UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

Rafael Máximo da Silva

Controladores Clássicos e com Modos Deslizantes aplicados ao Pêndulo Invertido com Roda de Reação

Ilha Solteira
2025

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

Rafael Máximo da Silva

Controladores Clássicos e com Modos Deslizantes aplicados ao Pêndulo Invertido com Roda de Reação

Tese de doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Estadual Paulista, câmpus de Ilha Solteira, para obtenção do título de Doutor em Engenharia Elétrica.

Área de Concentração: Automação.

Orientador: Prof. Dr. Jean Marcos de Souza Ribeiro

Ilha Solteira
2025

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

S586c

Silva, Rafael Máximo da .
Controladores clássicos e com modos deslizantes aplicados ao pêndulo invertido com roda de reação / Rafael Máximo da Silva. -- Ilha Solteira: [s.n.], 2025
143 f. : il.

Tese (doutorado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2025

Orientador: Jean Marcos de Souza Ribeiro

Inclui bibliografia

1. ESP32. 2. IDE arduino. 3. Controlador com modos deslizantes. 4. PID. 5. Alocação de autovalores. 6. Índices de desempenho.

Amanda Sertori dos Santos
Assinado de forma digital por Amanda Sertori dos Santos
Dados: 2025.12.10 16:26:34 -03'00'

Impacto Potencial Desta Pesquisa

Esta pesquisa apresenta impacto científico, tecnológico, educacional e social ao propor e validar, em tempo real, estratégias de controle clássico e robusto aplicadas a um sistema não linear e subatuado, por meio de uma bancada experimental desenvolvida com *hardware* de baixo custo. Os resultados contribuem para o avanço do conhecimento em controle automático, favorecem a formação de recursos humanos qualificados, estimulam a inovação em plataformas experimentais, reduzem custos de implementação e fortalecem a inserção nacional e internacional da pesquisa na área.

Potential Impact of This Research

This research presents scientific, technological, educational, and social impact by proposing and validating, in real time, classical and robust control strategies applied to a nonlinear and underactuated system through an experimental platform developed with low-cost hardware. The results contribute to the advancement of knowledge in automatic control, promote the training of qualified human resources, foster innovation in experimental platforms, reduce implementation costs, and strengthen the national and international integration of the research in the field.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA TESE: Controladores Clássicos e com Modos Deslizantes aplicados ao Pêndulo com Roda de Reação

AUTOR: RAFAEL MÁXIMO DA SILVA
ORIENTADOR: JEAN MARCOS DE SOUZA RIBEIRO

Aprovado como parte das exigências para obtenção do Título de Doutor em Engenharia Elétrica, área: Automação pela Comissão Examinadora:

gov.br Documento assinado digitalmente
JEAN MARCOS DE SOUZA RIBEIRO
Data: 05/11/2025 18:43:40-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. JEAN MARCOS DE SOUZA RIBEIRO (Participaçao Presencial)
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira - UNESP

gov.br Documento assinado digitalmente
RODRIGO CARDIM
Data: 05/11/2025 20:09:17-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. RODRIGO CARDIM (Participaçao Presencial)
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira - UNESP

gov.br Documento assinado digitalmente
CRISTIANO QUEVEDO ANDREA
Data: 06/11/2025 00:01:43-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. CRISTIANO QUEVEDO ANDREA (Participaçao Virtual)
Faculdade de Engenharias, Arquitetura e Urbanismo e Geografia - FAENG / Universidade Federal de Mato Grosso do Sul - UFMS

gov.br Documento assinado digitalmente
DIOGO RAMALHO DE OLIVEIRA
Data: 06/11/2025 00:09:40-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. DIOGO RAMALHO DE OLIVEIRA (Participaçao Virtual)
Departamento de Eletroeletrônica / Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso do Sul - IFMS

gov.br Documento assinado digitalmente
BRUNO SERENI
Data: 06/11/2025 12:42:23-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. BRUNO SERENI (Participaçao Virtual)
Faculdade de Engenharias, Arquitetura e Urbanismo e Geografia - FAENG / Universidade Federal de Mato Grosso do Sul - UFMS

Dedico este trabalho à minha família.

Agradecimentos

Primeiramente a Deus que me capacitou, deu-me saúde e forças para que tudo isso fosse possível. Em especial, dedico meus agradecimentos:

- Aos meus pais, Maria Evanil e Edilson Alfredo, e minha irmã Naara, pela ajuda, compreensão e amor que recebi; mesmo na distância eles nunca deixaram de me socorrer e apoiar. A todos da minha família, que nunca se esqueceram de mim, sempre realizando orações e sempre me apoiando;
- Ao meu orientador, Prof. Dr. Jean Marcos de Souza Ribeiro, pela oportunidade oferecida, pela orientação, pelo tempo dedicado e ensinamentos, os quais foram fundamentais no desenvolvimento deste trabalho;
- Aos professores Prof. Dr. Marcelo C. M. Teixeira, Prof. Dr. Edvaldo Assunção e Prof. Dr. Rodrigo Cardim, pelos ensinamentos e pela contribuição ao longo do desenvolvimento desta pesquisa;
- Aos professores e colegas do laboratório LPC, pelo apoio e pela colaboração durante a pesquisa;
- Aos técnicos de laboratório Bruno Oikawa, Valdemir Chaves e Wendel Jordão, pelo apoio prestado na etapa de implementação;
- Agradeço a todos os professores e funcionários da FEIS-UNESP, que diretamente ou indiretamente contribuíram para a realização deste trabalho;
- Aos colegas do PPGEE - Programa de Pós-Graduação em Engenharia Elétrica, pela convivência e pela ajuda de forma direta ou indireta.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

"Posso todas as coisas em Cristo que me fortalece."
(Filipenses 4:13)

Resumo

Neste trabalho foi realizada a simulação e implementação prática do controle do sistema Pêndulo Invertido por Roda de Reação (PIRR), utilizando componentes de *hardware* de baixo custo. Foi considerado uma estratégia de controle *switch* para atuação dos controladores *swing up* e de estabilização. Esse método de controle *switch* integra dois controladores distintos: um para elevar o pêndulo até a região invertida, e outro para controlá-lo no ponto de equilíbrio instável na posição invertida. Assim, foi considerado um controlador *swing up* não linear para resolver o problema de movimentar o pêndulo da posição de equilíbrio estável até atingir uma posição favorável para a aplicação de um controlador de estabilização. Para a região de estabilização, foram projetados controladores lineares, PID e alocação de autovalores, e um controlador não linear baseado na técnica de Controle por Modos Deslizantes convencional (CMD). O comportamento do sistema PIRR, com os controladores projetados, foi verificado através de simulações com o MATLAB/SIMULINK®. As leis de controle foram programadas em *script* desenvolvido na IDE Arduino (do inglês, *Integrated Development Environment*), utilizando o microcontrolador ESP32, e a coleta de dados foi realizada através da comunicação serial entre o MATLAB® e ESP32. Foram calculados índices de desempenho para avaliar a resposta da variável controlada na região de estabilização.

Palavras-chave: ESP32; IDE Arduino; modos deslizantes; PID; alocação de autovalores; índices de desempenho.

Abstract

In this work, the practical implementation of the control system for the Reaction Wheel Inverted Pendulum (RWP) was carried out using low-cost hardware components. A hybrid control strategy was adopted to operate the swing-up and stabilization controllers. This hybrid control method integrates two distinct controllers: one designed to raise the pendulum to the inverted region, and another to stabilize it at the unstable equilibrium point in the inverted position. A nonlinear swing-up controller was designed to address the problem of moving the pendulum from the stable equilibrium position to a favorable position for the application of a stabilization controller. For the stabilization region, linear controllers—PID and state-feedback—were designed, as well as a nonlinear controller based on the conventional Sliding Mode Control (SMC) technique. The behavior of the RWP system, with the designed controllers, was verified through simulations performed in MATLAB/SIMULINK®. The control laws were programmed using scripts developed on the Arduino IDE with the ESP32 prototyping board, and data collection was performed through serial communication between MATLAB® and the ESP32. Performance indices were calculated to evaluate the response of the controlled variable in the stabilization region.

Keywords: ESP32; Arduino IDE; sliding modes; PID; pole placement; performance indices.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

A/D	Analógico-Digital
CC	Corrente Contínua
CEV	Controle com Estrutura Variável
CEV/MD	Controle com Estrutura Variável e Modos Deslizantes
CMD	Controle com Modo Deslizante
D/A	Digital-Analógico
DEE-FEIS	Departamento de Engenharia Elétrica - Faculdade de Engenharia de Ilha Solteira
DOF	*Degrees of Freedom* (Graus de Liberdade)
IAE	*Integral of Absolute Error* (Integral do Valor Absoluto do Erro)
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
ISE	*Integral of Squared Error* (Integral do Quadrado do Erro)
ITAE	*Integral of Time multiplied by Absolute Error* (Integral do Tempo Multiplicado pelo Valor Absoluto do Erro)
ITSE	*Integral of Time multiplied by the Squared Error* (Integral do Tempo Multiplicado pelo Quadrado do Erro)
LPC	Laboratório de Pesquisa em Controle
MIMO	*Multiple-input and Multiple-output* (Múltiplas Entradas e Múltiplas Saídas)
PID	Proporcional, Integral e Derivativo
PIRR	Pêndulo Invertido Por Roda de Reação
PPR	Pulsos por revolução
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
RMS	*Root Mean Square* (Valor Médio Quadrático)
RPM	Rotações por minuto

SCADA	*Supervisory Control And Data Acquisiton* (Controle Supervisório e Aquisição de Dados)
SISO	*Single Input-Single Output* (Única entrada e Única saída)

Lista de símbolos

A_m	Atrito do motor
$C.I.$	Condição Inicial
C_i	Constantes de inércia
e	Erro de rastreamento
E_{ct}	Energia cinética
E_{pg}	Energia potencial
F_c	Atrito de coulomb do pêndulo
F_{fr}	Atrito do pêndulo
F_v	Atrito viscoso do pêndulo
g	Aceleração da gravidade
h_v	Tempo de amostragem da velocidade
I_a	Corrente de armadura
I_{ab}	Corrente de armadura do rotor bloqueado
I_p	Momento de inércia do pêndulo
I_r	Momento de inércia da roda
K	Ganhos do controle de estabilização
K_t	Constante de torque
K_v	Constante de força eletromotriz
K_{sw}	Ganho do conrole *swing up*
L	Comprimento da haste
L_{ag}	Função de Lagrange
L_{ch1}	Distância do pivô ao centro de massa da haste 1
L_{ch2}	Distância do pivô ao centro de massa da haste 2
L_{h1}	Comprimento da haste da roda em relação ao pivô

L_{h2}	Comprimento da haste do contrapeso em relação ao pivô
L_{h3}	Distância do centro de massa do conjunto de massas na roda em relação ao pivô
L_{h4}	Distância do centro de massa do contrapeso em relação ao pivô
$\bar{M_1}$	Centro de massas da roda
$\bar{M_1}$	Centro de massas do contrapeso
M_a	Massa do acoplamento
M_c	Massa do contrapeso
M_{h1}	Massa da haste 1
M_{h2}	Massa da haste 2
M_m	Massa do motor
M_r	Massa da roda
M_{tr}	Conjunto de massas na roda
n	Redução do motor
P_{ma}	Polos malha aberta
P_{mf}	Polos malha fechada
R_a	Resistência de armadura
sgn	Função sinal
sat	Função saturação
t_r	Tempo de alcance
T	Tempo de amostragem
u_{sw}	Sinal de controle *swing up*
u_{ap}	Sinal de controle de estabilização (Alocação de autovalores)
u_{md}	Sinal de controle por modos deslizantes
u_{eq}	Sinal de controle equivalente
u_n	Sinal de controle chaveado

$V(t)$	Tensão de alimentação
δ	Controle posição de referência para o controle de estabilização
δ_s	Posição de referência para o controle de *swing up*
θ	Posição angular do pêndulo
θ_d	Posição angular desejado do pêndulo
τ	Torque
τ_e	Torque elétrico nominal
τ_{eb}	Torque de rotor bloqueado
τ_i	Torque generalizado do atuador
ϕ	Posição angular da roda
$\mathcal{V}$	Função de Lyapunov
$\dot{\mathcal{V}}$	Derivada da função de Lyapunov
σ	Função de chaveamento
$\dot{\sigma}$	Derivada da função de chaveamento
γ	Valor constante da função de chaveamento
η	Ganho chaveado positivo (lei de alcance exponencial)
k	Ganho chaveado positivo (lei de alcance exponencial)
ε	Escalar pequeno e positivo

Sumário

1 INTRODUÇÃO

Os sistemas com pêndulo invertido tem um contexto histórico interessante, pois em razão de sua característica não linear e instável, vem sendo utilizado como testes de bancada, para validação de novos conceitos e teorias de controle; também esse sistema é mencionado como exemplo clássico em livros didáticos que tratam de sistemas dinâmicos (Dorf; Bishop, 2009; Ogata, 2010). Existem diferentes versões de pêndulos invertidos disponíveis, como por exemplo, o pêndulo rotacional desenvolvido por Furuta (Yamakita; Hoshino; Furuta, 1999), o Acrobot (Spong, 1995), o Pendubot (Fantoni; Lozano; Spong, 2000), e o pêndulo por roda de reação estudado em Block, Åström e Spong (2007).

Na indústria aeroespacial, é comum a utilização de rodas de reação devido à sua capacidade de proporcionar controle preciso e eficaz, sobre a orientação ou atitude de naves espaciais e satélites (Xu; Kanade, 1992; Hu, 2010; Yin *et al.*, 2016; Yang, 2017; Kumar; Abreu; Sinha, 2018; Rahimi; Kumar; Alighanbari, 2019; Dennehy, 2019; Shiyou *et al.*, 2022). Rodas de reação consistem em uma roda giratória, alinhada com um eixo fixo, e sua aceleração pode ser ajustada. Quando as rodas de reação aceleram ou desaceleram, geram torques que influenciam o movimento (Schaub; Junkins, 2003). Assim, esses atuadores são utilizados como dispositivos de troca de momento angular, fornecendo torque de reação e acumulando momento angular. O princípio físico básico envolve a conservação do momento angular, segundo o qual a taxa de variação do momento angular de um sistema é igual ao torque externo resultante aplicado sobre ele (Goldstein; Poole; Safko, 2002; Meriam; Kraige, 2006; Noack; Brieß, 2014; Mehrjardi; Sanusi; Ali, 2015; Colagrossi, 2023; Lira, 2025).

A aplicação eficaz de atuadores de roda de reação estende-se para além do setor aeroespacial, encontrando implementação prática no campo da robótica. Por exemplo em Weber, Cuvillon e Gangloff (2014), a roda de reação foi utilizada para cancelar oscilações rotacionais no robô paralelo atuado por cabos. Brown e Schmiedeler (2014) consideraram a roda de reação como atuador inercial, para melhorar a estabilidade e robustez do robô bípede. Lee *et al.* (2023) apresentaram um sistema de roda de reação que aprimora as capacidades de equilíbrio e estabilidade de robôs quadrúpedes durante tarefas desafiadoras de locomoção.

No ambiente de trabalho de laboratório, pesquisadores apresentaram um novo sistema mecânico que envolve um pêndulo invertido e consideraram a roda de reação como atuador (Spong; Corke; Lozano, 2001). Este sistema representa uma nova abordagem em relação ao clássico pêndulo invertido montado sobre um carro e aos diversos experimentos de pêndulo disponíveis. Ele se diferencia pela sua dinâmica mais simples em relação a esses experimentos, sendo caracterizado como um sistema não linear e subatuado, ou seja, o número de atuadores disponíveis é menor do que o número de graus de liberdade (Fantoni;

Lozano; Spong, 2001; Fantoni; Lozano, 2002; Block; Åström; Spong, 2007).

Na literatura, encontram-se diversos métodos de controle que garantem a estabilidade de um sistema Pêndulo Invertido por Roda de Reação (PIRR). Em Olivares e Albertos (2013) e Trentin *et al.* (2020a), por exemplo, foi aplicada a metodologia clássica de controle Proporcional, Integral e Derivativa (PID), para controle do sistema PIRR, no ponto de equilíbrio instável presente na região invertida do pêndulo.

Para o problema de controle do PIRR na posição invertida, Spong, Corke e Lozano (2001), aplicaram o controle por alocação de polos; e para o problema de *swing up*, apresentaram uma abordagem de energia, baseada na linearização de realimentação parcial. Hernandez e Sira-Ramirez (2003), apresentaram uma aplicação do controle PI generalizado, combinado com o Controle com Modos Deslizantes (CMD), para oscilar e estabilizar o PIRR em torno de sua posição vertical instável. Bapiraju *et al.* (2004) investigaram diferentes técnicas de linearização e projetaram um controlador lógico fuzzy para o PIRR.

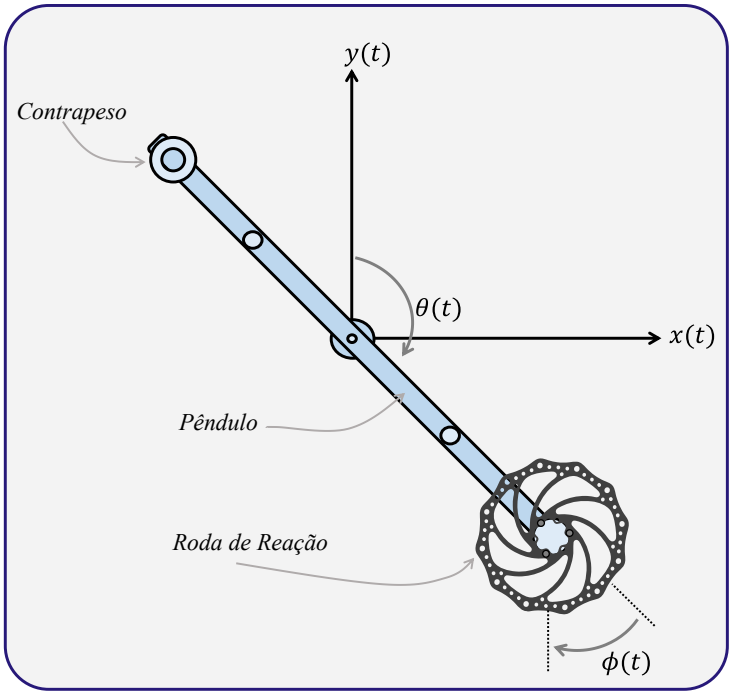
Rizal *et al.* (2018) apresentaram o controle de estabilização do PIRR baseado no CMD de segunda ordem utilizando o algoritmo *Super-twisting*. Trentin *et al.* (2020b) projetaram e compararam os resultados do CMD convencional aplicado no PIRR, utilizando dois atuadores, com o controle PID.

Assim, sabe-se que o controle com modos deslizantes convencional oferece diversas vantagens, incluindo sua capacidade de convergência em tempo finito para uma superfície estável, e robustez em relação a incertezas paramétricas, perturbações casadas com o sinal de controle e dinâmicas não-modeladas, além de apresentar um bom comportamento transitório. No entanto, tem algumas desvantagens, como o *chattering*, fase de alcance e sensibilidade a perturbações exageradas.

Visando controlar o sistema do pêndulo invertido por meio da roda de reação, é proposto um método de controle *switch* que envolve dois tipos de controladores: um para elevar o pêndulo até a região invertida, e outro para controlar o pêndulo em seu ponto de equilíbrio instável na posição invertida. Desse modo, o controle do sistema PIRR pode ser realizado em dois estágios distintos: no primeiro estágio, é aplicado um controlador do tipo *swing up*, responsável por atuar na região em que o pêndulo se encontra nas proximidades do ponto de equilíbrio estável, conduzindo-o em direção à posição vertical; no segundo estágio, realiza-se a comutação para um controlador estabilizante, cuja finalidade é manter o pêndulo em torno do ponto de equilíbrio instável correspondente à posição vertical invertida.

Na Figura 1 é apresentado um diagrama esquemático do sistema PIRR, sendo $\theta(t)$ a posição angular do pêndulo, e $\phi(t)$ a posição angular da roda. Além disso, são ilustrados o pêndulo, o contrapeso e a roda de reação.

Figura 1 – Pêndulo invertido por roda de reação.



Fonte: próprio autor.

A necessidade de projetar um controlador para conduzir o pêndulo até a região invertida, é devido ao torque insuficiente do motor para elevar o pêndulo considerando apenas o controle de estabilização. Em Trentin *et al.* (2020a) por exemplo, foi apresentado o controle de estabilização do PIRR com controlador PID, no entanto, não foi preciso utilizar o controle *swing up* para levar o pêndulo até a região invertida, devido o motor possuir torque suficiente e alta velocidade de rotação. A utilização de um motor CC com alto torque é vantajoso para controlar o PIRR, em contrapartida, aumenta os custos do projeto de implementação do sistema PIRR.

Para controlar o pêndulo na região de estabilização, deve-se elaborar um controlador de posição para o PIRR, por meio da variação do momento angular da roda de reação. A variação do momento angular é causada pela aceleração do motor CC acoplado à roda de reação. A velocidade angular do motor CC deve ser controlada de modo que a aceleração angular da roda também seja controlada, porque é por meio da aceleração e desaceleração da roda de reação, que é possível controlar o pêndulo no ponto de equilíbrio instável.

