Pós-Graduação em Ciência da Computação

Victor Hugo Sabino dos Santos Araújo

UMA ABORDAGEM PARA TUNING DE UM CONTROLADOR PI PARA MOTORES BRUSHLESS DC: um estudo de caso aplicado ao controle de movimento de um robô omnidirecional

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2020

Victor Hugo Sabino dos Santos Araújo

UMA ABORDAGEM PARA TUNING DE UM CONTROLADOR PI PARA MOTORES BRUSHLESS DC: um estudo de caso aplicado ao controle de movimento de um robô omnidirecional

Este trabalho foi apresentado à Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia da Computação

Orientadora: Edna Natividade da Silva Barros

Recife

2020

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A663a Araújo, Victor Hugo Sabino dos Santos
Uma abordagem para tuning de um controlador PI para motores brushless DC: um estudo de caso aplicado ao controle de movimento de um robô omnidirecional / Victor Hugo Sabino dos Santos Araújo. – 2020.
123 f.: il., fig., tab.

Orientadora: Edna Natividade da Silva Barros.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências e apêndices.

1. Engenharia da computação. 2. Sistemas de controle. I. Barros, Edna Natividade da Silva (orientadora). II. Título.

621.39 CDD (23. ed.) UFPE - CCEN 2020 - 78

Victor Hugo Sabino dos Santos Araújo

“Uma Abordagem para Tuning de um Controlador PI para Motores Brushless DC: um estudo de caso aplicado ao controle de movimento de um robô omnidirecional”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 13 de fevereiro de 2020.

BANCA EXAMINADORA

_____

Prof. Dr. Adriano Augusto de Moraes Sarmento
Centro de Informática / UFPE

_____

Prof. Dr. Victor Wanderley Costa de Medeiros
Departamento de Estatística e Informática/UFRPE

_____

Profa. Dra. Edna Natividade da Silva Barros
Centro de Informática / UFPE
(**Orientadora**)

Dedico este trabalho a todos os apaixonados por robótica.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado forças pra seguir nesta caminhada, e a todos que de alguma forma contribuíram para eu chegar até este momento.

Agradeço à minha mãe Ladijane Sabino por ter batalhado por mim incessantemente, abrindo mão de tudo para me oferecer sempre o melhor possível.

Agradeço ao meu pai Vicente Araújo pelo incentivo e apoio nos momentos mais difíceis dessa jornada.

Agradeço à Mariana Larissa pelo apoio incondicional, companheirismo, e por me encorajar a sempre ser uma pessoa melhor a cada dia.

Agradeço à minha avó Alice Pereira, aos meus tios, que considero irmãos, Josenildo e Almir Sabino, à minha querida madrinha Marlene Gomes.

Agradeço à minha orientadora Edna Barros por ter confiado em mim, me apoiado e me guiado de maneira excepcional durante toda essa jornada.

Agradeço ao meu grupo pesquisa da pós graduação, formado por: João Gabriel Machado, Lucas Cambuim, Paulo Vasconcelos, Pedro Ishimaru, Rafael Macieira, Vanessa Ogg e Victor Gomes. Obrigado pela ajuda, compartilhamento de experiências, e por todas as discussões proveitosas e ideias que surgiram durante as reuniões de acompanhamento.

Agradeço a todos os membros do grupo de robótica RobôCIn, em especial, à equipe de SSL: Cecília Silva, Felipe Martins, Heitor Rapela, Lucas Cavalcanti, Matheus Vinícius, Renato Sousa, Riei Joaquim, Roberto Fernandes e Ryan Vinícius.

Agradeço também à FACEPE pelo fomento de meus estudos possibilitando a dedicação integral a esta pesquisa.

RESUMO

O controle de motores é fundamental no estudo e desenvolvimento de Robôs Móveis Autônomos (RMAs). A técnica de controle mais utilizada é a implementação de controladores Proporcional, Integral e Derivativo (PID), porém, a escolha das constantes de controle ótimas para a aplicação é uma questão crítica e complexa em algumas aplicações e às vezes negligenciada por alguns projetistas de RMAs. Um levantamento feito entre as equipes participantes da *RoboCup*, maior campeonato de robótica do mundo, na categoria *Small Size League* (SSL) no ano de 2019 mostrou que $79,2\%$ das equipes utiliza um controlador PID (ou uma variação sem o efeito integral ou sem o efeito derivativo) para controlar os motores de seus robôs. Dentre essas equipes, $52,63\%$ não apresenta uma justificativa para a escolha das constantes de controle, já $21,05\%$ afirma ter encontrado os parâmetros sub-ótimos através de tentativas e erros. Os RMAs devem ser capazes de atuar em ambientes dinâmicos e desconhecidos e de reagir em situações de imprevisibilidade, e a precisão e acurácia dessas reações estão atreladas a um controle bem otimizado. Este trabalho propõe uma abordagem para identificar parâmetros de controle adequados a partir de um conjunto de amostras de um motor *Brushless* DC para a implementação de um controlador PI para controle de velocidade desse motor. Será apresentada uma metodologia para identificação de um modelo matemático que melhor descreve a resposta do motor, em seguida, são identificadas as constantes mais adequadas do controlador PI para o modelo proposto com o auxílio da ferramenta PIDTuner do software MATLAB. A técnica foi aplicada a um estudo de caso real e o controlador foi implementado e validado na movimentação de um robô omnidirecional projetado dentro das regras da categoria RoboCup Small Size League. Os resultados mostram uma melhora significativa na resposta do motor para uma velocidade desejada em relação ao controle previamente utilizado, projetado de maneira empírica, e também uma melhora no desempenho do robô ao seguir uma trajetória determinada sem *feedback* da posição atual do robô.

Palavras-chaves: Sistema de Controle. Motores Brushless DC. Robótica.

ABSTRACT

Motor control plays a vital role in the research and development of Autonomous Mobile Robots (AMRs). The most known motor control technique is the Proportional, Integral, and Derivative (PID) control. However, the choice of the more suitable control constants for this technique can be critical and complex for some applications and has been neglected by some control designers. A survey carried out among the teams participating in RoboCup, the largest robotics competition in the world, in the Small Size League (SSL) category showed that 79.2% of the teams uses a PID controller (or its variation without the integral or derivative term) to control their robot motors. From these teams, 52.3% does not present a technique to choose their controller constants, and 21.05% said that their parameters were chosen in an empirical way using trial and error. AMRs should be able to operate in dynamic and unknown environments and to react through unpredictable situations. The precision and accuracy of these reactions are linked to an optimized control. This work proposes an approach to identify optimal control parameters from a set of samples extracted from a *brushless* DC motor for the implementation of a PI controller for speed control of that motor. A methodology to identify a mathematical model that best describes the motor response will be presented. The most suitable constant values of the PI controller for the proposed model are identified with the aid of the MATLAB software using the PIDTuner toolbox. The controller is implemented validated in a case study applied to an omnidirectional robot designed within the rules of the RoboCup Small Size League category. The results show a significant improvement in the motor response to the desired speed in comparison to the previous controller empirically designed, and also an improvement in the robot's performance when following a pre-determined trajectory without the feedback of the robot's current position.

Keywords: Control Systems. Brushless DC Motor. Robotics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	Conversor Analógico-Digital
ARX	Autorregressivo com Entrada Externa
ASICs	Circuitos Integrados de Aplicação Específica
CI	Circuito Integrado
DC	Corrente Contínua
DCA	Conversor Digital-Analógico
ERRT	Extended Rapid-Exploring Random Trees
FIFA	Federação Internacional de Futebol
FPGA	*Field-Programming Gate Array*
GA	Algoritmo Genético
GFSK	Modulação por Chaveamento de Frequência Gaussiana
HDL	Linguagem de Descrição de Hardware
HIL	*Hardware-in-the-Loop*
IAE	*Integral Absolute Error*
IoT	Internet das Coisas
ISE	*Integral Squared Error*
LARC	Competição Latino-Americana de Robótica
LQR	*Linear Quadratic Regulator*
LQT	*Linear Quadratic Tracking*
MATLAB	*Matrix Laboratory*
MOSFETs	Transistores de Efeito de Campo de Óxido de Metal Semicondutor
MPRS	*Multi-level Pseudo Random Sequence*
ms	milissegundos
MSE	Erro Médio Quadrático
NMSE	Erro Médio Quadrático Normalizado
NRMSE	Raiz Quadrada do Erro Médio Quadrático Normalizado
PD	Proporcional e Derivativo
PI	Proporcional e Integral
PID	Proporcional, Integral e Derivativo
PLA	Ácido Polilático

PPR	Pulsos Por Revolução
PRBS	*Pseudo-Random Binary Sequence*
PSO	*Particle Swarm Optimization*
PWM	*Pulse-Width Modulation*
RMAs	Robôs Móveis Autônomos
RMSE	Raiz do Erro Médio Quadrático
RPM	Rotações Por Minuto
SSL	*Small Size League*
TDPs	*Team Description Papers*
UFPE	Universidade Federal de Pernambuco
ZOH	*Zero-Order-Hold*

LISTA DE SÍMBOLOS

α	Letra grega minúscula Aplha
γ	Letra grega Gamma
β	Letra grega Beta
θ	Letra grega Theta
$\forall$	Para todo
$\geq$	Maior ou Igual
$\leq$	Menor ou Igual
π	Letra Grega Pi
ϕ	Letra grega Phi
θ	Letra grega Theta
ω	Letra grega Ômega

SUMÁRIO

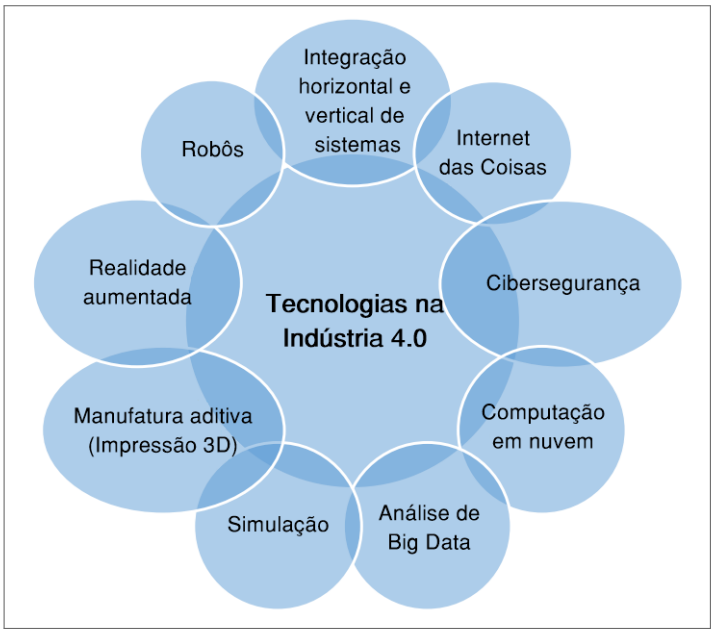
1 INTRODUÇÃO

Este capítulo introdutório apresenta as principais motivações deste trabalho, trazendo o contexto em que o tema está inserido e onde será feita a validação da abordagem proposta, apresentando alguns conceitos-chave que servirão como base para os próximos capítulos. Em seguida, será caracterizado o problema e serão apresentados os objetivos geral e específicos, além de mostrar como o restante do trabalho está subdividido.

1.1 MOTIVAÇÃO

Até pouco tempo atrás, o uso de robôs era restrito a tarefas rigorosamente controladas em indústrias específicas, por exemplo, a indústria automotiva (SCHWAB, 2017, p.20-21). Porém, esta visão mudou com a ascensão da Indústria 4.0. De acordo com Bahrin et al. (2016), a Indústria 4.0 é uma área onde a Internet das Coisas (IoT) junto com os sistemas ciber-físicos estão interconectados de forma que a combinação de software, sensores, processadores, e tecnologias de comunicação irá promover melhorias nos processos do setor industrial. Entretanto, não se deve restringir a aplicação dos robôs somente ao setor industrial, pois, a utilização de robôs aumenta rapidamente em áreas totalmente distintas, como na medicina, no agronegócio, e no ambiente domiciliar. Esse progresso irá fazer com que, em breve, haja uma colaboração diária entre robôs e humanos nas mais diversas tarefas. Algumas das tecnologias relacionadas à Indústria 4.0 podem ser mostradas na Figura 1.

Figura 1 – Tecnologias relacionadas à Indústria 4.0



Fonte: (BAHRIN et al., 2016) (ADAPTADO).

O estudo e desenvolvimento de Robôs Móveis Autônomos (RMAs) têm um papel cada vez mais importante na robótica. Um relatório anual intitulado *Tech Trends Report* elaborado pelo *The Future Institute* apontou o desenvolvimento de equipes de robôs autônomos como uma das tendências do ano de 2019 (INSTITUTE, 2019). Segundo o relatório, robôs autônomos construídos para trabalhar em conjunto como um time podem ser utilizados em diversas aplicações, como agricultura, infraestrutura, e prevenção de acidentes. Os robôs, ao contrário dos agentes controlados por softwares de simulação, operam no mundo real, e as restrições de tempo real – onde sensores e atuadores possuem características físicas específicas – precisam ser controladas(KRAMER; SCHEUTZ, 2007, p.101). De acordo com trabalhos desenvolvidos por Roseli Romero et al. (2014), um dos maiores desafios no desenvolvimento de robôs móveis é a capacidade de interação com o ambiente e tomada de decisões corretas, pois, os RMAs devem ser capazes de atuar em ambientes dinâmicos e desconhecidos, e de reagir mediante a situações de imprevisibilidade.

Um projeto de robô bem desenvolvido deve estar baseado na integração eficaz entre as seguintes áreas: sistemas embarcados, mecânica, eletrônica e programação. Falando um pouco destas áreas individualmente, podemos citar trabalhos de Vahid Vahid e Givargis (1999), que definem um sistema embarcado como qualquer sistema computacional que não seja um *desktop*, *laptop*, ou computador *mainframe*. Esse sistema deve ser reativo, funcionar em tempo real e possuir restrições rígidas de projeto, utilizando métricas como desempenho, custo, tamanho e potência, além de ser capaz de executar um mesmo programa diversas vezes. A área que engloba a mecânica é responsável pela locomoção do robô e/ou manipulação de objetos. Em geral, o projeto mecânico de um robô móvel envolve a modelagem das partes físicas do robô, modelagem cinemática, modelagem dinâmica, e dimensionamento de motores, sensores e atuadores de acordo com os requisitos de projeto. Técnicas de desenvolvimento de sistemas eletrônicos garantem o correto funcionamento do sistema como um todo, tanto dos componentes do sistema embarcado, quanto de motores e sensores previamente escolhidos, garantindo tensão e corrente necessários para o bom funcionamento dos componentes. Já a programação é responsável por implementar um algoritmo robusto, capaz de lidar com as incertezas presentes nas leituras dos sensores e nas ações dos motores e/ou manipuladores num ambiente real, que é bastante dinâmico e imprevisível. A aplicação do conhecimento dessas áreas em conjunto pode resultar num agente autônomo capaz de realizar tarefas predeterminadas com capacidade de tomar decisões sem a interferência humana.

Para estimular pesquisa e desenvolvimento no campo da robótica e inteligência artificial, foi proposta a iniciativa de uma copa do mundo de robôs em 1995, a *RoboCup*. O objetivo inicial da *RoboCup* era de organizar um campeonato mundial com robôs reais, fornecendo uma tarefa padrão para pesquisas em movimento rápido num ambiente de múltiplos robôs que colaboram em conjunto para resolver problemas dinâmicos (KITANO et al., 1995). Em 1998, uma meta corajosa foi proposta por Kitano e Asada (1998):

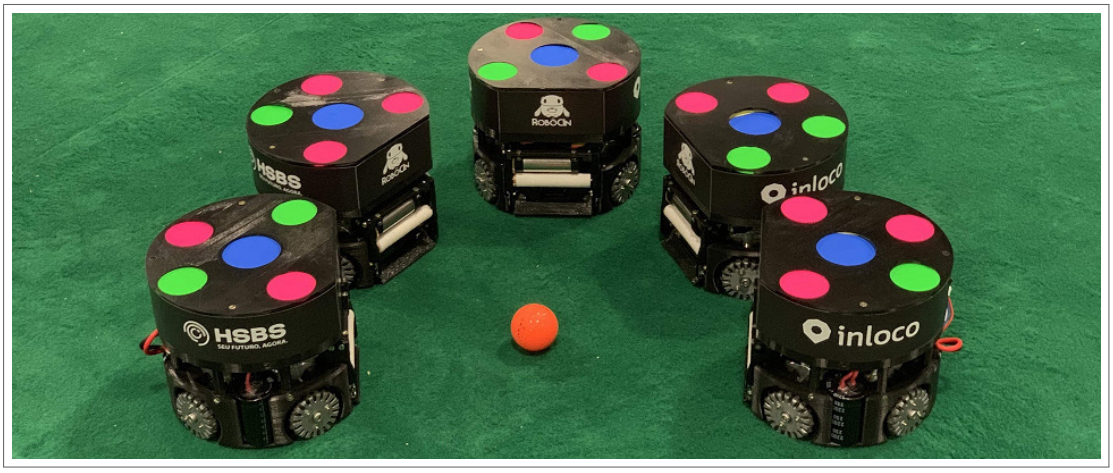
> Até a metade do século XXI, um time de futebol completo composto

> por robôs humanoides autônomos deve ganhar uma partida de futebol, de acordo com as regras oficiais da Federação Internacional de Futebol (FIFA), contra o vencedor da Copa do Mundo mais recente (p. 723, tradução nossa).

Um roteiro para alcançar essa meta foi proposto por Burkhard et al. (2002), onde em 2030 os robôs deveriam jogar num campo de tamanho oficial estabelecido pela FIFA, com 11 robôs para cada time, com iluminação em ambiente aberto e sem ajuda humana, com um sistema de comunicação sem fio.

Uma das principais categorias de futebol de robôs da *RoboCup* é a *Small Size League*, também chamada de F180, que consiste em uma partida entre dois times de robôs (um azul, e um amarelo) e a quantidade de robôs de cada time depende da divisão em que a equipe está participando. Atualmente a categoria possui duas divisões: a divisão A, onde cada equipe pode ter no máximo 8 robôs e o campo possui dimensões $(12 \text{ x } 9)m$ e a divisão B, onde cada equipe pode ter no máximo 6 robôs e o campo possui dimensões $(9 \text{ x } 6)m$ (COMMITEE, 2019). Existe, também, uma restrição para o tamanho de cada robô, onde cada um pode ter no máximo $180\ mm$ de diâmetro e $150\ mm$ de altura, e a bola utilizada no jogo é uma bola de golfe padrão, de aproximadamente $43\ mm$ de diâmetro. A Figura 2 mostra cinco robôs utilizados na competição juntamente com uma bola de golfe padrão.

Figura 2 – Cinco robôs omnidirecionais da categoria *Small Size League*.

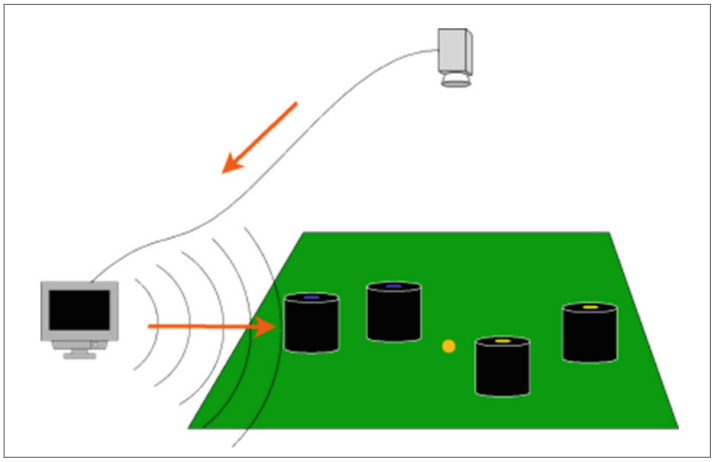


Fonte: (SILVA et al., 2020).

Atualmente, todos os objetos no campo são rastreados por um sistema de visão global que se encontra a, aproximadamente, 4 metros de altura e é composto por quatro câmeras dispostas de forma a cobrir todo o campo. As imagens dessas câmeras são processadas por um sistema de visão compartilhado, um projeto *open-source* chamado SSL-Vision (ZICKLER et al., 2009). Cada equipe é responsável por tratar as informações recebidas pelo sistema SSL-Vision, bem como as informações recebidas pelo árbitro que controla o jogo. As equipes devem implementar um sistema de comunicação sem fio para enviar os comandos corretos para cada robô de acordo com a situação do jogo. A Figura 3 ilustra o funcionamento do controle de jogo de uma equipe, onde o programa de estratégia executando em um computador da equipe

recebe os dados do campo em tempo real do sistema SSL-Vision através de um computador fora do campo, processa e envia através de uma comunicação sem fio informações para o seu time.

Figura 3 – Controle de jogo da categoria Small Size League



Fonte: (WEITZENFELD et al., 2014).

Um robô móvel precisa de mecanismos de locomoção que o permitam movimentar e interagir com outros robôs no ambiente (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Decidir qual instrumento deve ser utilizado para a execução de movimentos é fundamental no projeto de robôs móveis autônomos devido ao grande número de possibilidades existentes. A roda é o mais popular meio de locomoção em robôs móveis e veículos em geral, pois, não é necessário se preocupar com o balanço, e sim com parâmetros como tração, estabilidade, manobrabilidade e controle. Qualquer veículo com rodas está sujeito a restrições cinemáticas que limitam sua mobilidade localmente, porém, é possível chegar a configurações arbitrárias executando manobras da maneira correta (SICILIANO et al., 2010).

Para controlar um robô de maneira eficiente, é preciso conhecer a cinemática do robô como um todo, e o comportamento dinâmico de seus motores e atuadores. A cinemática estuda, classifica e compara o movimento dos mecanismos em termos de espaço e tempo, sem levar em consideração as forças que agem neste mecanismo (WILSON; SADLER; MICHELS, 2003; HALLIDAY; RESNICK; WALKER, 2012; JEWETT; SERWAY, 2008, p.2). Já a dinâmica estuda o movimento de corpos individuais e mecanismos sob influência de forças e torques (WILSON; SADLER; MICHELS, 2003, p.2). A compreensão através da dinâmica irá fornecer informações úteis para a análise do movimento dos objetos em estudo (MERIAM; KRAIGE, 2012, p.3).

Os motores mais utilizados em robôs de pequeno porte são os motores de Corrente Contínua (DC), mais especificamente, os motores não-escovados de corrente contínua, ou *brushless* DC. Estudos realizados por Xia (2012) apontam como principais vantagens dos motores *brushless* DC sua estrutura simples, a boa eficiência e grande torque; enquanto que os motores tradicionais com escovas (*brushes*) apresentam problemas de fricção mecânica que resultam em baixo tempo de vida, podendo apresentar ruído, interferência de radio, e

até mesmo faíscas. Algumas das aplicações dos motores *brushless* DC são a indústria da manufatura (ex.: linha de produção automatizada), veículos elétricos, medicina (ex.: robôs cirurgiões e odontologia), indústria aeronáutica (ex.: drones) e também no agronegócio.

Uma das formas mais utilizadas para controlar um motor *brushless* DC é através de Circuitos Integrados de Aplicação Específica (ASICs), esse circuito também é chamado de *driver* do motor. O *driver* do motor faz a abstração do processo de controle, tornando o processo uma caixa-preta, onde é necessário apenas enviar uma tensão de entrada, e ler a resposta do motor a esse estímulo. Porém, a adição de um *driver* torna o processo de identificação do modelo do motor mais complexo, pois, além de obter uma equação que represente o motor, é necessário modelar o circuito integrado que o controla.

Para controlar um motor *brushless* DC é necessário descrever uma relação entre o sinal de entrada e saída do motor, esse processo também é conhecido como modelagem. Alguns autores demonstraram maneiras de obter um modelo matemático que simula o comportamento de motores *brushless* DC no mundo real sem a presença de um *driver*. Esse modelo pode ser obtido através de equações derivadas do comportamento dinâmico do motor como os pesquisadores Oguntoyinbo et al. (2009), Singh, Katal e Modani (2014), Patel e Pandey (2013) fizeram em seus trabalhos, ou experimentalmente, como Baede (2006) demonstrou. Para ambos os casos, análises podem ser feitas através de um *software* de simulação como, por exemplo, o *Matrix Laboratory* (MATLAB). Já o processo de obtenção de um modelo do motor com a adição de um *driver* torna o processo de modelagem matemática bem mais complicado, por isso, os modelos para esse tipo de aplicação são obtidos de maneira experimental.

O controle automático dos motores, também chamado de controle de baixo nível, é essencial para sistemas robóticos. Esse controle pode ser feito através da medição do estado atual da variável que está sendo controlada, em seguida é aplicado um valor a essa variável que irá corrigir o erro entre o valor atual e o desejado para essa variável (OGATA, 2003). A técnica mais comum para controlar os motores *brushless* DC é através de um controlador Proporcional, Integral e Derivativo (PID). De acordo com Nise (2007), esse tipo de controlador é capaz de eliminar o erro do sistema no regine estacionário, enquanto otimiza a resposta no regime transiente. Outras abordagens também podem ser utilizadas, como *Linear Quadratic Tracking* (LQT) (KATAYAMA et al., 1985), que foi utilizado por Hashemi, Jadidi e Jadidi (2011) para o controle de um robô omnidirecional, e *Linear Quadratic Regulator* (LQR) (ANDERSON; MOORE, 2007), que também foi utilizado para controlar um robô omnidirecional por Mamun, Nasir e Khayyat (2018).

Uma vez projetado o controlador, é preciso obter um modelo cinemático do robô para que seja possível prever qual será a movimentação do robô dada a contribuição de cada um dos seus motores. Trabalhos desenvolvidos por Siegwart afirmam que derivar um modelo do robô é um processo que ocorre de baixo para cima, pois, cada roda contribui para a movimentação do robô e, ao mesmo tempo, impõe restrições ao seu movimento (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Portanto, a modelagem cinemática depende diretamente da geometria

do robô e da disposição dos motores.

O planejamento de percurso, também chamado de controle de alto nível, é responsável por decidir qual a melhor rota para o robô percorrer. Dada a posição inicial e o destino, o algoritmo de planejamento determina de forma automática a trajetória que o robô deve seguir, evitando a colisão com obstáculos presentes no ambiente em que o robô está inserido (ROMERO et al., 2014). Algumas das técnicas de planejamento de rotas mais conhecidas e aplicadas em RMAs são: campos potenciais (KHATIB, 1986), grafos de visibilidade (LOZANO-PÉREZ; WESLEY, 1979), curvas de Bézier (JOLLY; KUMAR; VIJAYAKUMAR, 2009) e Extended Rapid-Exploring Random Trees (ERRT) (BRUCE; VELOSO, 2002). Este trabalho, entretanto, irá focar no controle de baixo nível, que é responsável por garantir que os motores se comportem de acordo com a velocidade determinada pelo controle de alto nível.

1.2 PROBLEMA

A categoria *RoboCup Small Size League* na RoboCup vem crescendo a cada ano, chegando em 2019 a 10 equipes inscritas na Divisão A e 15 equipes inscritas na Divisão B. A equipe RobôCIn do Centro de Informática da Universidade Federal de Pernambuco (UFPE) participa da categoria desde a Competição Latino-Americana de Robótica (LARC) 2018 realizada em João Pessoa, na Paraíba. Um dos principais problemas do RobôCIn ao iniciar na competição foi controlar os motores da maneira correta e eficiente, extraindo o máximo do potencial dos motores *brushless* DC utilizados.

Uma análise feita entre os Artigos de Descrição das Equipes, ou *Team Description Papers* (TDPs) que se qualificaram para competir na RoboCup 2019 em Sydney, Austrália, mostrou que $79,2\%$ das equipes utilizam um controlador Proporcional e Integral (PI), Proporcional e Derivativo (PD), ou Proporcional, Integral e Derivativo (PID) para o controle de baixo nível de seus robôs. Dentre essas equipes, $52,63\%$ não apresentaram uma heurística para obter as constantes de controle, já $21,05\%$ afirmam terem encontrado parâmetros de controle de maneira empírica, ou seja, através de tentativas e erros.

De acordo com Abbenseth (2015), é importante que os robôs dessa categoria se movam rapidamente e de maneira precisa, pois as oportunidades de marcar um gol são rapidamente fechadas pelos defensores da equipe adversária. Para alcançar esses dois objetivos (rapidez e precisão), é necessário implementar um controle de movimento eficiente para cada robô. Esse é um dos principais desafios das equipes, pois, além de compreender o princípio de funcionamento de motores *brushless* DC, o projetista deve obter um modelo que descreva da melhor maneira possível o comportamento do motor para aplicar técnicas de controle discreto de acordo com as limitações de hardware impostas. Uma vez obtido o modelo do motor, é possível simular seu comportamento utilizando técnicas de software para obter valores para as constantes de controle mais adequados e validá-los virtualmente antes da validação no ambiente real, evitando danos físicos causados por valores de constantes mal dimensionados ao motor que está sendo controlado. Além disso, para validar o controle de movimento do

robô, é necessário obter um modelo cinemático de acordo com a geometria do robô para que os motores atuem de maneira a possibilitar que o robô siga a direção desejada.

1.3 OBJETIVOS DO TRABALHO

1.3.1 Objetivo geral

O objetivo geral deste trabalho foi propor e avaliar um método para sistematizar a obtenção de um modelo e constantes de um controlador PID para motores *brushless* DC baseada em telemetria com validação num robô real.

1.3.2 Objetivos específicos

- Propor um método para obter as características do motor (tempo de subida, tempo de assentamento e curva de resposta do motor);
- Propor um método para obter um modelo discreto do motor *brushless* DC utilizando telemetria;
- Realizar *tuning* das constantes de controle a partir do modelo discreto atendendo aos requisitos de projeto;
- Implementar um controlador PID discreto para validar as constantes de controle obtidas utilizando um robô real como estudo de caso.

1.3.3 Contribuições

Este trabalho apresenta contribuições para a área de controle de motores *brushless* DC no contexto da robótica móvel, apresentando uma técnica para o *tuning* de um controlador PI que pode ser aplicado a motores *brushless* DC com *encoder*, validado num estudo de caso aplicado ao controle de velocidade e de movimento de robôs omnidirecionais num ambiente parcialmente controlado. Apesar do estudo de caso ser na área de robótica móvel, as técnicas propostas podem ser aplicadas na robótica industrial, na área médica e na área social, pois em todas estas aplicações motores são usados para interação com o ambiente.

1.4 ESTRUTURA DA DISSERTAÇÃO

No Capítulo 2 será apresentada a fundamentação teórica, que descreve conceitos básicos de motores *brushless* DC, utilização de *encoders* para obtenção da velocidade dos motores, e uma introdução a sistemas de controle e controladores PID. No Capítulo 3, serão apresentados os trabalhos relacionados envolvendo abordagens de controle de motores *brushless* DC. A metodologia da abordagem proposta por esse trabalho será descrita no Capítulo 4. Um estudo de

caso com o intuito de validar a abordagem proposta é apresentado no Capítulo 5. Os resultados do trabalho serão mostrados e discutidos no Capítulo 6, onde será feita uma comparação entre a abordagem proposta e uma abordagem empírica que era utilizada anteriormente. O Capítulo 7 apresenta as conclusões do trabalho, e discute futuras melhorias que podem ser feitas à abordagem proposta.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos básicos relacionados a abordagem proposta. Inicialmente, será apresentado o princípio de funcionamento de motores *brushless* DC e como eles são controlados. Em seguida, será apresentada uma forma de medição da velocidade dos motores utilizando um codificador (*encoder*). Também serão descritos conceitos sobre identificação e modelagem de sistemas de controle. Por fim, uma breve introdução a modelagem cinemática será apresentada, que servirá como base para o desenvolvimento do estudo de caso realizado no capítulo 5.

2.1 MOTORES BRUSHLESS DC

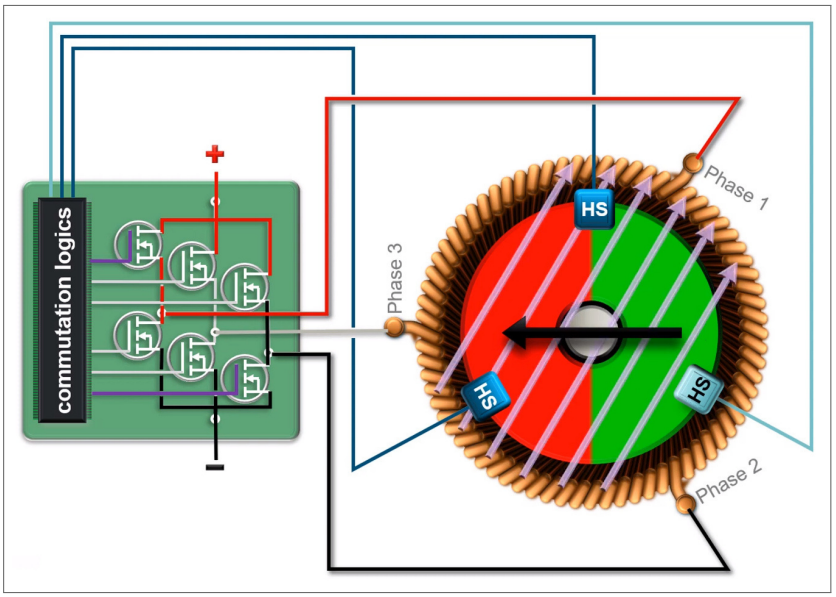
Como mencionado na seção 1.1, a maior parte dos motores utilizados na robótica são de corrente contínua, ou DC. Dentre os tipos de motores DC, os não-escovados (*brushless*) têm se destacado em relação aos motores com escovas devido a melhor eficiência e potência apresentadas, além de ter um ciclo de vida mais longo, ruído e faíscas reduzidos, alcançar velocidades maiores de rotação, podendo ser selados para minimizar a presença de sujeira (PATEL; PANDEY, 2013). Os próximos tópicos irão discorrer sobre o princípio de funcionamento dos motores *brushless* DC, e sobre as maneiras de controlar esse tipo de motor.

2.1.1 Princípio de funcionamento de motores brushless DC

De acordo com Yedala , os motores *brushless* são um tipo de motor síncrono, isto é, o campo magnético gerado pelo estator (parte estática do motor) e o campo magnético gerado pelo rotor (parte girante do motor) operam na mesma frequência (YEDALE, 2003). O tipo de motor *brushless* mais utilizado é o de três fases, isso quer dizer que o estator possui um enrolamento, usualmente em estrela, formando um arranjo de bobinas, enquanto que o rotor possui um ímã permanente. Quando uma tensão contínua é aplicada à uma das fases, ela fica energizada, tornando-se um ímã elétrico. A operação do motor baseia-se na interação entre o ímã permanente do rotor, e os ímãs elétricos do estator.

A Figura 4 mostra um exemplo do funcionamento de um motor de três fases. O rotor é localizado no centro do motor, e possui um ímã permanente de dois polos simetricamente divididos, polo positivo (para onde a seta aponta), e polo negativo (oposto ao positivo), onde o fluxo magnético resultante é representado pela seta preta. O enrolamento de três fases está posicionado no estator, e para aferição das fases são posicionados sensores de efeito Hall separados 120^o uns dos outros para detectar a posição do rotor através dos polos do ímã permanente. Tipicamente, o sinal de saída de um sensor de Efeito Hall é $5V$ no polo positivo, e $0V$ no polo negativo.

Figura 4 – Exemplo de funcionamento de um motor *brushless* DC de três fases.



Fonte: (MAXON, 2019).

Para controlar o movimento do rotor é necessário um dispositivo eletrônico que irá executar o processo de energização do enrolamento. Esse processo de troca de energia entre o rotor e estator, produzindo movimento, é chamado de comutação. Para motores *brushless*, é necessário um sistema de comutação eletrônico, que consiste num dispositivo que possui uma alimentação DC de acordo com a tensão de operação do motor e recebe como entrada a leitura dos sensores de Efeito Hall, controlando o acionamento de seis Transistores de Efeito de Campo de Óxido de Metal Semicondutor (MOSFETs). A lógica de comutação é projetada a partir dos valores de leitura dos sensores de efeito Hall, permitindo a alimentação das fases com sinal positivo, ou negativo, de acordo com o sentido desejado de rotação do rotor.

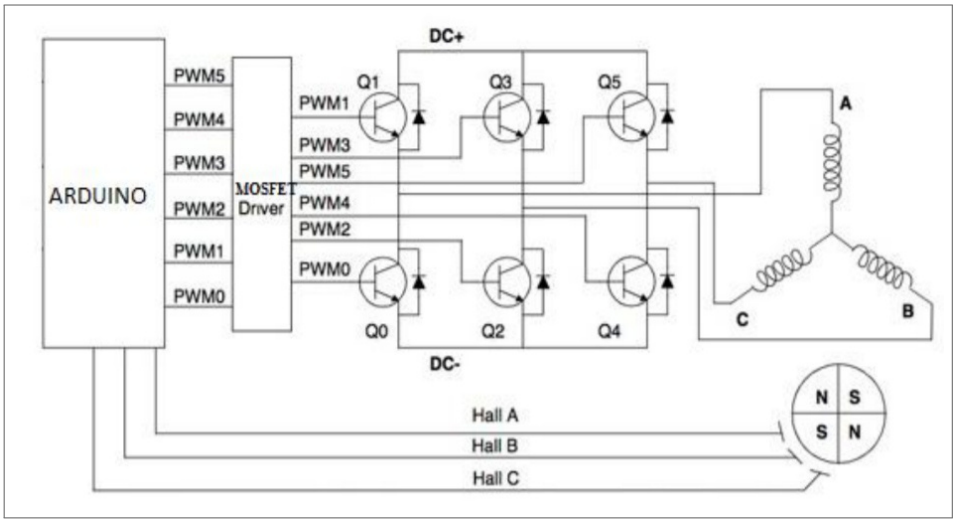
Na Figura 4, dois sensores de Efeito Hall estão ativos por estarem no polo positivo (um deles está na transição do negativo para positivo), portanto, a lógica de comutação programada para essa situação com o sentido de rotação horário indica a ativação dos MOSFETs de modo que a corrente siga da fase 1 para a fase 2, gerando um campo magnético apontando para a direção nordeste. Com isso, o ímã permanente irá buscar se alinhar com o campo magnético produzido momentaneamente, e após uma rotação de 60^o, a leitura dos sensores de Efeito Hall será diferente, e a lógica de comutação programada irá acionar os MOSFETs referentes ao estado dos sensores.

2.1.2 Controle de motores brushless

O sistema de comutação eletrônico pode ser feito através de um microcontrolador, um circuito reconfigurável, também chamado de *Field-Programming Gate Array* (FPGA), ou um Circuito Integrado de Aplicação Específica (ASIC). Bhadani apresentou uma abordagem para controlar motores *brushless* DC de três fases utilizando um microcontrolador Arduino

ATMEGA328, como mostra a Figura 5 (BHADANI et al., 2016). A saída do processador Arduino, que também é espelhada pelos MOSFETs, é na forma de *Pulse-Width Modulation* (PWM), que determina a tensão média e a corrente média para as fases. O motor conta com três sensores de Efeito Hall para indicar a posição do rotor e enviar esse sinal para o processador Arduino, que irá gerar uma sequência de PWM de acordo com a leitura dos sensores para realizar a movimentação desejada. Ao todo, seis sequências de PWM são programadas como entrada para os MOSFETs de acordo com a lógica de funcionamento do motor.

Figura 5 – Exemplo de controle de um motor *brushless* DC de três fases utilizando um microcontrolador Arduino.



Fonte: (BHADANI et al., 2016).

Por outro lado, os circuitos FPGAs são dispositivos que permitem o desenvolvimento de circuitos lógicos reprogramáveis. Estes dispositivos apresentam características de ASICs, como excelente desempenho e baixo consumo de energia, entretanto, sua implementação de projetos em FPGA é feita de maneira mais rápida que os ASICs. Em geral, esses dispositivos possuem um conjunto de ferramentas de síntese, simulação, implementação e validação relativamente simples. Portanto, o tempo para o mercado (*time-to-market*) em relação aos ASICs é menor. Projetos em FPGAs são desenvolvidos utilizando uma Linguagem de Descrição de Hardware (HDL), possuem um custo de projeto médio, e um excelente desempenho.

O trabalho de Tashakori, Hassanudeen e Ektesabi (2015) mostra uma implementação de um controlador de motores *brushless* em um FPGA modelo Spartan 3E. Embora a arquitetura seja semelhante à da Figura 5, o autor justifica que em comparação com os microcontroladores, os FPGAs são capazes de processar operações computacionais repetitivas simples e complexas de maneira mais rápida devido à exploração de paralelismo. A comutação de motores *brushless* de alta rotação demanda um frequência de *clock* do dispositivo bastante elevada. No caso de microcontroladores, que tem que dividir a tarefa de comutação com outras operações do sistema e que não permitem explorar paralelismo, isso pode prejudicar o desempenho do sistema

como um todo devido a uma grande quantidade de interrupções. Por isso, é recomendado a utilização de FPGAs ou ASICs para essa tarefa.

