theta_{12} \\ \frac{\theta_6}{\theta_2} = \theta_{13} \\ \frac{\theta_8}{\theta_2} = \theta_{14} \end{cases},$$

Tem-se então,

$$\begin{cases} \dot{v} = \frac{v_{ref}}{\theta_1} + \frac{\delta_v}{\theta_1} + \theta_9\omega^2 - \theta_{10}v + \theta_{11}\dot{\omega} \\ \dot{\omega} = \frac{\omega_{ref}}{\theta_2} + \frac{\delta_\omega}{\theta_2} - \theta_{12}v\omega - \theta_{13}\omega + \theta_{14}\dot{v} \end{cases}. \tag{4.2}$$

A equação de $\dot{v}$ é dependente da equação de $\dot{\omega}$ e vice-versa, para eliminar tal dependência, primeiramente substitui-se $\dot{\omega}$ em $\dot{v}$,

$$\dot{v} = \frac{v_{ref}}{\theta_1} + \frac{\delta_v}{\theta_1} + \theta_9\omega^2 - \theta_{10}v + \theta_{11}\left(\frac{\omega_{ref}}{\theta_2} + \frac{\delta_\omega}{\theta_2} - \theta_{12}v\omega - \theta_{13}\omega + \theta_{14}\dot{v}\right),$$

que após as devidas manipulações assume o seguinte formato,

$$\dot{v} = f_1 = \theta_{15}v_{ref} + \theta_{15}\delta_v + \theta_{16}\omega^2 - \theta_{17}v + \theta_{18}\omega_{ref} + \theta_{18}\delta_\omega - \theta_{19}v\omega - \theta_{20}\omega\,, \quad (4.3)$$

sendo,

$$\begin{cases} \dfrac{1}{\theta_1(1-\theta_{11}\theta_{14})} = \theta_{15} \\ \dfrac{\theta_9}{(1-\theta_{11}\theta_{14})} = \theta_{16} \\ \dfrac{\theta_{10}}{(1-\theta_{11}\theta_{14})} = \theta_{17} \\ \dfrac{\theta_{11}}{\theta_2(1-\theta_{11}\theta_{14})} = \theta_{18} \\ \dfrac{\theta_{11}\theta_{12}}{(1-\theta_{11}\theta_{14})} = \theta_{19} \\ \dfrac{\theta_{11}\theta_{13}}{(1-\theta_{11}\theta_{14})} = \theta_{20} \end{cases}.$$

De forma análoga, substitui-se $\dot{v}$ em $\dot{\omega}$,

$$\dot{\omega} = \frac{\omega_{ref}}{\theta_2} + \frac{\delta_\omega}{\theta_2} - \theta_{12}v\omega - \theta_{13}\omega + \theta_{14}\left(\frac{v_{ref}}{\theta_1} + \frac{\delta_v}{\theta_1} + \theta_9\omega^2 - \theta_{10}v + \theta_{11}\dot{\omega}\right)$$

que após as devidas manipulações assume o seguinte formato,

$$\dot{\omega} = f_2 = \theta_{21}\omega_{ref} + \theta_{21}\delta_\omega - \theta_{22}v\omega - \theta_{23}\omega + \theta_{24}v_{ref} + \theta_{24}\delta_v + \theta_{25}\omega^2 - \theta_{26}v\,, \quad (4.4)$$

sendo,

$$\begin{cases} \dfrac{1}{\theta_2(1-\theta_{14}\theta_{11})} = \theta_{21} \\ \dfrac{\theta_{12}}{(1-\theta_{14}\theta_{11})} = \theta_{22} \\ \dfrac{\theta_{13}}{(1-\theta_{14}\theta_{11})} = \theta_{23} \\ \dfrac{\theta_{14}}{\theta_1(1-\theta_{14}\theta_{11})} = \theta_{24} \\ \dfrac{\theta_{14}\theta_9}{(1-\theta_{14}\theta_{11})} = \theta_{25} \\ \dfrac{\theta_{14}\theta_{10}}{(1-\theta_{14}\theta_{11})} = \theta_{26} \end{cases}.$$

Utilizando as equações (4.3) e (4.4) e considerando que as perturbações externas são nulas, pode-se, assim, realizar o procedimento de linearização que se segue:

$$\begin{cases} \dot{v} = \overline{f_1} = f_1(\bar{v}, \bar{\omega}, \overline{v}_{ref}, \overline{\omega}_{ref}) + (-\theta_{17} - \theta_{19}\bar{\omega})(v - \bar{v}) + (2\theta_{16}\bar{\omega} - \theta_{19}\bar{v} - \theta_{20})(\omega - \bar{\omega}) + \\ \qquad\qquad \theta_{15}(v_{ref} + \overline{v}_{ref}) \\ \dot{\omega} = \overline{f_2} = f_2(\bar{v}, \bar{\omega}, \overline{v}_{ref}, \overline{\omega}_{ref}) + (-\theta_{22}\bar{\omega} - \theta_{26})(v - \bar{v}) + (-\theta_{22}\bar{v} - \theta_{23} + 2\theta_{25}\bar{\omega})(\omega - \bar{\omega}) + \\ \qquad\qquad \theta_{21}(\omega_{ref} - \overline{\omega}_{ref}) \end{cases},$$

Para uma primeira tentativa de linearização do modelo, considerou-se um sistema dinâmico *PWA* com uma única partição $D_i = x \times u$, onde $x = [-0{,}4\ ,\ 0{,}4] \times [-0{,}5\ ,\ 0{,}5]$ e $u = x$, sendo os pontos de operação os pontos médios dos intervalos citados. Sendo assim tem-se,

$$\begin{cases} \dot{v} = -\theta_{17}v - \theta_{20}\omega + \theta_{15}v_{ref} + \theta_{18}\omega_{ref} \\ \dot{\omega} = -\theta_{26}v - \theta_{23}\omega + \theta_{24}v_{ref} + \theta_{21}\omega_{ref} \end{cases}. \tag{4.5}$$

Escrevendo esta equação na forma matricial:

$$\begin{aligned} \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} &= \underbrace{\begin{bmatrix} -\theta_{17} & -\theta_{20} \\ -\theta_{26} & -\theta_{23} \end{bmatrix}}_{A} \begin{bmatrix} v \\ \omega \end{bmatrix} + \underbrace{\begin{bmatrix} \theta_{15} & \theta_{18} \\ \theta_{24} & \theta_{21} \end{bmatrix}}_{B} \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} \\ y &= \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{C} \begin{bmatrix} v \\ \omega \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}}_{D} \end{aligned}, \tag{4.6}$$

onde v_{ref} e ω_{ref} são os sinais de controle determinados pelo controlador dinâmico, os quais são *set points* dos PIDs de cada roda de tração e v e ω são as variáveis de estado e as saídas do sistema.

A Figura 4.1 mostra uma simulação definida através do programa mostrado no apêndice A. Foi aplicado um sinal *PRBS*, (*Pseudo Random Binary Sequence*), de entrada com amplitudes para velocidade linear variando entre $-0{,}4m/s$, $-0{,}2m/s$, $0m/s$, $0{,}2m/s$ e $0{,}4m/s$ e para a velocidade angular variando entre $-0{,}5rad/s$, $-0{,}25rad/s$, $0rad/s$, $0{,}25rad/s$, $0{,}5rad/s$ para se comparar o sistema não linear com o sistema linearizado com uma única partição D_i, considerando a cadeira de rodas com passageiro.

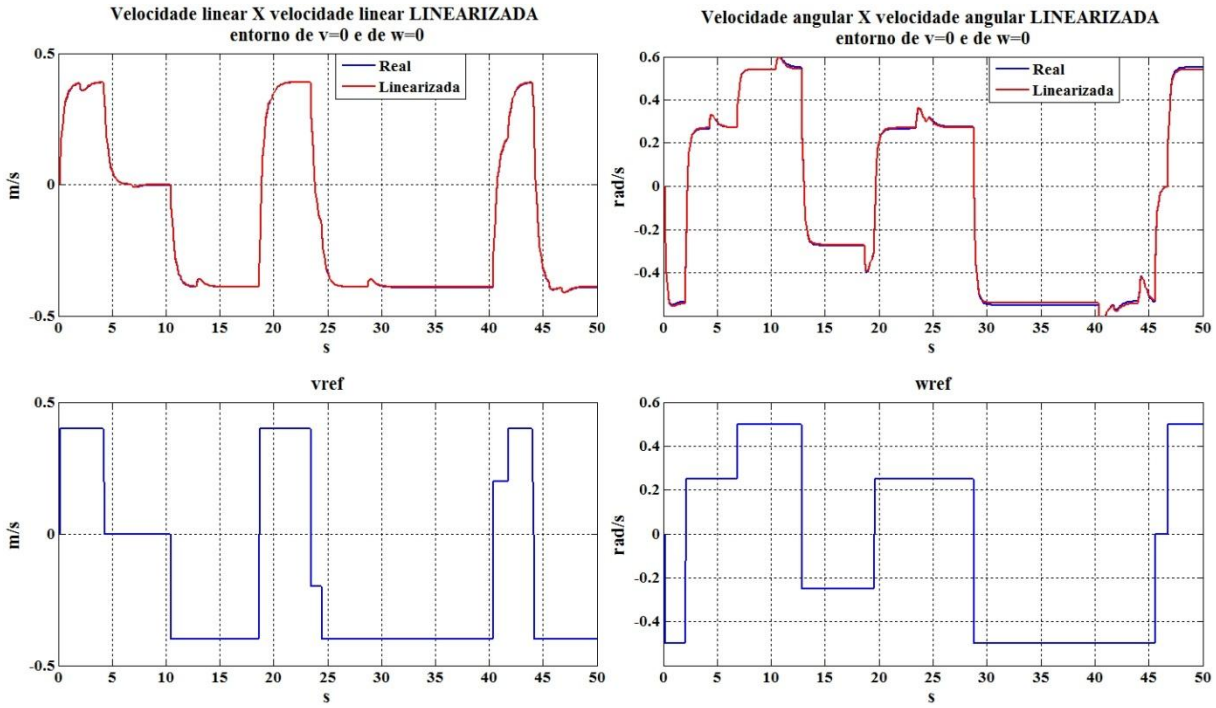


Figura 4.1: Simulação dos modelos dinâmico não linear e o linearizado.

A Figura 4.2 mostra o mesmo procedimento anterior, porém considerando os efeitos das perturbações δ_v e δ_ω, externas ao sistema.

O erro entre o modelo não linear e o modelo linearizado, para a primeira linha da equação (4.5), é de aproximadamente 0,13% sem a perturbação e de 0,34% com a perturbação e, para a segunda linha, de aproximadamente 0,27% sem a perturbação e de 1,65% com a perturbação, validando assim o modelo linear com uma única partição $D_i = x \times u$, linearizada em torno dos pontos $\overline{v} = 0$ e $\overline{\omega} = 0$.

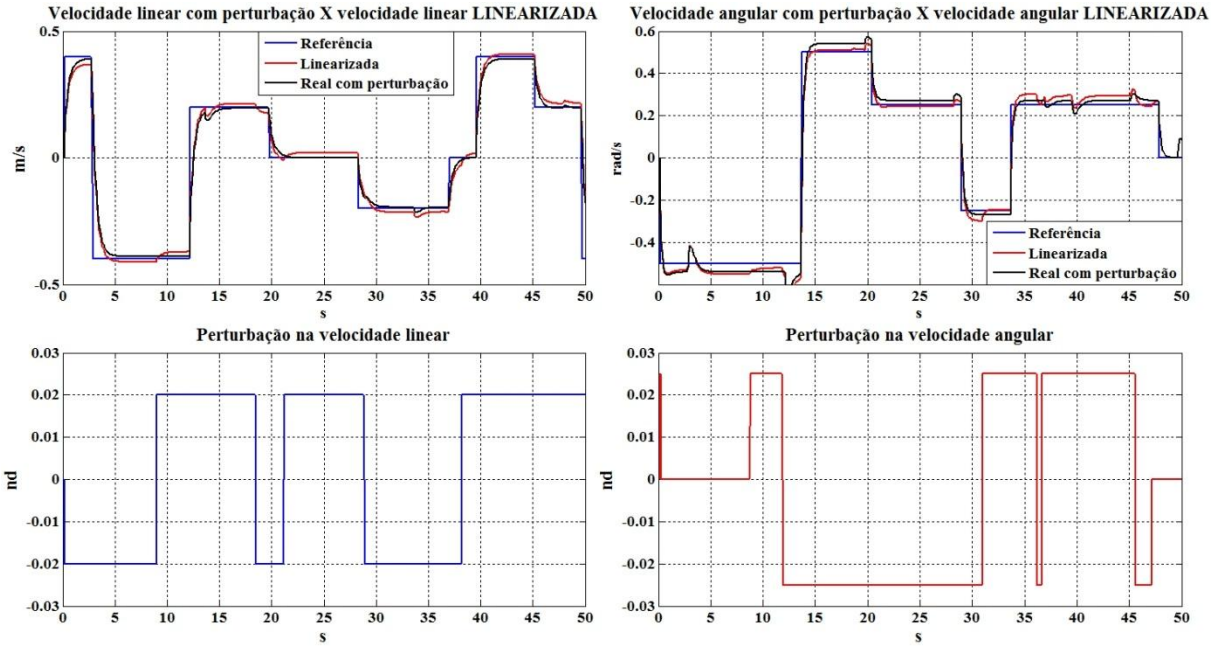


Figura 4.2: Simulação dos modelos dinâmico não linear com perturbação e o linearizado.

4.1.2 Sistema cinemático

O sistema não linear descrito por (2.4) possui duas equações que precisam ser linearizadas em relação às variáveis, velocidade linear v e ângulo de orientação φ. Assumindo que x, y e φ estão próximos aos pontos de operação $\overline{x}, \overline{y}$ e $\overline{\varphi}$, a linearização do sistema cinemático é dada pela seguinte expressão,

$$\begin{cases} \dot{x} = \overline{f_1} = f_1(\overline{x}, \overline{y}, \overline{\varphi}, \overline{v}) - \bar{v}\,\text{sen}\,\bar{\varphi}\,(\varphi - \bar{\varphi}) + \cos\bar{\varphi}\,(v - \bar{v}) \\ \dot{y} = \overline{f_2} = f_2(\overline{x}, \overline{y}, \overline{\varphi}, \overline{v}) + \bar{v}\cos\bar{\varphi}\,(\varphi - \bar{\varphi}) + \text{sen}\,\bar{\varphi}\,(v - \bar{v}) \\ \dot{\varphi} = \omega \end{cases}$$

Após simplificação, tem-se

$$\begin{cases} \dot{x} = (-\bar{v}\,\text{sen}\,\bar{\varphi})\varphi + (\cos\bar{\varphi})v + \bar{v}\bar{\varphi}\,\text{sen}\,\bar{\varphi} \\ \dot{y} = (\bar{v}\cos\bar{\varphi})\varphi + (\text{sen}\,\bar{\varphi})v - \bar{v}\bar{\varphi}\cos\bar{\varphi} \\ \dot{\varphi} = \omega \end{cases},$$

Que também pode ser escrita na forma matricial,

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\varphi}\end{bmatrix} = \begin{bmatrix}0 & 0 & -\bar{v}\,\text{sen}\,\bar{\varphi}\\ 0 & 0 & \bar{v}\cos\bar{\varphi}\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ \varphi\end{bmatrix} + \begin{bmatrix}\cos\bar{\varphi} & 0\\ \text{sen}\,\bar{\varphi} & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} + \begin{bmatrix}\bar{v}\bar{\varphi}\,\text{sen}\,\bar{\varphi}\\ -\bar{v}\bar{\varphi}\cos\bar{\varphi}\\ 0\end{bmatrix},$$
$$\begin{bmatrix}y_1\\ y_2\\ y_3\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ \varphi\end{bmatrix} \qquad (4.7)$$

onde v e ω são os sinais de controle determinados pelo controlador cinemático, os quais são os *setpoints* do sistema dinâmico.

Para esta linearização, optou-se em realizar o procedimento por etapas, sempre observando o erro entre a trajetória real e a linearizada. A primeira tentativa foi feita com apenas uma partição, ou seja, $P_1 = \mathbb{R} \times \mathbb{R} \times [0°, 360°] \times [-0{,}4\ ,\ 0{,}4] \times [-0{,}5\ ,\ 0{,}5]$, linearizando o sistema em torno dos pontos médios dos intervalos, neste caso, $\overline{v} = 0$ para a velocidade linear e $\overline{\varphi} = 180°$ para o ânguolo de orientação. O resultado pode ser observado na Figura 4.3.