Dessa forma, foram projetados três controladores para estabilização do pêndulo no ponto de equilíbrio instável $\theta(t) = 0°$, os quais são PID, alocação de autovalores, e controlador CMD. Também foi projetado um controlador *swing up* para levar o pêndulo do ponto de equilíbrio estável $\theta(t) = 180°$, até próximo ao ponto de equilíbrio instável definido em $\theta(t) = 0°$. Os desempenhos dos controladores PID, alocação de autovalores,

e controlador CMD, foram verificados experimentalmente utilizando uma planta que pertence ao Laboratório de Pesquisa em Controle (LPC) e considerando *hardware* de baixo custo para a implementação dos controladores em tempo real. Para verificar o comportamento do sistema PIRR com os controladores projetados, foram feitas simulações com o MATLAB/SIMULINK® levando em conta o modelo não linear da planta.

1.1 OBJETIVO GERAL

O principal objetivo deste doutorado é a implementação prática do controle do sistema PIRR, utilizando componentes de *hardware* de baixo custo. A metodologia proposta adota uma estrutura de controle *switch*, composta por dois controladores complementares: o primeiro destinado à elevação do pêndulo até a região próxima ao ponto de equilíbrio instável, e o segundo responsável pela estabilização do sistema em torno do ponto de operação.

Os objetivos específicos podem ser resumidos da seguinte forma:

a) Implementar em bancada o controle *switch* do sistema PIRR, integrando as etapas de elevação e estabilização do pêndulo;

b) Avaliar, por meio de implementações experimentais, qual dos três controladores projetados, PID, alocação de autovalores, ou controlador CMD, apresenta o melhor índice de desempenho para o controle de posição em torno do ponto de operação ($\theta(t) = 0°$) na região de estabilização.

1.2 ESTRUTURA DO TRABALHO

Este trabalho foi organizado da seguinte forma:

No Capítulo 1 são apresentadas a contextualização do sistema PIRR e a definição do tema abordado na tese. Além disso, são descritos os objetivos e a estrutura geral do trabalho.

No Capítulo 2 são apresentados conceitos fundamentais necessários para o projeto de controladores não lineares, e o projeto de controladores lineares para o controle do sistema PIRR.

No Capítulo 3 é descrita a modelagem dinâmica do sistema PIRR. São apresentadas as equações que descrevem a dinâmica do pêndulo, bem como a dinâmica do motor de corrente contínua a ser implementada na equação de movimento do PIRR

No Capítulo 4 é apresentada a descrição do modelo experimental do PIRR. São apresentados o sistema mecânico, sistema elétrico, diagrama do circuito elétrico, fluxo-

grama do algoritmo implementado na IDE Arduino (do inglês, *Integrated Development Environment*), atrito do pêndulo e o procedimento de coleta de dados com o MATLAB®.

No Capítulo 5 são apresentados os projetos dos controladores aplicados ao PIRR. São apresentados o projeto do CMD convencional, o projeto do controle por alocação de autovalores, e o projeto do controle PID para estabilizar o PIRR em torno do ponto de operação. Também é apresentado o projeto do controlador *swing up* para elevação do pêndulo próximo do ponto de equilíbrio instável. Por fim, é descrito o método de controle *switch* implementado no sistema PIRR. No método de controle *switch* são considerados dois controladores: um para elevar o pêndulo próximo à região invertida (ponto de equilíbrio instável) e outro para o controle estabilizante.

No Capítulo 6 são apresentados e discutidos resultados experimentais obtidos para os controladores projetados. Também são exibidos os resultados de implementação para os controladores lineares. Além disso, são mostrados e comparados os resultados dos controladores lineares através de índices de desempenho, bem como os resultados do controle *swing up* para levantar o pêndulo até a região invertida. Por fim, são apresentados os resultados do controle *switch* aplicado no sistema PIRR.

No Capítulo 7 são apresentadas as considerações finais.

2 CONCEITOS FUNDAMENTAIS

Neste capítulo são apresentados os conceitos fundamentais para o projeto de controladores não lineares, e o projeto de controladores lineares para o controle do sistema PIRR. Na Seção 2.1 são expostos os principais elementos da teoria de controle com estrutura variável e modos deslizantes. Na Seção 2.2 é apresentado o controle *swing up* necessário para levar o pêndulo para a posição invertida. Na Seção 2.3 são apresentados conceitos de sistemas contínuos em espaço de estados, estabilidade de sistemas dinâmicos contínuos no tempo, e controlabilidade e observabilidade. Na Seção 2.4 é apresentado o controle por alocação de autovalores. Na Seção 2.5 aborda o controlador PID. Na Seção 2.6 são apresentados conceitos de sistemas de controle digital, controladores PID digitais, amostragem e discretização.

2.1 CONTROLE COM ESTRUTURA VARIÁVEL E MODOS DESLIZANTES

Nesta seção são abordados conceitos básicos do controle com estrutura variável e modos deslizantes convencionais, metodologia de controle com modo deslizante, condição de convergência em tempo finito, método do controle equivalente, e sistema de CMD utilizando a abordagem da lei de alcance.

2.1.1 Conceitos Básicos de Estrutura Variável e Modos Deslizantes

Os sistemas com estrutura variável e seu principal modo de operação, o CMD, são reconhecidos como ferramentas altamente eficazes para lidar com sistemas incertos, devido à sua robustez e capacidade de resistir a perturbações (Edwards; Spurgeon, 1998; Shtessel *et al.*, 2014).

O Controle com Estrutura Variável (CEV) foi iniciado na Rússia por vários pesquisadores, incluindo Barbashin e Geraschenko (1965), Utkin (1992), Emelyanov (1985), etc. A abordagem foi principalmente investigada em sistemas de tempo contínuo utilizando o controle de modo deslizante (Furuta; Pan, 1999). Assim, historicamente, os modos deslizantes foram descobertos como um caso especial em sistemas de estrutura variável (Shtessel *et al.*, 2014). No início, apenas os sistemas SISO (do inglês, *Single Input-Single Output*) no espaço canônico foram estudados; desta forma a teoria do CMD tem suas origens há mais de 50 anos. O interesse no espaço do sinal de saída e suas derivadas no tempo pode ser facilmente explicado: a equação do modo deslizante não depende de variações de parâmetros nem de perturbações externas (Utkin *et al.*, 2020).

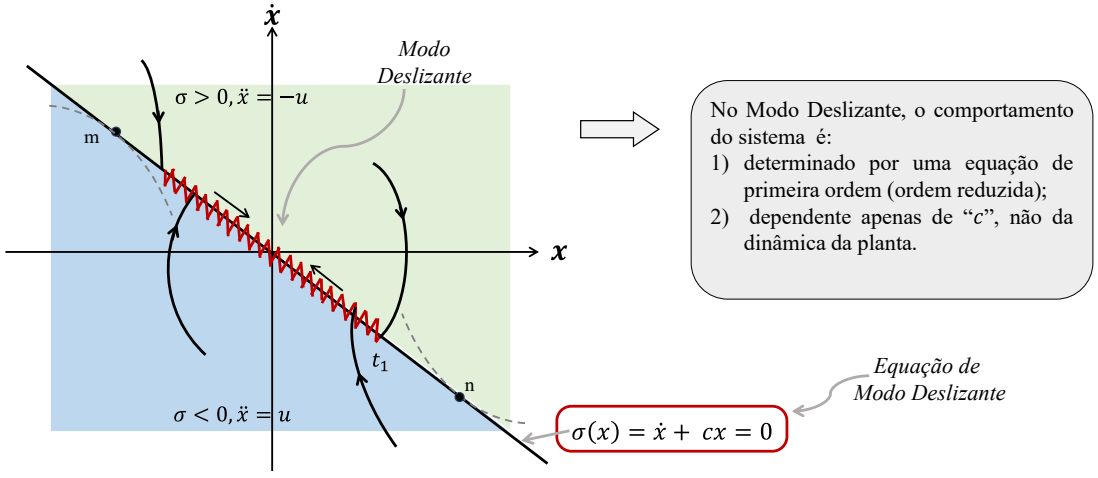
Os modos deslizantes podem surgir em um sistema dinâmico regido por equações diferenciais ordinárias com lados direitos descontínuos. Um exemplo típico usado para ilustrar os modos deslizantes no contexto do método de espaço de estados, é um sistema de relé de segunda ordem invariante no tempo (Utkin; Guldner; Shi, 2017):

$$\begin{aligned} \ddot{x}(t) + a_2\dot{x}(t) + a_1 x(t) &= u(t) + f(t), \\ u(t) &= -M\text{sgn}(\sigma(x)), \\ \sigma(x) &= \dot{x}(t) + cx(t), \end{aligned} \tag{1}$$

tal que M, a_1, a_2 e c são parâmetros constantes, e $f(t)$ é um distúrbio limitado; $u(t)$ é uma função descontínua do estado. Esta abordagem é comumente chamada de Controle de Modo Deslizante convencional (Modos Deslizantes de primeira ordem).

O comportamento do sistema pode ser analisado no plano de estados $(x, \dot{x})$. Na Figura 2 é mostrado o plano de fase para $a_1 = a_2 = 0$. O sinal de controle $u(t)$ sofre descontinuidades na função de deslizamento (ou função de chaveamento) $\sigma(x) = 0$, e as trajetórias de estado são constituídas por duas classes: a primeira corresponde a $\sigma(x) > 0$ e $u(t) = -M$ (semiplano superior), e a segunda corresponde a $\sigma(x) < 0$ e $u(t) = M$ (semiplano inferior). Dentro do segmento da reta entre m e n na função de deslizamento, as trajetórias de estado são direcionadas para essa reta. Tendo atingido o setor em algum tempo t_1, o estado não pode sair da função de deslizamento. Isso significa que a trajetória do estado pertencerá à função de deslizamento para $t > t_1$. Esse movimento, com as trajetórias de estado nessa reta, é chamado de Modo Deslizante (Utkin; Guldner; Shi, 2017).

Figura 2 – Modo deslizante em sistema de segunda ordem.



Fonte: adaptado de Utkin *et al.* (2020).

Como, no decorrer do Modo Deslizante, a trajetória do estado coincide com a função de deslizamento $\sigma = 0$, sua equação pode ser interpretada como a equação de movimento:

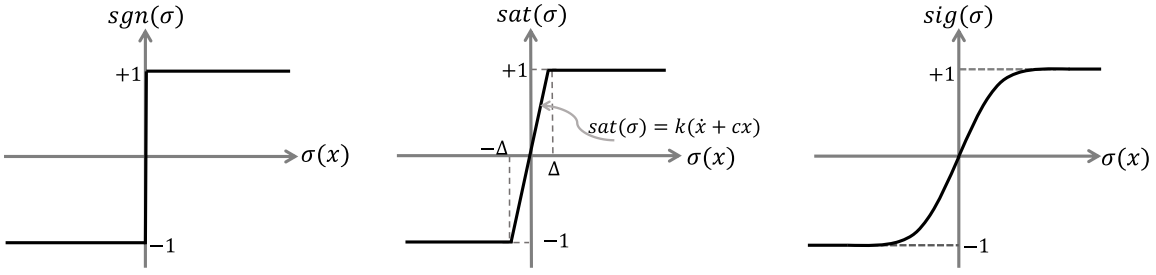
$$\dot{x}(t) + cx(t) = 0. \tag{2}$$

É importante notar que a solução $x(t) = x(t_1)e^{-c(t-t_1)}$ não depende de variações de parâmetros da planta, nem de perturbações externas. Assim, a propriedade de "invariância" oferece boas perspectivas para o projeto de controle realimentado em plantas dinâmicas operando em condições de incertezas (Utkin *et al.*, 2020).

O que foi descrito anteriormente ocorre em um modelo matemático ideal. Contudo, em implementações reais, as trajetórias estão confinadas a alguma vizinhança da função de deslizamento. Esta inconsistência do modelo ideal pode ser causada por imperfeições dos dispositivos de chaveamento, como pequenos atrasos, zonas mortas ou histerese, que podem levar a oscilações de alta frequência. O mesmo fenômeno pode ocorrer como resultado da negligência das pequenas constantes de tempo dos sensores e atuadores no modelo ideal. Esta oscilação chamada de trepidação, foi um sério obstáculo para o uso de modos deslizantes em sistemas de controle. No entanto, existem métodos para redução dessa oscilação indesejada (Utkin; Guldner; Shi, 2017).

Uma abordagem para reduzir a trepidação é empregar uma das funções do método *quasi-sliding mode* (saturação ou sigmoide), que ajuda a manter o estado dentro de uma faixa especificada por Δ. Na Figura 3 são apresentadas a função sinal, Figura 3(a), a função saturação, Figura 3(b), e a função sigmoide, Figura 3(c). Frequentemente, o Δ é referido como camada limite. Observe-se na Figura 3(b), que as trajetórias de estado estão confinadas a alguma vizinhança (Δ) da função de deslizamento para aproximação contínua, (Figura 3(b)), de uma função relé descontínua, (Figura 3(a)). Na Figura 3(c) é apresentada a função sigmoide a qual considera a função $\text{sig}(\sigma) = \frac{\sigma}{|\sigma|+\epsilon}$, sendo ϵ um escalar pequeno e positivo.

Figura 3 – Aproximação contínua do controle descontínuo.



(a) Função sinal. (b) Função saturação. (c) Função sigmoide.

Fonte: adaptado de Utkin, Guldner e Shi (2017).

O exemplo do relé demonstrou a redução de ordem e, a invariância em relação às incertezas da planta no sistema com modo deslizante. O uso dessas propriedades foi a ideia-chave da teoria da Estrutura Variável na fase inicial, quando apenas sistemas SISO com equações de movimento no espaço canônico foram estudados (Emelyanov *et al.*, 1970; Utkin; Guldner; Shi, 2017).

Nos processos tecnológicos modernos, é comum que o controle e a saída do sistema sejam grandezas com valor vetorial, e apenas alguns componentes do vetor de estado sejam acessíveis para medição. A abordagem do espaço canônico não forneceu nenhum método de como o controle poderia ser projetado em tais situações (Utkin; Guldner; Shi, 2017). A segunda etapa de estudos de sistemas de estrutura variável, foi dedicada ao desenvolvimento de métodos de projeto para sistemas com equações de movimento em um espaço de estados arbitrário, com ação de controle vetorial e variáveis vetoriais a serem controladas (Utkin, 1983). Assim, o conceito fundamental por trás da maioria dos métodos de controle envolve a implementação de modos deslizantes multidimensionais (Utkin; Guldner; Shi, 2017).

No exemplo anterior (Equação 1), o controle é uma função escalar do estado, e o modo deslizante é determinado por uma equação diferencial com ordem uma unidade menor do que a ordem do sistema original. Portanto, pode-se assumir que o modo deslizante pode surgir na interseção de várias superfícies, se o controle for um valor vetorial e cada componente apresentar descontinuidades em sua própria superfície de deslizamento (Utkin; Guldner; Shi, 2017).

Em sistemas de controle, o CMD é um método de controle não linear que altera a dinâmica de um sistema não linear pela aplicação de um sinal de controle descontínuo no tempo. Este sinal, força o sistema a deslizar ao longo de um espaço fechado com trajetórias desejadas no espaço de estados do sistema (Utkin *et al.*, 2020).

A lei de controle de realimentação de estados não é uma função contínua do tempo; em vez disso, esse sistema pode alternar de uma estrutura contínua para outra com base na posição atual no espaço de estados. Isso explica porque o CMD deriva do chamado método de CEV. No contexto da teoria de controle moderna, qualquer sistema de estrutura variável, como um sistema sob CMD, pode ser visto como um caso especial de sistemas dinâmicos que incorporam lógica de chaveamento (Utkin *et al.*, 2020).

Dessa forma, a estratégia de CEV/MD utiliza uma lei de controle com chaveamento, para direcionar e manter a trajetória dos estados de uma planta em uma superfície específica (chamada de superfície de deslizamento ou superfície de chaveamento), ou na intersecção de todas as superfícies escolhidas no espaço de estados. Quando a trajetória dos estados alcança essa superfície e nela se mantém, o sistema é considerado estar em uma condição de deslizamento, ou em modo deslizante. Nessa situação, o comportamento do sistema sofre menor influência por parte de alterações paramétricas ou de distúrbios externos, conferindo ao sistema controlado uma maior robustez (Ribeiro, 2006).

Um modo deslizante existe em um sistema se, nas proximidades da superfície de deslizamento, os vetores tangente ou velocidade da trajetória de estado sempre apontam em direção à superfície de deslizamento (DeCarlo; Zak; Matthews, 1988). Assim, uma lei de controle chaveada deve ser projetada para assegurar que a trajetória de estados

se dirija à superfície de deslizamento (alcançabilidade) e nela permaneça durante todo o tempo subsequente (atratividade) (Utkin, 1992; Ribeiro, 2006).

Garantir a existência do Modo Deslizante sobre a superfície de deslizamento é uma necessidade fundamental no projeto do CEV/MD. Projetar a dinâmica da superfície é um problema complementar (DeCarlo; Zak; Matthews, 1988; Ribeiro, 2006).

No projeto de um sistema de CEV/MD, é necessário:

a) Projetar uma superfície de deslizamento (i.e., uma superfície de chaveamento ou uma função de chaveamento) de forma que a dinâmica da planta, quando em deslizamento, siga uma trajetória desejada;

b) Determinar uma lógica de chaveamento (i.e., uma lei CEV) de modo a alcançar a superfície de deslizamento em tempo finito, ou seja, ocorrer um modo deslizante, e o sistema de ordem reduzida no modo deslizante seja estável.

2.1.2 Metodologia de Controle com Modo Deslizante

Nesta subseção é tratado principalmente de processos descritos por equações diferenciais não lineares em um espaço de estados n-dimensional arbitrário, com ações de controle vetoriais m-dimensional, conforme considerado em (DeCarlo; Zak; Matthews, 1988):

$$\dot{x}(t) = f(t,x) + g(t,x)u(t), \tag{3}$$

sendo $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $f(t,x) \in \mathbb{R}^n$, e $g(t,x) \in \mathbb{R}^{n\times m}$ e t denotando o tempo. Além disso, cada elemento de $f(t,x)$ e $g(t,x)$ são assumidos contínuos, com derivadas contínuas e limitadas com respeito à t e x.

O controle é selecionado como uma função descontínua do estado. Por exemplo, cada componente do controle $u_i(t)$ pode sofrer descontinuidades em alguma superfície não linear $\sigma_i(x(t)) = 0$ no espaço de estados. Dessa forma, cada entrada $u_i(t)$ do controle chaveado $u(t) \in \mathbb{R}^m$, tem a forma:

$$u_i(t,x) = \begin{cases} u_i^+(t,x) & \text{com} \quad \sigma_i(x(t)) > 0 \\ u_i^-(t,x) & \text{com} \quad \sigma_i(x(t)) < 0 \end{cases} \qquad i = 1,\ldots,m, \tag{4}$$

tal que $\sigma_i(x(t)) = 0$ é a i-ésima superfície de deslizamento associada com a superfície de deslizamento de dimensão $(n-m)$ dada por:

$$\sigma(x(t)) = [\sigma_1(x(t)),\ldots,\sigma_m(x(t))]^T = 0. \tag{5}$$

Além disso, $u_i^+(t,x)$ e $u_i^-(t,x)$ são funções de estado contínuas, com $u_i^+(t,x) \neq u_i^-(t,x)$, e cada função escalar $\sigma_i(x(t))$ determina a descontinuidade da i-ésima componente da função vetorial $u(t)$ (Utkin; Guldner; Shi, 2017).

A superfície de deslizamento $\sigma(x(t)) = 0$ é um espaço fechado $(n-m)$ dimensional em $\mathbb{R}^n$, determinado pela intersecção de m superfícies de chaveamento de dimensão $(n-m)$. Essas superfícies de chaveamento são projetadas tais que a resposta do sistema restrito à $\{x(t)|\sigma(x(t)) = 0\}$, apresente um comportamento desejado em termos de estabilidade, linearidade ou rastreamento.

Dessa forma, considerando a superfície na seguinte forma:

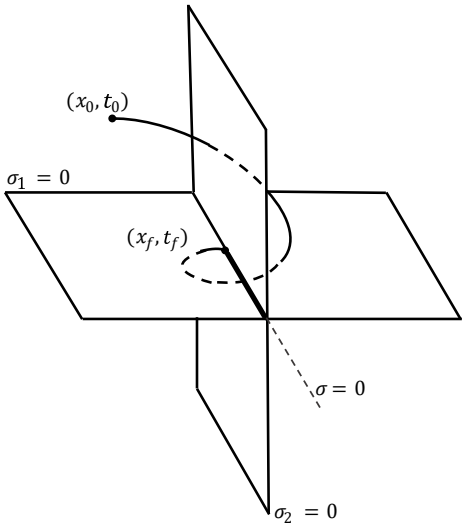
$$\{x(t)|\sigma(x(t)) = Sx(t) = 0\}, \tag{6}$$

tal que S é a matriz da superfície de deslizamento, $S \in \mathbb{R}^{m\times n}$; e simplificando a notação relacionada à superfície de deslizamento, obtém-se:

$$\sigma(x(t)) = Sx(t) = 0. \tag{7}$$

Após o projeto da superfície de deslizamento desejada, a próxima etapa crucial do CEV/MD é assegurar a existência de um modo deslizante. A existência de um modo deslizante ocorre quando, nas proximidades da superfície de deslizamento, $\sigma(x(t)) = 0$, a tangente ou vetor velocidade da trajetória de estado está sempre direcionada para a superfície de deslizamento. Assim, se a trajetória do estado intercepta a superfície de deslizamento, o valor da trajetória de estado permanece dentro de uma vizinhança ϵ de $\{x|\sigma(x) = 0\}$. Se o modo deslizante existe em $\sigma(x(t)) = 0$, então $\sigma(x(t))$ é chamado superfície de deslizamento. Conforme é mostrado na Figura 4, o modo deslizante não pode existir na i-ésima superfície deslizante $\sigma_i(x) = 0$ separadamente, mas somente na intersecção de todas superfícies.