Os ASICs são amplamente utilizados na indústria por serem de menor tamanho, apresentarem excelente desempenho e serem de fácil manuseio. De acordo com Krklješ et al. (2010), algumas características importantes presentes em circuitos integrados de controladores de motor *brushless* DC são a inserção de *dead times* para prevenir um curto-circuito, controle de corrente nas fases, e temporização correta na hora de energizar o enrolamento. Embora a inserção de algumas dessas características acarrete em maior custo, maior de tempo de desenvolvimento e verificação, elas garantem a funcionalidade, flexibilidade e robustez do circuito.

O trabalho desenvolvido por Gülerman (2014) justifica a utilização de um Circuito Integrado (CI) controlador de motores *brushless* DC de três fases modelo A3930 fabricado pela Allegro Microsystems por ser um modelo que possui uma tabela de comutação já incorporada, com várias detecções de falha embutidas, além de reduzir o tempo de desenvolvimento do projeto. Algumas das falhas que podem ser detectadas por esse CI são: curto-circuito (com ground, com a alimentação, e com o enrolamento), sobretemperatura, falha lógica e baixa corrente. Mais informações sobre o controlador A3930, junto com a tabela de comutação, um exemplo de como ele deve ser utilizado, e suas conexões com os MOSFETs e sensores de Efeito Hall são encontrados no *datasheet* fornecido pela empresa (MICROSYSTEMS, 2019).

2.2 ENCODERS

Para medir a velocidade de um motor são utilizados *encoders*. Um *encoder*, é um dispositivo que entrega um sinal digital para cada pequena porção de movimento (NIKU, 2001). Alguns *encoders* utilizam um disco dividido em pequenas seções (arcos), e a movimentação do disco é lida através de sensores óticos, sensores de efeito hall, ou sensores indutivos, que traduzem o movimento do disco em pulsos para indicar direção e movimento, ou posição de um motor.

2.2.1 Princípio de funcionamento de um encoder

Os *encoders* podem ser classificados em dois tipos, o tipo incremental e o tipo absoluto. O *encoder* incremental é capaz de indicar a variação na posição angular em relação ao ponto em que foi ativado, mas não é capaz de indicar a atual posição do sensor. Um *encoder* incremental possui dois canais de saída (A e B) defasados em 90^o, que permitem identificar o sentido da rotação e emitem pulsos quando são acionados e sua resolução está relacionada ao número de arcos que ele possui. A quantidade de arcos indica a quantidade de Pulsos Por Revolução (PPR) que um *encoder* incremental pode ler. Em geral, os *encoders* incrementais possuem uma resolução de 256, 512, 1024, 2048 ou 4096 PPR. O *encoder* absoluto é mais complexo, e é capaz de determinar a posição exata do disco a qualquer momento através

de um referencial estático, sem a necessidade de saber a posição inicial do sensor. Devido à complexidade de um *encoder* absoluto, esse projeto irá utilizar um *encoder* incremental e o modo de funcionamento desse tipo de *encoder* será detalhado a seguir.

O processo de contagem de pulsos para medir a velocidade e a direção do dispositivo a ser medido é chamado de codificação. Há três tipos básicos de codificação: X1, X2 e X4. A codificação X1 utiliza apenas a subida do sinal do canal A como gatilho para incrementar ou decrementar o contador. Quando o sinal do canal A está à frente do canal B, o incremento ocorre na subida do canal A, já quando o sinal do canal B está à frente do canal A, o decremento ocorre na descida do canal A. A codificação X2 possui o mesmo comportamento da codificação X1, mas o incremento ou decremento ocorrem tanto na subida quanto na descida do canal A. Dessa forma, cada pulso resulta em dois incrementos ou decrementos. Já a codificação X4 utiliza a subida e descida dos dois canais, isso faz com que, em um pulso, sejam gerados quatro sinais de incremento ou decremento. Ou seja, um *encoder* com codicação X4 e com resolução de 1024 PPR irá gerar 4096 contagens por revolução.

Alguns processadores possuem temporizadores (*timers*) que já possuem implementação de codificação em hardware. Por exemplo, os microprocessadores da família STM32F7 possuem um modo nos temporizadores de propósito geral chamado *encoder mode*, onde um *encoder* incremental pode ser conectado diretamente no microprocessador sem nenhuma interface lógica externa (STMICROELECTRONICS, 2019). A Figura 6 mostra um exemplo de operação de contagem de pulsos com codificação X1, X2 e X4 num microprocessador da família STM32F7. Os canais A e B são conectados, respectivamente, às entradas dos *timers* de propósito geral TI1 e TI2, e a direção da contagem depende de qual canal está à frente na subida. Se o canal A está à frente do canal B, acontece o incremento, já quando o canal B está à frente do canal A, ocorre o decremento. A quantidade de pulsos lida pelo *encoder* pode ser transformada em velocidade rotacional, como será detalhado na seção 2.2.2 a seguir.

2.2.2 Cálculo da velocidade

Para calcular a velocidade de um motor, é preciso ter a medida da quantidade de pulsos que foram lidos num intervalo de tempo específico. Com isso, é possível estimar a taxa de deslocamento de acordo com as unidades de medida utilizadas. A obtenção da velocidade (em rad/s) utilizando um *encoder* do tipo incremental pode ser obtida através da Equação 2.1:

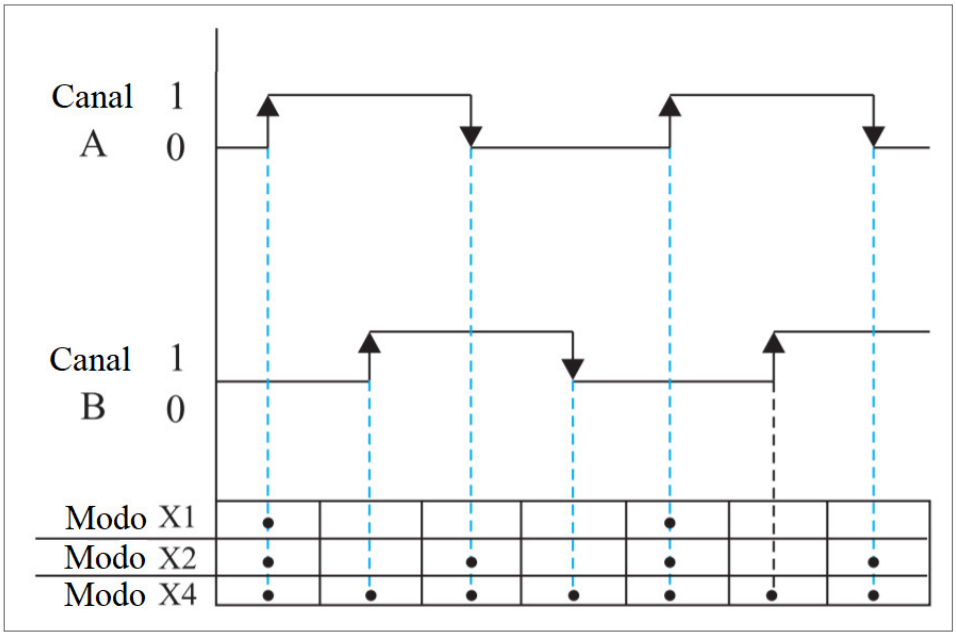
$$\omega_{motor} = 2\pi \cdot \frac{Pulses}{EncoderMode \cdot PPR \cdot t_{sample}} \tag{2.1}$$

$$\Delta = 2\pi \cdot \frac{1}{EncoderMode \cdot PPR \cdot t_{sample}} \tag{2.2}$$

Onde:

- $Pulses$ é a quantidade de pulsos lido pelo *encoder* no intervalo.

Figura 6 – Exemplo de operação de contagem de um temporizador no *encoder mode* para os três tipos de codificação.



Fonte: (STMICROELECTRONICS, 2019) (ADAPTADO).

- $EncoderMode$ é o número referente ao modo de codificação utilizado, por exemplo, se for utilizado o modo X2, o $EncoderMode$ será 2.
- PPR é o valor de Pulsos Por Revolução do *encoder*, já definido na seção 2.2.1.
- t_{sample} é o tempo de amostragem do *encoder* (em segundos), ou seja, o quão rápido o *encoder* é atualizado.

Para o controle de motores, o valor de tempo de amostragem (t_{sample}) é crucial, pois, ele determina a frequência de atualização do sistema, ou seja, quanto tempo vai levar para o sistema atualizar a medida da velocidade. O impacto da escolha do tempo de amostragem é estudado para um *encoder* real na seção 4.3.1. No caso de motores com transmissão por engrenagens, a velocidade da roda pode ser calculada multiplicando a velocidade do motor, obtida através da equação 2.1, pela relação de transmissão entre as engrenagens do motor e da roda.

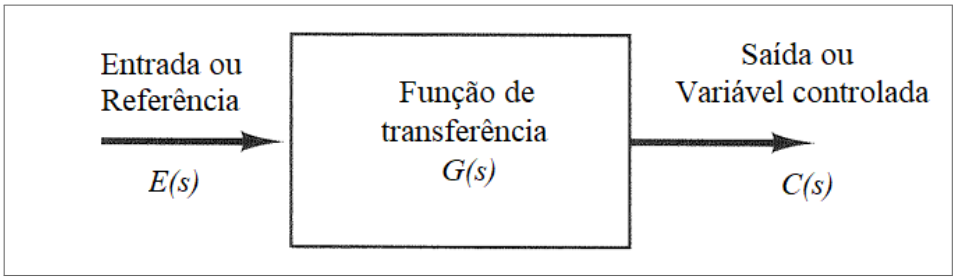
2.3 SISTEMAS DE CONTROLE

De acordo com Keesman (2011), um sistema é um objeto no qual diferentes variáveis interagem em várias escalas de tempo e espaço, produzindo sinais observáveis. Já um sistema de controle, também chamado de planta, é uma parte de um equipamento cuja finalidade é desempenhar uma determinada operação (OGATA, 2003). Um sistema de controle pode ser qualquer objeto físico a ser controlado através de um processo, que é uma operação a ser

controlada. A representação das funções exercidas por um sistema de controle pode ser feita através de um diagrama de blocos, que é uma representação gráfica do fluxo de sinais de entrada e saída de um sistema.

Os elementos de um diagrama de blocos podem ser descritos a partir de um sistema de primeira ordem no domínio da frequência mostrado na Figura 7, onde a seta apontada para o bloco se refere ao sinal de entrada, e a seta saindo do bloco indica o sinal de saída, e o bloco em si é a representação de uma função de transferência, que contém informações referentes ao comportamento dinâmico do sistema, ou seja, como ele deve atuar na saída, dados os valores de entrada.

Figura 7 – Diagrama de blocos de um sistema padrão.



Fonte: (NISE, 2007) (ADAPTADO).

A relação entre a entrada e saída pode ser dada pela equação 2.3 a seguir:

$$C(s) = G(s)E(s) \tag{2.3}$$

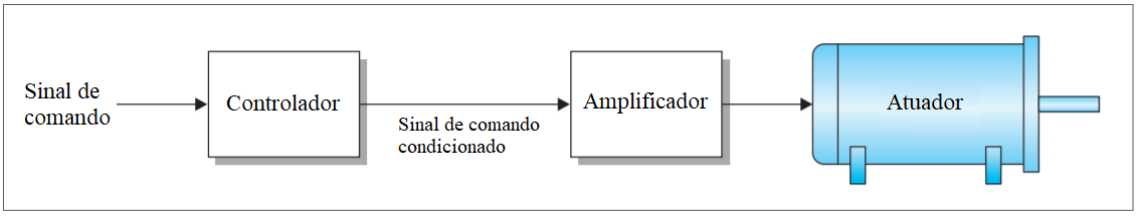
Portanto, um diagrama de blocos mostra como o fluxo de sinais entre os elementos do sistema acontece, para suportar a derivação de modelos matemáticos que os representam. Além disso, os diagramas auxiliam encontrar uma relação entre diferentes elementos do sistema, como o processo, os sinais, e o controlador.

2.3.1 Sistemas em malha aberta e malha fechada

De acordo com Nise, existem duas configurações principais de sistemas de controle: sistemas de controle em malha aberta e sistemas em malha fechada. Um exemplo de sistema em malha aberta é ilustrado na Figura 8 (NISE, 2007). O sinal de entrada, também chamado de sinal de referência, ou sinal de comando, é enviado para um controlador que entrega um sinal condicionado para um amplificador, que por sua vez envia o sinal amplificado para uma planta (representada por um atuador), que irá produzir uma saída, também chamada de variável controlada. É importante ressaltar que distúrbios causados pelo ambiente em que o atuador está inserido podem interferir no sistema real, somando ações que alteram o comportamento do processo. Uma das desvantagens de sistemas em malha aberta é que eles não podem compensar esses distúrbios de maneira a corrigir o resultado da variável controlada para o valor desejado pelo sinal de entrada, porque não há uma realimentação (*feedback*) da leitura

do atuador. Em alguns casos, os distúrbios podem corromper o sinal de saída de forma a tornar impossível que o sistema seja controlado em malha aberta.

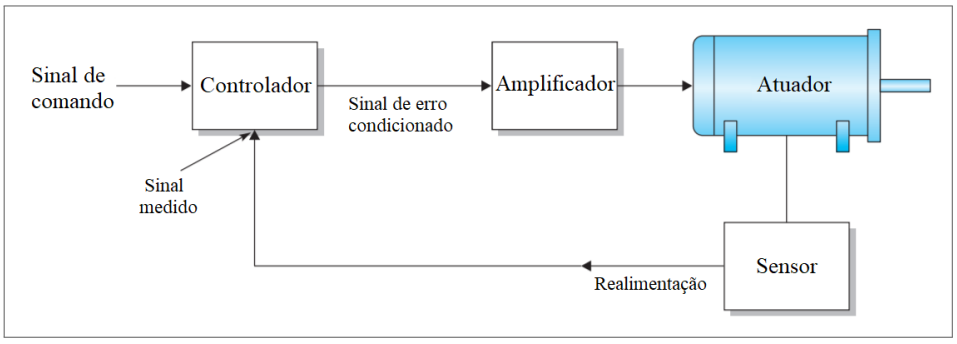
Figura 8 – Diagrama de blocos de um sistema em malha aberta.



Fonte: (BARTELT, 2010) (ADAPTADO).

Já os sistemas em malha fechada possuem um *feedback* da saída, ou seja, além do sinal de comando, um sinal de resposta do atuador é medido através de um sensor, e enviado para o controlador, para que ele possa agir de maneira a compensar o erro entre o sinal atual e o sinal de comando, como mostra a Figura 9. A principal vantagem de sistemas em malha fechada é que, mesmo com a presença de distúrbios, o controle tende a reduzir a diferença entre o sinal de saída do atuador, e o sinal de referência, ou seja, o sistema compensa perturbações previsíveis (OGATA, 2003). O controlador faz a comparação entre o valor real de saída e o valor de referência, e de acordo com o erro entre os valores desses dois sinais, produz um sinal de controle de modo a reduzir esse erro para um valor próximo a zero. Em geral, para obter um controlador bem projetado, é preciso que o sistema seja representado matematicamente através de uma função de transferência. O processo de obtenção da função de transferência de um sistema é chamado de modelagem. Uma metodologia para modelagem de sistemas será vista na subseção 2.3.2 a seguir.

Figura 9 – Diagrama de blocos de um sistema em malha fechada.



Fonte: (BARTELT, 2010) (ADAPTADO).

2.3.2 Modelagem de sistemas

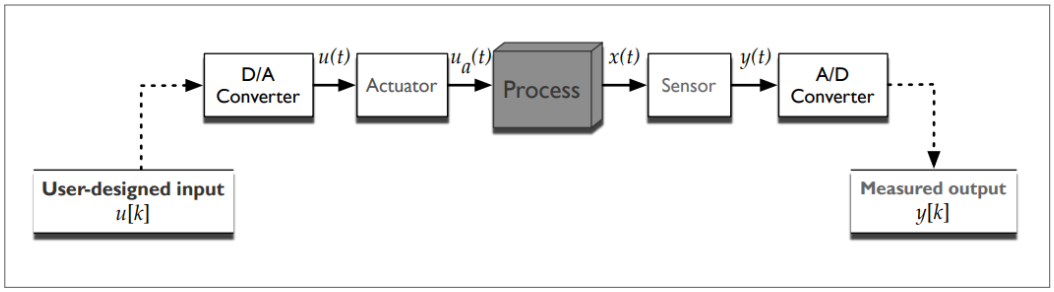
De acordo com Keesman (2011), durante o procedimento de identificação de sistemas, a escolha de modelos candidatos é o passo mais importante. Tangirala divide um procedi-

mento para identificação de sistemas em cinco passos: geração e aquisição de dados, pré-processamento de dados, visualização dos dados, desenvolvimento do modelo e validação do modelo (TANGIRALA, 2014).

2.3.2.1 Geração e aquisição de dados

O primeiro passo é o mais importante na identificação de sistemas, pois, cada estágio seguinte depende da quantidade e qualidade dos dados adquiridos. Um diagrama de aquisição de dados amostrais típico pode ser visualizado na figura 10. A entrada em tempo discreto, usualmente fornecida por um microcontrolador digital é convertida para um sinal contínuo por um Conversor Digital-Analógico (DCA), que age sobre o processo, de acordo com a figura, com a ajuda de um atuador. A leitura da resposta do sistema é feita com a ajuda de um sensor que irá amostrar e quantizar a medida. Os sensores incluem, em geral, um Conversor Analógico-Digital (ADC), que irá amostrar o dado e enviar para algum dispositivo de armazenamento de dados.

Figura 10 – Sistema de amostragem de dados padrão.



Fonte: (TANGIRALA, 2014).

As principais variáveis de decisão neste passo são o tipo de sinal de entrada em tempo discreto e a taxa de amostragem do sensor. O tipo de sinal de entrada deve causar um efeito na saída maior que os efeitos causados por ruídos ou distúrbios no sensor. Já a taxa de amostragem do sensor se refere a capacidade de obter dados que são informativos, que deve ser grande o suficiente para que seja possível identificar o estado transiente da resposta e que permita ao microcontrolador processar os dados adquiridos dentro do ciclo de amostragem.

2.3.2.2 Pré-processamento e visualização dos dados

Os dados adquiridos de forma bruta nem sempre estão prontos para serem utilizados. Em geral, existe um passo de pré-processamento antes dos dados serem enviados para o algoritmo de estimação de um modelo. Além do ruído do sensor, outros fatores afetam a qualidade dos dados, os dois principais são: a presença de (*outliers*), e dados faltantes.

Os *outliers* são valores de dados que não são condizentes com a maioria dos valores dos dados adquiridos, e a sua leitura pode ter sido causada por algum tipo de mau funcionamento do sensor, ou de um problema no processo de execução do sistema. Caracterizar e lidar com

outliers no conjunto de dados pode ser bastante desafiador, e devem ser usados métodos estatísticos para identificá-los e eliminá-los.

O problema de dados faltantes é um dos problemas mais comuns na amostragem de sensores. Algumas das razões para a perda de dados são: falha do sensor, amostragem não uniforme e perda durante a transferência de dados (principalmente em comunicações sem-fio). A perda de dados pode prejudicar a análise do sistema em regime transiente, mas, algumas técnicas para lidar com dados faltantes baseadas em aprendizagem de máquina são apresentados por Tangirala (2014). Em geral, a escolha de uma taxa de amostragem muito alta pode colaborar para o aumento da perda de dados.

O conjunto de métodos para lidar com os dados brutos contendo ruídos e *outliers* é chamado de pré-filtragem. Essa etapa pode levar um tempo e esforço significativos, por isso, é importante escolher uma metodologia de aquisição de dados confiável, e executar o experimento da maneira correta.

A inspeção visual dos dados é uma etapa chave no processo extração de informações e análise de sinais. Essa etapa permite a identificação manual de *outliers* e outras anomalias que não foram tratadas durante a etapa de pré-processamento, além de permitir a verificação dos dados do ponto de vista qualitativo. A visualização permite também a identificação de informações preliminares como o comportamento dinâmico, e extração de características do sistema que podem auxiliar na escolha do algoritmo de identificação de modelo mais apropriado na próxima etapa.

2.3.2.3 Desenvolvimento do modelo

Desenvolver o modelo é o principal objetivo da identificação, e envolve a construção de um modelo determinístico e estocástico. Ou seja, de um modelo que seja capaz de capturar a dinâmica do processo físico (determinístico), e que tenha uma descrição matemática das incertezas com precisão e acurácia (estocástico). Esse processo de identificação envolve dois passos: especificar a estrutura e ordem do modelo, e estimar os parâmetros do modelo especificado.

Escolher os modelos candidatos muitas vezes consiste num processo iterativo que envolve um conjunto de características. Os requisitos de precisão e acurácia devem ser parâmetros da estrutura do algoritmo de identificação do modelo, e devem levar em consideração a frequência de amostragem do sensor utilizado. A aplicação final do modelo também impõe alguns requisitos adicionais, por exemplo, para aplicações de controle, o modelo deve ser o mais simples possível (menor ordem). Um conhecimento a priori de informações do modelo também pode auxiliar na escolha da estrutura paramétrica, como saber se o sistema é ou não linear, ou se a ordem do sistema é alta ou baixa.

Uma vez com o modelo candidato selecionado, resta a estimativa dos parâmetros, que consiste em um problema de otimização. Muitos métodos de estimativa buscam a minimização de algum tipo de função de custo, por exemplo, o método dos mínimos quadrados busca

minimizar a distância euclidiana entre os valores previsto e observado. Os parâmetros que devem ser considerados na hora de escolher um algoritmo de estimativa são a qualidade do ajuste da estimativa e o custo computacional, que em geral, são fatores conflitantes, no entanto é possível encontrar um meio termo entre ambos.

2.3.2.4 Validação do Modelo

Em geral, as etapas anteriores são feitas num mesmo conjunto de dados, chamado de dados de treinamento. Para entender o quão bem o modelo obtido representa as variações de saída dos dados de treinamento, algumas análises são feitas com o objetivo de obter um modelo com o menor erro de previsão possível e boa precisão, são elas: análise residual, análise de estimativas, e análise de ajuste do modelo (métricas como *R-square* são utilizadas nesta análise).

Entretanto, para validar o modelo, é necessário saber o quão bem esse modelo responde a um conjunto de dados de teste totalmente novo. Essa etapa é chamada de validação cruzada, e tem o objetivo de descobrir se os dados de treinamento capturaram as características globais do processo, e se conseguem generalizar bem o comportamento do sistema para outros comportamentos, eliminando o *overfitting*.

2.3.2.5 Conhecimento a priori do processo

Na prática, um conhecimento prévio do sistema a ser trabalhado facilita a escolha de parâmetros durante a obtenção de um modelo matemático que representa um sistema. Informações como a estrutura do modelo, ordem do modelo, valores de algumas medidas, e parâmetros de fronteira podem estar disponíveis. Quanto ao grau de conhecimento do sistema, ele pode ser classificado de três maneiras: *white-box*, *grey-box* ou *black-box*.

Quando um modelo matemático do sistema é conhecido e restam apenas os parâmetros a serem determinados, o processo de obtenção do modelo é chamado de *white-box*. Quando apenas alguns aspectos do sistema são conhecidos, ou existe apenas um conhecimento parcial da estrutura e dos parâmetros, o processo é chamado modelagem *grey-box*. Já quando não há nenhuma informação acerca do processo, e as únicas informações disponíveis são os dados de entrada e de saída do sistema, o processo é chamado de modelagem *black-box*.

Embora algumas críticas sejam feitas ao processo *black-box* por não levar em conta os aspectos físicos do processo, com uma análise criteriosa dos dados, a identificação do sistema pode produzir um modelo funcional tão bom quanto um obtido através de outras modelagens, desde que sejam usados dados de boa qualidade para o treinamento e validação. Na Seção 2.3.3, serão descritos alguns experimentos que usualmente auxiliam na identificação de modelos de motores DC utilizando a metodologia *black-box*.

2.3.3 Experimentos de identificação de sistemas

A escolha do experimento a ser realizado para identificação do sistema impacta diretamente no resultado do modelo final. Esta seção irá tratar de três tipos de experimento, resposta em degrau, resposta utilizando uma sequência binária pseudo-aleatória e sinal pseudo-aleatório de múltiplos níveis. A seguir serão apresentadas as características de cada experimento, bem como onde sua aplicação é mais indicada.

2.3.3.1 Entrada em Degrau

Um sinal em degrau representa um comando constante, como uma tensão, corrente ou velocidade, por exemplo. Em geral, a entrada representa um valor de referência, ou valor desejado do mesmo tipo da resposta do sistema. Ou seja, se o sistema retorna uma velocidade atual, a entrada irá corresponder a uma velocidade desejada. Uma entrada em degrau consiste em variar a entrada uma única vez, do zero para uma entrada unitária, e é definida pela seguinte relação:

$$u(t) = \begin{cases} 1, & \text{para } t \geq 0. \\ 0, & \text{para } t < 0. \end{cases} \tag{2.4}$$

De acordo com Ogata (2003), através de uma entrada em degrau, podem ser identificados alguns parâmetros de sistemas de primeira ordem, como: constante de tempo e valor em regime permanente. Os experimentos utilizando entrada em degrau são mais utilizados em sistemas lineares de primeira ordem devido a incapacidade de identificar não-linearidades causadas por aspectos de frequência.

2.3.3.2 Sequência Binária Pseudo-Aleatória

Um dos métodos mais utilizados para geração de sinais de entrada se baseia em uma Sequência Binária Pseudo-Aleatória, ou *Pseudo-Random Binary Sequence* (PRBS), que pode ser utilizada como forma de estímulo para entradas em motores com o intuito de obter seu modelo dinâmico (SCHREIBER; ISERMANN, 2007). Os sinais contendo sequências binárias pseudo-aleatórias são compostos por pulsos retangulares de frequência variada, onde a amplitude do sinal deve variar dentro da faixa de operação do sistema (por volta de 10% da amplitude máxima), desde que seja possível diferenciar a resposta do sistema de ruídos ou distúrbios inerentes ao ambiente. A existência de sinais em múltiplas frequências dá uma maior robustez ao sistema quanto à presença de não-linearidades e faz com que essa técnica seja indicada para identificação de sistemas de ordens mais elevadas.

A geração de um sinal PRBS deve seguir algumas restrições, por exemplo, o período de amostragem da sequência deve ter um valor maior que o tempo de subida do sistema. Já a

quantidade de amostras do sinal deve seguir a seguinte relação:

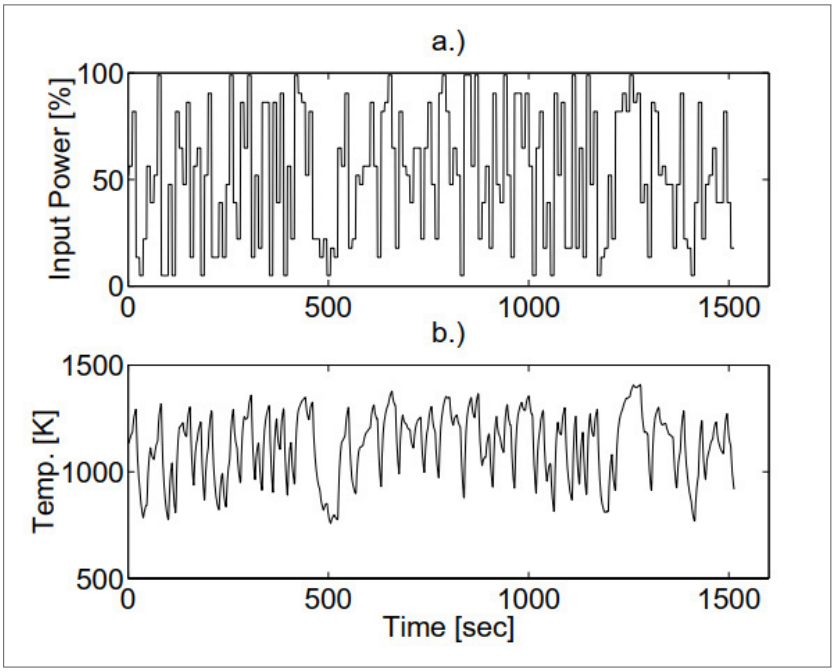
$$N = 2^n - 1 \tag{2.5}$$

Onde N é o número de amostras, e n é a quantidade de bits que serão utilizados para a geração do sinal PRBS. A justificativa para esse valor do sinal é devido a facilidade de gerar sinais utilizando registradores de deslocamento *shift-registers* (TAN; GODFREY, 2001). O período de amostragem deve ser grande o suficiente para que seja possível diferenciar ruídos e não-linearidades do sistema da resposta real.

2.3.3.3 Sinal Pseudo-Aleatório de Múltiplos Níveis

De acordo com Braun et al. (1999), os sinais PRBS podem não fornecer informações suficientes para identificação de não-linearidades do sistema. Para isso, são utilizadas as sequências pseudo-aleatórias de múltiplos níveis, ou *Multi-level Pseudo Random Sequence* (MPRS). A principal diferença entre uma sequência PRBS e MPRS é a variação na amplitude do sinal de entrada, que ao invés de ser entre dois valores dentro da faixa de operação do sistema, são utilizados sinais pseudo-aleatórios variando entre toda a faixa de operação do sistema. Essa sequência permite uma estimativa do comportamento dinâmico do sistema eliminando as não-linearidades do sistema. A Figura 11 mostra um exemplo de sinal MPRS como entrada de um sistema de controle de temperatura, onde o sinal de entrada do sistema varia entre 5% e 99% da potência máxima.

Figura 11 – Sinais de Entrada (a) e Saída (b) de um sistema utilizando MPRS.



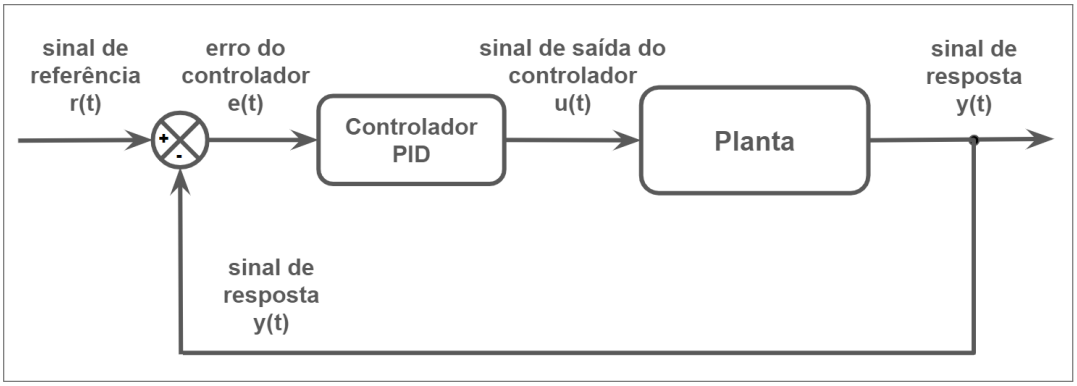
Fonte: (BRAUN et al., 1999).

2.4 CONTROLADORES PID

Após a identificação do sistema, é necessário controlá-lo. Esta seção irá discorrer sobre o projeto de um controlador PID, apontando suas vantagens e desvantagens com o objetivo de fundamentar a escolha da técnica implementada neste trabalho. A Figura 12 mostra um sistema de controle típico em malha fechada. O erro $e(t)$ é a diferença entre o sinal de referência $r(t)$ e o sinal de resposta da planta $y(t)$. O sinal de saída do controlador PID $u(t)$ alimenta a entrada para a planta do sistema e é calculado através da seguinte equação:

$$u(t) = K_p \cdot e(t) + K_i \int e(t)dt + K_p \frac{de(t)}{dt} \tag{2.6}$$

Figura 12 – Sistema de controle padrão em malha fechada.



Fonte: o autor.

Ou seja, o sinal de controle é igual ao ganho proporcional (K_p) multiplicado pelo erro, somado ao ganho integral (K_i) multiplicado pelo erro acumulado, mais o ganho derivativo (K_d) multiplicado pela taxa de variação do erro. A seguir serão discutidas algumas características de cada um dos ganhos citados.

O ganho proporcional fornece uma contribuição que depende apenas do valor instantâneo do erro de controle, ou seja, ele age de acordo com o estado atual do sistema, sem levar em consideração iterações anteriores (GOODWIN et al., 2001). Embora os controladores proporcionais sejam capazes de controlar qualquer sistema estável de maneira rápida, esse controle possui um desempenho limitado, aumenta a a probabilidade de *overshoot*, isto é valores muito altos na entrada podem retirar o sistema do regime permanente; e podem não eliminar o erro em regime permanente.

O acréscimo do ganho integral adiciona uma ação de saída que é proporcional ao erro acumulado, o que implica numa reação um pouco mais lenta, mas que força o erro em regime permanente para zero mesmo na presença de distúrbios (GOODWIN et al., 2001). Um aspecto que deve ser levado em consideração no projeto de controladores com ganho integral é a presença de um efeito indesejado diante da saturação do atuador conhecido como *wind-up*. Esse fenômeno ocorre quando o erro persiste por um longo tempo, e o termo integral fica muito grande, causando a saturação do atuador, ou seja, leva o atuador para o seu limite

físico por longos períodos de tempo. Uma forma simples de prevenir esse fenômeno é limitar o ganho integral, fazendo com que o valor do ganho integral não ultrapasse um valor específico dentro dos limites de funcionamento do atuador.

O ganho derivativo age na taxa de mudança do erro de controle, isso faz com que o sistema apresente uma ação preditiva para acompanhar a tendência do erro (GOODWIN et al., 2001). Entretanto, isso torna o sistema mais suscetível a ruídos e perturbações que podem prejudicar o desempenho do controlador. Portanto, não é indicado para sistemas com taxas de amostragem muito altas onde o ruído em altas frequências tem maior influência. Alguns dos efeitos do ganho derivativo são a redução do *overshoot* e uma melhora na resposta transiente.

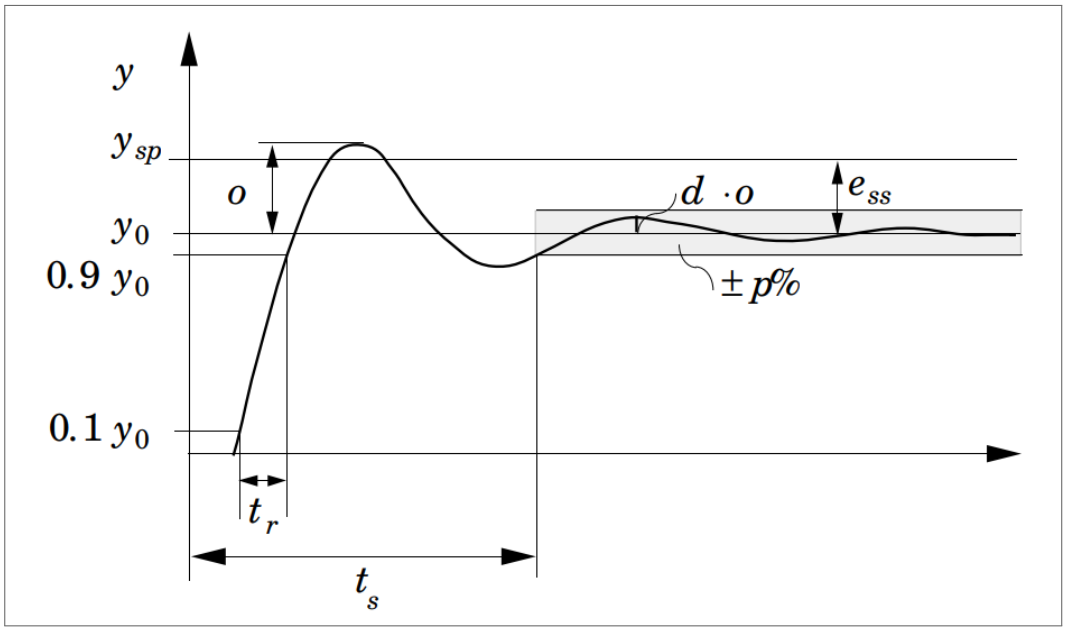
2.4.1 Ferramentas de tuning de controladores PID

O procedimento para selecionar os parâmetros do controlador de modo a atender às especificações de desempenho é chamado de ajuste fino (*tuning*) do controlador. Em geral, os controladores PID buscam resolver dois problemas principais: seguir um valor de referência e rejeitar distúrbios (ÅSTRÖM; HÄGGLUND, 1995). Portanto, deve-se conhecer as restrições do sistema, que em geral estão relacionadas à dinâmica do sistema, não-linearidades, distúrbios e incertezas associadas ao processo. Já as especificações do sistema podem ser expressas na forma de resposta ao tempo, ou resposta à frequência. Este trabalho irá utilizar especificações na forma de resposta ao tempo.

Os requisitos do controlador podem estar relacionados ao tempo de subida, tempo de assentamento, (*overshoot*) e erro em regime permanente. A Figura 13 mostra como são definidos esses requisitos quando aplicada uma entrada em degrau unitário com entrada de referência (y_{sp}). O tempo de subida (t_r) é definido como o tempo que a resposta à entrada em degrau passa de 10% para 90% do valor em regime permanente (y_o). O tempo de assentamento (t_s) é o tempo em que a resposta à entrada em degrau leva para ficar dentro de um intervalo de ($p\%$) de y_o, em geral o valor de $p = 2\%$ é adotado pela literatura. O valor de *overshoot* (o) é a razão da diferença entre o valor do primeiro pico de resposta e o valor em regime permanente y_o, em geral o valor do $overshoot$ não deve ultrapassar os 10% do valor em regime permanente. O erro em regime permanente (e_{ss}) é o valor do erro do controlador quando o sistema estabiliza, ou seja, a diferença entre o sinal de entrada de referência y_{sp} e o valor da saída em regime permanente y_o. Lembrando que quando há uma ganho integral no controlador, o valor de e_{ss} é sempre zero.

Fazer o *tuning* de um controlador PID aparenta ser uma tarefa fácil, pois, basta encontrar os valores das constantes de ganho proporcional, integral e derivativo. Entretanto, encontrar esse conjunto de constantes garantindo o melhor desempenho do sistema a ser controlado é uma tarefa extremamente complexa (ABU-KHALAF; TUREVSKIY, 2009). Tradicionalmente, os controladores PID são otimizados de maneira manual ou utilizando abordagens clássicas. Os métodos manuais, como o de tentativas e erros, são iterativos e demandam experiência do projetista e muito tempo de teste, além da possibilidade de causar danos ao sistema em

Figura 13 – Especificações de resposta de um sistema de controle



Fonte: (ÅSTRÖM; HÄGGLUND, 1995).

ajuste. Já as abordagens clássicas possuem limitações de modelo e não são capazes de tratar plantas de ordem superior. A seguir serão abordados o método clássico de Ziegler-Nichols, e um método utilizando a ferramenta *PIDTuner* do software MATLAB.

A Tabela 1 resume os efeitos de aumentar cada parâmetro de um controlador PID num sistema em malha fechada. É importante ressaltar que as orientações da tabela são válidas para boa parte dos casos, mas não para todos. Ou seja, ainda é necessário executar testes no sistema a ser controlado para validar as informações obtidas.

Tabela 1 – Efeitos relacionados ao aumento de cada um dos parâmetros de controle.

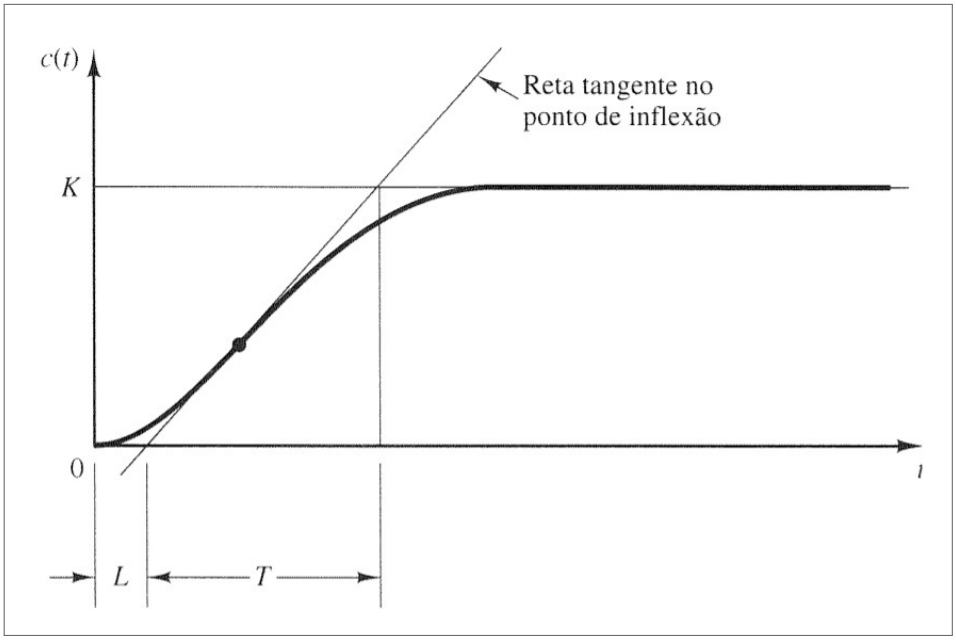
Parâmetro de Controle	**Tempo de Subida**	***Overshoot***	**Tempo de Assentamento**	**Erro em regime Permanente**
Kp	Diminui	Aumenta	Pouca Mudança	Diminui
Ki	Pouca Mudança	Aumenta	Aumenta	Diminui
Kd	Diminui	Diminui	Diminui	Não Muda

Fonte: (TUTORIALS, 2019).