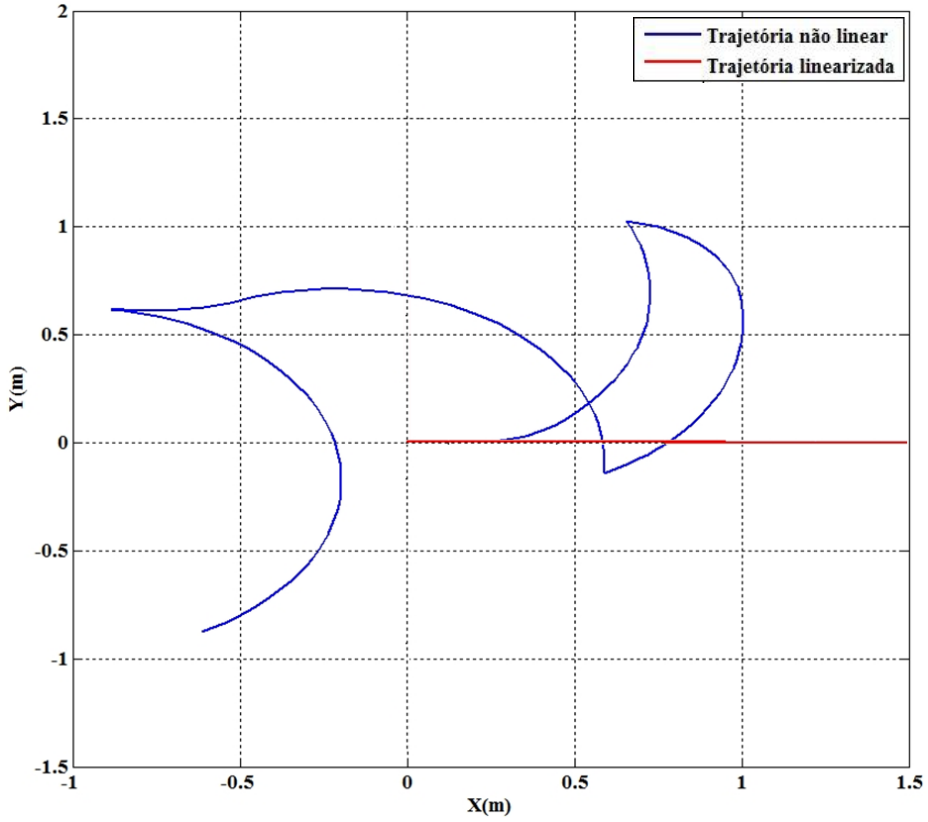


Figura 4.3: Simulação dos modelos cinemático real e linearizado com uma partição.

O resultado obtido foi muito aquém do que se desejava, motivo que determinou um avanço nas etapas de linearização.

O teste seguinte, Figura 4.4, foi então realizado para 12 linearizações, dividindo-se a faixa da velocidade linear em três intervalos,

$$Iv_1 = [-0{,}1\ ,\ 0{,}1],\ Iv_2 = [0{,}1\ ,\ 0{,}4] \text{ e } Iv_3 = [-0{,}4\ ,\ -0{,}1]\ , \tag{4.8}$$

com pontos de operação como sendo os pontos médios de seus intervalos: $\overline{v}_1 = 0$, $\overline{v}_2 = 0{,}25$ e $\overline{v}_3 = -0{,}25$.

Para φ, dividiu-se o círculo trigonométrico em quatro intervalos,

$$I\varphi_1 = [0°\ ,\ 90°],\ I\varphi_2 = [90°\ ,\ 180°],\ I\varphi_3 = [180°\ ,\ 270°] \text{ e } I\varphi_4 = [270°\ ,\ 360°]\ ,$$

com pontos de operação como sendo os pontos médios de seus intervalos: $\overline{\varphi}_1 = 45°, \overline{\varphi}_2 = 135°,\ \overline{\varphi}_3 = 225°\ e\ \overline{\varphi}_4 = 315°$.

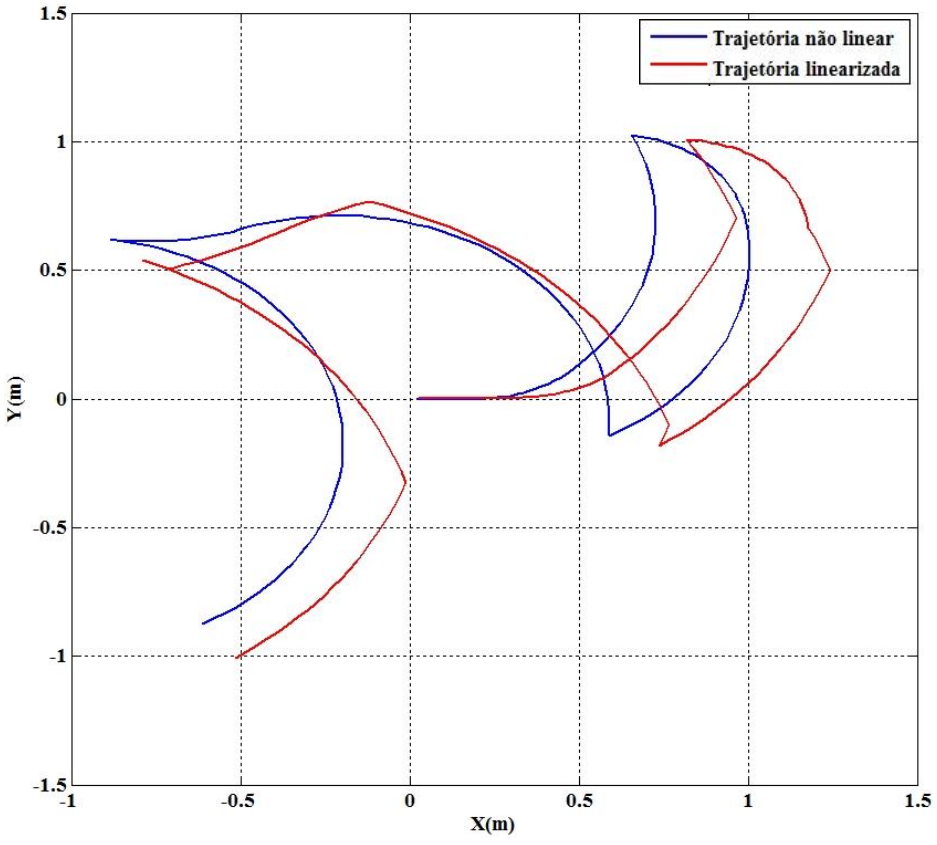


Figura 4.4: Simulação dos modelos cinemático real e linearizado com 12 partições.

Observa-se uma significativa melhora, porém o erro entre as duas trajetórias é de aproximadamente 20%, ainda aquém do que se deseja.

Os testes continuaram, seguindo o mesmo padrão, até que se obteve um modelo linearizado com menos de 5% de erro. Para tal, se fez necessário 3 pontos de linearização para v, os mesmos 3 citados em (4.8), e 16 intervalos para o ângulo de orientação, totalizando assim 48 intervalos diferentes.

Para o ângulo de orientação tem-se

$$\mathrm{I}\varphi_{i+1} = [22{,}5i \ , \ 22{,}5(i+1)] \ \text{ sendo } i = 0, 1, \cdots, 15. \tag{4.9}$$

com pontos de operação como sendo os pontos médios de seus intervalos:

$$\overline{\varphi}_{i+1} = 22{,}5i + 11{,}25 \ .$$

A Figura 4.5 mostra um gráfico da simulação do sistema cinemático real comparado ao sistema linearizado em 48 partições distintas e com erros de aproximadamente 1,5% no deslocamento em horizontal e de 0,8% no deslocamento vertical.

Todas as programações das simulações do modelo cinemático encontram-se no apêndice B

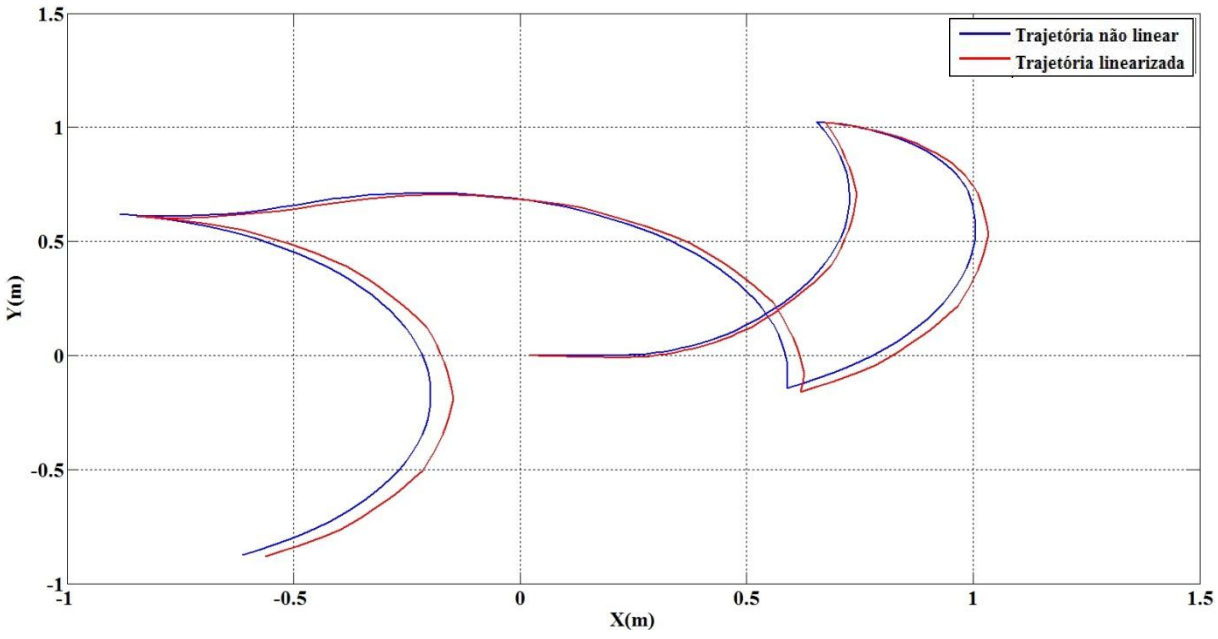


Figura 4.5: Simulação dos modelos cinemático real e linearizada com 48 partições.

4.2 DESENVOLVIMENTO DO CONTROLE DINÂMICO

O projeto do controlador foi feito levando-se em consideração o sistema dinâmico linear (4.6) e as simulações foram realizadas usando-se o modelo não linearizado regido pelas equações (4.3) e (4.4), considerando os efeitos das perturbações externas δ_v e δ_ω.

Para se desenvolver o controlador preditivo para o modelo dinâmico da cadeira de rodas, primeiramente estipulou-se quais são as restrições envolvidas no sistema. Por questões de segurança e conforto para o passageiro, a velocidade linear e a velocidade angular têm de ser limitadas entre:

$$\begin{cases} -0{,}4m/s \leq v \leq 0{,}4m/s \\ -0{,}5rad/s \leq \omega \leq 0{,}5rads/s \end{cases}. \tag{4.10}$$

A escolha do horizonte de previsão, horizonte de controle e dos pesos para sintonia do controlador foram obtidos por tentativa e erro. Foi tomada como base a minimização do tempo gasto na realização dos cálculos do controlador (*off-line*), do tempo gasto por iteração na ação de controle (*online*) e do erro percentual total entre a referência e a saída do sistema.

4.2.1 Configuração do *MPT*

O *MPT* é composto de diversas ferramentas, funções e parâmetros que devem ser configurados de maneira adequada para a obtenção de um controlador satisfatório.

Todos os parâmetros que devem ser inseridos estão separados em dois blocos: o bloco de sistema, *sysStruct*, onde todas as informações do modelo do sistema devem ser inseridas, e o bloco de configurações dos parâmetros de sintonia do controlador, *probStruct*. Ambos os blocos recebem o sufixo *Struct*, pois, no *MatLab*, são um aglomerado de variáveis reunidas em uma única variável do tipo *Struct*.

No bloco *sysStruct*, definem-se as matrizes A, B, C e D, do modelo em espaço de estados (equação 4.6); o tempo de amostragem; as restrições nos estados, nas saídas e nas variações do sinal de controle, como mostra o código abaixo.

```
% define o modelo discretizado baseado no modelo original em espaço de estados
```

```
sysStruct = mpt_sys(ss(A, B, C, D), Ts);

% Definição das restrições nos estados
sysStruct.xmax = [inf; inf];
sysStruct.xmin = [-inf; -inf];

% Definição das restrições nas entradas
sysStruct.umax = [0.4;0.5]
sysStruct.umin = [-0.4;-0.5];

% Definição das restrições nas saídas
sysStruct.ymax = [0.4;0.5]
sysStruct.ymin = [-0.4;-0.5];

% Definição das restrições nas variações do sinal de controle
sysStruct.dumax = inf;
sysStruct.dumin = inf;
```

Como pode ser observado, tanto as restrições de entrada quando as de saída foram configuradas para atender ao critério escolhido para as velocidades v e ω, enquanto que nos outros parâmetros não foi inserida quaisquer tipo de restrições.

No caso do *probStruct*, foram configurados os valores do horizonte de previsão, h_p, do horizonte de controle, h_c, dos pesos da saída (Q_y), da entrada (R) e dos estados (Q), da norma adotada, do tipo de controle (servo ou regulador) e o método de solução do controlador (horizonte finito, horizonte infinito, tempo mínimo ou baixa complexibilidade) que será adotada.

Os parâmetros fixos adotados foram, a norma, como sendo a quadrática, o controle, como sendo do tipo servo ou seguidor e o método de solução do controlador, como sendo a solução ótima da função custo para um horizonte finito. Foi estipulado como ponto de partida para as simulações que o horizonte de previsão e de controle seriam $h_p = 4$ e $h_c = 2$, respectivamente, e que os pesos teriam seus valores iguais a

$$Q_y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad Q_r = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Tais configurações podem ser observadas no código que segue:

```
% Configuração do horizonte de previsão
probStruct.N = 4;

% Configuração do horizonte de controle
probStruct.Nc = 2;

Definição do tipo de norma utilizada pelo controlador
probStruct.norm = 2;
```

```
Configurações dos pesos nos estados
probStruct.Q = eye(2);

Configurações dos pesos nas saídas
probStruct.Qy = eye(2);

Configurações dos pesos nas entradas
probStruct.R = eye(2);

Configuração do tipo de controle
probStruct.tracking = 1;

Configuração do método de solução do controlador
probStruct.subopt_lev=0;
```

As simulações foram realizadas com um tempo de amostragem de 0,1s durante um intervalo de 15s. O *setpoint* da velocidade linear foi dado na forma de degraus variando entre $-0{,}4m/s$, $0m/s$ *e* $0{,}4m/s$. O mesmo foi feito para a velocidade angular, porém as variações de seus valores foram entre $-0{,}5rad/s, 0rad/s$ *e* $0{,}5rad/s$. As perturbações δ_v e δ_ω também foram inseridas na forma de degraus com amplitude igual a 5% das amplitudes das velocidades.

Após realizar a primeira simulação com os dados supracitados, Figura 4.6, observa-se que o erro de *offset* foi alto, em torno de 56%. A partir deste início, a sintonia foi feita por tentativa e erro, sempre levando em consideração o tempo gasto na simulação. Por fim chegou-se em uma configuração bem satisfatória, onde o horizonte de previsão foi alterado para $h_p = 8$ e o ganho na saída do sistema alterado para:

$$Q_y = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$

A simulação com a nova sintonia pode ser observada na Figura 4.7.

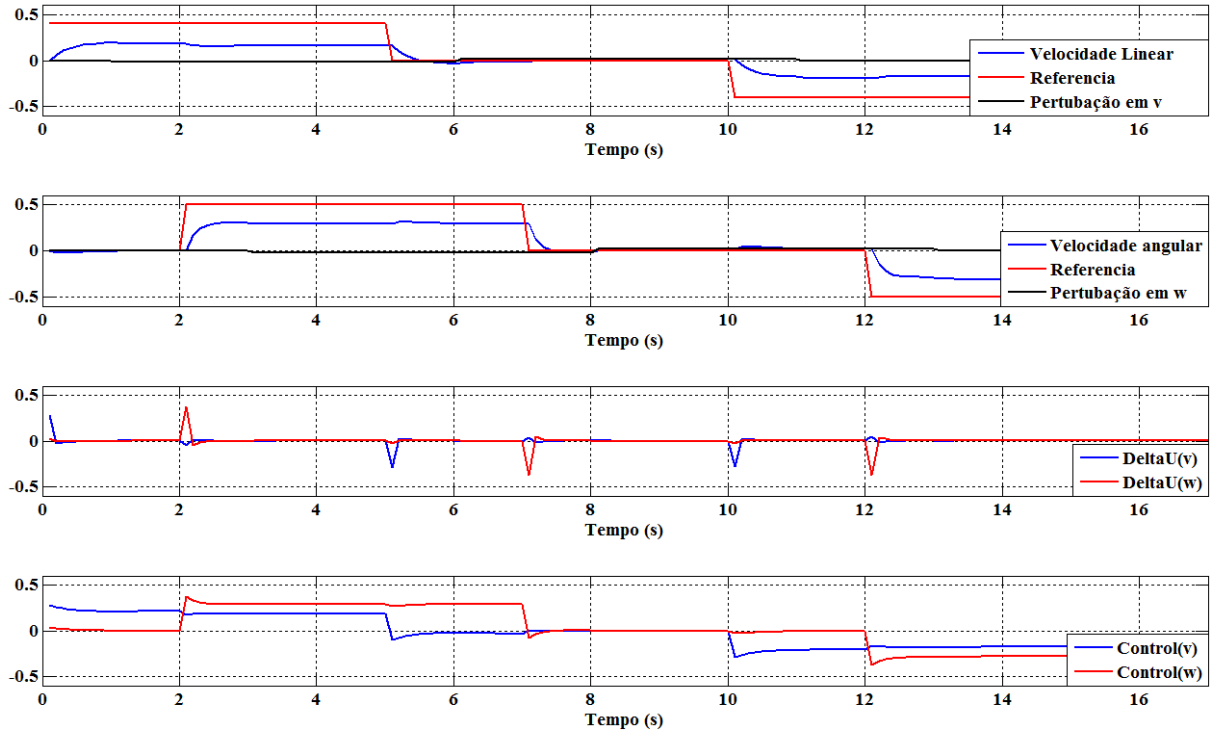


Figura 4.6: Resposta a variações em degrau do sistema dinâmico com perturbação para $h_p = 4$ e $Q_y = I$.

