Universidade de São Paulo
Escola de Engenharia de São Carlos

METODOLOGIA DE TESTES PARA DISPOSITIVO DE RECEBIMENTO DO SUBSISTEMA ATUADOR BASEADO EM MOTOR DC

por

Igor Mayer Soares

Dr. Mateus Moreira de Souza

Orientador

São Carlos

2025

IGOR MAYER SOARES

Metodologia de Testes para Dispositivo de Recebimento do Subsistema Atuador Baseado em Motor DC

Monografia apresentada ao Curso de Especialização em Sistemas Aeronáuticos da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Especialista em Sistemas Aeronáuticos.

Orientador: Dr. Mateus Moreira.

São Carlos
2025

Ficha catalográfica elaborada pela Biblioteca Prof. Sérgio Rodrigues Fontes
e pelo Serviço de Comunicação e Marketing da EESC-USP,
com dados inseridos pelo(a) autor(a).

S676m

Soares, Igor Mayer

Metodologia de Testes para Dispositivo de Recebimento do Subsistema Atuador Baseado em Motor DC / Igor Mayer Soares ; orientador Mateus Moreira de Souza. -- São Carlos, 2025.

74 p.

Trabalho de Conclusão de Curso - Especialização em Sistemas Aeronáuticos -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2025.

1. MBSE; Identificação de Sistemas; Motor DC; Controle PID; Simulação de Monte Carlo.. I. Moreira de Souza, Mateus, orient. II. Título.

Responsáveis pela estrutura de catalogação da publicação segundo a AACR2: Bibliotecários da EESC/USP.

FOLHA DE APROVAÇÃO

Candidato(a): Igor Mayer Soares

Título do Trabalho: Metodologia de testes para dispositivo de recebimento do subsistema atuador baseado em motor DC

Data da defesa: **10/12/2025**

Comissão julgadora

Avaliador	**Resultado (nota)**
Mateus Moreira de Souza (orientador)	10,00
Jorge Henrique Bidinotto	10,00

Resultado final: **Aprovado**

Coordenador do Curso de Especialização em Sistemas Aeronáuticos

Prof. Associado **Jorge Henrique Bidinotto**

Vice-coordenador do Curso de Especialização em Sistemas Aeronáuticos

Prof. Associado **Hernán Darío Cerón Muñoz**

Aos meus amigos da SIATT que fizeram desta Pós-Graduação uma jornada inesquecível, e a própria SIATT por proporcionar todos os meios de concluí-la.

Agradecimentos

Primeiramente, agradeço a Olorum pelo sopro da vida e pela oportunidade de trilhar mais esta etapa em minha jornada acadêmica e profissional.

Aos meus pais, José Eleutério Filho e Edneia Maria Mayer, expresso minha profunda gratidão por todo o amor, incentivo e apoio incondicional, que sempre me impulsionaram a perseguir meus objetivos.

À minha amada noiva, Letícia Rosa Amaral, agradeço pela paciência (muita), compreensão e presença constante durante todo o desenvolvimento deste trabalho. Sua dedicação e carinho foram fundamentais para que eu chegasse até aqui.

Aos meus amigos da SIATT, em especial ao Bruno Marcondes, agradeço pelo companheirismo e pelas discussões técnicas que tanto contribuíram para meu crescimento profissional.

Ao meu orientador, Prof. Dr. Mateus Moreira de Souza, registro meu sincero agradecimento pela orientação, pela paciência e pela confiança depositada ao longo do processo.

Por fim, agradeço à SIATT, representada por Daniel Cerignoni e Wagner Amaral, pela oportunidade de crescimento profissional e pessoal, bem como pelo suporte e pelos meios disponibilizados para que eu pudesse concluir esta Pós-Graduação.

"Pode se encontrar a felicidade mesmo nas horas mais sombrias
se a pessoa se lembrar de acender a luz."
— Alvo Dumbledore

Resumo

Este trabalho apresenta uma metodologia integrada para identificação paramétrica, validação e projeto de controladores aplicada a um motor de corrente contínua (DC). A identificação dos parâmetros elétricos e mecânicos do motor foi realizada a partir de dados gerados por meio de uma Simulação de Monte Carlo, permitindo a avaliação da sensibilidade do modelo frente a variações paramétricas típicas de sistemas eletromecânicos reais. Os parâmetros estimados apresentaram erros percentuais reduzidos, evidenciando a eficácia do procedimento de identificação. A etapa de validação quantitativa empregou as métricas *Root Mean Square Error* (RMSE) e *Goodness of Fit* (FIT), cujos resultados confirmaram a elevada precisão do modelo identificado, com valores de FIT superiores a 96% e RMSE relativo inferior a 4% para todas as variáveis analisadas. Com o modelo validado, procedeu-se ao projeto de um controlador do tipo PID utilizando um método de otimização baseado na minimização de uma função custo multidimensional. O controlador obtido atendeu plenamente às especificações de desempenho no modelo nominal. Entretanto, a avaliação de robustez por meio de novas simulações de Monte Carlo revelou limitações do controlador frente a variações paramétricas mais acentuadas. Uma nova sintonia, baseada no ajuste da estimativa inicial do algoritmo de otimização, resultou em um conjunto de ganhos significativamente mais robusto, capaz de manter o desempenho desejado em cenários amplificados de incerteza. Os resultados obtidos demonstram que a combinação entre identificação paramétrica, validação estatística e síntese de controladores constitui uma abordagem eficaz para o desenvolvimento de sistemas de controle aplicados a motores DC. Este trabalho disponibiliza ainda os códigos embarcados e de simulação utilizados, de modo a permitir a reprodutibilidade dos experimentos e fomentar estudos subsequentes no tema.

Palavras-chave: MBSE; Identificação de Sistemas; Motor DC; Controle PID; Simulação de Monte Carlo.

Abstract

This work presents an integrated methodology for parameter identification, model validation, and controller design applied to a direct current (DC) motor. The electrical and mechanical parameters of the motor were identified using data generated through Monte Carlo Simulation, enabling an assessment of model sensitivity to parametric variations typically found in real electromechanical systems. The estimated parameters exhibited low percentage errors, demonstrating the effectiveness of the identification procedure. The validation stage employed quantitative metrics such as the Root Mean Square Error (RMSE) and the Goodness of Fit (FIT). The results corroborated the high accuracy of the identified model, with FIT values above 96% and relative RMSE values below 4% for all evaluated variables. With the validated model, a Proportional–Integral–Derivative (PID) controller was designed using an optimization-based tuning strategy formulated through the minimization of a multidimensional cost function. Although the nominal model performance satisfied all predefined requirements, robustness analysis via Monte Carlo simulations revealed performance degradation under stronger parametric uncertainties. A retuning step, involving manually adjusted initial conditions for the optimization algorithm, produced a significantly more robust controller capable of maintaining adequate performance across a broader range of operating scenarios. The results demonstrate that the combination of parameter identification, statistical validation, and controller optimization constitutes an effective approach for the development of control systems for DC motors. Additionally, this work provides the embedded and simulation code used throughout the study, enabling reproducibility and encouraging future research developments in this area.

Keywords: MBSE; System Identification; DC Motor; PID Control; Monte Carlo Simulation.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

MBSE	Model-Based Systems Engineering
MIMO	multiple input multiple output
OE	Output Error
PD	proporcional derivativo
PID	proporcional integrativo derivativo
PTP	point to point
RMSE	Root Mean Square Error
SI	Sistema Internacional de Unidades
SysML	Systems Modeling Language

Lista de Símbolos

A	Matriz de estados
B	Matriz de entrada
B_m	Coeficiente de atrito viscoso do motor
$e(t)$	Erro de controle
I	Corrente elétrica no enrolamento do motor
J_m	Momento de inércia do motor
K_e	Constante de força contraeletromotriz
K_t	Constante de torque do motor
K_p	Ganho proporcional do controlador PID
K_i	Ganho integral do controlador PID
K_d	Ganho derivativo do controlador PID
R	Resistência elétrica do enrolamento
T_m	Torque eletromagnético gerado pelo motor
$u(t)$	Sinal de controle aplicado ao motor
$v(t)$	Tensão aplicada ao motor
$\omega(t)$	Velocidade angular do eixo
$\theta(t)$	Posição angular do eixo
τ	Constante de tempo mecânica ($\tau = \frac{J_m}{B_m}$)
$\hat{x}$	Estimativa do estado
$y(t)$	Variável medida (saída do sistema)
$Z(t)$	Sinal de referência/verdade de solo
RMSE	Root Mean Square Error
FIT	Índice de qualidade de ajuste

Sumário

1 Introdução

1.1 Contextualização e Motivação

A Engenharia de Sistemas Baseada em Modelos (MBSE) é uma metodologia que utiliza modelos para suportar todo o ciclo de vida de um produto ou sistema, desde sua concepção até a fase de desativação do produto. Além disso, o MBSE oferece diversas vantagens em relação às abordagens tradicionais. Ele melhora a comunicação e a colaboração entre os stakeholders, aumenta a qualidade e a precisão ao reduzir erros e inconsistências, e proporciona eficiência ao acelerar a identificação e correção de problemas, reduzindo assim custos e riscos.

O fluxo típico de desenvolvimento em uma abordagem MBSE segue as seguintes etapas:

1. Captura de Requisitos;
2. Proposta de Arquitetura;
3. Modelagem Dinâmica;
4. Análise e Simulação;
5. Verificação e Validação;
6. Implementação;

Esse fluxo pode ser aplicado tanto ao sistema de interesse como um todo quanto ao desenvolvimento de seus subsistemas individuais, os quais, especialmente na indústria aeroespacial, frequentemente apresentam níveis de complexidade comparáveis aos do produto final.

Nesse contexto, a etapa de Verificação e Validação (V&V) exerce um papel fundamental: garantir que os subsistemas entregues atendam aos requisitos do projeto. Essa etapa também possibilita a realimentação dos modelos utilizados nas fases anteriores, contribuindo para o refinamento contínuo da solução até sua consolidação.

No setor aeroespacial, é comum a terceirização de diversos subsistemas, o que torna essencial o comprometimento de todos os stakeholders com a qualidade do produto final. Embora os fornecedores atuem como parceiros no processo de desenvolvimento, nem sempre é conveniente, ou desejável, que tenham acesso a soluções internas do fabricante. Nesses casos, uma das estratégias adotadas para assegurar a qualidade do que é fornecido consiste no uso de dispositivos de teste funcional nos ensaios de recebimento. Tais dispositivos são desenvolvidos internamente, com base em critérios claros e bem definidos sobre o desempenho esperado de cada subsistema.

A crescente complexidade dos sistemas aeroespaciais e a dependência de subsistemas fornecidos por terceiros exigem métodos de Verificação e Validação (V&V) que sejam robustos, eficientes e rastreáveis. O presente trabalho se justifica pela necessidade de estabelecer um processo de teste funcional que minimize o risco de integração e maximize a confiança na qualidade do componente.

A aplicação da abordagem MBSE neste contexto é crucial, pois permite:

1. **Rastreabilidade:** Conectar diretamente os requisitos do sistema aos critérios de teste do dispositivo funcional, garantindo que o teste seja relevante.
2. **Simulação e Análise de Incertezas:** Integrar a modelagem do motor DC e a simulação de Monte Carlo para avaliar a robustez do subsistema Atuador frente a variações de parâmetros (tolerâncias de fabricação, envelhecimento) e condições operacionais.
3. **Otimização do Controle:** Utilizar técnicas avançadas de identificação de sistemas (como o sinal Chirp e o Método de Erro de Saída) para obter um modelo preciso do motor DC, essencial para o projeto de um controlador PID de alto desempenho.

Portanto, este estudo contribui para a área de Engenharia de Sistemas ao demonstrar uma metodologia integrada, baseada em modelos, para o desenvolvimento de bancadas de teste críticas, elevando o padrão de V&V em ambientes de alta exigência.

1.2 Subsistema Atuador

Um dos subsistemas mais críticos de uma aeronave é o subsistema de atuação, responsável pelo controle das superfícies aerodinâmicas, tais como ailerons, leme, profundores, spoilers, entre outros. Esse subsistema tem um papel fundamental na estabilidade e na manobrabilidade da aeronave, sendo, portanto, vital para sua operação segura e eficiente.

O subsistema Atuador é tipicamente composto por um motor de corrente contínua (DC), um mecanismo de transmissão e o driver de potência. O dispositivo de teste funcional visa assegurar que o atuador atenda integralmente aos requisitos técnicos estabeleci-

dos durante a fase de projeto, mesmo sem que o fornecedor tenha acesso direto às soluções internas do fabricante.

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver uma metodologia de testes a serem realizados por um dispositivo de recebimento do subsistema Atuador de uma RPA, fornecido por terceiros, com capacidade de excitar, medir e avaliar seu desempenho, assegurando o atendimento aos requisitos definidos durante a fase de projeto do sistema.

1.3.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Desenvolver um modelo dinâmico de referência para o motor de corrente contínua, utilizando técnicas de identificação de sistemas.
- Planejar e gerar sinais de excitação adequados, como o sinal *chirp* e o degrau, garantindo persistência de excitação suficiente para a estimação consistente dos parâmetros do modelo.
- Realizar a identificação dos parâmetros elétricos e mecânicos do motor DC, incluindo resistência de armadura, constante de torque, constante de força contraeletromotriz, inércia e coeficiente de amortecimento, avaliando a qualidade das estimativas por meio das métricas RMSE e *Goodness of Fit* (FIT).
- Validar o modelo identificado, comparando as respostas simuladas às respostas de referência e analisando sua aderência às dinâmicas relevantes por meio de métricas quantitativas e análise gráfica.
- Projetar e sintonizar um controlador do tipo PID, empregando técnicas de otimização baseadas em funções custo multicritério, de modo a atender requisitos de desempenho como tempo de subida, sobressinal e tempo de acomodação.
- Avaliar a robustez do controlador projetado por meio de simulações de Monte Carlo, verificando a sensibilidade do desempenho às variações paramétricas do motor e propondo ajustes que aumentem a robustez quando necessário.

- Implementar os algoritmos e rotinas de controle em ambiente computacional, de forma compatível com futura execução embarcada no dispositivo de recebimento.
- Documentar e organizar, em apêndice, os códigos desenvolvidos em linguagem C (para execução embarcada) e em MATLAB (para identificação, validação e sintonia), garantindo a reprodutibilidade dos experimentos.

1.4 Estrutura do Trabalho

Este trabalho está organizado em cinco capítulos, a saber:

- **Capítulo 1 - Introdução:** Apresenta a contextualização do problema, a motivação para o uso de MBSE, o escopo do dispositivo de recebimento, os objetivos geral e específicos, a justificativa e a estrutura do trabalho.
- **Capítulo 2 - Revisão Bibliográfica:** Aborda os conceitos fundamentais da Engenharia de Sistemas Baseada em Modelos (MBSE), Simulação de Monte Carlo, modelagem de Motores de Corrente Contínua (DC), Identificação de Sistemas e Controladores PID, fornecendo a base teórica para o desenvolvimento.
- **Capítulo 3 - Metodologia:** Apresenta os procedimentos adotados para o desenvolvimento do trabalho, incluindo a modelagem física do motor DC, a definição da estratégia de excitação do sistema, os métodos empregados na identificação paramétrica, a formulação das funções custo utilizadas na sintonia do controlador PID e a abordagem de validação baseada em métricas quantitativas. Também descreve a estrutura da simulação, o uso de técnicas de Monte Carlo para avaliação de robustez e a organização dos experimentos realizados em ambiente MATLAB/Simulink.
- **Capítulo 4 - Desenvolvimento e Resultados:** Descreve o processo de identificação de sistemas, o projeto da malha de controle PID, a implementação do dispositivo de teste funcional e apresenta os resultados dos testes de bancada.
- **Capítulo 5 - Conclusão:** Sumariza as contribuições do trabalho, avalia o cumprimento dos objetivos propostos e sugere trabalhos futuros.

2 Revisão Bibliográfica

2.1 Engenharia de Sistemas Baseada em Modelos

2.1.1 Conceitos Fundamentais

A **Engenharia de Sistemas Baseada em Modelos (Model-Based Systems Engineering — MBSE)** representa uma abordagem moderna e formalizada para o desenvolvimento de sistemas complexos, na qual os modelos assumem o papel central como principal meio de comunicação, análise e integração entre as diversas partes interessadas do projeto (LOPEZ, 2021; CARROL, 2016).

Diferentemente das metodologias tradicionais, predominantemente centradas em documentos e especificações textuais, o MBSE estabelece o modelo como a **Fonte Única de Verdade** (*Single Source of Truth*), isto é, o repositório unificado de informações sobre requisitos, arquitetura, comportamento e verificações de desempenho ao longo de todo o ciclo de vida do sistema (GU *et al.*, 2024). Essa centralização assegura maior consistência, rastreabilidade e coerência entre os diversos níveis de abstração do projeto, além de reduzir ambiguidades e redundâncias de informação.