2.4.1.1 Método de Ziegler-Nichols

De acordo com Ogata (2003), o método de Ziegler-Nichols é amplamente utilizado para o *tuning* de controladores PID em sistemas de controle de processos quando a dinâmica do processo a controlar é desconhecida. A abordagem de Ziegler-Nichols consiste em determinar experimentalmente os valores de ganho proporcional (K_P), tempo integral (T_I) e tempo derivativo (T_D) baseado nas informações em regime transiente do processo a ser controlado. O experimento consiste em obter a resposta do sistema a uma entrada degrau, que deve se assemelhar à curva da Figura 14, em formato de "S". Ao traçar uma reta tangente ao ponto de inflexão da curva, é possível obter duas constantes: o tempo de retardo (L) e a constante de tempo (T). Os ponto de interesse são os pontos onde a reta tangente ao ponto de inflexão da curva tocam as retas $c(t) = K$ e $c(t) = 0$.

Figura 14 – Exemplo de curva de resposta em forma de "S".



Fonte: (OGATA, 2003).

A função de transferência de um controlador PID pode ser representado da seguinte forma:

$$G_c(s) = K_P + \frac{K_I}{s} + K_D \cdot s \tag{2.7}$$

A equação 2.7 pode ser reescrita de maneira a explicitar os termos que são utilizados na técnica de Ziegler-Nichols, sendo assim, temos:

$$G_c(s) = K_P \cdot \left(1 + \frac{1}{T_I} \cdot s + T_D \cdot s\right) \tag{2.8}$$

A abordagem de Ziegler-Nichols sugere o ajuste dos valores das constantes K_P, T_I e T_D de acordo com o tipo de controlador a ser projetado. A Tabela 2 mostra quais devem ser os ajustes necessários para as constantes, dada a escolha do tipo de controlador.

Tabela 2 – Regra de *tuning* de Ziegler-Nichols baseada em uma resposta em degrau.

Controlador	K_P	T_I	T_D
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Fonte: (OGATA, 2003).

Entretanto, segundo Ogata (2003), essa abordagem só é aplicada para funções de transferência que apresentam, na resposta a um degrau unitário, uma curva transiente em forma de 's', ou se o sinal de saída apresentar oscilações. Caso contrário, não é possível aplicar essa técnica à planta obtida.

2.4.1.2 PIDTuner

O software MATLAB conta com uma ferramenta para *tuning* automático de controladores PID chamada de *PIDTuner*. Com essa ferramenta, os parâmetros do controlador PID podem ser ajustados automaticamente para se ter um projeto robusto e com a resposta ao tempo desejada (DONGALE et al., 2012).

Com a ferramenta *PIDTuner*, é possível projetar um controlador adequado automaticamente, especificando o tipo de controlador (P, PI, PD, PID) em diferentes formas. Se o sistema a controlar for um modelo em tempo discreto com tempo de amostragem T_s, a ferramenta projeta um controlador em tempo discreto utilizando o método de Euler avançado com integrador discreto.

O primeiro passo para a utilização da ferramenta é importar uma planta ou modelo a controlar. Em seguida, deve-se escolher qual o tipo de controlador será utilizado, e também é possível especificar se o foco do projeto é em seguir uma referência, rejeitar distúrbios, ou um balanço entre as duas opções. É possível refinar o projeto de maneira interativa tanto no domínio do tempo, quanto no domínio da frequência, este trabalho irá utilizar o domínio do tempo para refinar o controlador.

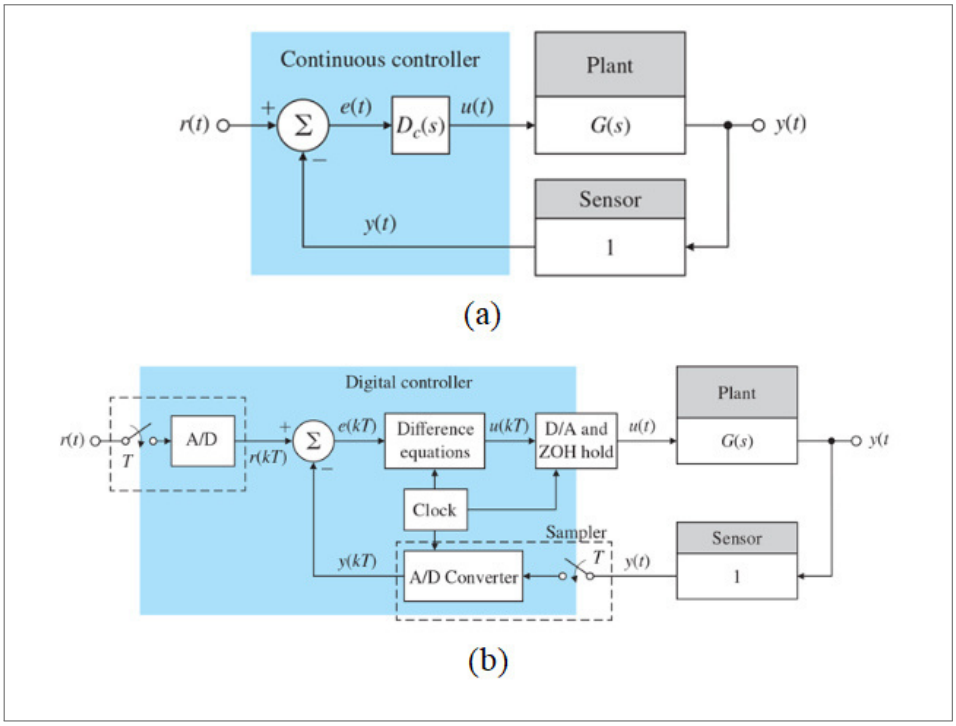
Uma das variáveis de controle no domínio do tempo é o tempo de resposta para que o sistema reaja de forma mais rápida ou mais lenta. Outra variável no domínio do tempo é o comportamento em regime transiente, que torna o controlador mais agressivo quanto

à rejeição de distúrbios ou mais robusto quanto às incertezas. A cada mudança é possível visualizar características do sistema como tempo de subida, tempo de assentamento e erro em regime permanente para decidir se o controlador atende ou não aos requisitos desejados. Por fim, a ferramenta retorna as constantes de controle para que o projetista possa implementar e validar o controlador em seu projeto.

2.5 DIGITALIZAÇÃO

A maioria dos sistemas de controle de hoje utilizam computadores digitais (usualmente microprocessadores ou microcontroladores) para implementar os controladores (FRANKLIN; POWELL; EMAMI-NAEINI, 2014). Diferentemente da eletrônica analógica, para um computador digital integrar e diferenciar sinais, as equações diferenciais que regem esses sinais devem ser simplificadas a expressões envolvendo adição, multiplicação e divisão. A Figura 15 (a) mostra uma topologia de um sistema contínuo típico que vem sendo considerado nas seções anteriores. Já a Figura 15 (b) apresenta um sistema equivalente, só que executado num computador digital. A principal diferença entre um sistema contínuo e um digital é a necessidade de mecanismo de amostragem no sensor de saída da planta. Da mesma forma, o sinal de controle fornecido pelo controlador precisa ser gerado em tempo discreto e aproximado utilizando métodos numéricos também chamados de equação de diferença.

Figura 15 – Diagrama de blocos de um sistema de controle básico: (a) sistema contínuo; (b) sistema digital.



Fonte: (FRANKLIN; POWELL; EMAMI-NAEINI, 2014).

Num processo digital, os valores da saída analógica da planta são amostrados e con-

vertidos em valores digitais através de um ADC. Olhando para a Figura 15(b), a conversão de um sinal contínuo analógico ($y(t)$) para amostras digitais discretas ($y(kT)$) ocorre a cada período de tempo (T), que é chamado de período de amostragem. Esse período de amostragem é gerado pelo computador através de um *clock*, que gera um pulso ou interrupção a cada T segundos. O sinal do tipo $y(kT)$ é chamado de sinal discreto, e um sistema que contém ambos sinais contínuos e discretos é chamado de sistema de dados amostrais. O resultado da equação de diferenças é um sinal discreto ($u(kT)$) para cada sinal amostrado. Esse sinal é convertido para um sinal de controle contínuo ($u(t)$) através de um DCA e um dos métodos mais utilizados para manter o sinal constante durante o período de amostragem é chamado de *Zero-Order-Hold* (ZOH). O sinal de controle é, então, aplicado na planta da mesma maneira que ocorre em uma implementação com valores contínuos.

2.6 CINEMÁTICA DE ROBÔS MÓVEIS

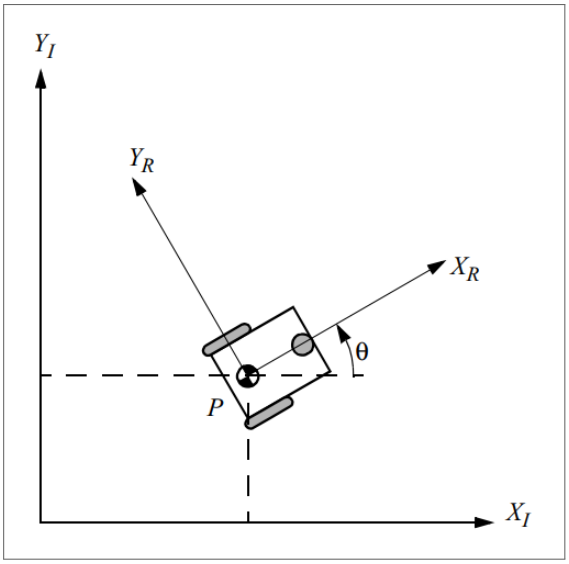
Esta seção irá contribuir para um melhor entendimento de como funciona o processo de modelagem cinemática do robô utilizado no estudo de caso para validação da abordagem proposta. A cinemática é o estudo mais básico de como os sistemas mecânicos se comportam. Na robótica móvel, é preciso entender o comportamento mecânico do robô. A principal diferença entre um robô móvel e um braço robótico é a necessidade da estimativa de posição, uma vez que manipuladores tem uma posição inicial e final fixas no ambiente.

O processo de entender o movimento de um robô no espaço começa com a descrição da contribuição de cada roda para o movimento. Cada roda contribui para a movimentação do robô assim como também pode impor restrições ao movimento do robô. Nos próximos parágrafos, serão introduzidos conceitos e notações relacionados ao movimento do robô tanto considerando um referencial global como um referencial local. A partir destes conceitos e notações será descrita a cinemática direta do modelo de movimento, descrevendo como o robô inteiro se move em função de sua geometria e do comportamento individual de cada roda. Em seguida, serão descritas restrições cinemáticas de cada roda individualmente, e estas restrições serão combinadas para gerar a restrição cinemática de robô completo na seção 5.3. Com essas ferramentas, serão avaliados os caminhos e trajetórias que definem a manobrabilidade do robô.

Durante o processo de modelagem, será utilizada a notação e terminologia apresentadas por Campion, Bastin e Dandrea-Novel (1996). Considerando o robô (chassis) como um corpo rígido sobre rodas que opera em um plano horizontal, a dimensionalidade total para esse robô no plano é três. Duas dimensões para posição no plano e uma para orientação ao longo do eixo vertical, que é ortogonal ao plano. Para especificar a posição do robô no plano, é necessário estabelecer uma relação entre o quadro (*frame*) global de referência do plano e o local do robô. A Figura 16 ilustra um frame de referência global, dado pelos eixos X_I e Y_I, e local, dado pelos eixos X_R, Y_R. A posição P do chassis do robô é especificada pelas coordenadas x e y do ponto P, e a diferença angular entre os *frames* global e local de referência é dado por

θ. A posição do robô pode ser descrita como um um vetor com esses três elementos, dada pela Equação 2.9 a seguir:

Figura 16 – *Frame* de referência global e *frame* de referência local.



Fonte: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.9}$$

O subscrito I é utilizado para deixar claro que a base desta posição é o *frame* de referência global. Para descrever o movimento do robô em termos de componente de movimento, será necessário mapear o movimento ao longo dos eixos do *frame* de referência global para o movimento ao longo do eixo do *frame* de referência local do robô. Esse mapeamento é uma função da posição atual do robô, e é realizado através da matriz de rotação ortogonal dada pela Equação 2.10 a seguir:

$$R(\theta) = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Utiliza-se essa matriz para mapear o movimento no quadro (*frame*) de referência local X_I, Y_I para o movimento em termos de *frame* de referência local X_R, Y_R. Essa operação, também chamada de rotação, é descrita ela Equação 2.11 a seguir:

$$\dot{\xi_R} = R(\theta)\dot{\xi_I} \tag{2.11}$$

2.6.1 Locomoção

Um robô móvel precisa de mecanismos de locomoção que o permitam interagir com o ambiente (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Decidir qual abordagem deve ser utilizada para a execução de movimentos é fundamental no projeto de robôs móveis autônomos devido ao grande número de possibilidades existentes. A roda é o mais popular meio de locomoção em robôs móveis e veículos em geral, pois, não é necessário se preocupar com o balanço, e sim com parâmetros como tração, estabilidade, manobrabilidade e controle. Mais de três rodas fornecem redundância ao sistema, tornando-o sobre-determinado.

De acordo com Siegwart, Nourbakhsh e Scaramuzza (2011), existem quatro grandes classes de rodas: roda padrão, roda castor, roda sueca (omnidirecional) e roda esférica. A escolha do tipo de roda tem um impacto grande em toda a cinemática do robô. A roda padrão lida com a aceleração sem nenhum efeito lateral, enquanto que a roda castor rotaciona em torno de um eixo deslocado, transmitindo uma força ao chassis durante a aceleração. A roda sueca possui as mesmas funções da roda padrão, mas fornece menor resistência a outras direções devido à existência de pequenos roletes anexados ao redor da circunferência da roda. A principal vantagem da roda sueca é sua capacidade de se movimentar cinematicamente com pouca tração ao longo de várias trajetórias possíveis, não somente para frente e para trás. Já a roda esférica não oferece restrições cinemáticas diretas no movimento devido ao fato de não haver um eixo principal de rotação, portanto, não existem restrições de rolamento ou deslizamento.

2.6.2 Geometria da roda

Ao desenvolver um mecanismo de locomoção para um robô com rodas, o projetista deve levar em consideração o arranjo e o tipo das rodas, pois, esses dois parâmetros influenciam em três características fundamentais do robô: manobrabilidade, controlabilidade e estabilidade. A manobrabilidade envolve a facilidade com que o robô se movimenta, a controlabilidade se relaciona com a capacidade do sistema responder a um comando, já estabilidade é melhorada de acordo com o posicionamento das rodas, e também aumentando o número de rodas.

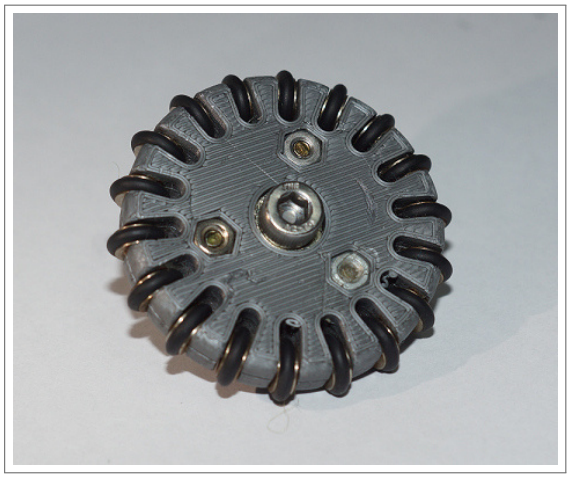
Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), robôs omnidirecionais são capazes de se mover a qualquer momento para qualquer direção ao longo do plano de atuação (x, y), independentemente da orientação do robô em relação ao eixo vertical. Robôs omnidirecionais frequentemente possuem rodas suecas, também chamadas de rodas omnidirecionais.

As rodas omnidirecionais permitem que o robô se mova em direção a um objetivo combinando movimentos de translação e rotação, fazendo com que o robô alcance o destino já com o ângulo correto. O princípio de funcionamento das rodas omnidirecionais permite que a roda deslize sem atrito na direção do eixo do motor. Para isso, são acopladas pequenas rodas nas extremidades da roda principal, chamadas de roletes. A Figura 17 mostra um exemplo de roda omnidirecional com roletes a um ângulo de 90^o.

Cada roda fornece uma tração na direção normal ao eixo do motor e paralela ao plano. Essas forças se somam e fornecem um movimento translacional e/ou rotacional para o robô. Em geral, as rodas são montadas nas extremidades do chassis e mais de duas rodas são utilizadas, tornando mais fácil cancelar qualquer torque rotacional que possa dificultar o controle do robô num percurso reto. As configurações mais populares são três e quatro rodas.

Em termos de controlabilidade, as rodas omnidirecionais tem um conjunto de roletes livres ao longo do perímetro da roda. Estes graus de liberdade causam uma acumulação do deslizamento, tendendo a reduzir a acurácia e aumentar a complexidade do projeto. Controlar um robô omnidirecional para uma direção específica de movimento é mais difícil e menos preciso do que controlar projetos de menor manobrabilidade.

Figura 17 – Exemplo de roda omnidirecional com roletes.



Fonte: o autor.

Robôs omnidirecionais são holonômicos, ou seja, são capazes de se mover em qualquer direção (x, y, θ) a qualquer momento sem a necessidade de girar no seu próprio eixo. O arranjo de quatro rodas da roda sueca não é mínima em termos de controle, pois, há apenas três graus de liberdade no plano. Entretanto, alguns robôs são projetados com quatro rodas devido à necessidade de maiores velocidades e de estabilidade.

2.6.3 Modelos de cinemática direta

Considerando um robô com quatro rodas, seja um ponto P no centro entre as rodas de um robô, cada roda deve estar a uma distância l do ponto P. Dado o raio de cada roda (r), o ângulo entre o *frame* global e local (θ), e a velocidade de cada roda, $\dot{\phi}_1$, $\dot{\phi}_2$, $\dot{\phi}_3$ e $\dot{\phi}_4$, o modelo da cinemática direta que irá prever a velocidade média do robô no *frame* de referência global é descrito como uma função de cada um dos parâmetros apresentados, como mostra a

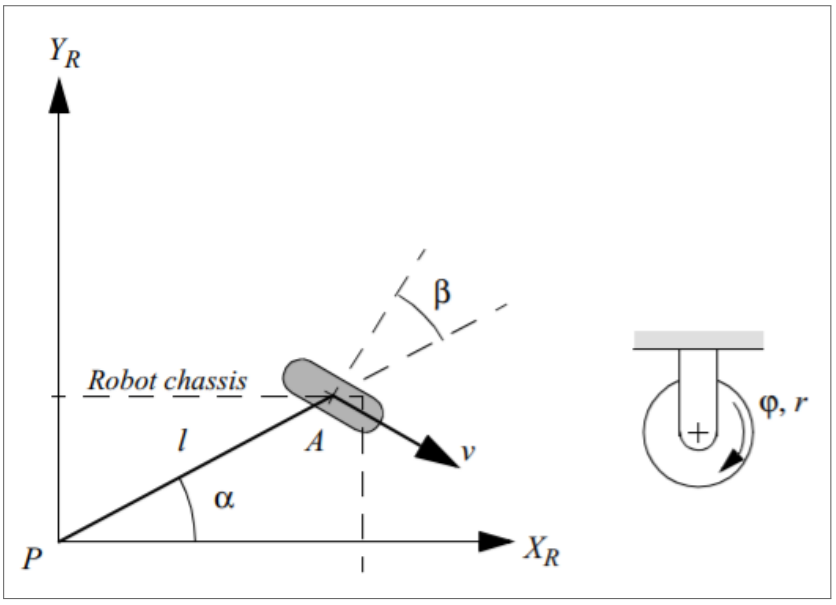
Equação 2.12 a seguir:

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3, \dot{\phi}_4) \tag{2.12}$$

O primeiro passo para criar um modelo cinemático do robô é expressar restrições do movimento de cada roda individualmente. Entretanto, serão adotados alguns critérios para simplificar este modelo. Assume-se que o plano da roda irá permanecer sempre na vertical, e que em todos os casos, há somente um ponto de contato entre a roda e o plano. Além disso, assume-se também que não há deslizamento nesse ponto de contato, ou seja, o movimento ocorre sobre condições de rolamento puro.

A Figura 18 ilustra uma roda padrão fixa A e indica sua posição em relação ao robô expressa em coordenadas polares. De acordo com Siegwart, Nourbakhsh e Scaramuzza (2011), a restrição de rolamento para essa roda garante que todo o movimento ao longo da direção do plano deve ser acompanhado de uma quantidade de movimento da roda, então, existe uma rotação pura no ponto de contato dada pela equação 2.13 a seguir:

Figura 18 – Roda fixa padrão em coordenadas polares.



Fonte: (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011)

$$\begin{bmatrix} sin(\alpha+\beta) & -cos(\alpha+\beta) & (-l)cos(\beta) \end{bmatrix} R(\theta)\dot{\xi}_I - r(\dot{\phi}) = 0 \tag{2.13}$$

As rodas omnidirecionais não possuem eixo de rotação vertical, entretanto são capazes de se mover para várias direções. Isso é possível adicionando um grau de liberdade na roda padrão fixa. As rodas suecas consistem numa roda padrão com roletes anexados ao perímetro

da roda. O ângulo γ entre o eixo do rolete e o eixo principal pode variar. Com isso, a restrição do movimento da equação 2.13 modificada é dada pela relação a seguir:

$$\begin{bmatrix} sin(\alpha+\beta+\gamma) & -cos(\alpha+\beta+\gamma) & (-l)cos(\beta+\gamma) \end{bmatrix} R(\theta)\dot{\xi}_I - r\dot{\phi}cos\gamma = 0 \qquad (2.14)$$

Considerando $\gamma = 0$, que representa a roda sueca de 90^o mostrada na Figura 17 e utilizada neste projeto, nesse caso, a velocidade é exatamente a mesma de uma roda padrão fixa. Mas, devido aos roletes, não há restrição de deslizamento ortogonal ao plano da roda.

2.6.4 Restrições cinemáticas do robô

Dado um robô móvel com M rodas, as restrições cinemáticas do chassis do robô podem ser computadas. A ideia chave é que cada roda imponha zero ou mais restrições no movimento do robô, e depois deve-se combinar todas as restrições vindas de todas as rodas no chassis. As equações da seção 2.6.3 não impõem restrições cinemáticas no chassis do robô. Apenas rodas padrão fixas tem impacto e móveis tem impacto na cinemática do chassis do robô.

Considerando que o robô tem um total de N rodas padrão fixas, a matriz que agrega os valores das velocidades de cada uma das N rodas é denominada usado $\dot{\phi}$. As restrições de rolamento de todas as rodas podem ser reunidas em uma única expressão:

$$J_{1f} \cdot R(\theta)\dot{\xi}_I - J_2\dot{\phi} = 0 \qquad (2.15)$$

Esta expressão tem grande semelhança com a restrição de rolamento de uma única roda, mas, utiliza matrizes ao invés de valores únicos, levando em conta a influência de todas as rodas. J_{1f} denota a matriz com projeções de cada uma das rodas padrão fixas com seus movimentos ao longo de seus planos de rodas individuais representados pela equação 2.12 e tem tamanho $(N \times 3)$ (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Já J_2 é uma matriz identidade NxN cujas entradas são os raios r de todas as rodas padrão, para um robô de N rodas. Por exemplo, para um robô com três rodas padrão fixas de raio r, a matriz J_2 é dada pela equação 2.16 a seguir:

$$J_2 = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix} \qquad (2.16)$$

Com isso, é possível descrever o comportamento cinemático de robôs com N rodas padrão fixas, fazer uma análise do seu movimento de acordo com o raio das rodas, disposição das rodas no robô, e distância das rodas para o eixo, e projetar um controle de velocidades para esse robô em relação a um *frame* de referência global.

3 TRABALHOS RELACIONADOS

Com o aumento da utilização de motores *brushless* devido, principalmente a sua maior eficiência e durabilidade, o número de trabalhos relacionados à área tem aumentado. Uma pesquisa através da ferramenta *Google Scholar* mostra que entre os anos de 2015 e 2019, aproximadamente 1.970 trabalhos mencionando controle de motores *brushless* DC no título foram publicados. Este capítulo irá apresentar alguns trabalhos do estado-da-arte que foram analisados mais detalhadamente durante a pesquisa e desenvolvimento desta dissertação. Ao final, será apresentado um quadro comparativo entre as técnicas apresentadas e como a abordagem proposta irá contribuir com o avanço das pesquisas na área.

3.1 EMBEDDED SYSTEM FOR MOTION CONTROL OF AN OMNIDIRECTIONAL MOBILE ROBOT

O trabalho de Mamun, Nasir e Khayyat (2018) teve como objetivo projetar e implementar um sistema embarcado para uma aplicação em um robô omnidirecional que pode ser utilizada na competição *RoboCup Small Size League* (SSL), garantindo um controle de movimento com acurácia e planejamento de percurso utilizando controladores de baixo nível e de alto nível. O controle de baixo nível é responsável pelo controle da velocidade das rodas, enquanto que o controle de alto nível controla a posição do robô.

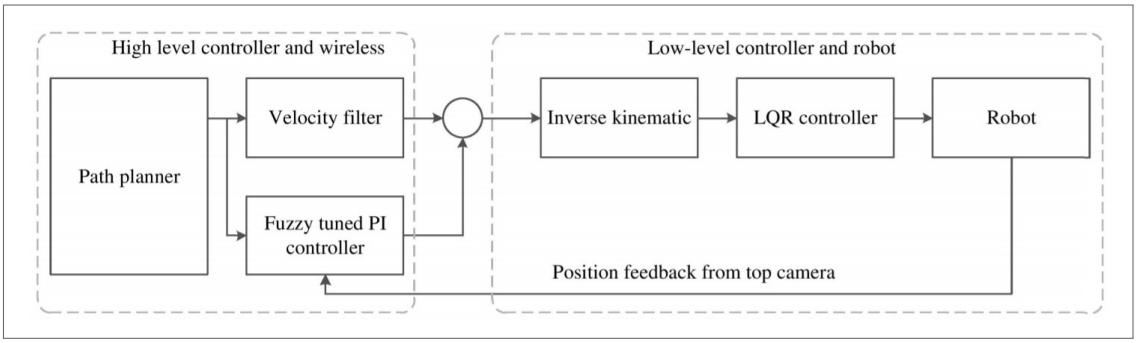
A proposta de Mamun, Nasir e Khayyat (2018) consiste em implementar um controlador de baixo nível utilizando a técnica *Linear Quadractic Regulator* (LQR), enquanto que para o controle de alto nível foi utilizado um controle adaptativo de lógica *fuzzy* juntamente com um controlador PI. O trabalho foi dividido em duas seções: projeto do sistema embarcado, e controle de movimento omnidirecional.

No projeto do sistema embarcado foi dada uma visão geral sobre a arquitetura de hardware, arquitetura de software e a comunicação do robô. Na parte de hardware foram descritos o microcontrolador, os sensores, motores e módulo de comunicação utilizados. Com respeito à arquitetura de software, Mamun, Nasir e Khayyat (2018) destaca que a arquitetura foi implementada num microcontrolador Arduino Due, e o *firmware* consiste em módulos de controle de motor, controle de bateria, controle sem fio, controle de sensores, e controle central. A comunicação do robô é feita através de um módulo e protocolo de comunicação ZigBee.

Já no controle de movimento omnidirecional, o autor descreve a modelagem cinemática e dinâmica do robô, além de projetar os controladores de baixo nível e de alto nível, destacando as contribuições de sua abordagem. Uma visão geral do sistema de controle proposto pode ser visualizado na Figura 19. De acordo com o autor, uma trajetória é definida a partir do planejamento de percurso, que irá gerar posições desejadas de maneira contínua. O filtro de velocidade verifica as velocidades rotacional e translacional, evitando ultrapassar os limites

para não danificar o circuito. O controlador fuzzy-PI adaptativo estima a posição alvo baseado na posição atual do robô, buscando minimizar o erro de posição. Já cinemática inversa deriva a velocidade do robô a partir da velocidade global. A saída do controlador LQR, segundo o autor, é um conjunto ótimo de tensões de entrada para os motores, baseado num modelo que envolve parâmetros como a massa do robô, inércia do motor, entre outros.

Figura 19 – Diagrama de blocos esquemático do controlador proposto.



Fonte: (MAMUN; NASIR; KHAYYAT, 2018)

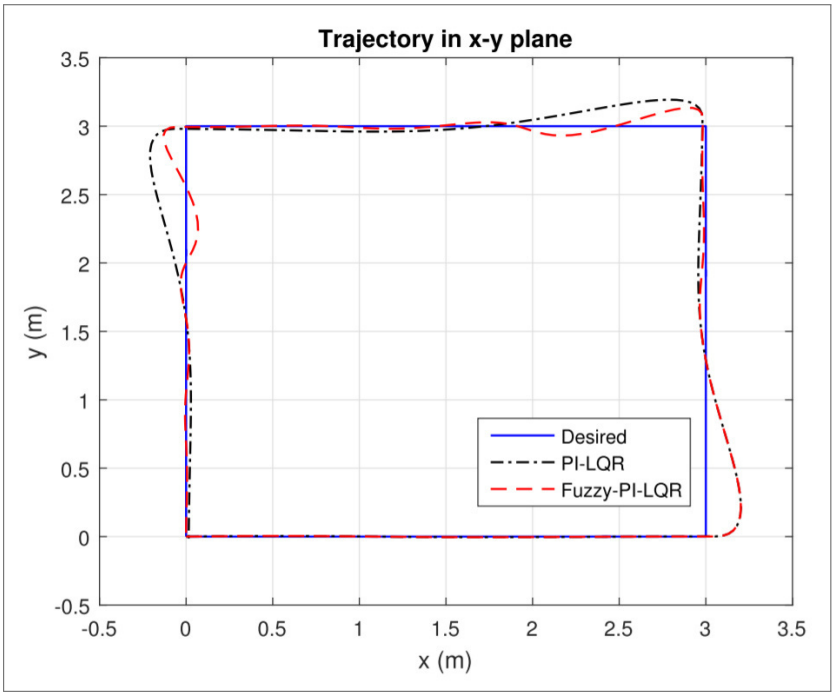
O autor utilizou a ferramenta *Simulink* do MATLAB para simular o funcionamento do robô omnidirecional com a implementação dos algoritmos de controle e apresentou alguns resultados para uma trajetória quadricular de $3m$ de lado, para velocidade de translado do robô de $3m/s$, como mostra a Figura 20. O autor fez testes com e sem a inclusão de lógica *fuzzy* e comparou os resultados com as simulações feitas pelo autor Hashemi, Jadidi e Jadidi (2011), que implementou um controlador de baixo nível utilizando uma técnica *Linear Quadractic Tracking* (LQT), mas também utilizou um controlador fuzzy-PI adaptativo no alto nível. A tabela 3 mostra um comparativo entre as duas propostas em termos de percentual de erro com relação à trajetória desejada da Figura 20.

Tabela 3 – Comparação do erro entre abordagens.

Abordagem	**Tipo de Controlador**	Erro eixo X (%)	Erro eixo Y (%)
(HASHEMI et al., 2018)	PI-LQT	7,1	14,2
(HASHEMI et al., 2018)	PI-Fuzzy-LQT	5,8	11,4
(MAMUN et al., 2018)	PI-LQT	3,4	9,0
(MAMUN et al., 2018)	PI-Fuzzy-LQT	1,2	6,6

Fonte: (MAMUN et al., 2018).

Figura 20 – Acurácia de um percurso quadrangular para o robô omnidirecional com velocidade de $3\ m/s$.



Fonte: (MAMUN; NASIR; KHAYYAT, 2018).

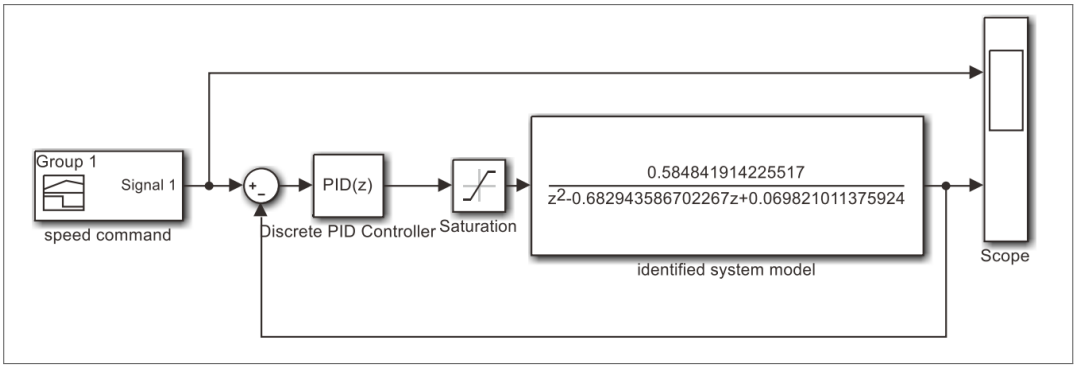
3.2 DC MOTOR SPEED CONTROL BASED ON SYSTEM IDENTIFICATION AND PID AUTO TUNING

O trabalho de Tang, Liu e Wang (2017) teve como objetivo implementar o controle de velocidade de um motor DC sem o conhecimento de parâmetros específicos do motor. Para isso, um modelo aproximado do sistema de controle foi obtido através da identificação do sistema, relacionando os sinais de saída com os sinais de entrada. Um controlador PID foi projetado e os parâmetros foram obtidos através de um *tuning* automático. Para validação da abordagem, foi utilizado um robô diferencial controlado por um arduino, e um experimento *Hardware-in-the-Loop* (HIL) foi executado com o auxílio do software MATLAB.

Para a identificação do modelo do motor, foram gerados sinais de entrada variados, sem seguir nenhuma abordagem específica. O autor utilizou o teorema de amostragem de Shannon para determinar a frequência de amostragem do sistema, e os dados foram importados para a ferramenta de identificação de sistemas do MATLAB. O resultado foi um modelo de segunda ordem e sem zeros, com acurácia de $68,71\%$. O autor não justifica a escolha da ordem do modelo discreto. O sistema de controle baseado num controlador PID foi implementado usando ferramenta *Simulink* do MATLAB e é mostrado na Figura 21. A figura apresenta um bloco de saturação que garante que o resultado do controlador não ultrapasse os limites físicos do robô, e os dados de entrada e de saída são coletados. O autor utilizou o modelo aproximado para fazer um *tuning* automático e obter constantes ótimas do controlador PID. Embora não

tenha mencionado quais foram os requisitos de entrada para o *tuning*, o autor apresenta os valores de constantes $K_p = 0,730$, $K_i = 1,247$ e $K_d = 0,008$.

Figura 21 – Controle PID discreto no sistema identificado.



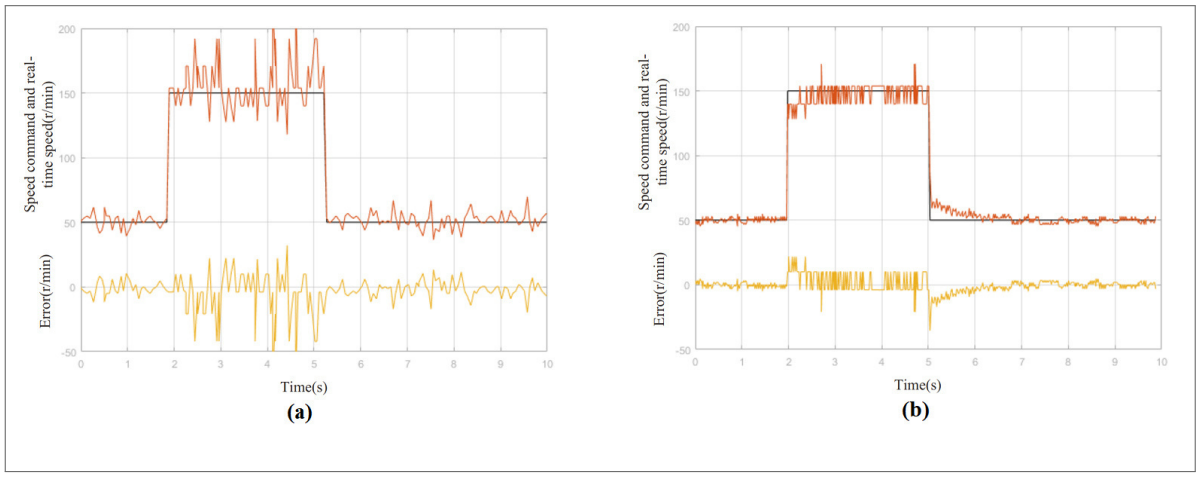
Fonte: (TANG; LIU; WANG, 2017).

Um experimento HIL (Figura 22) foi executado por Tang, Liu e Wang (2017) para a validação do controlador com as constantes obtidas pelo *tuning* automático, conectando o arduino ao software através de uma interface de comunicação serial com um cabo USB. Na Figura 22(a) foi utilizado um valor aleatório de constantes de controle, variando a velocidade de 50 Rotações Por Minuto (RPM) para 150 RPM em determinado momento no experimento, depois retornando para a velocidade de origem, e o gráfico mostra a velocidade de referência a ser seguida, a velocidade real do motor e também o erro residual entre as velocidades. O erro residual nada mais é do que a diferença entre o valor de referência e o valor atual medido pelo sensor, e é uma grandeza bastante utilizada para medir a variabilidade do sinal medido. Já a Figura 22(b) mostra o mesmo resultado utilizando as constantes de controle obtidas pelo *tuning* automático mencionado mencionado no parágrafo anterior, apresentando um erro residual bem menor quando comparado ao resultado aleatório.

A abordagem proposta por Tang, Liu e Wang (2017) mostra que é possível obter um modelo aproximado do sistema de controle através de experimentos, sem a necessidade de extrair o modelo matemático do motor. O autor destaca a facilidade de obtenção dos parâmetros PID de maneira automática através do MATLAB, e mostra uma melhora significativa no desempenho utilizando as constantes obtidas automaticamente em relação às constantes escolhidas aleatoriamente. O autor também destaca um aumento na incerteza do *encoder* quando a velocidade aumenta, e considera o uso de filtros para diminuir essa medida em trabalhos futuros.

Embora o autor tenha validado a abordagem num robô físico, algumas considerações podem ser feitas acerca do experimento realizado. No procedimento de identificação do modelo, é necessário aplicar uma entrada que seja capaz de eliminar ruídos e fatores que possam enviesar os resultados, por isso, na seção 2.3.3 foram discutidos tipos de entrada para esse tipo de identificação, o autor parece não ter levado em consideração esses pontos ao aplicar sinais de entrada arbitrários como valor de referência. É importante ressaltar, também, que

Figura 22 – Resultado do experimento HIL com constantes (a) $K_p = 5$, $K_i = 5$, $K_d = 0$ e (b) $K_p = 0,730$, $K_i = 1,247$ e $K_d = 0,008$.



Fonte: (TANG; LIU; WANG, 2017).

a utilização de comunicação serial através de um cabo USB limita o espaço de atuação do robô. Embora o autor não tenha deixado claro as circunstâncias nas quais o experimento HIL foi executado, o ideal seria que o robô pudesse se movimentar livremente no local de atuação sem a influência de componentes externos e que a comunicação seja feita sem fio, pois, essa é a situação na qual o robô irá executar sua função. A respeito do *tuning* automático das constantes o autor não estabelece critérios de projeto para a escolha das constantes ótimas como *overshoot*, tempo de subida máximo, ou tempo de assentamento máximo. Sendo assim, as observações feitas sobre o trabalho de Tang, Liu e Wang (2017) levam ao desenvolvimento de melhorias na abordagem que será proposta neste trabalho.

3.3 PID CONTROL OF BRUSHLESS DC MOTOR AND ROBOT TRAJECTORY PLANNING SIMULATION WITH MATLAB/SIMULINK

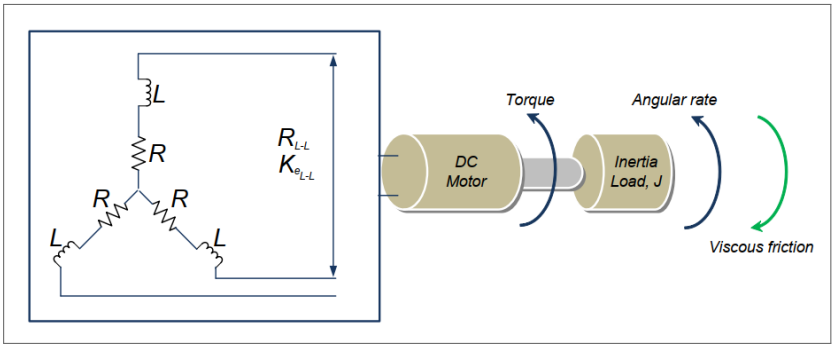
O trabalho descrito em Oguntoyinbo et al. (2009) apresenta uma abordagem para controle de motores *brushless* DC utilizando um controlador PID com o auxílio do software MATLAB. O autor deriva um modelo matemático de um motor *brushless* DC a partir de um diagrama esquemático mostrado na Figura 23. A partir da lei de Kirchoff é possível deduzir uma equação diferencial que descreve o comportamento do motor. Aplicando a transformada de laplace e fazendo uma série de simplificações, é possível chegar a uma função de transferência para um motor *brushless* DC dado pela equação 3.1 a seguir:

$$G(s) = \frac{\frac{1}{K_e}}{\tau_m \cdot \tau_e \cdot s^2 + \tau_m \cdot s + 1} \tag{3.1}$$

Onde:

- K_e é a constante eletromotriz de movimento;

Figura 23 – Diagrama esquemático de um motor *brushless* DC.