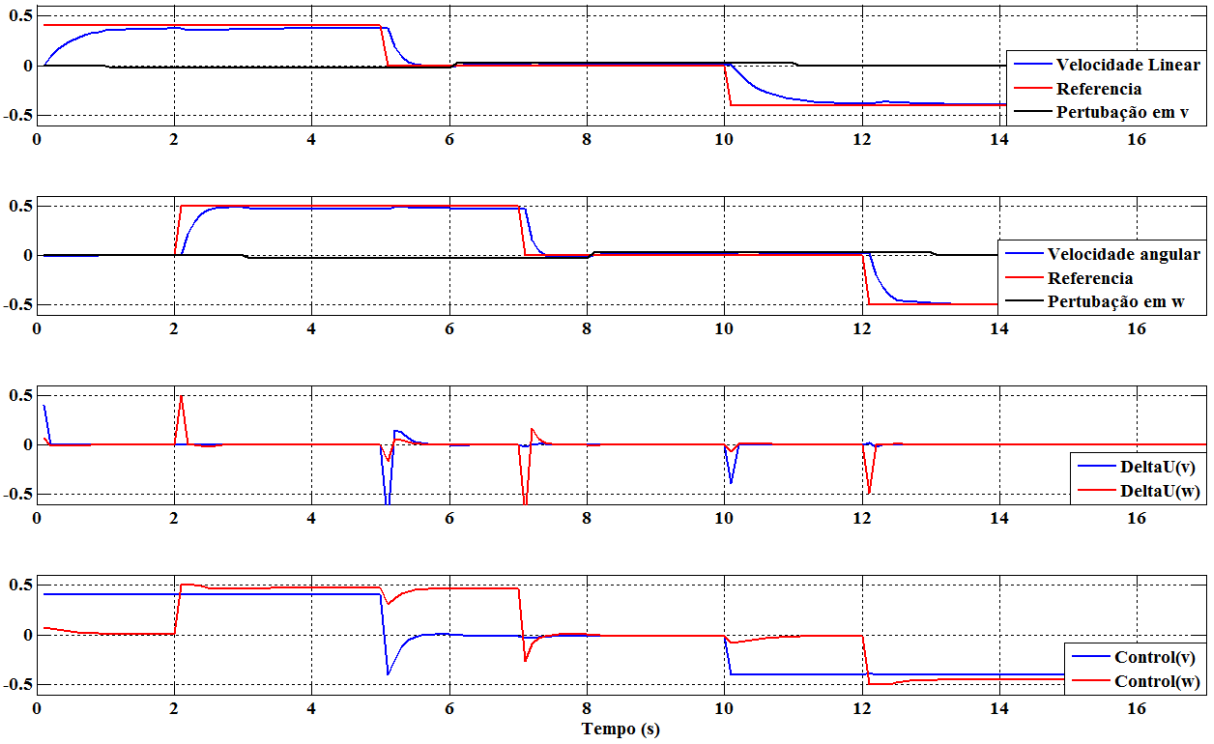


Figura 4.7: Resposta a variações em degrau com controlador sintonizado, $h_p = 8$ e $Q_y = 20I$.

A nova resposta do sistema apresentou um erro total de aproximadamente 9,5% para a velocidade linear e de 4,5% para a velocidade angular, o tempo de cálculo *off-line* do controlador foi de 4,29s e o tempo para fazer os cálculos online foi de 35ms por amostragem.

O programa utilizado nas simulações encontra-se no apêndice C.

4.2.2 Características do controlador dinâmico

Apesar de o sistema ser do tipo invariante no tempo, *LTI*, o controlador criado é do tipo *PWA*, tendo em vista os teoremas 3.1 e 3.2. Durante a etapa de cálculo do controlador *off-line*, os estados são separados em partições, onde cada uma delas sofrerá uma ação de controle específica.

Para o controlador dinâmico desenvolvido, os estados foram separados em 14 regiões, Figura 4.8 (a), cada uma com sua ação de controle, Figura 4.8 (b).

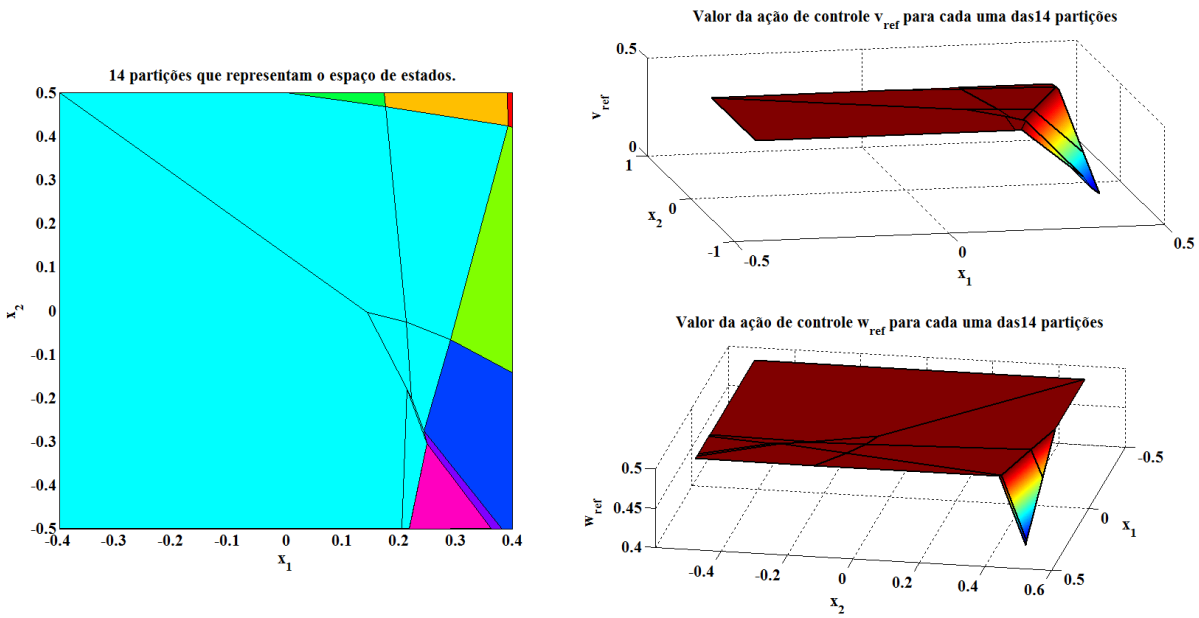


Figura 4.8: (a) Regiões do espaço de estados. (b) Valor da ação de controle ótima em função das regiões dos espaços de estados.

A Figura 4.8 (a), mostra que toda a região dos estados está preenchida, o que significa que o sistema pode partir de qualquer ponto inicial x_0 e chegar ao *setpoint* especificado.

Para se plotar os gráficos da Figura 4.8 utilizam-se os seguintes comandos:

```
Plotagem do gráfico de regiões do espaço de estados
plot(ctrl);

Plotagem dos gráficos das ações de controle em função das regiões
plotu(ctrl);
```

Onde ctrl é o controlador que foi calculado *off-line*.

Neste exemplo, as ações de controle são iguais a:

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{cases} \begin{bmatrix} -1{,}895 & -0{,}045 \\ -0{,}580 & -1{,}090 \\ -0{,}181 & -0{,}027 \\ -0{,}084 & -0{,}036 \end{bmatrix} X + \begin{bmatrix} 1{,}164 \\ 1{,}190 \\ 0{,}400 \\ 0{,}454 \end{bmatrix} \; se \; \begin{bmatrix} -1{,}000 & -0{,}024 \\ -0{,}470 & -0{,}883 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} X \leq \begin{bmatrix} -0{,}403 \\ -0{,}559 \\ 0{,}400 \\ 0{,}500 \end{bmatrix} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad Região\ 01 \\ \begin{bmatrix} 0 & 0 \\ -0{,}215 & -1{,}082 \\ -0{,}388 & -0{,}032 \\ -0{,}157 & -0{,}038 \end{bmatrix} X + \begin{bmatrix} 0{,}400 \\ 1{,}043 \\ 0{,}483 \\ 0{,}484 \end{bmatrix} \; se \; \begin{bmatrix} -0{,}997 & -0{,}082 \\ -0{,}195 & -0{,}981 \\ 0 & 1 \\ 1{,}000 & 0{,}024 \end{bmatrix} X \leq \begin{bmatrix} -0{,}214 \\ -0{,}492 \\ 0{,}500 \\ 0{,}403 \end{bmatrix} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad Região\ 02 \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -0{,}336 & 0{,}009 \\ 0 & 0 \end{bmatrix} X + \begin{bmatrix} 0{,}400 \\ 0{,}500 \\ 0{,}473 \\ 0{,}500 \end{bmatrix} \; se \; \begin{bmatrix} 0{,}963 & 0{,}269 \\ -1{,}000 & 0{,}026 \\ 0 & -1 \\ 0{,}988 & -0{,}154 \end{bmatrix} X \leq \begin{bmatrix} 0{,}158 \\ -0{,}219 \\ 0{,}500 \\ 0{,}293 \end{bmatrix} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad Região\ 14 \end{cases},$$

onde $X = [v\ \omega]^T$.

A Figura 4.9 mostra a trajetória do espaço de estados quando o sistema parte de diferentes pontos iniciais com o intuito de convergir em $v = 0{,}4\ m/s$ e $\omega = 0{,}5\ rad/s$. Pode-se observar que para cada condição inicial x_0 os estados descrevem diferentes trajetórias até atingir o estado final.

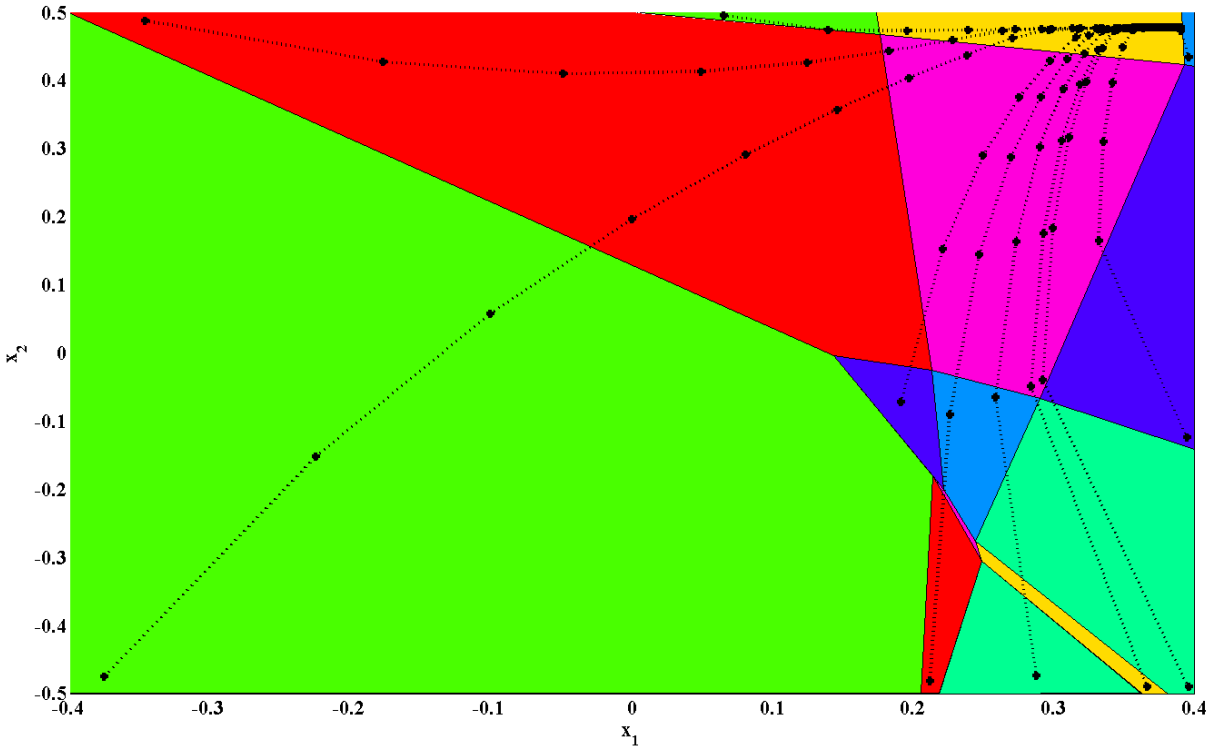


Figura 4.9: Trajetória dos estados $x = [v \;\; \omega]^T$.

4.3 DESENVOLVIMENTO DO CONTROLE CINEMÁTICO

Como o controlador só irá atuar sobre o deslocamento na coordenada (x, y), sistema apresentado em (4.7) não dependerá da saída φ, ficando então,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\bar{v}\,\text{sen}\,\bar{\varphi} \\ 0 & 0 & \bar{v}\cos\bar{\varphi} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} + \begin{bmatrix} \cos\bar{\varphi} & 0 \\ \text{sen}\,\bar{\varphi} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} + \begin{bmatrix} \bar{v}\bar{\varphi}\,\text{sen}\,\bar{\varphi} \\ -\bar{v}\bar{\varphi}\cos\bar{\varphi} \\ 0 \end{bmatrix}.$$
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \varphi \end{bmatrix}$$

Diferentemente do modelo dinâmico, o cinemático possui 48 linearizações, uma para cada intervalo escolhido, conforme sesão 4.1.2, sendo então necessário desenvolver o projeto de um sistema dinâmico do tipo *PWA*. Para tal, é necessário representar todos os intervalos por uma desigualdade linear do tipo,

$$G^x x(k) + G^u u(k) \leq G^c. \tag{4.11}$$

Antes, porém, é necessário definir as seguintes desigualdades para cada uma das 48 partições, ficando:

Para $-0{,}1 \leq v \leq 0{,}1$

$$\begin{array}{lll} Setor\ 1: & 0° \leq \varphi < 22{,}5° & se\ \overline{\varphi} = 11{,}25° \\ Setor\ 2: & 22{,}5° \leq \varphi < 45° & se\ \overline{\varphi} = 33{,}75° \\ & \vdots & \\ Setor\ 16 & 337{,}5° \leq \varphi < 360° & se\ \overline{\varphi} = 371{,}25° \end{array}$$

Para $0{,}1 < v \leq 0{,}4$

$$\begin{array}{lll} Setor\ 17: & 0° \leq \varphi < 22{,}5° & se\ \overline{\varphi} = 11{,}25° \\ Setor\ 18: & 22{,}5° \leq \varphi < 45° & se\ \overline{\varphi} = 33{,}75° \\ & \vdots & \\ Setor\ 32 & 337{,}5° \leq \varphi < 360° & se\ \overline{\varphi} = 371{,}25° \end{array} \qquad (4.12)$$

Para $-0{,}4 \leq v < -0{,}1$

$$\begin{array}{lll} Setor\ 33: & 0° \leq \varphi < 22{,}5° & se\ \overline{\varphi} = 11{,}25° \\ Setor\ 34: & 22{,}5° \leq \varphi < 45° & se\ \overline{\varphi} = 33{,}75° \\ & \vdots & \\ Setor\ 48 & 337{,}5° \leq \varphi < 360° & se\ \overline{\varphi} = 371{,}25° \end{array}$$

Observe que a desigualdade $b \leq x \leq a$ é equivalente a:

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} x \leq \begin{bmatrix} a \\ -b \end{bmatrix}.$$

Repetindo este raciocínio para cada desigualdade descrita em (4.12), obtém-se a seguinte desigualdade matricial:

$$
\left\{
\begin{array}{l}
\left.
\begin{array}{l}
\begin{bmatrix}0&0&-1\\0&0&1\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}x\\y\\\varphi\end{bmatrix}+\begin{bmatrix}0&0\\0&0\\-1&0\\1&0\end{bmatrix}\begin{bmatrix}v\\\omega\end{bmatrix}\le\begin{bmatrix}0\\22,5\\0,1\\0,1\end{bmatrix}\\
\qquad\qquad\qquad\vdots\\
\begin{bmatrix}0&0&-1\\0&0&1\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}x\\y\\\varphi\end{bmatrix}+\begin{bmatrix}0&0\\0&0\\-1&0\\1&0\end{bmatrix}\begin{bmatrix}v\\\omega\end{bmatrix}\le\begin{bmatrix}-337,5\\360\\0,1\\0,1\end{bmatrix}
\end{array}\right\} Setores\ 1\ \text{à}\ 16\\
\left.
\begin{array}{l}
\begin{bmatrix}0&0&-1\\0&0&1\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}x\\y\\\varphi\end{bmatrix}+\begin{bmatrix}0&0\\0&0\\-1&0\\1&0\end{bmatrix}\begin{bmatrix}v\\\omega\end{bmatrix}\le\begin{bmatrix}0\\22,5\\0,1\\0,4\end{bmatrix}\\
\qquad\qquad\qquad\vdots\\
\begin{bmatrix}0&0&-1\\0&0&1\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}x\\y\\\varphi\end{bmatrix}+\begin{bmatrix}0&0\\0&0\\-1&0\\1&0\end{bmatrix}\begin{bmatrix}v\\\omega\end{bmatrix}\le\begin{bmatrix}-337,5\\360\\0,1\\0,4\end{bmatrix}
\end{array}\right\} Setores\ 17\ \text{à}\ 32\\
\left.
\begin{array}{l}
\begin{bmatrix}0&0&-1\\0&0&1\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}x\\y\\\varphi\end{bmatrix}+\begin{bmatrix}0&0\\0&0\\-1&0\\1&0\end{bmatrix}\begin{bmatrix}v\\\omega\end{bmatrix}\le\begin{bmatrix}0\\22,5\\-0,1\\0,4\end{bmatrix}\\
\qquad\qquad\qquad\vdots\\
\underbrace{\begin{bmatrix}0&0&-1\\0&0&1\\0&0&0\\0&0&0\end{bmatrix}}_{G^x}\begin{bmatrix}x\\y\\\varphi\end{bmatrix}+\underbrace{\begin{bmatrix}0&0\\0&0\\-1&0\\1&0\end{bmatrix}}_{G^u}\begin{bmatrix}v\\\omega\end{bmatrix}\le\underbrace{\begin{bmatrix}-337,5\\360\\-0,1\\0,4\end{bmatrix}}_{G^c}
\end{array}\right\} Setores\ 33\ \text{à}\ 48
\end{array}
\right.
$$

Com isso, os setores do sistema PWA podem ser escritos conforme equação (4.11), sendo:

$$G^x{}_{192X3}\,X(k)+G^u{}_{192X2}\,U(k)\le G^c{}_{192X1} \tag{4.13}$$

De maneira análoga ao controlador dinâmico, as restrições envolvidas no sistema são as mesmas, a velocidade linear sendo limitada entre $-0{,}4m/s \le v \le 0{,}4m/s$ e a velocidade angular limitada entre $-0{,}5rad/s \le \omega \le 0{,}5rads/s$.