Figura 4 – Modo deslizante na intersecção das i-ésimas superfícies existentes.



Fonte: adaptado de DeCarlo, Zak e Matthews (1988).

Um modo deslizante ideal existe somente quando a trajetória de estado $x(t)$ da planta controlada, satisfaz $\sigma(x(t)) = 0$ para todo $t \geq t_0$, para algum t_0. No entanto,

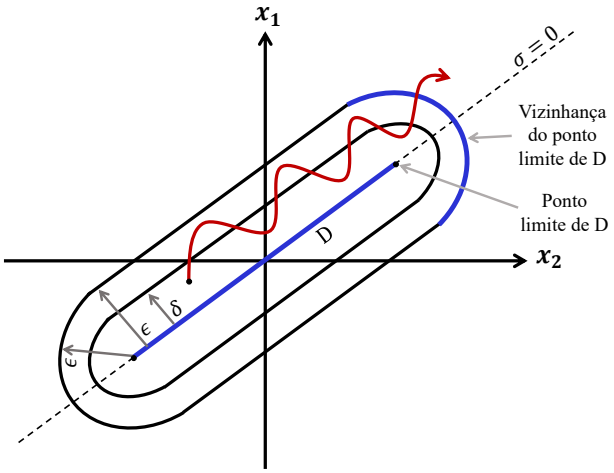
isso exige chaveamentos infinitamente rápidos. Em sistemas reais, todos os equipamentos responsáveis pelo controle chaveado têm imperfeições, como retardamento, histerese, etc., que fazem com que os deslizamentos ocorram em uma frequência finita. A trajetória do estado, então, oscila em uma certa vizinhança da superfície de deslizamento. Esta oscilação é chamada trepidação. Portanto, o modo deslizante real não ocorre sobre as superfícies descontínuas, mas dentro de uma camada limite (Utkin, 1992; Ribeiro, 2006; Esteves, 2016).

A existência de um modo deslizante requer estabilidade da trajetória de estado para a superfície de deslizamento $\sigma(x(t)) = 0$, no mínimo em uma vizinhança de $\{x|\sigma(x(t)) = 0\}$, ou seja, os estados devem aproximar-se da superfície assintoticamente. A vizinhança é chamada região de atração. Geometricamente, o vetor tangente ou derivada no tempo do vetor de estados, deve apontar para a superfície de deslizamento na região de atração (DeCarlo; Zak; Matthews, 1988; Ribeiro, 2006).

O problema de existência assemelha-se ao problema de estabilidade generalizada. Então, o segundo método de Lyapunov fornece um conjunto natural para a análise. Dessa forma, a estabilidade para a superfície de chaveamento requer a seleção de uma função de Lyapunov generalizada $\mathcal{V}(t, x)$, que é definida positiva e tem uma derivada negativa em relação ao tempo, na região de atração (DeCarlo; Zak; Matthews, 1988; Ribeiro, 2006). A seguir, tem-se a Definição 1 descrevendo formalmente o problema:

Definição 1 (Domínio do Modo Deslizante)**.** *Um domínio D no espaço fechado $\sigma = 0$ é um domínio de modo deslizante se para cada $\epsilon > 0$, existe $\delta > 0$, tal que qualquer movimento iniciado dentro de uma vizinhança δ de dimensão n de D pode deixar a vizinhança ϵ de dimensão n de D somente através da vizinhança ϵ de dimensão n da fronteira de D.*

Figura 5 – Ilustração bidimensional do domínio do modo deslizante.



Fonte: adaptado de DeCarlo, Zak e Matthews (1988).

Como a região D está situada na superfície $\sigma(x(t) = 0)$, a dimensão de D é $n - m$.

Teorema 1. *(DeCarlo; Zak; Matthews, 1988) Para o domínio D, de dimensão $(n-m)$ ser o domínio de um modo deslizante, é suficiente que, para $\Omega \supset D$, de dimensão n, exista uma função $\mathcal{V}(t,x,\sigma)$ diferenciável com respeito a todos os seus argumentos, satisfazendo as seguintes condições:*

a) $\mathcal{V}(t,x,\sigma)$ é definida positiva em relação à σ, isto é, $\mathcal{V}(t,x,\sigma) > 0$ com $\sigma \neq 0$ e t, x arbitrários, $\mathcal{V}(t,x,0) = 0$; e na esfera $||\sigma|| = \rho$ para todo $x \in \Omega$ e algum t, tem-se:

$$\inf_{\|\sigma\|=\rho} \mathcal{V}(t,x,\sigma) = h_p, \quad h_p > 0, \tag{8}$$

$$\sup_{\|\sigma\|=\rho} \mathcal{V}(t,x,\sigma) = H_p, \quad H_p > 0, \tag{9}$$

tal que h_p e H_p dependem de $\rho (h_p \neq 0$ se $\rho \neq 0)$.

b) A derivada em relação ao tempo de $\mathcal{V}(t,x,\sigma)$ para o sistema (3) tem um supremo negativo para todo $x \in \Omega$, exceto para x na superfície de deslizamento, onde o controle na entrada não está definido, e por isso a derivada de $\mathcal{V}(t,x,\sigma)$ não existe.

Um modo deslizante é globalmente alcançado se o domínio de atração é todo o espaço de estados. Caso contrário, o domínio de atração é um subconjunto do espaço de estado (DeCarlo; Zak; Matthews, 1988; Ribeiro, 2006; Esteves, 2016).

2.1.3 Condição de Convergência em Tempo Finito

Na abordagem de controle de modo deslizante, definem-se superfícies assintoticamente estáveis $\sigma(x(t))$ de modo que todas as trajetórias do sistema convirjam para essas superfícies em um tempo finito, denominado tempo de alcance t_r. Em seguida, as trajetórias deslizam ao longo dessas superfícies até alcançarem o destino desejado na interseção delas. Para sistemas com uma única entrada, as condições de alcance da superfície são normalmente estabelecidas definindo-se

$$\mathcal{V}(\sigma) = \frac{1}{2}\sigma^2, \tag{10}$$

como a função de Lyapunov, sendo $\mathcal{V}(\sigma)$ positiva definida, e garantindo que sua derivada com respeito ao tempo $\dot{\mathcal{V}}(\sigma)$ seja negativa definida (Slotine; Li, 1991; Shtessel *et al.*, 2014). Para garantir a estabilidade assintótica da superfície escolhida sobre o ponto de equilíbrio $\sigma(x) = 0$, os seguintes critérios precisam ser atendidos (Shtessel *et al.*, 2014):

$$\dot{\mathcal{V}}(\sigma) = \sigma\dot{\sigma} < 0, \quad \text{para} \quad \sigma \neq 0, \tag{11}$$

$$\lim_{|\sigma|\to\infty} \mathcal{V} = \infty. \tag{12}$$

Ambas as condições precisam ser atendidas. Contudo, na prática, a análise se concentra apenas na condição (11), visto que a condição (12) é satisfeita por $\mathcal{V}(\sigma)$ na Equação (10). Dessa forma, derivando (10) em relação ao tempo, tem-se:

$$\dot{\mathcal{V}}(\sigma) = \sigma\dot{\sigma} \leq \frac{-\xi}{\sqrt{2}}|\sigma| < 0, \quad \xi > 0. \tag{13}$$

Isolando σ em (10), tem-se:

$$|\sigma| = \sqrt{2}\,\mathcal{V}^{1/2}. \tag{14}$$

Substituindo (14) em (13), obtém-se:

$$\dot{\mathcal{V}} \leq -\xi\,\mathcal{V}^{1/2}. \tag{15}$$

Separando as variáveis da desigualdade descrita em (15), e integrando-a ao longo do intervalo de tempo $0 \leq \tau \leq t$, resulta-se em:

$$\mathcal{V}^{1/2}(t) \leq -\frac{1}{2}\xi t + \mathcal{V}^{1/2}(0). \tag{16}$$

Consequentemente, $\mathcal{V}(t)$ atinge zero dentro de um intervalo de tempo finito t_r (tempo de alcance), que é limitado por:

$$t_r \leq \frac{2\mathcal{V}^{1/2}(0)}{\xi}. \tag{17}$$

Portanto, o controlador projetado para satisfazer (13), deve conduzir a variável de deslizamento σ a zero em tempo finito, menor ou igual a t_r, bem como mantê-la em zero para todo $t > t_r$ (Shtessel *et al.*, 2014).

2.1.4 Método do Controle Equivalente

Uma forma de realizar a análise para determinar o movimento do sistema restrito à superfície de deslizamento $\sigma(x(t)) = 0$, é pelo método do controle equivalente abordado em Utkin (1992), que é definido como a ação de controle necessária para manter um movimento de deslizamento ideal sobre $\sigma(x)$ (superfície de deslizamento). Segundo Ferrara (2017), o conceito é utilizar a igualdade $\dot{\sigma}(x(t)) = \sigma(x(t)) = 0$, expressa como:

$$\dot{\sigma} = \frac{\partial\sigma}{\partial x}\frac{dx}{dt} = \frac{\partial\sigma}{\partial x}f(t, x, u) = 0. \tag{18}$$

A última é uma equação algébrica em x, u e t. Assim, por meio desta equação é possível obter o sinal de controle equivalente $u_{eq}(t)$, o qual é uma função de controle contínua necessária para manter a igualdade:

$$\frac{\partial\sigma}{\partial x}f(t, x, u_{eq}) = 0. \tag{19}$$

Note que, uma vez que o sinal de controle equivalente é obtido, pode se substituir $u_{eq(t)}$ em $\dot{x}(t) = f(t, x, u)$ para obter uma equação diferencial contínua:

$$\dot{x}(t) = f(t, x, u_{eq}). \tag{20}$$

Portanto, para condições iniciais $\sigma(x(0)) = 0$, e em vista de (18), a solução da equação diferencial ordinária (20) representa as trajetórias dos estados no plano tangencial para o espaço $\sigma(x)$.

Para ilustrar a obtenção deste método, considera-se o sistema (3). Além disso, suponha que o modo deslizante seja aplicado no instante de tempo $t_r \geq t_0$, de modo que se tenha $\sigma(x) = \dot{\sigma}(x) = 0$, para qualquer $t \geq t_r$.

De acordo com o método do controle equivalente apresentado em (Utkin, 1992), pode-se escrever:

$$\sigma(x) = \left[\frac{\partial \sigma}{\partial x}\dot{x}\right] = \left[\frac{\partial \sigma}{\partial x}\right][f(t, x) + g(t, x)u_{eq}(t)] = 0, \tag{21}$$

sendo u_{eq} o controle equivalente. Substituindo u_{eq} em (3), obtém-se:

$$\dot{x}(t) = f(t, x) + g(t, x)u_{eq}(t), \tag{22}$$

que é uma equação diferencial com funções contínuas que descreve a evolução do sistema a partir de $x(t_r)$ e $\sigma(x(t_r)) = 0$.

Suponha que $\frac{\partial \sigma}{\partial x}g(t, x)$ é não singular para todo t e x, isto é:

$$\det\left(\frac{\partial \sigma}{\partial x}g(t, x)\right) \neq 0. \tag{23}$$

Assim,

$$u_{eq} = -\left\{\frac{\partial \sigma}{\partial x}g(t, x)\right\}^{-1}\frac{\partial \sigma}{\partial x}f(t, x). \tag{24}$$

Então, iniciando em t_0 com $\sigma(t_0) = 0$, as dinâmicas do sistema controlado pode ser expressa como:

$$\dot{x} = \left\{I_n - g(t, x)\left[\frac{\partial \sigma}{\partial x}g(t, x)\right]^{-1}\frac{\partial \sigma}{\partial x}\right\}f(t, x), \tag{25}$$

tal que o membro à direita é contínuo e representa a evolução equivalente correspondente do sistema quando controlado pela lei CMD descontínua. Nota-se que, em caso de sistema linear, $\sigma(x) = Cx$, a Equação (25) se torna:

$$\dot{x} = \left\{I_n - g(t, x)\left[Cg(t, x)\right]^{-1}C\right\}f(t, x), \tag{26}$$

com $\partial\sigma/\partial x = C$.

Dessa forma, a lei de controle baseada em CMD pode consistir de componentes principais: controle equivalente e controle chaveado, expressa como

$$u = u_{eq} + u_n. \tag{27}$$

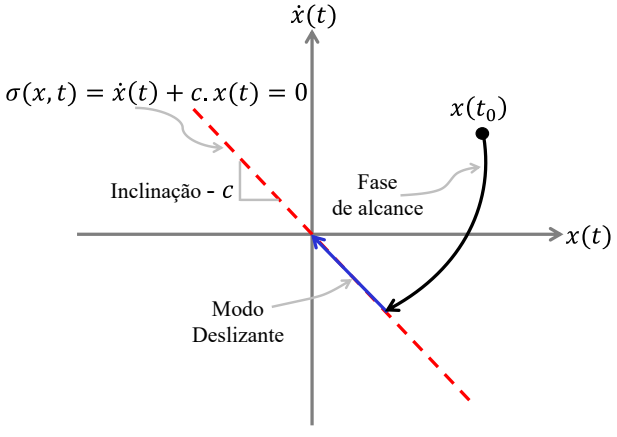
A componente de chaveamento da lei de controle (u_n) leva o sistema à função de deslizamento, e a componente equivalente (u_{eq}) é importante quando o sistema está em condição de deslizamento, pois esta é definida como a ação de controle necessária para manter um movimento de deslizamento ideal sobre a superfície de deslizamento $\sigma(x) = 0$ (Eker, 2010; Mozayan *et al.*, 2016; Krim *et al.*, 2018; Gambhire *et al.*, 2021).

2.1.5 Sistema de Controle com Modo Deslizante Utilizando a Abordagem da Lei de Alcance

A condição na qual o estado se moverá em direção a uma superfície de deslizamento e a alcançará, é chamada de condição de alcance. A trajetória do sistema sob a condição de alcance é denominada modo de alcance ou fase de alcance. Três abordagens para especificar a condição de alcance foram propostas na literatura: a abordagem da função de chaveamento direta, a abordagem da função Lyapunov, e a abordagem da lei de alcance (Hung; Gao; Hung, 1993; Ramezani-al; Tavanaei-Sereshki, 2019).

Pode-se observar, a partir da revisão da literatura, que a dinâmica transitória de um sistema de controle com estrutura variável pode compreender dois modos: um modo de alcance, seguido por um modo deslizante. Na Figura 6 é mostrado o movimento da trajetória do CMD convencional, consistindo em movimentos de modo de aproximação e deslizamento, sendo esse exemplo, uma interpretação gráfica para a função de deslizamento.

Figura 6 – Interpretação gráfica para uma função de chaveamento $\sigma(x, t)$.



Fonte: próprio autor.

Na primeira fase, denominada modo de alcance, a trajetória do sistema é forçada a deslocar-se de uma condição inicial $x(t_0)$ até a função de deslizamento previamente definida ($\sigma(x,t)$). Na segunda fase, o controlador CMD mantém os estados sobre a função de deslizamento ao longo de todo o tempo subsequente, conforme Figura 6.

Dessa forma, o projeto do CEV/MD possui dois estágios: um é a superfície do modo deslizante e o outro é a lei de alcance. No entanto, a lei de alcance pode gerar fenômenos de trepidação no sistema, como já mencionado anteriormente. Este problema tem motivado os pesquisadores a implementar técnicas adequadas com leis de alcance para reduzir o *chattering*. A dinâmica desejada em modo deslizante geralmente é caracterizada por uma resposta rápida e estável, livre de erros e sem sobressinal (do inglês,*overshoot*) (Gao; Hung, 1993; Gao; Wang; Homaifa, 1995; Yu; Kaynak, 2009; Kumar; Kumar; Ganesh, 2016; Liao; Xu, 2017; Wang *et al.*, 2019; Junejo *et al.*, 2020). Na fase de alcance os estados do sistema deve ser conduzido à superfície de deslizamento (ou função de deslizamento) em tempo finito, enquanto na fase de deslizamento os estados devem deslizar até o ponto de equilíbrio.

A lei de alcance, é uma equação diferencial que especifica a dinâmica de uma função de deslizamento, $\sigma(x)$. Além disso, através da seleção de parâmetros nessa equação diferencial, é possível controlar o comportamento dinâmico do sistema CEV/MD na fase de alcance, bem como diminuir efetivamente o *chattering* através de uma lei de alcance cuidadosamente elaborada (Singh; Holé, 2004; Fallaha *et al.*, 2010; Moldoveanu, 2014; Wang; Jia; Dong, 2013; Bartoszewicz, 2015; Sun *et al.*, 2021; Wu *et al.*, 2021; Komurcugil *et al.*, 2023).

Esta lei de alcance pode ser descrita pela seguinte equação diferencial:

$$\dot{\sigma}(x) = -\eta\,\mathrm{sgn}(\sigma(x)) - kf(\sigma(x)), \quad \eta > 0, k > 0, \tag{28}$$

sendo $f(0) = 0$, e quando $\sigma \neq 0$, $\sigma f(\sigma) > 0$. A equação (28) é apenas uma forma geral da abordagem da lei de alcance. Aliás, há diversas leis de alcance, e alguns casos especiais são:

a) A lei de alcance com taxa constante:

$$\dot{\sigma}(x) = -\eta\mathrm{sgn}(\sigma(x)), \quad \eta > 0, \tag{29}$$

b) A lei de alcance com taxa constante e proporcional:

$$\dot{\sigma}(x) = -\eta\mathrm{sgn}(\sigma(x)) - k\sigma(x), \quad \eta > 0, k > 0, \tag{30}$$

c) A lei de alcance com taxa de potência:

$$\dot{\sigma}(x) = -k|\sigma(x)|^{\alpha}\mathrm{sgn}(\sigma(x)), \quad 0 < \alpha < 1. \tag{31}$$

A abordagem da lei de alcance não apenas assegura a condição de alcance, mas também especifica as características dinâmicas do movimento durante a fase de alcance (He; Xu; Cheng, 2010; Devika; Thomas, 2018; Mohd Zaihidee; Mekhilef; Mubin, 2019; Jeeranantasin; Nungam, 2022).

2.2 CONTROLE *SWING UP*

O controle *swing up*, é a transferência do pêndulo de uma posição de pêndulo simples, para a posição invertida. Diferentes métodos de controle *swing up* para pêndulo invertido tem sido considerado por pesquisadores, como por exemplo, em Furuta, Yamakita e Kobayashi (1992) foi proposto um controle *swing up* usando *pseudo-state feedback*, em Ohsumi e Izumikawa (1995) foi simulado e aplicado a técnica de *feedback linearization* para elevar o pêndulo na região de estabilidade invertida; em Åström e Furuta (1996) e Åström (1999) foi apresentado um conceito de controle de energia para o controle *swing up* do pêndulo invertido. Neste trabalho é adotado o método de controle de energia para realizar o controle *swing up*.

A energia é acumulada no sistema PIRR a fim de aumentar o nível de energia do pêndulo. A forma de inserir essa energia é aplicando alguma aceleração ao pivô do pêndulo, ou seja, ao ângulo da haste, cuja direção é determinada pela informação da taxa de energia inserida. Assim, o pêndulo consegue alcançar a região invertida, utilizando a própria energia acumulada do movimento oscilatório para impulsionar o movimento do pêndulo (Muskinja; Tovornik, 2006; Mathew; Rao; Sivakumaran, 2013; Abdullah *et al.*, 2021).

Conforme Åström e Furuta (2000), a derivada da função de Lyapunov $L_y = \frac{(E-E_{ref})^2}{2}$ com a lei de controle $u = k(E - E_{ref})\,\dot{\theta}(t)\cos\theta(t)$ implica que a função decresce considerando que não esteja nos pontos onde a controlabilidade é perdida. Então, uma outra lei de controle é desenvolvida na qual $u = \text{sat}(k(E - E_{ref})\,\text{sgn}(\dot{\theta}(t)\cos(\theta(t)))$ resultando na estratégia que atua como controlador linear. A função energia de referência E_{ref} faz com que o controlador atue até atingir o valor desejado, onde a energia potencial é nula, isto é válido enquanto o cosseno da posição $\theta(t)$ e aceleração $\ddot{\theta}(t)$ do pêndulo forem diferentes de 0 (Åström; Furuta, 2000).

De acordo com Alves, Neves e Angélico (2019), uma forma simplificada do controle *swing up* proposto em Åström e Furuta (2000) pode ser considerada, tal que:

$$u_{sw}(t) = \quad -K_{sw}\theta^n(t)\,\text{sgn}(\dot{\theta}(t)\cos(\theta(t)), \tag{32}$$

sendo que a constante K_{sw} pode ser ajustada por meio de ensaios, e n é o termo relacionado com a relação de redução do motor $(1 : n)$. O termo $\theta^n(t)$ pondera o torque aplicado pelo motor na roda, de modo que para valores de ângulos próximos à 0° (ponto de equilíbrio

instável), sua atuação é baixa, enquanto para ângulos maiores há um aumento. O sinal de controle deverá ser positivo enquanto o termo $\dot{\theta}(t)\cos\theta(t)$ for negativo. Desta forma, em conjunto com o termo $\theta^n(t)$ a velocidade do pêndulo é aumentada gradualmente até alcançar a região próxima a posição desejada.

2.3 SISTEMAS CONTÍNUOS EM ESPAÇO DE ESTADOS

Um sistema dinâmico linear invariante no tempo pode ser representado na seguinte forma

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{33}$$

quando ele se comporta a tempo contínuo, sendo $u(t) \in \mathbb{R}^{p\times 1}$ a entrada controladora, $x(t) \in \mathbb{R}^{n\times 1}$ o vetor dos estados do sistema, $y(t) \in \mathbb{R}^{q\times 1}$ o vetor de saída; as matrizes $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{q\times n}$ e $D \in \mathbb{R}^{q\times p}$, são as matrizes com parâmetros do sistema. As matrizes A, C, B, D têm dimensões compatíveis.