Fonte: (OGUNTOYINBO et al., 2009).

- τ_m é a constante mecânica de tempo do motor;
- τ_e é a constante elétrica do motor.

Dados R e L, a resistência e a indutância de cada fase do motor (informações fornecidas pelo fabricante através do *datasheet* do motor), respectivamente, a constante elétrica do motor pode ser dada pela equação 3.2 a seguir:

$$\tau_e = \frac{L}{3R} \tag{3.2}$$

Dados o momento de inércia do rotor (J_{rotor}), a constante de torque (K_t) e a constante mecânica de tempo (τ_m) fornecidos pelo fabricante do motor, a constante eletromotriz de movimento (K_e) pode ser obtida através da equação 3.3 a seguir:

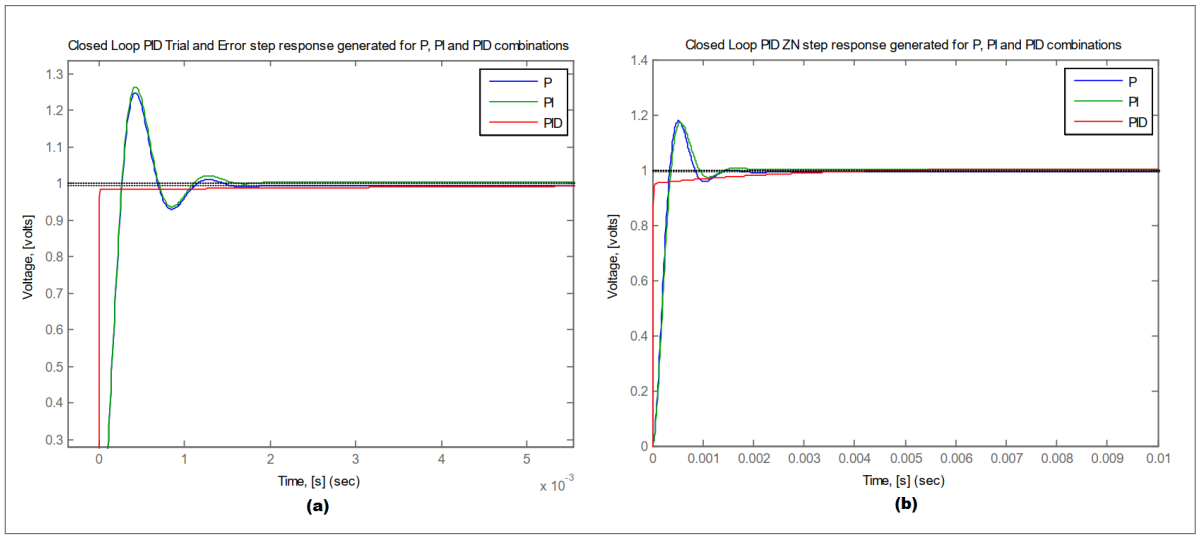
$$\tau_m = \frac{3 \cdot R \cdot J}{K_e \cdot K_t} \tag{3.3}$$

O autor utiliza como referência o *datasheet* de um motor Maxon EC-45 flat de $30W$ para obter as constantes e derivar uma função de transferência a partir da equação 3.1.

Para o projeto de um controlador PID para o motor, o autor implementa duas técnicas de controladores: uma baseada no método de tentativas e erros, e a técnica de Ziegler-Nichols para encontrar parâmetros de controle ótimos e faz uma comparação entre ambas. Para a técnica de tentativas e erros o autor parte do critério de estabilidade de Routh-Hurwitz, que consiste em encontrar um parâmetro chamado de ganho crítico que irá auxiliar na escolha dos valores iniciais das constantes de controle. Já o método de Ziegler-Nichols, como explicado na seção 2.4.1.1, consiste em analisar uma resposta em degrau, e ao traçar uma reta tangente ao ponto de inflexão dessa resposta, podem ser extraídos dois parâmetros (constante de atraso e constante de tempo), que irão auxiliar na obtenção das constantes de controle. A Figura 24 (a) ilustra os resultados de uma simulação da resposta do modelo teórico do motor obtido utilizando o software MATLAB com a implementação das constantes de controle utilizando o método de tentativas e erros. Já a Figura 24 (b) mostra o resultado de uma simulação da

resposta do modelo do motor com a implementação das constantes ótimas obtidas através da técnica de Ziegler-Nichols.

Figura 24 – Resposta de um controlador PID com constantes obtidas através do (a) método de tentativas e erros e (b) do método de Ziegler-Nichols.



Fonte: (OGUNTOYINBO et al., 2009).

Como o autor realizou o projeto utilizando apenas uma modelagem em simulação de software, foi implementado um controlador PID contínuo sem levar em consideração a discretização para a implementação em hardware que posteriormente deverá ser feita, onde o tempo de amostragem tem papel fundamental. Em ambas as simulações da Figura 24, o tempo de subida é muito pequeno, o que faz com que a amostragem tenha que ser feita numa taxa muito baixa para conseguir detectar a resposta transiente do sistema, e quanto mais baixa a taxa de amostragem, maior a incerteza associada ao sistema, e maiores os ruídos do sensor de realimentação. Uma das desvantagens da técnica de Ziegler-Nichols é justamente o fato do método considerar a resposta transiente, mas não o tempo de amostragem do sistema, portanto, nem sempre as constantes encontradas pelo método, quando implementadas num controlador baseado em tempo discreto no mundo real, funcionam da maneira esperada. Portanto, deve-se procurar uma implementação baseada no tempo discreto ao projetar um controlador real, a fim de analisar a viabilidade das constantes de controle.

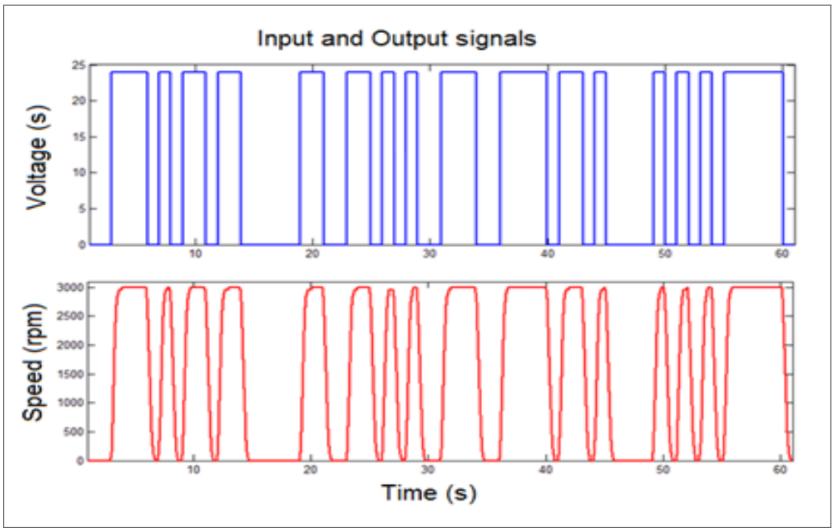
3.4 MODEL BASED DESIGN OF PID CONTROLLER FOR BLDC MOTOR WITH IMPLEMENTATION OF EMBEDDED ARDUINO MEGA CONTROLLER

O trabalho de Hat et al. (2015) apresenta a implementação de uma estratégia de controle de motores *brushless* DC através do Projeto Baseado em Modelos, ou *Model-Based Design* (MBD) com implementação num microcontrolador arduino. O modelo do motor é obtido através da metodologia *black-box* apresentada na seção 2.3.2.5 com o auxílio da ferramenta *System Identification* do MATLAB. Além disso, Hat et al. (2015) projeta um controlador

PID com tuning dos parâmetros para aplicação em tempo real. O fluxo de trabalho proposto pelo autor segue os seguintes passos: aquisição de dados, identificação do modelo da planta, projeto e simulação dos controladores com realimentação e implementação dos controladores num microprocessador embarcado para testes em tempo real.

A aquisição dos dados para identificação do modelo é feita através de um microcontrolador Arduino Mega embarcado com uma interface de comunicação serial RS232 entre o Arduino e o computador. O trabalho de Hat et al. (2015) simula um sinal PRBS como entrada de tensão na forma de PWM, sem detalhar os parâmetros utilizados para a geração da sequência pseudoaleatória, e utiliza uma taxa de amostragem do *encoder* de $0,01s$. A Figura 25 mostra os sinais de entrada e saída capturados para obtenção do modelo.

Figura 25 – Sinal PRBS de entrada e velocidade do motor capturados pelo MATLAB.



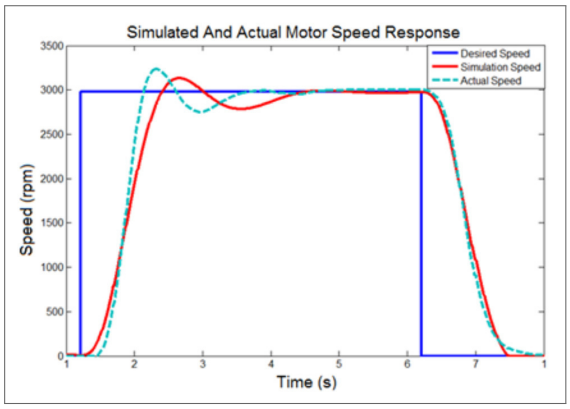
Fonte: (HAT et al., 2015).

É possível notar que o autor variou o sinal entre 0 e $24V$, ou seja, entre 0 e 100% do *duty cycle* do sinal PWM e não entre dois valores dentro da faixa de operação, e até 10% do duty cycle como recomendado pela literatura para identificação de sistemas, isso faz com que o sistema não elimine variabilidades e identifique não linearidades inerentes ao processo, principalmente durante a saída da inércia do motor. Em seguida, o autor importou os sinais para a ferramenta *System Identification* do MATLAB e diz ter encontrado uma função de transferência em tempo contínuo com ajuste de $91,44\%$. A função obtida pela técnica *black-box* é utilizada para o projeto de um controlador PID convencional, e a ferramenta Simulink do MATLAB é utilizada para a análise e tuning do sistema de controle obtido. Segundo Hat et al. (2015), o fato do tempo de subida da planta com o controlador PID implementado ser maior ou igual ao tempo de subida da resposta ao degrau do motor real durante a aquisição dos dados é um sinal de que o controlador irá funcionar de maneira adequada para qualquer valor de entrada.

A Figura 26 mostra os resultados do controlador implementado na simulação e o con-

trolador real, que mostra diferenças entre os resultados estimados e o medido na prática.

Figura 26 – Comparação entre a resposta do controlador real e a simulada.



Fonte: (HAT et al., 2015).

A Tabela 4 mostra os parâmetros obtidos pelo autor para uma resposta em degrau unitário do motor real, uma resposta em degrau com controlador PID simulado, e a resposta em degrau com o controlador PID no experimento real.

Tabela 4 – Parâmetros de resposta do motor em variadas situações.

Parâmetro	Modelo do Motor	PID (Simulado)	PID (Real)
Tempo de Subida	$0,535s$	$0,715s$	$0,894$
Tempo de Assentamento	$1,52s$	$2,99s$	$2,49$
Overshoot	$4,87\%$	$5,07\%$	$4,35\%$
Valor em Regime Permanente	$1,05$	$0,992$	$1,01$

Fonte: (HAT et al., 2015).

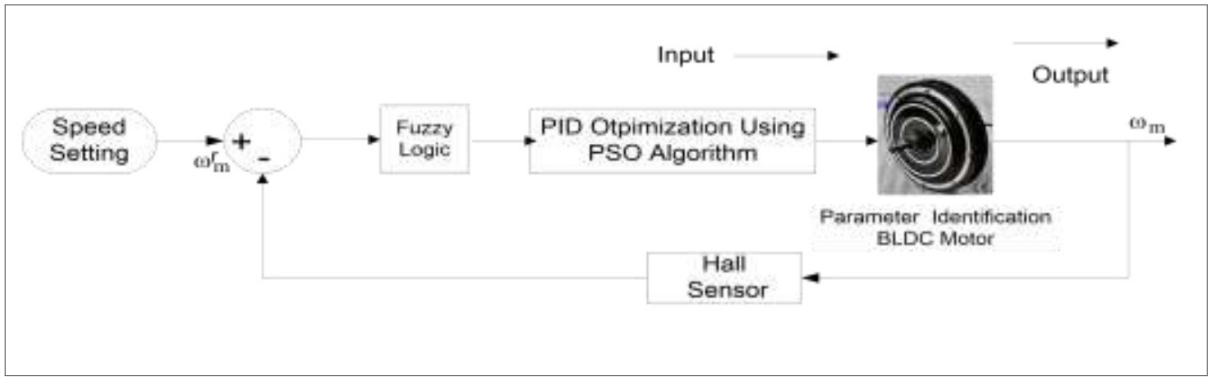
É importante ressaltar que, o autor utiliza uma planta e controlador contínuos, deixando para o software MATLAB interpretar e fazer as abstrações necessárias para o correto funcionamento do sistema de controle dentro da plataforma, sendo necessária a discretização para implementação em microcontroladores fora do ambiente do software utilizado.

3.5 SYSTEM IDENTIFICATION OF BLDC MOTOR AND OPTIMIZATION SPEED CONTROL USING ARTIFICIAL INTELLIGENT

O trabalho de Anshory et al. (2019) é baseado em um método de simulação de entradas utilizando o método Autorregressivo com Entrada Externa (ARX) através do software

MATLAB para identificação de uma função de transferência de motores *brushless* DC. Após a obtenção do modelo, é feita uma análise comparativa de métodos de otimização de parâmetros, são eles: PID, lógica *Fuzzy*, e uma combinação do algoritmo PID e lógica *Fuzzy*, e Otimização Por Enxame de Partículas, ou *Particle Swarm Optimization* (PSO) junto com lógica *Fuzzy* e PID. Uma visão geral da abordagem proposta é mostrada na Figura 27. Uma vez obtido um modelo que melhor representa o funcionamento do motor *brushless* DC, o valor da velocidade de referência é comparado com a velocidade real do motor, e o erro alimenta um controlador com lógica *fuzzy*, que por sua vez envia comandos para um controlador PID otimizado por um algoritmo PSO.

Figura 27 – Visão geral da abordagem proposta com lógica *fuzzy* e PID otimizado por um algoritmo PSO.



Fonte: (ANSHORY et al., 2019).

O autor simulou o funcionamento de um motor *brushless* DC através da ferramenta *Simulink*, e implementou o método ARX através da ferramenta de identificação de sistemas do MATLAB para obter uma função de transferência em tempo discreto que relaciona o sinal de entrada com o sinal de saída do motor simulado. Como o trabalho abordou apenas resultados no ambiente de simulação, o autor converteu a função discreta para a forma contínua, obtendo a função de transferência dada pela equação 3.4 a seguir:

$$H_{arx} = \frac{-3,473s^2 + 27,11s + 0,7475}{s^3 + 3,24s^2 + 12,55s + 0,4053} \tag{3.4}$$

O controlador de lógica *fuzzy* tem dois parâmetros de entrada, o erro e a variação do erro, e a saída é baseada num conjunto de regras que relacionam esses dois parâmetros de entrada. O autor aponta como uma das fraquezas do controlador PID o fato das constantes não se adaptarem de maneira automática ao ambiente em que ele está inserido, por isso, sugere a utilização de um algoritmo para otimizar as constantes. Um algoritmo PSO é um método de otimização inspirado no movimento de criaturas vivas, por exemplo, pássaros. No trabalho de Anshory et al. (2019), os parâmetros utilizados para o algoritmo PSO são mostrados na Tabela 5, esses parâmetros são utilizados para o *tuning* do controlador PID.

Tabela 5 – Parâmetros do algoritmo PSO para o controlador PID (ANSHORY et al., 2019).

Parâmetro	Valor
Número de partículas	3
Máximo de iterações	3
ω	$0,9$
$c1$	2
$c2$	2

Fonte: (ANSHORY et al., 2019).

Os resultados da simulação utilizando a função de transferência contínua aproximada dada pela equação 3.4 são mostrados na Tabela 6 a seguir. É feita uma comparação entre as respostas em degrau dos controladores implementados. Foram implementados os controladores PID utilizando somente o método de tentativas e erros, somente com lógica *Fuzzy*, *Fuzzy-PID* e com a combinação *Fuzzy-PID-PSO*. No estudo feito, o algoritmo utilizando a combinação de *Fuzzy* e PID otimizado pelo algoritmo PSO obtiveram os melhores resultados.

Embora o autor tenha apresentado uma estratégia de controle que combina o controlador PID com lógica fuzzy e otimização PSO, que apresenta um melhor resultado, os resultados foram obtidos somente com simulação, sendo necessário validar essa abordagem em um controlador real para verificar a viabilidade da abordagem, tanto devido ao custo computacional, quanto a influência de distúrbios causados pelo ambiente e sensores.

Tabela 6 – Comparação entre as abordagens implementadas (ANSHORY et al., 2019).

Abordagem	Amplitude	Tempo de Subida	*Overshoot*
Open Loop	$2,1V$	$365,3ms$	$25,949\%$
PID	$0,988V$	$300,03ms$	$-0,829\%$
Fuzzy	$0,994V$	$161,46ms$	$68,644\%$
Fuzzy-PID	$1V$	$241,844ms$	$5,851\%$
Fuzzy-PID-PSO	$1V$	$6,007ms$	$-0,238\%$

Fonte: (ANSHORY et al., 2019).

3.6 OPTIMAL PID CONTROLLER OF BRUSHLESS DC MOTOR USING GENETIC ALGORITHM

O trabalho desenvolvido por Ibrahim, Mahmood e Sultan (2019) apresentou uma abordagem para encontrar os coeficientes do controlador PID baseada em um Algoritmo Genético (GA), e avaliou o resultado utilizando duas funções objetivo: *Integral Absolute Error* (IAE) e *Integral Squared Error* (ISE). A função IAE é dada pela equação 3.5 a seguir:

$$IAE = \int_0^{T_{ss}} abs(e(t))dt \tag{3.5}$$

Onde:

- T_{ss} é o tempo de execução do algoritmo;
- $e(t)$ é o erro com relação ao sinal de referência de entrada.

Já a função ISE é dada pela equação 3.6 a seguir:

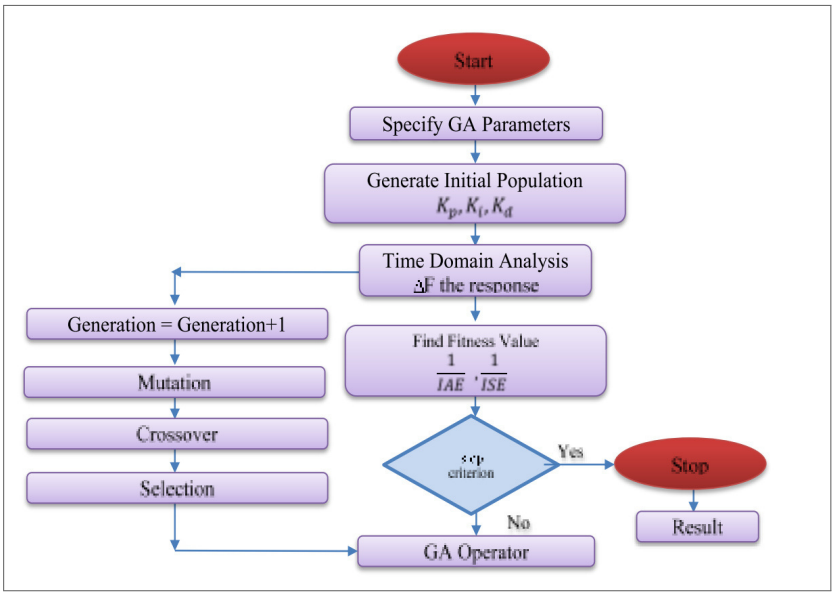
$$IsE = \int_0^{T_{ss}} e(t)^2 dt \tag{3.6}$$

O controlador foi projetado, modelado e simulado utilizando o software MATLAB. O autor utilizou um modelo do motor obtido de maneira similar ao modelo obtido por Oguntoyinbo et al. (2009), utilizando os parâmetros de acordo com o motor a ser estudado. Um controlador PID contínuo foi utilizado, e os resultados do algoritmo genético proposto foram comparados com os métodos de *tuning* baseado em tentativas e erros (considerando a experiência do projetista, segundo o autor) e também da ferramenta *PIDTuner* do MATLAB.

Um fluxograma com o procedimento de avaliação do algoritmo genético é ilustrado na Figura 28. Inicialmente, alguns parâmetros do GA devem ser especificados, como a função *fitness*, que é o indicador de desempenho do algoritmo, o tamanho da população, as probabilidades de mutação e de cruzamento, entre outros. Uma população inicial é gerada a partir de uma matriz contendo um conjunto de soluções possíveis de acordo com tamanho da população, e a cada iteração,a função *fitness* é calculada para cada conjunto de soluções. São selecionadas as soluções com melhor desempenho, e partir desses parâmetros, é criado um novo conjunto de soluções utilizando operadores genéticos (cruzamento e mutação). Os passos são repetidos até ser atingido um critério de parada. O critério de parada pode ser ou um número de iterações, ou um valor da função *fitness* preestabelecido. A Tabela 7 mostra os parâmetros do GA utilizados pelo autor. Esses parâmetros foram utilizados para a implementação do algoritmo no software MATLAB.

A Tabela 8 mostra uma comparação entre as características de resposta das funções *fitness* da abordagem proposta utilizando os critérios de parada ISE e IAE, e os métodos de tentativas e erros, e o resultado da ferramenta *PIDTuner* do MATLAB. São analisados o tempo de subida, tempo de assentamento, *overshoot*, e erro em regime permanente. Percebe-se que os parâmetros de tempo de subida e de assentamento são muito pequenos, pois, o

Figura 28 – Fluxograma de projeto do algoritmo gwenético proposto.



Fonte: (IBRAHIM; MAHMOOD; SULTAN, 2019).

autor considera um controle contínuo. Numa implementação real e discreta, deve-se adicionar a influência do tempo de amostragem do sensor utilizado, consequentemente os parâmetros deverão se adaptar a essa realidade.

Tabela 7 – Parâmetros do algoritmo genético utilizado.

Função Fitness	$1/IAE$ & $1/ISE$
Método de seleção	Roleta
Tamanho da geração	40
Tamanho da população	30
Escala Fitness	Proporcional
Método de cruzamento	Cruzamento aritmético
Probabilidade de cruzamento	90%
Probabilidade de mutação	2%
Método de mutação	Mutação uniforme
Método de cruzamento	Cruzamento aritmético

Fonte: (IBRAHIM; MAHMOOD; SULTAN, 2019).

Tabela 8 – Parâmetros de resposta do motor em variadas situações.

Parâmetro	Tentativas e Erros	*PIDtuner*	GA-IAE	GA-ISE
Tempo de Subida(s)	$6,87e-5$	$5,4e-5$	$5,11e-7$	$4,56e-8$
Tempo de Assentamento(s)	$6,92e-4$	$3,75e-4$	$9,11e-7$	$8,12e-8$
Overshoot($\%$)	$37,4$	$14,5$	0	0
Erro em Regime Permanente	0	0	0	0

Fonte: (IBRAHIM; MAHMOOD; SULTAN, 2019).

3.7 ANÁLISE COMPARATIVA ENTRE OS TRABALHOS RELACIONADOS

Nesta seção será feita uma análise de características dos trabalhos relacionados para que seja feita uma comparação entre a abordagem proposta e os trabalhos vistos nesta seção. O objetivo desta análise é entender como a abordagem proposta irá contribuir para o avanço das pesquisas na área. Os trabalhos serão analisados com relação aos seguintes critérios:

- **Utilização de motores *brushless* DC.** É analisado se o autor utilizou ou não motores *brushless* DC para validar a abordagem proposta.
- **Inclusão de Técnica para Obtenção do Modelo do Motor.** É verificada a utilização de alguma técnica para extrair o modelo do motor, por exemplo, utilização de sinais de entrada PRBS, ou aplicação de uma entrada em degrau unitário para análise da resposta do motor.
- **Obtenção do Modelo do Motor de Maneira Experimental.** É avaliado se o autor obtém o modelo do motor utilizando testes em um motor real, ou através de uma simulação em software como o MATLAB, ou deriva o modelo através de uma equações diferenciais.
- **Modelagem em Tempo Discreto.** É feita uma análise se o autor faz a discretização do modelo do motor obtido de acordo com o tempo de amostragem do sensor utilizado para *feedback*. Esse passo é essencial para a implementação do controle de motores em microcontroladores e/ou sistemas embarcados.
- **Implementação de um Controlador P, PI, PD, ou PID**. Verifica-se qual o controlador utilizado pelo autor. Se é um controlador PID, ou uma variação do mesmo, ou se é um controle alternativo, como por exemplo, LQR ou LQT.

- **Reajuste Automático das Constantes de Controle.** É verificado se o autor implementa algum tipo de algoritmo que atualize as constantes de controle automaticamente, sem a necessidade de modificação do código-fonte.
- **Validação em Ambiente Real.** É analisado se o autor validou a abordagem proposta em um motor real, ou apenas em um software de simulação, como por exemplo, o MATLAB.
- **Comunicação sem fio entre Hardware e Computador.** É avaliado o tipo de comunicação entre o hardware utilizado (microcontrolador) e o computador para aqueles que validam o motor em um ambiente real. Para o caso da validação dos motores em robôs móveis, é importante a comunicação sem fio para que o robô possa se movimentar livremente no ambiente real.

A Tabela 9 apresenta as características listadas nesta seção de maneira resumida para cada uma das abordagens analisadas durante o capítulo e também da abordagem proposta, que será detalhada no Capítulo 4.

Tabela 9 – Comparativo entre os trabalhos relacionados

	Suporta Motores Brushless DC	Apresenta Técnica para Obtenção do Modelo do Motor	Obtenção do Modelo do Motor de Maneira Experimental	Modelagem em Tempo Discreto	Implementação de um Controlador P, PI, PD ou PID	Reajuste Automático das Constantes de Controle	Validação em Ambiente Real	Comunicação sem fio entre Hardware e Computador
(Oguntoyinbo, 2009)	Sim	Sim	Não	Não	Sim	Não	Não	Não
(Hat et al., 2015)	Sim	Sim	Sim	Não	Sim	Não	Sim	Não
(Tang et al., 2017)	Não	Não	Sim	Sim	Sim	Não	Sim	Não
(Mamun et al., 2018)	Sim	Não	Não	Sim	Não	Não	Não	Não
(Anshory et al., 2019)	Sim	Não	Não	Sim	Sim	Não	Não	Não
(Ibrahim et al., 2019)	Sim	Sim	Não	Não	Sim	Sim	Não	Não

Fonte: o autor.

4 ABORDAGEM PROPOSTA PARA TUNING DO CONTROLADOR PI PARA MOTORES BRUSHLESS DC

Este capítulo irá apresentar em detalhes a abordagem proposta para identificação do modelo de motores *brushless* DC, e de uma estratégia para implementação e sintonização (*tuning*) de um controlador PID para esse tipo de motor. O capítulo apresenta, inicialmente, uma descrição geral da abordagem e de cada uma de suas etapas. A seguir, cada etapa será detalhada em suas respectivas seções levando em consideração aspectos de implementação para a realização dos experimentos em ambiente real. Embora para este trabalho seja utilizado como estudo de caso uma aplicação de robôs projetados para a competição *RoboCup Small Size League*, é importante ressaltar que a abordagem proposta pode ser generalizada para outros motores *brushless* DC com *encoder* e um *driver* que recebe como entrada um sinal de PWM e realiza a comutação do motor de maneira automática. Para isso, devem ser feitas as devidas adaptações em termos de amostragem e identificação da faixa de operação do motor.

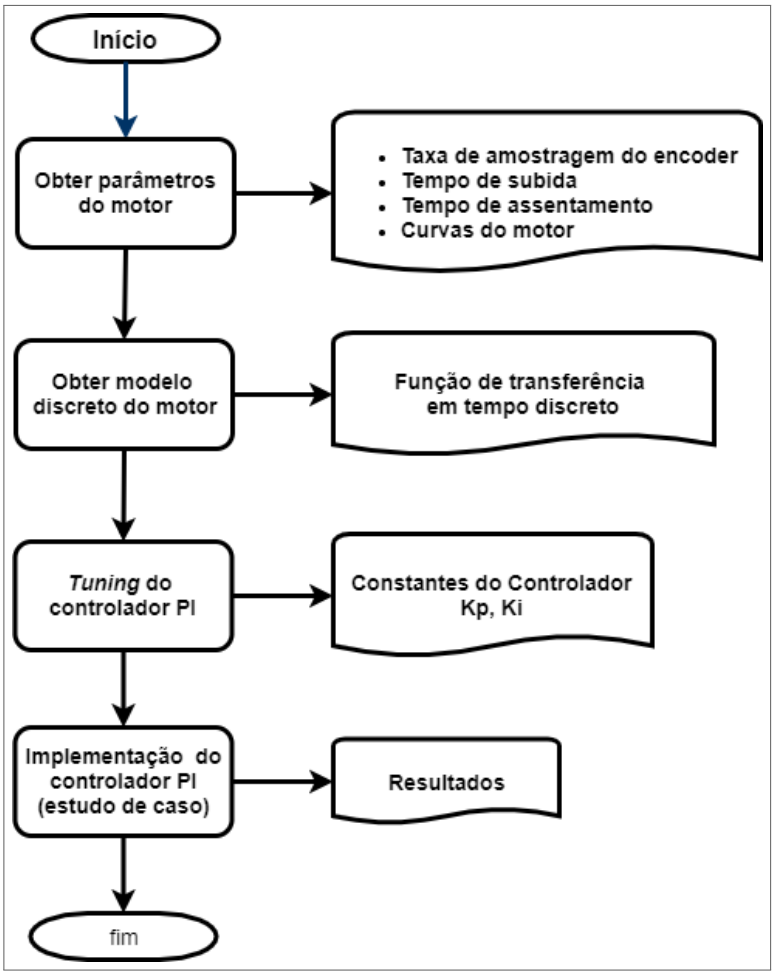
4.1 VISÃO GERAL DA ABORDAGEM PROPOSTA

Uma visão geral da abordagem proposta pode ser visualizada na Figura 29. Primeiramente, são gerados estímulos no motor para a identificação dos parâmetros básicos como o tempo de subida e tempo de assentamento, que representam o quão rápido o sistema reage, e quão rápido o sistema se estabiliza, respectivamente. Outro parâmetro importante obtido nesta etapa, é a taxa de amostragem do *encoder*. Essa taxa amostragem deve ser grande o suficiente para detectar a resposta transiente do motor e para que seja possível identificar a influência de ruídos externos, de forma que eles não atrapalhem o funcionamento do sistema. Além disso, também será traçada uma curva de resposta aproximada com dados reais relacionando o sinal de entrada do motor e sua resposta, afim de identificar qual é a faixa de operação do motor.

A próxima etapa consiste em obter uma função de transferência em tempo discreto que melhor represente a resposta do motor no tempo. Para isso, foi elaborado um experimento que será detalhado na seção 4.4. Esse experimento consiste em aplicar um sinal pseudo-aleatório de amplitude variada dentro da faixa de operação do motor para identificar seu comportamento. Os dados do experimento são processados no software MATLAB, que retorna uma função que melhor representa o motor testado, de acordo com os dados adquiridos no experimento real.

Uma vez com um modelo aproximado do motor, pode-se projetar e fazer o *tuning* de um controlador PI para encontrar as constantes adequadas para a aplicação. Alguns aspectos como a saturação do atuador causada pela ação integral e requisitos de tempo de subida, máximo *overshoot* e tempo de assentamento serão tratados na seção 4.5. O procedimento descrito nessa seção deve ser utilizado para a identificação das caraterísticas de cada motor individualmente, porém, os experimentos podem ser realizados com múltiplos motores ao

Figura 29 – Visão geral da abordagem proposta.



Fonte: o autor.

mesmo tempo.

Como forma de validação da abordagem proposta, esta metodologia foi aplicada no estudo de caso de um robô projetado para competição *RoboCup Small Size League*, conforme descrito no capítulo 5. Detalhes da implementação do controlador discreto e do *hardware* utilizado serão apresentados e, além da implementação do controlador, a modelagem cinemática que viabilizou o controle omnidirecional do robô será descrita. Essa modelagem, em conjunto com os parâmetros do controlador PID obtidos na abordagem proposta, são utilizados e alguns resultados do experimento serão apresentados no capítulo 6.

4.2 REQUISITOS DE PROJETO

Os requisitos de projeto que são impostos aos sistemas de controle são chamados de especificações de desempenho, e são relacionados à exatidão, estabilidade e velocidade de resposta de um sistema. Uma das etapas mais importantes do projeto de sistemas de controle é estabelecer, de maneira precisa, as especificações de desempenho para que esses requisitos levem a um sistema de controle ótimo em relação ao objetivo do projeto (OGATA, 2003). Ao

analisar essas especificações de desempenho, deve-se atentar à execução da tarefa final, ou seja, focar no aspecto do sistema que é o mais importante para a tarefa final. Entretanto, há um *trade-off* a ser levado em consideração durante a escolha desses aspectos prioritários, por exemplo, priorizar a velocidade de resposta pode acarretar num maior *overshoot*.

Em geral, há um conjunto de requisitos de controle diferente para cada aplicação. Especificamente para este trabalho, os requisitos de projeto para cada motor são apresentados na Tabela 11. Esses parâmetros devem ser avaliados durante a validação do projeto final do controlador, e permitem que o robô se movimente em tempo real de maneira fluida e buscando minimizar o erro entre a velocidade atual do motor e a velocidade desejada. Para essa aplicação, o aspecto primordial é a exatidão na operação em regime permanente. Ou seja, o quão próximo da velocidade desejada o sistema consegue chegar, sendo tolerada apenas a incerteza associada ao *encoder* utilizado para medir a velocidade. Por isso, os aspectos em regime transiente são um pouco mais relaxados, mas ainda sim devem atender aos requisitos de tempo real.

Tabela 11 – Requisitos de projeto para os motores utilizados.

Parâmetro	**Valor**
Tempo de Subida Máximo	$0,3\ s$
Tempo de Assentamento Máximo	$0,5\ s$
Overshoot Máximo	5%
Erro em Regime Permanente Máximo	$0,767\ rad/s$

Fonte: o autor.

4.3 DETERMINAÇÃO DOS PARÂMETROS DO MOTOR

Além das características físicas fornecidas pelo fabricante dos motores, o conhecimento de alguns parâmetros de resposta como tempo de subida, tempo de assentamento e taxa de amostragem do *encoder* são fundamentais parâmetros que irão viabilizar a realização de alguns dos experimentos utilizados nessa abordagem. A seguir os experimentos realizados para a obtenção de cada um desses parâmetros de resposta dos motores utilizados serão detalhados, juntamente com o procedimento para identificação da curva de resposta de cada motor para que seja identificada sua faixa de operação.

4.3.1 Taxa de amostragem do encoder

A incerteza associada à leitura da velocidade do motor é determinada pela taxa de amostragem do *encoder*. O *encoder* utilizado neste trabalho é do modelo MILE projetado para os motores do tipo *EC-45 flat* da empresa Maxon Global. Os *encoders* modelo MILE são incrementais e de quadratura, e utilizam um sistema de medição de ângulo indutivo para reconhecer o sentido de movimento e gerar os sinais de saída. Este modelo conta com dois canais (A e B), e uma resolução e 1024 PPR (MAXON, 2019).

Como explicado na seção 2.2, a taxa de amostragem do *encoder* determina o tempo entre uma leitura e outra da quantidade de pulsos lidos pelo sensor óptico tanto no sentido horário, quanto no anti-horário. Logo, quanto maior a taxa de amostragem, mais rápida a velocidade do motor poderá ser atualizada. Entretanto, deve ser encontrado um compromisso (*trade-off*) entre a taxa de amostragem e o ciclo de funcionamento do algoritmo de controle, ou seja, quanto maior a taxa de amostragem, maior a incerteza associada à velocidade calculada (Δ). Isso quer dizer que a velocidade lida pelo *encoder* pode estar no intervalo $(v-\Delta, v+\Delta)$, onde a incerteza pode ser calculada substituindo a quantidade de pulsos ($Pulses$) da Equação 2.1 por 1. A tabela 12 mostra alguns valores de incerteza (Δ) para cada intervalo entre amostras (t_{sample}), utilizando o *encoder* MILE de 1024 PPR.

Tabela 12 – Relação entre tempo de amostragem e incerteza associada.

Tempo de amostragem $[T_{sample}](s)$	**Incerteza** $[\Delta](RPM)$	**Incerteza** $[\Delta](rad/s)$
$0,001$	$14,6484$	$1,534$
$0,002$	$7,3242$	$0,767$
$0,005$	$2,9297$	$0,307$
0.01	$1,4648$	$0,153$
0.02	$0,7324$	$0,077$
0.05	$0,2930$	$0,031$

Fonte: o autor.

Embora uma taxa de amostragem baixa apresente uma incerteza pequena, não é recomendado para motores com resposta muito rápida, pois informações de regime transiente do motor são perdidas, tornando difícil a obtenção de características de resposta do motor. Haidecker recomenda a utilização de uma amostragem de *encoder* de, no mínimo, de 5 a 10 vezes o tempo de subida do motor para que seja possível controlar um motor de maneira

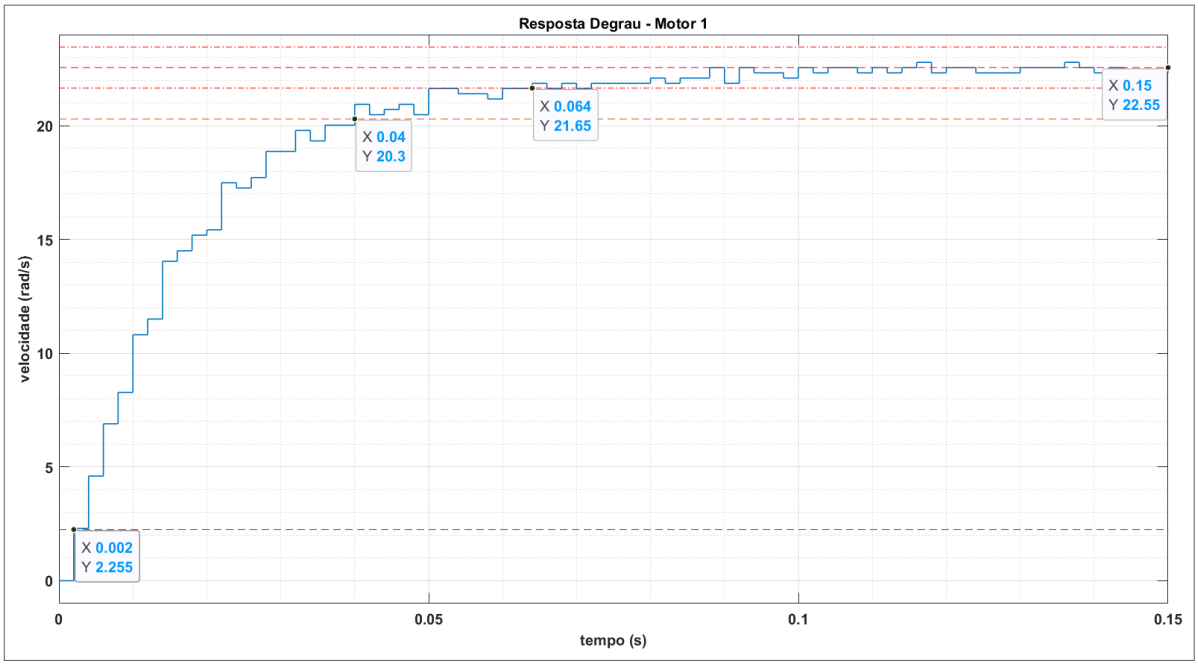
eficiente (HAIDEKKER, 2013).

4.3.2 Características de resposta dos motores

Como explicado na seção 2.3, um experimento para obter esses parâmetros é aplicar uma entrada degrau unitário aos motores, e extrair informações como tempo de subida e tempo de assentamento a partir do gráfico da resposta.