A escolha do horizonte de previsão, horizonte de controle e dos pesos do controlador também foi por tentativa e erro. Foi tomada como base a minimização do tempo gasto na a realização dos cálculos do controlador (*off-line*), do tempo gasto por iteração na ação de controle (*online*) e do erro percentual total entre a referência e a saída do sistema.

4.3.1 Configuração do *MPT*

De forma análoga ao controlador dinâmico, os parâmetros devem ser inseridos separadamente, no bloco de sistema, *sysStruct*, e no bloco de configurações do problema, *probStruct*.

Para o controlador cinemático do tipo *PWA*, o *sysStruct* deve receber algumas matrizes a mais, se comparado ao *sysStruct* do controlador dinâmico. Como o sistema em espaço de estados *PWA* segue a equação (2.4), uma matriz f, matriz constante proveniente das linearizações, tem de ser incluída. As matrizes da equação (4.13), que representam os setores do sistema, também são inseridas no *sysStruct*, como mostra o código abaixo.

```
% define o modelo discretizado baseado no modelo original em espaço de estados
sysStruct = mpt_sys(ss(A, B, C, D), Ts);

% Inserção da matriz f referente: x(k+1)=A{1}x(k)+B{1}u(k)+f{1}
sysStruct.f{setor} = [vb*fib*sin(fib);-vb*fib*cos(fib);0];

% Inserção dos setores: Gx{1}*x(k)+Gu{1}*u(k)<=Gc{1}
sysStruct.guardX{setor}=[0 0 -1;0 0 1;0 0 0;0 0 0];
sysStruct.guardU{setor}=[0 0;0 0;-1 0;1 0];
sysStruct.guardC{setor}=[-fi1;fi2;-v1;v2];

 % Definição das restrições nos estados
sysStruct.xmax = [inf; inf];
sysStruct.xmin = [-inf; -inf];

% Definição das restrições nas entradas
sysStruct.umax = [0.4;0.5]
sysStruct.umin = [-0.4;-0.5];

% Definição das restrições nas saídas
sysStruct.ymax = [inf; inf];
sysStruct.ymin = [-inf; -inf];

% Definição das restrições nas variações do controlador
sysStruct.dumax = inf;
sysStruct.dumin = inf;
```

Neste caso, como a saída são valores da coordenada x e y, nenhuma restrição foi inserida e as restrições sobre v e ω foram mantidas.

Os parâmetros fixos adotados no *probStruct* foram; a norma, como sendo a infinita, uma vez que a quadrática não se aplica neste caso, capítulo 3.4.3; o tipo de controle, como sendo do tipo servo ou seguidor; e o método de solução do controlador, como sendo a solução ótima da função custo para um horizonte finito. Foi estipulado, como ponto de partida para as

simulações, que o horizonte de previsão seria $h_p = 4$ e que os pesos teriam seus valores iguais a:

$$Q_y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad Q_r = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad Q_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

Tais configurações podem ser observadas no código que segue:

```
% Configuração do horizonte de previsão
probStruct.N = 4;

Definição do tipo de norma utilizada pelo controlador
probStruct.norm = inf;

Configurações dos ganhos nos estados
probStruct.Q = eye(3);

Configurações dos ganhos nas saídas
probStruct.Qy = eye(2);

Configurações dos ganhos nas entradas
probStruct.R = eye(2);

Configuração do tipo de sistema
probStruct.tracking = 1;

Configuração do grau de otimização do controle
probStruct.subopt_lev=0;
```

Inicialmente as simulações foram realizadas com um tempo de amostragem de 0,1s, considerando todo o círculo triginométrico, com a coordenada inicial sendo $(0,0)$ e as velocidades linear e angular nulas. Porém, devido ao tempo gasto na simulação, 1,2 min por iteração, e no cálculo *off-line* do controlador, 5,4 min, optou-se por simular apenas o primeiro quadrante do círculo trigonométrico. Neste caso apenas 12 dos 48 setores do sistema *PWA*. Além disso, usou-se um tempo de amostragem de 0,5s e diminuiu-se o horizonte de previsão para $h_p = 3$.

Na primeira simulação com os dados acima, Figura 4.10, foi aplicado um *setpoint* com a coordenada $(2,1)$. Observa-se que ocorreu um erro de *offset* e que o sistema fica oscilando em regime. Mais uma vez a sintonia foi obtida por tentativa e erro, obtendo-se uma configuração satisfatória, onde o ganho na saída para y foi alterado para:

$$Q_y = \begin{bmatrix} 7 & 0 \\ 0 & 0{,}4 \end{bmatrix}.$$

A Figura 4.11 mostra a simulação com o controlador sintonizado.

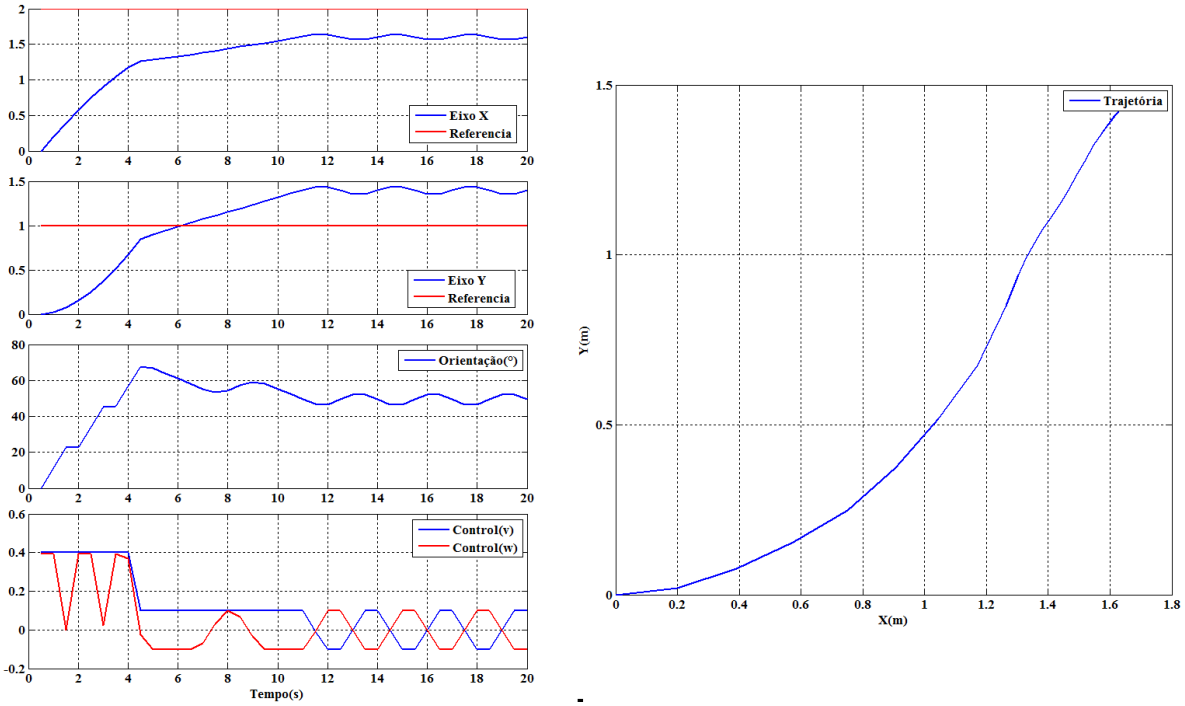


Figura 4.10: Resposta ao degrau de (2,1) com $h_p = 4$ e $Q_y = I$.

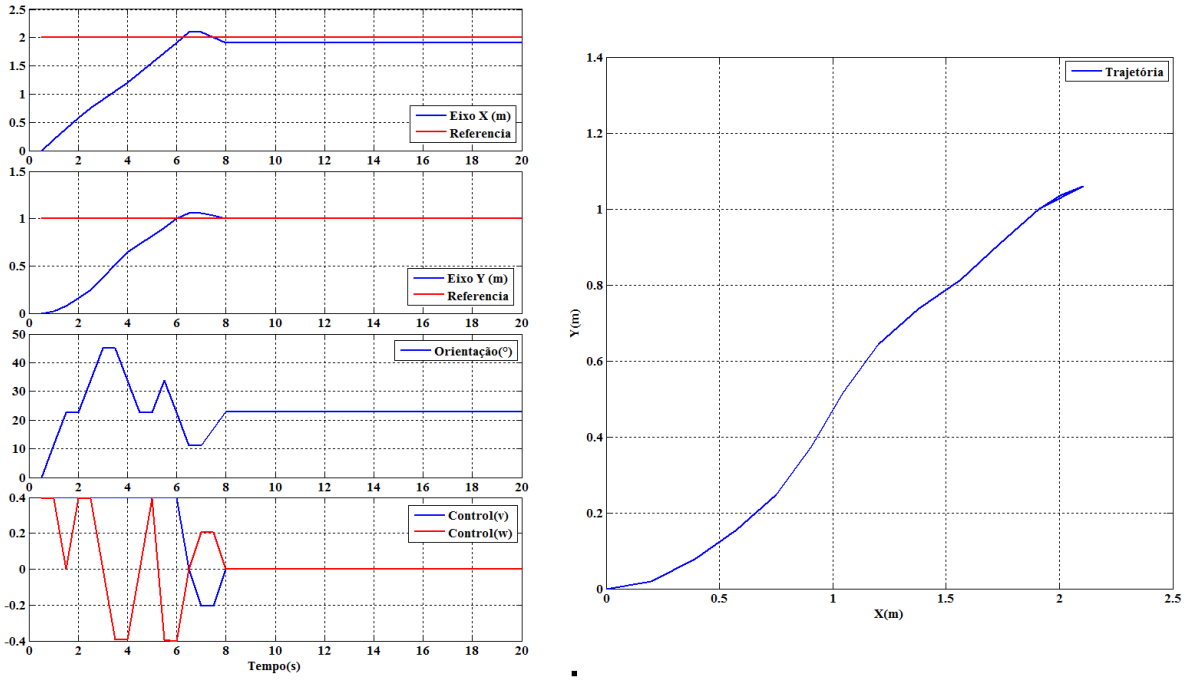


Figura 4.11: Resposta ao degrau de (2,1) sintonizado com $h_p = 3$ e $Q_y = \begin{bmatrix} 7 & 0 \\ 0 & 0{,}4 \end{bmatrix}$.

Uma nova simulação foi realizada, agora para um degrau na coordenada (1,3) e observa-se que o sistema estabilizou com um grande erro de *offset*, sendo necessária uma nova sintonia, Figura 4.12, este fato ocorreu todas as vezes que houve uma mudança no *offset*.

O tempo de cálculo *off-line* do controlador foi de 4,5s e o tempo para fazer os cálculos online foi de 55,8ms por amostragem.

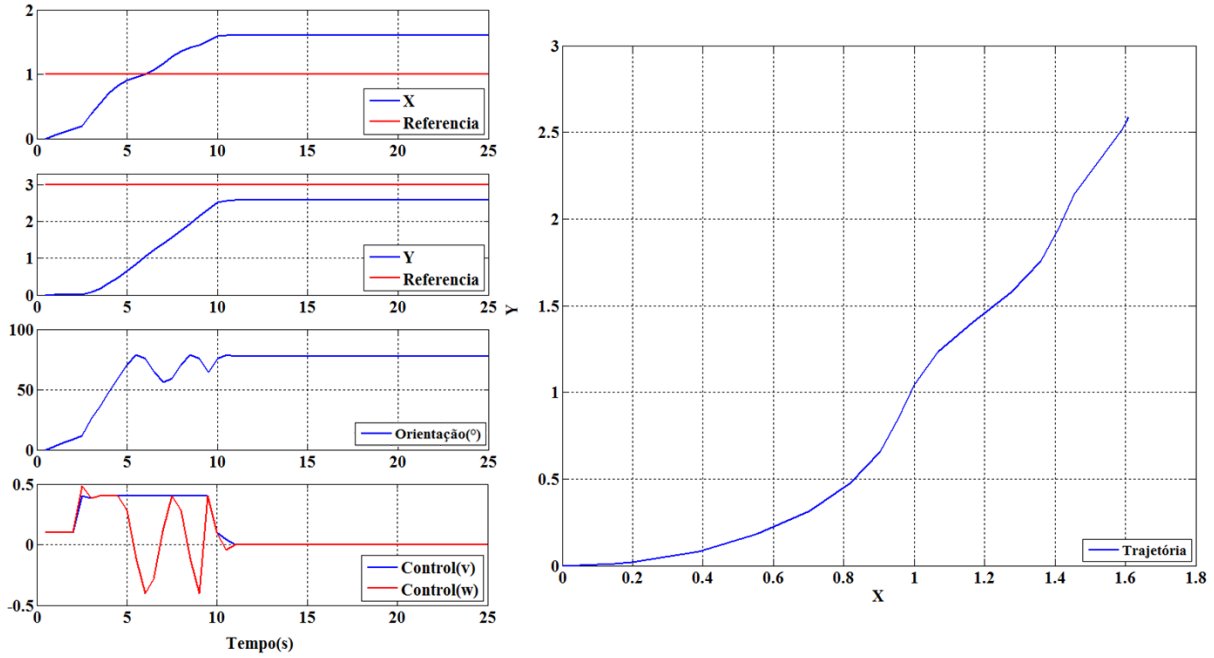


Figura 4.12: Resposta ao degrau de (1,2).

A Figura 4.13 mostra uma comparação da trajetória gerada por este controlador e uma gerada pela lei de controle de Lyapunov, equação (2.7). A trajetória proveniente da lei de controle de Lyapunov é mais linear e o algoritmo de controle necessitou de 63 iterações para atingir o regime, gastando 3ms para realizar os cálculos de cada iteração e consequentemente 0,2s para realizar toda a simulação. A trajetória gerada pelo controlador dinâmico preditivo precisou de apenas 9 iterações, sendo o tempo necessário para o cálculo de cada uma delas de 55,8ms, utilizando 0,5s para que o sistema atingisse o regime, porém com um pequeno erro.

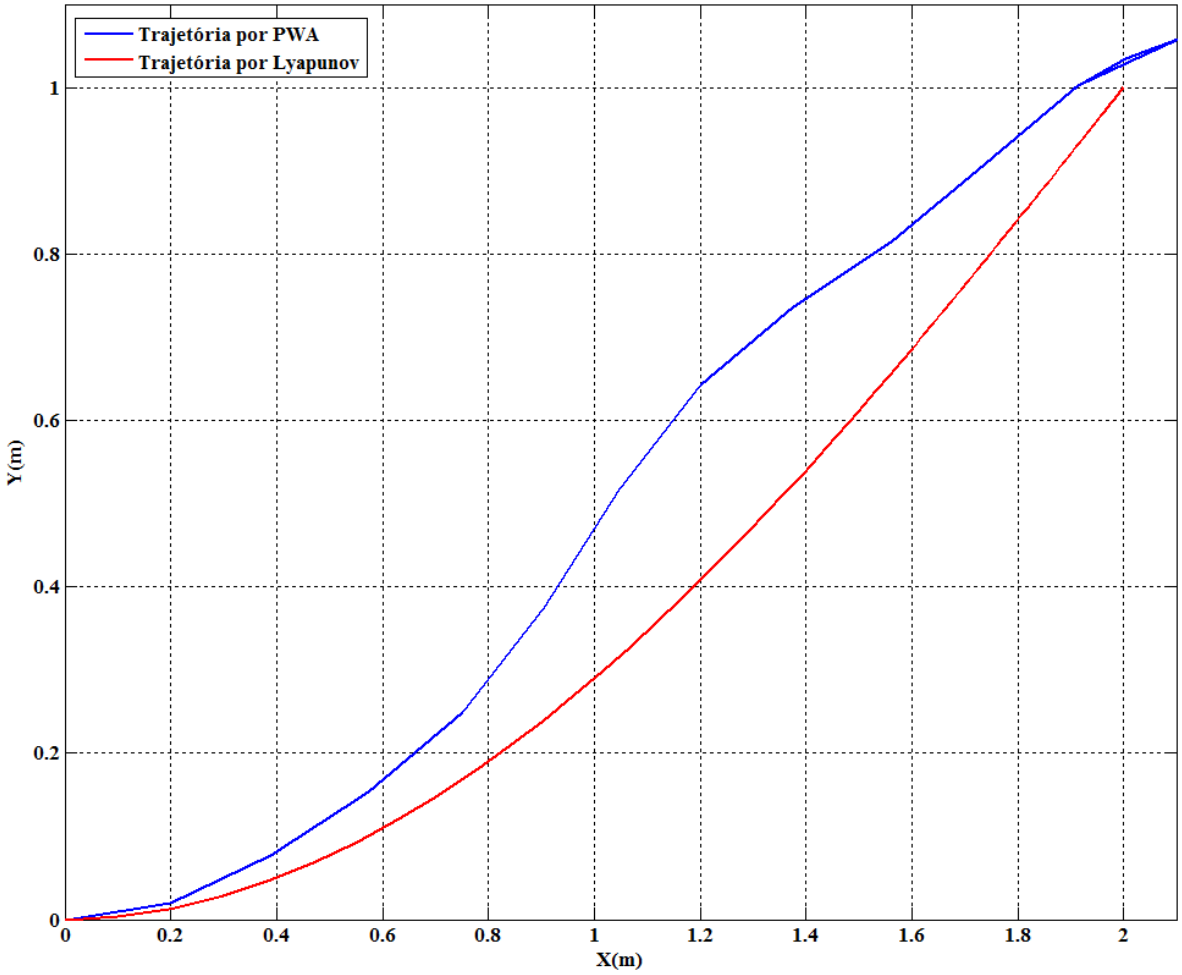


Figura 4.13: Comparação entre trajetória feita por Lyapunov vs feita por PWA.