O surgimento do MBSE está intimamente relacionado à crescente complexidade dos sistemas ciberfísicos modernos, especialmente nas áreas aeroespacial, automotiva e de automação industrial. À medida que os sistemas passaram a incorporar subsistemas eletrônicos, de software e de controle interdependentes, tornou-se inviável manter a coerência de projeto apenas com documentação estática. O MBSE propõe, portanto, uma transição de paradigma: do foco em artefatos textuais para a construção e manipulação de modelos formais, que descrevem de maneira integrada as interações entre hardware, software, ambiente e operadores humanos.

Entre os principais benefícios do MBSE destacam-se:

- **Melhoria da comunicação interdisciplinar**: engenheiros de diferentes domínios (mecânico, elétrico, de software e controle) podem interagir com base em modelos comuns e interconectados;

- **Aumento da rastreabilidade**: alterações em requisitos ou parâmetros de projeto propagam-se automaticamente pelos modelos associados;
- **Redução de erros e inconsistências**: o uso de modelos formais e simulações permite a detecção precoce de falhas conceituais;
- **Suporte à validação antecipada**: a utilização de modelos executáveis viabiliza a verificação virtual do sistema antes da implementação física.

2.1.2 O Ciclo de Vida de Sistemas e o V-Model

O desenvolvimento de sistemas complexos no contexto da engenharia moderna segue, em grande parte, uma estrutura inspirada no **V-Model** (ou *Modelo em V*), que representa o ciclo de vida de um sistema desde a concepção até a operação e manutenção (CICCHETTI, 2024). O modelo em V enfatiza a natureza iterativa e hierárquica do processo de engenharia, apresentando no lado esquerdo as fases de decomposição e definição, tais como: a análise de requisitos, a modelagem funcional e a arquitetura de sistema; E no lado direito as fases de integração, verificação e validação (V&V).

Cada fase de desenvolvimento no lado descendente do V-Model possui uma etapa correspondente de validação no lado ascendente. Por exemplo, os requisitos de sistema são verificados durante os testes de aceitação, enquanto os componentes de hardware e software são validados durante os testes de integração e qualificação. Essa estrutura garante que as decisões de projeto sejam continuamente verificadas em relação aos objetivos originais, promovendo um ciclo de realimentação contínua entre modelagem e teste.

O MBSE potencializa o V-Model ao introduzir modelos formais e simuláveis como elementos centrais do ciclo de vida. Essa integração permite que atividades de verificação e validação sejam iniciadas de forma antecipada, ainda nas fases iniciais de concepção, por meio de simulações computacionais, análises de sensibilidade e técnicas de avaliação probabilística, como a Simulação de Monte Carlo. Dessa forma, é possível identificar falhas de concepção e inconsistências de integração antes da fabricação ou implementação física, reduzindo significativamente o custo e o tempo de desenvolvimento.

Além disso, a aplicação conjunta do MBSE e do V-Model facilita a implementação de um ciclo contínuo de melhoria e atualização de modelos. À medida que dados reais de teste e operação são coletados, os modelos podem ser calibrados e refinados, realimentando o processo de engenharia. Esse fluxo de informações bidirecional é essencial em sistemas críticos, como aeronaves remotamente controladas, onde a confiabilidade e a rastreabilidade de decisões de projeto são requisitos mandatórios.

Em síntese, o V-Model fornece a estrutura organizacional do processo de desenvolvi-

mento, enquanto o MBSE oferece os meios técnicos para sua execução orientada a modelos. Juntos, eles constituem uma abordagem robusta e integrada para o desenvolvimento de sistemas complexos, permitindo que atividades de projeto, simulação, verificação e validação coexistam de forma contínua e coerente ao longo de todo o ciclo de vida.

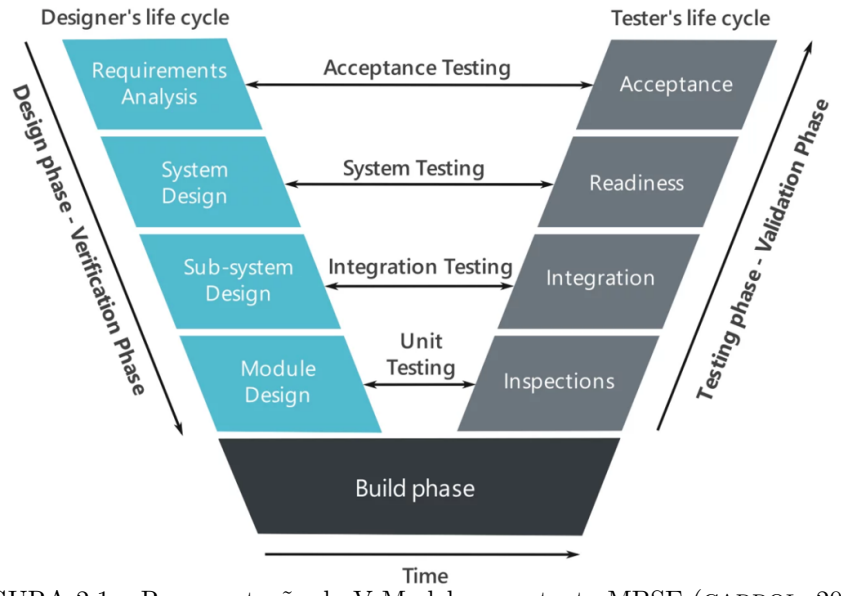

FIGURA 2.1 – Representação do V-Model no contexto MBSE (CARROL, 2016).

2.1.3 Linguagens e Ferramentas

No contexto da Engenharia de Sistemas Baseada em Modelos (MBSE), a escolha da linguagem de modelagem é um elemento central para garantir a consistência, rastreabilidade e integração dos diversos níveis do projeto. A linguagem de modelagem mais amplamente adotada é a **Systems Modeling Language (SysML)**, uma extensão da UML (Unified Modeling Language) adaptada especificamente para a engenharia de sistemas. Diferentemente da UML tradicional, que se concentra principalmente em sistemas de software, a SysML abrange diagramas capazes de representar requisitos, estrutura, comportamento e parâmetros de um sistema, fornecendo uma visão holística que facilita o entendimento, a comunicação e a documentação do projeto (MathWorks, 2024a).

A SysML permite a criação de diagramas específicos, como diagramas de requisitos, de blocos, de atividades, de estados e diagramas paramétricos. Cada tipo de diagrama desempenha um papel distinto no ciclo de vida do sistema: os diagramas de requisitos asseguram que todas as necessidades do cliente sejam formalmente capturadas; os diagramas de blocos descrevem a decomposição estrutural do sistema; os diagramas de atividades e de estados modelam o comportamento dinâmico; e os diagramas paramétricos suportam a análise quantitativa de desempenho e restrições físicas.

A integração entre a modelagem SysML e as ferramentas de simulação é um aspecto

crítico do MBSE, pois permite que os modelos de arquitetura e requisitos influenciem diretamente a análise de comportamento do sistema. Ferramentas como **MATLAB/Simulink** são amplamente utilizadas para modelagem dinâmica e simulação de sistemas, abrangendo exemplos práticos como a modelagem de motores DC, controladores PID e sistemas de controle embarcados. Por meio de conectores específicos ou plataformas integradoras, como **ModelCenter**, é possível sincronizar os modelos de simulação com a arquitetura de alto nível definida em SysML, garantindo que alterações em requisitos ou no design estrutural sejam refletidas automaticamente nos modelos de simulação (CAO *et al.*, 2011).

2.1.4 Integração de Modelagem, Simulação e Validação

A integração entre modelagem e simulação viabiliza um processo iterativo de desenvolvimento, no qual alterações em qualquer nível do projeto podem ser rapidamente avaliadas em termos de impacto funcional, estrutural e de desempenho. Essa abordagem promove a *co-simulação*, permitindo que modelos de arquitetura em SysML troquem informações com modelos de simulação em Simulink, garantindo consistência e rastreabilidade de ponta a ponta (CHABIBI, 2011).

A validação precoce, proporcionada por essa integração, é um dos maiores benefícios do MBSE. Ao permitir que requisitos e modelos de alto nível sejam diretamente conectados aos modelos de simulação, engenheiros podem identificar inconsistências, restrições de desempenho e falhas de projeto ainda nas fases iniciais de desenvolvimento, reduzindo significativamente custos e retrabalho em fases posteriores. Além disso, a abordagem iterativa favorece a evolução contínua do projeto, permitindo ajustes em tempo real e suportando decisões baseadas em dados quantitativos gerados pela simulação.

2.2 Motores de Corrente Contínua

2.2.1 Princípios de Funcionamento

Motores de Corrente Contínua (DC) são máquinas eletromecânicas projetadas para converter energia elétrica contínua em energia mecânica rotacional. Seu princípio de funcionamento baseia-se na interação entre o campo magnético do estator (fixo) e o campo magnético gerado pela corrente elétrica que percorre o rotor (armadura). Quando a corrente elétrica circula pelos enrolamentos do rotor, forças magnéticas são geradas em conformidade com a Lei de Lorentz, resultando em torque que provoca o movimento rotacional do eixo do motor.

Esses motores apresentam características favoráveis como controle preciso de veloci-

dade e torque, facilidade de implementação de sistemas de controle, além de uma ampla gama de aplicações em robótica, automação industrial e veículos elétricos. Dependendo do tipo de excitação do campo, podem ser classificados como *shunt*, *serie* ou *composto*, cada um com características específicas de torque-velocidade e resposta dinâmica.

2.2.2 Equações Elétricas e Mecânicas

O comportamento dinâmico de um motor DC pode ser descrito por um conjunto de equações diferenciais acopladas que representam suas propriedades elétricas e mecânicas. A modelagem dessas equações é fundamental para análise de desempenho, simulação e projeto de controladores (University of Michigan, 2024).

2.2.3 Modelo Matemático do Motor DC

2.2.3.1 Equações no Domínio do Tempo

A equação elétrica que descreve a dinâmica da armadura é dada por:

$$L_a \frac{di_a(t)}{dt} + R_a i_a(t) + E_b(t) = V_a(t) \tag{2.1}$$

onde $V_a(t)$ é a tensão aplicada à armadura, $i_a(t)$ é a corrente de armadura, R_a e L_a representam, respectivamente, a resistência e a indutância do enrolamento, e $E_b(t)$ é a força contra-eletromotriz (FCEM). A FCEM é proporcional à velocidade angular do rotor:

$$E_b(t) = K_b\, \omega(t) \tag{2.2}$$

A equação mecânica que governa o movimento do rotor é:

$$J \frac{d\omega(t)}{dt} + B\omega(t) + \tau_L(t) = \tau_m(t) \tag{2.3}$$

onde J é o momento de inércia total do rotor e carga acoplada, B é o coeficiente de atrito viscoso, $\tau_L(t)$ representa o torque de carga aplicado externamente e $\tau_m(t)$ é o torque eletromagnético gerado pelo motor. O torque eletromagnético é proporcional à corrente de armadura:

$$\tau_m(t) = K_t\, i_a(t) \tag{2.4}$$

2.2.4 Parâmetros Característicos e Datasheet

Os parâmetros R_a, L_a, J, B, K_t e K_b são essenciais para uma modelagem precisa do motor DC. Idealmente, esses valores são fornecidos no **Datasheet** do fabricante. No entanto, fatores como variações de fabricação, desgaste, temperatura e condições operacionais podem gerar discrepâncias. Por isso, técnicas de **Identificação de Sistemas** experimental são frequentemente aplicadas para ajustar os modelos teóricos aos comportamentos reais, garantindo precisão em simulações e no projeto de controladores.

2.3 Simulação de Monte Carlo

2.3.1 Fundamentos Estatísticos

A **Simulação de Monte Carlo (SMC)** é um método numérico que se baseia em amostragens aleatórias para a solução de problemas matemáticos e de engenharia que, embora determinísticos em sua formulação, apresentam incertezas significativas em seus parâmetros ou condições de contorno (Wikipedia contributors, 2024). O termo "Monte Carlo" foi popularizado durante a década de 1940 pelos pesquisadores Stanislaw Ulam e John von Neumann, no contexto dos estudos de difusão de nêutrons no Projeto Manhattan, e desde então tornou-se uma ferramenta essencial em diversas áreas do conhecimento.

A fundamentação teórica da SMC repousa sobre princípios da estatística e da teoria das probabilidades, em especial na *Lei dos Grandes Números* e no *Teorema Central do Limite*. A Lei dos Grandes Números estabelece que, à medida que o número de experimentos independentes aumenta, a média aritmética dos resultados converge para o valor esperado da variável aleatória em análise. Já o Teorema Central do Limite garante que, sob condições gerais, a distribuição da média amostral tende a uma distribuição Normal, independentemente da distribuição original das amostras.

Esses princípios asseguram que, quanto maior o número de iterações simuladas, menor será a incerteza associada à estimativa numérica obtida. Dessa forma, a SMC permite avaliar o comportamento estatístico de sistemas complexos, cuja solução analítica seria inviável ou excessivamente custosa. Essa abordagem é particularmente útil na engenharia de sistemas, onde a presença de incertezas em parâmetros físicos, medições ou modelos matemáticos pode afetar significativamente o desempenho e a confiabilidade de um projeto.

2.3.2 Geração de Amostras Aleatórias e Distribuições de Probabilidade

O processo de simulação de Monte Carlo depende fortemente da capacidade de gerar amostras aleatórias que sigam distribuições de probabilidade predefinidas, representando adequadamente as incertezas do sistema modelado. As distribuições mais comumente utilizadas incluem a *Normal* (ou Gaussiana), a *Uniforme*, a *Exponencial*, a *Weibull* e a *Log-Normal*, entre outras (Scratchapixel, 2024). A escolha da distribuição apropriada é uma etapa crítica, pois define a forma estatística das variações consideradas no modelo.

No contexto da engenharia de controle e automação, essas distribuições são frequentemente utilizadas para modelar:

- **Tolerâncias de fabricação**, que introduzem variações nos parâmetros elétricos e mecânicos de motores, sensores e atuadores;
- **Ruídos de medição**, associados a imprecisões de sensores como encoders e sensores de efeito Hall;
- **Condições operacionais variáveis**, como temperatura, atrito, ou tensões de alimentação;
- **Incertezas de modelagem**, decorrentes de simplificações matemáticas ou de erros de identificação de parâmetros.

Para cada execução de uma simulação de Monte Carlo, um conjunto distinto de valores amostrados é atribuído aos parâmetros incertos do sistema. O modelo é então avaliado, e os resultados de saída como resposta temporal, erro de regime permanente, ou margens de estabilidade, etc, são registrados. Após um número elevado de iterações, é possível construir distribuições empíricas dessas métricas, permitindo analisar sua dispersão e sensibilidade.

Essa análise fornece informações valiosas para o processo de tomada de decisão em engenharia, possibilitando, por exemplo, a avaliação da **robustez** de controladores e a identificação de parâmetros críticos que impactam o desempenho global do sistema. Em especial, a SMC tem sido amplamente utilizada para validar modelos obtidos por técnicas de identificação de sistemas, bem como para otimizar parâmetros de controladores do tipo PID sob condições de incerteza paramétrica.

2.3.3 Aplicações em Engenharia de Sistemas e Controle

No contexto da **Engenharia de Sistemas Baseada em Modelos (MBSE)**, a Simulação de Monte Carlo representa uma ferramenta fundamental para a análise probabilística

de desempenho e confiabilidade de subsistemas. Em um projeto de sistema de atuação eletromecânico, a SMC pode ser empregada para:

- Avaliar o impacto de incertezas nos parâmetros identificados do modelo do motor DC;
- Quantificar a variabilidade na resposta de posição ou velocidade diante de ruídos de medição e perturbações externas;
- Suportar a etapa de *tuning* automático de controladores PI-D, ao analisar a dispersão das métricas de desempenho para diferentes conjuntos de ganhos;
- Fornecer subsídios estatísticos para a definição de requisitos de confiabilidade e segurança no ciclo de desenvolvimento MBSE.

Assim, a Simulação de Monte Carlo se integra de maneira natural ao paradigma do MBSE, complementando os processos de modelagem, validação e verificação de sistemas complexos. Sua aplicação contribui para a obtenção de um projeto mais robusto e confiável, com maior previsibilidade de comportamento mesmo diante de incertezas inerentes ao ambiente de operação.

2.4 Identificação de Sistemas

2.4.1 Conceitos Fundamentais

A **Identificação de Sistemas** é uma disciplina que busca construir modelos matemáticos de sistemas dinâmicos a partir de dados experimentais de entrada e saída (MathWorks, 2024b). Diferente da modelagem baseada em princípios físicos, a identificação de sistemas permite gerar modelos mesmo quando a dinâmica interna do sistema não é completamente conhecida ou é muito complexa para modelagem analítica.