Aplicando-se a transformada de Laplace a ambos os lados da equação de estado dada em (33), resulta

$$sX(s) - x(0) = AX(s) + BU(s). \tag{34}$$

Substituindo $sX(s)$ por $sIX(s)$, em que I é uma matriz identidade $n \times n$, e n é a ordem do sistema. Combinando-se todos os termos em $X(s)$, obtém-se

$$(sI - A)X(s) = x(0) + BU(s). \tag{35}$$

Resolvendo para $X(s)$, multiplicando à esquerda ambos os lados da equação (35) por $(sI - A)^{-1}$, a solução final para $X(s)$ é

$$\begin{aligned} X(s) = &\ (sI - A)^{-1}x(0) + (sI - A)^{-1}BU(s), \\ = &\ \frac{\text{adj}(sI - A)}{\det(sI - A)}[x(0) + BU(s)], \end{aligned} \tag{36}$$

sendo o denominador de (36) os autovalores da matriz A.

Aplicando-se a transformada de Laplace à equação de saída dada em (33) resulta

$$Y(s) = CX(s) + DU(s). \tag{37}$$

Para mostrar que os autovalores são iguais aos polos da função de transferência do sistema, considere que a saída e a entrada sejam grandezas escalares $Y(s)$ e $U(s)$, respectivamente (Nise, 2012). Além disso, considere $x(0)$ igual a 0. Substituindo-se a equação (36) na equação (37) e resolvendo para a função de transferência, $\frac{Y(s)}{U(s)}$, resulta

$$\begin{aligned} \frac{Y(s)}{U(s)} &= C\left[\frac{\text{adj}(sI-A)}{\det(sI-A)}\right]B + D, \\ &= \frac{C\,\text{adj}(sI-A)\,B + D\det(sI-A)}{\det(sI-A)}, \end{aligned} \tag{38}$$

tal que as raízes do denominador de (38), são os polos do sistema. Uma vez que os denominadores das equações (36) e (38) são idênticos, os polos do sistema são iguais aos autovalores (Nise, 2012). Assim, se um sistema é representado no espaço de estados, pode-se determinar os polos a partir de $\det(sI-A) = 0$. Dessa forma, a análise de estabilidade de um sistema linear é essencialmente baseada na matriz que define sua dinâmica $A \in \mathbb{R}^{n\times n}$.

Por fim, a abordagem de representação em espaço de estados (teoria de controle moderno) baseia-se na descrição dos sistemas dinâmicos em função de seus estados, possibilitando o tratamento de sistemas de várias entradas e várias saídas (do inglês, *Multiple-Input and Multiple-Output* - MIMO). Na subseção a seguir, será discutida a determinação do conceito de estabilidade.

2.3.1 Estabilidade de Sistemas Dinâmicos Contínuo no Tempo

Nesta subseção é abordada a estabilidade de sistemas dinâmicos, que é de extrema importância para o projeto de sistemas de controle. Considera-se o conceito de estabilidade para o caso em que o sistema (33) não possui entrada forçada, isto é, $u(t) = 0$. A seguir é definido o conceito de ponto de equilíbrio e estabilidade de sistemas dinâmicos.

Definição 2 (Ponto de Equilíbrio)**.** *Um ponto x_e é um ponto de equilíbrio de um sistema dinâmico se, uma vez que o vetor de estado do sistema assume o valor x_e, este permanece com o valor x_e em todo tempo futuro.*

Desta forma, x_e é ponto de equilíbrio de (33) supondo $u(t) = 0$, se $x(t_0) = x_e$ implica $x(t) = x_e$ para todo $t \geq t_0 \geq 0$. Portanto, como o vetor de estado deve permanecer constante nos pontos de equilíbrio, pode-se notar que x_e satisfaz $Ax_e = 0$, ou seja, quando um sistema está em um estado de equilíbrio, a derivada dos estados é nula e, portanto, os estados não variam no tempo. Note que, para o sistema (33) com $u(t) = 0$, se A for uma matriz constante e não singular, o único ponto de equilíbrio é a origem (Slotine; Li, 1991).

Definição 3 (Estabilidade)**.** *O ponto de equilíbrio $x_e = 0$ é dito estável se, para qualquer $\epsilon > 0$, existe $\delta(\epsilon)$ tal que.*

(i) $\|x(0)\|_2 < \delta(\epsilon) \Rightarrow \|x(t)\|_2 < \epsilon$, $\forall\, t \geq 0$.
Caso contrário, o sistema é dito instável. Adicionalmente, a origem é dita assintóticamente estável se for estável e se, para qualquer condição inicial,

(ii) $\lim_{t\to\infty} x(t) = 0$.

Ou seja, partindo de qualquer condição inicial, as variáveis de estado do sistema, sempre convergem para o ponto de equilíbrio $x = 0$ quando $t \to \infty$ (Slotine; Li, 1991). Pode-se mostrar que o sistema (33) é assintóticamente estável se, e somente se, a matriz A for Hurwitz (matriz cujos autovalores têm parte real negativa).

2.3.2 Controlabilidade e Observabilidade

Como é de interesse realimentar o sistema para poder controlá-lo, deve-se verificar antes disso, se o sistema é controlável. Controlabilidade define se o estado da equação de espaço de estado, pode ou não, ser controlado a partir da entrada, e observabilidade define se o estado inicial pode ou não, ser observado a partir da saída (Chen, 1993). Em seguida os conceitos de controlabilidade e observabilidade de sistemas dinâmicos são formalizados.

Definição 4 (Controlabilidade)**.** *(Chen, 1999) O sistema descrito no espaço de estado (33), ou par (A, B), é dito ser controlável se para qualquer condição inicial $x(0) = x_0$ e qualquer estado final x_1, existe uma entrada que transfere x_0 para x_1 em um tempo finito. Do contrário (A, B) é dito ser não controlável.*

Sabe-se que o sistema linear é controlável, se o posto (*rank*) da matriz de controlabilidade for completo, ou seja, $C_{ctrb} = [B\ AB \dots A^{(n-1)}\, B]$ deve ser inversível (não singular), desta forma a matriz C_{ctrb} deve ter determinante diferente de zero para que o par (A, B) seja controlável.

Definição 5 (Observabilidade)**.** *(Chen, 1999) O sistema descrito no espaço de estado (33), é dito ser observável se todo o estado $x(t_0)$ puder ser determinado pela observação de $y(t)$ durante um intervalo finito de tempo $t_0 \leq t \leq t_1$. Caso contrário, o sistema é dito ser não observável.*

Portanto, sabe-se que um sistema é observável, se o posto (*rank*) da matriz de observabilidade for completo, ou seja, $O_{obsv} = [C\ CA \dots CA^{(n-1)}]^T$ deve ser inversível (não singular), assim a matriz O_{obsv} deve ter determinante diferente de zero para que o par (A,C) seja observável.

O conceito de observabilidade é útil na solução de problemas de reconstrução de variáveis de estado não mensuráveis a partir de variáveis mensuráveis. Se todo o estado for observável, então o sistema será considerado de estado completamente observável, (Ogata, 2010).

2.4 REALIMENTAÇÃO DE ESTADOS

A abordagem do projeto de realimentação de estado utiliza este modo de descrever a planta, descrito em (33), que assume a influência de um sinal de entrada $u(t)$ na dinâmica do sistema, descrita através da matriz de entrada B. A finalidade da lei de controle, é permitir o projeto de alocação de autovalores para o sistema em malha fechada, o qual apresente uma resposta dinâmica satisfatória, em termos de tempo de subida, sobressinal, ou outros parâmetros da resposta transitória (Franklin; Powell; Workman, 1998). A lei de controle por realimentação de estados é dada por:

$$u(t) = -Kx(t) = -\begin{bmatrix} K_1 & K_2 & \dots & K_n \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}, \tag{39}$$

sendo que vetor de estado ($x_1(t)$ à $x_n(t)$) está disponível para realimentação.

O objetivo de controle na realimentação de estado, é projetar uma matriz de ganhos K tal que a lei de controle $u(t) = -Kx(t)$ estabilize assintoticamente o sistema (33). Assim, a dinâmica do sistema em malha fechada é descrita pela equação a seguir

$$\dot{x}(t) = (A - BK)x(t). \tag{40}$$

Este projeto de ganhos K pode ser realizado de diversas formas, uma delas é aplicando a fórmula de Ackerman, que por meio de uma escolha adequada de K, pode-se alocar todos os autovalores com partes reais negativas, fazendo com que $x(t)$ tenda a zero, quando t tende a infinito, ou seja, atendendo ao requisito de estabilidade assintótica do sistema em malha fechada.

2.5 CONTROLADORES PID

O modelo matemático de um controlador PID, conforme o padrão ISA (do inglês, *International Society of Automation*), é dado por:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^T e(t)dt + T_d\frac{de(t)}{dt}\right), \tag{41}$$

sendo $u(t)$ o sinal de saída do controlador, chamado de variável manipulada, $e(t) = (y_{sp}(t) - y(t))$ o sinal de entrada do controlador, chamado de erro atuante. Os parâmetros de ajuste do controlador são K, T_i e T_d. As três componentes do controlador PID correspondem aos efeitos Proporcional, Integrador e Derivador do sinal de erro atuante (Åström; Hägglund, 1995; Castrucci; Bittar; Sales, 2018; Dorf; Bishop, 2009).

Tradicionalmente, utiliza-se a seguinte denominação relativa aos parâmetros de ajuste do controlador PID: $K_p = K$ é o ganho proporcional, $K_i = \frac{K}{T_i}$ o ganho integral, e $K_d = KT_d$ o ganho derivativo. No domínio da frequência (Laplace), a equação (41) é:

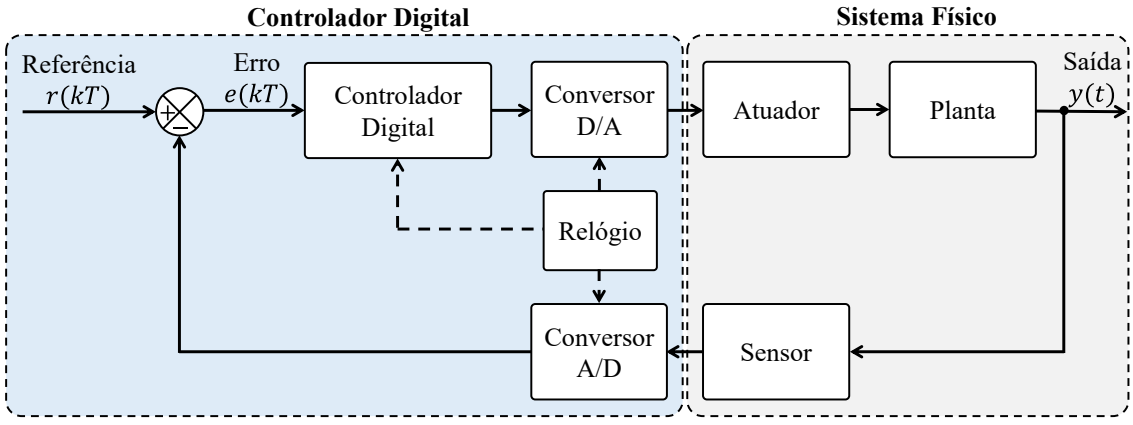
$$\begin{aligned} U_p(s) &= KE(s), \\ U_i(s) &= \frac{K}{sT_i}E(s), \\ U_d(s) &= KsT_dE(s). \end{aligned} \tag{42}$$

2.6 SISTEMA DE CONTROLE DIGITAL

Um controlador digital oferece ao projetista maior flexibilidade para modificar a lei de controle após o desenvolvimento, já que o cálculo do sinal de controle é feito por meio de um programa, em vez de depender de um circuito analógico (Franklin; Powell; Emami-Naeini, 2013; Castrucci; Bittar; Sales, 2018). Neste trabalho, a planta PIRR é de representação contínua no tempo, e o controlador implementado no microcontrolador ESP32 é um controlador digital.

Na Figura 7 é apresentado um diagrama de blocos de um sistema em malha fechada cujo controle é realizado por um controlador digital. A planta corresponde ao sistema dinâmico que deve ser controlado. O atuador, responsável por fornecer potência, permite que o controlador interfira diretamente na planta. O sinal emitido pelo sensor é inicialmente analógico, sendo convertido em digital por um conversor analógico-digital (A/D).

Figura 7 – Diagrama de blocos de um controlador digital.



Fonte: próprio autor.

O conversor A/D é utilizado para obter as amostras da saída $y(kT)$, $k \in \mathbb{Z}$, a cada $T \in \mathbb{R}$ segundos, em que T é o período de amostragem. Essas quantidades são convertidas em números binários que são processados pelo microcontrolador.

Um controlador discreto pode ser obtido de maneira a reproduzir, aproximadamente, o mesmo comportamento de um controlador em tempo contínuo. Obter um equi-

valente discreto para um controlador contínuo já projetado é uma técnica conhecida como emulação.

Após o controlador digital calcular o valor apropriado do sinal de controle, esse valor pode ser convertido por um conversor digital-analógico (D/A), para então ser aplicado ao atuador do processo. Nesta etapa pode-se optar por um sinal de tensão por modulação por largura de pulsos.

O sincronismo de conversão dos sinais, é realizado por um relógio, portanto, o *timer* garante que o sistema discreto irá trabalhar com um período de amostragem constante. Em geral, a referência (sinal de entrada) é gerada internamente pelo processador (ou microcontrolador).

2.6.1 Controladores PID Digitais

Atualmente, é comum que os controladores PID sejam implementados com microprocessadores. Dessa forma, diversos fatores devem ser levados em conta na implementação digital, tais como amostragem e discretização (Åström; Hägglund, 1995; Alves, 2013; Loghis; Xiros, 2022).

2.6.1.1 Amostragem

Quando um computador digital é usado para implementar uma lei de controle, todo o processamento de sinal é realizado em instantes de tempo discreto. A sequência de operações é a seguinte (Åström; Hägglund, 2006):

a) Esperar pela interrupção do relógio;

b) Ler a entrada analógica;

c) Computar o sinal de controle;

d) Calcular a saída analógica;

e) Atualizar as variáveis de controle;

f) Voltar ao passo (a).

A ação de controle é baseada nos valores da saída do processo apenas em tempos discretos. Esse procedimento é chamado de amostragem (Åström; Hägglund, 1995; Alves, 2013).

2.6.1.2 Discretização

Para implementar uma lei de controle em tempo contínuo, como um controlador PID em um computador digital, é necessário aproximar as derivadas e a integral que aparecem na lei de controle (Åström; Hägglund, 2006; Abramovitch, 2015; Durand; Guerrero-Castellanos, 2015; Abramovitch, 2023). Entre as diferentes maneiras de se fazer isso, são apresentadas alguns exemplos a seguir.

O termo proporcional em sistema contínuo

$$u_p(t) = K(y_{sp}(t) - y(t)), \tag{43}$$

deve ser simplesmente substituído pelo equivalente discreto:

$$u_p(t_k) = K(y_{sp}(t_k) - y(t_k)). \tag{44}$$

O termo integral no modo analógico é dado por:

$$u_i(t) = \frac{1}{T_i} \int_0^T e(t)dt. \tag{45}$$

Segue-se que

$$\frac{du_i}{dt} = \frac{K}{T_i}\, e. \tag{46}$$

Aproximando a derivada por uma diferença progressiva pelo método de Euler (*forward difference*), resulta-se em

$$\frac{u_i(t_{k+1}) - u_i(t_k)}{T_s} = \frac{K}{T_i}\, e(t_k). \tag{47}$$

Isso leva à seguinte equação recursiva para o termo integral

$$u_i(t_{k+1}) = u_i(t_k) + \frac{K\, T_s}{T_i}\, e(t_k), \tag{48}$$

sendo t_k o instante de amostragem, t_{k+1} o instante da amostragem seguinte, e T_s o intervalo de tempo entre amostragens (ou período de amostragem).

O termo derivativo no modo analógico com o filtro clássico de primeira ordem, é dado por:

$$u_d = -\frac{T_d}{N}\frac{du_d}{dt} - KT_d\frac{dy}{dt}, \tag{49}$$

sendo N o ganho do filtro, e o ganho da parte derivativa fica limitado a KN, mesmo para altas frequências (o valor típico para N é entre 8 e 20).

Aproximando a derivada por uma diferença regressiva (*backward difference*) obtém-se

$$\frac{T_d}{N}\frac{u_d(t_k) - u_d(t_{k-1})}{T_s} + u_d(t_k) = -KT_d\frac{y(t_k) - y(t_{k-1})}{T_s}. \tag{50}$$

Isso pode ser reescrito como

$$u_d(t_k) = \frac{T_d}{T_d + NT_s} u_d(t_{k-1}) - \frac{KT_dN}{T_d + NT_s}(y(t_k) - y(t_{k-1})). \tag{51}$$

No Capítulo 3 será apresentada a modelagem dinâmica do pêndulo invertido por roda de reação.

3 MODELAGEM DINÂMICA DO PÊNDULO INVERTIDO POR RODA DE REAÇÃO

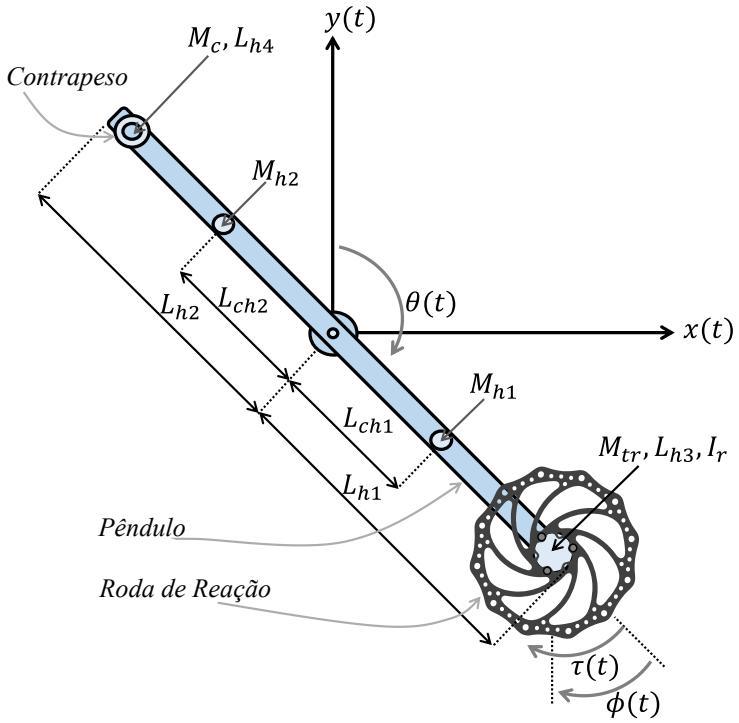
Neste capítulo é descrita a modelagem dinâmica do sistema PIRR. Na Seção 3.1 é apresentada a modelagem matemática do sistema pela equação de Euler-Lagrange, e modelo linearizado em torno do ponto de equilíbrio instável $x_e = [0; 0; 0; 0]$. Na Seção 3.2 são apresentadas as equações que descrevem a dinâmica do motor de corrente contínua a ser implementada na equação de movimento do PIRR.

3.1 MODELAGEM MATEMÁTICA DO PIRR

A modelagem dinâmica pela equação de Euler-Lagrange do PIRR, é apresentada nesta seção. Na Figura 8 é apresentado um diagrama esquemático do sistema.

É levado em consideração o sistema de coordenadas generalizadas, com $\theta(t)$ sendo a posição angular do pêndulo, e $\phi(t)$ a posição angular da roda. Além disso, é ilustrado o momento de inércia da roda I_r, sendo $\tau(t)$ o torque motor aplicado na roda de reação pelo motor.

Figura 8 – Sistema pêndulo invertido por roda de reação.



Fonte: adaptado de Fantoni e Lozano (2002).

Os parâmetros $M_{tr} = M_a + M_m + M_r$, M_c, e M_h, representam as massas de cada componente, sendo M_a a massa do acoplamento da roda com o motor, M_m a massa do

motor e M_r representa a massa da roda, M_c a massa do contrapeso, e $M_{hk}, k = 1, 2$ a massa das hastes no qual $M_{h1} = M_{h2}$ constitui um caso particular do modelo proposto deste trabalho. O comprimento da haste a partir do pivô, é dado por $L_{hi}, i = 1, 2, 3, 4$, sendo L_{h1} o comprimento da haste da roda com relação ao pivô, e L_{h2} o comprimento da haste do contrapeso com relação ao pivô, L_{h3} é a distância do centro de massa do conjunto de massas na roda em relação ao pivô, e L_{h4} é a distância do centro de massa do contrapeso em relação ao pivô, sendo $L_{h1} = L_{h2}$ um caso particular do presente trabalho. Finalmente, é definida a distância do centro de massa das hastes em relação ao pivô, que é dada por $L_{chj}, j = 1, 2$ em que $L_{ch1} = L_{ch2}$ corresponde a um caso particular deste trabalho. A partir dos parâmetros definidos neste parágrafo, são consideradas as seguintes quantidades para simplificar o arranjo de equações de dinâmica do pêndulo:

$$\begin{aligned} M_{tr} &= M_a + M_m + M_r, \\ I_p &= \frac{1}{3}M_{h1}L_{h1}^2 + \frac{1}{3}M_{h2}L_{h2}^2 + M_{tr}L_{h3}^2 + M_cL_{h4} + I_r, \\ \bar{M}_1 &= M_{h1}L_{ch1} + M_{tr}L_{h3}, \\ \bar{M}_2 &= M_{h2}L_{ch2} + M_cL_{h4}. \end{aligned} \tag{52}$$

A equação dinâmica de Lagrange é formulada usando a Lagrangiana ($L = E_{ct} - E_{pg}$), e definindo-se um conjunto de coordenadas generalizadas, $q_1, \ldots, q_n$, para representar um sistema de n graus de liberdade:

$$L_{ag}(q_i, \dot{q}_i, t) = E_{ct}(q_i, \dot{q}_i, t) - E_{pg}(q_i), \tag{53}$$

tal que $E_{ct}(q_i, \dot{q}_i, t)$ é a energia cinética do sistema e $E_{pg}(q_i)$ é a energia potencial do sistema. Assim, as equações dinâmicas são expressas em termos da Lagrangiana da seguinte forma:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}(q_i, \dot{q}_i)\right) - \frac{\partial L}{\partial q_i}(q_i, \dot{q}_i) = \tau_i, \quad i = 1, \ldots, n, \tag{54}$$

sendo τ_i o torque generalizado do atuador.