As simulações realizadas com o controlador cinemático encontram-se no apêndice D

4.3.2 Características do controlador cinemático

Para o controlador cinemático desenvolvido, os estados foram separados em 94.629 regiões, Figura 4.14, e como este sistema possui 3 estados, não é viável plotar o gráfico das ações de controle, pois seria necessária uma quarta dimensão para representá-la.

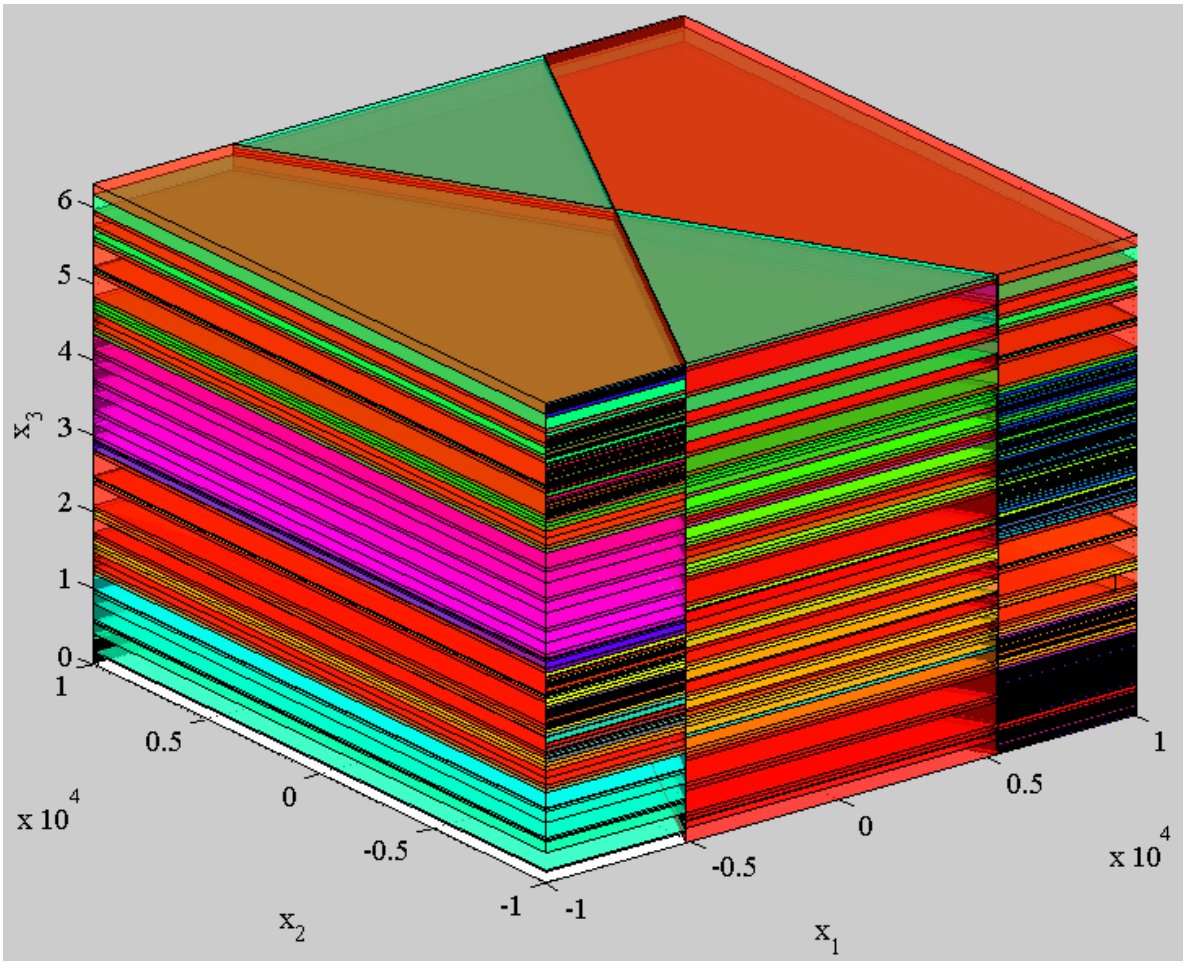


Figura 4.14: Regiões do espaço de estados.

Neste exemplo, as ações de controle são iguais a:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{cases} \begin{bmatrix} -1{,}85 & -0{,}97 & 0 \\ -0{,}12 & 0{,}60 & -0{,}67 \end{bmatrix} X + \begin{bmatrix} 5{,}64 \\ -0{,}67 \end{bmatrix} \; se \; \begin{bmatrix} 0{,}89 & -0{,}47 & 0 \\ -0{,}89 & -0{,}47 & 0 \\ 0{,}93 & 0{,}18 & 0{,}32 \\ -0{,}08 & 0{,}41 & 0{,}91 \\ -0{,}88 & 0{,}47 & 0{,}04 \\ 0 & 0 & -1 \end{bmatrix} X \leq \begin{bmatrix} 0{,}84 \\ -2{,}70 \\ 2{,}35 \\ 0{,}99 \\ -0{,}79 \\ -0{,}34 \end{bmatrix} \\ Região\ 01 \\ \begin{bmatrix} -1{,}85 & -0{,}97 & 0 \\ -0{,}12 & 0{,}60 & -0{,}67 \end{bmatrix} X + \begin{bmatrix} 5{,}64 \\ -0{,}67 \end{bmatrix} \; se \; \begin{bmatrix} 0{,}89 & -0{,}47 & 0 \\ -0{,}89 & -0{,}47 & 0 \\ 0{,}93 & 0{,}18 & 0{,}32 \\ 0 & 0 & 1 \\ -0{,}88 & 0{,}46 & -0{,}08 \end{bmatrix} X \leq \begin{bmatrix} 0{,}84 \\ -2{,}70 \\ 2{,}35 \\ 0{,}34 \\ -0{,}86 \end{bmatrix} \\ \vdots \quad \vdots \quad Região\ 02 \quad \vdots \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -2 \end{bmatrix} X + \begin{bmatrix} 0 \\ 12{,}57 \end{bmatrix} \; se \; \begin{bmatrix} -0{,}88 & -0{,}47 & 0{,}04 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ -0{,}88 & 0{,}47 & -0{,}04 \\ 0 & 0 & -1 \\ 0{,}89 & -0{,}47 & -0{,}01 \end{bmatrix} X \leq \begin{bmatrix} -4{,}50 \\ 6{,}28 \\ 10000 \\ -0{,}12 \\ -6{,}27 \\ 0{,}68 \end{bmatrix} \\ Região\ 94.629 \end{cases},$$

onde $X = [x \; y \; \varphi]^T$.

Capítulo 5: CONCLUSÃO

O controlador dinâmico foi desenvolvido baseado em um modelo dinâmico da cadeira de rodas, onde, os *setpoints* são valores de velocidade linear e angular provenientes do controlador cinemático e suas saídas são o *setpoint* do controlador de baixo nível. Para garantir o conforto e a segurança do passageiro, restrições nas velocidades foram incorporadas ao controlador.

Como o modelo dinâmico possui não linearidades, foi feita uma primeira linearização em torno de apenas um ponto de operação. O resultado se mostrou muito satisfatório, não havendo necessidade de novas linearizações em outros pontos de operação. De posse do modelo dinâmico linear foi projetado seu controlador por intermédio do *MPT*, que utiliza como algoritmo de otimização a Programação Multiparamétrica. A PM se caracteriza por gerar as leis de controle explicitamente, tornando o controle *online* mais rápido.

O controlador dinâmico obtido via PM possui 14 regiões pertencentes ao espaço de estados, onde cada região é regida por uma lei de controle específica. Pôde ser observado por meio do gráfico de politopos, que toda a região dos estados é factível, garantindo assim a estabilidade do sistema. O cálculo para gerar o controlador explícito demandou aproximadamente 4,3s de tempo computacional, lembrando que, uma vez que a solução é armazenada, o controle é feito em tempo real.

Para testar o controlador, foram realizadas simulações com tempo de amostragem de 0,1s, onde as entradas do sistema foram dadas na forma de degraus. Após a devida sintonia, o controlador apresentou uma resposta muito satisfatória, utilizando 35ms para determinar qual ação de controle deveria ser aplicada por instante de tempo, sem erro de *offset* e com um erro total de 10%.

O uso do método de otimização por Programação Multiparamétrica, se mostrou uma boa alternativa na modelagem de um controle preditivo aplicado ao sistema dinâmico. Ele garantiu a estabilidade do sistema, forneceu as ações de controle em tempo mais que suficiente, obedeceu às restrições impostas e convergiu sem erro em regime. A princípio, seu desempenho o torna apto para testes em uma plataforma real.

O controlador cinemático foi projetado com base no modelo cinemático da cadeira de rodas. Este sistema possui três entradas, coordenada x , y e um ângulo φ de referência, que são

fornecidas pelo usuário; e suas saídas, velocidade linear e angular, são a referência do sistema dinâmico.

Assim, como no modelo dinâmico, o cinemático teve de ser linearizado, porém, o resultado para um ponto de operação ficou muito aquém do desejado, sendo necessária a inclusão de novos pontos de operação. Achar os pontos de operação se mostrou uma tarefa árdua e desgastante, pois, por se tratarem de *senos* e *cossenos*, todo o círculo trigonométrico teve de ser abrangido de uma forma que, a quantidade de pontos fosse suficiente para gerar a trajetória solicitada sem utilizar muito poder computacional. Por fim, o modelo cinemático foi linearizado em 48 pontos de operação.

Inicialmente, o modelo cinemático foi representado por um sistema *PWA* com 48 setores. Isto resultou em um controlador com 94.629 regiões de espaço de estados ou politopos. O cálculo *off-line* para a obtenção deste controlador demandou 5,4 min de tempo computacional. Observou-se na primeira simulação com este controlador *online* que o tempo para realizar uma iteração foi de 1,2 min, concluindo que este controlador pode não ser viável quando se requer a geração de *setpoint* de maneira rápida.

Na segunda tentativa foi obtido um modelo dinâmico *PWA* somente para o primeiro quadrante do círculo trigonométrico, considerando 12 setores. O tempo gasto para realizar os cálculos *off-line* deste controlador foi de 4,5 min, e na simulação *online*, cada iteração foi realizada no tempo de 55,8ms, podendo ser viável sua aplicação na cadeira de rodas.

Considerando o modelo cinemático definido no primeiro quadrante, foram feitos testes de mudança de coordenadas (*setpoint*), onde observou-se um grande erro de *offset*, tendo de se realizar nova sintonia. Tal fato ocorreu sempre que foi feita uma mudança drástica na entrada.

O comparativo entre a trajetória controlada pelo controlador via Lyapunov e o controlador via MP não pôde ser melhor explorada devido ao fato do modelo cinemático ter sido restringido apenas para o primeiro quadrante do círculo trigonométrico. A *priori*, o controlador via Lyapunov se mostrou mais rápido, além de ter gerado uma trajetória mais eficiente. Porém, um estudo mais aprofundado na modelagem do controlador cinemático pode trazer resultados mais satisfatórios, principalmente pelo fato do contralador via MP ter utilizado menos iterações para fazer com que o sistema convergisse ao seu destino.

Por conta do ocorrido, surgiram algumas idéias de trabalhos futuros. Em uma delas a abordagem do modelo cinemático seria diferente, transformando-o de coordenada retangular para polar. Talvez com isso a quantidade de linearizações seja inferior, acarretando em menos

cálculos. Outra abordagem envolvendo o controlador cinemático seria a inclusão de algoritmo genético ou redes neurais para estimar todas as sintonias possíveis, uma vez que, a cada novo degrau, uma nova sintonia foi necessária.

A sugestão para o controlador dinâmico envolve sua implementação e posterior comparação com o sistema de controle original de uma cadeira de rodas robótica, verificando de fato sua eficiência.

Apesar do resultado satisfatório na sintonia do controlador dinâmico, o ajuste feito nesse trabalho foi por tentativa e erro, não podendo assim afirmar que este ajuste foi ótimo. Portanto outra linha de pesquisa poderia focar-se em otimizar o controlador, melhorando ainda mais seu desempenho.

Uma alternativa para o modelo dinâmico seria considerar suas saídas como sendo sinais de tensão. Esses sinais alimentariam diretamente os motores, eliminando assim a necessidade de PID.

BIBLIOGRAFIA

AGOSTINO, S. et al. **Classification of unmanned aerial vehicles**. Material didático. Departamento de Engenharia mecânica. Universidade de Adelaide. Austrália.

BARRIENTOS, A.; PEÑÍN, L. F.; BALAGUER, C. **Fundamentos de Robótica**. 2ª. ed. [S.l.]: McGraw-Hill, 1997.

BEMPORAD, A. et al. The Explicit Linear Quadratic Regulator for Contrained Systems. **Automática**, v. 38(1), p. 3-20, Janeiro 2002.

BEMPORAD, A.; MORARI, M. Control of Systems Integrating Logic, Dynamics, and Constraints. **Automatica**, v. 35(3), p. 407-427, Março 1999.

BORRELLI, F. **Constrained Optimal Control of Linear and Hybrid Systems**. 1ª. ed. [S.l.]: Springer, v. 290 do Lecture Notes in Controle and Information Sciences, 2003.

CAMACHO, E. F.; BORDONS, C. **Model Predictive Control**. 2ª. ed. [S.l.]: Springer, 2004.

CELESTE, W. C. **Um sistema autônomo para navegação de cadeiras de rodas robóticas orientadas a pessoas com deficiência motora severa**. Tese de doutorado. PPGE. UFES. [S.l.]. 2009.

CHEN, X.; LI,. Neural Network Predictive Control for Mobile Robot Using PSO with Controllable Random Exploration Velocity. **International Journaol Of Intelligent Control And Systems**, v. 12, n. 3, p. 217-229, Set. 2007.

CHIOU, J.-S.; WANG, K.-Y. Application of a hybrid controller to a mobile robot. **Simulation Modelling Practice and Theory**, v. 16, p. 783-795, 2008.

CLARKE, D. W.; MOHTADI, C.; TUFFS, P. S. Generalized predictive control.part 1: The basic algorithm. part 2: Extensions and interpretations. **Automatica**, v. 23, p. 137–160, 1987.

CLARKE, D. W.; ORDYS, A. W. A state-space description for GPC controllers. **International Journal of Systems Science**, v. 24, p. 1727–1744, 1993.

CRUZ, C. D. L. **Controle de um Robô Móvel de Tração Diferencial**. Material didático. PPGE. UFES. [S.l.]. 2009.

CUTLER, C. R.; RAMAKER, B. L. **Dynamic matrix control-a computer control algorithm**. Presented at the Meeting of the American Institute of Chemical Engineers. Houston, Texas: [s.n.]. 1979.

FALCONE, P. et al. Predictive Active Steering Control for Autonomous Vehicle Systems. **IEEE Transactions On Control Systems Technology**, v. 15, n. 3, p. 566-580, maio 2007.

FLYNN, A. M.; JONES, J. L.; SEIGER, B. A. **Mobile robots inspiration to implementation**. 2ª. ed. [S.l.]: CRC Press, 1998.

GARCIA, C. E.; MORSHEDI, A. M. Quadratic programming solution to Dynamic Matrix Control (QDMC). **Chemical Engineering Communication**, v. 46, p. 73–87, 1986.

GRIEDER, P. et al. Low Complexity Control of Piecewise Affine Systems with Stability Guarantee. **American Control Conference**, Boston. USA, p. 1196-1201, 2004.

GRIEDER, P. et al. Stability low complexity feedback control of constrained piecewise affine systems. **Automatica**, v. 41(10), p. 1683-1694, 2005.

HUTCHINSON, S.; SPONG, M. W.; VIDYASAGAR, M. **Robot modeling and control**. [S.l.]: Editora Wiley, 2005.

KANJANAWANISHKUL, K.; ZELL, A. Distributed Model Predictive Control for Coordinated Path Following Control of Omnidirectional Mobile Robots. **IEEE International Conference on Systems, Man and Cybernetics**, p. 3120-3125, Out 2008.