O processo de identificação envolve várias etapas interdependentes: a escolha da *estrutura do modelo* (por exemplo, Função de Transferência, Espaço de Estados, modelos ARX/ARMAX), o *projeto do sinal de excitação* que será aplicado ao sistema, a *aquisição de dados* de forma precisa e confiável, e a *estimação dos parâmetros* que melhor reproduzem a dinâmica observada. Cada etapa deve ser cuidadosamente planejada para garantir que o modelo obtido seja válido, robusto e representativo do sistema real.

2.4.2 Tipos de Sinais de Excitação

O sinal de entrada, também chamado de excitação, é crítico para o sucesso da identificação. Ele deve ser **suficientemente rico** para excitar todas as dinâmicas relevantes do sistema. Caso contrário, o modelo estimado poderá apresentar comportamento incorreto ou incompleto.

Sinais comumente utilizados incluem:

- **Degrau (Step)**: ideal para identificar respostas de primeira e segunda ordem, devido à sua simplicidade e facilidade de implementação.
- **Ruído Branco (White Noise)**: possui conteúdo espectral amplo, útil para caracterizar sistemas em uma faixa extensa de frequências.
- **Sinal Chirp (Sweep Sinusoidal)**: varia continuamente a frequência ao longo do tempo, permitindo excitar diferentes modos do sistema de maneira eficiente.

2.4.3 Métodos de Estimação de Sistemas

A estimação de parâmetros em sistemas dinâmicos é uma etapa fundamental no processo de modelagem e identificação, permitindo ajustar um modelo matemático de forma que este represente adequadamente o comportamento do sistema físico real. Dentre as abordagens clássicas de estimação, destacam-se o **Método do Erro na Saída** (*Output Error Method* – OEM) e o **Método do Erro no Filtro** (*Filter Error Method* – FEM), amplamente empregados na identificação de sistemas lineares e não lineares (LJUNG, 1999; UNBEHAUEN; RAO, 1990).

2.4.3.1 Método do Erro na Saída (Output Error Method)

O método do Erro na Saída tem como princípio minimizar a diferença entre a saída medida do sistema físico e a saída simulada pelo modelo. Considerando um modelo paramétrico de um sistema dinâmico discreto, descrito por:

$$y(k) = G(q, \vec{\Theta})\, u(k) + H(q, \vec{\Theta})\, e(k), \tag{2.5}$$

onde:

- $y(k)$ é a saída medida no instante k;
- $u(k)$ é a entrada aplicada;

- $G(q, \vec{\Theta})$ é o modelo dinâmico (função de transferência) parametrizado;
- $H(q, \vec{\Theta})$ é o modelo do ruído (geralmente assumido como $H(q) = 1$ para OEM);
- $e(k)$ é o erro branco com variância σ^2.

A função custo a ser minimizada é dada por:

$$J(\vec{\Theta}) = \frac{1}{N} \sum_{k=1}^{N} \left[y(k) - \hat{y}(k, \vec{\Theta}) \right]^2, \tag{2.6}$$

em que $\hat{y}(k, \vec{\Theta})$ representa a saída estimada pelo modelo para o vetor de parâmetros $\vec{\Theta}$. O processo de otimização busca o conjunto de parâmetros que minimiza o erro quadrático médio entre a saída real e a saída simulada.

O método OEM é classificado como um estimador *não recursivo*, pois o cálculo do erro requer a simulação completa do modelo para cada iteração de otimização. Embora seja sensível a ruídos de medição, apresenta boa robustez para sistemas determinísticos e de dinâmica predominantemente linear.

2.4.3.2 Método do Erro no Filtro (Filter Error Method)

O método do Erro no Filtro (*Filter Error Method* – FEM) constitui uma extensão do OEM, introduzindo um filtro interno para tratar explicitamente os efeitos do ruído no processo de estimação (GOODWIN; PAYNE, 1977). A ideia central consiste em aplicar um filtro estável $F(q, \vec{\Theta})$ à saída e à entrada do sistema, de modo a reduzir a influência do ruído e permitir a estimação conjunta dos parâmetros do modelo e da dinâmica do ruído.

A função custo correspondente é expressa como:

$$J_F(\vec{\Theta}) = \frac{1}{N} \sum_{k=1}^{N} \left[F(q, \vec{\Theta}) \left(y(k) - \hat{y}(k, \vec{\Theta}) \right) \right]^2. \tag{2.7}$$

O filtro $F(q, \vec{\Theta})$ é, em geral, escolhido como o inverso do modelo de ruído $H(q, \vec{\Theta})$, o que torna o erro filtrado uma sequência aproximadamente branca quando os parâmetros estão próximos dos valores verdadeiros. O FEM, portanto, fornece estimativas com menor viés em presença de ruído de medição, sendo particularmente eficaz em sistemas com perturbações estocásticas significativas.

2.5 Controlador PID

2.5.1 Controladores Clássicos PID e PI-D

O **Controlador Proporcional-Integral-Derivativo (PID)** é o algoritmo de controle mais utilizado na indústria devido à sua simplicidade, robustez e eficácia em uma ampla variedade de sistemas. O PID atua sobre o erro de controle, definido como a diferença entre a referência desejada ($r(t)$) e a saída medida ($y(t)$), e calcula um sinal de controle $u(t)$ que combina três ações:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{2.8}$$

onde K_p, K_i e K_d são os ganhos proporcional, integral e derivativo, respectivamente.

A ação **Proporcional** reduz o erro instantâneo, a ação **Integral** elimina o erro de regime estacionário, e a ação **Derivativa** antecipa tendências de erro futuro, melhorando a resposta transitória e amortecendo oscilações. O diagrama de blocos do controlador PID é ilustrado na Figura 2.2.

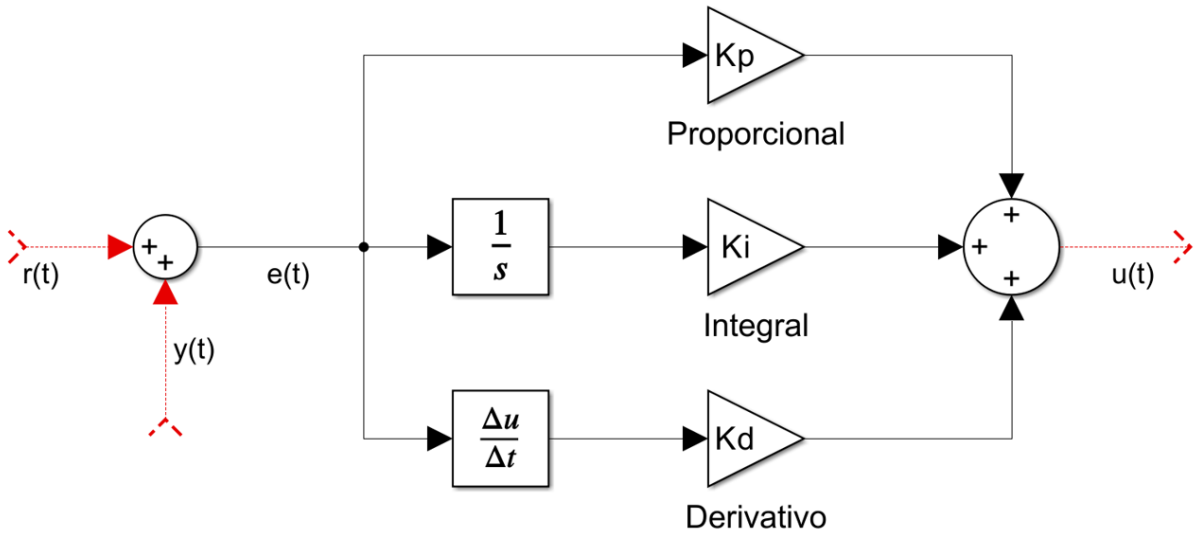

FIGURA 2.2 – Diagrama de blocos do controlador PID.

Uma variação conhecida é o controlador **PI-D**, também conhecido como realimentação de estados, onde a ação derivativa é aplicada apenas à uma variável controlada (saída) ou medida e não ao erro, minimizando picos de controle em respostas abruptas do setpoint, especialmente em sistemas com alta sensibilidade ou ruído de medição. Sua expressão é exposta na Equação 2.9 e o diagrama de blocos na Figura 2.3.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau - K_d \frac{dy(t)}{dt} \tag{2.9}$$

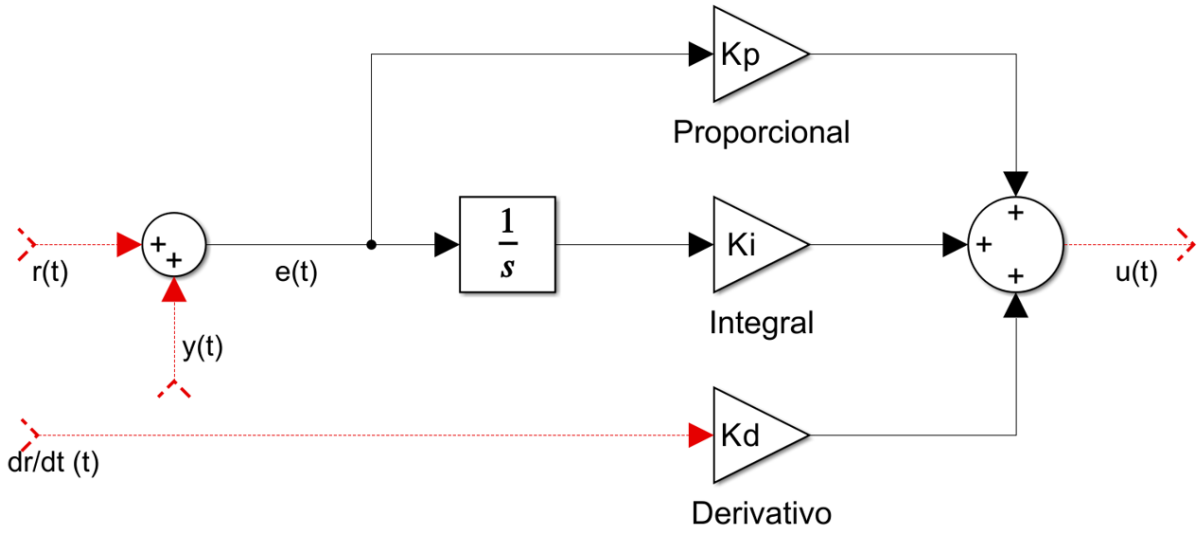

FIGURA 2.3 – Diagrama de blocos do controlador PI-D.

2.5.2 Critérios de Desempenho e Estabilidade

O projeto de controladores PID é orientado por métricas de desempenho no domínio do tempo e da frequência. Entre os critérios mais importantes estão:

- **Tempo de Subida (*Rise Time*,** t_r**)**: tempo necessário para a saída atingir pela primeira vez o valor de referência.
- **Sobressinal (*Overshoot*,** M_p**)**: máximo valor da saída acima do valor de referência, expresso em percentual.
- **Tempo de Acomodação (*Settling Time*,** t_s**)**: tempo requerido para que a saída permaneça dentro de uma faixa de tolerância em torno do valor final.
- **Erro de Regime Estacionário (*Steady-State Error*,** e_{ss}**)**: diferença entre a saída e o valor de referência após o sistema atingir o regime permanente.

A **estabilidade** é o requisito mais fundamental, garantindo que o sistema não apresente oscilações não controladas ou divergência da resposta. Técnicas clássicas como o critério de Routh-Hurwitz ou análise de polos em malha fechada podem ser utilizadas para avaliar estabilidade do sistema controlado.

2.5.3 Métodos Clássicos de Sintonia

A sintonia do PID consiste em determinar os valores adequados de K_p, K_i e K_d para atender aos critérios de desempenho desejados. Alguns métodos clássicos incluem:

- **Tentativa e Erro (*Trial and Error*)**: abordagem empírica, ajustando os ganhos até atingir desempenho aceitável, porém demorada e dependente da experiência do projetista.
- **Ziegler-Nichols**: utiliza a resposta ao degrau ou a oscilações sustentadas em malha fechada para calcular os ganhos com base em regras heurísticas.
- **Cohen-Coon**: adequado para sistemas com tempo morto (atraso), ajusta os ganhos com base em parâmetros da resposta ao degrau.

2.5.4 Abordagens de Sintonia Automática Baseadas em Modelo

Com o avanço da modelagem e identificação de sistemas, métodos modernos utilizam o modelo matemático do sistema para determinar automaticamente os ganhos PID que otimizam um critério de desempenho específico. Esses critérios podem incluir:

- **Erro Absoluto Integrado (IAE)**: minimiza o valor absoluto do erro ao longo do tempo.
- **Erro Quadrático Integrado (ISE)**: minimiza o quadrado do erro, penalizando grandes desvios.
- **Erro Temporal Absoluto Integrado (ITAE)**: pondera o erro ao longo do tempo, enfatizando a redução de oscilações e tempo de acomodação.

A sintonia baseada em modelo permite uma abordagem sistemática, reduzindo o tempo de projeto e aumentando a precisão do controlador em comparação com métodos puramente empíricos, especialmente em sistemas complexos ou multivariáveis (RIBEIRO *et al.*, 2017).

2.6 Sensores e Instrumentação

2.6.1 Sensor de Posição: Encoder

O **Encoder** é um dispositivo eletromecânico que converte movimento angular ou linear em sinais digitais, permitindo a medição precisa da posição do eixo do motor. Encoders podem ser classificados em duas categorias principais:

- **Incrementais**: fornecem pulsos proporcionais ao deslocamento. A contagem desses pulsos permite determinar a posição relativa e, a partir da variação temporal, calcular a velocidade angular do eixo.

- **Absolutos**: fornecem um código digital único para cada posição do eixo, permitindo determinar a posição absoluta mesmo após interrupções de energia.

Os encoders são fundamentais em sistemas de controle de malha fechada, servindo como fonte de feedback de posição para controladores PID e sistemas de identificação de parâmetros.

2.6.2 Sensor de Velocidade: Tacômetro

O **Tacômetro** é um dispositivo projetado para medir diretamente a velocidade angular do motor. Em aplicações práticas, a velocidade também pode ser obtida derivando o sinal de posição fornecido pelo encoder.

A medição precisa da velocidade é crucial para sistemas de controle de velocidade e torque, como nos motores DC estudados anteriormente, pois permite o ajuste dinâmico do controlador PID e evita instabilidades ou oscilações indesejadas.

2.6.3 Sensor de Corrente: Efeito Hall

Sensores baseados no **Efeito Hall** são amplamente utilizados para medir a corrente de armadura (i_a) em motores DC. Esses sensores oferecem vantagens importantes:

- **Isolamento Galvânico**: protege o sistema eletrônico de alta tensão.
- **Alta Precisão**: permite monitoramento preciso da corrente instantânea.
- **Aplicação em Identificação de Sistemas**: os dados de corrente são essenciais para estimar parâmetros elétricos do motor, como resistência (R_a) e constante de torque (K_t).

2.6.4 Aquisição de Dados, Ruído e Taxa de Amostragem

A **Aquisição de Dados (DAQ)** consiste na conversão de sinais analógicos (tensão, corrente) em sinais digitais, que podem ser processados por computadores ou microcontroladores. Aspectos fundamentais incluem:

- **Ruído**: inevitável em qualquer medição real, causado por interferência eletromagnética, acoplamento indutivo e ruído eletrônico dos sensores. Técnicas de filtragem são necessárias para minimizar seus efeitos.

- **Taxa de Amostragem (*Sampling Rate*)**: deve ser escolhida de acordo com o Teorema de Nyquist, garantindo que a frequência de amostragem seja pelo menos dez vezes maior que a maior frequência presente no sinal, evitando aliasing e capturando fielmente a dinâmica do sistema.

2.6.5 Tratamento de Sinais e Filtragem

O **Tratamento de Sinais** é essencial para extrair informações precisas dos sensores e fornecer dados confiáveis para controle e identificação de sistemas. Técnicas comuns incluem:

- **Filtros de Média Móvel**: simples, suaviza ruído de alta frequência sem introduzir grande atraso.
- **Filtros de Kalman**: permitem estimar estados do sistema a partir de medições ruidosas, combinando modelo do sistema e medidas reais para fornecer estimativas ótimas.
- **Outras técnicas digitais**: como filtros passa-baixa, passa-faixa ou notch, dependendo das características do ruído e da aplicação.

O processamento adequado dos sinais garante que os dados fornecidos ao controlador PID e aos algoritmos de Identificação de Sistemas sejam consistentes, melhorando a precisão do modelo matemático e a estabilidade do sistema controlado.