No sistema, há energia cinética do pêndulo, rotor e motor; a soma dessas energias representa a energia cinética total E_{ct} que está definida a seguir:

$$E_{ct} = \frac{1}{2}I_p\,\dot{\theta}^2(t) + I_r\,\dot{\theta}(t)\dot{\phi}(t) + \frac{1}{2}I_r\,\dot{\phi}^2(t). \tag{55}$$

A energia potencial obtida desconsidera a elasticidade presente no eixo do motor ou na haste do pêndulo, sendo apenas a energia potencial gravitacional E_{pg} considerada como:

$$E_{pg} = \bar{M}_1\,g\,(\cos(\theta(t)) - 1) + \bar{M}_2\,g\,(1 - \cos(\theta(t))). \tag{56}$$

Como a haste possui massa aproximadamente semelhante nos dois extremos, causada pelos encaixes na haste, suas diferenças resultam próximo de 0, e portanto é desconsiderada da equação. Com as equações (55) e (56), e aplicando a equação de Euler-Lagrange

(53), obtém-se:

$$L_{ag}(q_i, \dot{q}_i, t) = E_{ct}(q_i, \dot{q}_i, t) - E_{pg}(q_i),$$

$$\begin{aligned} L_{ag}(q_i, \dot{q}_i) =& \frac{1}{2} I_p \, \dot{\theta}^2(t) + I_r \, \dot{\theta}(t)\dot{\phi}(t) + \frac{1}{2} I_r \, \dot{\phi}^2(t) + \\ &+ \bar{M}_1 \, g \, (1 - \cos(\theta(t))) + \bar{M}_2 \, g \, (\cos(\theta(t)) - 1). \end{aligned} \tag{57}$$

Então, é realizada a derivada parcial da equação (57) considerando (54). Desta forma, resultam-se as equações de movimento (58) e (59), que descrevem o sistema PIRR.

$$I_p \ddot{\theta}(t) + I_r \ddot{\phi}(t) - \bar{M}_1 \, g \, \text{sen}(\theta(t)) + \bar{M}_2 \, g \, \text{sen}(\theta(t)) = 0, \tag{58}$$

$$I_r \ddot{\theta}(t) + I_r \ddot{\phi} = \tau. \tag{59}$$

A partir da equação dinâmica do sistema, pode-se isolar (58) em função de $\ddot{\theta}$ e (59) em função de $\ddot{\phi}$, para obter uma representação do comportamento do sistema em função da aceleração do pêndulo e da roda de reação. Por meio de substituições de $\ddot{\theta}$ na equação (59), e $\ddot{\phi}$ na (58), obtém-se (60), que são as equações diferenciais que representam a dinâmica do PIRR em função do tempo.

$$\begin{aligned} \ddot{\theta}(t) = & \quad \frac{1}{I_r I_p - I_r^2} \left(I_r \bar{M}_1 \, g \, \text{sen}(\theta(t)) - I_r \bar{M}_2 \, g \, \text{sen}(\theta(t)) - I_r \tau(t) \right), \\ \ddot{\phi}(t) = & \quad \frac{1}{I_r I_p - I_r^2} \left(-I_r \bar{M}_1 \, g \, \text{sen}(\theta(t)) + I_r \bar{M}_2 \, g \, \text{sen}(\theta(t)) + I_p \tau(t) \right). \end{aligned} \tag{60}$$

É desejado linearizar o sistema em um ponto de equilíbrio, pois a equação linearizada possibilita o projeto de controle linear. Assim, considerando o ponto de equilíbrio $x_e = [0; 0; 0; 0]$ (instável), a equação pode ser linearizada em torno deste ponto de operação, pois para pequenos valores de $\theta(t)$, pode-se assumir que $\text{sen}(\theta(t)) \approx \theta(t)$. Desta forma, o modelo linearizado do PIRR é descrito como:

$$\begin{aligned} \ddot{\theta}(t) = & \quad \frac{1}{I_r I_p - I_r^2} \left(I_r \bar{M}_1 \, g \, \theta(t) - I_r \bar{M}_2 \, g \, \theta(t) - I_r \tau(t) \right), \\ \ddot{\phi}(t) = & \quad \frac{1}{I_r I_p - I_r^2} \left(-I_r \bar{M}_1 \, g \, \theta(t) + I_r \bar{M}_2 \, g \, \theta(t) + I_p \tau(t) \right). \end{aligned} \tag{61}$$

3.2 DINÂMICA DO MOTOR CC

A dinâmica do motor de corrente contínua (CC) deve ser implementada na equação de movimento do PIRR a partir de um rearranjo de equações, de modo que através do sistema em malha fechada, permita fornecer a tensão elétrica necessária ao sistema com base na entrada de controle. As equações que descrevem a dinâmica do motor de corrente contínua, considerando o fluxo constante, podem ser descritas a partir de (62) e (63).

$$\tau(t) = n K_t I_a(t) - A_m \dot{\phi}(t), \tag{62}$$

sendo K_t a constante de torque e A_m o atrito presente nas escovas do motor CC. A corrente de armadura representada por $I_a(t)$ é descrita através de:

$$V(t) = R_a I_a(t) + nK_v\dot{\phi}(t), \tag{63}$$

em que $V(t)$ é a tensão a ser aplicada ao motor CC, n é a redução do motor, R_a a resistência de armadura, e K_v a constante de força eletromotriz. A indutância presente no motor é desprezada das equações para o acoplamento das dinâmicas. Isolando (63) em função da corrente de armadura, e aplicando-a na equação (62), resulta em:

$$\tau(t) = \frac{nK_tV(t)}{R_a} - \frac{n^2K_tK_v\dot{\phi}(t)}{R_a} - A_m\dot{\phi}(t), \tag{64}$$

que representa o torque em função da tensão de alimentação, que foi definida como a entrada de controle do sistema. Finalmente, substituindo a equação (64) em (61) obtém-se:

$$\begin{aligned}
\ddot{\theta}(t) = &\ \frac{I_r\bar{M}_1\, g\,\theta(t)}{C_i} - \frac{I_r\bar{M}_2\, g\,\theta(t)}{C_i} - \frac{I_rnK_tV(t)}{C_iR_a} + \frac{I_rn^2K_tK_v\dot{\phi}(t)}{C_iR_a} + \frac{I_rA_m\dot{\phi}(t)}{C_i}, \\
\ddot{\phi}(t) = &\ \frac{I_r\bar{M}_2\, g\,\theta(t)}{C_i} - \frac{I_r\bar{M}_1\, g\,\theta(t)}{C_i} + \frac{I_pnK_tV(t)}{C_iR_a} - \frac{I_pn^2K_tK_v\dot{\phi}(t)}{C_iR_a} - \frac{I_pA_m\dot{\phi}(t)}{C_i},
\end{aligned} \tag{65}$$

sendo $C_i = I_rI_p - I_r^2$ uma simplificação para as constantes de inércia. Esta equação (65) será utilizada posteriormente para descrever o sistema em espaços de estados.

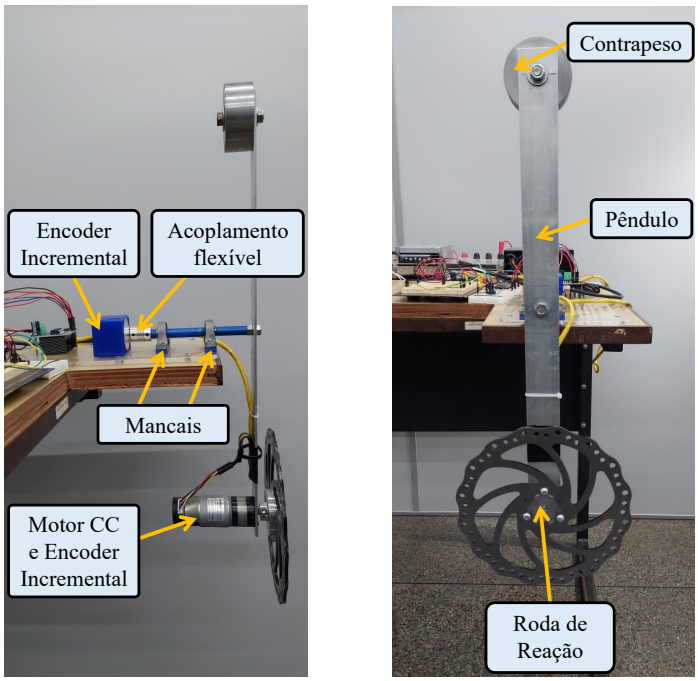
4 MATERIAIS E MÉTODOS

O objetivo deste capítulo é apresentar as metodologias adotadas, com o intuito de realizar posterior implementação em bancada para comparar esses resultados com os simulados. Neste capítulo são apresentadas primeiramente na Seção 4.1, a descrição do modelo experimental do sistema PIRR, mostrando a estrutura mecânica, estrutura elétrica e diagrama do circuito elétrico. Na Seção 4.2 é apresentado o fluxograma do algoritmo implementado na IDE Arduino. Na Seção 4.3 é descrito um método para obtenção da estimativa do atrito do pêndulo a ser incluído na equação de movimento. Por fim, na Seção 4.4 é apresentada a abordagem de comunicação adotada para coleta de dados via MATLAB®.

4.1 DESCRIÇÃO DA CONFIGURAÇÃO EXPERIMENTAL DO PIRR

Nesta seção são apresentados os detalhes da estrutura mecânica do sistema PIRR, bem como a estrutura elétrica utilizada. Na Figura 9(a) é mostrado o encoder incremental para medir o ângulo do pêndulo, o motor CC com encoder acoplados à roda de reação, os mancais que seguram o eixo do pêndulo, e o acoplamento flexível. Na Figura 9(b) é apresentada a roda de reação, o pêndulo, e o contrapeso que é utilizado para contrabalancear o peso do motor CC e roda de reação.

Figura 9 – Estrutura mecânica do sistema PIRR.



(a) Vista lateral do PIRR. (b) Vista frontal do PIRR.

Fonte: próprio autor.

O disco, feito de aço inoxidável, apresenta 20,3 cm de diâmetro e 4 mm de espessura. A haste é feita em alumínio, e possui 50 cm de comprimento e 3,8 cm de largura e 3 mm de espessura.

O encoder incremental de 2000 ppr (pulsos por revolução) alocado no eixo de giro do pêndulo opera na faixa de 8 à 24 V com limite de até 3000 rpm. Sua função é efetuar a leitura da posição $\theta(t)$ do pêndulo em cada instante de tempo. O motor já possui um encoder Holzer acoplado em si, que opera na faixa de 3,3 à 5 V em 177 ppr, com a finalidade de realizar a leitura de posição $\phi(t)$ da roda de reação.

Todos os parâmetros físicos da planta estão presentes na Tabela 1.

Tabela 1 – Parâmetros e variáveis da modelagem mecânica.

Símbolo	**Parâmetro**	**Valor**
L	Comprimento da haste (m)	0,50
$L_{hi,i=1,2}$	Comprimento da haste em relação ao pivô (m)	0,25
$L_{chj,j=1,2}$	Distância do pivô ao centro de massa da haste (m)	0,125
L_{h3}	Distância do pivô ao centro de massa da roda (m)	0,2278
L_{h4}	Distância do pivô ao centro de massa do contrapeso (m)	0,2272
M_c	Massa do contrapeso (kg)	0,789
M_h	Massa da haste (kg)	0,160
$M_{hk,k=1,2}$	Massa de meia haste (kg)	0,08
M_m	Massa do motor (kg)	0,390
M_r	Massa da roda (kg)	0,414
M_a	Massa do acoplamento da roda (kg)	0,065
I_r	Momento de inércia da roda (kgm^2)	$21{,}7 \times 10^{-4}$
g	Aceleração da gravidade (m/s^2)	9,81

Fonte: próprio autor.

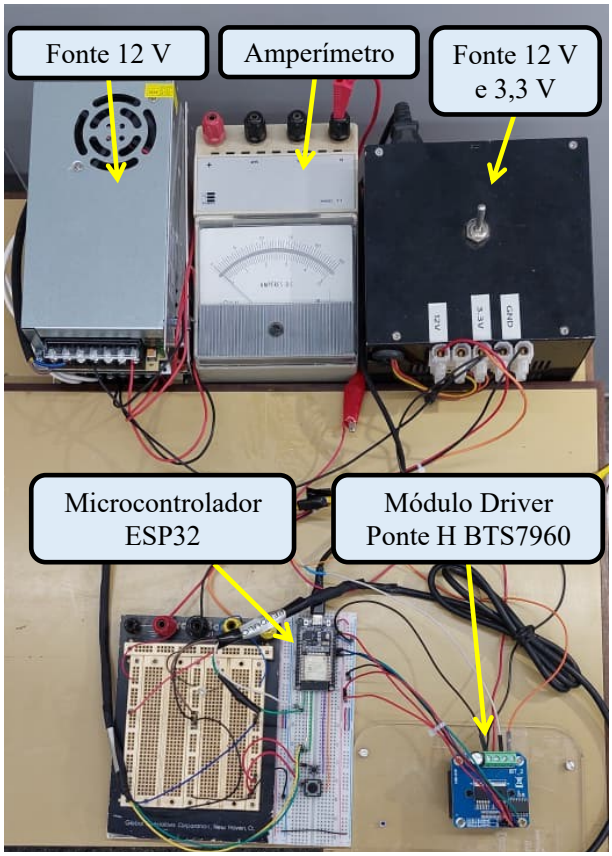
A estrutura elétrica é composta por um microcontrolador para controle e monitoramento das variáveis de estado, uma ponte H para acionamento do motor CC, duas fontes de 12 V para alimentação da ponte H, e alimentação dos sensores, também contém um amperímetro para medir a corrente, conforme ilustra a Figura 10.

De modo a controlar e monitorar as variáveis do sistema, utilizou-se o microcontrolador ESP32 Wroom Devkit. Este dispositivo é responsável também de fornecer o sinal de controle para o driver de acionamento do motor, o qual é uma ponte H BTS7960. A ponte H fornece a tensão de modulação por largura de pulsos (do inglês, *Pulse Width Modulation* - PWM) para o motor CC.

O motor CC do modelo CHP-36GP-555-ABHL de 12 V foi utilizado para gerar o torque da roda de reação. Este consegue atingir 880 rpm em plena carga, sendo capaz de fornecer um torque de 0,29 Nm. Todos os parâmetros e constantes do motor estão dispostos na Tabela 2. Ressalta-se que o modelo dinâmico do motor utilizado neste trabalho

foi previamente obtido em Andreassa (2022), no qual foram realizados os procedimentos de identificação e validação necessários.

Figura 10 – Estrutura elétrica do sistema PIRR.



Fonte: próprio autor.

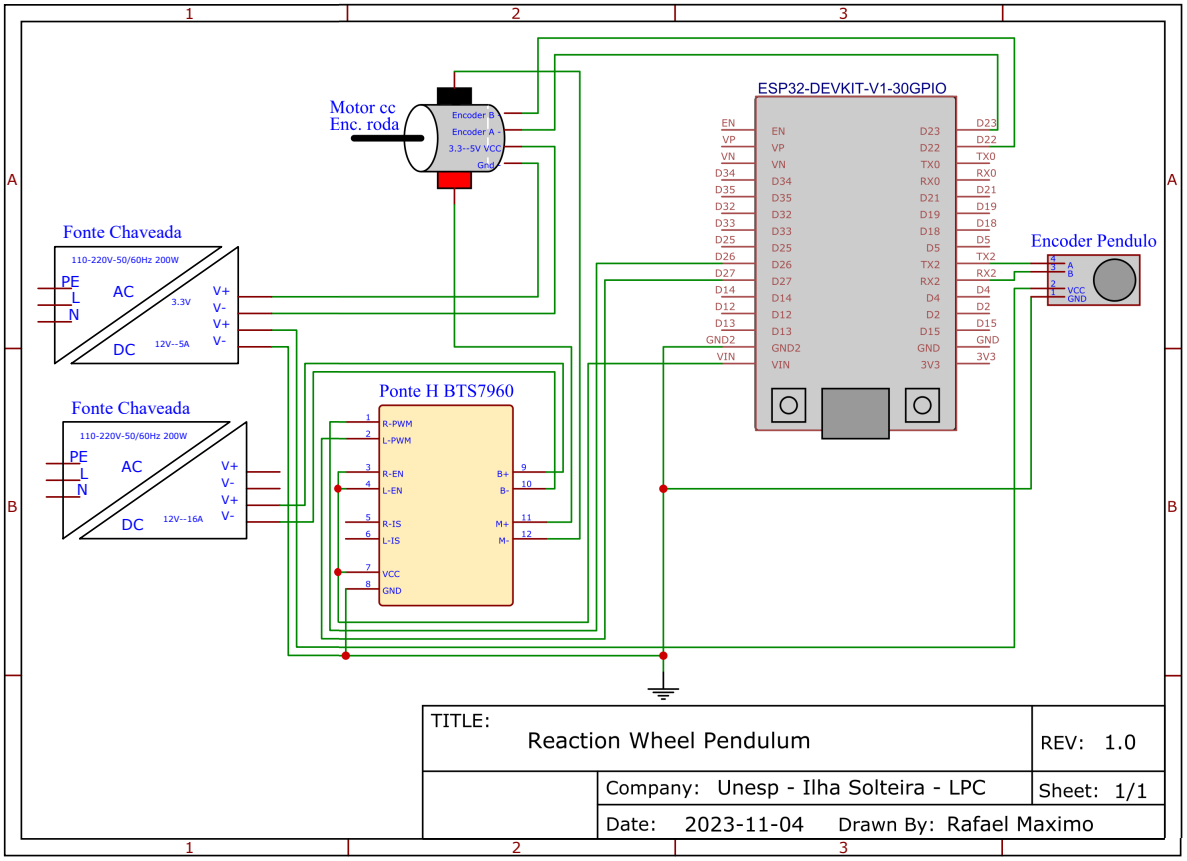
Tabela 2 – Parâmetros do motor CC de imã permanente.

Símbolo	**Parâmetro**	**Valor**
A_m	Coeficiente de atrito do motor $(\frac{Nm}{(rad/s)})$	0,0014
I_a	Corrente de armadura à vazio (A)	$\leq$ 0,28
$\dot{\phi}$	Velocidade à vazio (rpm)	1100
τ_e	Torque elétrico nominal (Nm)	0,1882
$\dot{\phi}$	Velocidade nominal (rpm)	880
I_a	Corrente de armadura nominal (A)	$\leq$ 2,0
τ_{eb}	Torque de rotor bloqueado (Nm)	$\geq$ 7,0
I_{ab}	Corrente de armadura de rotor bloqueado (A)	$\leq$ 9,5
R_a	Resistência de armadura (Ω)	1,6632
K_t	Constante de torque $(\frac{Nm}{A})$	0,0181
K_v	Constante de campo $(\frac{V}{rad/s})$	0,0181
n	Redução do motor	5,2

Fonte: próprio autor.

Na Figura 11 é ilustrado o diagrama do circuito elétrico mostrando as conexões dos componentes. A ponte H BTS7960 é alimentada por uma fonte chaveada de $12\,V$. Outra fonte de alimentação fornece $12\,V$ para o encoder incremental do pêndulo, também $3{,}3\,V$ para o encoder da roda.

Figura 11 – Diagrama do circuito elétrico.



Fonte: próprio autor.

Nas estruturas mecânica e elétrica do sistema PIRR, foram empregados componentes de baixo custo, cujo valor total foi de aproximadamente $R\$804{,}79$, enquanto o custo típico de uma bancada comercial básica é de aproximadamente $R\$55000{,}00$.

Na Tabela 3 são apresentados os custos de alguns dos componentes utilizados na bancada PIRR. Vale ressaltar que os materiais não apresentados na tabela foram reaproveitados ou fabricados no laboratório LPC.

Tabela 3 – Custo dos materiais do PIRR.

Material	**Valor em reais**
Motor CHP-36GP-555-ABHL	191,35
Microcontrolador ESP32	65,00
Fonte de alimentação para PC	100,00
Fonte chaveada $12\,V$	100,00
Codificador rotativo 2000ppr	90,44
Ponte H BTS7960	50,00
Protoboard	25,00
Disco de freio de bicicleta	70,00
Barra chata de alumínio	52,00
Mancal pedestal	61,00
Total	*R*$804,79

Fonte: próprio autor.