KEVICZKY, T. et al. Decentralized Receding Horizon Control and Coordination of Autonomous Vehicle Formations. **IEEE Transactions On Control Systems Technology**, v. 16, n. 1, p. 19-32, 2008.

KLANCAR, G.; SKRJANC, I. Tracking-error model-based predictive control for mobile robots in real time. **Robotics and Autonomous Systems 55**, p. 460-469, 2007.

KÜHNE, F.; DA SILVA, J. M. G.; LAGES, W. F. Mobile Robot Trajectory Tracking Using Model Predictive Control. **IEEE Latin-American Robotics Symposium**, Set. 2005.

KVASNICA, M. **Real-Time Model Predictive Control via Multi-Parametric Programming**. 1ª. ed. Saarbrücken, Alemanha: VDM, 2009.

KVASNICA, M.; GRIEDER, P.; BAOTI, M. Multi-Parametric Toolbox, 2010. Disponivel em: <http://control.ee.ethz.ch/~mpt/>. Acesso em: Julho 2012.

MARCHI, J. **Navegação de robôs móveis autônomos Estudo e implementação de abordagens.** Dissertação de Mestrado. UFSC. [S.l.]. 2001.

MAYNE, D. Q. et al. Constrained model predictive control: Stability and optimality. **Automatica**, v. 36, p. 789-914, 2000.

MIURA, R. O. **Controle Preditivo de um Sistema de Levitação Magnética**. Dissertação de Mestrado. ITA. [S.l.]. 2003.

NIÑO, C. H. V. **Estudo de um veículo autônomo submersível alimentado por energia solar**. Dissertação de Mestrado. UFRJ. [S.l.]. 2009.

NOURBAKHSH, I. R.; SIEGWART, R. **Introduction to Autonomous Mobile Robots**. Cambridge: The MIT Press, 2004.

PALÚ, F. **Controle preditivo de colunas de absorção com o metodo de controle por matriz dinamica**. Tese de Doutorado. UNICAMP. [S.l.]. 2001.

PIERI, E. R. D. **Curso de robótica móvel**. Material didático. Dep. Engenaria Elétrica. UFSC. [S.l.]. 2002.

SHINYA, A.; MUNEAKI, I.; TADANAO, Z. Model predictive control based optimal crusing control of two-wheeled mobile robots. **Robotics Automation and Mechatronics (RAM), IEEE Conference**, p. 170-175, Jun. 2010.

SICILIANO, B.; KHATIB, O. **Springer handbook of robotics**. Berlin: Springer, 2008.

SILVA, R. L. **Desenvolvimento de uma Interface Homem-Máquina Aplicada a uma Cadeira de Rodas Robótica por Meio de PDA**. Dissertação de Mestrado. UFES. [S.l.]. 2007.

TEIXEIRA, J. **Cérebro & Cognição**. [S.l.]: Editora Vozes, 2000.

WANG, L. **Model Predictive Control System Design and Implementation Using MATLAB**. Melbourne. Australia: Springer-Verlag London Limited, v. 1, 2009.

WEINKELLER, G. P. C.; SALLES, J. L. F.; BASTOS, T. F. Predictive Control via Multi-Parametric Programming Applied to the Dynamic Model of a Robotic Wheelchair. **SBR/LARS**, 2012.

WIT, C. C. D.; SICILIANO, B.; BASTIN, G. **Theory of Robot Control.** [S.l.]: Great Britain: Springer-Verlag London Limited, 1997.

YOO, S. J.; CHOI, Y. H.; PARK, J. B. Generalized Predictive Control Based on Self-Recurrent Wavelet Neural Network for Stable Path Tracking of Mobile Robots: Adaptive Learning Rates Approach. **IEEE Transactions On Circuits And Systems**, v. 53, n. 6, p. 1381-1394, Jun 2006.

APÊNDICES

APÊNDICE A

SIMULAÇÕES DO MODELO DINÂMICO

SEM AS PERTURBAÇÕES EXTERNAS

```
function não_linear(tfinal,aux,v,w)

  global vref wref vb wb

  close all;

  vref=0;
  wref=0;
  vb=v;
  wb=w;

  % vetor estado: velocidade linear, velocidade angular
  Xs = [0 0 0 0 0 0];

  % tempo de amostragem
  Dt = 0.1;

  % index
  passo = 0;
  tempo = 0;
  trand1 = 0;
  trand2 = 0;
  sre=[0;0];
  slin=[0;0];
  if (aux==1)
     while (tempo < tfinal)
%     PRBS

        if (tempo > trand1)
           trand1 = trand1 + 10*rand(1);
           if (rand(1) > 0.8)
              vref = 0.4;
           elseif (rand(1) > 0.6 && rand(1) <= 0.8)
              vref = 0.2;
           elseif (rand(1) > 0.4 && rand(1) <= 0.6)
              vref = 0;
           elseif (rand(1) > 0.2 && rand(1) <= 0.4)
              vref = -0.2;
           else
              vref=-0.4;
           end
        end

        passo = passo + 1;
        ts = passo*Dt;

        % Integraçao
        [T,X] = ode45('não_linear_sis',[tempo ts], Xs);

        % Armazenamento de 90ineariza
        n = length(T);
        tempo = ts;
        Xs = X(n,:);

        vetvref(passo, :) = vref;
        vetwref(passo, :) = wref;
```

```
        vetout(passo,:) = [X(n,1) X(n,2) X(n,3) X(n,4) X(n,5) X(n,6)];
        vettime(passo) = T(n, :);

        não=não+abs([X(n,1);X(n,2)]);
        slin=slin+abs([X(n,5);X(n,6)]);
    end
    %calculo do erro
    erro=(não-slin)*100/não;
 elseif (aux==2)
    while (tempo < tfinal)
%    PRBS

        if (tempo > trand2)
            trand2 = trand2 + 10*rand(1);
            if (rand(1) >0.8)
                wref = 0.5;
            elseif (rand(1) > 0.6 && rand(1) <= 0.8)
                wref = 0.25;
            elseif (rand(1) >0.4 && rand(1) <= 0.6)
                wref = 0;
            elseif (rand(1) > 0.2 && rand(1) <= 0.4)
                wref = -0.25;
            else
                wref=-0.5;
            end
        end

        passo = passo + 1;
        ts = passo*Dt;

        % Integraçao
        [T,X] = ode45('não_linear_sis',[tempo ts], Xs);

        % Armazenamento de 91ineariza
        n = length(T);
        tempo = ts;
        Xs = X(n, :);

        vetvref(passo, :) = vref;
        vetwref(passo, :) = wref;
        vetout(passo,:) = [X(n,1) X(n,2) X(n,3) X(n,4) X(n,5) X(n,6)];
        vettime(passo) = T(n,:);

        não=não+abs([X(n,1);X(n,2)]);
        slin=slin+abs([X(n,5);X(n,6)]);
    end
    %calculo do erro
    erro=(não-slin)*100/não;
 else
    while (tempo < tfinal)
%    PRBS para o modelo de 91inea

        if (tempo > trand1)
            trand1 = trand1 + 10*rand(1);
            if (rand(1) > 0.8)
                vref = 0.4;
            elseif (rand(1) > 0.6 && rand(1) <= 0.8)
                vref = 0.2;
            elseif (rand(1) > 0.4 && rand(1) <= 0.6)
```

```
            vref = 0;
        elseif (rand(1) > 0.2 && rand(1) <= 0.4)
            vref = -0.2;
        else
            vref=-0.4;
        end
    end

    if (tempo > trand2)
        trand2 = trand2 + 10*rand(1);
        if (rand(1) >0.8)
            wref = 0.5;
        elseif (rand(1) > 0.6 && rand(1) <= 0.8)
            wref = 0.25;
        elseif (rand(1) >0.4 && rand(1) <= 0.6)
            wref = 0;
        elseif (rand(1) > 0.2 && rand(1) <= 0.4)
            wref = -0.25;
        else
            wref=-0.5;
        end
    end

    passo = passo + 1;
    ts = passo*Dt;

    % Integraçao
    [T,X] = ode45('não_linear_sis',[tempo ts], Xs);

    % Armazenamento de 92ineariza
    n = length(T);
    tempo = ts;
    Xs = X(n,:);

    vetvref(passo, :) = vref;
    vetwref(passo, :) = wref;
    vetout(passo,:) = [X(n,1) X(n,2) X(n,3) X(n,4) X(n,5) X(n,6)];
    vettime(passo) = T(n,:);

    não=não+abs([X(n,1);X(n,2)]);
    slin=slin+abs([X(n,5);X(n,6)]);
  end
  %calculo do erro
  erro=abs(não-slin)*100/não;
end

f = figure('name','Velocidade Linear');
subplot(211)
plot(vettime, vetvref,'b');
title('vref');
hold on;
plot(vettime, vetout(:,1),'r');
title('Velocidade linear sem carga');
axis([0 tfinal -0.5 0.5]);
xlabel('s');ylabel('m/s');
legend('Referência','real');
grid;
hold off;
subplot(212)
```

```
plot(vettime, vetvref,'b');
title('vref');
hold on;
plot(vettime, vetout(:,3),'r');
title('Velocidade linear com carga pesada');
axis([0 tfinal -0.5 0.5]);
xlabel('s');ylabel('m/s');
legend('Referência','real');
grid;
hold off;

f = figure('name','Velocidade Angular');
subplot(211)
plot(vettime, vetwref,'b');
title('wref');
hold on;
plot(vettime, vetout(:,2),'r');
title('Velocidade angular sem carga');
axis([0 tfinal -0.6 0.6]);
xlabel('s');ylabel('rad/s');
legend('Referência','real');
grid;
hold off;
subplot(212)
plot(vettime, vetwref,'b');
title('wref');
hold on;
plot(vettime, vetout(:,4),'r');
title('Velocidade angular com carga pesada');
axis([0 tfinal -0.6 0.6]);
xlabel('s');ylabel('rad/s');
legend('Referência','real');
grid;
hold off;

f = figure('name','Linearização');
subplot(211)
plot(vettime, vetvref,'b');
title('vref');
hold on;
plot(vettime, vetout(:,5),'r');
title(['Velocidade linear LINEARIZADA sem carga entorno de v=', num2str(vb) ,' e de w=', num2str(wb)]);
axis([0 tfinal -0.5 0.5]);
xlabel('s');ylabel('m/s');
legend('Referência','real');
grid;
hold off;
subplot(212)
plot(vettime, vetwref,'b');
title('wref');
hold on;
plot(vettime, vetout(:,6),'r');
title(['Velocidade angular LINEARIZADA sem carga entorno de v=', num2str(vb) ,' e de w=', num2str(wb)]);
axis([0 tfinal -0.6 0.6]);
xlabel('s');ylabel('rad/s');
legend('Referência','real');
grid;
hold off;

f = figure('name','Comparação');
```

```
subplot(221)
plot(vettime, vetout(:,1),'b');
hold on;
plot(vettime, vetout(:,5),'r');
title(['Velocidade linear X velocidade linear LINEARIZADA sem carga entorno de v=', num2str(vb) ,' e de
w=', num2str(wb)]);
axis([0 tfinal -0.5 0.5]);
xlabel('s');ylabel('m/s');
legend('Real','Linearizada');
grid;
hold off;
subplot(223)
plot(vettime, vetvref,'b');
axis([0 tfinal -0.5 0.5]);
title('vref');
grid
subplot(222)
plot(vettime, vetout(:,2),'b');
title('wref');
hold on;
plot(vettime, vetout(:,6),'r');
title(['Velocidade angular X velocidade angular LINEARIZADA sem carga entorno de v=', num2str(vb) ,' e de
w=', num2str(wb)]);
axis([0 tfinal -0.6 0.6]);
xlabel('s');ylabel('rad/s');
legend('Real','Linearizada');
grid;
hold off;
subplot(224)
plot(vettime, vetwref,'b');
axis([0 tfinal -0.6 0.6]);
title('wref');
grid
clc
disp(['Erro para velocidade linear: ',num2str(erro(1,2)),'%']);
disp(['Erro para velocidade angular: ',num2str(erro(2,2)),'%']);
```

FUNÇÕES

Função não_linear_sis

```
function d = não_linear_sis(t,x);
  global vref wref vb wb

  teta;

  lin=95inearização(teta1,teta2,x);

  d(1)=teta1(1,15)*vref+teta1(1,16)*x(2)^2-teta1(1,17)*x(1)+teta1(1,18)*wref-teta1(1,19)*x(1)*x(2)-
teta1(1,20)*x(2);
  d(2)=teta1(1,21)*wref-teta1(1,22)*x(1)*x(2)-teta1(1,23)*x(2)+teta1(1,24)*vref+teta1(1,25)*x(2)^2-
teta1(1,26)*x(1);
  d(3)=teta2(1,15)*vref+teta2(1,16)*x(4)^2-teta2(1,17)*x(3)+teta2(1,18)*wref-teta2(1,19)*x(3)*x(4)-
teta2(1,20)*x(4);
  d(4)=teta2(1,21)*wref-teta2(1,22)*x(3)*x(4)-teta2(1,23)*x(4)+teta2(1,24)*vref+teta2(1,25)*x(4)^2-
teta2(1,26)*x(3);
  d(5)=lin(1);
  d(6)=lin(2);

  d = d';
```

Função 95inearização

```
function lin = 95inearização(teta1,teta2,x);
global vref wref vb wb

%%LINEARIZANDO NOS PONTOS vb e wb
% vp=teta15*vref + teta16*w^2 – teta17*v + teta18*wref – teta19*v*w – teta20*w
% vplin(v,2)=vp(vb,2b)+(-teta17-teta19*wb)*(v-vb)+(2*teta16*wb-teta19*vb-teta20)*(w-wb)
% vplin(v,2)=a1 + b1(v-vb) + c1(w-wb)
a1=teta1(1,15)*vref + teta1(1,16)*wb^2 – teta1(1,17)*vb + teta1(1,18)*wref – teta1(1,19)*vb*wb –
teta1(1,20)*wb;
b1=-teta1(1,17) – teta1(1,19)*wb;
c1=2*teta1(1,16)*wb – teta1(1,19)*vb – teta1(1,20);
% vplin(v,w)=(a1 – b1*vb – c1wb) + b1*v + c1*w
% vplin(v,w)=d1 + b1*v + c1*w
d1=a1 – b1*vb – c1*wb;
lin(1)=d1 + b1*x(5) + c1*x(6);
% wp=teta21*wref – teta22*v*w – teta23*w + teta24*vref + teta25*w^2 – teta26*v
% wplin(v,w)=wp(vb,wb)+(-teta22*wb-teta26)*(v-vb)+(-teta22vb-teta23+2*teta25*wb)*(w-wb)
% wplin(v,w)=a1 + b1(v-vb) + c1(w-wb)
a1=teta1(1,21)*wref – teta1(1,22)*vb*wb – teta1(1,23)*wb + teta1(1,24)*vref + teta1(1,25)*wb^2 –
teta1(1,26)*vb;
b1=-teta1(1,22)*wb – teta1(1,26);
c1=-teta1(1,22)*vb – teta1(1,23) + 2*teta1(1,25)*wb;
% wplin(v,w)=(a1 – b1*vb – c1wb) + b1*v + c1*w
% wplin(v,w)=d1 + b1*v + c1*w
d1=a1 – b1*vb – c1*wb;
lin(2)=d1 + b1*x(5) + c1*x(6);
```

Função teta

```
%criação do vetor teta sem usuário
 teta1=[0.4042 0.1877 -0.0069 1.0261 0.0405 0.9239 -0.0304 -0.1162];
 teta1=[teta1  (teta1(1,3)/teta1(1,1)) (teta1(1,4)/teta1(1,1)) (teta1(1,7)/teta1(1,1)) ...
      (teta1(1,5)/teta1(1,2)) (teta1(1,6)/teta1(1,2)) (teta1(1,8)/teta1(1,2))];
 teta1=[teta1 (1/(teta1(1,1)*(1-teta1(1,11)*teta1(1,14)))) (teta1(1,9)/(1-teta1(1,11)*teta1(1,14))) ...
      (teta1(1,10)/(1-teta1(1,11)*teta1(1,14))) (teta1(1,11)/(teta1(1,2)*(1-teta1(1,11)*teta1(1,14)))) ...
      ((teta1(1,11)*teta1(1,12))/(1-teta1(1,11)*teta1(1,14))) ((teta1(1,11)*teta1(1,13))/(1-
teta1(1,11)*teta1(1,14)))];
 teta1=[teta1 (1/(teta1(1,2)*(1-teta1(1,11)*teta1(1,14)))) (teta1(1,12)/(1-teta1(1,11)*teta1(1,14))) ...
      (teta1(1,13)/(1-teta1(1,11)*teta1(1,14))) (teta1(1,14)/(teta1(1,1)*(1-teta1(1,11)*teta1(1,14)))) ...
      ((teta1(1,14)*teta1(1,9))/(1-teta1(1,11)*teta1(1,14))) ((teta1(1,14)*teta1(1,10))/(1-
teta1(1,11)*teta1(1,14)))];

 %criação do vetor teta com usuário pesado
 teta2=[0.3241 0.0120 0.0092 0.9969 0.0634 0.9898 -0.0562 -0.0002];
 teta2=[teta2  (teta2(1,3)/teta2(1,1)) (teta2(1,4)/teta2(1,1)) (teta2(1,7)/teta2(1,1)) ...
      (teta2(1,5)/teta2(1,2)) (teta2(1,6)/teta2(1,2)) (teta2(1,8)/teta2(1,2))];
 teta2=[teta2 (1/(teta2(1,1)*(1-teta2(1,11)*teta2(1,14)))) (teta2(1,9)/(1-teta2(1,11)*teta2(1,14))) ...
      (teta2(1,10)/(1-teta2(1,11)*teta2(1,14))) (teta2(1,11)/(teta2(1,2)*(1-teta2(1,11)*teta2(1,14)))) ...
      ((teta2(1,11)*teta2(1,12))/(1-teta2(1,11)*teta2(1,14))) ((teta2(1,11)*teta2(1,13))/(1-
teta2(1,11)*teta2(1,14)))];
 teta2=[teta2 (1/(teta2(1,2)*(1-teta2(1,11)*teta2(1,14)))) (teta2(1,12)/(1-teta2(1,11)*teta2(1,14))) ...
      (teta2(1,13)/(1-teta2(1,11)*teta2(1,14))) (teta2(1,14)/(teta2(1,1)*(1-teta2(1,11)*teta2(1,14)))) ...
      ((teta2(1,14)*teta2(1,9))/(1-teta2(1,11)*teta2(1,14))) ((teta2(1,14)*teta2(1,10))/(1-
teta2(1,11)*teta2(1,14)))];
```