3 Metodologia

3.1 Dispositivo de Recebimento

3.1.1 Descrição Geral

O Dispositivo de Recebimento é responsável por realizar a leitura dos sensores, o controle do motor e a comunicação com a interface homem-máquina (IHM). Ele é constituído pelos seguintes componentes principais:

- Microcontrolador STM32 Bluepill: Executa as malhas de controle, gera o sinal PWM destinado ao driver de potência, realiza a leitura dos sensores e transmite as informações à IHM por meio da rede CAN;
- Driver de Potência L298: Composto por duas pontes H completas, possibilita o controle bidirecional de motores de corrente contínua;
- Encoder Magnético Absoluto: Sensor de posição acoplado ao eixo do motor, com resolução de 16 bits, utilizado para determinar a posição angular com alta precisão;
- Comunicação via CAN: Conjunto formado pelos transceptores MAX490 e MCP2551, responsáveis por estabelecer a comunicação entre o microcontrolador e os demais dispositivos conectados à rede.

Além dos elementos principais, o dispositivo pode incluir os seguintes componentes adicionais:

- Medidor de Velocidade: Permite o monitoramento da rotação do eixo do motor, fornecendo dados complementares para o controle de movimento;
- Sensor de Corrente do Tipo Hall: Mede a corrente elétrica consumida pelo motor, possibilitando o monitoramento de desempenho e a implementação de proteções contra sobrecorrente.

3.2 Modelos de Simulação

O software MATLAB/Simulink foi utilizado como ferramenta de modelagem e simulação do dispositivo de recebimento proposto, conforme ilustrado na Figura 3.1.

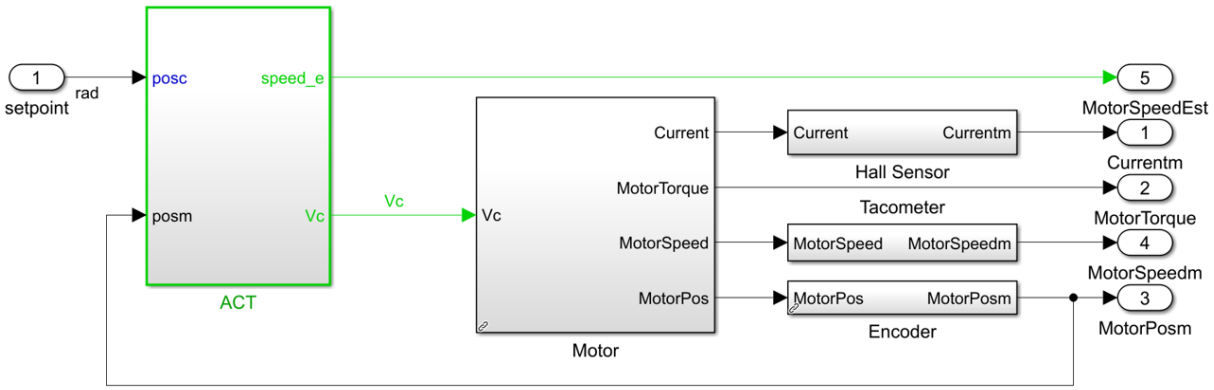

FIGURA 3.1 – Modelo do dispositivo de recebimento.

O modelo do motor foi desenvolvido com base nos principais parâmetros elétricos e mecânicos que influenciam sua dinâmica, conforme descrito na Seção 2.2.3. Adicionalmente, foram inseridos valores de *bias* nos parâmetros, a fim de representar possíveis desvios decorrentes do processo de montagem e de tolerâncias construtivas.

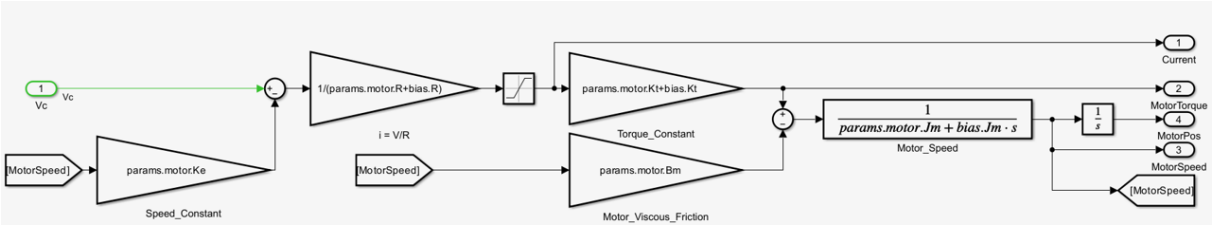

FIGURA 3.2 – Modelo do motor.

A Tabela 3.1 apresenta os parâmetros do motor utilizados no modelo, assim como os bias considerados nas simulações de Monte Carlo.

TABELA 3.1 – Parâmetros do motor

Variável	Parâmetro	Valor	Unidade	Variação
R	Resistência de Armadura	5,8	Ω	$\pm 20\%$
K_t	Constante de Torque	0,079	Nm/A	$\pm 10\%$
J_m	Inércia	$1{,}73e^{-4}$	kg·m^2	$\pm 10\%$
K_e	Constante de Força Eletromotriz	0,079	V/(rad/s)	N/A
B_m	Amortecimento	$700e^{-6}$	kg·m^2/s	N/A
$\tau = \frac{J_m}{B_m}$	Constante de Tempo	0,247	s	N/A

Os sensores foram modelados considerando o efeito de quantização para 12 bits, conforme ilustrado na Figura 3.3.

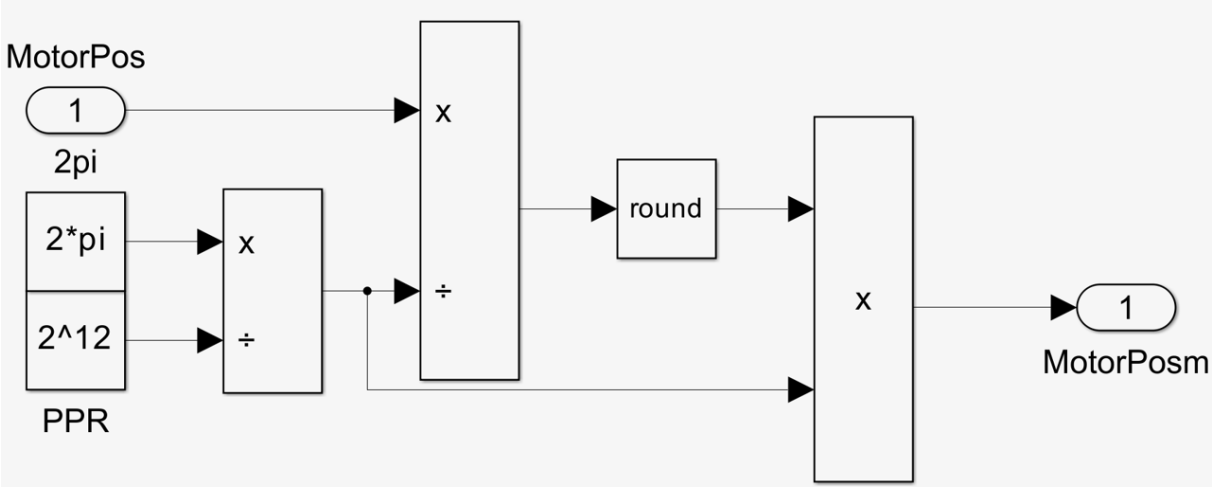

FIGURA 3.3 – Modelo do sensor de posição.

Os módulos correspondentes ao driver de potência e ao sistema de transmissão de dados foram omitidos da simulação, assumindo-se que o projeto de hardware opera de forma ideal. O bloco denominado ACT, apresentado na Figura 3.4, contém a estrutura de controle PI-D proposta para o dispositivo desenvolvido neste trabalho.

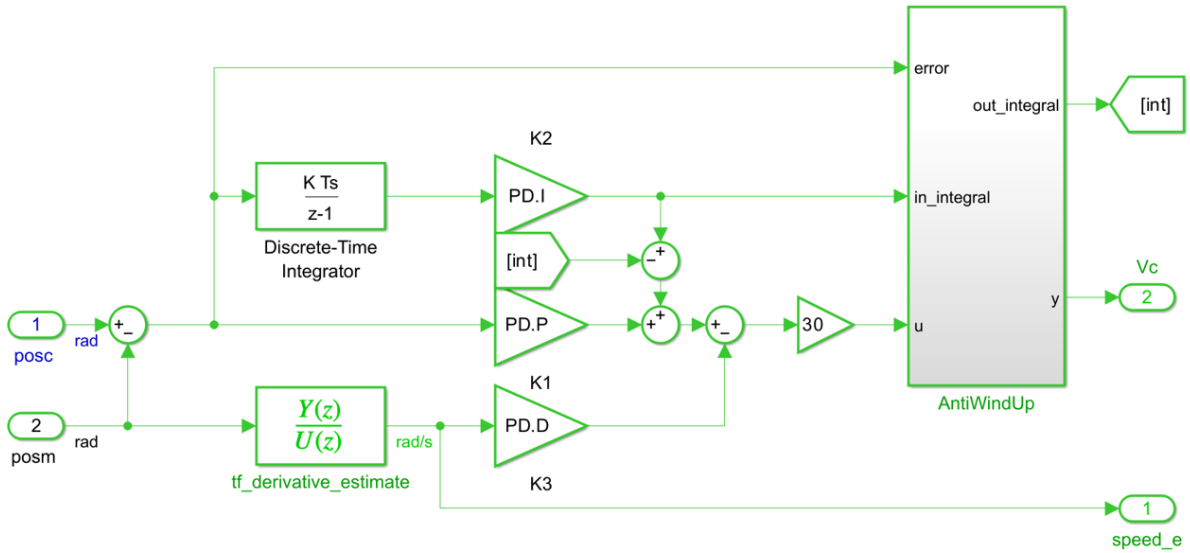

FIGURA 3.4 – Bloco de controle PI-D.

3.2.1 Análise Dinâmica do Motor

A análise dinâmica do motor de corrente contínua tem como objetivo obter a relação entre a tensão aplicada ao enrolamento da armadura e a velocidade angular no eixo do motor, permitindo a posterior análise em frequência e o projeto da malha de controle. Para essa finalidade, adota-se o modelo clássico do motor DC, no qual se considera o comportamento eletromecânico acoplado e despreza-se a indutância elétrica da armadura, hipótese válida para a faixa de frequências de interesse neste trabalho.

A partir das equações que regem a dinâmica elétrica e mecânica do motor, tem-se que a tensão aplicada pode ser expressa como

$$V(t) = Ri(t) + K_e\omega(t), \tag{3.1}$$

enquanto a equação do movimento mecânico é dada por

$$J_m\dot{\omega}(t) + B_m\omega(t) = K_t i(t), \tag{3.2}$$

onde R é a resistência da armadura, $i(t)$ é a corrente elétrica, K_e é a constante de força contraeletromotriz, J_m é o momento de inércia do rotor, B_m é o coeficiente de atrito viscoso e K_t é a constante de torque do motor.

Aplicando a Transformada de Laplace às equações anteriores, assumindo condições iniciais nulas, e eliminando a variável intermediária $I(s)$, obtém-se a função de transferência entre a velocidade angular e a tensão aplicada:

$$\frac{\omega(s)}{V(s)} = \frac{\dfrac{K_t}{R}}{J_m s + B_m + \dfrac{K_t K_e}{R}}. \tag{3.3}$$

A Equação (3.3) representa a função de transferência do modelo físico do motor, porém ainda não se encontra na forma padrão utilizada em análises de sistemas de controle. Para tal, fatoriza-se o termo constante do denominador, resultando em

$$J_m s + B_m + \frac{K_t K_e}{R} = \left(B_m + \frac{K_t K_e}{R}\right)\left(\frac{J_m}{B_m + \frac{K_t K_e}{R}} s + 1\right). \tag{3.4}$$

Substituindo essa expressão na Equação (3.3) e reorganizando os termos, a função de transferência pode ser reescrita como

$$\frac{\omega(s)}{V(s)} = \frac{K_t}{RB_m + K_t K_e} \cdot \frac{1}{\left(\dfrac{RJ_m}{RB_m + K_t K_e}\right) s + 1}. \tag{3.5}$$

Dessa forma, define-se o ganho estático do sistema e a constante de tempo como

$$K = \frac{K_t}{RB_m + K_t K_e}, \qquad \tau = \frac{RJ_m}{RB_m + K_t K_e}, \tag{3.6}$$

obtendo-se a função de transferência na forma padrão de um sistema de primeira ordem:

$$\frac{\omega(s)}{V(s)} = \frac{K}{\tau s + 1}. \tag{3.7}$$

Substituindo os valores dos parâmetros identificados, apresentados na Tabela 3.1, chega-se à função de transferência numérica do motor:

$$\frac{\omega(s)}{V(s)} = \frac{7{,}669}{0{,}0974s + 1}. \tag{3.8}$$

Observa-se que o ganho em regime permanente do sistema é de 7,669 rad/s/V, enquanto a constante de tempo é $\tau = 0{,}0974$ s, correspondendo a uma frequência de polo de aproximadamente $f_p = 1{,}634$ Hz. O comportamento dinâmico do motor em frequência é analisado por meio do diagrama de Bode, apresentado na Figura 3.5.

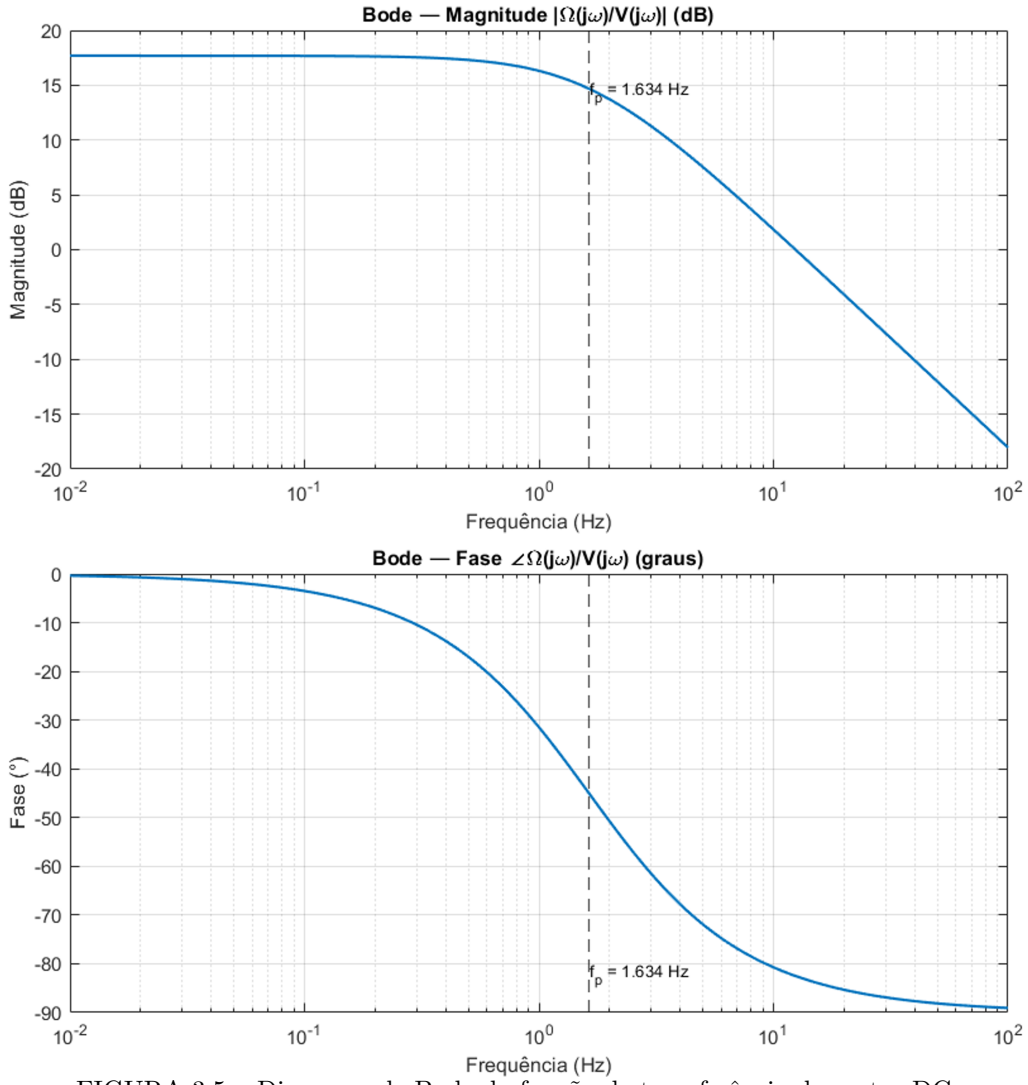

FIGURA 3.5 – Diagrama de Bode da função de transferência do motor DC.