A Figura 30 apresenta a resposta a um degrau de um dos motores Maxon EC-45 flat 50W, analisado com o auxílio do software MATLAB. O teste foi aplicado em quatro motores reais do mesmo modelo, e para este caso, o estímulo foi uma entrada em degrau com 20% do *duty cycle* de PWM, com os motores partindo da inércia. O intervalo entre cada amostra do *encoder* utilizado no experimento foi de 2 milissegundos (ms), pois, a incerteza associada a essa taxa é menor que $1\ rad/s$, considerado aceitável para a aplicação. Os parâmetros obtidos com o teste são: tempo de subida (*Rise Time*) e tempo de assentamento (*Settling time*). A Tabela 13 apresenta, resumidamente, os parâmetros para os quatro motores testados, esses valores serão utilizados como base para decisão de alguns dos experimentos que serão descritos a seguir. Percebe-se que o desempenho dos motores é bem próximo, mesmo quando expostos a não-linearidades associadas ao circuito do motor e ao ambiente de testes real.

Figura 30 – Resposta à entrada degrau para um dos motores.



Fonte: o autor.

Tabela 13 – Características da resposta ao degrau dos quatro motores

Motor	$T_{subida}(s)$	$T_{assentamento}(s)$
$Motor\ 1$	0,0380	0,0640
$Motor\ 2$	0,0320	0,0540
$Motor\ 3$	0,0320	0,0580
$Motor\ 4$	0,0320	0,0580

Fonte: o autor.

4.3.3 Hardware utilizado

Para esta etapa, e também as demais, são utilizados dois módulos que se comunicam via radiofrequência. O módulo de comunicação utilizado neste projeto possui uma placa de desenvolvimento NUCLEO-H743ZI2, fabricada pela empresa STMicroelectronics, essa placa conta com um microcontrolador de alto desempenho baseado num núcleo Arm Cortex-M7, com frequência de até 400 MHz, 2 MB de memória FLASH, 512 KB de memória RAM, e conta com uma grande quantidade de portas digitais e analógicas, e de interfaces de comunicação (STMICROELECTRONICS, 2017). Além disso, o módulo de comunicação também conta um transceptor modelo nRF24L01+ com antena fabricado pela empresa Nordic Semiconductor, essa antena utiliza o método de transmissão de Modulação por Chaveamento de Frequência Gaussiana (GFSK), sendo a taxa de transmissão configurável entre 250 Kbps, 1 Mbps ou 2 Mbps e a banda de operação $2,4$ GHz (SEMICONDUCTOR, 2008) que permite a realização da coleta de dados através de telemetria.

O módulo de controle possui uma placa de desenvolvimento NUCLEO-F767ZI, que conta com um microcontrolador de alto desempenho baseado num núcleo Arm Cortex-M7, com frequência de até 216 MHz, 2 MB de memória FLASH, 1 MB de memória RAM, e conta com uma grande quantidade de portas digitais e analógicas, e de interfaces de comunicação (STMICROELECTRONICS, 2017). Junto à placa de desenvolvimento, há outro transceptor nRF24L01+ com a adição de *drivers* de motor *brushless* DC A3930 fabricados pela empresa Allegro MicroSystems, e motores *brushless* DC modelo EC-45 flat 50W com *encoder* fabricados pela empresa Maxon Group.

Os *drivers* A3930 fornecem comutação automática e controle de corrente para motores *brushless* DC de três fases de acordo com um valor de PWM com frequência fixa de referência, acionando MOSFETs externos de canal N de acordo com o valor atual dos sensores de efeito Hall do motor (MICROSYSTEMS, 2019). Os motores Maxon são de alta eficiência e são indicados para aplicações de alta precisão, podendo chegar à velocidade nominal máxima de até 5190

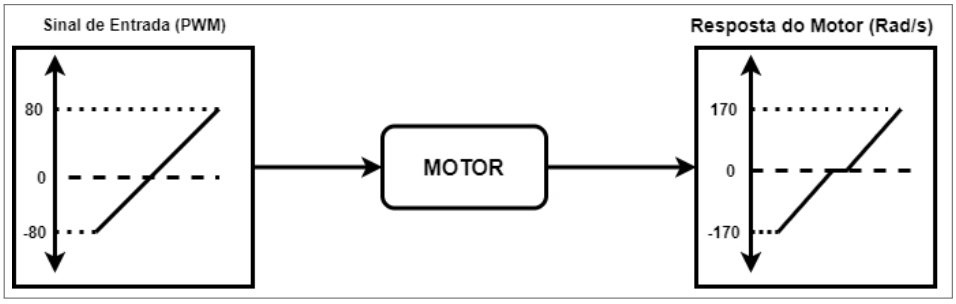
RPM para o modelo EC-45 flat 50W (MAXON, 2019).

4.3.4 Obtenção da curva dos motores

O objetivo desta etapa é entender o comportamento do motor quando estimulado com sinais de entrada variados, e determinar regiões de saturação do motor. Como resultado, será possível a identificação de parâmetros como a faixa de operação do motor, as zonas mortas (*dead zones*), que são as zonas onde o motor, mesmo sendo estimulado, não demonstra alteração na resposta, e também será possível cruzar esses dados com os requisitos de sistema e calcular se será necessário utilizar algum tipo de redução ou amplificação por engrenagens.

A Figura 31 ilustra, de maneira resumida, o experimento para a obtenção da curva de resposta de um motor. O sinal de entrada na forma de PWM é enviado para o motor numa frequência fixa adequada, e o motor irá responder de acordo com a variação do percentual de *duty cycle*. A resposta do motor é medida por um *encoder* e traduzida para velocidade rotacional em rad/s.

Figura 31 – Processo de obtenção da curva de um motor de maneira simplificada.



Fonte: o autor.

4.3.5 Descrição da técnica para obtenção da curva do motor

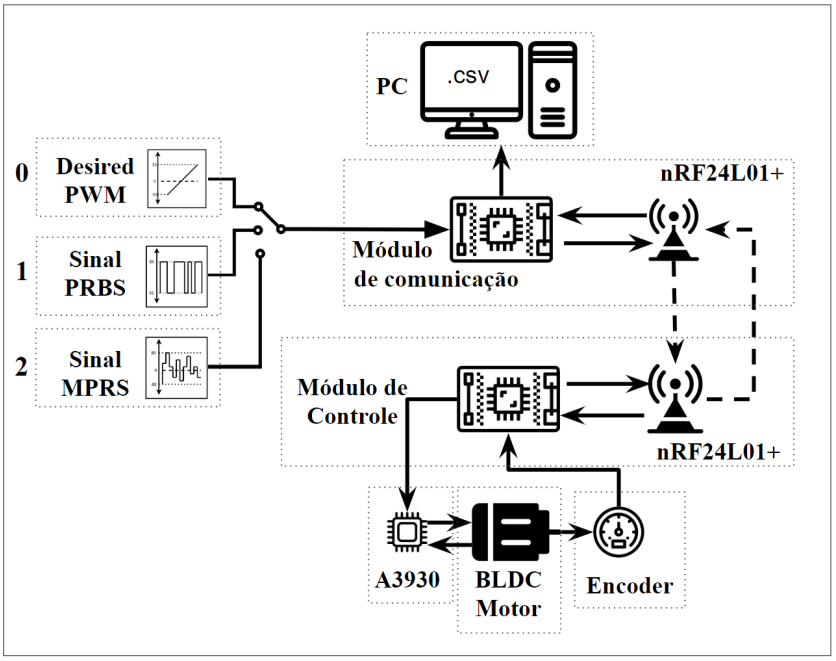
O ambiente de testes para obter a curva de resposta dos motores pode ser visto na Figura 32. Inicialmente, a chave é posicionada no valor 0, e valores de PWM são enviados pelo módulo de comunicação para o módulo de controle, que recebe a mensagem e executa os comandos recebidos. O objetivo final é encontrar a faixa de operação do motor medindo sua resposta a diferentes valores de PWM através do software embarcado que é executado na placa de desenvolvimento do módulo de controle. Também chamado de *basestation*, o módulo de comunicação será responsável por enviar pacotes de dados via radiofrequência para o módulo de controle, que também possui um transceptor para recepção dos pacotes, e está embarcado num robô omnidirecional.

O protocolo de comunicação utilizado foi proposto por Cavalcanti (2019), que estabelece uma comunicação de duas vias (duplex) entre os módulos, permitindo o envio e recebimento de pacotes com baixa taxa de perda. Uma vez com a mensagem decodificada, o módulo de

controle irá gerar o sinal PWM de acordo com o valor recebido, e o estímulo será enviado para o *driver* do motor (A3930), que por sua vez irá realizar a comutação do motor *brushless* de maneira automática de acordo com o PWM de entrada, e com a resposta do sensor que é fornecida para o controlador pelos sensores de efeito Hall do motor.

A velocidade do motor é lida por um *encoder* incremental, como descrito na Seção 2.2, e os sinais de saída (pulsos) são enviados para os canais A e B de um dos temporizadores (*timers*) do módulo de controle, programado no modo de codificação (*encoder mode*) do tipo X4. Esse modo realiza o processo de contagem dos pulsos e o número de pulsos é convertido para velocidade do motor. O valor da velocidade atual do motor é enviado, também por radiofrequência, para o modulo de comunicação. Uma vez recebido o pacote de velocidades, ele é decodificado e enviado para um computador através de comunicação serial, e os valores da velocidade são salvos num arquivo, para posteriormente serem analisados.

Figura 32 – Visão geral do experimento para obter a curva de um motor.



Fonte: o autor.

Os valores utilizados para encontrar a faixa de operação do motor foram no intervalo $[-80\ ;\ 80]\%$ de *duty cycle* com passo de $0,1\%$ a cada trinta leituras com intervalo entre amostras (T_{sample}). A escolha desse intervalo se dá pelos valores de tempo de assentamento obtidos na Seção 4.3.1, seguindo a seguinte relação:

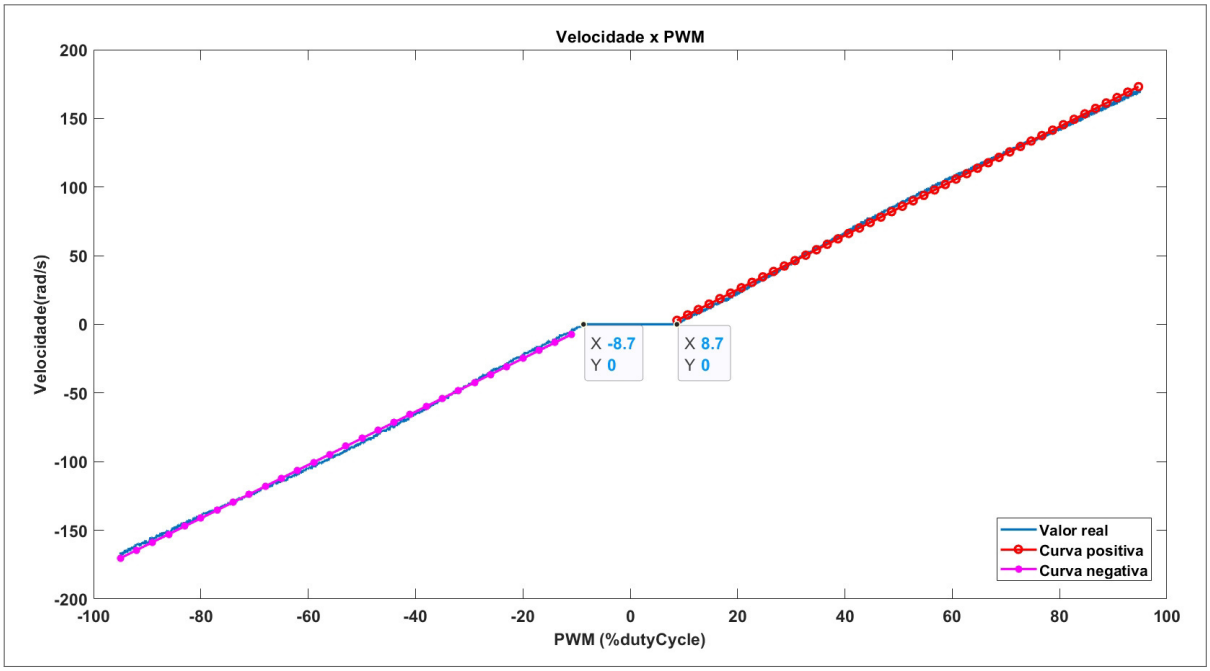
$$T_{sample} \geq MAX\{T_{assentamento}\} \tag{4.1}$$

Portanto, com base nos valores do tempo de assentamento obtidos na seção 4.3.2, foi escolhido o intervalo entre amostras de $0,1s$. A leitura de trinta amostras de velocidade igualmente espaçadas é feita com o objetivo de diminuir o ruído gerado pelo *encoder*. Durante

o pós-processamento, é feita uma média aritmética das trinta amostras e o valor dessa média é considerado na análise de curva do motor.

Uma das características dos motores *brushless* é a presença de uma zona morta (*dead zone*), que é um intervalo onde o motor, mesmo recebendo sinais de entrada, não apresenta movimento. A identificação dessa zona morta pode ser feita através do envio de estímulos ao motor que resultam em velocidade igual a zero. Um exemplo típico de resposta de motor é ilustrado na Figura 33, que mostra a resposta de um motor a um estímulo entre $[-95, 95\%]$ de *duty cycle*, onde a *dead zone* para este caso foi encontrada no intervalo $[-8, 7; 8, 7]$. Ao mesmo tempo, percebe-se a formação de duas curvas que podem ser aproximadas à retas: uma para velocidades negativas e uma para velocidades positivas. Essas retas podem ser utilizadas para auxiliar no projeto de um controlador mais complexo, ou também podem ser utilizadas na implementação de um controle simples em malha aberta.

Figura 33 – Curva velocidade *versus* PWM típica de um motor *brushless*.



Fonte: o autor.

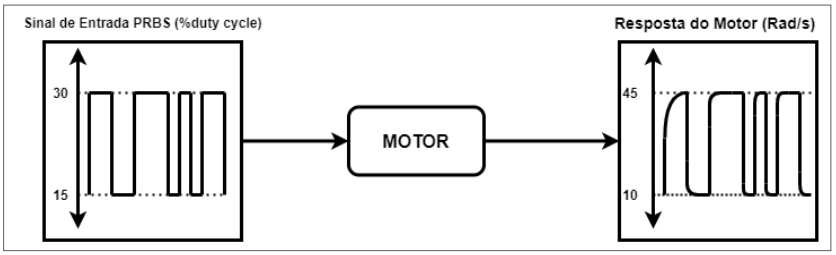
Os coeficientes das retas foram obtidos através da ferramenta de aproximação polinomial do MATLAB *polyfit*, que faz uso do método dos mínimos quadrados. Dado como parâmetros de entrada um conjunto de pontos (x, y) e um parâmetro n, a função *polyfit* retorna os coeficientes para um polinômio $p(x)$ de grau n que melhor se ajustam aos valores de y (MATHWORKS, 2019). Para esta aplicação, foram encontrados polinômios de grau 1 ($n = 1$), pois, ao analisar as características das duas curvas em situação real, nota-se um comportamento linear em ambas.

4.4 DETERMINAÇÃO DO MODELO DO MOTOR

O objetivo desta etapa é obter um modelo aproximado do motor em uso para a aplicação específica de controle de velocidade com base nos dados observados e no conhecimento do sistema a priori. Keesman afirma que os principais componentes no procedimento de identificação de um modelo são: um conhecimento a priori do sistema, objetivos, e os dados coletados (KEESMAN, 2011). Como explicado na seção 2.3.2.5, devido à complexidade que envolve a modelagem matemática do conjunto *driver* mais motor *brushless* DC, o método de identificação do modelo utilizado nesta etapa foi o método *black-box*. Para medir o percentual de ajuste entre o modelo de saída e os dados observados, foi utilizada a ferramenta de identificação de sistemas do MATLAB, o processo será descrito na seção 4.4.2.

A Figura 34 ilustra, de maneira resumida, o processo para aquisição dos dados utilizados para obter o modelo do motor. Neste caso, um sinal de entrada na forma de PRBS é enviado para o motor com os parâmetros referentes ao tipo de motor da aplicação, e a resposta do motor é lida através do *encoder* com uma taxa de amostragem específica e traduzida para velocidade rotacional em rad/s.

Figura 34 – Processo de obtenção do modelo de um motor de maneira simplificada.



Fonte: o autor.

Os dados coletados na aquisição de dados são utilizados como entrada na ferramenta de identificação de sistemas do MATLAB, que irá estimar um modelo de resposta do motor que melhor represente a relação entre os sinais de entrada e saída. Foi elaborado um experimento envolvendo sinais PRBS e outro experimento com sinais MPRS com os motores no cenário real de atuação, e a resposta a esses sinais foi gravada em um arquivo, e posteriormente processada pelo software MATLAB, que dá como resultado uma função de transferência em tempo discreto que descreve, de maneira aproximada, o comportamento físico do sistema. O hardware utilizado foi o mesmo descrito na seção 4.3.3, e os métodos utilizados para gerar os estímulos de entrada foram descritos na seção 2.3.3.

Embora alguns autores como Oguntoyinbo et al. (2009), Patel e Pandey (2013) e Anshory et al. (2019) tenham derivado um modelo do motor *brushless* DC através de equações diferenciais, a adição de um *driver* específico para a comutação do motor torna a obtenção do modelo teórico do sistema bem mais complexa. Portanto, para a obtenção de modelos mais complexos, a metodologia de identificação de modelo mais indicada é a *black-box*, pois não é necessário entender o modelo interno do sistema, somente as respostas do motor aos sinais

de entrada é avaliada. Alguns trabalhos, como Hussin, Azuwir e Zaiazmin (2011) e Hat et al. (2015) utilizaram essa metodologia para obtenção do modelo do motor.

4.4.1 Descrição dos experimentos para identificação do modelo do motor

Foram executados dois experimentos para a identificação do modelo, e foi feita uma comparação entre os resultados desses experimentos para que fosse escolhido o modelo com o maior percentual de ajuste de acordo com os dados de validação. O primeiro experimento consiste em utilizar sinais PRBS para estimular o motor, já o segundo consiste em utilizar sinais MPRS como entrada. A metodologia para implementação de ambas as abordagens será descrita nas próximas seções.

4.4.1.1 Obtenção do modelo utilizando sinais PRBS

A geração dos sinais PRBS segue as recomendações da seção 2.3.3.2, e é feita através da função *idinput* do software MATLAB. De acordo com LJUNG (1995), essa função retorna uma matriz ou vetor coluna com os sinais de entrada que devem ser utilizados para identificação de um sistema. Os parâmetros de entrada da função *idinput* estão resumidos na Tabela 14. O número de amostras pseudo-aleatórias de entrada geradas é dado pela variável P, e deve seguir as recomendações da Equação 2.5. O tipo de sinal de entrada a ser gerado também precisa ser especificado. Os limites inferior e superior da banda de passagem também podem ser alterados através da variável $band$. A variável $level$, que é um vetor que define os valores máximo e mínimo do sinal de entrada.

Tabela 14 – Parâmetros utilizados para a geração do sinal de entrada.

Descrição	**Variável**	**Valor**
Número de amostras	P	4095
Período de amostragem	T	0.05
Número de canais de entrada	nu	1
Tipo de entrada	$type$	$prbs$
Banda	$band$	$[0; 1]$
Valores máximo e mínimo do sinal de entrada	$level$	$[15; 35]$

Fonte: o autor.

O período de amostragem do sinal (T) deve ser maior que o tempo de subida do sistema utilizado, que considerando o maior tempo dentre os valores coletados na Seção 4.3.2, deve

ser maior que $0.038s$. O número de valores da variável (P), que influencia diretamente na duração da geração dos va, deve ser grande o suficiente para que saída do sistema seja capaz de eliminar distúrbios ou não-linearidades que atuam sobre o sistema durante a identificação. A banda padrão significa que serão exploradas todas as faixas possíveis na geração do sinal pseudo-aleatório. A escolha do sinal de excitação, como discutido na seção 2.3.3.2, é cerca de 10% do valor máximo da variável de entrada, que na nossa aplicação é de $[-100; 100]$, por isso, foi escolhida a variação no intervalo $[15; 35]$.

O estratégia desenvolvida para obter o modelo do motor através da metodologia *black-box* também é descrito pela Figura 32 apresentada anteriormente. Para esta etapa da estratégia a chave deve estar na posição 1 para a geração do sinal PRBS no módulo de comunicação, ao invés do sinal PWM. Os valores de *duty cycle* correspondentes ao sinal PRBS gerado são definidos no software embarcado desenvolvido e executando na *basestation* do módulo de comunicação, e não é necessário fazer alterações adicionais em relação à etapa descrita na seção 4.3.5.

Os pacotes com as informações são enviados para o módulo de controle embarcado no robô omnidirecional, que por sua vez irá retornar o valor da velocidade atual de cada um dos motores ao receber o estímulo. O pacote de velocidades enviado pelo robô é decodificado e enviado para o módulo principal executando no computador utilizando comunicação serial, esses dados são guardados em um arquivo para análise pela ferramenta de identificação de sistemas do MATLAB. A frequência de atualização do valor do *encoder* é de $2ms$. O experimento, quando executado sob as condições estabelecidas nesta seção, dura cerca de 270 segundos.

4.4.1.2 Obtenção do modelo utilizando sinais MPRS

Para a geração dos sinais MPRS é necessário determinar alguns parâmetros, são eles: número de amostras do experimento, período entre amostras, taxa de amostragem do sensor e faixa de operação do motor. O primeiro parâmetro a ser determinado será o número de amostras do sinal (n). O método utilizado para determinar a quantidade de amostras é o proposto por Cochran (1977). Assume-se uma população grande para este caso, pois, a quantidade de amostras que pode ser extraída pode ser tão grande quanto o projetista escolher. Portanto, a Equação 4.2 pode ser utilizada para obter o número de amostras:

$$n = \frac{Z^2 pq}{e^2} \tag{4.2}$$

onde:

- Z é obtido através de tabelas estatísticas, corresponde à abcissa da curva normal que cobre a área para um intervalo de confiança α;
- p é a proporção estimada entre a população que contém ou não o atributo a ser testado;
- q é dado por $1 - p$;

- e é o grau de precisão desejado.

Considerando um nível de confiança de 95%, o valor de Z correspondente é $1,96$. Não conhece-se a variabilidade da proporção entre a população que contém ou não o modelo de motor a ser obtido, portanto o valor de p será considerado como o máximo de variabilidade $(0,5)$, logo, q também será $0,5$. É desejado que o resultado esteja dentro do intervalo de precisão de $\pm 5\%$. Substituindo os valores na equação 4.2, temos:

$$n = \frac{(1,96)^2(0,5)(0,5)}{(0,05)^2} = 385$$

Portanto, o número de amostras deve ser, no mínimo, 385 para os requisitos considerados, portanto, para facilitar a geração dos sinais pseudo-aleatórios serão utilizadas 511 amostras, que corresponde a substituição de n por 9 na Equação 2.5 para facilitar a geração do sinal. O período entre as amostras deve ser maior ou igual ao tempo de assentamento do motor medido experimentalmente. O valor de $0,3s$ para o período entre amostras foi escolhido para que seja possível detectar a variabilidade da leitura quando a resposta alcança o regime permanente. A taxa de amostragem do sensor do (*encoder*) determina o intervalo entre as amostras da resposta do motor, que será de $2ms$ para que seja possível visualizar o comportamento transiente do motor. Já a faixa de operação depende do experimento da seção 4.3.5 que indica em qual faixa de operação o motor se comporta da maneira esperada pelo projetista.

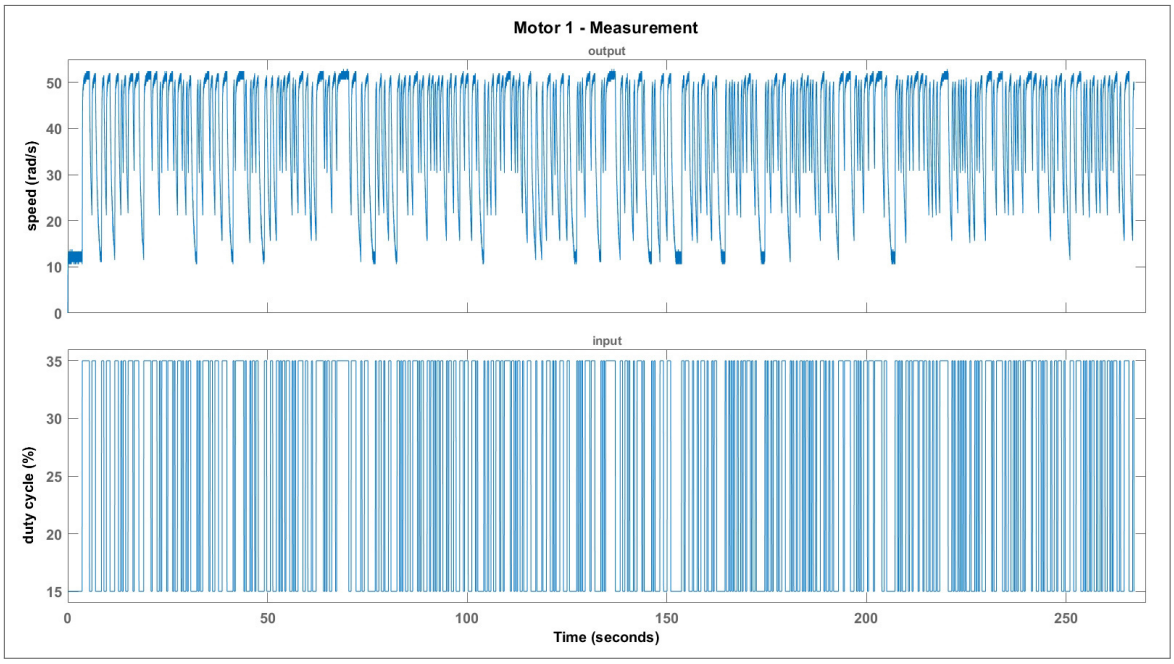
4.4.2 Ferramenta de identificação do modelo

De acordo com LJUNG (1995), a identificação de sistemas é uma metodologia para construir modelos matemáticos de sistemas dinâmicos utilizando medidas de entrada do sistema e sinais de saída. Inicialmente, o processo de identificação de um sistema requer que o projetista faça leituras dos sinais de entrada e saída do sistema, seja no domínio da frequência, seja no domínio do tempo. Em seguida, é necessário selecionar um modelo estrutural para que seja aplicado um método de estimativa para o ajuste dos parâmetros do modelo candidato, e por fim deve se avaliar se o modelo estimado se está adequado à aplicação.

O sinal de entrada e a resposta do motor são importados para o espaço de trabalho do MATLAB. Esses dados, juntamente com o intervalo de amostragem, são unidos num objeto do tipo *iddata*, que encapsula os valores e propriedades dos dados numa única entidade para facilitar a manipulação dos dados. A Figura 35 apresenta uma visualização do objeto *iddata* resultado da técnica utilizando sinais PRBS da seção 4.4.1.1 aplicado a um dos motores. Da mesma forma, a Figura 36 apresenta uma visualização do objeto *iddata* resultado da técnica utilizando sinais MPRS da seção 4.4.1.2 aplicado a um dos motores.

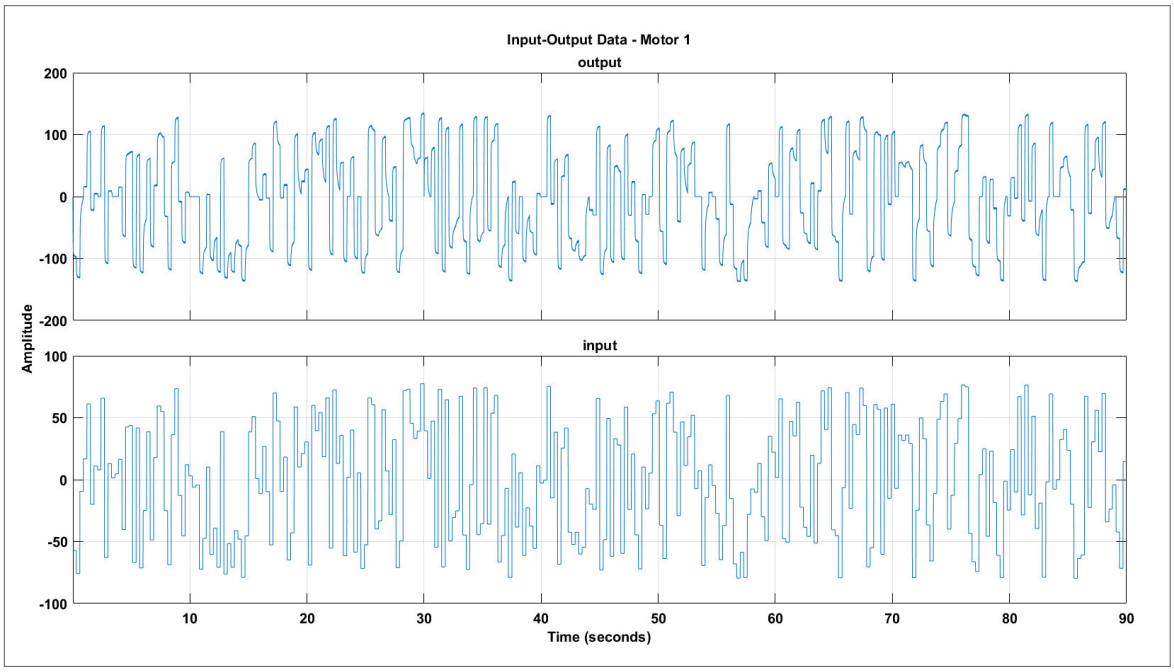
Uma vez neste formato, os dados de cada motor são importados para a ferramenta *system identification* do MATLAB, onde serão pré-processados através do método descrito em LJUNG (1995), e um modelo discreto que melhor se aproxima da resposta de cada motor é

Figura 35 – Resposta do motor a uma entrada PRBS.



Fonte: o autor.

Figura 36 – Resposta do motor a uma entrada MPRS.



Fonte: o autor.

obtido. O objetivo da etapa de pré-processamento é preparar os dados para que o algoritmo de identificação do sistema possa extrair a melhor representação possível, e também subdividir os dados em conjuntos de teste e validação.

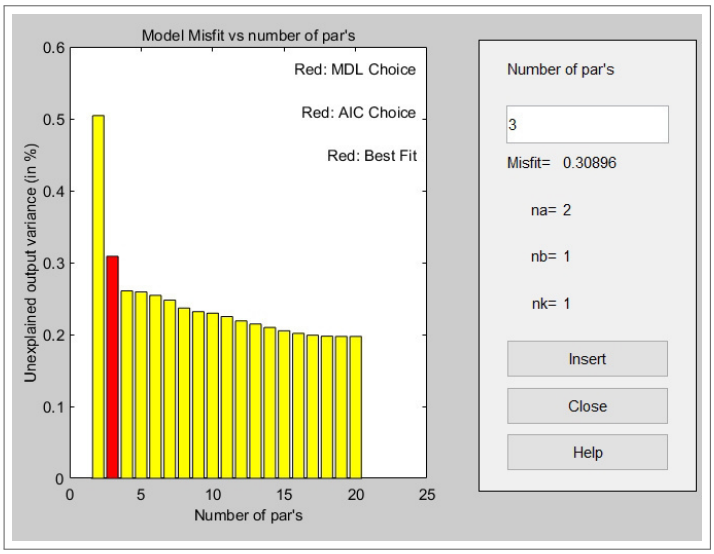
O primeiro passo da identificação de modelos *black-box* é especificar qual é a ordem do modelo, ou seja, o número de parâmetros a serem otimizados. A escolha de um número de pa-

râmetros pequenos torna o modelo muito simples, porém pode não explorar todo o conjunto de dados; enquanto um grande número de parâmetros pode tornar o modelo desnecessariamente complexo, podendo falhar na generalização para outros casos, fenômeno também conhecido como *overfitting*.

Uma maneira simples de encontrar a ordem mais adequada da função de transferência a ser estimada é desenvolver modelos polinomiais autorregressivos do tipo ARX de várias ordens e atrasos (*delays*) (ANSHORY et al., 2019). A comparação do desempenho desses modelos é feita a partir de uma grandeza chamada de "variância inesperada na saída", que representa o percentual da saída do sistema que não pode ser explicado pelo modelo obtido. Essa grandeza permite identificar o grau polinomial mais indicado, pois, a partir dele, não há uma melhora significativa na representação do modelo.

A estimativa de ordem obtida para os dados da Figura 35 pode ser visualizada na Figura 37. Percebe-se que a partir da coluna destacada em vermelho o percentual de variância da saída se mantém aproximadamente constante, indicando que o desempenho do modelo não é melhor em ordens superiores. As grandezas na, nb e nk na figura representam, respectivamente, o número de polos, número de zeros subtraído de um e o *delay* de amostras do modelo da coluna em destaque. Esses valores serão utilizados para estimar uma função de transferência em tempo discreto para os dados.

Figura 37 – Modelos polinomiais ARX estimados para o motor.



Fonte: o autor.

Esse teste foi repetido para os quatro motores utilizados na validação deste projeto, tanto com o conjunto de dados com sinais PRBS quanto com o conjunto de dados com sinais MPRS. Os resultados apresentados mostraram os mesmos número de polos, zeros e *delay* do motor avaliado neste parágrafo, portanto, as funções de transferência estimadas terão dois pólos, nenhum zero, e o *delay* de uma amostra.

O algoritmo de identificação da função de transferência em tempo discreto com dados no domínio do tempo utiliza o método de estimativa do erro polinomial de saída para determinar os coeficientes polinomiais do numerador e denominador (MATWORKS, 2019). Neste algoritmo, a inicialização é feita com dos dados de resposta, e a quantidade de polos, zeros e *delay*, em seguida utiliza-se o método de mínimos quadrados não linear para minimizar a norma do erro de previsão ponderado. Como resultado, o algoritmo retorna os valores estimados dos parâmetros do modelo, um valor do erro de previsão final do modelo, e um percentual do quão a resposta do modelo se ajusta aos dados de entrada representado pela Raiz Quadrada do Erro Médio Quadrático Normalizado (NRMSE). Uma vez estimadas a função de transferência, é possível projetar um controlador PID para o controle do motor de acordo com os requisitos de projeto.

4.5 TUNING DOS PARÂMETROS DO CONTROLADOR PI

A etapa de *tuning* de parâmetros é executada totalmente em *software*, e consiste em encontrar as constantes de controle ótimas para que a planta responda de acordo com os requisitos de projeto. Em geral, os métodos de *tuning* mais utilizados são a abordagem proposta por Ziegler-Nichols e tentativas e erros (YADAV; TAYAL, 2018). Uma alternativa a essas técnicas é a utilização da API do MATLAB denominada *PIDTuner*, que faz parte do conjunto de ferramentas de controle chamado *Control Systems Toolbox*.

Os controladores PI são mais comuns para respostas mais rápidas, mas são mais suscetíveis a *overshoot*. A presença do termo derivativo torna a subida do sistema mais agressiva, mas o sistema fica mais sensível a ruídos dos sensores e aumenta sua complexidade. Para este projeto, foi escolhido o uso de um controlador PI pois o foco principal da aplicação é eliminar o erro em regime permanente, e, de acordo com os requisitos de tempo de resposta, não é preciso que o sistema apresente uma subida tão agressiva.

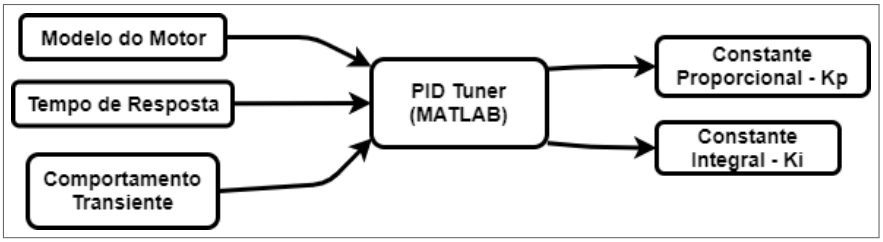
4.5.1 Metodologia de tuning

Este trabalho propõe a utilização da ferramenta *PIDTuner* do MATLAB para obtenção dos parâmetros mais adequados do controlador PI. Essa ferramenta permite o projeto de controladores de maneira interativa, tornando possível a visualização em tempo real dos resultados simulados a partir de ajustes realizados manualmente tanto na resposta ao tempo, quanto no comportamento transiente do sistema. De acordo com Tang, Liu e Wang (2017), o algoritmo do *PIDTuner* busca ajustar os ganhos do controlador PID para alcançar um sistema com equilíbrio entre desempenho e robustez, buscando a estabilidade dinâmica do sistema em malha fechada.

A Figura 38 mostra o processo de *tuning* do modelo, proposto para obtenção das constantes de controle proporcional e integral. O *PIDTuner* recebe como parâmetros de entrada o modelo do motor obtido pelo método descrito na seção 4.4, o tempo de resposta de controle,

que é um parâmetro que condiciona o tempo de subida do controlador, e uma constante de comportamento transiente, que irá atuar de maneira a diminuir o *overshoot* do sistema.

Figura 38 – Processo de *tuning* do controlador PI utilizando a ferramenta *PID Tuner*



Fonte: o autor.

De acordo com a seção 4.2 referente aos requisitos de projeto, o tempo de subida precisa ser menor que $0,3s$, e o *overshoot* não deve ser maior que 5%. Portanto, faz-se necessário uma avaliação através da ferramenta interativa para cada modelo para ajustar os parâmetros de tempo de resposta de controle e o comportamento transiente de forma que o resultado fique dentro das expectativas dos requisitos.

É importante ressaltar que alguns aspectos como a saturação do atuador causada pela ação integral e a não linearidade associada aos motores e ao *encoder* no mundo real não são considerados pelo *PIDTuner*. Por isso, alguns dos resultados apresentados podem mostrar tempos de resposta superiores aos projetados, mas essa divergência pode ser considerada aceitável desde que o resultado final esteja dentro dos requisitos desejados.

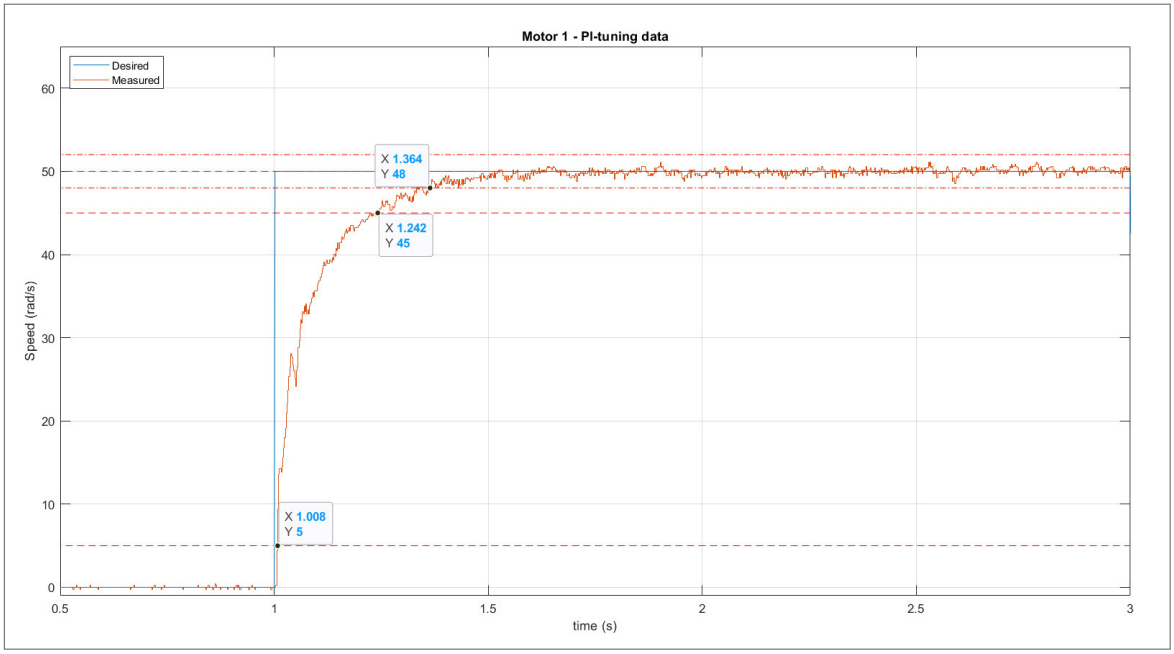
4.5.2 Experimento para validação do controlador PI

Nesta seção será descrito o experimento utilizado para validar a implementação do controlador PI. É necessário analisar se, de fato, as constantes de controle obtidas na etapa de *tuning* apresentam resultados condizentes com os parâmetros especificados. Os experimentos serão executados num ambiente de validação real, utilizando o robô descrito do estudo de caso descrito no capítulo 5, e a coleta de dados é feita através de comunicação sem fio, assim como nos experimentos anteriores.

A primeira parte do experimento consiste em aplicar uma entrada em degrau com velocidade de referência igual para todos os motores. O objetivo desta etapa é medir a resposta do motor e identificar parâmetros como erro em regime permanente, tempo de subida, tempo de assentamento e *overshoot*. A Figura 39 mostra uma entrada em degrau aplicada a um dos motores utilizados.