COM AS PERTURBAÇÕES EXTERNAS

```
clear all;
global vref wref pert

vref=0;
wref=0;
tfinal=50;

% vetor estado: velocidade linear, velocidade angular e perturbação
Xs = [0 0];
Xs1 = [0 0];
pert = [0 0];

% tempo de amostragem
Dt = 0.1;

% index
passo = 0;
tempo = 0;
trand1 = 0;
trand2 = 0;
trand3 = 0;
trand4 = 0;
sre=[0;0];
slin=[0;0];

while (tempo < tfinal)
%    PRBS

      if (tempo > trand1)
         trand1 = trand1 + 10*rand(1);
         if (rand(1) > 0.8)
            vref = 0.4;
         elseif (rand(1) > 0.6 && rand(1) <= 0.8)
            vref = 0.2;
         elseif (rand(1) > 0.4 && rand(1) <= 0.6)
            vref = 0;
         elseif (rand(1) > 0.2 && rand(1) <= 0.4)
            vref = -0.2;
         else
            vref=-0.4;
         end
      end

      if (tempo > trand2)
         trand2 = trand2 + 10*rand(1);
         if (rand(1) >0.8)
            wref = 0.5;
         elseif (rand(1) > 0.6 && rand(1) <= 0.8)
            wref = 0.25;
         elseif (rand(1) >0.4 && rand(1) <= 0.6)
            wref = 0;
         elseif (rand(1) > 0.2 && rand(1) <= 0.4)
            wref = -0.25;
         else
            wref=-0.5;
```

```
        end
    end

    if (tempo > trand3)
        trand3 = trand3 + 10*rand(1);
        if (rand(1) > 0.8)
            pert(1,1) = 0.02;
        elseif (rand(1) > 0.6 && rand(1) <= 0.8)
            pert(1,1) = 0.02;
        elseif (rand(1) > 0.4 && rand(1) <= 0.6)
            pert(1,1) = 0;
        elseif (rand(1) > 0.2 && rand(1) <= 0.4)
            pert(1,1) = -0.02;
        else
            pert(1,1)=-0.02;
        end
    end

    if (tempo > trand4)
        trand4 = trand4 + 10*rand(1);
        if (rand(1) >0.8)
            pert(1,2) = 0.025;
        elseif (rand(1) > 0.6 && rand(1) <= 0.8)
            pert(1,2) = 0.025;
        elseif (rand(1) >0.4 && rand(1) <= 0.6)
            pert(1,2) = 0;
        elseif (rand(1) > 0.2 && rand(1) <= 0.4)
            pert(1,2) = -0.025;
        else
            pert(1,2)=-0.025;
        end
    end

    passo = passo + 1;
    ts = passo*Dt;

    % Integraçao
    [T,X] = ode45('não_linear_sis_pert',[tempo ts], Xs);
    [T1,X1] = ode45('não_linear_sis1',[tempo ts], Xs1);

    % Armazenamento de 98inearizа
    n = length(T);
    n1 = length(T1);
    tempo = ts;
    Xs = X(n,:);
    Xs1 = X1(n1,:);

    %criação dos vetores
    vetvref(passo,:) = vref;
    vetwref(passo,:) = wref;
    vetout1(passo,:) = [X(n,1) X(n,2)];
    vetout2(passo,:) = [X1(n1,1) X1(n1,2)];
    vettime(passo) = T(n,:);
    vetpert(passo,:) = pert;

    não=não+abs([X1(n1,1);X1(n1,2)]);
    slin=slin+abs([X(n,1);X(n,2)]);
end
%calculo do erro
```

```
    erro=abs(não-slin)*100/não;

f = figure('name','Comparação');
subplot(221)
plot(vettime, vetvref,'b');
title('Velocidade linear com perturbação X velocidade linear LINEARIZADA');
hold on;
plot(vettime, vetout1(:,1),'r');
plot(vettime, vetout2(:,1),'k');
axis([0 tfinal -0.5 0.5]);
xlabel('s');ylabel('m/s');
legend('Referência','Linearizada','Real com perturbação');
grid;
hold off;
subplot(223)
plot(vettime, vetpert(:,1));
title('Perturbação na velocidade linear');
axis([0 tfinal -0.03 0.03]);
xlabel('s');ylabel('nd');
grid
subplot(222)
plot(vettime, vetwref,'b');
title('Velocidade angular com perturbação X velocidade angular LINEARIZADA');
hold on;
plot(vettime, vetout1(:,2),'r');
plot(vettime, vetout2(:,2),'k');
axis([0 tfinal -0.6 0.6]);
xlabel('s');ylabel('rad/s');
legend('Referência','Linearizada','Real com perturbação');
grid;
hold off;
subplot(224)
plot(vettime, vetpert(:,2),'r');
title('Perturbação na velocidade angular');
axis([0 tfinal -0.03 0.03]);
xlabel('s');ylabel('nd');
grid;
clc
disp(['Erro para velocidade linear: ',num2str(erro(1,2)),'%']);
disp(['Erro para velocidade angular: ',num2str(erro(2,2)),'%']);
```

FUNÇÕES

Função não_linear_sis_pert

```
function d = nao_linear_sis_pert(t,x);
  global vref wref pert

  teta;

  d(1)=teta1(1,15)*vref+teta1(1,15)*pert(1,1)+teta1(1,16)*x(2)^2-
teta1(1,17)*x(1)+teta1(1,18)*wref+teta1(1,18)*pert(1,2)-teta1(1,19)*x(1)*x(2)-teta1(1,20)*x(2);
  d(2)=teta1(1,21)*wref+teta1(1,21)*pert(1,2)-teta1(1,22)*x(1)*x(2)-
teta1(1,23)*x(2)+teta1(1,24)*vref+teta1(1,24)*pert(1,1)+teta1(1,25)*x(2)^2-teta1(1,26)*x(1);

  d = d';
```

Função não_linear_sis_pert

```
function d = nao_linear_sis1(t,x);
  global vref wref

  teta;

  lin=linearizacao2(teta1,teta2,x);

  d(1)=lin(1);
  d(2)=lin(2);

  d = d';
```

Função linearizacao2

```
function lin = linearizacao2(teta1,teta2,x);
global vref wref

vb=0;
wb=0;

%%LINEARIZANDO NOS PONTOS vb e wb

% vp=teta15*vref + teta16*w^2 – teta17*v + teta18*wref – teta19*v*w – teta20*w
% vplin(v,2)=vp(vb,2b)+(-teta17-teta19*wb)*(v-vb)+(2*teta16*wb-teta19*vb-teta20)*(w-wb)
% vplin(v,2)=a1 + b1(v-vb) + c1(w-wb)
a1=teta1(1,15)*vref + teta1(1,16)*wb^2 – teta1(1,17)*vb + teta1(1,18)*wref – teta1(1,19)*vb*wb –
teta1(1,20)*wb;
b1=-teta1(1,17) – teta1(1,19)*wb;
c1=2*teta1(1,16)*wb – teta1(1,19)*vb – teta1(1,20);

% vplin(v,w)=(a1 – b1*vb – c1wb) + b1*v + c1*w
% vplin(v,w)=d1 + b1*v + c1*w
```

```
d1=a1 – b1*vb – c1*wb;

lin(1)=d1 + b1*x(1) + c1*x(2);

% wp=teta21*wref – teta22*v*w – teta23*w + teta24*vref + teta25*w^2 – teta26*v
% wplin(v,w)=wp(vb,wb)+(-teta22*wb-teta26)*(v-vb)+(-teta22vb-teta23+2*teta25*wb)*(w-wb)
% wplin(v,w)=a1 + b1(v-vb) + c1(w-wb)
a1=teta1(1,21)*wref – teta1(1,22)*vb*wb – teta1(1,23)*wb + teta1(1,24)*vref + teta1(1,25)*wb^2 –
teta1(1,26)*vb;
b1=-teta1(1,22)*wb – teta1(1,26);
c1=-teta1(1,22)*vb – teta1(1,23) + 2*teta1(1,25)*wb;

% wplin(v,w)=(a1 – b1*vb – c1wb) + b1*v + c1*w
% wplin(v,w)=d1 + b1*v + c1*w
d1=a1 – b1*vb – c1*wb;

lin(2)=d1 + b1*x(1) + c1*x(2);
```

APÊNDICE B

SIMULAÇÕES DO MODELO CINEMÁTICO

COM APENAS UMA PARTIÇÃO

```
clear all;
  global v w fib vb

  v=0;
  w=0;

  % definindo dos pontos de operação
  vb=0;
  fib=pi*0/180;

  % vetor estado: velocidade linear, velocidade angular
  Xs = [0 0 0 0 0];
  fi=Xs(1,3);

  % tempo de amostragem
  Dt = 0.1;

  N_sim=320;
  v=[0.2*ones(1,25) 0.4*ones(1,25) zeros(1,50) -0.2*ones(1,25) -0.4*ones(1,25) zeros(1,50) 0.4*ones(1,50) -
0.4*ones(1,50) zeros(1,20)];
  w=[zeros(1,10) 0.5*ones(1,70) zeros(1,10) -0.5*ones(1,70) zeros(1,10) 0.5*ones(1,70) -0.5*ones(1,70)
zeros(1,10)];

  r=[v;w];
  passo = 0;
  tempo = 0;
  sre=[0;0];
  slin=[0;0];

     for kk=1:N_sim

        v=r(1,kk);
        w=r(2,kk);
        passo = passo + 1;
        ts = passo*Dt;


        % Integraçao
        [T,X] = ode45('sis_nao_linear',[tempo ts], Xs);

        % Armazenamento de 103inearizа
        n = length(T);
        tempo = ts;
        Xs = X(n,:);
        fi=X(n,3);

        vetvref(passo, :) = v;
        vetwref(passo, :) = w;
        vetout(passo,:) = [X(n,1) X(n,2) 180*X(n,3)/pi X(n,4) X(n,5)];
        vettime(passo) = T(n,:);

        não=não+abs([X(n,1);X(n,2)]);
        slin=slin+abs([X(n,4);X(n,5)]);
     end
```

```
    %calculo do erro
    erro(1,1)=abs(não(1,1)-slin(1,1))*100/não(1,1);
    erro(2,1)=abs(sre(2,1)-slin(2,1))*100/sre(2,1);

figure;
plot(vetout(:,1), vetout(:,2),'b');
title('Trajetória real X trajetória linearizada');
hold on;
plot(vetout(:,4), vetout(:,5),'r');
xlabel('X(m)');ylabel('Y(m)');
legend('Tragetória não linear','Tragetória linearizada');
grid;

figure
subplot(411)
plot(vettime, vetout(:,1),'b');
hold on;
plot(vettime, vetout(:,4),'r');
title('X em função do tempo');
xlabel('s');ylabel('X');
legend('não linear','linearizada');
hold off
grid;
subplot(412)
plot(vettime, vetout(:,2),'b');
hold on;
plot(vettime, vetout(:,5),'r');
title('Y em função do tempo');
xlabel('s');ylabel('Y');
legend('não linear','linearizada');
hold off
grid
subplot(413)
plot(vettime, vetout(:,3),'b');
title('FI em função do tempo');
xlabel('s');ylabel('º');
grid;
subplot(414)
plot(vettime, vetvref,'b');
title('wref');
hold on;
plot(vettime, vetwref,'r');
title('Referencia');
xlabel('s');
legend('v','w');
grid;
hold off;
clc
disp(['Erro para deslocamento em X: ',num2str(erro(1,1)),'%']);
disp(['Erro para deslocamento em Y: ',num2str(erro(2,1)),'%']);
```

COM 12 PARTIÇÃO

```
clear all;
global v w fib vb

v=0;
w=0;
% vetor estado: velocidade linear, velocidade angular
Xs = [0 0 0 0 0];
fi=Xs(1,3);

% tempo de amostragem
Dt = 0.1;
% index
N_sim=320;
v=[0.2*ones(1,25) 0.4*ones(1,25) zeros(1,50) -0.2*ones(1,25) -0.4*ones(1,25) zeros(1,50) 0.4*ones(1,50) -
0.4*ones(1,50) zeros(1,20)];
w=[zeros(1,10) 0.5*ones(1,70) zeros(1,10) -0.5*ones(1,70) zeros(1,10) 0.5*ones(1,70) -0.5*ones(1,70)
zeros(1,10)];
r=[v;w];
passo = 0;
tempo = 0;
sre=[0;0];
slin=[0;0];

    for kk=1:N_sim

        v=r(1,kk);
        w=r(2,kk);
        passo = passo + 1;
        ts = passo*Dt;

        if (fi<=pi*90/180 && fi>=pi*0/180)
            fib=pi*45/180;
        elseif (fi<=pi*180/180 && fi>pi*90/180)
            fib=pi*135/180;
        elseif ((fi<pi*270/180 && fi>pi*180/180))
            fib=pi*225/180;
        elseif ((fi<pi*360/180 && fi>pi*270/180))
            fib=pi*315/180;
        end

        if (v>=-0.1 && v<=0.1)
            vb=0;
        elseif (v>0.1 && v<=0.4)
            vb=0.25;
        elseif (v<-0.1 && v>=-0.4)
            vb=-0.25;
        end

        % Integraçao
        [T,X] = ode45('sis_nao_linear',[tempo ts], Xs);

        % Armazenamento de 105inearizа
        n = length(T);
        tempo = ts;
        Xs = X(n,:);
        fi=X(n,3);
```

```
        vetvref(passo, :) = v;
        vetwref(passo, :) = w;
        vetout(passo,:) = [X(n,1) X(n,2) 180*X(n,3)/pi X(n,4) X(n,5)];
        vettime(passo) = T(n,:);

        não=não+abs([X(n,1);X(n,2)]);
        slin=slin+abs([X(n,4);X(n,5)]);
    end

    %calculo do erro
    erro(1,1)=abs(não(1,1)-slin(1,1))*100/não(1,1);
    erro(2,1)=abs(não(2,1)-slin(2,1))*100/não(2,1);

figure;
plot(vetout(:,1), vetout(:,2),'b');
title('Trajetória real X trajetória linearizada');
hold on;
plot(vetout(:,4), vetout(:,5),'r');
xlabel('X(m)');ylabel('Y(m)');
legend('Tragetória não linear','Tragetória linearizada');
grid;

figure
subplot(411)
plot(vettime, vetout(:,1),'b');
hold on;
plot(vettime, vetout(:,4),'r');
title('X em função do tempo');
xlabel('s');ylabel('X');
legend('não linear','linearizada');
hold off
grid;
subplot(412)
plot(vettime, vetout(:,2),'b');
hold on;
plot(vettime, vetout(:,5),'r');
title('Y em função do tempo');
xlabel('s');ylabel('Y');
legend('não linear','linearizada');
hold off
grid
subplot(413)
plot(vettime, vetout(:,3),'b');
title('FI em função do tempo');
xlabel('s');ylabel('º');
grid;
subplot(414)
plot(vettime, vetvref,'b');
title('wref');
hold on;
plot(vettime, vetwref,'r');
title('Referencia');
xlabel('s');
legend('v','w');
grid;
hold off;
clc
disp(['Erro para velocidade linear: ',num2str(erro(1,1)),'%']);
```

```
  disp(['Erro para velocidade angular: ',num2str(erro(2,1)),'%']);
```