3.3 Procedimento de Identificação de Parâmetros

3.3.1 Modelo Caixa Branca do Motor DC

Adotou-se uma abordagem baseada em um *modelo caixa-branca* para o motor. No modelo de identificação, a variável de entrada do modelo corresponde à tensão aplicada à armadura, enquanto a variável de saída é a posição angular do eixo do motor.

Os parâmetros a serem identificados são:

$$\vec{\Theta} = [R; K_t; J_m; \tau] \tag{3.9}$$

A estimativa da velocidade é realizada por meio de um filtro derivativo, com faixa de passagem ajustada para 100 Hz, de modo a preservar as componentes dinâmicas relevantes e atenuar o ruído de alta frequência. Alternativamente, a utilização de um tacômetro permite a medição direta da velocidade, e a inclusão de um sensor de corrente do tipo *Hall* possibilita a utilização da corrente de armadura como variável de saída adicional.

Entretanto, no escopo deste trabalho, considera-se apenas a medição da posição angular. Assim, o vetor de saída do modelo é definido por:

$$y = \begin{bmatrix} \theta \\ \omega \end{bmatrix}, \tag{3.10}$$

em que θ representa a posição angular e ω a velocidade angular estimada.

3.3.2 Entrada de Excitação: Sinal Chirp

O sinal *chirp*, ou varredura senoidal, foi escolhido por sua capacidade de cobrir uma grande banda de frequências de forma contínua e controlada (PAAVLE, 2011; VUOJOLAINEN *et al.*, 2017).

A equação empregada da *chirp* é dada por:

$$u(t) = A \sin(2\pi f t), \tag{3.11}$$

No presente trabalho, o sinal foi configurado para excitar o motor na faixa de $1H_z$ a $6H_z$, com 10 oscilações em cada frequência, com uma amplitude de $10V$, como demonstrado na Figura 3.6.

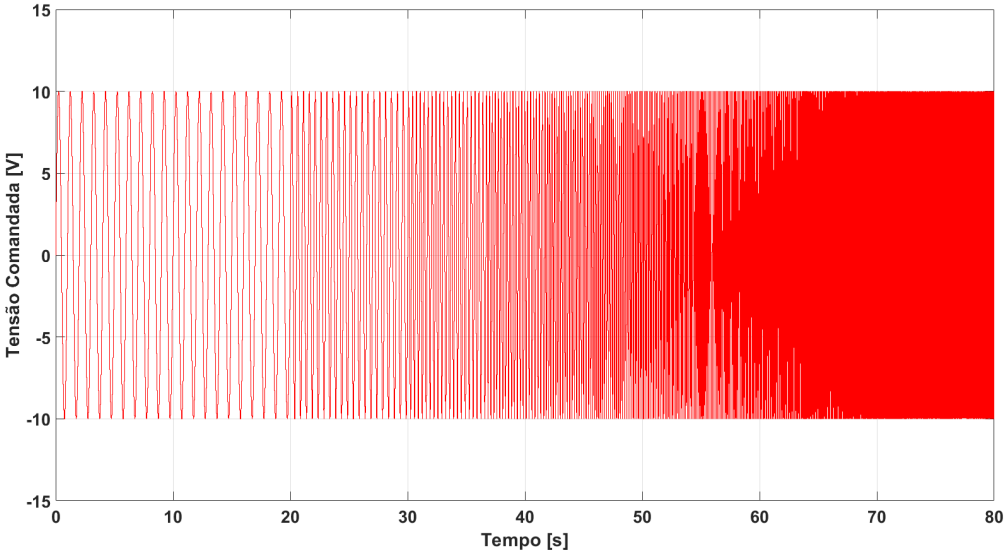

FIGURA 3.6 – Sinal de excitação *chirp* - Exemplo até $30Hz$.

3.3.3 Método de Identificação: Output Error Method

O método de identificação adotado neste trabalho é o *Output Error Method* (método de erro de saída), amplamente utilizado na estimação de sistemas lineares e não lineares (RAMACHANDRAN *et al.*, 1982; SOARES, 2020). Esse método busca minimizar o erro quadrático entre a saída medida e a saída simulada pelo modelo para um dado vetor de parâmetros $\vec{\Theta}$:

$$J(\vec{\Theta}) = \sum_{t=1}^{N} \left(y(t) - \hat{y}(t; \vec{\Theta}) \right)^2, \tag{3.12}$$

onde $y(t)$ representa a saída medida e $\hat{y}(t; \vec{\Theta})$ a saída estimada pelo modelo. O objetivo é encontrar $\hat{\vec{\Theta}}$ que minimize $J(\vec{\Theta})$.

O processo de identificação segue as seguintes etapas:

1. Definição da estrutura do modelo com base na física do sistema;
2. Coleta dos dados de entrada e saída do experimento com sinal *chirp*;
3. Estimação dos parâmetros via minimização numérica;
4. Avaliação de convergência e coerência física dos parâmetros obtidos.

3.3.4 Medida das Saídas

Como mencionado na Seção 3.1.1, utiliza-se um *Encoder Magnético Absoluto* para a medição da posição angular do eixo do motor. Esse tipo de sensor fornece uma leitura digital direta da posição absoluta, eliminando a necessidade de calibração inicial ou contagem incremental, o que contribui para maior precisão e confiabilidade na realimentação de posição.

A estimativa da velocidade é realizada por meio da aplicação de um filtro derivativo sobre o sinal de posição. Esse método permite obter uma aproximação da velocidade angular em tempo real, embora possa amplificar componentes de ruído de alta frequência, um fenômeno mitigado pela escolha adequada da largura de banda do filtro.

O dispositivo de recebimento também oferece suporte à utilização de sensores que efetuam a medição direta da velocidade no eixo do motor, como o tacômetro. Alternativamente, podem ser empregados sensores baseados em efeito *Hall* ou encoders incrementais de alta resolução, dependendo dos requisitos de desempenho do sistema.

Em aplicações que demandem controle em múltiplos níveis hierárquicos, como aquelas que incluem uma malha interna de corrente, o dispositivo prevê a integração de um sensor de corrente. Esse sensor possibilita o monitoramento em tempo real da corrente de armadura, permitindo a implementação de estratégias de controle mais precisas e robustas.

A taxa de aquisição dos sinais de posição e velocidade é de 1 kHz, o que se mostra adequado para a maioria das aplicações de controle de velocidade e posição em motores de corrente contínua. No entanto, para medições de corrente, recomenda-se uma taxa de amostragem superior, a ser especificada conforme as exigências dinâmicas do projeto de controle.

3.3.5 Extração e Validação dos Parâmetros Identificados

Após a estimação, procede-se à extração e validação dos parâmetros obtidos. Primeiramente, os valores estimados são comparados com os dados de catálogo do motor e com referências da literatura. Em seguida, realiza-se a validação cruzada com novos conjuntos de dados experimentais.

3.4 Sintonia Automática do Controlador PI-D

Um dos principais objetivos do dispositivo de recebimento é fornecer suporte ao desenvolvimento e à validação de uma malha de controle do tipo PI-D, bem como permitir a análise da resposta do sistema a diferentes valores de referência (*setpoints*). Além disso,

o dispositivo foi concebido de forma a possibilitar a implementação e o teste de outras configurações de controladores, conforme as necessidades específicas de cada aplicação.

Para o projeto e a avaliação do desempenho do controlador, o dispositivo considera como parâmetros de entrada os principais requisitos de desempenho e robustez do sistema, definidos a seguir:

- **Tempo de subida (t_r):** intervalo de tempo necessário para que a resposta do sistema transite de 10% a 90% do valor final de regime permanente, após a aplicação de um degrau na entrada. Esse parâmetro indica a rapidez com que o sistema responde a uma variação de referência (OGATA, 2010; NISE, 2011).
- **Sobressinal (M_p):** valor máximo da resposta do sistema que excede o valor de regime permanente, normalmente expresso em percentual. O sobressinal reflete a tendência do sistema a ultrapassar o valor desejado antes de se estabilizar (DORF; BISHOP, 2017).
- **Valor de assentamento (δ_s):** resposta dentro de uma faixa predefinida em torno do valor final (geralmente $\pm 2\%$ ou $\pm 5\%$), sem ultrapassá-la novamente. Esse parâmetro indica o quão rapidamente o sistema alcança e mantém a estabilidade (OGATA, 2010).

Outros requisitos de desempenho, como a **banda passante do sistema**, a **margem de fase** e a **margem de ganho**, também podem ser incorporados ao processo de projeto do controlador. Entretanto, no escopo deste trabalho, tais parâmetros foram desconsiderados, com foco exclusivo nos critérios de desempenho temporal.

Para o processo de sintonia do controlador, definiu-se uma *função custo* a ser minimizada que estabelece uma relação ponderada entre os requisitos de desempenho do sistema e seus respectivos pesos de importância, definidos pelo projetista. Essa função tem por objetivo quantificar, em um único índice, o desvio entre o desempenho desejado e o desempenho obtido pelo sistema controlado, permitindo a otimização automática dos parâmetros do controlador (ÅSTRöM; HäGGLUND, 2008; OGATA, 2010).

A função custo $J_{PID}(\vec{K})$ é expressa como:

$$J_{PID}(\vec{K}) = w_1 \cdot (t_r - t_{r_{desired}}) + w_2 \cdot (M_p - M_{p_{desired}}) + w_3 \cdot \left(\frac{\delta_s}{\delta_{s_{desired}}} - 1 \right), \qquad (3.13)$$

Onde $\vec{K} = [K_p, K_i, K_d]$ representa o vetor de ganhos do controlador PI-D a ser otimizados. Os coeficientes w_1, w_2 e w_3 correspondem aos **pesos de importância** atribuídos a cada critério de desempenho, ajustados pelo usuário conforme as prioridades de projeto.

Para este trabalho, os valores adotados foram: $w_1 = 50$, $w_2 = 100$ e $w_3 = 25$. A escolha deste conjunto de pesos reflete a enfase no assentamento final da solução, seguido pelo tempo de subida, podendo tolerar algum sobressinal. Os valores adotados foram determinados com base em experimentações preliminares e na literatura de controle clássico (ÅSTRöM; HäGGLUND, 2008).

Vale ressaltar que cada termo da função custo $J_{PID}(\vec{K})$ é computado apenas quando ocorre violação do requisito de desempenho previamente estabelecido. Dessa forma, a função custo é definida: $J_{PID}(\vec{K}) \in \mathbb{R}_0^+$. Os requisitos desejados foram fixados como: $t_{r_{desired}} = 0{,}2\ s$, $M_{p_{desired}} \leq 10\%$ e $\delta_{s_{desired}} \leq 2\%$, valores também baseados em experimentações preliminares.

3.5 Métodos de Otimização

Foram empregados dois métodos de otimização distintos, selecionados de acordo com as características das funções custo apresentadas nas Seções 3.4 e 3.3.3. O primeiro método baseia-se no algoritmo de **Nelder–Mead**, enquanto o segundo utiliza a técnica de **Programação Quadrática Sequencial** (*Sequential Quadratic Programming* – SQP). Ambos os métodos têm como objetivo minimizar funções custo não lineares que representam o erro entre o comportamento desejado e o comportamento obtido do sistema.

3.5.1 Método de Nelder–Mead

O método de Nelder–Mead é um procedimento iterativo de busca direta, proposto originalmente em 1965, que busca o mínimo local de uma função escalar sem a necessidade de derivadas (LAGARIAS *et al.*, 1998a). Ele opera sobre um conjunto de $n + 1$ pontos denominado *simplex*, que define um poliedro no espaço de parâmetros $\mathbb{R}^n$. A cada iteração, o simplex é atualizado por meio de operações geométricas que exploram o espaço de busca: reflexão, expansão, contração e redução.

A regra de atualização pode ser expressa de forma geral como:

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_r = \mathbf{x}_c + \alpha(\mathbf{x}_c - \mathbf{x}_h), & \text{(reflexão)} \\ \mathbf{x}_e = \mathbf{x}_c + \gamma(\mathbf{x}_r - \mathbf{x}_c), & \text{(expansão)} \\ \mathbf{x}'_c = \mathbf{x}_c + \rho(\mathbf{x}_h - \mathbf{x}_c), & \text{(contração)} \\ \mathbf{x}_i = \mathbf{x}_l + \sigma(\mathbf{x}_i - \mathbf{x}_l), & \text{(redução)} \end{cases} \tag{3.14}$$

onde:

- $\mathbf{x}_h$ é o pior ponto (com maior valor de função custo);
- $\mathbf{x}_l$ é o melhor ponto (com menor valor de função custo);
- $\mathbf{x}_c$ é o centróide dos pontos exceto $\mathbf{x}_h$;
- α, γ, ρ, e σ são coeficientes de reflexão, expansão, contração e redução, tipicamente definidos como $\alpha = 1$, $\gamma = 2$, $\rho = 0{,}5$ e $\sigma = 0{,}5$ (LAGARIAS *et al.*, 1998b).

O processo iterativo é conduzido até que a variação entre os vértices do simplex e/ou os valores da função custo atinjam um limiar de convergência pré-definido. Por não depender de gradientes, esse método é adequado para problemas de identificação de parâmetros em modelos físicos, nos quais a função custo pode ser não diferenciável ou apresentar ruído numérico.

Este método foi empregado na sintonia automática do controlador PI-D, conforme descrito na Seção 3.4. A Tabela 3.2 demonstra o problema de otimização da função custo. OS valores iniciais dos ganhos do controlador foram definidos ad-hoc como sendo $K_p = 1$, $K_i = 1$ e $K_d = 0{,}1$.

TABELA 3.2 – Problema de Otimização para Sintonia do Controlador PI-D

	Variável	Descrição	Quantidade
Minimizar	$J_{PID}(\vec{K})$	Função custo do controlador PI-D	1
Em função de	K_p	Ganho Proporcional	1
	K_i	Ganho Integral	1
	K_d	Ganho Derivativo	1
	$\vec{K}$	Total de variáveis do problema	3

Os critérios de convergência adotados foram: (i) variação máxima entre os vértices do simplex menor que 10^{-3}; (ii) valor da função custo menor que 10^{-3}; (iii) número máximo de iterações igual a 1000.

3.5.2 Método de Programação Quadrática Sequencial (SQP)

O método de Programação Quadrática Sequencial (SQP) é uma técnica de otimização não linear baseada em gradientes, amplamente utilizada para resolver problemas com restrições de igualdade e desigualdade (NOCEDAL; WRIGHT, 2006). O SQP busca o mínimo de uma função objetivo $f(\mathbf{x})$ sujeita a restrições de igualdade $h_i(\mathbf{x}) = 0$ e desigualdade $g_j(\mathbf{x}) \leq 0$, conforme o problema geral:

$$\min_{\mathbf{x}} \; f(\mathbf{x}) \quad \text{sujeito a} \quad \begin{cases} h_i(\mathbf{x}) = 0, & i = 1, \ldots, m_e, \\ g_j(\mathbf{x}) \leq 0, & j = 1, \ldots, m_i. \end{cases} \tag{3.15}$$

A cada iteração k, o método resolve um subproblema de programação quadrática (QP) que lineariza as restrições e aproxima a função objetivo por uma expansão quadrática de Taylor:

$$\min_{\mathbf{d}_k} \; \nabla f(\mathbf{x}_k)^\top \mathbf{d}_k + \frac{1}{2}\mathbf{d}_k^\top \mathbf{H}_k \mathbf{d}_k \quad \text{sujeito a} \quad \begin{cases} \nabla h_i(\mathbf{x}_k)^\top \mathbf{d}_k + h_i(\mathbf{x}_k) = 0, \\ \nabla g_j(\mathbf{x}_k)^\top \mathbf{d}_k + g_j(\mathbf{x}_k) \leq 0. \end{cases} \tag{3.16}$$

O vetor $\mathbf{d}_k$ representa a direção de busca, e $\mathbf{H}_k$ é uma aproximação da matriz Hessiana da função Lagrangiana associada. A solução do subproblema fornece uma direção de descida que é utilizada para atualizar o vetor de parâmetros:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{d}_k, \tag{3.17}$$

em que λ_k é o passo de busca obtido por uma estratégia de *line search* que assegura a convergência. Este método foi empregado na identificação do sistema, conforme descrito na Seção 3.3.3. A Tabela 3.3 demonstra o problema de otimização da função custo. Os valores iniciais dos parâmetros do motor foram os definidos na Tabela 3.1.