A segunda parte se refere a uma análise de resposta do motor a diferentes velocidades de referência. Serão escolhidos vinte valores arbitrários que serão enviados para cada um dos motores com um intervalo de tempo predefinido entre cada velocidade de referência. A Figura 40 mostra o experimento aplicado a um dos motores utilizados com um intervalo de $1,5s$ entre cada velocidade. O objetivo desta etapa é analisar o comportamento do motor durante

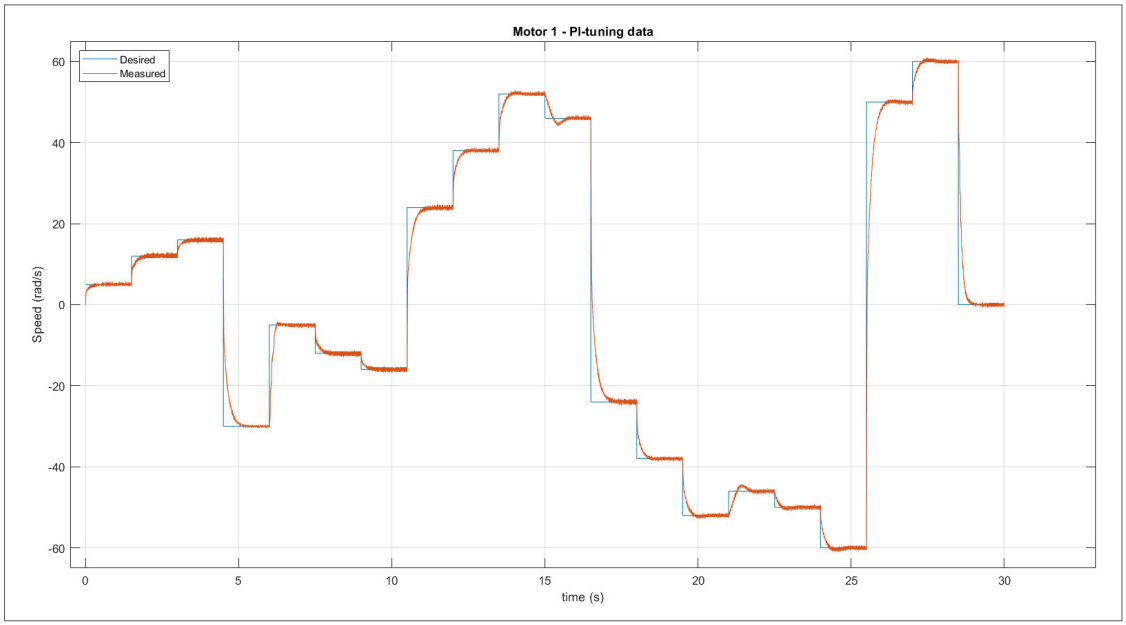
Figura 39 – Entrada em degrau com valor de referência $50\ rad/s$ aplicada a um dos motores.



Fonte: o autor.

a transição de velocidades e verificar se o motor consegue estabilizar na velocidade desejada dentro dos limites de tolerância estabelecidos pelos requisitos do projeto.

Figura 40 – Múltiplas velocidades de referência aplicadas a um dos motores.



Fonte: o autor.

Espera-se que um sistema controlado pela abordagem proposta que apresenta os parâmetros de controle dentro dos requisitos de projeto após a realização dos experimentos indicados esta seção esteja apto para o funcionamento no ambiente real.

5 ESTUDO DE CASO - APLICAÇÃO EM UM ROBÔ OMNIDIRECIONAL

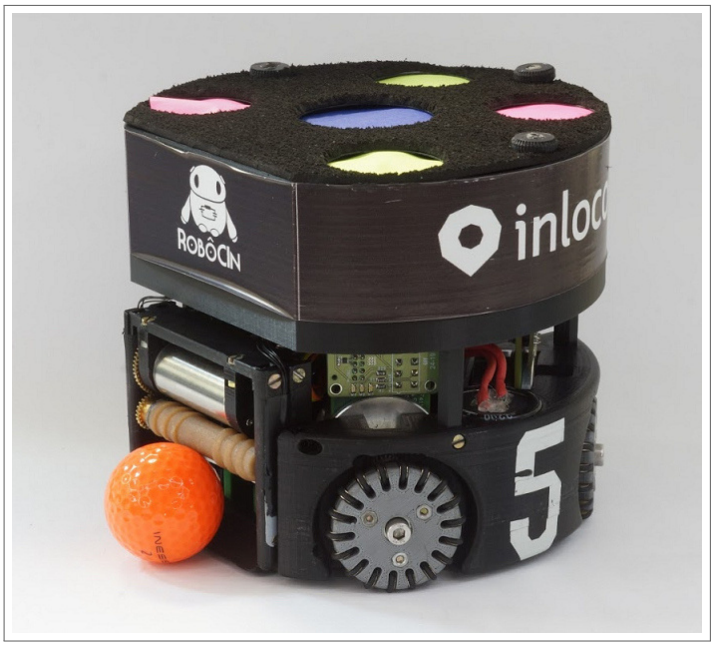
Esta etapa consiste em validar a abordagem proposta em uma aplicação real, bem como avaliar o desempenho da abordagem proposta com relação a uma abordagem empírica de controle baseado num controlador PD. Portanto, foi colocado como estudo de caso o projeto de um robô omnidirecional projetado para a competição *RoboCup* SSL, que inclui o hardware e o software embarcado do módulo de controle descrito na seção 4.3.3 com quatro motores modelo EC-45 flat 50W. Primeiramente, o robô e suas características físicas serão apresentados. Em seguida, um controlador PI discreto será implementado no robô para o controle de movimento. O próximo passo é entender como o robô se movimenta através de uma modelagem cinemática, tornando possível controlar o robô omnidirecional em qualquer direção no plano x-y. Por fim, será detalhado o experimento para coleta dos resultados usando a abordagem proposta neste trabalho.

5.1 DESCRIÇÃO DO ROBÔ UTILIZADO

O robô utilizado neste estudo de caso foi totalmente projetado de acordo com as regras da competição *RoboCup* SSL (COMMITEE, 2019). Existe uma restrição para o tamanho de cada robô,que pode ter no máximo $180\ mm$ de diâmetro e $150\ mm$ de altura. A Figura 41 apresenta o robô utilizado para validação juntamente com uma bola de golfe padrão que é utilizada no jogo. Os quatro motores são dispostos de maneira a permitir a livre movimentação do robô para qualquer direção com o auxílio de rodas omnidirecionais. Todo o robô foi modelado através do software *Autodesk Inventor 2019 professional*, e a estrutura do robô foi impressa em 3D utilizando como material o Ácido Polilático (PLA). Na parte superior do robô é colocado um padrão de cores fornecido pela competição para a identificação dos robôs através do sistema de visão compartilhado *SSL-vision*, com isso, é possível ter em tempo real informações da localização de cada robô no campo, tanto da posição em relação a um sistema de coordenadas global, quanto da orientação.

As informações de posicionamento do robô e campo são utilizadas pelo chamado módulo de controle de alto nível, que fica num computador central e toma decisões em tempo real, enviando para o robô mensagens de velocidade linear e angular para que o robô reaja às mudanças do ambiente em que ele está inserido, como será detalhado na seção 5.3. O módulo de controle descrito na seção 4.3.3 se encontra embarcado no robô, e é responsável pelo funcionamento de todo o sistema, que é alimentado por uma bateria de quatro células de 2200 mAh.

Figura 41 – Robô omnidirecional da categoria *Small Size League* utilizado no estudo de caso.

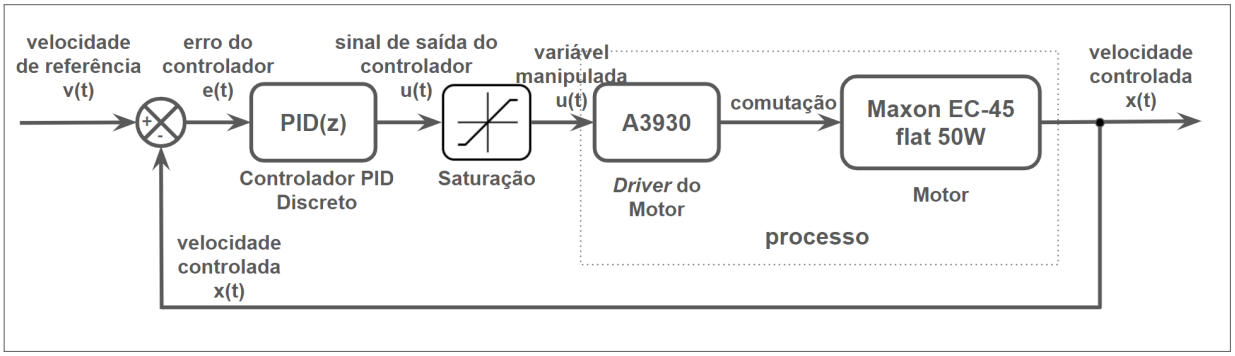


Fonte: (SILVA et al., 2020)

5.2 IMPLEMENTAÇÃO DO CONTROLADOR PI DISCRETO

O diagrama de blocos do controlador PI desenvolvido para o robô pode ser visualizado na Figura 42. O erro ($e(t)$) é a diferença entre a velocidade de referência ($v(t)$) e a velocidade controlada ($x(t)$). Esse erro alimenta o controlador PI, que por sua vez irá produzir um sinal de saída ($u(t)$) que passa por um bloco de saturação para garantir que os valores do controlador não ultrapassem o limite físico do atuador, só então o sinal é enviado para o processo.

Figura 42 – Visão geral do sistema de controle implementado



Fonte: o autor.

Como visto na seção 2.4, o comportamento de um controlador PID contínuo no tempo pode ser dado pela equação 2.6, onde, aplicando a transformada de Laplace, têm-se a seguinte

equação:

$$C(s) = K_p + \frac{K_i}{s} + K_d \cdot s \tag{5.1}$$

A melhor forma de discretizar esse controlador é converter os termos integral e derivativo para suas respectivas partes discretas no tempo. Para essa aplicação, o método escolhido para a discretização do controlador foi o da diferença atrasada, ou *backward difference* (ÅSTRÖM; HÄGGLUND, 1995; THAM, 1996). Esse método consiste em fazer um mapeamento entre a transformada de Laplace e o domínio z através de uma conversão aproximada, considerando o tempo de amostragem T_s da medida da variável controlada. Aplicando o método *backward difference* e eliminando o termo derivativo, que não será utilizado para neste estudo de caso, após algumas simplificações, têm-se o seguinte equação para o comportamento do controlador PI digital:

$$u(t) = K_p \cdot e(t) + K_i \cdot T_s \cdot \sum_{i=0}^{t} e(t) + u_0 \tag{5.2}$$

O termo u_0 é o valor inicial de saída do controlador, e o termo integral é dado pela multiplicação da constante integral (K_i) pelo tempo de amostragem e pelo o acúmulo de todos os erros durante o tempo de execução do controle. A saturação irá atuar para o caso de o termo integral ser muito grande e extrapolar os valores limites de atuação, impedindo o motor de operar fora da sua faixa de operação. Os valores de saturação são obtidos através do teste realizado na seção 4.3.5, onde foi possível identificar a margem de atuação de cada motor.

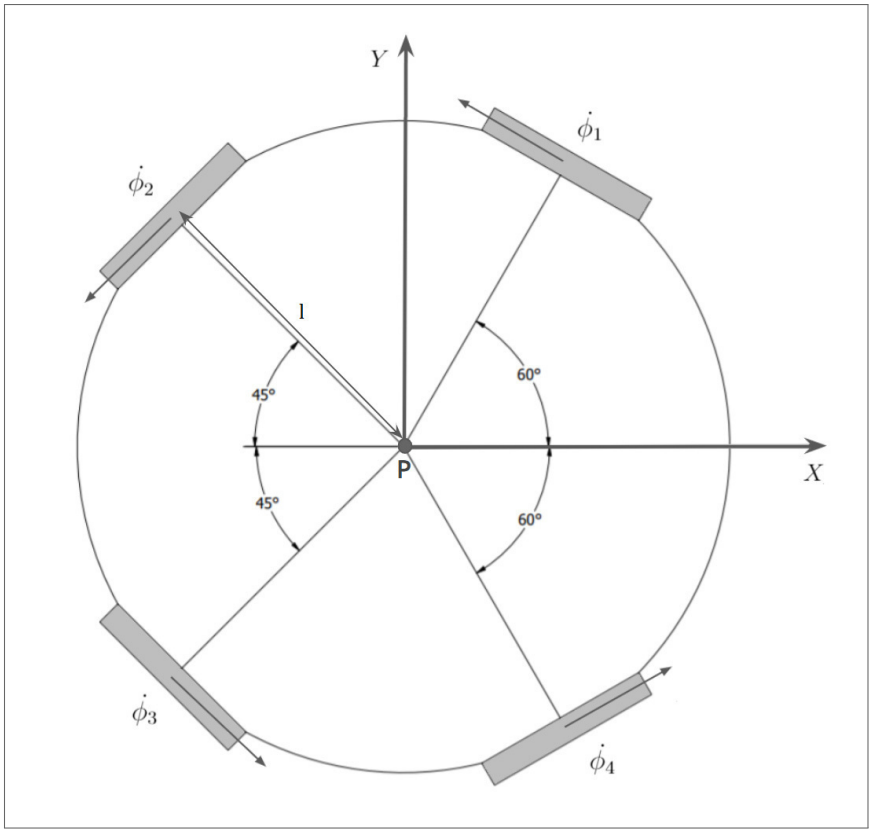
5.3 OBTENÇÃO DO MODELO CINEMÁTICO DO ROBÔ

Como forma de validar o controlador PI implementado no robô, é necessário entender como o robô se movimenta para que sejam enviados os valores corretos de velocidade de referência de cada motor. Como explicado na seção 2.6, o papel da modelagem cinemática do robô é descrever o movimento em termos de uma coordenada de referência. Considerando o robô omnidirecional para a competição *Small Size League* foi desenvolvida uma modelagem cinemática utilizando como base os estudos de Siegwart, Nourbakhsh e Scaramuzza (2011) e Rojas e Förster (2006).

A Figura 43 apresenta uma vista superior simplificada do robô utilizado. Este robô possui quatro rodas suecas (omnidirecionais) de $90°$, a distância entre cada roda e o ponto central do robô (P) é dada pela distância l. Considera-se, também, que todas as quatro rodas possuem o mesmo raio (r). A configuração das rodas não é uniforme, nota-se que duas rodas estão a um ângulo de $45°$ e as outras duas rodas estão a $60°$ do eixo X. Embora a configuração ideal para diminuir a influência do atrito na movimentação seja que todas as rodas estejam a um ângulo de $45°$, essa configuração foi implementada pelo fato de que as duas rodas frontais

do robô precisam ser mais abertas para que seja possível o robô dominar a bola (driblar), e chutar, aumentando o leque de jogadas possíveis.

Figura 43 – Arranjo do robô omnidirecional e sua distribuição de velocidades



Fonte: o autor.

Como mostrado na Equação 2.12, as velocidades linear e angular do robô são representadas pela grandeza $\dot{\xi}_I$. Essas velocidades podem ser obtidas através de uma combinação das restrições de rolamento do robô com suas quatro rodas omnidirecionais. O cálculo para obtenção das velocidades é feito resolvendo a Equação 2.15 para a variável $\dot{\xi}_I$, que resulta na Equação 5.3:

$$\dot{\xi}_I = R(\theta)^{-1} J_{1f}^{-1} J_2 \dot{\phi} \tag{5.3}$$

Já a matriz J_{1f}, como detalhado na seção 2.6, é calculada utilizando a matriz de restrições de rolamento para as rodas suecas proposto por Siegwart, Nourbakhsh e Scaramuzza (2011). Para a construção dessa matriz, é necessário obter os valores dos ângulos α, β e γ para cada roda. Para uma roda sueca de $90°$, têm-se que $\gamma = 0$, o valor de α para cada roda, em radianos, de acordo com o arranjo do robô omnidirecional utilizado para este estudo de caso, é dado por:

$$\alpha_n = \left(\frac{\pi}{3}, -\frac{\pi}{3}, \frac{3\pi}{4}, -\frac{3\pi}{4}\right) \forall \left\{n \in \mathbb{N} \mid 1 \leq n \leq 4\right\}. \tag{5.4}$$

Como todas as rodas são tangentes ao corpo circular do robô, temos $\beta = 0$. Considerando o sentido dos vetores de velocidade da Figura 43 e seus respectivos ângulos, a matriz J_{1f} é construída da seguinte forma:

$$J_{1f} = \begin{pmatrix} -sin\left(\frac{\pi}{3}\right) & cos\left(\frac{\pi}{3}\right) & l \\ -sin\left(\frac{3\pi}{4}\right) & -cos\left(\frac{3\pi}{4}\right) & l \\ sin\left(\frac{3\pi}{4}\right) & -cos\left(\frac{3\pi}{4}\right) & l \\ sin\left(\frac{\pi}{3}\right) & cos\left(\frac{\pi}{3}\right) & l \end{pmatrix} \tag{5.5}$$

O objetivo da modelagem cinemática para este estudo de caso é calcular a velocidade que cada roda deve ter para que a soma das contribuições das rodas faça com que o robô se movimente com as velocidades globais desejadas, dadas por $\dot{\xi}_I$. Para calcular a velocidade que cada roda deve ter, a equação 5.3 é rearranjada da seguinte forma:

$$\dot{\phi} = \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4 \end{bmatrix} = J_2^{-1} J_{1f} R(\theta) \dot{\xi}_I \tag{5.6}$$

A matriz rotacional $R(\theta)$, dada pela Equação 2.10, mostra uma dependência da orientação do robô em relação ao eixo de coordenadas de referência global adotado. Como a matriz J_2 é uma matriz identidade multiplicada por um escalar, sua inversa consiste apenas na inversão dos termos da diagonal principal. Portanto, substituindo os valores dos termos da Equação 5.6, temos que a velocidade de cada roda é dada pela Equação 5.7:

$$\dot{\phi} = \begin{bmatrix} \frac{1}{r} & 0 & 0 & 0 \\ 0 & \frac{1}{r} & 0 & 0 \\ 0 & 0 & \frac{1}{r} & 0 \\ 0 & 0 & 0 & \frac{1}{r} \end{bmatrix} \begin{bmatrix} -sin(60°) & cos(60°) & l \\ -sin(45°) & -cos(45°) & l \\ sin(45°) & -cos(45°) & l \\ sin(60°) & cos(60°) & l \end{bmatrix} \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{5.7}$$

Dados os parâmetros do robô utilizado, como o raio da roda $r = 0,02425\ m$ e a distância da roda para o centro do robô $l = 0,081\ m$, a equação 5.7 pode ser simplificada para a equação 5.8 a seguir:

$$\dot{\phi} = \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4 \end{bmatrix} = \begin{bmatrix} -35.7124 & 20.6186 & 3,3402 \\ -29.1590 & -29.1590 & 3,3402 \\ 29.1590 & -29.1590 & 3,3402 \\ 35.7124 & 20.6186 & 3,3402 \end{bmatrix} \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{5.8}$$

Multiplicando as matrizes da Equação 5.8, temos:

$$\begin{bmatrix} \dot{\phi_1} \\ \dot{\phi_2} \\ \dot{\phi_3} \\ \dot{\phi_4} \end{bmatrix} = \begin{bmatrix} -(35,7124cos\theta + 20,6186sin\theta)\dot{x} + (-35,7124sin\theta + 20,6186cos\theta)\dot{y} + 3,3402\dot{\theta} \\ (-29,1590cos\theta + 29,1590sin\theta)\dot{x} - (29,1590sin\theta + 29,1590cos\theta)\dot{y} + 3,3402\dot{\theta} \\ (29,1590cos\theta + 29,1590sin\theta)\dot{x} + (29,1590sin\theta - 29,1590cos\theta)\dot{y} + 3,3402\dot{\theta} \\ (35,7124cos\theta - 20,6186sin\theta)\dot{x} + (35,7124sin\theta + 20,6186cos\theta)\dot{y} + 3,3402\dot{\theta} \end{bmatrix} \tag{5.9}$$

A equação 5.9 mostra que, para controlar a movimentação de um robô omnidirecional de quatro rodas, é necessário saber o ângulo do robô em relação a uma coordenada de referência global (θ), enviar as velocidades lineares ($\dot{x}$ e $\dot{y}$) em relação à coordenada global, e enviar também a velocidade rotacional ($\dot{\theta}$) que o robô deve executar. O somatório vetorial dessas velocidades de cada roda irá fazer com que o robô se movimente para a direção especificada.

5.4 DESCRIÇÃO DOS EXPERIMENTOS PARA VALIDAÇÃO DO CONTROLADOR PI

Nesta seção serão descritos os experimentos realizados para a validação do controlador PI proposto quando aplicado a este estudo de caso. A hipótese a ser testada é a de que com um controlador PI bem projetado o robô irá seguir as trajetórias desejadas de maneira mais precisa. A justificativa para isso está no fato de que cada roda contribui para o movimento resultante do robô. Portanto, ao tentar eliminar o erro em regime estacionário de cada motor, o vetor resultante do movimento do robô irá apontar para uma direção muito próxima da direção ideal, fazendo com que o robô siga a trajetória almejada.

Os experimentos para validar este estudo de caso foram realizados a partir da definição de duas trajetórias de referência com relação ao eixo de coordenadas global: uma trajetória de referência linear e uma circular. A trajetória linear corresponde a um quadrado de lado $1,5m$. Por sua vez, a trajetória circular corresponde a um círculo de diâmetro $1m$. A comunicação é feita via radiofrequência para minimizar ruídos externos, e o robô recebe os comandos para executar as trajetórias em várias velocidades.

Foram enviadas para o robô as velocidades linear e angular que correspondem à trajetória de referência desejada. Para a execução da trajetória linear, foi escolhida uma velocidade linear específica e foi calculado o tempo teórico para que, nessa velocidade, o robô execute um dos lados do quadrado. Por exemplo, para o robô andar $1,5m$ na direção x, o tempo de teórico de execução da trajetória seria 3 segundos a uma velocidade linear de $0,5\ m/s$. Já para a execução da trajetória angular, foi escolhida uma combinação de velocidades linear com angular, fazendo com que o robô contorne um círculo ao redor de um ponto de rotação. Os resultados dos experimentos serão apresentados no próximo capítulo.

6 RESULTADOS

Neste capítulo serão apresentados os resultados dos experimentos realizados para validação da abordagem proposta. Inicialmente, serão apresentadas as curvas de resposta dos motores testados, tornando possível a extração de parâmetros para definição da faixa de operação dos motores. Em seguida, serão apresentados os resultados do experimento para obtenção do modelo dos motores, bem como para definição das funções de transferência em tempo discreto obtidas utilizando sinais de entrada PRBS e MPRS de acordo com a faixa de operação do motor definida no capítulo anterior.

O modelo do motor com maior percentual de *fitting* será considerado para a etapa de *tuning* do controlador PI através da ferramenta *PIDTuner* do MATLAB e satisfazendo os requisitos de projeto do sistema. Por fim, têm-se a etapa de validação do controlador, que foi implementado no módulo de controle dos motores reais estudados. Foram realizados testes com um valor de referência de entrada em degrau, e também com múltiplas entradas para averiguar se a resposta dos motores atende aos requisitos de tempo real do projeto. Os resultados são comparados a uma implementação de controlador baseado numa abordagem empírica de tentativas e erros.

Por fim, serão apresentados os resultados dos experimentos do estudo de caso. Esses resultados serão avaliados utilizando as constantes de controle obtidas pela abordagem proposta e comparados com os resultados do método empírico obtidos sob as mesmas condições.

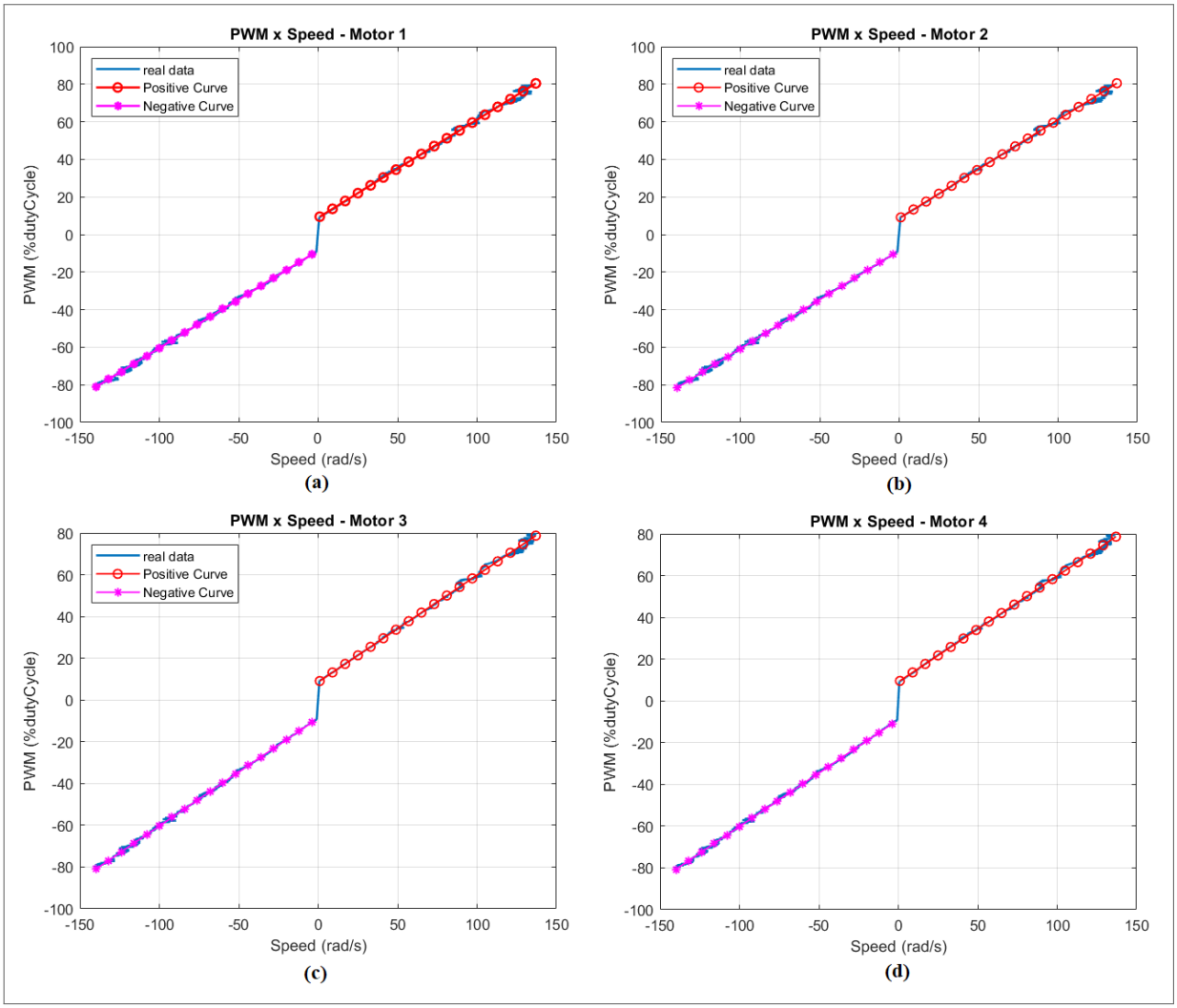
6.1 MODELO DOS MOTORES

O primeiro passo para identificação da função de transferência é obter a curva de resposta dos motores para se definir a faixa de operação do motor. Uma vez conhecida a faixa de atuação do motor, pode-se implementar os métodos de identificação de sistemas. As informações da faixa de operação também são importantes na implementação da saturação necessária para evitar que o controlador PI gere valores de saída que ultrapassem o limite físico do motor.

6.1.1 Curvas de resposta do motor

A Figura 44 apresenta os resultados do experimento da seção 4.3.5 para os quatro motores estudados. Foi aplicado um sinal PWM variando dentro do intervalo $[-80\ ;\ 80]$, com passo de $0,1$ PWM a cada $0,3s$. Para diminuir os ruídos que atuam sobre o *encoder* durante cada intervalo de tempo de $0,3s$, são medidas trinta amostras da resposta do motor e a média dessas amostras é considerada como ponto da curva real. É possível identificar duas curvas que representam de maneira simplificada a relação entre o percentual de PWM aplicado ao motor e a resposta do motor em rad/s, uma positiva e uma negativa. Essas curvas apresentam um comportamento linear, e podem ser aproximadas por equações de primeiro grau.

Figura 44 – Curva de resposta PWM $\times$ Velocidade juntamente com as curvas aproximadas para: (a) Motor 1; (b) Motor 2; (c) Motor 3; e (d) Motor 4.



Fonte: o autor.

A Tabela 15 mostra as equações de primeiro grau aproximadas. Foi escolhido o intervalo entre -80 e 80, pois, como o experimento foi utilizado em situação real, quando testados valores de PWM acima desse valor, o robô utilizado para validação entra em condição de deslizamento, e os dados começam a apresentar uma variabilidade muito alta. Portanto, foram estabelecidos limites superior e inferior para evitar a saturação do sistema. Além disso, foi estabelecido um valor de rotação mínimo de $\pm 1\ rad/s$ para que o robô consiga sair da inércia, evitando a chamada zona morta, que como explicado na seção 4.3.5, corresponde à faixa de operação onde o motor não apresenta resposta mesmo quando é estimulado por sinais de baixo valor de PWM.

Tabela 15 – Equações da curva PWM (y em $\%$ *duty cycle*) $\times$ Velocidade (x) (em rad/s) para cada motor.

Motor	**Equação**
Motor 1	$y = \begin{cases} 80 & \forall\ x \geq 140 \\ 0,5216 \cdot x + 8,973 & \forall\ 1 < x < 140 \\ 0 & \forall\ -1 \leq x \leq 1 \\ 0,5188 \cdot x - 8,572 & \forall\ -140 < x < -1 \\ -80 & \forall\ x \leq -140 \end{cases}$
Motor 2	$y = \begin{cases} 80 & \forall\ x \geq 140 \\ 0,5248 \cdot x + 8,608 & \forall\ 1 < x < 140 \\ 0 & \forall\ -1 \leq x \leq 1 \\ 0.5226 \cdot x - 8,520 & \forall\ -140 < x < -1 \\ -80 & \forall\ x \leq -140 \end{cases}$
Motor 3	$y = \begin{cases} 80 & \forall\ x \geq 140 \\ 0,5120 \cdot x + 8,585 & \forall\ 1 < x < 140 \\ 0 & \forall\ -1 \leq x \leq 1 \\ 0,5170 \cdot x - 8,679 & \forall\ -140 < x < -1 \\ -80 & \forall\ x \leq -140 \end{cases}$
Motor 4	$y = \begin{cases} 80 & \forall\ x \geq 140 \\ 0,5079 \cdot x + 9,063 & \forall\ 1 < x < 140 \\ 0 & \forall\ -1 \leq x \leq 1 \\ 0,5124 \cdot x - 8,948 & \forall\ -140 < x < -1 \\ -80 & \forall\ x \leq -140 \end{cases}$

Fonte: o autor.

As curvas positiva e negativa dos motores foram obtidas através da função *polyfit* do MATLAB, que utiliza o método dos mínimos quadrados para aproximar o conjunto de pontos em retas. A escolha de uma aproximação em retas de primeiro grau se deu pelas características

da resposta, pois, além de evitar o *overfitting*, quanto mais simples a equação, menor será o custo computacional da sua implementação. Embora cada um dos motores tenha suas não-linearidades associadas, bem como detalhes de montagem e influências externas, as curvas de resposta obtidas são bastante semelhantes, e a curva de um motor se ajusta a outro sem perder as características do motor.

A Tabela 16 mostra a qualidade do ajuste das curvas, ou seja, o quão bem as curvas aproximadas representam as leituras reais. As métricas utilizadas para avaliar a qualidade de ajuste foram Raiz do Erro Médio Quadrático (RMSE) e o *R-square*; quanto menor o RMSE, melhor a curva está representada. Já para a métrica *R-square*, quanto mais próximo de 1 for seu valor, mais bem ajustada é a curva.

Tabela 16 – Métricas para avaliar a qualidade de ajuste das retas aproximadas.

Motor	**Equação**	**RMSE**	**R-square**
Motor 1	$0,5216 \cdot x + 8,973$	0,7108	0,9988
	$0,5188 \cdot x - 8,572$	0,8867	0,9981
Motor 2	$0,5248 \cdot x + 8,608$	0,4876	0,9994
	$0.5226 \cdot x - 8,520$	0,7617	0,9986
Motor 3	$0,5120 \cdot x + 8,585$	0,5467	0,9993
	$0,5170 \cdot x - 8,679$	0,6790	0,9989
Motor4	$0,5079 \cdot x + 9,063$	0,6976	0,9988
	$0,5124 \cdot x - 8,948$	0,6778	0,9989

Fonte: o autor.

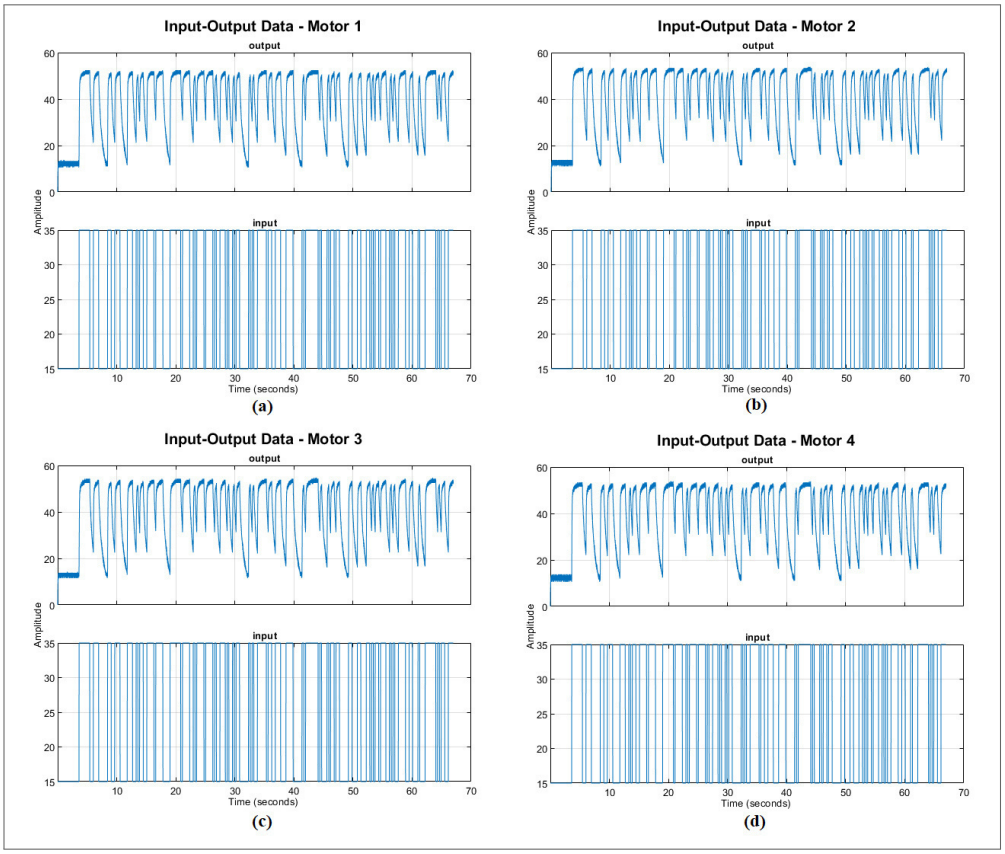
Outra maneira de validar as curvas obtidas é computar as diferenças entre os valores reais dos modelos e os valores previstos nas retas aproximadas, esse processo é conhecido como análise residual. Essas diferenças, também chamadas de resíduos, são utilizadas para estimar o erro do modelo aproximado, que pode ser utilizado como métrica para estipular um erro máximo para que o modelo seja considerado aceito. Uma análise residual de cada uma das curvas (positiva e negativa) obtidas é apresentada de maneira detalhada na Apêndice A. É possível notar na análise que os valores previstos estão variando dentro do intervalo ± 4 PWM, e que a variabilidade aumenta conforme a velocidade do motor também aumenta. Essa informação é muito importante para entender que, quando implementado, o controlador

PI proposto também terá maior instabilidade quando for requisitada a operação em altas velocidades.

6.1.2 Experimentos de obtenção do modelo dos motores

A identificação de modelos aproximados dos motores possibilita ter uma espécie de cópia digital dos motores simulando a resposta dos motores a estímulos de entrada. Uma vez conhecido um modelo aproximado, é possível projetar um controlador PI e fazer testes com constantes de controle para extrair e explorar o potencial do motor utilizado sem o risco de danificar o circuito de controle dos motores reais. Os experimentos para obtenção do modelo dos motores descritos na seção 4.4.1 foram executados. Os sinais de entrada PRBS e MPRS foram aplicados de acordo com seus respectivos experimentos em ambiente real, e a resposta para cada um dos casos foi obtida através de comunicação sem fio e importada para a ferramenta de identificação de sistemas (*System Identification*) do MATLAB. A Figura 45 mostra um fragmento do experimento com sinal PRBS de entrada aplicado aos quatro motores.

Figura 45 – Fragmento da resposta (em rad/s) dos motores a um sinal PRBS (em PWM) como entrada para: (a) Motor 1; (b) Motor 2; (c) Motor 3; e (d) Motor 4.



Fonte: o autor.

A entrada varia entre 15 e 35 % de PWM com frequência variada, e o sinal de saída corresponde ao valor da velocidade atual do motor (em rad/s) lido pelo *encoder*. Os quatro

motores são submetidos simultaneamente ao mesmo sinal de entrada, tornando o processo de obtenção do modelo do motor mais rápido.

Os dados são coletados pelo módulo de comunicação e armazenados num arquivo de texto, que é importado para a ferramenta *System Identification* do MATLAB. Nela, é feito um pré-processamento dos dados para remoção de *outliers*, bem como a divisão dos dados em conjuntos de treinamento e de validação. O próximo passo é estimar a função de transferência em tempo discreto a partir do conjunto de treinamento, e avaliar a qualidade de ajuste dessa função no conjunto de validação. Como descrito na seção 4.4.2, testes feitos nos motores mostraram que as funções estimadas devem possuir, no mínimo, dois polos, nenhum zero, e *delay* de uma amostra. Esses parâmetros são inseridos na ferramenta, e os resultados podem ser visualizados na Tabela 17.

Tabela 17 – Função de transferência identificada para os Motores utilizando sinais PRBS.

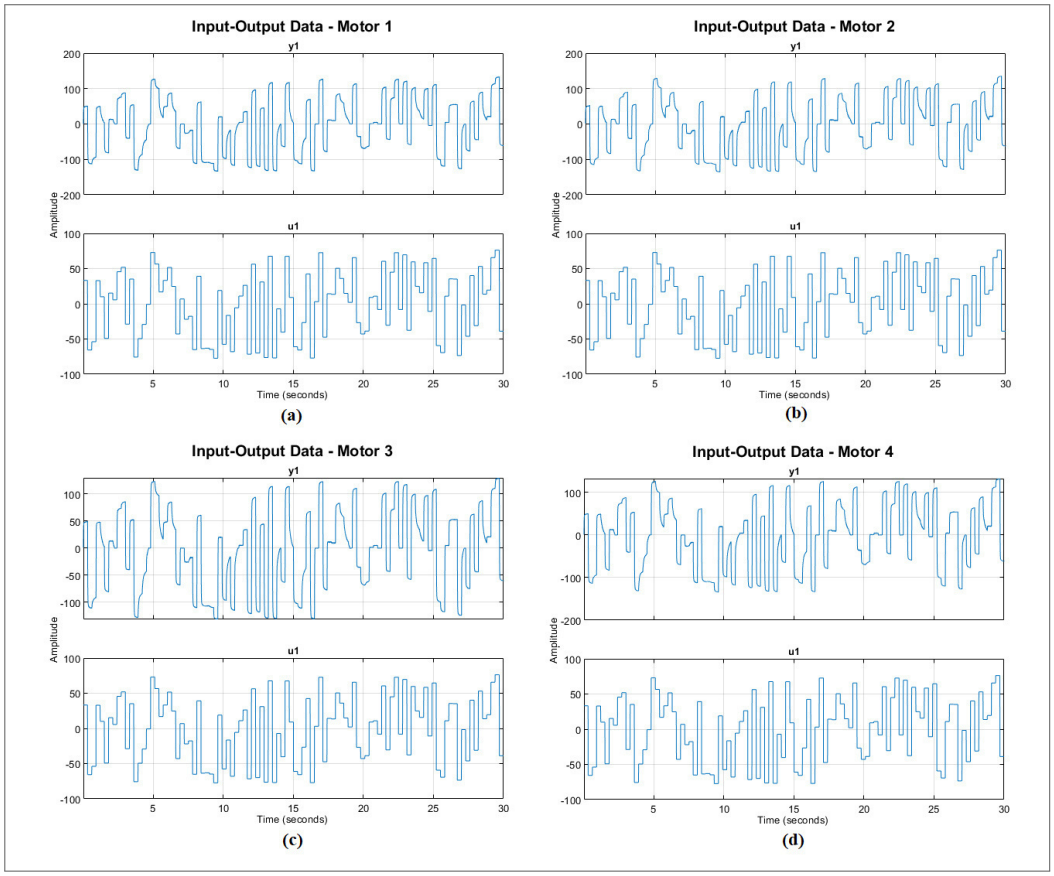
Motor	**Função de Transferência em Tempo Discreto**	**Fitting (%)**
$Motor\ 1$	$\frac{0{,}03621}{1-0{,}01787\cdot z^{-1}-0{,}09589\cdot z^{-2}}$	$63,81$
$Motor\ 2$	$\frac{0{,}03661}{1-0{,}02167\cdot z^{-1}-0{,}09553\cdot z^{-2}}$	$64,56$
$Motor\ 3$	$\frac{0{,}03717}{1-0{,}02247\cdot z^{-1}-0{,}09544\cdot z^{-2}}$	$64,92$
$Motor\ 4$	$\frac{0{,}03598}{1-0{,}02274\cdot z^{-1}-0{,}09545\cdot z^{-2}}$	$63,89$

Fonte: o autor.