4.2 ALGORITMO

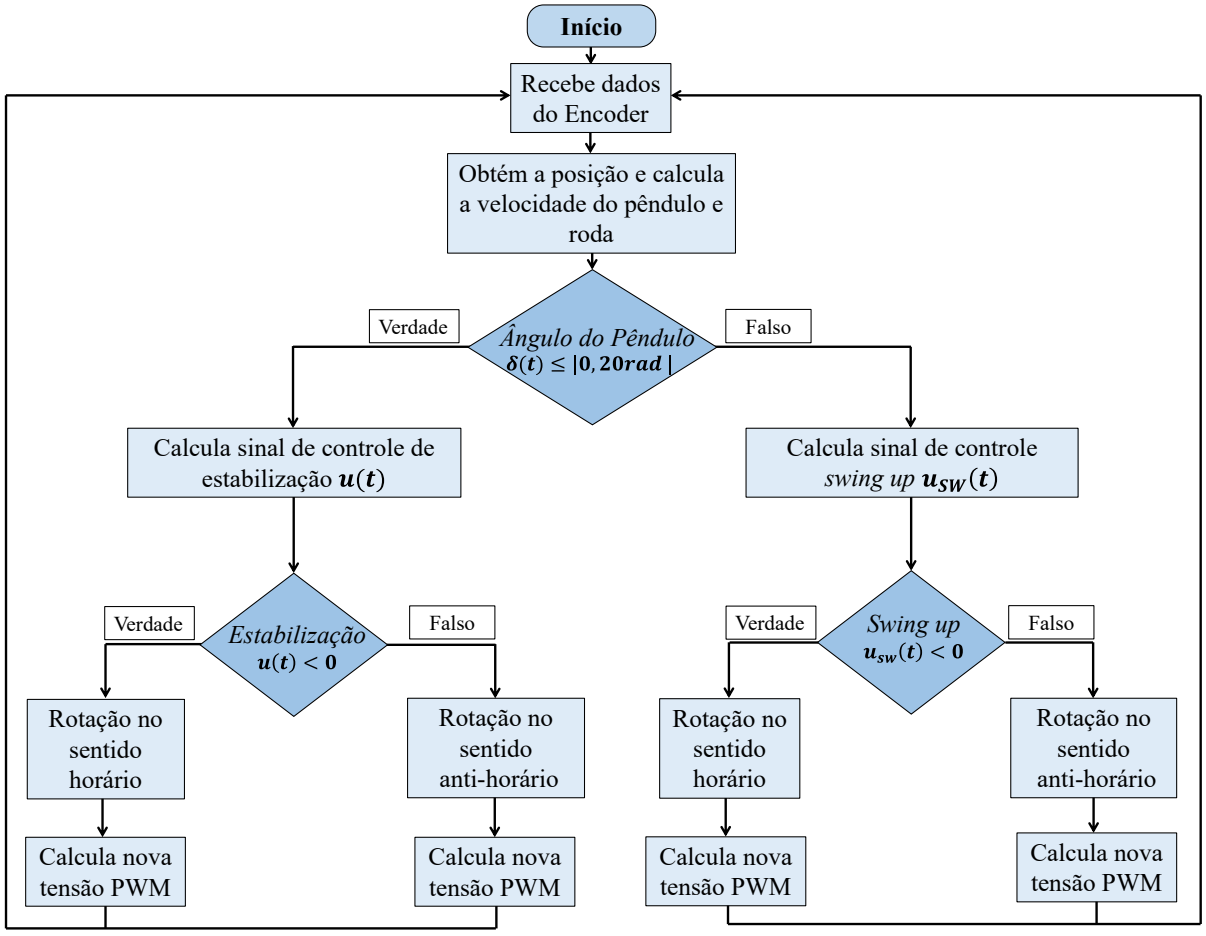
Nesta seção são apresentados o fluxograma do código desenvolvido para implementação das leis de controles projetadas, bem como metodologias consideradas para funcionamento do controle do PIRR.

O ESP32 Wroom Devkit foi escolhido para o controle do sistema, devido a facilidade de uso (pode ser programado na IDE Arduino), baixo custo, e por possuir um processador de $32\,Bits$ (CPU (do inglês, *Central Processing Unit*) *dual core* com frequência de relógio de $160\,MHz$ a $240\,MHz$). Esse processador apresenta capacidade de processamento superior quando comparado aos microcontroladores de $8\,bits$. A linguagem de programação utilizada para programar na IDE Arduino é a linguagem C (Monk, 2016).

Neste trabalho, é considerado um controlador *switch*, sendo o primeiro controlador *swing up* responsável pela elevação do pêndulo até a posição invertida, e o segundo controlador para estabilização do pêndulo no ponto de equilíbrio instável.

Assim, um código foi implementado e após ser escrito foi enviado ao ESP32 para os devidos testes com o sistema PIRR, e posterior coleta de dados com o MATLAB®. Uma simplificação do fluxo lógico do programa é ilustrada no fluxograma da Figura 12.

Figura 12 – Fluxograma do sistema de controle.



Fonte: próprio autor.

Nota-se que no *loop* principal do fluxograma os dados dos sensores são atualizados em cada iteração para os valores da posição do pêndulo $\theta(t)$ e da roda $\phi(t)$. A variável $\phi(t)$ é utilizada para obter a velocidade da roda $\dot{\phi}(t)$ por aproximação, a qual é utilizada no controlador por alocação de autovalores. O tempo de amostragem escolhido para o *timer* de controle foi de $5\,ms$. Vale destacar que foi feito uma alteração relacionada a variável $\theta(t)$, utilizada na análise de dados, para outra variável chamada de $\delta(t)$, implementada no algoritmo. Essa alteração da variável $\theta(t)$ será explicada em capítulos posteriores.

Utilizou-se a biblioteca ESP32Encoder.h a qual permite trabalhar com encoder incremental para contar pulsos, sem precisar lidar diretamente com as interrupções baseadas em *software*. Portanto, para obtenção das posições do pêndulo e roda, considerou-se a função *encoder.getCount()* da biblioteca ESP32Encoder.h.

O encoder no pêndulo fornece uma medição de ângulo com resolução de $2000\,ppr$, isso significa que, a menor mudança no ângulo capaz de ser medida é, $\Delta_\theta = \dfrac{2\pi}{2000} = 0{,}003142\,rad$, ou 0,18°.

As velocidades, foram obtidas pela aproximação a partir das medições de posição,

com o método de diferenciação regressiva (Backward-Euler), com taxa de amostragem de $5\,ms$. No Apêndice B de implementação do código é apresentado nas linhas 247 à 282 a metodologia para obtenção das velocidades.

Segundo Block, Åström e Spong (2007), a oscilação no sinal de velocidade provocada pela amostragem tende a afetar de forma significativa o desempenho do sistema de controle. Por isso, é recomendável aplicar um filtro no sinal de velocidade para atenuar essa oscilação.

Dessa forma, utilizou-se o filtro de primeira ordem, que considera a função de transferência no domínio de Laplace do filtro como:

$$\frac{Y_f(s)}{Y(s)} = \frac{1}{1 + sT_f}, \tag{66}$$

sendo $Y_f(s)$ o sinal de velocidade filtrado, $Y(s)$ o sinal de velocidade calculado, e T_f a constante de tempo de filtragem.

Aplicando-se a transformada inversa de Laplace considerando condições iniciais nulas, obtém-se a equação diferencial:

$$T_f \dot{y}_f(t) + y_f(t) = y(t). \tag{67}$$

Considerando-se que t_k é o instante de amostragem atual (tempo discreto), tem-se:

$$T_f \dot{y}_f(t_k) + y_f(t_k) = y(t_k). \tag{68}$$

Substituindo a derivada no tempo, pelo método de diferenciação regressiva, tem-se:

$$T_f \frac{y_f(t_k) - y_f(t_{k-1})}{T_s} + y_f(t_k) = y(t_k), \tag{69}$$

sendo T_s o tempo de amostragem da velocidade.

Resolvendo para $y(t_k)$, obtém-se:

$$y_f(t_k) = \frac{T_f}{T_f + T_s} y_f(t_{k-1}) + \frac{T_s}{T_f + T_s} y(t_k), \tag{70}$$

que é comumente escrita como:

$$y_f(t_k) = (1 - a) y_f(t_{k-1}) + a\, y(t_k), \tag{71}$$

sendo a o parâmetro do filtro:

$$a = \frac{T_s}{T_f + T_s}. \tag{72}$$

Portanto, com a equação (71) foi possível reduzir oscilações produzidas na velocidade, considerando $T_f = 0{,}1$ e $T_s = 0{,}01$. O valor da constante de tempo de filtragem T_f foi definida por meio de testes em bancada.

Observa-se também que, depois de calculadas as velocidades, o algoritmo deve tomar uma decisão de acordo com a condição apresentada. Assim, o sinal de controle de estabilização é atualizado quando a posição do pêndulo atingir a região de $\delta(t) \leq |0{,}20|\,rad$, ou seja, a variável de estado for $\delta(t) \leq 0{,}20\,rad$, e $\delta(t) \geq -0{,}20\,rad$, caso contrário, é calculado o controle de *swing up*, $u_{sw}(t) = -K\delta_s^n(t)\,\mathrm{sgn}(\dot{\delta}(t)\,\cos(\delta(t))$. O valor de $0{,}20\,rad$ foi selecionado, por meio de ensaios em bancada, para a implementação no algoritmo. Para fins de apresentação e interpretação visual nos gráficos, utilizou-se o valor correspondente em graus, 11,45°.

Logo após, outra estrutura condicional é utilizada para determinar o sentido de giro do motor, que, por consequência, definirá o sentido de rotação da roda de reação. Por fim, dado que o sentido de rotação já foi estabelecido no passo anterior, o sinal PWM é enviado ao pino de saída que aciona o motor para controle de uma direção (horário ou anti-horário). Finalizada essa última etapa, o algoritmo retorna ao início do *loop*.

Para o controle do motor CC, considerou-se o periférico LEDC (do inglês, *LED Control*) que é um módulo físico dentro do ESP32, e utiliza recursos de *hardware* do ESP32 para gerar sinais PWM precisos sem sobrecarregar a CPU. Esse periférico é configurável e permite o uso de timers de alta velocidade ($80\,Mhz$), com resoluções variáveis (até $16\,Bits$) e frequências ajustáveis. A configuração do PWM, foi feita utilizando a função *ledcSetup()*, que permite definir o canal de PWM, a frequência e a resolução do sinal PWM. Dessa forma, foi configurado duas funções *ledcSetup()*, dois canais, com frequência de 19500 Hz, e resolução de 12 $Bits$, ou seja, o valor de PWM varia de 0 à 4095.

4.3 CARACTERIZAÇÃO EXPERIMENTAL DO ATRITO DO PÊNDULO

Fenômenos de atrito se manifestam em todo sistema físico onde os componentes experimentam movimento relativo. Portanto, durante experimentos reais envolvendo movimento oscilatório, duas formas de atrito podem se manifestar (Borsotto *et al.*, 2009). Quando isso ocorre, a oscilação do pêndulo apresenta comportamento não linear (Gogoussis; Donath, 1987; Park; Chwa; Hong, 2006). Dessa forma, torna-se indispensável obter coeficientes de atrito para a simulação (Hu; Wan, 2009).

Para considerar o atrito na simulação, deve-se levar em conta a representação do sistema não linear em espaço de estados. Dessa forma, a seguir será realizada a representação do sistema não linear considerado neste trabalho.

Portanto, utilizando as informações da planta, e considerando a equação (60) com a mudança de variáveis $x_1(t) = \theta(t)$, $x_2(t) = \dot{\theta}(t)$, $x_3(t) = \phi(t)$, $x_4(t) = \dot{\phi}(t)$, e aplicando a derivada, tem-se $\dot{x}_1(t) = \dot{\theta}(t) = x_2(t)$, $\dot{x}_2(t) = \ddot{\theta}(t)$, $\dot{x}_3(t) = \dot{\phi}(t) = x_4(t)$, $\dot{x}_4(t) = \ddot{\phi}(t)$.

Assim o sistema não linear descrito em espaço de estados é:

$$\begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \\ \dot{x_3}(t) \\ \dot{x_4}(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ \frac{1}{I_r I_p - I_r^2}\left(I_r \bar{M}_1\, g\,\mathrm{sen}(x_1(t)) - I_r \bar{M}_2\, g\,\mathrm{sen}(x_1(t)) - I_r \tau(t)\right) \\ x_4(t) \\ \frac{1}{I_r I_p - I_r^2}\left(-I_r \bar{M}_1\, g\,\mathrm{sen}(x_1(t)) + I_r \bar{M}_2\, g\,\mathrm{sen}(x_1(t)) + I_p \tau(t)\right) \end{bmatrix}. \tag{73}$$

Embora o PIRR seja um sistema de dois graus de liberdade (do inglês, *Degrees of Freedom* - DOF) e, portanto, tenha quatro variáveis de estado, muitos autores consideram o problema de estabilização sem preocupar-se com à posição da roda ($x_3(t) = \phi(t)$), reduzindo-se assim o modelo a apenas três estados, sendo desnecessário o cálculo desta variável $x_3(t) = \phi(t)$. Portanto, neste trabalho é considerado o modelo com somente três estados $x_1(t) = \theta(t)$, $x_2(t) = \dot{\theta}(t)$, $x_4(t) = \dot{\phi}(t)$.

O modelo deve considerar as contribuições do atrito de Coulomb e viscoso para ser capaz de representar esse fenômeno. A equação considerando atrito de coloumb F_c e atrito vicoso F_v é descrita conforme apresentado em Kelly e Llamas (1999):

$$F_{fr} = \mathrm{sgn}(\dot{\theta}(t))\,(F_v|\dot{\theta}(t)| + F_c). \tag{74}$$

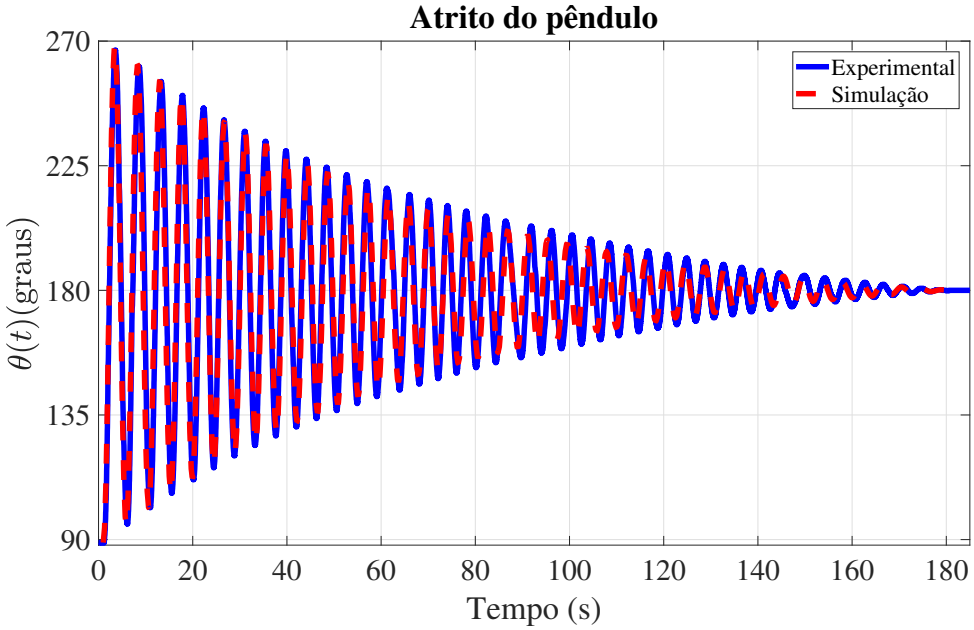
Os parâmetros deste modelo de atrito (74) foram identificados por meio da comparação entre os resultados de implementação em bancada e de simulação, e incluídos na equação do movimento do PIRR apresentado em (58). O resultado experimental com o resultado da simulação da equação de movimento, incluindo o modelo de atrito, é apresentado em:

$$I_p\ddot{\theta}(t) + I_r\ddot{\phi}(t) = \bar{M}_1\, g\,\mathrm{sen}(\theta(t)) - \bar{M}_2\, g\,\mathrm{sen}(\theta(t)) - F_{fr}, \tag{75}$$

sendo $F_v = 0{,}0012\,Nm/rad/s$ e $F_c = 0{,}00041\,Nm$.

Na Figura 13 é apresentada a comparação do resultado experimental em bancada com o resultado de simulação do sistema não linear em malha aberta para a condição inicial $C.I. = [90^\circ\ 0\ 0]^T$ no MATLAB®. O pêndulo na posição vertical para baixo correspondente ao ponto de equilíbrio estável do sistema (180°). Assim, o pêndulo é solto em uma posição de $\theta(t) = 90^\circ$ até atingir seu ponto de equilíbrio estável. O modelo não linear da planta implementado no MATLAB/SIMULINK®, é apresentado no Apêndice A.1.

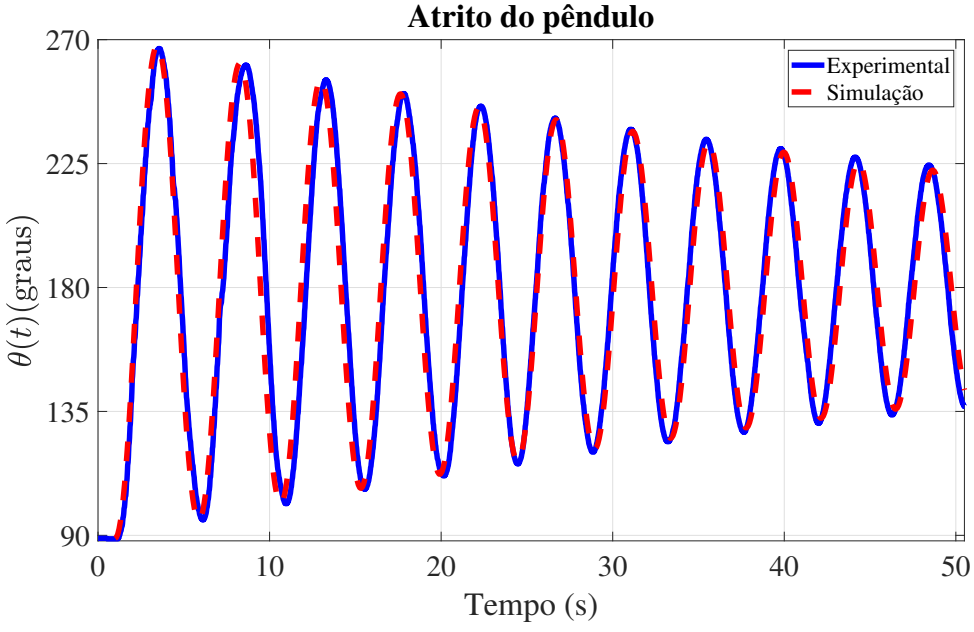
Figura 13 – Estimativa do atrito do pêndulo através da variação do ângulo $\theta(t)$ da haste.



Fonte: próprio autor.

A Figura 14 apresenta um recorte da Figura 13, no intervalo de 0 a 50 s, para melhor comparação das curvas.

Figura 14 – Estimativa do atrito do pêndulo através da variação do ângulo $\theta(t)$ da haste, no intervalo de 0 a 50 s.



Fonte: próprio autor.

A extremidade da haste contribui com uma parcela significativa para o momento de inércia do pêndulo devido à massa dos componentes instalados, especialmente o motor.

Esse aumento do momento de inércia modifica a resposta dinâmica e exige maior esforço de controle, podendo dificultar o desempenho do controlador em condições práticas. Dessa forma, torna-se necessária a utilização de um contrapeso para contrabalancear o sistema e melhorar sua dinâmica.

4.4 COLETA DE DADOS COM O MATLAB®

Existem diversas maneiras de se fazer comunicação entre o microcontrolador ESP32 e o MATLAB®, e os principais tipos são: comunicação serial (USB/UART), comunicação Wi-Fi, comunicação bluetooth, e através do pacote de suporte MATLAB® para *hardware* Arduino (ESP32). Vale destacar que a versão do MATLAB® utilizada neste trabalho é MATLAB® R2017b.

A abordagem de comunicação adotada para coleta de dados foi comunicação serial manual. Dessa forma, a troca de dados entre o MATLAB® e o ESP32 foi realizada por meio da interface serial padrão, permitindo o recebimento de informações.

Foi implementado um sistema simples de aquisição e registro de dados em tempo real, no qual o microcontrolador ESP32 transmite dados provenientes de sensores e sinais de tensão do sistema PIRR. Esses dados são recebidos pelo MATLAB®, exibidos em tempo real por meio de gráficos dinâmicos, e ao término da execução são armazenados em um arquivo no formato CSV para posterior análise.

No Apêndice C é apresentado um exemplo de um código utilizado para coleta de dados do controle *swing up* aplicado no PIRR.

5 PROJETOS DOS CONTROLADORES APLICADOS AO PIRR

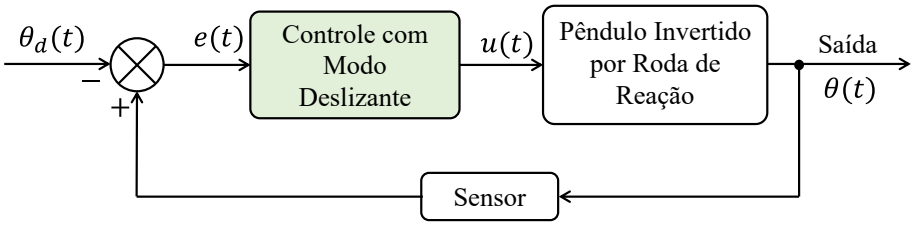
Neste capítulo são apresentados os projetos dos controladores para a estabilização do PIRR no ponto de equilíbrio instável $\theta(t) = 0°$, o projeto do controle *swing up*, e projeto do controlador *switch*. Na Seção 5.1 é apresentado o projeto do controlador CMD convencional, considerando a substituição da função sinal pela função sigmoide para redução da trepidação. Na Seção 5.2 é apresentado o projeto do controle por alocação de autovalores. Na Seção 5.3 é apresentado o projeto do controle PID para estabilizar o PIRR em torno do ponto de operação. Na Seção 5.4 é apresentado o controlador não linear *swing up* para elevação do pêndulo próximo do ponto de equilíbrio instável $\theta_d(t) = 0°$. Na Seção 5.5 é apresentado o projeto do controlador *switch*.