TABELA 3.3 – Problema de Otimização para Identificação

	Variável	Descrição	Quantidade
Minimizar	$J(\vec{\Theta})$	Função custo da Identificação	1
Em função de	$4{,}640 \leq R \leq 6{,}960$	Resistência de Armadura	1
	$0{,}071 \leq K_t \leq 0{,}087$	Constante de Torque	1
	$1{,}557 \leq J_m \leq 1{,}903$	Inércia (10^{-4})	1
	$0{,}222 \leq \tau \leq 0{,}272$	Constante de Tempo	1
	$\vec{\Theta}$	Total de variáveis do problema	4

3.6 Implementação e Testes Experimentais

O código da malha de controle foi embarcado no microcontrolador do dispositivo, e pode ser encontrado no Apêndice A.

Após a aquisição dos dados experimentais, as etapas de identificação de parâmetros e de sintonia do controlador são realizadas de forma *offline*, utilizando o ambiente MA-

TLAB. Essa escolha deve-se ao fato de que as técnicas implementadas, demandam recursos computacionais significativos e maior flexibilidade na análise dos resultados, o que torna o processamento em um ambiente de simulação mais apropriado.

Em etapas futuras, pretende-se incorporar ao sistema recursos para identificação e sintonia *online*, ou seja, em tempo real, diretamente no dispositivo embarcado. Entre as técnicas que podem ser adotadas para essa finalidade, destacam-se:

- **Método de Identificação Recursiva** (*Recursive Least Squares* – RLS): permite a atualização contínua dos parâmetros do modelo à medida que novos dados são adquiridos, possibilitando adaptação a variações de carga ou desgaste do atuador;
- **Filtro de Kalman Estendido** (EKF): possibilita a estimação simultânea de estados e parâmetros, mesmo na presença de ruído de medição;
- **Controle Adaptativo Baseado em Modelo de Referência** (MRAC): ajusta os parâmetros do controlador de forma automática, buscando que a resposta do sistema siga um comportamento de referência pré-definido;
- **Controle Preditivo Autoajustável** (*Self-Tuning Predictive Control*): utiliza modelos internos atualizados em tempo real para antecipar o comportamento do sistema e recalcular os parâmetros de controle;
- **Algoritmos de Otimização Heurística Online**, como *Particle Swarm Optimization* (PSO) ou *Genetic Algorithms* (GA): podem ser aplicados em controladores embarcados para sintonia automática, equilibrando desempenho e robustez.

A implementação dessas técnicas permitirá a realização de experimentos com identificação e ajuste dinâmico dos parâmetros de controle em tempo real, ampliando as capacidades de pesquisa e desenvolvimento do dispositivo proposto.

4 Resultados

Nesta seção são apresentados os resultados obtidos no processo de identificação dos parâmetros do modelo do motor de corrente contínua e na sintonia do controlador do tipo PI-D. Todos os experimentos descritos foram conduzidos em ambiente de simulação, utilizando-se a plataforma *MATLAB/Simulink*. Assim, assume-se que as variáveis corrente de armadura (I) e velocidade angular (ω) estão diretamente disponíveis para medição e livre de ruídos o que permite avaliar isoladamente o desempenho das metodologias empregadas.

A geração dos dados utilizados na identificação foi realizada por meio de uma Simulação de Monte Carlo, considerando uma entrada do tipo degrau de 10 V aplicada durante um intervalo de 1 s. O objetivo desse procedimento foi reproduzir um conjunto amplo e realista de possíveis condições operacionais do motor, considerando variações simultâneas nos parâmetros elétricos e mecânicos. A Figura 4.1 ilustra o conjunto de respostas obtidas para as diferentes combinações de parâmetros utilizadas.

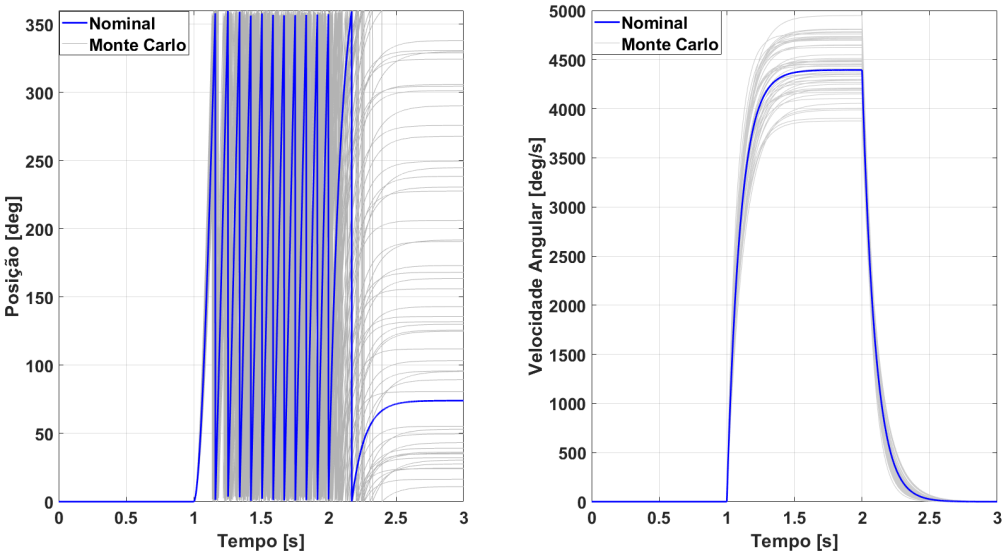

FIGURA 4.1 – Resultados das simulações de Monte Carlo para variações nos parâmetros do motor.

A partir das simulações realizadas, foi selecionada uma realização específica para servir como vetor de saída no processo de estimação dos parâmetros. A Tabela 4.1 apresenta

os valores nominais, os valores modificados empregados na simulação escolhida e a variação percentual correspondente, evidenciando a magnitude das perturbações impostas ao modelo.

TABELA 4.1 – Valores nominais, alterados e variação percentual dos parâmetros do motor utilizados na identificação.

Variável	**Valor Nominal**	**Valor Alterado**	**Unidade**	**Variação (%)**
R	5,800	6,444	Ω	+11,10%
K_t	0,079	0,075	N·m/A	−5,06%
J_m	$1{,}730\times10^{-4}$	$1{,}842\times10^{-4}$	kg·m^2	+6,47%
K_e	0,079	0,075	V/(rad/s)	−5,06%
B_m	700×10^{-6}	681×10^{-6}	kg·m^2/s	−2,70%
$\tau = \dfrac{J_m}{B_m}$	0,247	0,270	s	+9,31%

4.1 Identificação do Modelo do Motor

Com base no vetor de saída selecionado, procedeu-se com a estimação dos parâmetros do modelo. O vetor inicial de parâmetros para o algoritmo de otimização foi definido com base nos valores nominais apresentados na Tabela 4.1. A Tabela 4.2 apresenta os valores utilizados na simulação, os valores estimados pelos métodos de otimização e o erro percentual associado a cada parâmetro.

As Figuras 4.2 a 4.4 apresentam a comparação entre a resposta do modelo ajustado com os parâmetros estimados e a resposta obtida na simulação de Monte Carlo selecionada. Essa análise permite avaliar a capacidade do modelo identificado de reproduzir o comportamento dinâmico do sistema, servindo como validação preliminar do processo de estimação.

TABELA 4.2 – Parâmetros do motor estimados e erros percentuais recalculados.

Parâmetro	**Valor Alterado**	**Valor Estimado**	**Erro (%)**
R	6,444	6,218	−3,51%
K_t	$7{,}485 \times 10^{-4}$	$7{,}462 \times 10^{-4}$	1,00%
J_m	$1{,}842 \times 10^{-4}$	$1{,}903 \times 10^{-4}$	+3,31%
K_e	$7{,}485 \times 10^{-4}$	$7{,}462 \times 10^{-4}$	1,00%
B_m	681×10^{-6}	708×10^{-6}	+3,97%
$\tau = \frac{J_m}{B_m}$	0,270	0,262	−2,96%

O parâmetro de resistência de armadura (R) apresentou erro de −3,51%, indicando uma subestimação moderada, que pode explicar a diferença da resposta da corrente demonstrada na Figura 4.4. A discretização do sinal medido, induzida no modelo pelos sensores, pode contribuir para este pequeno erro de estimativa.

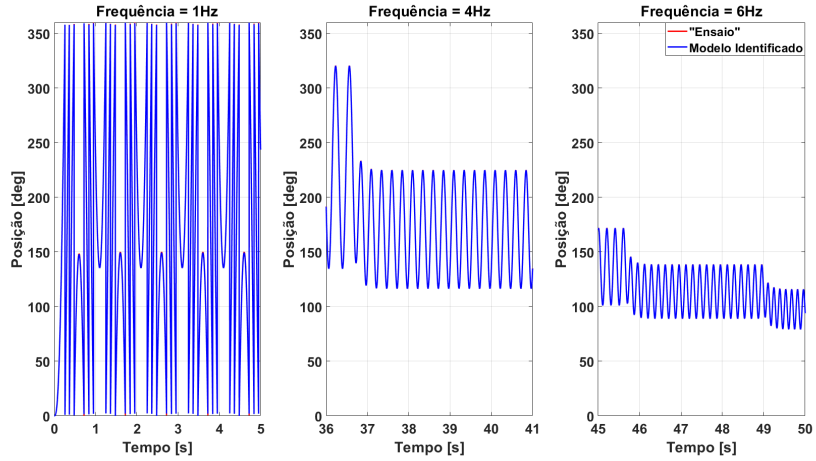

FIGURA 4.2 – Comparação entre a resposta de posição do modelo identificado e a resposta medida.

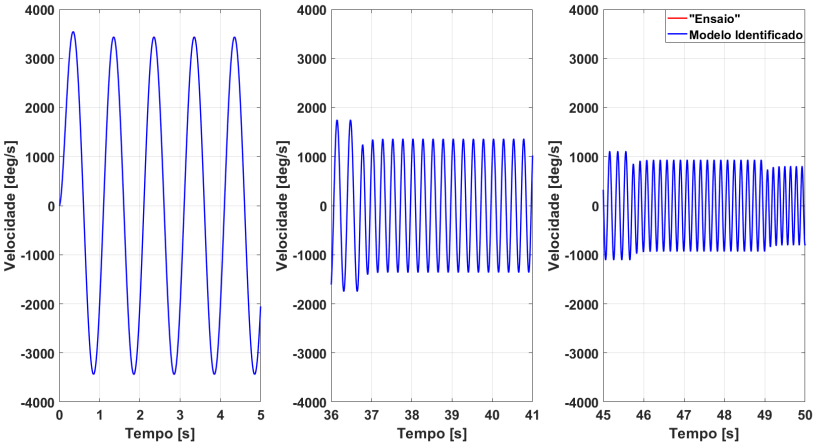

FIGURA 4.3 – Comparação entre a resposta de velocidade do modelo identificado e a resposta medida.

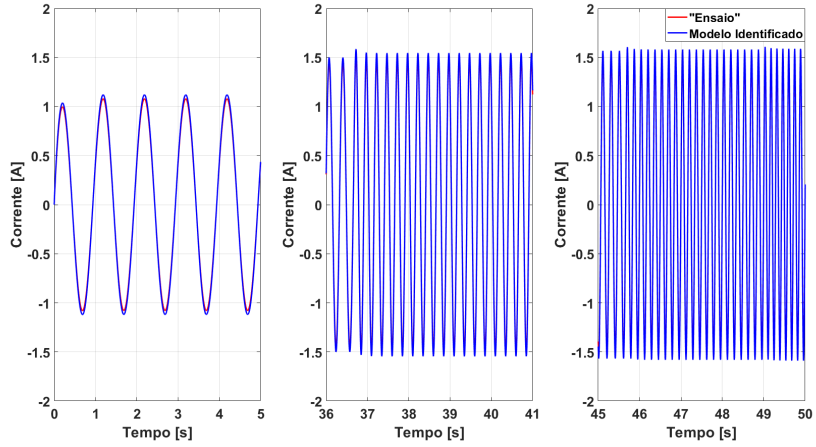

FIGURA 4.4 – Comparação entre a resposta de corrente do modelo identificado e a resposta medida.

Os parâmetros eletromecânicos K_t e K_e apresentaram erro de aproximadamente 1,00%. A inércia do rotor (J_m) apresentou erro de +3,31%, enquanto o coeficiente de atrito viscoso (B_m) apresentou erro de +3,97%. Esses valores, também inferiores a 5%, estão dentro

do esperado para parâmetros mecânicos, que em geral são mais difíceis de identificar precisamente devido à sua menor influência direta nos sinais medidos. A proximidade dos erros entre J_m e B_m indica consistência do modelo mecânico estimado.

A constante de tempo mecânica $\tau = J_m/B_m$ apresentou erro de $-2{,}96\%$, coerente com as variações observadas individualmente em J_m e B_m. Como τ é uma grandeza derivada, o resultado confirma a consistência interna da identificação, uma vez que os erros acumulados permanecem baixos.

No conjunto, os erros percentuais obtido, inferiores a 4%, indicam que o processo de identificação foi bem-sucedido.

A evolução da função custo ao longo das iterações do processo de otimização é ilustrada na Figura 4.5, evidenciando a convergência do algoritmo empregado.

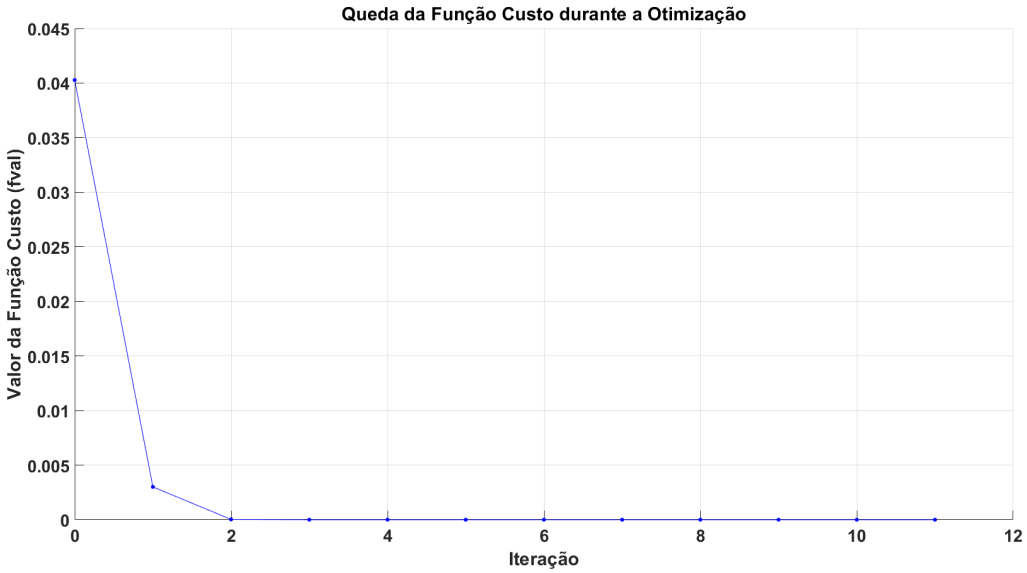

FIGURA 4.5 – Evolução da função custo durante o processo de otimização.

4.1.1 Validação do Modelo por Meio das Métricas RMSE e Goodness of Fit

A validação do modelo identificado foi realizada com base na comparação entre as saídas medidas do sistema, denotadas por $Z(k)$, e as saídas geradas pelo modelo identificado, denotadas por $Y(k)$. Para essa avaliação, utilizaram-se duas métricas amplamente aceitas na literatura de identificação de sistemas: o *Root Mean Square Error* (RMSE) e a métrica *Goodness of Fit* (FIT).

Métrica RMSE

O RMSE quantifica o erro médio quadrático entre a resposta medida e a resposta simulada pelo modelo, sendo definido como:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(Z(k) - Y(k)\right)^2}. \tag{4.1}$$

Em termos qualitativos, utiliza-se como referência prática:

- RMSE $< 2\%$ da amplitude da variável: **Excelente aderência**;
- $2\% \leq$ RMSE $< 5\%$: **Boa aderência**;
- RMSE $\geq 5\%$: **Aderência insuficiente**.

Métrica Goodness of Fit (FIT)

A métrica FIT, também chamada de *Nível de Ajuste*, mede o quanto o modelo explica a variação dos dados medidos. É definida como:

$$\text{FIT} = \left(1 - \frac{\|Z - Y\|}{\|Z - \bar{Z}\|}\right) \times 100, \tag{4.2}$$

onde $\bar{Z}$ é o valor médio do sinal medido.

O valor de FIT é expresso em porcentagem e pode ser interpretado da seguinte maneira:

- FIT $\geq 90\%$: **Excelente ajuste**;
- $75\% \leq$ FIT $< 90\%$: **Bom ajuste**;
- $50\% \leq$ FIT $< 75\%$: **Ajuste moderado**;
- FIT $< 50\%$: **Ajuste insuficiente**.