O percentual de ajuste (*fitting*) entre o modelo e os dados reais do conjunto de validação para ambos os sinais de entrada é calculado de acordo com a métrica NRMSE. A Figura 46 mostra um fragmento do experimento com sinal MPRS de entrada aplicado aos quatro motores. A entrada varia o sinal PWM dentro da faixa de atuação do motor, ou seja, de -80 a $80\ \%$, e o sinal de saída corresponde ao valor da velocidade atual do motor (em rad/s) lido pelo *encoder*.

O experimento foi realizado nos quatro motores simultaneamente, e os resultados foram coletados e armazenados num arquivo de texto. O processo de obtenção da função de transferência também é o mesmo, passando pela etapa de pré-processamento e remoção de *outliers*. A Tabela 18 apresenta as funções de transferência obtidas para os mesmos motores do experimento anterior, só que utilizando como entrada um sinal MPRS. O *fitting* entre o modelo e os dados do conjunto de validação também é calculado de acordo com a métrica NRMSE. Percebe-se, portanto, que o experimento com sinal de entrada MPRS obteve um resultado com maior acurácia em relação ao experimento com entrada PRBS, isso pode ser

Figura 46 – Fragmento da resposta (em rad/s) dos motores a um sinal MPRS (em PWM) como entrada para: (a) Motor 1; (b) Motor 2; (c) Motor 3; e (d) Motor 4.



Fonte: o autor.

explicado pelo fato do sinal MPRS atuar em toda a faixa de operação do motor, enquanto que o sinal PRBS atua em apenas uma faixa específica.

Tabela 18 – Função de transferência identificada para os motores utilizando sinais MPRS.

Motor	**Função de Transferência em Tempo Discreto**	**Fitting (%)**
$Motor\ 1$	$\frac{0{,}1221}{1-1{,}327\cdot z^{-1}-0{,}3999\cdot z^{-2}}$	$90,67$
$Motor\ 2$	$\frac{0{,}1176}{1-1{,}349\cdot z^{-1}-0{,}4176\cdot z^{-2}}$	$90,83$
$Motor\ 3$	$\frac{0{,}1463}{1-1{,}258\cdot z^{-1}-0{,}3472\cdot z^{-2}}$	$90,89$
$Motor\ 4$	$\frac{0{,}1307}{1-1{,}298\cdot z^{-1}-0{,}3756\cdot z^{-2}}$	$90,62$

Fonte: o autor.

6.1.3 Comparação entre os modelos obtidos

Embora cada motor tenha suas não linearidades associadas, o método com sinais MPRS utilizado foi capaz de filtrar melhor os distúrbios e entregar um modelo mais próximo do real quando comparado ao conjunto de validação. Uma forma de avaliar as funções obtidas é comparar o desempenho das duas funções com um experimento real do motor, conforme mostrado na Tabela 19, que faz uma comparação entre o tempo de subida dos modelos ajustados e os valores reais identificados na seção 4.3.2 quanto a uma entrada em degrau.

Tabela 19 – Comparação do tempo de subida de resposta ao degrau entre os modelos aproximados e o real.

Motor	**Tempo de Subida (Real)**	**Tempo de Subida (PRBS)**	**Tempo de Subida (MPRS)**
$Motor\ 1$	$0,038$	$0,370$	$0,032$
$Motor\ 2$	$0,032$	$0,372$	$0,030$
$Motor\ 3$	$0,032$	$0,370$	$0,028$
$Motor\ 4$	$0,032$	$0,378$	$0,030$

Fonte: o autor.

Já a Tabela 20 faz a comparação entre o tempo de assentamento entre os modelos experimentais e os valores reais quanto a uma entrada em degrau é aplicada. A partir dos dados obtidos, percebe-se que os parâmetros obtidos pelo método MPRS são bem próximos do real. Enquanto que os obtidos através do método PRBS não conseguem representar bem o comportamento transiente do motor analisado. Tanto o tempo de subida quanto o tempo de assentamento dos modelos obtidos utilizando sinais de entrada PRBS são aproximadamente 10 vezes maiores que o tempo de subida real para todos os motores. Já nos resultados do método MPRS, os valores estão bem próximos para ambos os parâmetros. Portanto, os modelos dos motores que serão considerado para as próximas etapas de *tuning* e validação do controlador PI são os obtidos através do método MPRS.

É importante ressaltar que os resultados analisados pelos métodos PRBS e MPRS foram obtidos para apenas um modelo de motor *brushless* DC, sendo necessário replicar os testes para outros modelos para concluir que o método MPRS é mais eficiente que o método PRBS para qualquer motor. Da mesma forma, os valores são para um sistema em malha aberta, ou seja, sem a implementação de um controlador. Portanto, esses parâmetros devem ser analisados posteriormente quando o controlador for implementado no sistema.

Tabela 20 – Comparação do tempo de assentamento de resposta ao degrau entre os modelos aproximados e o real.

Motor	**Tempo de Assentamento (Real)**	**Tempo de Assentamento (PRBS)**	**Tempo de Assentamento (MPRS)**
$Motor\ 1$	$0,064$	$0,662$	$0,056$
$Motor\ 2$	$0,054$	$0,666$	$0,058$
$Motor\ 3$	$0,058$	$0,662$	$0,050$
$Motor\ 4$	$0,058$	$0,674$	$0,054$

Fonte: o autor.

6.2 TUNING DO CONTROLADOR PI

Nesta etapa, foi utilizada a ferramenta interativa *PIDTuner* do MATLAB. Os modelos dos motores obtidos na etapa anterior são importados, e é necessário especificar alguns parâmetros para que a ferramenta retorne os valores das constantes de controle. A Tabela 21 ilustra os parâmetros utilizados para obtenção das constantes de controle do estudo de caso.

Tabela 21 – Parâmetros utilizados para o *tuning* para obtenção das constantes de controle ótimas.

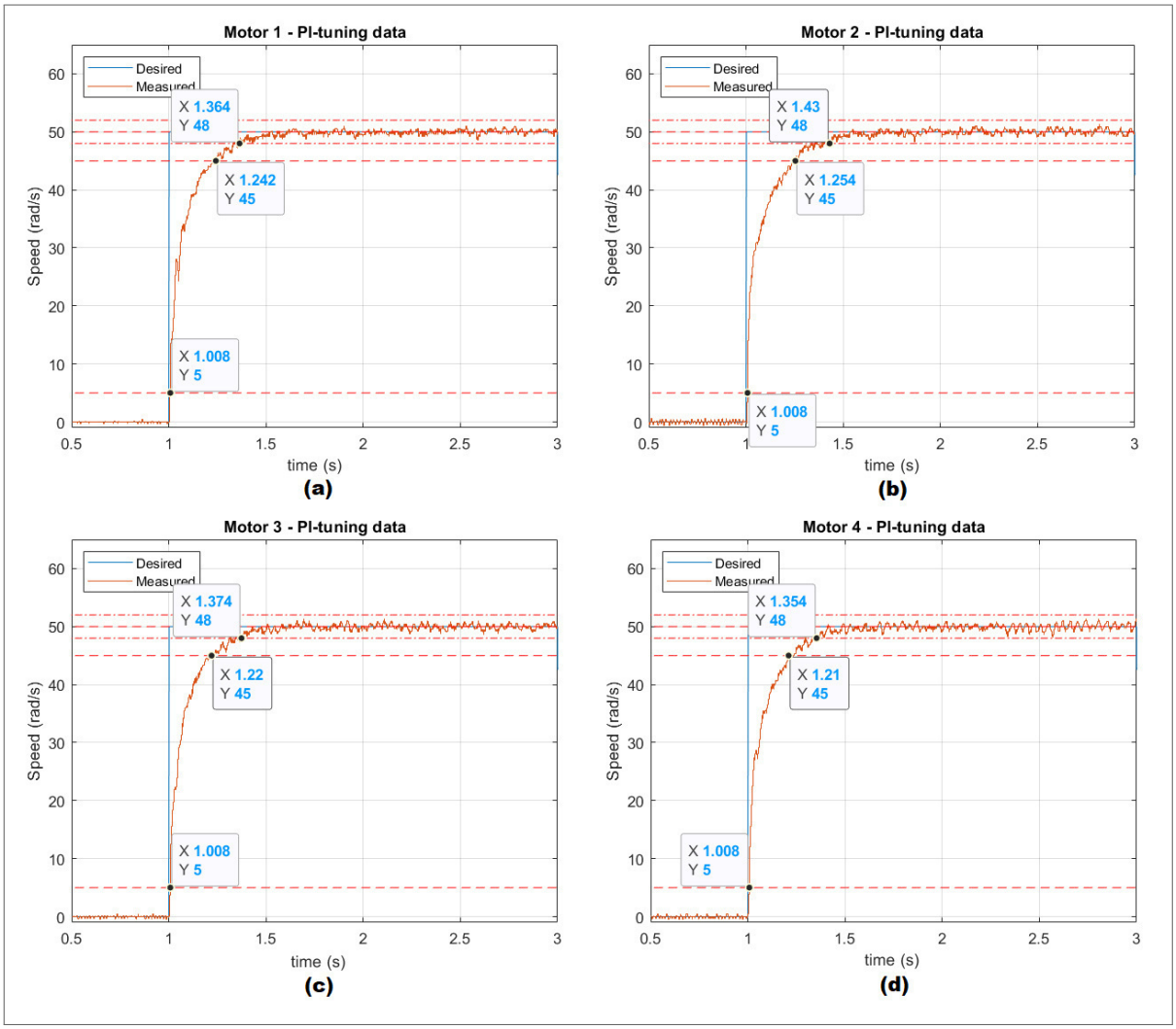
Parâmetro	**Valor**
Tipo	PI
Domínio	Tempo
Tempo de Resposta	$0,1$
Comportamento Transiente	$0,9$

Fonte: o autor.

Uma vez adicionados os parâmetros, a ferramenta atualiza em tempo real o comportamento da resposta do modelo já com o controlador integrado, e retorna as constantes do controlador PI indicadas para os parâmetros predefinidos. Essas constantes devem ser utilizadas num controlador real para verificar se o sistema se comporta da maneira esperada. A

primeira parte do experimento de validação proposto na seção 4.5.2 foi executada, e pode ser visualizada na Figura 47, que mostra a resposta dos motores a uma entrada em degrau de $50\ rad/s$ com a abordagem proposta implementada. A Tabela 22 resume os parâmetros de resposta dos motores obtidos para a abordagem proposta.

Figura 47 – Resposta (em rad/s) dos motores com controlador PI-tuning quanto a uma entrada em degrau de 50 rad/s: (a) Motor 1; (b) Motor 2; (c) Motor 3; e (d) Motor 4.



Fonte: o autor.

É possível perceber que o sistema controlado mantém o erro em regime permanente dentro da taxa de amostragem do *encoder*, não há *overshoot* e os tempos de subida e de assentamento estão dentro dos requisitos de projeto definidos na seção 4.2.

A abordagem proposta com controlador PI é comparada com uma técnica empírica que implementa um controlador PD. Esta técnica vinha sendo utilizada para controle do robô do estudo de caso deste trabalho. O ganho integral não foi implementado anteriormente, devido ao fato de não terem sido encontrados valores para as constantes com baixo *overshoot* durante os testes. A técnica empírica é baseada em tentativas e erros, onde as constantes são ajustadas

e testadas manualmente de acordo com a resposta do motor.

Tabela 22 – Parâmetros de resposta do motor para a abordagem PI-tuning.

Parâmetro	Motor 1	Motor 2	Motor 3	Motor 4
Tempo de Subida (s)	$0,234$	$0,246$	$0,212$	$0,202$
Tempo de Assentamento(s)	$0,364$	$0,430$	$0,374$	$0,354$
***Overshoot*(%)**	0	0	0	0
Erro em Regime Permanente(rad/s)	$0,364$	$0,43$	$0,374$	$0,354$

Fonte: o autor.

A Figura 48 mostra a resposta dos motores a uma entrada em degrau de $50\ rad/s$ com o controlador PD implementado utilizando a técnica empírica. O experimento foi sob as mesmas condições submetidas ao controlador PI proposto. A Tabela 23 resume os parâmetros de resposta dos motores obtidos para a abordagem empírica com o controlador PD.

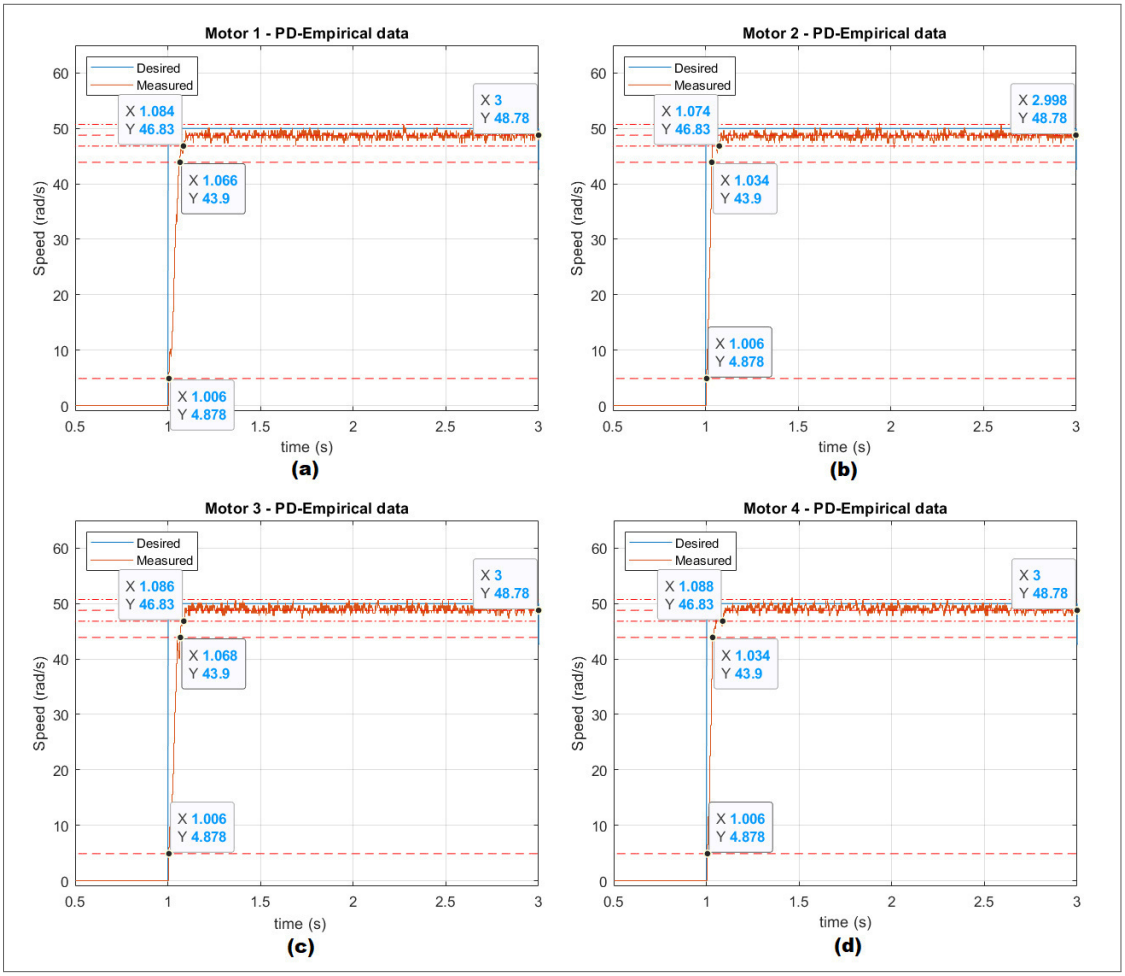
Tabela 23 – Parâmetros de resposta do motor para a abordagem PD-Empírica.

Parâmetro	Motor 1	Motor 2	Motor 3	Motor 4
Tempo de Subida (s)	$0,06$	$0,028$	$0,062$	$0,028$
Tempo de Assentamento(s)	$0,084$	$0,074$	$0,086$	$0,088$
***Overshoot*(%)**	0	0	0	0
Erro em Regime Permanente(rad/s)	$1,22$	$1,22$	$1,22$	$1,22$

Fonte: o autor.

A abordagem empírica traz uma série de complicações, por exemplo, o tempo de projeto é indefinido, pois depende de uma grande quantidade de testes até encontrar parâmetros aceitáveis, que não necessariamente irão atender a todos os requisitos de projeto. Outro fator

Figura 48 – Resposta (em rad/s) dos motores com controlador PD-Empírico quanto a uma entrada em degrau de 50 rad/s: (a) Motor 1; (b) Motor 2; (c) Motor 3; e (d) Motor 4.



Fonte: o autor.

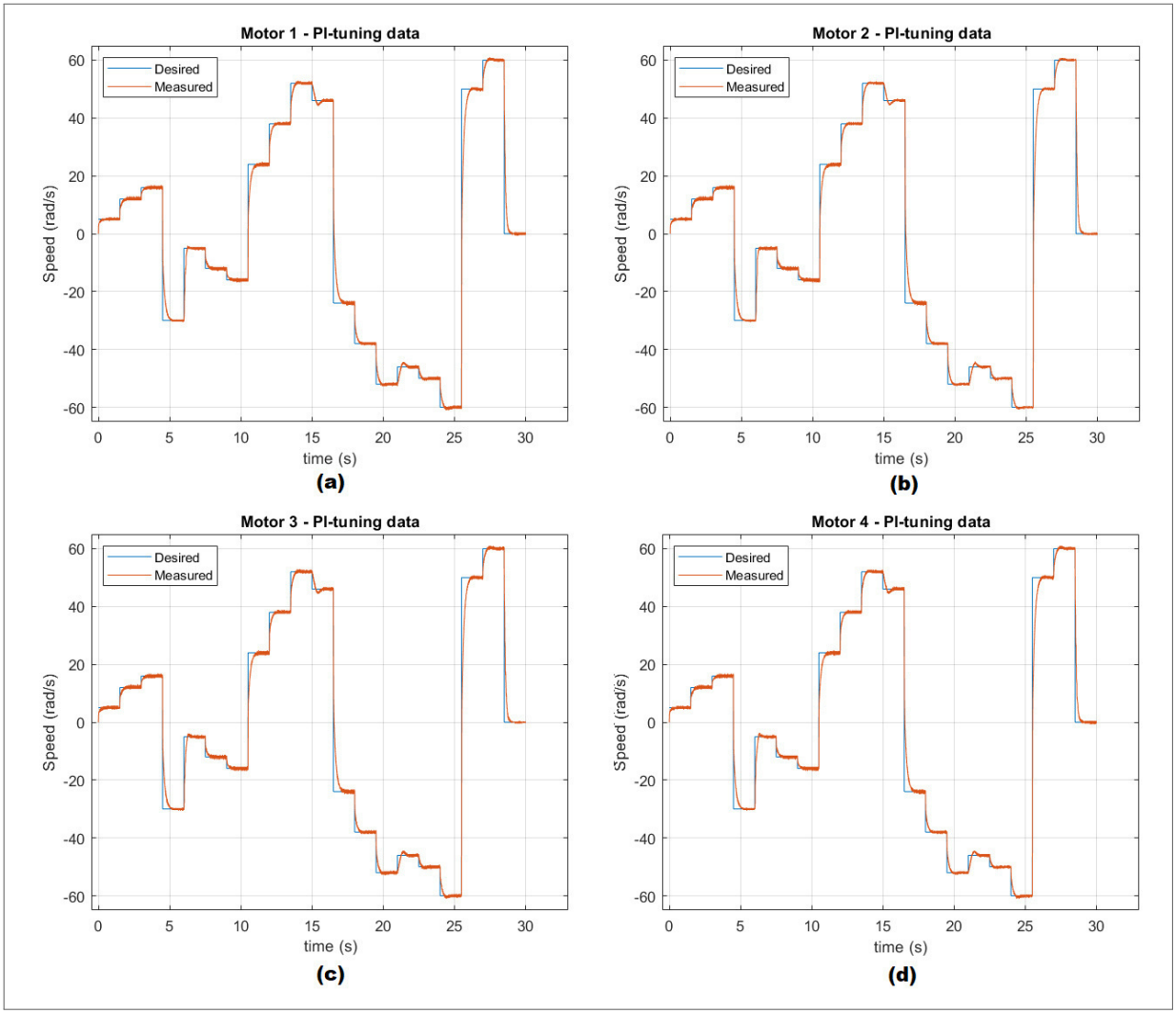
agravante desta técnica é o risco de danificar o circuito ou o próprio motor utilizado caso sejam aplicadas constantes de controle que levem o controlador à instabilidade. Além disso, dependendo do ambiente em que o motor está inserido, pode ser necessário recalibrar as constantes para que o motor se adapte a nova realidade, tornando o processo de tuning manual ainda mais lento.

Para a aplicação deste projeto, o motor deve ser capaz de atingir qualquer velocidade dentro de sua faixa de atuação de acordo com as especificações. Portanto, a segunda parte do experimento de validação proposto na seção 4.5.2 foi executada, para analisar o comportamento do controlador para múltiplos valores de referência ao gerar estímulos de velocidade arbitrários que foram aplicados ao motor por um determinado período, analisando se o controlador implementado funcionava ou não da maneira esperada.

A Figura 49 mostra a resposta dos motores utilizados a um conjunto de vinte velocidades de referência arbitrárias dentro da faixa de operação do motor com a abordagem proposta implementada. O intervalo entre cada uma das vinte velocidades foi de $1,5s$. Para analisar

o quão bem ajustados os resultados reais estão em relação ao valor desejado, foi utilizada a função *GoodnessOfFit* do MATLAB. Dadas duas matrizes, uma contendo valores de referência, e uma contendo valores coletados, a função *GoodnessOfFit* retorna uma representação quantitativa do quão próximos os valores coletados estão dos valores de referência, de acordo com a métrica estabelecida. Para este experimento, os dados foram avaliados de acordo com três métricas: Erro Médio Quadrático (MSE), Raiz Quadrada do Erro Médio Quadrático Normalizado (NRMSE) e Erro Médio Quadrático Normalizado (NMSE). A Tabela 24 mostra os resultados da função *GoodnessOfFit* para os quatro motores testados utilizando a abordagem proposta.

Figura 49 – Resposta (em rad/s) dos motores com controlador PI-tuning quanto a um conjunto de velocidades de entrada para: (a) Motor 1; (b) Motor 2; (c) Motor 3; e (d) Motor 4.



Fonte: o autor.

Tabela 24 – Métricas de avaliação de *fitting* para abordagem PI-tuning.

Parâmetro	Motor 1	Motor 2	Motor 3	Motor 4
MSE	$25,0762$	$25,6459$	$25,3740$	$24,5727$
NRMSE	0.8652	$0,8636$	$0,8644$	$0,8665$
NMSE	0.9818	$0,9814$	$0,9816$	$0,9822$

Fonte: o autor.

O mesmo experimento é executado utilizando a abordagem empírica que implementa um controlador PD. A Figura 50 apresenta a resposta dos motores para o mesmo conjunto de entradas utilizado para validar a abordagem proposta. A Tabela 25 apresenta os resultados da função *goodnessOfFit* para os quatro motores testados utilizando a abordagem empírica.

Ao analisar os experimentos das duas abordagens, percebe-se uma maior oscilação nos resultados apresentados pela abordagem empírica, que parece ser bem mais ruidosa, já na abordagem PI proposta os dados parecem bem mais estáveis. Além disso, diferentemente da abordagem proposta, a abordagem empírica não elimina o erro em regime permanente, que é um dos requisitos de projeto mais importantes. No Apêndice B é visto um fragmento dos experimentos das Figuras 49 e 50 de maneira mais detalhada, para que sejam visualizados o erro em regime permanente da abordagem empírica em várias velocidades, e o ruído existente na abordagem empírica.

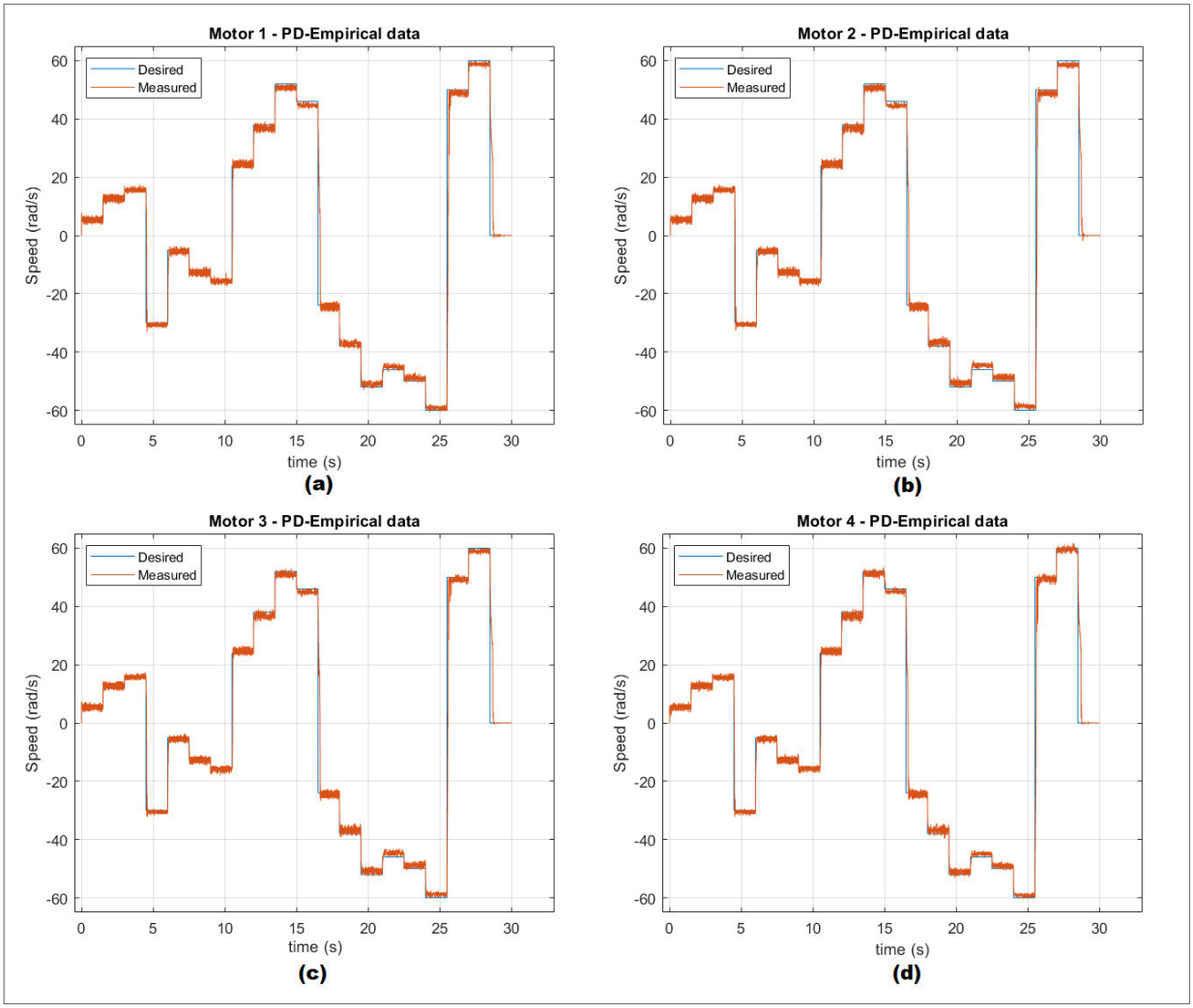
Tabela 25 – Métricas de avaliação de *fitting* para abordagem PD-Empírica.

Parâmetro	Motor 1	Motor 2	Motor 3	Motor 4
MSE	$41,6481$	$42,1681$	$40,8693$	$39,5703$
NRMSE	0.7856	$0,7843$	$0,7876$	$0,7910$
NMSE	0.9540	$0,9535$	$0,9549$	$0,9563$

Fonte: o autor.

Além disso, as métricas de avaliação mostram um desempenho superior da abordagem proposta em comparação com a abordagem empírica para todos os motores e em todos os aspectos.

Figura 50 – Resposta (em rad/s) dos motores com controlador PD com abordagem empírica para um conjunto de velocidades de entrada para: (a) Motor 1; (b) Motor 2; (c) Motor 3; e (d) Motor 4.



Fonte: o autor.

6.3 VALIDAÇÃO DO ESTUDO DE CASO

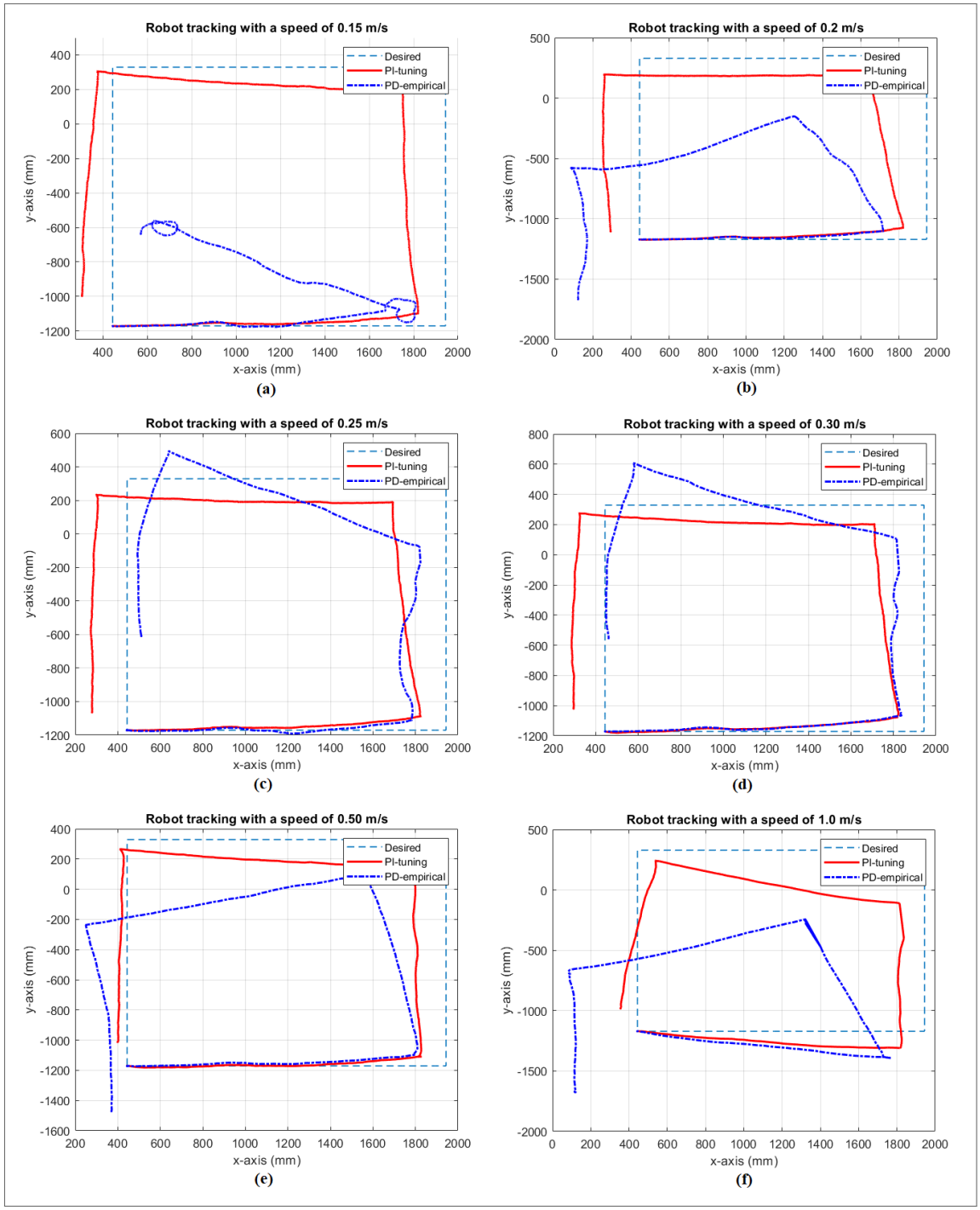
Como pôde ser visto na seção anterior, uma das desvantagens da abordagem empírica foi o fato dela não eliminar o erro em regime permanente. No estudo de caso aplicado, para que o robô possa seguir uma trajetória desejada, é necessário que a contribuição de cada uma das rodas, quando somadas, resultem no movimento para a direção correta. Entretanto, caso algumas das rodas não se movimentem na velocidade calculada pelo modelo cinemático, o resultado é uma trajetória diferente da que foi calculada.

Os experimentos da Seção 5.4 foram executados a fim de analisar o impacto das melhorias do controlador PI proposto em relação ao controlador PD empírico. Como mencionado, o primeiro experimento consiste em fazer o robô executar um quadrado de lado $1,5m$ com diversas velocidades. Ao todo, seis velocidades foram testadas, e os resultados podem ser visualizados na Figura 51. A linha tracejada representa a trajetória ideal, quando os mesmos sinais

de comando são dados a um robô num simulador da categoria. Os resultados da movimentação em ambiente real são obtidos através do sistema de visão compartilhada *SSL-Vision*, que armazena em tempo real a posição do robô no campo, para que os dados de movimentação possam ser analisados pelo software MATLAB para visualização. O segundo experimento consiste em fazer o robô executar uma trajetória circular de $1m$ de diâmetro com diversas velocidades. Para esse caso, quatro velocidades foram testadas, e os resultados podem ser visualizados na Figura 52. Os resultados são capturados e visualizados da mesma forma que o experimento anterior.

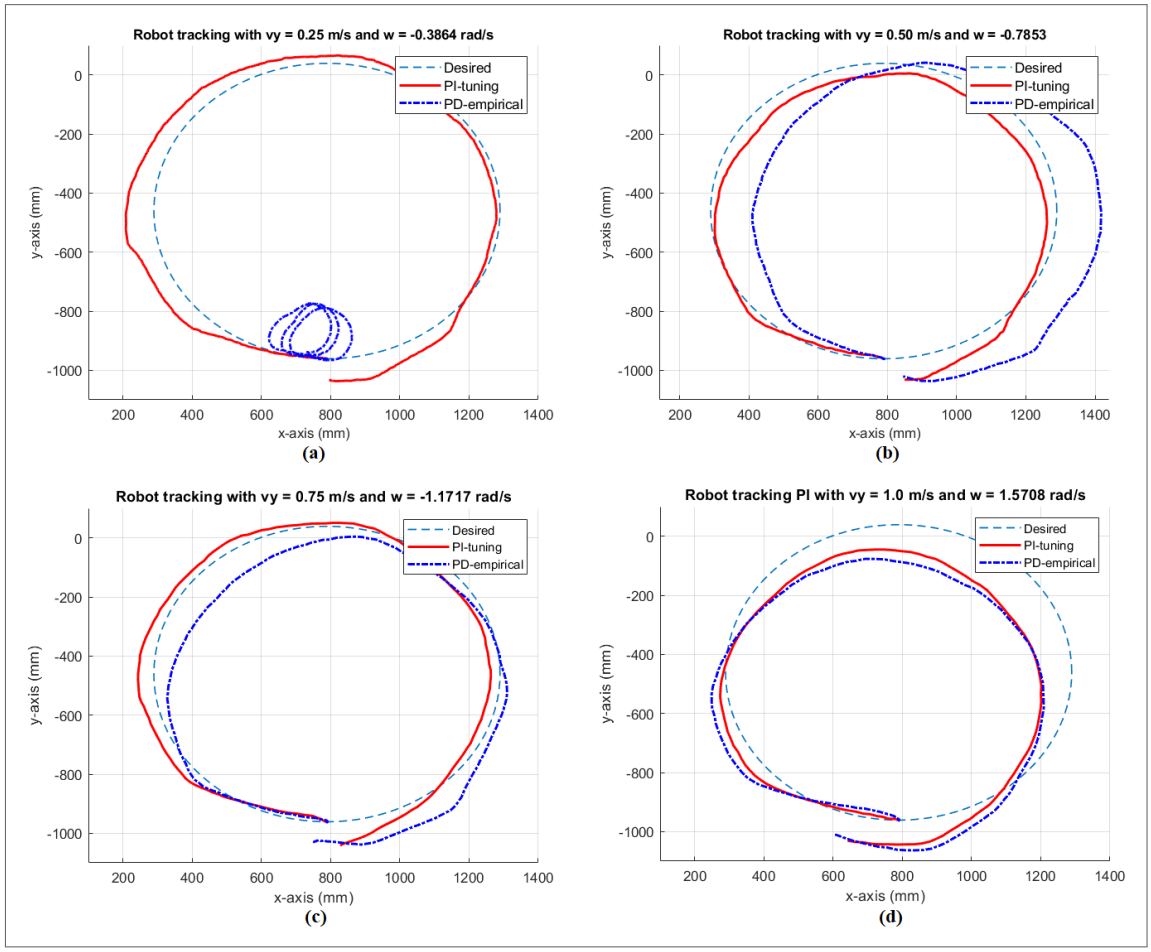
Os resultados mostram uma melhoria significativa na trajetória do robô, principalmente em velocidades menores. Isso se deve ao fato de que o erro em regime permanente em velocidades mais baixas tem um impacto maior na velocidade final do robô como um todo. Outro fato a ser considerado é que as velocidades são aplicadas sem um controle de aceleração do robô, ou seja, as não-linearidades associadas ao atrito do robô com o chão ao sair da inércia influenciam na aceleração inicial, fazendo com que em velocidades maiores, o robô tenha maior dificuldade de acompanhar a trajetória desejada.

Figura 51 – Trajetória do robô ao executar um quadrado de lado $1,5\ m$ para as velocidades (a) $0,15\ m/s$; (b) $0,20\ m/s$; (c) $0,25\ m/s$; (d) $0,30\ m/s$; (e) $0,50\ m/s$; e (f) $1,0\ m/s$.



Fonte: o autor.

Figura 52 – Trajetória do robô ao executar um círculo de diâmetro $1\ m$ para as velocidades (a) vy $= 0,25\ m/s$ e w $= -0.3864\ rad/s$; (b) $0,50\ m/s$ w $= -0,7853\ rad/s$; (c) vy $= 0,75\ m/s$ e w $= -1,1717\ rad/s$; e (d) vy $= 1\ m/s$ e w $= -1,5708\ rad/s$.



Fonte: o autor.

7 CONCLUSÃO

7.1 CONCLUSÃO

Este trabalho apresentou uma abordagem baseada em telemetria para obtenção de um modelo aproximado de um motor e o *tuning* das constantes de controle de um controlador PI que suporta diretamente motores *brushless* DC. Embora para este estudo de caso tenha sido implementado um controlador PI, a abordagem também pode ser utilizada para encontrar controladores proporcional, PD, ou PID. A estratégia proposta inclui a etapa de identificação de parâmetros de funcionamento do motor, como tempo de subida e tempo de assentamento, adicionalmente foi apresentada uma estratégia para a escolha da taxa de amostragem do *encoder*, bem como para a identificação experimental da faixa de operação do motor, ou seja, valores mínimos e máximo de PWM em que o motor funciona da maneira esperada no ambiente em que ele irá atuar.

Foram comparadas duas técnicas para a obtenção da função de transferência que melhor representa o comportamento de um motor: a técnica utilizando sinais PRBS e a técnica utilizando sinais MPRS. Os modelos obtidos utilizando a técnica com sinais PRBS apresentaram entre 63% e 65% de *fitting*, enquanto que os modelos obtidos através da técnica com sinais MPRS apresentaram *fitting* entre 90% e 91%. A função de transferência aproximada pelo método MPRS foi capaz de representar a resposta do motor de maneira coerente com os experimentos reais tanto no conjunto de dados de treinamento quanto no conjunto de dados de validação. Portanto, as funções obtidas pelo método MPRS são mais indicadas para utilização no *tuning* de controladores dos motores *brushless* DC utilizados neste estudo de caso. Para que seja possível generalizar esta afirmação para outros motores *brushless* DC, é necessário repetir o experimento para outros modelos de motores. Em geral, os parâmetros de cada um dos motores testados apresentaram uma similaridade muito alta o que mostra robustez quanto à presença de distúrbios durante o processo aquisição dos dados.