5.1 PROJETO DO CONTROLADOR COM MODOS DESLIZANTES

Nesta seção é apresentado o projeto do controlador com modos deslizante para o PIRR com o intuito de avaliar o desempenho dos resultados de uma abordagem simples de controle robusto, quando comparado com controle PID e alocação de autovalores.

O diagrama de blocos do sistema de controle é mostrado na Figura 15. A referência é um valor de posição desejada ($\theta_d = 0°$) em que o sistema deve alcançar. Assim, o controlador deve ser capaz de fazer com que o sistema atinja o valor referenciado θ_d. A diferença entre a referência e a saída (o valor de realimentação) é um erro que se torna uma entrada para o controlador CMD.

Figura 15 – Diagrama de blocos do sistema de CMD.



Fonte: próprio autor.

Dessa forma, o problema de controle é fazer com que o estado $\theta(t)$ rastreie o estado variante no tempo $\theta_d(t)$. Assumindo que $\theta_d(t)$ é um sinal de posição ideal, então o erro de rastreamento pode ser definido como $e(t) = \theta(t) - \theta_d(t)$.

É possível inserir uma dinâmica desejada para o erro de rastreamento. Uma opção favorável para essa dinâmica compensada é a seguinte equação diferencial homogênea de primeira ordem:

$$\dot{e} + \gamma e = 0, \tag{76}$$

sendo, $\gamma \in \mathbb{R}^+$. Assim, a solução geral de (76) é dada por:

$$e(t) = e(0)\exp^{(-\gamma t)}. \tag{77}$$

Portanto, observa-se que $e(t)$ converge assintoticamente a zero, sem o efeito de qualquer perturbação na dinâmica da equação de erro.

Para alcançar a dinâmica descrita na equação (76), pode-se definir uma função de chaveamento $\sigma(t)$ para a dinâmica de erro e projetar uma lei de controle $u(t)$ que leve essa função a zero em tempo finito. Dessa forma, a solução apresentada em (77), indica que quando $t \to \infty$, a posição e velocidade do erro de rastreamento $e(t)$, convergem para zero exponencialmente com valor γ, e como resultado, $\theta \to \theta_d$.

No projeto do CMD, optou-se por utilizar uma função de chaveamento $\sigma(x)$ em vez da superfície de deslizamento. Assim, a função de chaveamento é definida como

$$\sigma(e, t) = \dot{e} + \gamma e = 0, \tag{78}$$

sendo e o erro que está associado à trajetória desejada do sistema, e γ é um valor constante que satisfaz $\gamma > 0$. A equação (78) representa uma linha reta de deslizamento no espaço de configuração do erro.

A condição para alcance da função de deslizamento, é normalmente estabelecida definindo-se

$$\mathcal{V}(\sigma) = \frac{1}{2}\sigma^2, \tag{79}$$

como a função de Lyapunov, sendo $\mathcal{V}(\sigma)$ positiva definida, e garantindo que sua derivada $\dot{\mathcal{V}}(\sigma)$ seja negativa definida (Slotine; Li, 1991; Shtessel *et al.*, 2014). De acordo com a teoria de Lyapunov, para a lei de alcance ser estabelecida, a seguinte condição de alcance do modo deslizante deve ser satisfeita:

$$\dot{\mathcal{V}}(\sigma) = \sigma\dot{\sigma} < 0 \quad \text{para} \quad \sigma \neq 0. \tag{80}$$

Assim, a lei de alcance exponencial convencional é definida como:

$$\dot{\sigma}(e, t) = -\eta\,\mathrm{sgn}(\sigma(e)) - k\sigma(e), \quad \eta > 0,\, k > 0. \tag{81}$$

Portanto, substituindo (81) na equação (80) tem-se:

$$\begin{aligned} \dot{\mathcal{V}}(\sigma) &= \sigma(-\eta\,\mathrm{sgn}(\sigma) - k\sigma), \\ \dot{\mathcal{V}}(\sigma) &= -\eta\,\sigma\,\mathrm{sgn}(\sigma) - k\sigma^2, \\ \dot{\mathcal{V}}(\sigma) &\leq -\eta|\sigma| - k\sigma^2, \end{aligned} \tag{82}$$

tal que η e k são constantes, com $\eta > 0$ e $k > 0$. Como $\dot{\mathcal{V}}(\sigma)$ é negativa definida, então a dinâmica do sistema irá convergir para a função de deslizamento em tempo finito.

A função de deslizamento já foi definida, portanto, a posição e velocidade do erro de rastreamento podem ser definidas como

$$\begin{cases} e(t) = \theta(t) - \theta_d, \\ \dot{e}(t) = \dot{\theta}(t) - \dot{\theta}_d. \end{cases} \tag{83}$$

Logo, considerando a equação de movimento do PIRR que inclui o atrito obtida em (75), em função de $\ddot{\theta}$, e substituindo em $\dot{\sigma}(e,t) = \ddot{e} + \gamma\dot{e}$, pode-se obter:

$$\begin{aligned} \dot{\sigma}(e,t) &= \ddot{e} + \gamma\dot{e} = [\ddot{\theta}(t) - \ddot{\theta}_d] + \gamma[\dot{\theta}(t) - \dot{\theta}_d], \\ \dot{\sigma}(e,t) &= \frac{\bar{M}_1 g \operatorname{sen}(\theta(t))}{I_p} - \frac{\bar{M}_2 g \operatorname{sen}(\theta(t))}{I_p} - \underbrace{\frac{I_r}{I_p}\ddot{\phi}}_{u(t)} - \frac{F_{fr}}{I_p} - \ddot{\theta}_d + \gamma\dot{e}. \end{aligned} \tag{84}$$

Baseado na lei de alcance definida em (81) e considerando a dinâmica da variável deslizante obtida em (84), o controlador com modos deslizante projetado é:

$$u_{md}(t) = \frac{I_r}{I_p}\ddot{\phi} = \frac{\bar{M}_1 g \operatorname{sen}\theta}{I_p} - \frac{\bar{M}_2 g \operatorname{sen}\theta}{I_p} - \frac{F_{fr}}{I_p} - \ddot{\theta}_d + \gamma\dot{e} + \eta\,\mathrm{sgn}(\sigma(e)) + k\sigma(e). \tag{85}$$

O termo $\mathrm{sgn}(\sigma)$ é definido como:

$$\mathrm{sgn}(\sigma) = \begin{cases} +1 & \text{se } \sigma > 0, \\ 0 & \text{se } \sigma = 0, \\ -1 & \text{se } \sigma < 0. \end{cases} \tag{86}$$

Esse termo de chaveamento $\eta\mathrm{sgn}(\sigma)$ incluído na lei de controle $u_{md}(t)$ pode resultar em *chattering*, elevando o risco de danificar os componentes do sistema. Além disso, do ponto de vista prático, a existência de um modo deslizante é impossível, uma vez que isso demandaria que o controlador realizasse chaveamentos a uma frequência infinita.

Portanto, existe na literatura o método *quasi-sliding mode*, frequentemente utilizado para reduzir o *chattering*, o qual envolve a introdução de uma camada limite ao redor da função de deslizamento. Esse método ajuda a manter o estado dentro de uma faixa especificada nas proximidades da função de chaveamento, e emprega um controle contínuo ou suave dentro da camada limite.

Em substituição à função sinal $\mathrm{sgn}(\sigma)$, é comum utilizar um dos métodos para reduzir a trepidação: a função de saturação $\mathrm{sat}(\sigma)$ ou a função sigmoide $\mathrm{sig}(\sigma)$. Neste trabalho, é considerado a função sigmoide $\mathrm{sig}(\sigma)$:

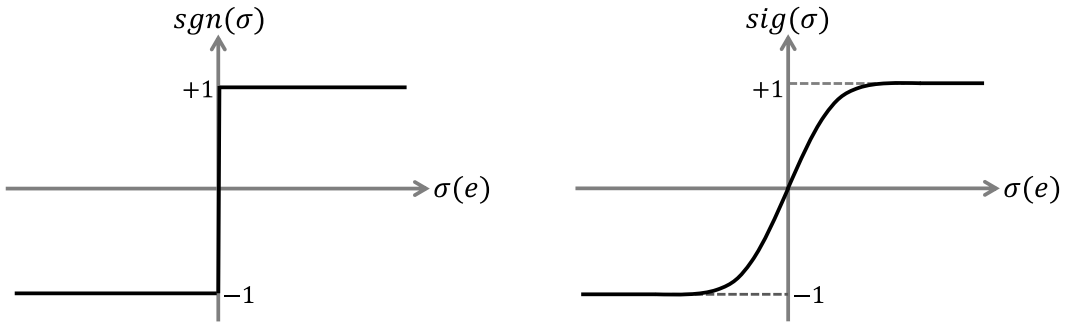
$$\mathrm{sig}(\sigma) = \frac{\sigma}{|\sigma| + \varepsilon} \approx \mathrm{sgn}(\sigma), \tag{87}$$

sendo ε um escalar pequeno e positivo. Pode-se observar que

$$\lim_{\varepsilon \to 0} \frac{\sigma}{|\sigma| + \varepsilon} = \text{sgn}(\sigma). \tag{88}$$

Nos gráficos da Figura 16, são apresentadas a função sinal, Figura 16(a), e função sigmoide, Figura 16(b), observa-se que a função sigmoide possui uma transição suave em torno do zero.

Figura 16 – Aproximação contínua da função sinal com a função sigmoide.



(a) Função sinal. (b) Função sigmoide.

Fonte: próprio autor.

Assim, a lei de controle (85) pode ser reescrita como:

$$u_{md}(t) = \frac{I_r}{I_p}\ddot{\phi} = \frac{\bar{M}_1 g \,\text{sen}\theta}{I_p} - \frac{\bar{M}_2 g \,\text{sen}\theta}{I_p} - \frac{F_{fr}}{I_p} - \ddot{\theta}_d + \gamma\dot{e} + \eta\,\text{sig}(\sigma(e)) + k\sigma(e). \tag{89}$$

A lei de controle baseada no CMD convencional consiste em duas partes: controle equivalente (u_{eq}) e controle chaveado (u_n), que pode ser escrita como:

$$u_{md}(t) = u_{eq}(t) + u_n(t), \tag{90}$$

tal que

$$u_{eq}(t) = \frac{\bar{M}_1 g \,\text{sen}\theta}{I_p} - \frac{\bar{M}_2 g \,\text{sen}\theta}{I_p} - \frac{F_{fr}}{I_p} - \ddot{\theta}_d + \gamma\dot{e}, \tag{91}$$

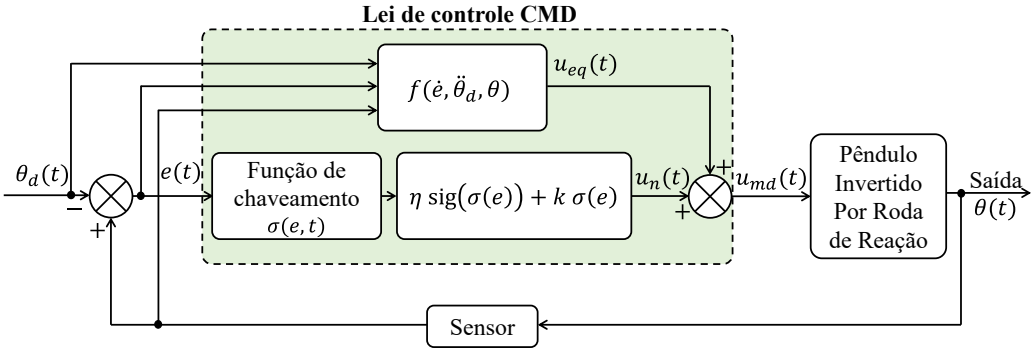
e

$$u_n(t) = \eta\,\text{sig}(\sigma(e)) + k\sigma(e), \tag{92}$$

com $u_n(t)$ representando a componente descontínua, sendo esse responsável a conduzir o estado do sistema para a função de deslizamento; a componente equivalente (u_{eq}) corresponde à componente contínua. Essa componente é fundamental quando o sistema está em condição de deslizamento, pois é definida como a ação de controle necessária para manter um movimento de deslizamento ideal sobre a função de deslizamento, isso significa que $\sigma(e) = \dot{\sigma}(e) = 0$ para todo $t \geq t_r$.

Um diagrama de blocos foi traçado para representar a lei de controle baseada no CMD convencional. O diagrama de blocos é apresentado na Figura 17, em que consiste de controle equivalente e controle chaveado.

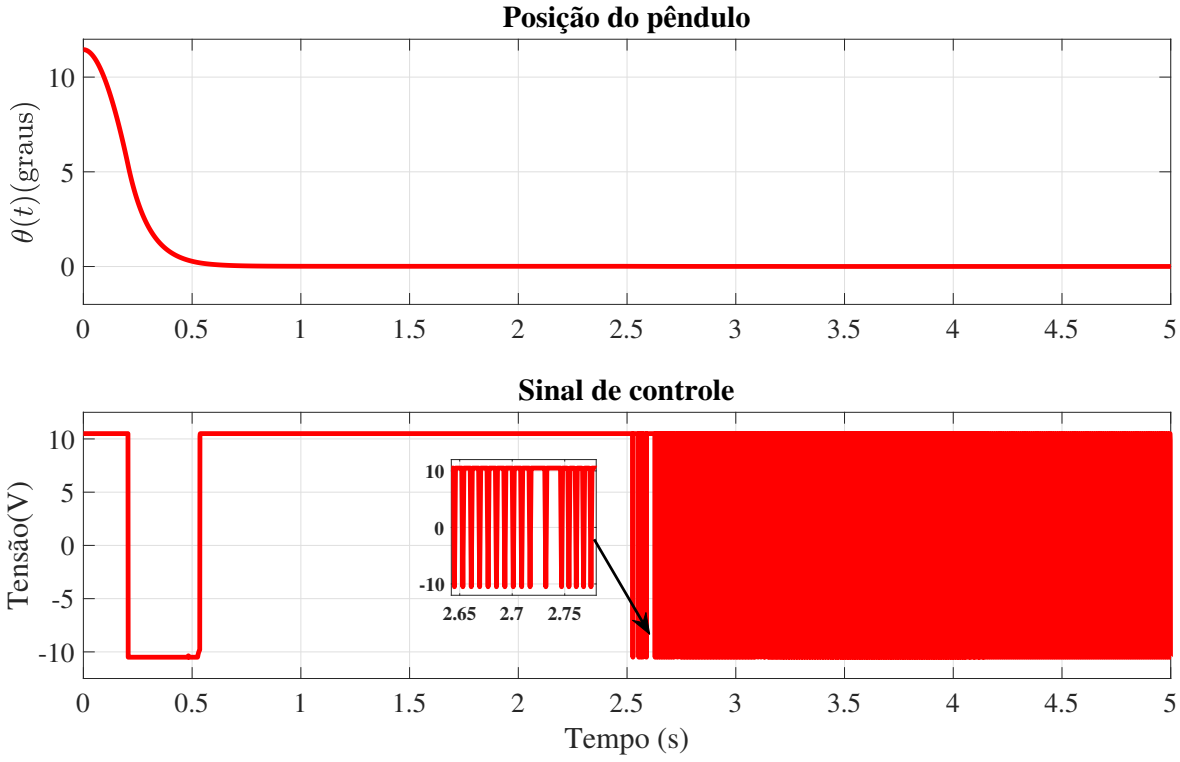
Figura 17 – Diagrama de blocos da lei de controle CMD.



Fonte: próprio autor.

Para exemplificar o problema das altas frequências presente no sinal de controle $u_{md}(t)$ projetado em (85), é apresentado na Figura 18 o resultado de simulação do controle CMD com posição do pêndulo e o sinal de controle apresentando *chattering*. A função descontínua sgn(σ), é a responsável por causar essas oscilações de alta frequência na entrada de controle de tensão $u(t)$, pois essa função descontínua demanda que o controlador realize chaveamentos em frequência infinita.

Figura 18 – Simulação do controle CMD convencional utilizando a função sinal.



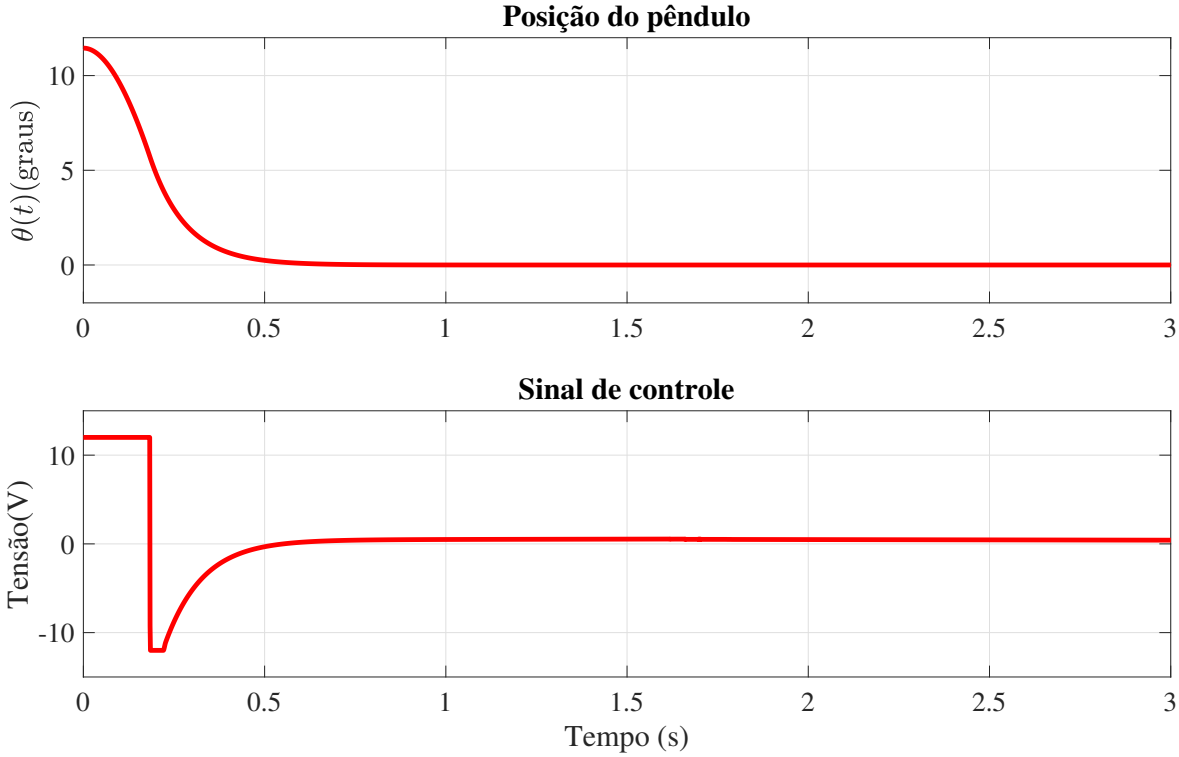
Fonte: próprio autor.

Portanto, para implementação da lei de controle CMD em bancada, deve-se considerar que o chaveamento do dispositivo de controle é limitado.

Como já mencionado, a redução do *chattering* no sinal de controle, pode ser obtida substituindo a função sinal pela função sigmoide. Na Figura 19 é apresentado o resultado de simulação do controlador CMD projetado em (89) que utiliza a função sigmoide, para o controle da posição do pêndulo. Foi feita uma análise da resposta da posição do pêndulo durante a atuação do sinal de controle. A condição inicial do estado controlado $\theta(t)$ é definida como $C.I. = [\theta\ \dot{\theta}\ \dot{\phi}]^T = [11{,}45°\ 0\ 0]^T$.

O valor de condição inicial foi definido através de ensaios em bancada com objetivo de identificar o ângulo máximo de inclinação do pêndulo, a partir do qual o controlador de estabilização foi capaz de estabilizar o pêndulo. A definição desse valor será explicada com mais detalhes em capítulos posteriores.

Figura 19 – Simulação do controle CMD convencional utilizando a função sigmoide.