Resultados Obtidos

Os resultados apresentados na Tabela 4.3 evidenciam que o modelo identificado apresenta elevada capacidade de representar as dinâmicas do motor de corrente contínua. Os valores de RMSE expressos em porcentagem indicam erros relativos reduzidos, especialmente para as variáveis de posição (0,49%) e velocidade (1,30%), valores considerados muito baixos diante das amplitudes típicas desses sinais. Embora a variável de corrente

apresente um erro maior (3,60%), este ainda se mantém dentro de uma faixa aceitável, sobretudo devido à maior sensibilidade da corrente a variações paramétricas e não linearidades do sistema.

A métrica *Goodness of Fit* (FIT) confirma o bom desempenho do modelo identificado. Os valores superiores a 99% para posição e velocidade demonstram praticamente uma coincidência entre as respostas simulada e de referência. O FIT de 96,35% para a corrente, ainda que inferior aos demais, indica uma representação satisfatória da dinâmica elétrica do motor, reforçando a consistência do processo de identificação.

TABELA 4.3 – Valores de RMSE e FIT obtidos na validação do modelo identificado.

Variável	**RMSE (%)**	**FIT (%)**
Posição	0.49	99.91
Corrente	3.60	96.35
Velocidade	1.30	99.95

4.2 Sintonia do Controlador PI-D

Com base no modelo identificado, procedeu-se à sintonia do controlador PI-D utilizando o método descrito nas Seções 3.4 e 3.5.1. A Tabela 4.4 apresenta os requisitos de desempenho estabelecidos para o sistema controlado. A Figura 4.6 ilustra a resposta do sistema controlado com o controlador PI-D sintonizado, evidenciando o atendimento aos requisitos especificados. Os ganhos otimizados do controlador PI-D foram encontrados como: $K_p = 1{,}267$, $K_i = 1{,}183$ e $K_d = 0{,}053$.

TABELA 4.4 – Requisitos de desempenho para o sistema controlado.

Parâmetro	**Requisito**	**Valor otimizado**
Sobressinal Máximo (%)	< 10	0
Valor de Assentamento (deg)	$45 \pm 2\%$	45
Tempo de Subida (s)	0,100	0,084

Os resultados obtidos indicam que todos os requisitos de desempenho foram plenamente atendidos no modelo nominal, demonstrando a eficácia do método de sintonia empregado. O controlador PI-D projetado apresentou resposta rápida e precisa, com tempo de subida inferior ao especificado e ausência de sobressinal, característica desejável em aplicações que demandam elevada precisão, estabilidade e segurança operacional.

Ao avaliar a robustez do controlador PI-D por meio de simulações de Monte Carlo, observou-se uma degradação significativa do desempenho em alguns cenários. Especificamente, foram identificados casos em que o sistema apresentou sobressinal e tempos de assentamento superiores aos limites estabelecidos, conforme ilustrado na Figura 4.7,

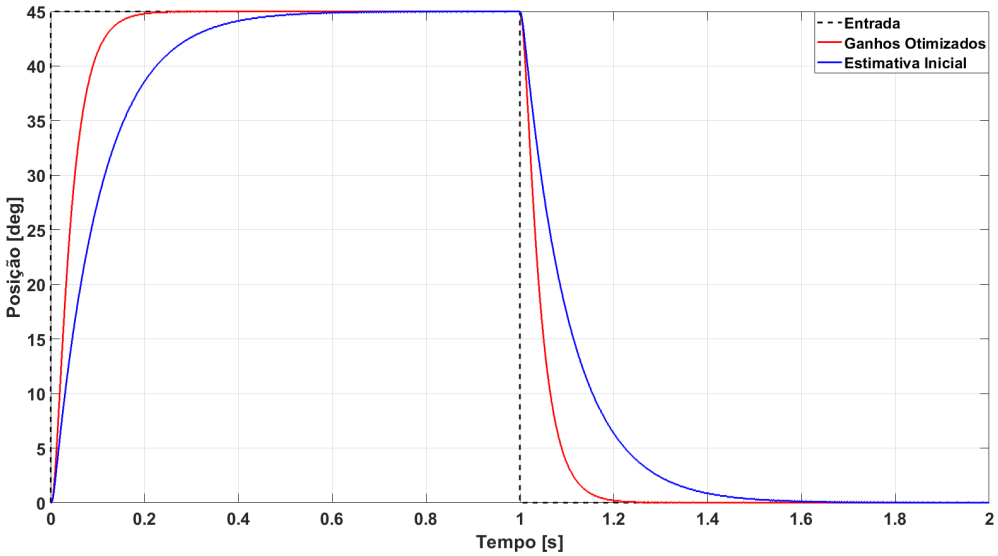

FIGURA 4.6 – Resposta do sistema controlado com o controlador PI-D sintonizado.

também houve casos de divergência numérica da solução. Esses resultados demonstram que, embora o controlador PI-D seja adequado para o modelo nominal identificado, sua robustez frente às incertezas paramétricas é limitada.

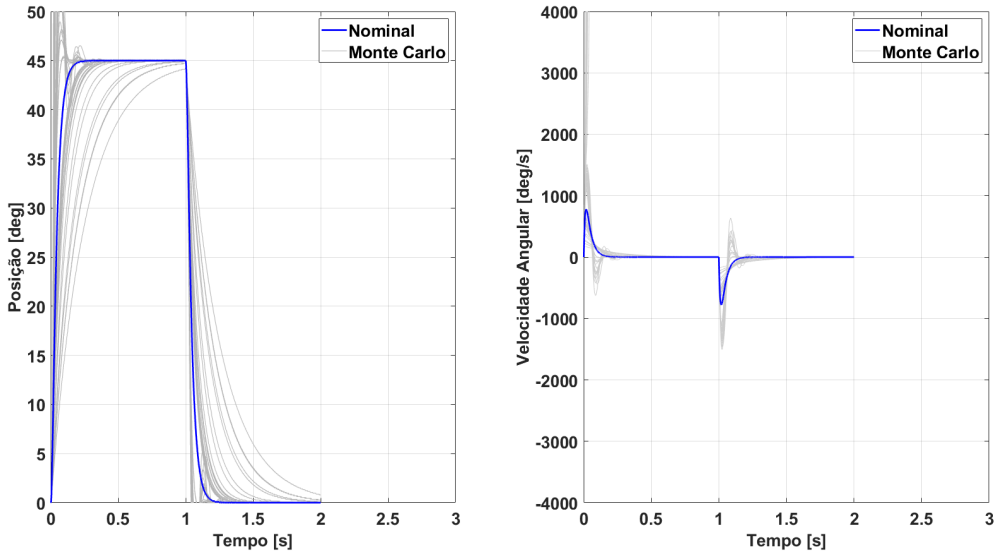

FIGURA 4.7 – Resposta do sistema controlado com o controlador PI-D em diferentes simulações de Monte Carlo.

Diante dessa limitação, procedeu-se a uma nova etapa de sintonia, na qual o ponto inicial do algoritmo de otimização foi ajustado manualmente. Essa intervenção permitiu ao processo de otimização explorar regiões mais adequadas do espaço de busca e, consequentemente, identificar um conjunto de ganhos mais robustos. Os valores obtidos para o controlador ajustado foram: $K_p = 7{,}583$, $K_i = 5{,}583$ e $K_d = 0{,}233$.

Com esses novos ganhos, observou-se que para a maioria das simulações o controlador

continuou atendendo aos requisitos de desempenho no modelo nominal. A Figura 4.8 demonstra essa melhoria, evidenciando respostas mais consistentes nas diversas realizações de Monte Carlo. Este resultado revela não apenas a necessidade de ajustes no processo de sintonia, mas também a importância da intervenção de um engenheiro no processo, mesmo quando se recorre a métodos automáticos de otimização.

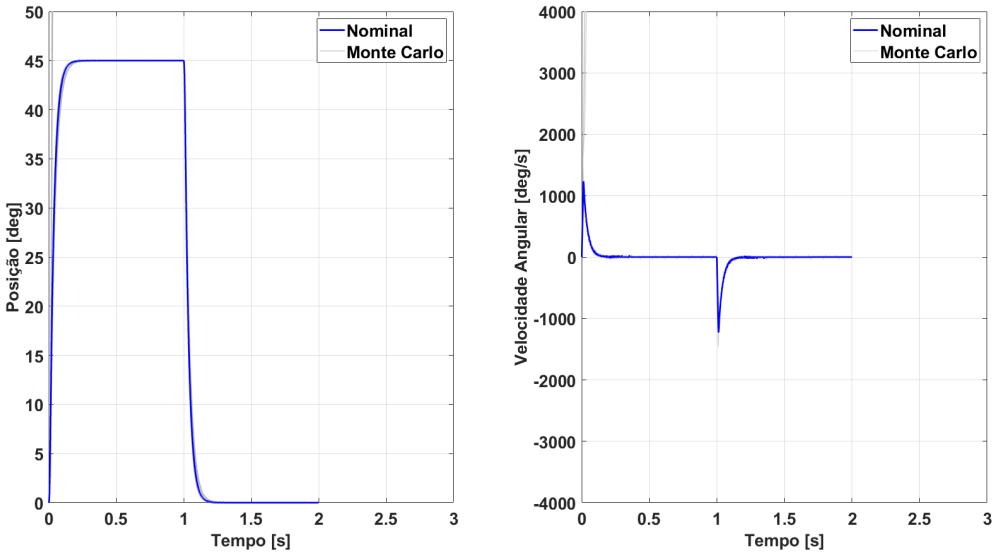

FIGURA 4.8 – Resposta do sistema controlado com o controlador PI-D robusto em diferentes simulações de Monte Carlo.

5 Conclusão

Este trabalho teve como objetivo principal o desenvolvimento, implementação e validação de uma metodologia para a identificação paramétrica e projeto de controladores para um motor de corrente contínua, considerando um ambiente inicial de simulação e visando, em etapas posteriores, sua aplicação em um dispositivo físico de recebimento. A abordagem adotada envolveu técnicas clássicas de modelagem dinâmica, métodos de estimação de parâmetros baseados na formulação *Output Error*, e a utilização de estratégias de otimização para a sintonia de um controlador do tipo PI-D. Ao longo do desenvolvimento, buscou-se construir uma base sólida para futura implementação experimental e validação em um sistema real.

5.1 Síntese dos Resultados Obtidos

A metodologia proposta demonstrou ser eficaz para a identificação dos parâmetros do modelo do motor. A simulação de Monte Carlo permitiu gerar um conjunto abrangente de cenários, representando variações plausíveis nos parâmetros físicos do motor, como resistência de armadura, constante de torque e inércia. A partir desses dados, foi possível aplicar os métodos de estimação e obter parâmetros que reproduzem com elevada fidelidade a dinâmica do sistema de referência.

Os resultados da validação, baseados nas métricas RMSE e *Goodness of Fit* (FIT), indicaram que o modelo identificado é capaz de representar com elevada precisão o comportamento do motor no domínio temporal. Foram observados valores de RMSE inferiores a 4% e valores de FIT superiores a 96% em todas as variáveis analisadas, incluindo posição, velocidade e corrente. Tais indicadores confirmam que o modelo obtido é consistente e adequado para utilização em tarefas de síntese e avaliação de controladores.

A etapa de sintonia do controlador PI-D, formulada como um problema de otimização, também apresentou resultados satisfatórios. A função custo proposta permitiu relacionar diretamente os critérios de desempenho desejados aos parâmetros do controlador.

5.2 Discussão das Limitações

Apesar dos avanços obtidos, algumas limitações inerentes à abordagem adotada devem ser destacadas. Primeiramente, todas as análises foram conduzidas em ambiente simulado. Embora o modelo utilizado seja fisicamente coerente e parametrizado com valores realistas, ele não incorpora fenômenos presentes em sistemas reais, tais como:

- ruídos de medição em sensores de posição, corrente e velocidade;
- não linearidades dos atuadores, incluindo saturação de tensão e corrente;
- atrito do tipo Coulomb e efeitos de histerese;
- variações térmicas na resistência do enrolamento;
- atrasos de processamento e quantização digital.

Essas limitações implicam que, embora o modelo identificado seja adequado para fins de simulação e projeto preliminar, sua validade deve ser reavaliada em condições experimentais reais. Além disso, a identificação paramétrica foi realizada de forma *offline*, ou seja, utilizando dados previamente coletados, sem mecanismos de adaptação em tempo real. Esta etapa *offline* é um trabalho adicional na automatização do processo.

5.3 Contribuições do Trabalho

Dentre as principais contribuições deste trabalho, destacam-se:

- o desenvolvimento de uma estrutura modular para identificação e validação de modelos de motores DC;
- a aplicação de técnicas de otimização para a sintonia sistemática de controladores do tipo PI-D;
- a formulação de métricas de desempenho e validação consistentes, permitindo avaliar quantitativamente a qualidade do modelo obtido;
- a preparação de uma base metodológica que possibilita futuras extensões envolvendo implementação em hardware e controle em tempo real.

Essas contribuições constituem um conjunto sólido de ferramentas e procedimentos que podem ser aplicados tanto em ambientes acadêmicos quanto industriais, especialmente em tarefas relacionadas à automação, controle e identificação de sistemas eletromecânicos.

5.4 Perspectivas para Trabalhos Futuros

Com base nas limitações observadas e no potencial de expansão da metodologia desenvolvida, diversas linhas de pesquisa e aprimoramento podem ser exploradas:

Implementação Experimental

A validação do modelo e do controlador em um ambiente físico constitui a extensão natural deste trabalho. A inclusão de sensores reais, amplificadores de potência e aquisição de dados permitirá avaliar a robustez da metodologia frente às incertezas e ruídos presentes no sistema.

Identificação On-line

A utilização de métodos adaptativos permitiria que o modelo acompanhasse variações dinâmicas do sistema. Podem ser investigados:

- Mínimos Quadrados Recursivos (RLS);
- RLS com fator de esquecimento ou adaptação variável;
- Algoritmos do tipo LMS (*Least Mean Squares*);
- Métodos baseados em Filtro de Kalman estendido ou adaptativo;
- Redes neurais para identificação em tempo real;
- Técnicas de controle adaptativo direto ou indireto.

Controle Avançado

Para sistemas mais exigentes, controladores de maior desempenho podem ser implementados, como:

- Controle ótimo (LQR, LQG);
- Controle robusto (H_∞);
- Controle preditivo (MPC);
- Controle não linear para compensação de atrito e saturação.

Modelos com Não Linearidades

A inclusão de efeitos como atrito estático, saturação magnética e dinâmica térmica pode resultar em modelos ainda mais representativos da planta real.

5.5 Considerações Finais

Em síntese, a metodologia desenvolvida neste trabalho mostrou-se eficaz para a identificação e validação de modelos de motores DC em ambiente simulado, bem como para a sintonia sistematizada de controladores clássicos. As métricas de desempenho obtidas indicam elevada correspondência entre o modelo identificado e o sistema de referência, o que demonstra a qualidade e a consistência da abordagem adotada. Embora ainda seja necessária uma etapa de validação experimental, os resultados alcançados constituem uma base sólida para futuras implementações práticas, além de oferecerem um conjunto de ferramentas que podem ser empregadas e ampliadas em estudos posteriores.

Conclui-se, portanto, que o trabalho atingiu seus objetivos, estabelecendo uma metodologia estruturada e eficaz, e abrindo caminho para aplicações mais avançadas no campo da identificação e controle de sistemas eletromecânicos.

Referências

CAO, Y.; LIU, Y.; PAREDIS, C. J. System-level model integration of design and simulation for mechatronic systems based on sysml. **Mechatronics**, v. 21, n. 6, p. 1063–1075, 2011. ISSN 0957-4158. Available at: https://www.sciencedirect.com/science/article/pii/S095741581100095X.

CARROL, E. **How is Model-Based Systems Engineering Justified?** 2016. INCOSE. Available at: https://www.incose.org/docs/default-source/enchantment/161109-carrolled-howismodel-basedsystemsengineeringjustified-researchreport.pdf?sfvrsn=2\%26sfvrsn=2.

CHABIBI, B. **Model integration approach from SysML to MATLAB/simulink**. 2011. Academia.edu. Available at: https://www.academia.edu/download/78355389-/0ec121c4fa8837bb4d212a830e0d949f7553.pdf.

CICCHETTI, J. C. A. Early validation and verification of system behaviour in model-based systems engineering: A systematic literature review. *In*: **ACM Digital Library**. **Proceedings** [...]. [*S.l.: s.n.*], 2024. Available at: https://dl.acm.org/doi/10.1145/3631976.

DORF, R. C.; BISHOP, R. H. **Modern Control Systems**. 13. ed. [*S.l.*]: Pearson, 2017.