A abordagem proposta pode ser aplicada a qualquer motor *brushless* com *encoder*, e permite a obtenção de um modelo digital "gêmeo"(*digital twin*) de um motor ou de um conjunto de motores atuando de forma integrada, uma vez que os dados para a modelagem digital são obtidos por telemetria (RASHEED; SAN; KVAMSDAL, 2019). Este modelo permite a obtenção de controladores PID otimizados para diferentes ambientes e situações em contextos dinâmicos de forma segura evitando danos ao motor real nas etapas iniciais de projeto.

No caso do *tuning* de motores, a metodologia proposta ainda reduz o tempo de projeto por permitir que as técnicas de otimização sejam realizadas com dados do sistema em funcionamento num ambiente real. A técnica proposta está baseada no software MATLAB, que abstrai a implementação de funções complexas como o algoritmo para estimar a função de transferência, e o algoritmo de tuning das constantes.

O fato da comunicação com o sistema real ser sem fio permite o robô atuar no ambiente de validação real considerando os ruídos e situações reais do ambiente com e sem distúrbios. Uma comparação da abordagem proposta com os trabalhos relacionados está sumarizada na tabela 26.

Alguns aspectos a serem considerados para a implementação da abordagem são a necessidade de ter um ambiente de validação montado, por exemplo, para validação deste estudo de caso é necessário ter robô físico e uma parte do campo de jogo. Mais testes precisam ser feitos para identificar a necessidade de refazer os experimentos caso mudem as condições externas ou de hardware do ambiente de validação, como mudanças no ambiente (campo de jogo), ou substituição de partes do robô.

7.2 TRABALHOS FUTUROS

Duas sugestões de trabalhos futuros são propostas. A primeira consiste em implementar um auto-ajuste das constantes de controle através da telemetria, para que não seja necessário modificar o código embarcado a cada mudança de constantes. A segunda sugestão é utilizar os parâmetros obtidos pela ferramenta *PIDTuner* como parâmetros iniciais de um algoritmo genético que utiliza otimização multi-objetiva para encontrar as constantes de controle ótimas que minimizem o tempo de subida e de assentamento, eliminando o erro em regime permanente e com baixo *overshoot*.

Tabela 26 – Comparativo entre os trabalhos relacionados e a abordagem proposta.

	Suporta Motores Brushless DC	**Apresenta Técnica para Obtenção do Modelo do Motor**	**Obtenção do Modelo do Motor de Maneira Experimental**	**Modelagem em Tempo Discreto**	**Implementação de um Controlador P, PI, PD ou PID**	**Reajuste Automático das Constantes de Controle**	**Validação em Ambiente Real**	**Comunicação sem fio entre Hardware e Computador**
(Oguntoyinbo, 2009)	Sim	Sim	Não	Não	Sim	Não	Não	Não
(Hat et al., 2015)	Sim	Sim	Sim	Não	Sim	Não	Sim	Não
(Tang et al., 2017)	Não	Não	Sim	Sim	Sim	Não	Sim	Não
(Mamun et al., 2018)	Sim	Não	Não	Sim	Não	Não	Não	Não
(Anshory et al., 2019)	Sim	Não	Não	Sim	Sim	Não	Não	Não
(Ibrahim et al., 2019)	Sim	Sim	Não	Não	Sim	Sim	Não	Não
(Araújo, 2020)	Sim	Sim	Sim	Sim	Sim	Não	Sim	Sim

Fonte: o autor.

REFERÊNCIAS

ABBENSETH, N. O. J. **Position Control of an Omnidirectional Mobile Robot**. 2015.

ABU-KHALAF, R. C. M.; TUREVSKIY, A. ***PID Control Design Made Easy***. 2009. Disponível em <http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID>. [Acesso em: 31 de dezembro de 2019].

ANDERSON, B. D.; MOORE, J. B. ***Optimal control: linear quadratic methods***. [S.l.]: Courier Corporation, 2007.

ANSHORY, I.; ROBANDI, I.; OHKI, M. et al. System identification of bldc motor and optimization speed control using artificial intelligent. ***International Journal of Civil Engineering and Technology***, v. 10, n. 7, 2019.

ÅSTRÖM, K. J.; HÄGGLUND, T. ***PID controllers: theory, design, and tuning***. [S.l.]: Instrument society of America Research Triangle Park, NC, 1995. v. 2.

BAEDE, T. Motion control of an omnidirectional mobile robot. ***Traineeship report DCT***, v. 2006, 2006.

BAHRIN, M. A. K.; OTHMAN, M. F.; AZLI, N. H. N.; TALIB, M. F. Industry 4.0: A review on industrial automation and robotic. ***Jurnal Teknologi***, v. 78, n. 6-13, 2016.

BARTELT, T. L. ***Industrial automated systems: instrumentation and motion control***. [S.l.]: Nelson Education, 2010.

BHADANI, A.; KOLADIYA, D.; DEVANI, J.; TAHILIANI, A. Modelling and controlling of bldc motor. ***Development***, v. 3, n. 3, 2016.

BRAUN, M.; RIVERA, D.; STENMAN, A.; FOSLIEN, W.; HRENYA, C. Multi-level pseudo-random signal design and"model-on-demand"estimation applied to nonlinear identification of a rtp wafer reactor. In: IEEE. ***Proceedings of the 1999 American Control Conference***. [S.l.], 1999. v. 3, p. 1573–1577.

BRUCE, J.; VELOSO, M. Real-time randomized path planning for robot navigation. In: IEEE. ***IEEE/RSJ international conference on intelligent robots and systems***. [S.l.], 2002. v. 3, p. 2383–2388.

BURKHARD, H.-D.; DUHAUT, D.; FUJITA, M.; LIMA, P.; MURPHY, R.; ROJAS, R. The road to robocup 2050. ***IEEE Robotics & Automation Magazine***, IEEE, v. 9, n. 2, p. 31–38, 2002.

CAMPION, G.; BASTIN, G.; DANDREA-NOVEL, B. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. ***IEEE transactions on robotics and automation***, IEEE, v. 12, n. 1, p. 47–62, 1996.

CAVALCANTI, L. ***Optimized Real Time Radio Frequency Network for Multiple Mobile Robots Communication***. 2019. Trabalho de Graduação. Centro de Informática, UFPE.

COCHRAN, W. G. **Sampling techniques**. New York, NY (USA) Wiley, 1977.

COMMITEE, R. S. T. ***Rules of the RoboCup Small Size League 2019***. 2019. Disponível em <https://robocup-ssl.github.io/ssl-rules/sslrules.pdf>. [Acesso em: 31 de dezembro de 2019].

DONGALE, T.; JADHAV, S.; KULKARNI, S.; KULKARNI, T.; MUDHOLKAR, R.; UPLANE, M. Performance comparison of pid and fuzzy control techniques in three phase induction motor control. ***International Journal on Recent Trends in Engineering and Technology***, v. 7, n. 2, 2012.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. ***Feedback control of dynamic systems***. [S.l.]: Prentice Hall Press, 2014.

GOODWIN, G. C.; GRAEBE, S. F.; SALGADO, M. E. et al. ***Control system design***. [S.l.]: Prentice Hall New Jersey, 2001. v. 240.

GÜLERMAN, E. ***Advanced Throttle Control Hardware Implementation***. 2014.

HAIDEKKER, M. A. ***Linear feedback controls: the essentials***. [S.l.]: Newnes, 2013.

HALLIDAY, D.; RESNICK, R.; WALKER, J. ***Fundamentos de Física, vol. 1: mecânica***. [S.l.]: LTC. 9ª, 2012.

HASHEMI, E.; JADIDI, M. G.; JADIDI, N. G. Model-based pi–fuzzy control of four-wheeled omni-directional mobile robots. ***Robotics and Autonomous Systems***, Elsevier, v. 59, n. 11, p. 930–942, 2011.

HAT, M.; IBRAHIM, B.; MOHD, T.; HASSAN, M. Model based design of pid controller for bldc motor with implementation of embedded arduino mega controller. ***ARPM Journal of Engineering and Applied Sciences***, 2015.

HUSSIN, M.; AZUWIR, M.; ZAIAZMIN, Y. Modeling and validation of brushless dc motor. In: IEEE. ***2011 Fourth International Conference on Modeling, Simulation and Applied Optimization***. [S.l.], 2011. p. 1–4.

IBRAHIM, M. A.; MAHMOOD, A. K.; SULTAN, N. S. Optimal pid controller of a brushless dc motor using genetic algorithm. ***International Journal on Power Electronics & Driver Systems***, v. 2088, n. 8694, p. 8694, 2019.

INSTITUTE, T. F. T. ***2019 Tech Trends Report***. 2019. Disponível em <https://futuretodayinstitute.com/2019-tech-trends/>. [Acesso em: 31 de dezembro de 2019].

JEWETT, J. W.; SERWAY, R. A. ***Physics for scientists and engineers with modern physics***. [S.l.]: Cengage Learning EMEA, 2008.

JOLLY, K.; KUMAR, R. S.; VIJAYAKUMAR, R. A bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. ***Robotics and Autonomous Systems***, Elsevier, v. 57, n. 1, p. 23–33, 2009.

KATAYAMA, T.; OHKI, T.; INOUE, T.; KATO, T. Design of an optimal controller for a discrete-time system subject to previewable demand. ***International Journal of Control***, Taylor & Francis, v. 41, n. 3, p. 677–699, 1985.

KEESMAN, K. J. ***System identification: an introduction***. [S.l.]: Springer Science & Business Media, 2011.

KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. In: ***Autonomous robot vehicles***. [S.l.]: Springer, 1986. p. 396–404.

KITANO, H.; ASADA, M. The robocup humanoid challenge as the millennium challenge for advanced robotics. ***Advanced Robotics***, Taylor & Francis, v. 13, n. 8, p. 723–736, 1998.

KITANO, H.; ASADA, M.; KUNIYOSHI, Y.; NODA, I.; OSAWA, E. **Robocup: The robot world cup initiative**. Citeseer, 1995.

KRAMER, J.; SCHEUTZ, M. Development environments for autonomous mobile robots: A survey. ***Autonomous Robots***, Springer, v. 22, n. 2, p. 101–132, 2007.

KRKLJEŠ, D.; MORVAI, Č.; BABKOVIĆ, K.; NAGY, L. Bldc motor driver—development of control and power electronics. In: IEEE. ***2010 27th International Conference on Microelectronics Proceedings***. [S.l.], 2010. p. 345–348.

LJUNG, L. ***System identification toolbox: User's guide***. 2nd. ed. [S.l.]: MathWorks Incorporated, 1995.

LOZANO-PÉREZ, T.; WESLEY, M. A. An algorithm for planning collision-free paths among polyhedral obstacles. ***Communications of the ACM***, ACM, v. 22, n. 10, p. 560–570, 1979.

MAMUN, M. A. A.; NASIR, M. T.; KHAYYAT, A. Embedded system for motion control of an omnidirectional mobile robot. ***IEEE Access***, IEEE, v. 6, p. 6722–6739, 2018.

MATHWORKS. ***Polynomial Curve Fitting, MATLAB polyfit***. 2019. Disponível em <https://www.mathworks.com/help/matlab/ref/polyfit.html>. [Acesso em: 31 de dezembro de 2019].

MATWORKS. ***Estimate Output-Error polynomial model using time or frequency domain data***. 2019. Disponível em <https://www.mathworks.com/help/ident/ref/oe.html>. [Acesso em: 1 de janeiro de 2020].

MAXON. ***EC-45 Flat 50W Datasheet***. 2019. Disponível em <https://www.maxongroup.com/medias/sys_master/root/8833813119006/19-EN-263.pdf>. [Acesso em: 31 de dezembro de 2019].

MERIAM, J. L.; KRAIGE, L. G. ***Engineering mechanics: dynamics***. [S.l.]: John Wiley & Sons, 2012. v. 2.

MICROSYSTEMS, A. **A3930 and A3931 Automotive 3-Phase BLDC Controller and MOSFET Driver**. 2019.

NIKU, S. B. ***Introduction to robotics: analysis, systems, applications***. [S.l.]: Prentice hall New Jersey, 2001. v. 7.

NISE, N. S. ***Control Systems Engineering***. [S.l.]: John Wiley & Sons, 2007.

OGATA, K. ***Engenharia de controle moderno***. [S.l.: s.n.], 2003.

OGUNTOYINBO, O. et al. **PID control of brushless DC motor and robot trajectory planning simulation with MATLAB®/SIMULINK®**. Vaasan ammattikorkeakoulu, 2009.

PATEL, V. K. S.; PANDEY, A. Modeling and performance analysis of pid controlled bldc motor and different schemes of pwm controlled bldc motor. ***International Journal of Scientific and Research Publications***, Citeseer, v. 3, n. 4, p. 1–14, 2013.

RASHEED, A.; SAN, O.; KVAMSDAL, T. Digital twin: Values, challenges and enablers. ***arXiv preprint arXiv:1910.01719***, 2019.

ROJAS, R.; FÖRSTER, A. G. Holonomic control of a robot with an omnidirectional drive. ***KI-Künstliche Intelligenz***, v. 20, n. 2, p. 12–17, 2006.

ROMERO, R. A. F.; JUNIOR, E. P. e. S.; OSóRIO, F. S.; WOLF, D. F. ***Robótica móvel***. [S.l.]: LTC, 2014.

SCHREIBER, A.; ISERMANN, R. Identification methods for experimental nonlinear modeling of combustion engines. ***IFAC Proceedings Volumes***, Elsevier, v. 40, n. 10, p. 351–357, 2007.

SCHWAB, K. ***The fourth industrial revolution***. [S.l.]: Currency, 2017.

SEMICONDUCTOR, A. N. ***nRF24L01 product specification***. 2008.

SICILIANO, B.; SCIAVICCO, L.; VILLANI, L.; ORIOLO, G. ***Robotics: modelling, planning and control***. [S.l.]: Springer Science & Business Media, 2010.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. ***Introduction to Autonomous Mobile Robots***. 2nd. ed. [S.l.]: The MIT Press, 2011. ISBN 0262015358, 9780262015356.

SILVA, C.; MARTINS, F.; MACHADO, J. G.; CAVALCANTI, L.; SOUSA, R.; FERNANDES, R.; ARAÚJO, V.; BARROS, E.; BASSANI, H. F. et al. **RobôCIn 2020 Team Description Paper**. 2020.

SINGH, S. K.; KATAL, N.; MODANI, S. Optimization of pid controller for brushless dc motor by using bio-inspired algorithms. ***Research Journal of Applied Sciences, Engineering and Technology***, Maxwell Science Publishing, v. 7, n. 7, p. 1302–1308, 2014.

STMICROELECTRONICS. **STM32 Nucleo-144 Boards User Manual**. 2017.

STMICROELECTRONICS. **RM0410 Reference Manual STM32F76xxx and STM32F77xxx Advanced Arm®-Based 32-bit MCUs**. 2019.

TAN, A. H.; GODFREY, K. R. The generation of binary and near-binary pseudo-random signals: An overview. In: IEEE. ***IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference***. [S.l.], 2001. v. 2, p. 766–771.

TANG, W.-J.; LIU, Z.-T.; WANG, Q. Dc motor speed control based on system identification and pid auto tuning. In: IEEE. ***2017 36th Chinese Control Conference (CCC)***. [S.l.], 2017. p. 6420–6423.

TANGIRALA, A. K. ***Principles of System Identification: theory and practice***. [S.l.]: CRC Press, 2014. v. 1.

TASHAKORI, A.; HASSANUDEEN, M.; EKTESABI, M. Fpga based controller drive of bldc motor using digital pwm technique. In: IEEE. ***2015 IEEE 11th International Conference on Power Electronics and Drive Systems***. [S.l.], 2015. p. 658–662.

THAM, M. Discretised pid controllers. ***Chemical Engineering and Advanced Materials, University of Newcastle Upon Tyne***, v. 1998, 1996.

TUTORIALS, C. ***Introduction: PID Controller Design***. 2019. Disponível em <http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID>. [Acesso em: 31 de dezembro de 2019].

VAHID, F.; GIVARGIS, T. ***Embedded system design: A unified hardware/software approach***. [S.l.]: Department of Computer Science and Engineering University of California, 1999.

WEITZENFELD, A.; BISWAS, J.; AKAR, M.; SUKVICHAI, K. Robocup small-size league: Past, present and future. In: SPRINGER. ***Robot Soccer World Cup***. [S.l.], 2014. p. 611–623.

WILSON, C. E.; SADLER, J. P.; MICHELS, W. J. ***Kinematics and dynamics of machinery***. [S.l.]: Pearson Education Upper Saddle River, NJ, 2003. v. 2.

XIA, C.-l. ***Permanent magnet brushless DC motor drives and controls***. [S.l.]: John Wiley & Sons, 2012.

YADAV, V.; TAYAL, V. K. Optimal controller design for a dc motor using pid tuner. In: IEEE. ***2018 International Conference on Power Energy, Environment and Intelligent Control (PEEIC)***. [S.l.], 2018. p. 442–445.

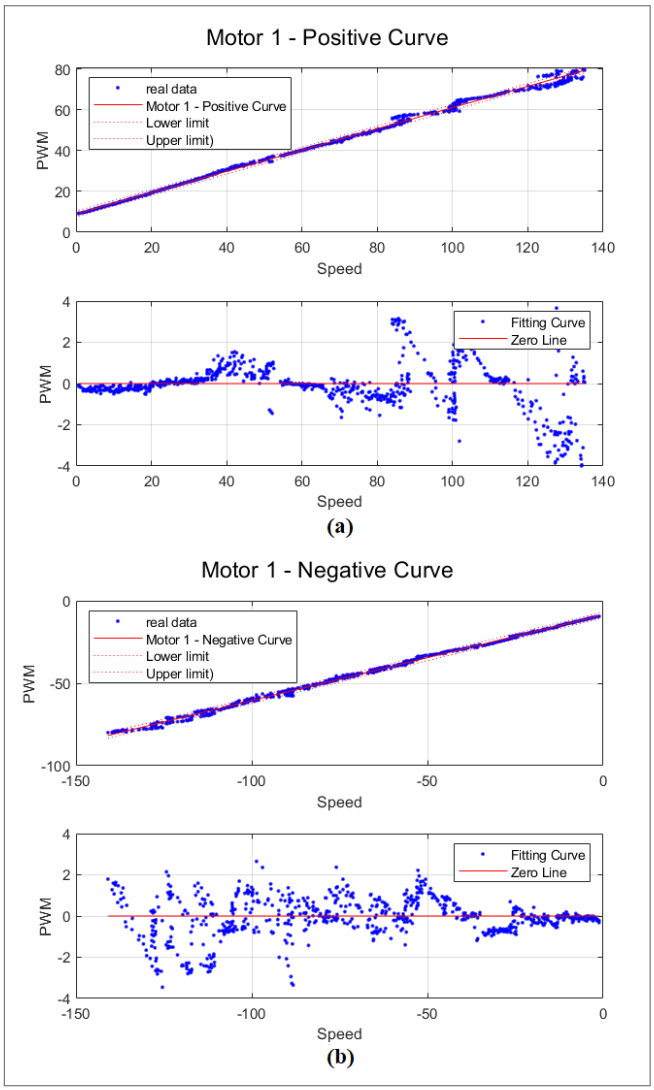
YEDALE, P. **Brushless DC Motor Fundamental**. *AN885, Microchip Technology Inc*, 2003.

ZICKLER, S.; LAUE, T.; BIRBACH, O.; WONGPHATI, M.; VELOSO, M. Ssl-vision: The shared vision system for the robocup small size league. In: SPRINGER. ***Robot Soccer World Cup***. [S.l.], 2009. p. 425–436.

APÊNDICE A – ANÁLISE RESIDUAL DAS CURVAS DE RESPOSTA DOS MOTORES

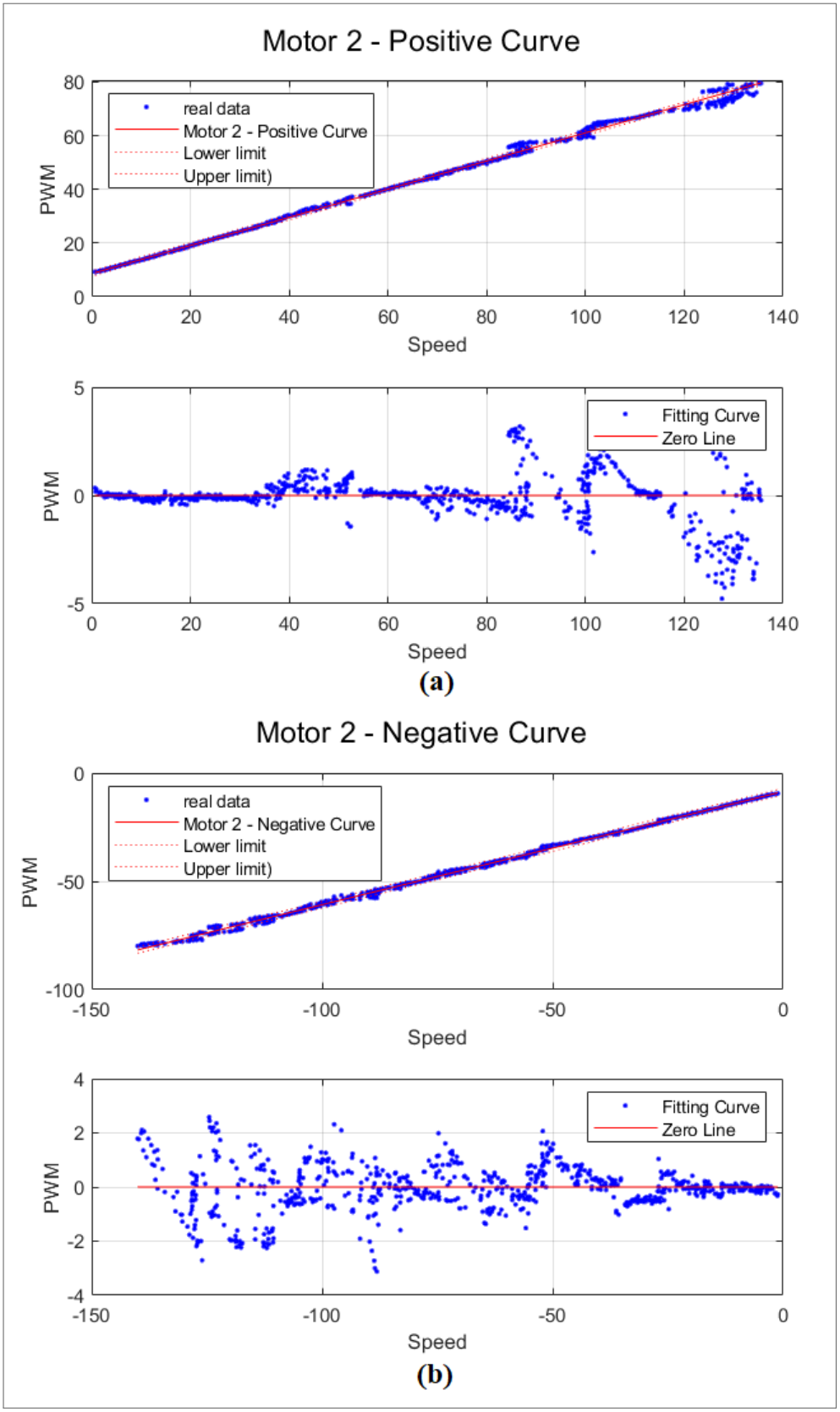
O objetivo da análise residual na curva de resposta dos motores é identificar os ruídos existentes no *encoder* do motor a altas velocidades, tanto para a curva positiva quanto para a negativa. As Figuras 53, 54, 55 e 56 ilustram a presença de um maior ruído em rotações maiores do motor utilizado, isso mostra que será mais difícil controlar o motor em velocidades altas em relação a velocidades menores.

Figura 53 – Análise residual da resposta do Motor 1 para as curvas (a) positiva e (b) negativa.



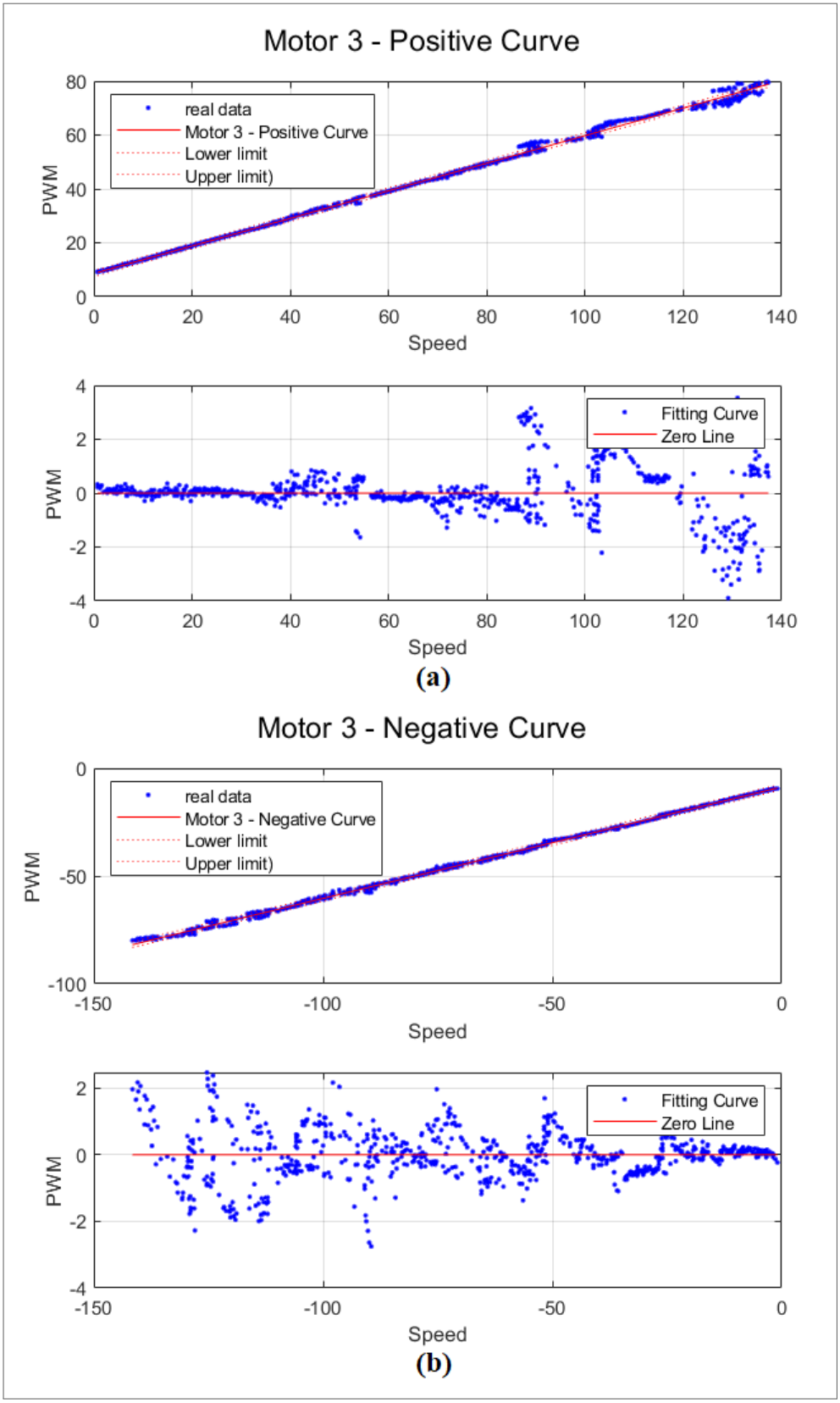
Fonte: o autor.

Figura 54 – Análise residual da resposta do Motor 2 para as curvas (a) positiva e (b) negativa.



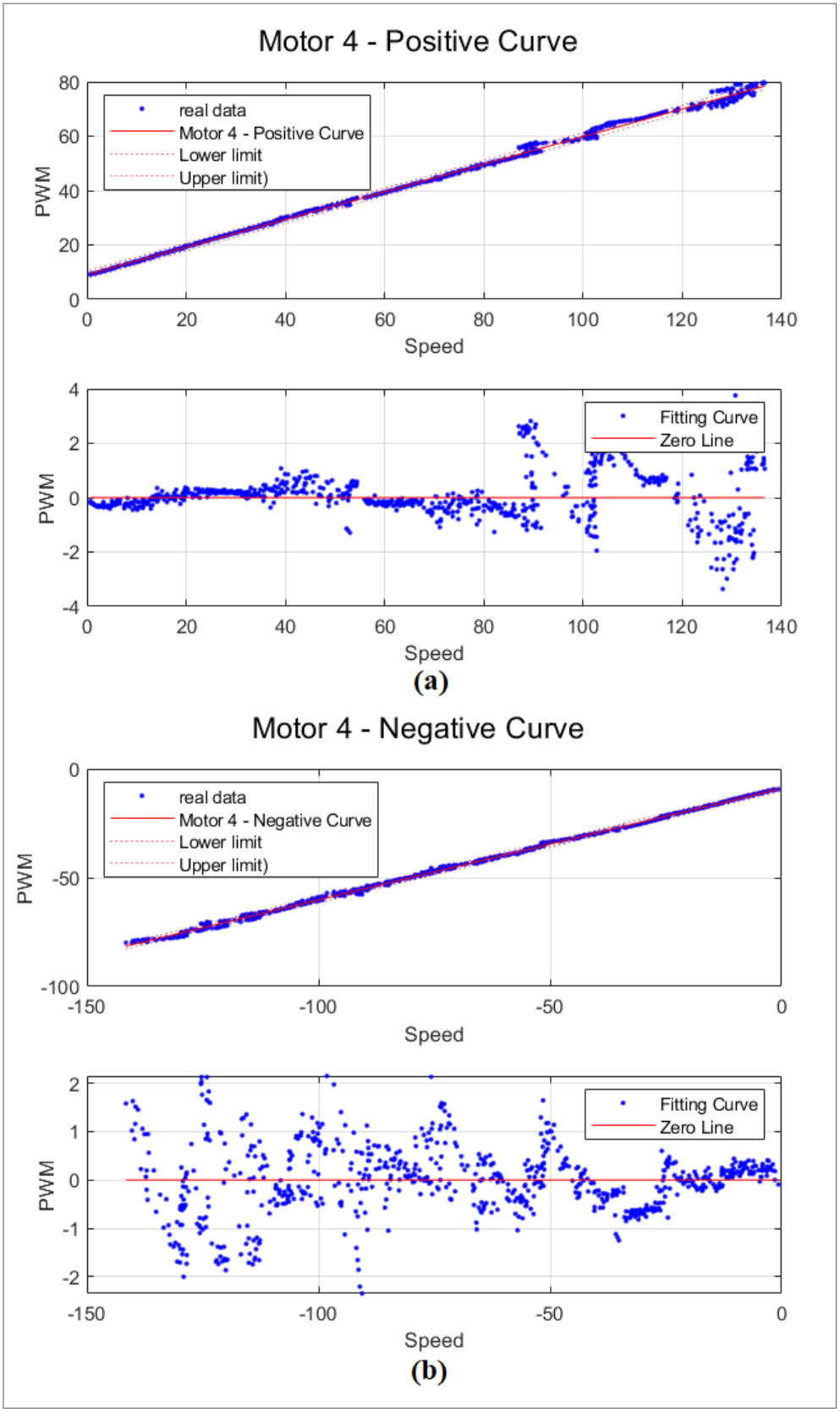
Fonte: o autor.

Figura 55 – Análise residual da resposta do Motor 3 para as curvas (a) positiva e (b) negativa.



Fonte: o autor.

Figura 56 – Análise residual da resposta do Motor 4 para as curvas (a) positiva e (b) negativa.

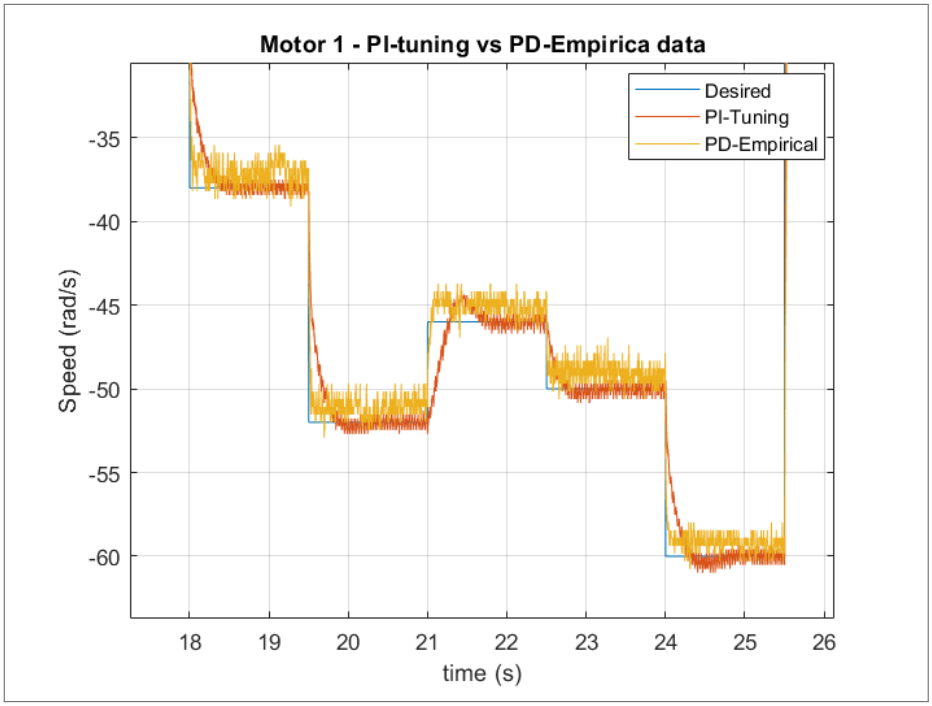


Fonte: o autor.

APÊNDICE B – VISUALIZAÇÃO DETALHADA DA RESPOSTA DOS MOTORES PARA AS ABORDAGENS TESTADAS

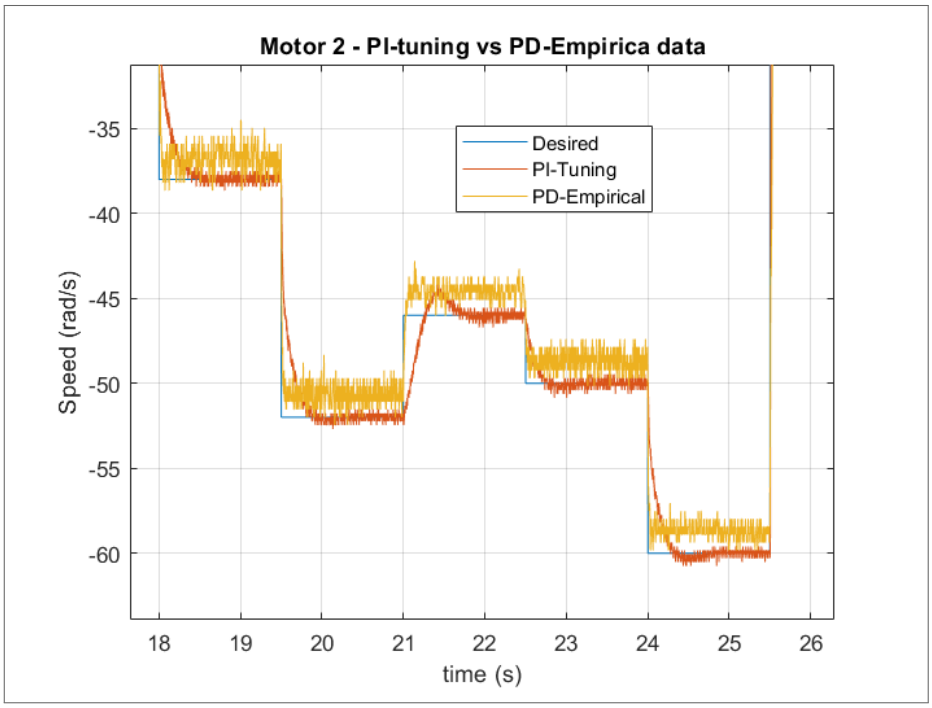
Uma visualização detalhada da resposta de múltiplas velocidades de referência dos motores permite entender melhor as diferenças entre a abordagem proposta e a abordagem empírica, tanto em termos de erro em regime permanente, quanto em ruído do sinal obtido pelo *encoder*. As Figuras 57, 58, 59 e 60 mostram um recorte mais detalhado de um fragmento das figuras 49 e 50 com o objetivo de auxiliar a visualização dos detalhes descritos na Seção 6.2.

Figura 57 – Visualização em Detalhes de um Fragmento da Resposta a Múltiplas Velocidades de Referência do Motor 1.



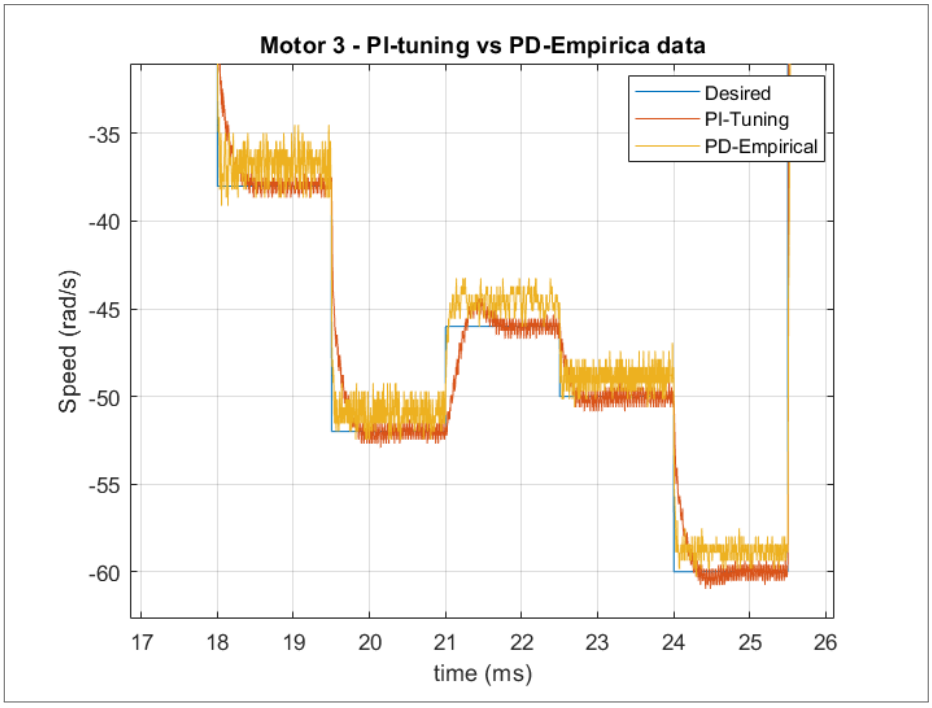
Fonte: o autor.

Figura 58 – Visualização em Detalhes de um Fragmento da Resposta a Múltiplas Velocidades de Referência do Motor 2.



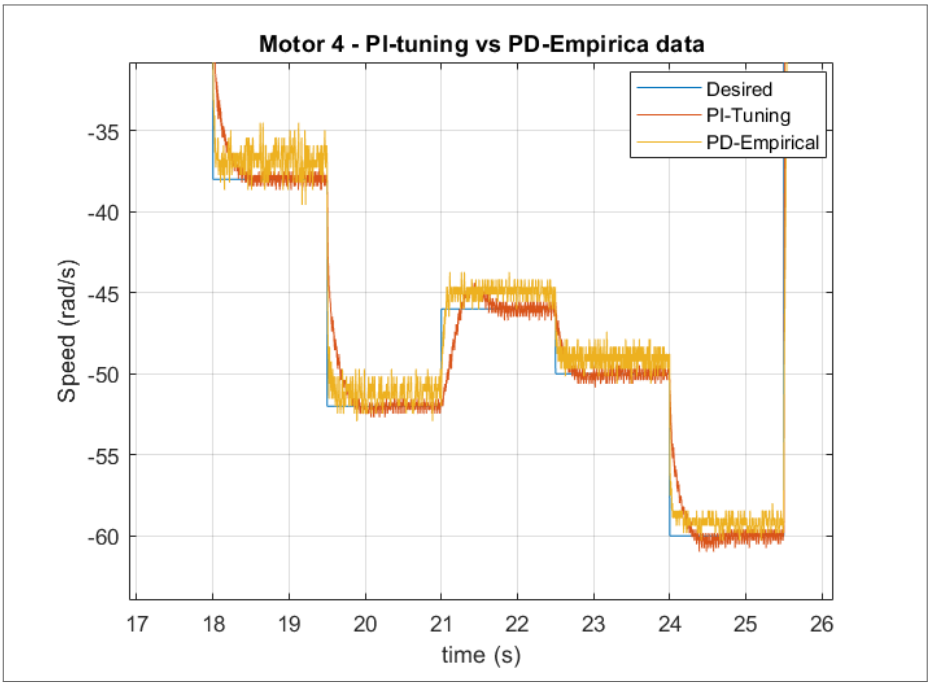
Fonte: o autor.

Figura 59 – Visualização em Detalhes de um Fragmento da Resposta a Múltiplas Velocidades de Referência do Motor 3.



Fonte: o autor.

Figura 60 – Visualização em Detalhes de um Fragmento da Resposta a Múltiplas Velocidades de Referência do Motor 4.



Fonte: o autor.