COM 48 PARTIÇÃO

```
clear all;
  global v w fib vb

  v=0;
  w=0;

  % definindo não máximo por pedaço de linearização
  aux_fi=pi*22.5/180;

  % vetor estado: velocidade linear, velocidade angular
  Xs = [0 0 0 0 0];
  fi=Xs(1,3);

  % tempo de amostragem
  Dt = 0.1;

  % index
  N_sim=320;
  v=[0.2*ones(1,25) 0.4*ones(1,25) zeros(1,50) -0.2*ones(1,25) -0.4*ones(1,25) zeros(1,50) 0.4*ones(1,50) -
0.4*ones(1,50) zeros(1,20)];
  w=[zeros(1,10) 0.5*ones(1,70) zeros(1,10) -0.5*ones(1,70) zeros(1,10) 0.5*ones(1,70) -0.5*ones(1,70)
zeros(1,10)];

  r=[v;w];
  passo = 0;
  tempo = 0;
  sre=[0;0];
  slin=[0;0];


     for kk=1:N_sim

        v=r(1,kk);
        w=r(2,kk);
        passo = passo + 1;
        ts = passo*Dt;

        if (fi<=aux_fi && fi>=0)
           fib=aux_fi/2;

        elseif ((fi<=2*aux_fi && fi>aux_fi))
           fib=aux_fi+(aux_fi/2);

        elseif ((fi<=3*aux_fi && fi>2*aux_fi))
           fib=2*aux_fi+(aux_fi/2);

        elseif ((fi<=4*aux_fi && fi>3*aux_fi))
           fib=3*aux_fi+(aux_fi/2);

        elseif ((fi<=5*aux_fi && fi>4*aux_fi))
           fib=4*aux_fi+(aux_fi/2);
```

```
elseif ((fi<=6*aux_fi && fi>5*aux_fi))
   fib=5*aux_fi+(aux_fi/2);

elseif ((fi<=7*aux_fi && fi>6*aux_fi))
   fib=6*aux_fi+(aux_fi/2);

elseif ((fi<=8*aux_fi && fi>7*aux_fi))
   fib=7*aux_fi+(aux_fi/2);

elseif ((fi<=9*aux_fi && fi>8*aux_fi))
   fib=8*aux_fi+(aux_fi/2);

elseif ((fi<=10*aux_fi && fi>9*aux_fi))
   fib=9*aux_fi+(aux_fi/2);

elseif ((fi<=11*aux_fi && fi>10*aux_fi))
   fib=10*aux_fi+(aux_fi/2);

elseif ((fi<=12*aux_fi && fi>11*aux_fi))
   fib=11*aux_fi+(aux_fi/2);

elseif ((fi<=13*aux_fi && fi>12*aux_fi))
   fib=12*aux_fi+(aux_fi/2);

elseif ((fi<=14*aux_fi && fi>13*aux_fi))
   fib=13*aux_fi+(aux_fi/2);

elseif ((fi<=15*aux_fi && fi>14*aux_fi))
   fib=14*aux_fi+(aux_fi/2);

elseif ((fi<=16*aux_fi && fi>15*aux_fi))
   fib=15*aux_fi+(aux_fi/2);

end


if (v>=-0.1 && v<=0.1)
   vb=0;
elseif (v>0.1 && v<=0.4)
   vb=0.25;
elseif (v<-0.1 && v>=-0.4)
   vb=-0.25;
end

% Integraçao
[T,X] = ode45('sis_nao_linear',[tempo ts], Xs);

% Armazenamento de 108inearizа
n = length(T);
tempo = ts;
Xs = X(n,:);
fi=X(n,3);

vetvref(passo, :) = v;
vetwref(passo, :) = w;
vetout(passo,:) = [X(n,1) X(n,2) 180*X(n,3)/pi X(n,4) X(n,5)];
vettime(passo) = T(n,:);
```

```
        não=não+abs([X(n,1);X(n,2)]);
        slin=slin+abs([X(n,4);X(n,5)]);
    end

    %calculo do erro
    erro(1,1)=abs(não(1,1)-slin(1,1))*100/não(1,1);
    erro(2,1)=abs(sre(2,1)-slin(2,1))*100/sre(2,1);

figure;
plot(vetout(:,1), vetout(:,2),'b');
title('Trajetória real X trajetória linearizada');
hold on;
plot(vetout(:,4), vetout(:,5),'r');
xlabel('X(m)');ylabel('Y(m)');
legend('Tragetória não linear','Tragetória linearizada');
grid;

figure
subplot(411)
plot(vettime, vetout(:,1),'b');
hold on;
plot(vettime, vetout(:,4),'r');
title('X em função do tempo');
xlabel('s');ylabel('X');
legend('não linear','linearizada');
hold off
grid;
subplot(412)
plot(vettime, vetout(:,2),'b');
hold on;
plot(vettime, vetout(:,5),'r');
title('Y em função do tempo');
xlabel('s');ylabel('Y');
legend('não linear','linearizada');
hold off
grid
subplot(413)
plot(vettime, vetout(:,3),'b');
title('FI em função do tempo');
xlabel('s');ylabel('º');
grid;
subplot(414)
plot(vettime, vetvref,'b');
title('wref');
hold on;
plot(vettime, vetwref,'r');
title('Referencia');
xlabel('s');
legend('v','w');
grid;
hold off;
clc
disp(['Erro para deslocamento em X: ',num2str(erro(1,1)),'%']);
disp(['Erro para deslocamento em Y: ',num2str(erro(2,1)),'%']);
```

FUNÇÕES

Função sis_nao_linear

```
function d = sis_nao_linear(t,x);
  global v w

  lin=linearizacao(x);

  d(1)=v*cos(x(3));
  d(2)=v*sin(x(3));
  d(3)=w;
  d(4)=lin(1);
  d(5)=lin(2);

  d=d';
```

Função linearização

```
function lin = 110inearização(x);
global fib v vb

%%LINEARIZANDO NOS PONTOS vb e fib
% vb=0;
% xp=vcos(fi)
% xplin(fi,v)=xp(fib,vb)+(-vb*sin(fib))*(fi-fib)+cos(fib)*(v-vb)
% xplin(fi)=a1 + b1(fi-fib) + c1(v-vb)
a1=vb*cos(fib);
b1=-vb*sin(fib);
c1=cos(fib);

% xplin(fi)=(a1 – b1*fib – c1*vb) + b1*fi + c1*v
% xplin(v,w)=d1 + b1*fi + c1*v
d1=a1 – b1*fib – c1*vb;

lin(1)=d1 + b1*x(3) + c1*v;

% yp=vsen(fi)
% yplin(fi)=yp(fib,vb)+(vb*cos(fib))*(fi-fib)+sin(fib)(v-vb)
% yplin(fi)=a1 + b1(fi-fib) c1(v-vb)
a1=vb*sin(fib);
b1=vb*cos(fib);
c1=sin(fib);

% yplin(fi)=(a1 – b1*fib – c1*vb) + b1*fi + c1*v
% yplin(v,w)=d1 + b1*fi + c1*v
d1=a1 – b1*fib – c1*vb;
```

```
lin(2)=d1 + b1*x(3) + c1*v;
```

APÊNDICE C

SIMULAÇÕES DO CONTROLADOR DINÂMICO

SIMULAÇÃO DO CONTROLADOR DINÂMICO SINTONIZADO

```
clear all;
global vref wref pert

tcont=cputime;
teta;
A=[-teta1(1,17) -teta1(1,20);-teta1(1,26) -teta1(1,23)];
B=[teta1(1,15) teta1(1,18);teta1(1,24) teta1(1,21)];
C=eye(2);
D=zeros(2);

% Tempo de amostragem
Ts = 0.1;

% Definição das matrizes do espaço de estados
sysStruct = mpt_sys(ss(A, B, C, D), Ts);

% restrições nos estados
sysStruct.xmax = [inf; inf];
sysStruct.xmin = [-inf; -inf];

% restrições no sinal de controle
sysStruct.umax = [0.4;0.5]
sysStruct.umin = [-0.4;-0.5];

% restrições nas saídas
sysStruct.ymax = [0.4;0.5]
sysStruct.ymin = [-0.4;-0.5];

% restrições na variação do sinal de controle
sysStruct.dumax = inf;
sysStruct.dumin = -inf;

% horizonte de previsão
probStruct.N = 8;

% horizonte de controle
probStruct.Nc = 2;

% Norma
probStruct.norm = 2;

% Peso nos estados
probStruct.Q = [1 0;0 1];

% Peso nas saídas
probStruct.Qy = 20*eye(2);

% Pesos no sinal de controle
probStruct.R = eye(2);

% Tipo de controlador
probStruct.tracking = 1;

% Método de solução do controlador
probStruct.subopt_lev=0;
```

```
% Cálculo co controlador explícito
ctrl = mpt_control(sysStruct, probStruct, 'online');
tcontf=cputime-tcont;


tempo = 0;
[n,n_in]=size(ctrl.Bi);
xm=[0;0];
Xf=zeros(n,1);
N_sim=170;
vref=[0.4*ones(1,50) zeros(1,50) -0.4*ones(1,70)];
wref=[zeros(1,20) 0.5*ones(1,50) zeros(1,50) -0.5*ones(1,50)];
r=[vref;wref];
vref=0;
wref=0;
delta(1,:)=[zeros(1,10) -0.02*ones(1,50) 0.02*ones(1,50) zeros(1,60)];
delta(2,:)=[zeros(1,30) -0.025*ones(1,50) 0.025*ones(1,50) zeros(1,40)];
u=[0;0]; % u(k-1) =0
y=[0;0];
u1=[0;0];
uprev=[0;0];
sent=[0;0];
ssai=[0;0];

tin=cputime;
for kk=1:N_sim

passo = kk;
ts = passo*Ts;

pert(1,1)=delta(1,kk);
pert(2,1)=delta(2,kk);
[T,X] = ode45('nao_linear_sis_cadeira_pert',[tempo ts], xm);

n = length(T);
tempo = ts;
xm = X(n,:);

%cálculo da ação de controle
[X1,U]=mpt_computeTrajectory(ctrl,xm,1,struct('reference',r(:,kk)));


U=U';
deltaU = U - uprev;
uprev = U;
u1(:,kk)=U;
y1(:,kk)=y;
Du1(:,kk)=deltaU;

vetvref(passo, :) = vref;
vetwref(passo, :) = wref;
vetout(passo,:) = [X(n,1) X(n,2)];
vettime(passo) = T(n,:);
vref=U(1,:);
wref=U(2,:);

sent=sent+abs(r(:,kk));
```

```
ssai=ssai+abs(X(n,:)');

end
tsim=cputime-tin;

%calculo do erro
erro=abs(sent-ssai)*100/sent

figure
subplot(411)
plot(vettime,vetout(:,1))
xlabel('Sampling Instant')
hold on;
plot(vettime,r(1,:),'r')
plot(vettime,delta(1,:),'k')
legend('Velocidade Linear','Referencia','Pertubação em v')
axis([0 N_sim*Ts -0.6 0.6]);
hold off;
grid;
subplot(412)
plot(vettime,vetout(:,2))
xlabel('Sampling Instant')
hold on;
plot(vettime,r(2,:),'r')
plot(vettime,delta(2,:),'k')
legend('Velocidade angular','Referencia','Pertubação em w')
axis([0 N_sim*Ts -0.6 0.6]);
hold off;
grid;
subplot(413)
plot(vettime,Du1(1,:))
hold on;
plot(vettime,Du1(2,:),'r')
xlabel('Sampling Instant')
legend('DeltaU(v)','DeltaU(w)')
axis([0 N_sim*Ts -0.6 0.6]);
hold off;
grid;
subplot(414)
plot(vettime,u1(1,:))
hold on;
plot(vettime,u1(2,:),'r')
xlabel('Sampling Instant')
legend('Control(v)','Control(w)')
axis([0 N_sim*Ts -0.6 0.6]);
grid;
clc
disp(['Tempo gasto na criação do controlador: ',num2str(tcontf),'s ']);
disp(['Tempo de simulação por iteração do modelo dinâmico MPT: ',num2str(tsim/N_sim),'s']);
disp(['Tempo de simulação total do modelo dinâmico MPT: ',num2str(tsim),'s']);
disp(['Erro para velocidade linear: ',num2str(erro(1,2)),'%']);
disp(['Erro para velocidade angular: ',num2str(erro(2,2)),'%']);
```

FUNÇÕES

Função nao_linear_sis_cadeira_pert

```
function d = nao_linear_sis_cadeira_pert(t,x);
  global vref wref pert

  teta;

  d(1)=teta1(1,15)*vref+teta1(1,15)*pert(1,1)+teta1(1,16)*x(2)^2-
teta1(1,17)*x(1)+teta1(1,18)*wref+teta1(1,18)*pert(2,1)-teta1(1,19)*x(1)*x(2)-teta1(1,20)*x(2);
  d(2)=teta1(1,21)*wref+teta1(1,21)*pert(2,1)-teta1(1,22)*x(1)*x(2)-
teta1(1,23)*x(2)+teta1(1,24)*vref+teta1(1,24)*pert(1,1)+teta1(1,25)*x(2)^2-teta1(1,26)*x(1);

  d = d';
```

APÊNDICE D

SIMULAÇÕES DO CONTROLADOR CINEMÁTICO

SIMULAÇÃO DO CONTROLADOR CINEMÁTICO SINTONIZADO

```
clear all;
global v w

tcont=cputime;
matrizes_PWA_3v_4fi;

ref=[2;1];

% sampling time
Ts = 0.5;

%state constraints
sysStruct.xmax = [inf; inf; inf];
sysStruct.xmin = [-inf; -inf; -inf];

% input constraints
sysStruct.umax = [0.4;0.5]
sysStruct.umin = [-0.4;-0.5];

% output constraints
sysStruct.ymax = [inf;inf];
sysStruct.ymin = [-inf;-inf];

% du constraints
sysStruct.dumax = inf;
sysStruct.dumin = -inf;

% prediction horizon
probStruct.N = 3;

% quadratic performance index
probStruct.norm = inf;

% penalty on states
probStruct.Q = eye(3);

% penalty on outputs
probStruct.Qy = [1.7 0;0 2.21];

% penalty on inputs
probStruct.R = eye(2);

% the controller must track a given reference

probStruct.subopt_lev=0;

probStruct.yref = ref;

% compute on-line controller
ctrl = mpt_control(sysStruct, probStruct, 'online');
tcontf=cputime-tcont;

tempo = 0;
[n,n_in]=size(ctrl.Bi);
xm=[0;0;0*pi/180];
```

```
Xf=zeros(n,1);
N_sim=22;
v=0;
w=0;
u=[0;0];
y=[0;0;0];
u1=[0;0];
uprev=[0;0];
sent=[0;0];
ssai=[0;0];

tin=cputime;
for kk=1:N_sim
kk
passo = kk;
ts = passo*Ts;

[X1,U]=mpt_computeTrajectory(ctrl,xm,1);

opt=odeset('MaxStep',0.1);
[T,X] = ode45('cinematica',[tempo ts], xm,opt);

n = length(T);
tempo = ts;
xm = X(n,:);

U=U';
deltaU = U - uprev;
uprev = U;
u1(:,kk)=U;
y1(:,kk)=y;
Du1(:,kk)=deltaU;

vetv(passo, :) = v;
vetw(passo, :) = w;
vetout(passo,:) = [X(n,1) X(n,2) X(n,3)*180/pi];
vettime(passo) = T(n,:);
vetref(passo,:) = ref;
v=U(1,:);
w=U(2,:);

end
tsim=cputime-tin;

figure
subplot(411)
plot(vettime,vetout(:,1))
xlabel('Sampling Instant')
hold on;
plot(vettime,vetref(:,1),'r')
legend('X','Referencia')
hold off;
grid;
subplot(412)
plot(vettime,vetout(:,2))
xlabel('Sampling Instant')
hold on;
plot(vettime,vetref(:,2),'r')
legend('Y','Referencia')
```

```
hold off;
grid;
subplot(413)
plot(vettime,vetout(:,3))
xlabel('Sampling Instant')
legend('fi')
grid;
subplot(414)
plot(vettime,u1(1,:))
hold on;
plot(vettime,u1(2,:),'r')
xlabel('Sampling Instant')
legend('Control(v)','Control(w)')
grid;

figure
plot(vetout(:,1),vetout(:,2))
xlabel('X')
ylabel('Y')
legend('Trajetória')
grid;

clc
disp(['Tempo gasto na criação do controlador: ',num2str(tcontf),'s ']);
disp(['Tempo de simulação por iteração do modelo dinâmico MPT: ',num2str(tsim/N_sim),'s']);
```

FUNÇÕES

Função matrizes_PWA_3v_4fi

```
setor=1;
for svb=1:3
    if svb==1
        vb=0;
        v1=-0.1;
        v2=0.1;
    elseif svb==2
        vb=0.25;
        v1=0.1;
        v2=0.4;
    else
        vb=-0.25;
        v1=-0.4;
        v2=-0.1;
    end

    for sfib=1:4
        if sfib==1
            fib=0*pi/180;
            fi1=0*pi/180;
            fi2=11.25*pi/180;
        elseif sfib==2
            fib=22.5*pi/180;
            fi1=11.25*pi/180;
            fi2=33.75*pi/180;
        elseif sfib==3
            fib=45*pi/180;
            fi1=33.75*pi/180;
            fi2=56.25*pi/180;
        else
            fib=67.5*pi/180;
            fi1=56.25*pi/180;
            fi2=78.75*pi/180;
        end

        A=[0 0 -vb*sin(fib);0 0 vb*cos(fib);0 0 0];
        B=[cos(fib) 0;sin(fib) 0;0 1];
        C=[1 0 0;0 1 0];
        D=[0 0;0 0];
        % PWA description 1: x(k+1)=A{1} x(k) + B{1} u(k) + f{1}
        [Ad,Bd,Cd,Dd] = c2dm(A,B,C,D,0.5);
        sysStruct.A{setor} = Ad;
        sysStruct.B{setor} = Bd;
        sysStruct.f{setor} = [vb*fib*sin(fib);-vb*fib*cos(fib);0];
        % y(k) = C{1} x(k) + D{1} u(k) + g{1}
        sysStruct.C{setor} = Cd;
        sysStruct.D{setor} = Dd;

        % guardX{1}*x(k)+guardU{1}*u(k)<=guardC{1}
            sysStruct.guardX{setor}=[0 0 -1;0 0 1;0 0 0;0 0 0];
            sysStruct.guardU{setor}=[0 0;0 0;-1 0;1 0];
            sysStruct.guardC{setor}=[-fi1;fi2;-v1;v2];
        setor=setor+1;
    end
end
```