GOODWIN, G. C.; PAYNE, R. L. **Dynamic System Identification: Experiment Design and Data Analysis**. New York: Academic Press, 1977.

GU, P.; CHEN, Z.; ZHANG, L.; ZHANG, Y.; XIE, K.; ZHAO, C.; YE, F.; TAO, Y. X-sem: A modeling and simulation-based system engineering methodology. **Journal of Manufacturing Systems**, v. 74, p. 198–221, 2024. ISSN 0278-6125. Available at: https://www.sciencedirect.com/science/article/pii/S0278612524000232.

LAGARIAS, J. C.; REEDS, J. A.; WRIGHT, M. H.; WRIGHT, P. E. Convergence properties of the nelder–mead simplex method in low dimensions. **SIAM Journal on Optimization**, v. 9, n. 1, p. 112–147, 1998.

LAGARIAS, J. C.; REEDS, J. A.; WRIGHT, M. H.; WRIGHT, P. E. Convergence properties of the nelder-mead simplex method in low dimensions. **SIAM J. Optim.**, v. 9, p. 112–147, 1998. Available at: https://api.semanticscholar.org/CorpusID:9245771.

LJUNG, L. **System Identification: Theory for the User**. 2nd. ed. Upper Saddle River, NJ: Prentice Hall, 1999. ISBN 978-0136566953.

LOPEZ, A. A. V. A review on the application of model-based systems engineering (mbse) to manufacturing and production engineering systems. **ResearchGate**, 2021. Available at: https://www.researchgate.net/publication-/352305625_A_Review_on_Application_of_Model_Based_Systems_Engineering_to_Manufacturing_and_P

MathWorks. **SysML Connector - MATLAB & Simulink**. 2024. MathWorks Documentation. Available at: https://www.mathworks.com/products/sysml.html.

MathWorks. **System Identification Overview**. 2024. MATLAB & Simulink Documentation. Available at: https://www.mathworks.com/help/ident/gs/about-system-identification.html.

NISE, N. S. **Engenharia de Sistemas de Controle**. 6. ed. [*S.l.*]: LTC, 2011.

NOCEDAL, J.; WRIGHT, S. J. **Numerical Optimization**. 2nd. ed. New York: Springer, 2006.

OGATA, K. **Engenharia de Controle Moderno**. 5. ed. [*S.l.*]: Pearson Prentice Hall, 2010.

PAAVLE, T. Short-time chirp excitation for wideband identification of dynamic objects. **Estonian Journal of Engineering**, v. 17, n. 2, p. 169–179, 2011. Available at: https://kirj.ee/public/Engineering/2011/issue_2/eng-2011-2-169-179.pdf.

RAMACHANDRAN, K. S. S.; STOTEN, D. P.; HYDE, R. A. On output-error methods for system identification. **IEEE Transactions on Automatic Control**, v. 27, n. 1, p. 186–191, 1982. Available at: https://ieeexplore.ieee.org/document/1103141.

RIBEIRO, J. M. S.; SANTOS, M. F.; CARMO, M. J.; SILVA, M. F. Comparison of pid controller tuning methods: analytical/classical techniques versus optimization algorithms. *In*: **2017 18th International Carpathian Control Conference (ICCC)**. **Proceedings** [...]. [*S.l.: s.n.*], 2017. p. 533–538.

Scratchapixel. **Mathematical Foundations of Monte Carlo Methods**. 2024. Scratchapixel Online. Available at: https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics-/monte-carlo-methods-mathematical-foundations/probability-distribution-part1.html.

SOARES, I. M. **Aircraft Flight Dynamics Identification Using Subscale Flight Test Methodology**. Dissertation (Dissertation of Master of Science) — Instituto Tecnológico de Aeronáutica, São José dos Campos, 2020. 98f.

UNBEHAUEN, H.; RAO, G. P. Identification of continuous systems. **IEE Proceedings D - Control Theory and Applications**, v. 137, n. 6, p. 349–362, 1990.

University of Michigan. **Activity 6 Part (a): Time-Response Analysis of a DC Motor**. 2024. Control Tutorials for MATLAB and Simulink (CTMS). Available at: https://ctms.engin.umich.edu/CTMS/index.php?aux=Activities_DCmotorA.

VUOJOLAINEN, J.; NEVARANTA, N.; JASTRZEBSKI, R.; PYRHöNEN, O. Comparison of excitation signals in active magnetic bearing system identification. **Modeling, Identification and Control**, v. 38, n. 3, p. 123–133, 2017. Available at: https://www.mic-journal.no/ABS/MIC-2017-3-2.asp.

Wikipedia contributors. **Monte Carlo method**. 2024. `https://en.wikipedia.org/wiki/Monte_Carlo_method`. Acessado em Outubro de 2025. Available at: https://en.wikipedia.org/wiki/Monte_Carlo_method.

ÅSTRöM, K. J.; HäGGLUND, T. **Advanced PID Control**. [*S.l.*]: ISA – The Instrumentation, Systems, and Automation Society, 2008.

Apêndice A - Malha de Controle do Atuador

Listing A.1 – ACT.h

```c
/*
 * ACT.h
 *
 *  Created on: Jul 24, 2025
 *      Author: igorsoares
 */

#ifndef ACT_INC_ACT_H_
#define ACT_INC_ACT_H_

/* Includes */
#include "InputTestTable.h"

/*****************************
 *     LOCAL VARIABLES
 *****************************/
#define SPEED_ESTIMATOR_A0 771.739
#define SPEED_ESTIMATOR_A1 -771.739
#define SPEED_ESTIMATOR_A2 0
#define SPEED_ESTIMATOR_B1 -0.228261
#define SPEED_ESTIMATOR_B2 0
#define SPEED_ESTIMATOR_SAMPLE_TIME 0.001
#define SPEED_ESTIMATOR_ANTIWINDUP 0
#define SPEED_ESTIMATOR_UPPER_SATURATION 0
#define SPEED_ESTIMATOR_LOWER_SATURATION 0
#define PD_P_GAIN 16.0
#define PD_D_GAIN 0.5
#define PD_I_GAIN 1.0
#define PD_SAMPLE_TIME 0.001
```

```
30 #define PD_V 30.0
31
32 /* Z1 State structure */
33 typedef struct {
34     volatile float state_u; /* u previous states */
35     volatile float state_y; /* y previous states */
36     volatile float coef_au[2]; /* Numerator coefficients (u) */
37     volatile float coef_by[2]; /* Denominator coefficients (b[0]
          must be 1) (y) */
38     volatile float superior; /* Anti-WindUp higher limit (if
          enabled) */
39     volatile float inferior; /* Anti-WindUp lower limit (if
          enabled) */
40     volatile unsigned int antiwindup; /* Anti-WindUp Enable [bool]
           */
41 } Z1_state_t;
42
43 /**** ACT Structures ****/
44
45 /* ACT Modes */
46 typedef enum {
47     ACT_STANDBY = 0, ACT_CHIRP = 1, ACT_SETPOINT = 2,
          ACT_INPUTTEST = 3, ACT_AUTOTUNING = 4,
48 } ACT_Mode_t;
49
50 /* ACT Mode Selector */
51 typedef enum {
52     ACT_CMD_STANDBY = 0, ACT_CMD_CHIRP = 1, ACT_CMD_SETPOINT = 2,
          ACT_CMD_AUTOTUNING = 3,
53 } ACT_CmdMode_t;
54
55 /* ACT Chirp state */
56 typedef struct {
57     float frequenciesUnique[50];
58     float currFrequency;
59     int chirpCont;
60     int chirpCounter;
61     int freqCount;
62     int isChirpDone;
63     int isChirpFirstExec;
64 } ACT_ChirpState_t;
```

```
65
66 /* ACT Input Test state */
67 typedef struct {
68     int contInputTestTable;
69     float inputTestTable;
70 } ACT_InputTestState_t;
71
72 /* ACT control inputs */
73 typedef struct {
74     ACT_CmdMode_t modeSel;
75     float posm;
76     float spdm;
77     float curm;
78     float setpoint;
79     float chirpAmplitude;
80     float chirpFreqEnd;
81     float chirpFreqStart;
82     float chirpNumberFrequencies;
83     float chirpFreqOsc;
84 } ACT_Input_t;
85
86 /* ACT state structure */
87 typedef struct {
88     ACT_Mode_t actMode; /* Mode structure [enum] */
89     ACT_ChirpState_t chirpState; /* Chirp States [struct] */
90     ACT_InputTestState_t inputTestState; /* Input Test States [
        struct] */
91
92     Z1_state_t tf_speed_estimator; /* Speed estimator transfer
        function */
93
94     float integral;
95
96     float position_c;
97     float position_r;
98
99     float speed_c;
100    float speed_e;
101    float speed_r;
102 } ACT_State_t;
103
```

```
104 /* ACT control inputs */
105 typedef struct {
106     float Vc;
107     int alive;
108     ACT_Mode_t actMode;
109 } ACT_Output_t;
110 
111 /*************************
112  *     GLOBAL FUNCTIONS
113  *************************/
114 
115 void tf_z1_init(Z1_state_t *state, float a0, float a1, float b1,
116         unsigned int antiwindup, float superior, float inferior);
117 
118 float tf_z1(Z1_state_t *state, float u0);
119 
120 void ACT_StateMachineUpdate(ACT_Input_t const *pActInput,
    ACT_State_t *pActState);
121 
122 void ACT_Update(ACT_Input_t const *pActInput, ACT_State_t *
    pActState,
123         ACT_Output_t *pActOutput);
124 
125 void ACT_Init(ACT_State_t *pActState, ACT_Output_t *pActOutput);
126 
127 #endif /* ACT_INC_ACT_H_ */
```

Listing A.2 – ACT.c

```c
/*
 * ACT.c
 *
 *  Created on: Jul 24, 2025
 *      Author: igorsoares
 */

#include "ACT.h"

/* Declarations */

static float chirp(ACT_Input_t const *pActInput, ACT_State_t *
    pActState);

static float setpoint(ACT_Input_t const *pActInput, ACT_State_t *
    pActState);

static float inputTest(ACT_Input_t const *pActInput, ACT_State_t *
    pActState);

static void ACT_AntiWindUp(ACT_State_t *pActState, ACT_Output_t *
    pActOutput);

static float ACT_Limit(float value, float min, float max);

/* Definitions */

static float ACT_Limit(float value, float min, float max) {
    if (value > max) {
        return max;
    } else if (value < min) {
        return min;
    } else {
        return value;
    }
}

static void ACT_AntiWindUp(ACT_State_t *pActState, ACT_Output_t *
    pActOutput) {
    if (pActOutput->Vc > PD_V) {
```

```
36          pActOutput->Vc = PD_V;
37          if (pActState->position_r > 0.0f) {
38              pActState->integral -= pActState->position_r *
                    PD_SAMPLE_TIME;
39          }
40      } else if (pActOutput->Vc < -PD_V) {
41          pActOutput->Vc = -PD_V;
42          if (pActState->position_r < 0.0f) {
43              pActState->integral -= pActState->position_r *
                    PD_SAMPLE_TIME;
44          }
45      } else {
46          pActOutput->Vc = pActOutput->Vc;
47      }
48  }
49
50
51  static float chirp(ACT_Input_t const *pActInput, ACT_State_t *
        pActState) {
52      return 0.0f;
53  }
54
55  static float setpoint(ACT_Input_t const *pActInput, ACT_State_t *
        pActState) {
56      return pActInput->setpoint;
57  }
58
59  void tf_z1_init(Z1_state_t *state, float a0, float a1, float b1,
60          unsigned int antiwindup, float superior, float inferior) {
61      state->coef_au[0] = a0;
62      state->coef_au[1] = a1;
63
64      state->coef_by[0] = 1;
65      state->coef_by[1] = b1;
66
67      state->state_u = 0;
68      state->state_y = 0;
69
70      state->antiwindup = antiwindup;
71      state->superior = superior;
72      state->inferior = inferior;
```

```
73 }
74
75 static float inputTest(ACT_Input_t const *pActInput, ACT_State_t *
       pActState){
76
77     float setpoint = 0.0f;
78
79     pActState->inputTestState.contInputTestTable++;
80     setpoint = pActState->inputTestState.inputTestTable[pActState
           ->inputTestState.contInputTestTable];
81
82     return setpoint;
83 }
84
85 float tf_z1(Z1_state_t *state, float u0) {
86     register float y;
87
88     y = u0 * state->coef_au[0] + state->state_u * state->coef_au
           [1]
89             - state->state_y * state->coef_by[1];
90
91     if (state->antiwindup) {
92         y = ACT_Limit(y, state->inferior, state->superior);
93     }
94     state->state_u = u0;
95     state->state_y = y;
96     return y;
97 }
98
99 void ACT_Init(ACT_State_t *pActState, ACT_Output_t *pActOutput) {
100     tf_z1_init(&pActState->tf_speed_estimator, (float)
           SPEED_ESTIMATOR_A0,
101             (float) SPEED_ESTIMATOR_A1, (float) SPEED_ESTIMATOR_B1
                ,
102             SPEED_ESTIMATOR_ANTIWINDUP,
103             (float) SPEED_ESTIMATOR_UPPER_SATURATION,
104             (float) SPEED_ESTIMATOR_LOWER_SATURATION);
105
106     pActState->position_r = 0.0f;
107     pActState->position_c = 0.0f;
108
```

```
109     pActState->speed_c = 0.0f;
110     pActState->speed_e = 0.0f;
111     pActState->speed_r = 0.0f;
112 
113     pActState->inputTestState.inputTestTable = inputTestTable;
114 
115     pActOutput->Vc = 0.0f;
116     pActOutput->actMode = ACT_STANDBY;
117     pActOutput->alive = 0;
118 }
119 
120 void ACT_StateMachineUpdate(ACT_Input_t const *pActInput,
    ACT_State_t *pActState) {
121     switch (pActInput->modeSel) {
122     case ACT_CMD_STANDBY:
123         pActState->actMode = ACT_STANDBY;
124         break;
125     case ACT_CMD_CHIRP:
126         pActState->actMode = ACT_CHIRP;
127         break;
128     case ACT_CMD_SETPOINT:
129         pActState->actMode = ACT_SETPOINT;
130         break;
131     case ACT_CMD_AUTOTUNING:
132         pActState->actMode = ACT_AUTOTUNING;
133         break;
134     }
135 }
136 
137 void ACT_Update(ACT_Input_t const *pActInput, ACT_State_t *
    pActState,
138         ACT_Output_t *pActOutput) {
139     int iterSM = 0;
140 
141     pActOutput->alive = 1;
142 
143     for (iterSM = 0; iterSM < 3; iterSM++) {
144         ACT_StateMachineUpdate(pActInput, pActState);
145     }
146 
147     switch (pActState->actMode) {
```

```
148     case ACT_STANDBY:
149         pActState->position_c = 0.0f;
150         break;
151     case ACT_CHIRP:
152         pActState->position_c = chirp(pActInput, pActState);
153         break;
154     case ACT_SETPOINT:
155         pActState->position_c = setpoint(pActInput, pActState);
156         break;
157     case ACT_INPUTTEST:
158         pActState->position_c = inputTest(pActInput, pActState);
159     case ACT_AUTOTUNING:
160         break;
161     }
162 
163     pActState->speed_e = tf_z1(&pActState->tf_speed_estimator,
        pActInput->posm);
164 
165     pActState->position_r = pActState->position_c - pActInput->
        posm;
166     pActState->integral += pActState->position_r * PD_SAMPLE_TIME
        * PD_I_GAIN;
167     pActState->speed_c = pActState->position_r * PD_P_GAIN
168             + pActState->integral;
169     pActState->speed_r = pActState->speed_c - (pActState->speed_e
        * PD_D_GAIN);
170 
171     pActOutput->Vc = pActState->speed_r * PD_V;
172 
173     ACT_AntiWindUp(pActState, pActOutput);
174 
175     pActOutput->actMode = pActState->actMode;
176 }
```

Apêndice B - Entrada Chirp

Listing B.1 – Chirp

```
function y = chirp_por_degraus(t)

f_start = 1;
f_end   = 6;
n_freqs = 5;
n_cycles = 10;

persistent freqs Tcum
if isempty(freqs)
    freqs = linspace(f_start, f_end, n_freqs);
    Tseg = n_cycles ./ freqs;
    Tcum = [0, cumsum(Tseg)];
end

idx = n_freqs;
for k = 1:n_freqs
    if t < Tcum(k+1)
        idx = k;
        break;
    end
end

t_local = t - Tcum(idx);
f = freqs(idx);

y = sin(2*pi*f*t_local)*10;

end
```

Apêndice C - Repositório

O repositório do desenvolvimento do trabalho está disponível no seguinte endereço: https://github.com/Igor-Mayer-Soares/tccPosUSP. Para obter acesso, contatar o autor via GitHub.