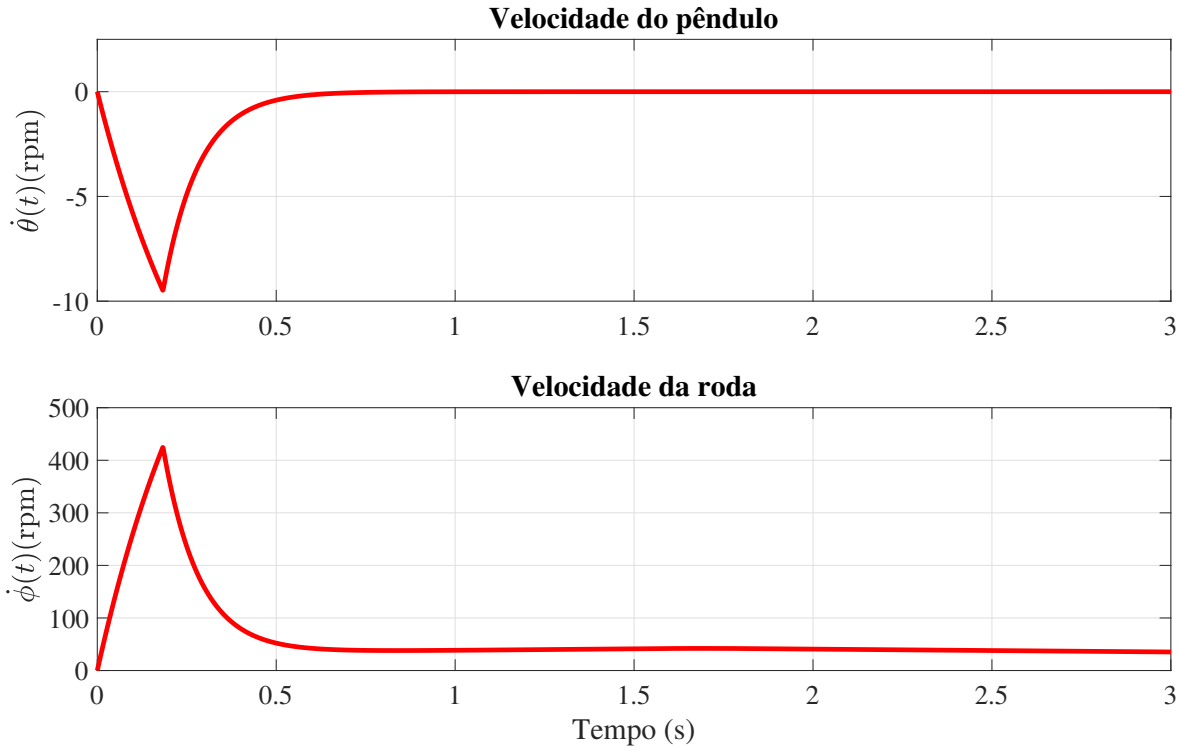


Fonte: próprio autor.

O sinal de controle corresponde à tensão fornecida por PWM ao motor. Para representar adequadamente o comportamento observado em experimentos, foi adicionado um limitador (saturador) que restringe a tensão máxima aplicada ao valor de $12\,V$. As respostas geradas pelos gráficos demonstraram um desempenho satisfatório, permitindo que o pêndulo seje equilibrado na posição invertida. No entanto, nos instantes iniciais o controlador operou em regime de saturação.

As variáveis de velocidade dos estados também foram determinadas. Embora o sinal de controle tenha atingido o limite de saturação, a velocidade não foi afetada por esse problema, conforme ilustrado na Figura 20.

Figura 20 – Simulação do controle CMD com os estados de velocidades.

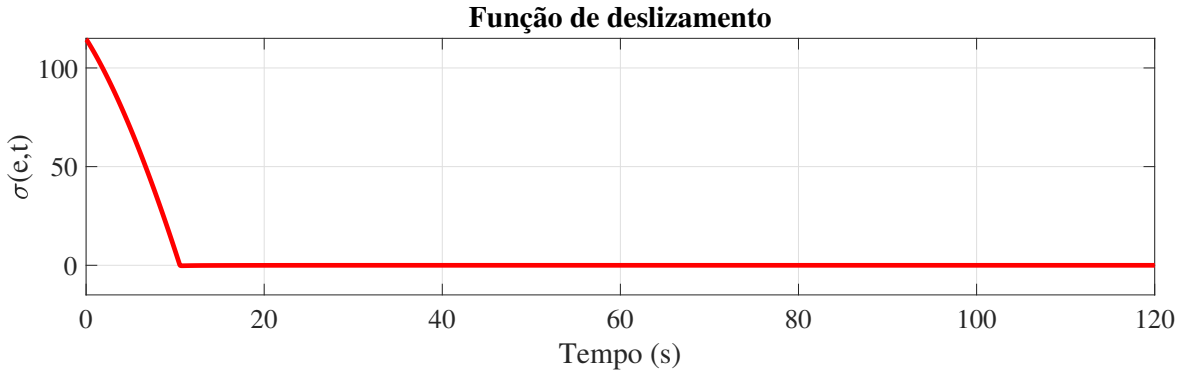


Fonte: próprio autor.

A simulação foi realizada no MATLAB/SIMULINK®, com os seguintes parâmetros: $\gamma = 10$ $\eta = 10$, $k = 2000$ utilizados na equação (85), e $\varepsilon = 0{,}01$ utilizado em (87). A posição desejada $\theta_d(t)$ foi definida como $\theta_d(t) = 0°$.

Na Figura 21 é ilustrado o comportamento da função $\sigma(e,t)$. Pode-se observar que, decorrido alguns segundos a função $\sigma(e,t)$ se iguala a zero, ou seja, um valor zero indica que o sistema alcançou a função de chaveamento definida na equação (78). Caso a função $\sigma(e,t)$ permaneça em zero, esse comportamento indica que o sistema está em deslizamento.

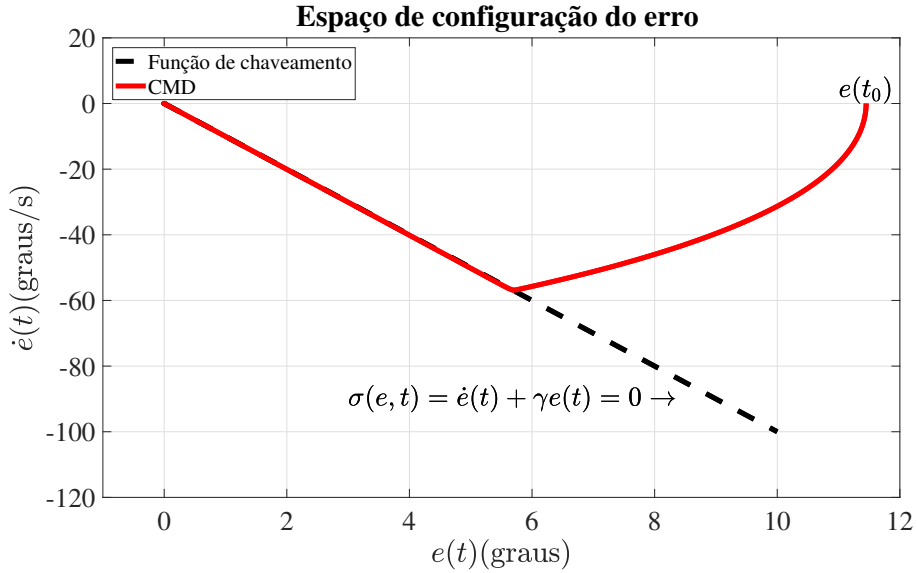
Figura 21 – Simulação da função de deslizamento $\sigma(e,t)$.



Fonte: próprio autor.

Na Figura 22 é apresentado a trajetória dos estados, descrevendo o processo de alcance do estado do sistema na superfície de modo deslizante projetada, bem como o deslizamento para o ponto de equilíbrio.

Figura 22 – Simulação da resposta da trajetória dos estados.



Fonte: próprio autor.

No Apêndice A.3 é apresentado o diagrama do PIRR com controle CMD implementado no SIMULINK®, e também a lei de controle.

5.2 CONTROLE POR ALOCAÇÃO DE AUTOVALORES

Para representar o sistema em espaço de estados, considera-se a equação do modelo linearizado (61) com a mudança de variáveis $x_1(t) = \theta(t)$, $x_2(t) = \dot{\theta}(t)$, $x_3(t) = \phi(t)$, $x_4(t) = \dot{\phi}(t)$. Aplicando a derivada, tem-se $\dot{x}_1(t) = \dot{\theta}(t) = x_2(t)$, $\dot{x}_2(t) = \ddot{\theta}(t)$, $\dot{x}_3(t) = \dot{\phi}(t) = x_4(t)$, $\dot{x}_4(t) = \ddot{\phi}(t)$.

Assim, as variáveis de estados do sistema são medidas a partir dos encoders: a posição do pêndulo, definida por $x_1(t)$, a velocidade do pêndulo $x_2(t)$, e a velocidade da roda de reação $x_4(t)$, como mencionado anteriormente que a posição da roda, pode ser desconsiderado no controle do sistema. Portanto o sistema linearizado no modelo de espaço de estados é:

$$\begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \\ \dot{x_4}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \dfrac{I_r\bar{M}_1\, g}{C_i} - \dfrac{I_r\bar{M}_2\, g}{C_i} & 0 & \dfrac{I_r n^2 K_t K_v}{C_i R_a} + \dfrac{I_r A_m}{C_i} \\ \dfrac{I_r\bar{M}_2\, g}{C_i} - \dfrac{I_r\bar{M}_1\, g}{C_i} & 0 & -\dfrac{I_p n^2 K_t K_v}{C_i R_a} - \dfrac{I_p A_m}{C_i} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -\dfrac{I_r n K_t}{C_i R_a} \\ \dfrac{I_p n K_t}{C_i R_a} \end{bmatrix} u(t),$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_4(t) \end{bmatrix}. \tag{93}$$

A partir das matrizes A, B, C e D do sistema, e dos parâmetros descritos nas Tabelas 1 e 2, obteve-se o sistema na forma de espaço de estados do tipo linear para simulação:

$$\begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \\ \dot{x_4}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 2,0573 & 0 & 0,0754 \\ -2,0573 & 0 & -3,1751 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -0,6347 \\ 26,7129 \end{bmatrix} u(t),$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_4(t) \end{bmatrix}. \tag{94}$$

Da equação (94) que representa a dinâmica do sistema em malha aberta, e utilizando a função $eig(A)$ do MATLAB®, verificou-se que os autovalores em malha aberta são $P_{ma1} = 1{,}4225$, $P_{ma2} = -3{,}1941$, $P_{ma3} = -1{,}4035$. Assim, é comprovada a instabilidade do sistema com a presença de um autovalor real positivo.

Para implementar um controlador por alocação de autovalores, foi verificado primeiramente que o sistema é controlável, portanto, é possível ser feito a alocação através

dos estados. Então foi considerada a lei de controle $u_{ap}(t)$ dada em $Volts$, utilizada para controlar o pêndulo sobre o ponto de equilíbrio instável, é descrita como:

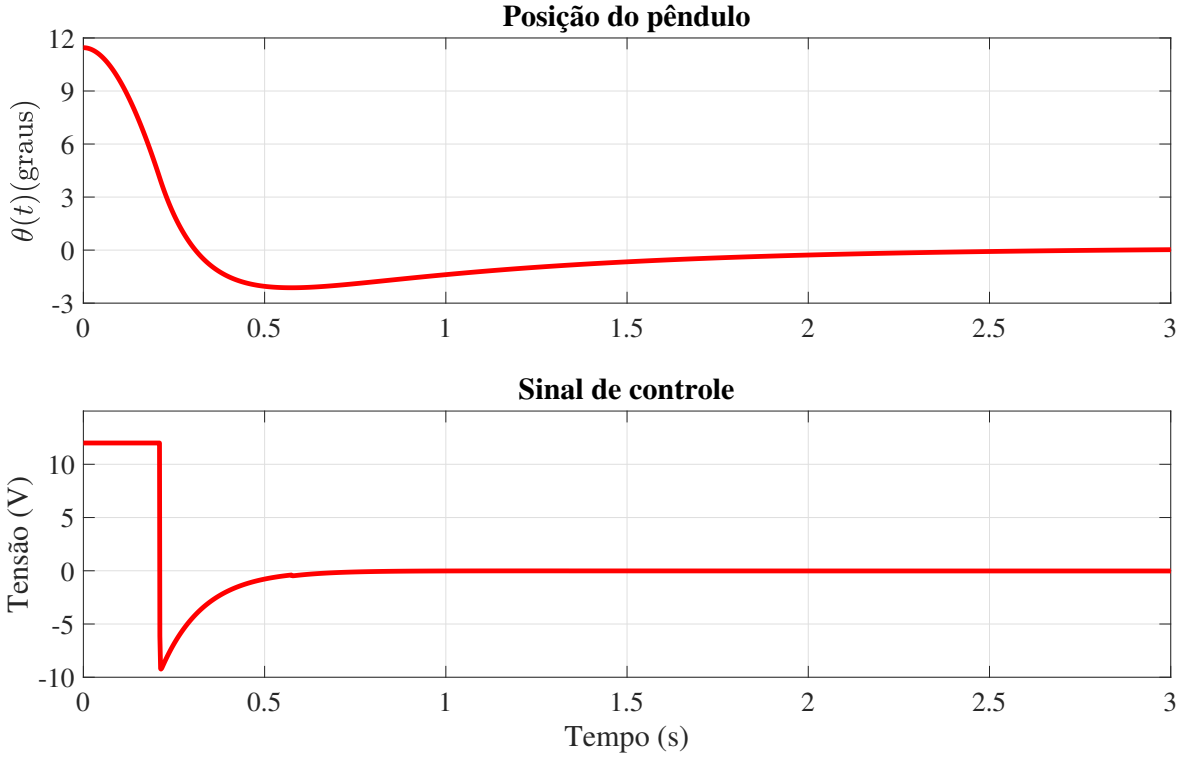
$$u_{ap}(t) = \quad -Kx(t), \tag{95}$$

sendo $K \in \mathbb{R}^{1\times 3}$.

Desta forma, calculou-se a matriz de ganhos K do controlador $u_{ap} = -Kx$, para os autovalores desejados em malha fechada sendo $P_{mf} = [-2210{,}37\ -8{,}67\ -1{,}301]$ com o objetivo de estabilizar o estado posição do pêndulo $\theta(t)$ no ponto de interesse. A posição destes autovalores foi definida com o objetivo de se obter uma resposta para o sistema que é de terceira ordem, tal que a resposta se aproxime de um sistema de primeira ordem. Assim foi calculado, através do MATLAB®, com a função $(acker(A, B, P_{mf}))$ a matriz de ganhos K do controlador por alocação de autovalores $K = 10^4\,.\,[-3{,}4744\ -2{,}3056\ -0{,}0465]$.

Foi realizada a simulação do modelo não linear, considerando a lei de controle (95) para uma condição inicial de $C.I. = [11{,}45°\ 0\ 0]^T$ para analisar a posição do pêndulo durante a atuação do sinal de controle, conforme Figura 23 a qual mostra o controle de posição, e o sinal de controle.

Figura 23 – Simulação do controle de estabilização com posição e sinal de controle.



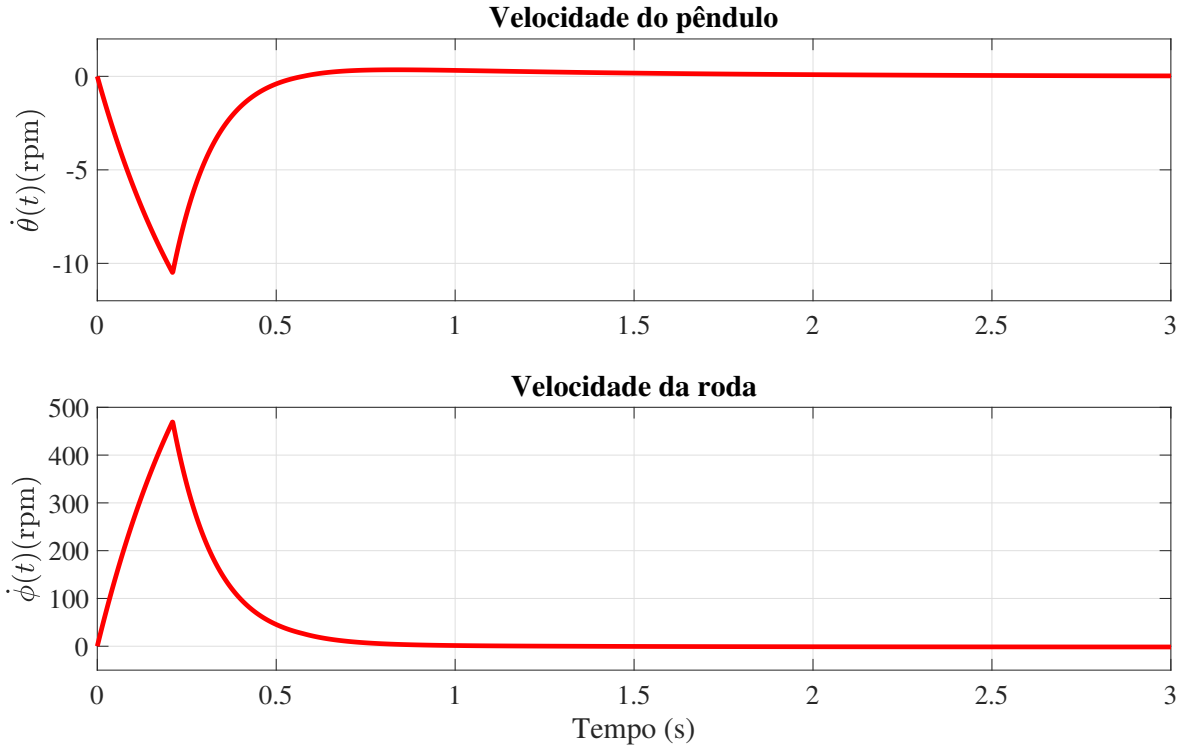
Fonte: próprio autor.

O sinal de controle representa a tensão de alimentação por PWM do motor. Para simular o comportamento experimental, um saturador foi implementado para manter a

relação de tensão de alimentação máxima em $12\,V$. Os gráficos apresentaram uma boa resposta, possibilitando a estabilização do pêndulo na posição invertida, contudo, nos primeiros instantes de tempo o controlador atingiu a saturação.

Na Figura 24 são apresentadas as velocidades do pêndulo $\dot{\theta}(t)$ e da roda $\dot{\phi}(t)$. Apesar do sinal de controle apresentar saturação, as velocidades não apresentaram saturação, dada a limitação mecânica do motor.

Figura 24 – Simulação do controle de estabilização com os estados de velocidades.



Fonte: próprio autor.

O modelo linearizado, a verificação dos autovalores em malha aberta, a matriz de controlabilidade, bem como o projeto do controlador por alocação de autovalores, é apresentado no código do Apêndice A.2.

5.3 CONTROLE PID

Em termos gerais, como o sistema PIRR é de entrada única, o sistema linearizado apresentado na equação (94), pode resultar em três funções de transferência, referente a entrada de tensão $u(t)$ e a cada variável de estado $\theta(t), \dot{\theta}(t)$ e $\dot{\phi}(t)$.

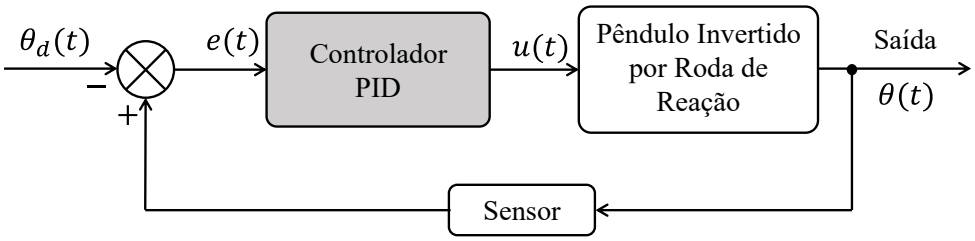
Dessa forma, uma vez que a variável de interesse é $\theta(t)$, então é considerado a função de transferência do ângulo do pêndulo pela entrada de tensão $\frac{\Theta(s)}{U(s)}$:

$$\frac{\Theta(s)}{U(s)} = \frac{-0{,}6347\,s}{s^3 + 3{,}175\,s^2 - 2{,}057\,s - 6{,}377}\ , \tag{96}$$

sendo a equação (96) obtida a partir da aplicação da transformada de Laplace na equação (94), supondo condições iniciais nulas.

Na Figura 25, é apresentado o diagrama de blocos do sistema PIRR com o controlador PID. A referência $\theta_d(t)$ corresponde ao valor desejado que o sistema deve atingir. Portanto, o controlador $u(t)$ tem a função de conduzir o sistema até esse valor de referência. A diferença entre a referência $\theta_d(t)$ e a saída $\theta(t)$ é denominada erro $e(t)$, o qual é utilizado como entrada para o controlador.

Figura 25 – Diagrama de blocos do sistema PIRR com controle PID.



Fonte: próprio autor.

Foi simulado o controlador PID para o PIRR com o propósito de realizar uma análise comparativa de desempenho em relação ao controlador implementado experimentalmente. A simulação do controlador PID foi realizada considerando o bloco de controlador PID na biblioteca do SIMULINK®, e a função de transferência do sistema apresentada em (96), operando em malha fechada. Com esse método, foi possível visualizar a relação entre os parâmetros de ajuste do controlador.

A função de transferência do controlador PID padrão, é geralmente escrita na forma paralela (97), ou na forma ideal (98).

$$G(s) = K_p + K_i\,\frac{1}{s} + K_d\,s, \tag{97}$$

$$G(s) = K_p\,(1 + \frac{1}{T_i\,s} + T_d\,s), \tag{98}$$

sendo K_p o ganho proporcional, K_i o ganho integral, K_d o ganho derivativo, T_i a constante de tempo integral, e T_d a constante de tempo derivativa.

Foi realizada a simulação do controle em malha fechada com o modelo não linear da planta PIRR, considerando os ganhos sintonizados através de experimentos realizados em bancada.

Os ganhos do controlador PID utilizados nas simulações e nas implementações em tempo real estão apresentados na Tabela 4. Esses ganhos foram obtidos experimentalmente por meio de ajuste fino no sistema PIRR, sendo selecionado o conjunto que apresentou melhor desempenho, conforme apresentado na Tabela 4.

Tabela 4 – Ganhos do controlador PID.

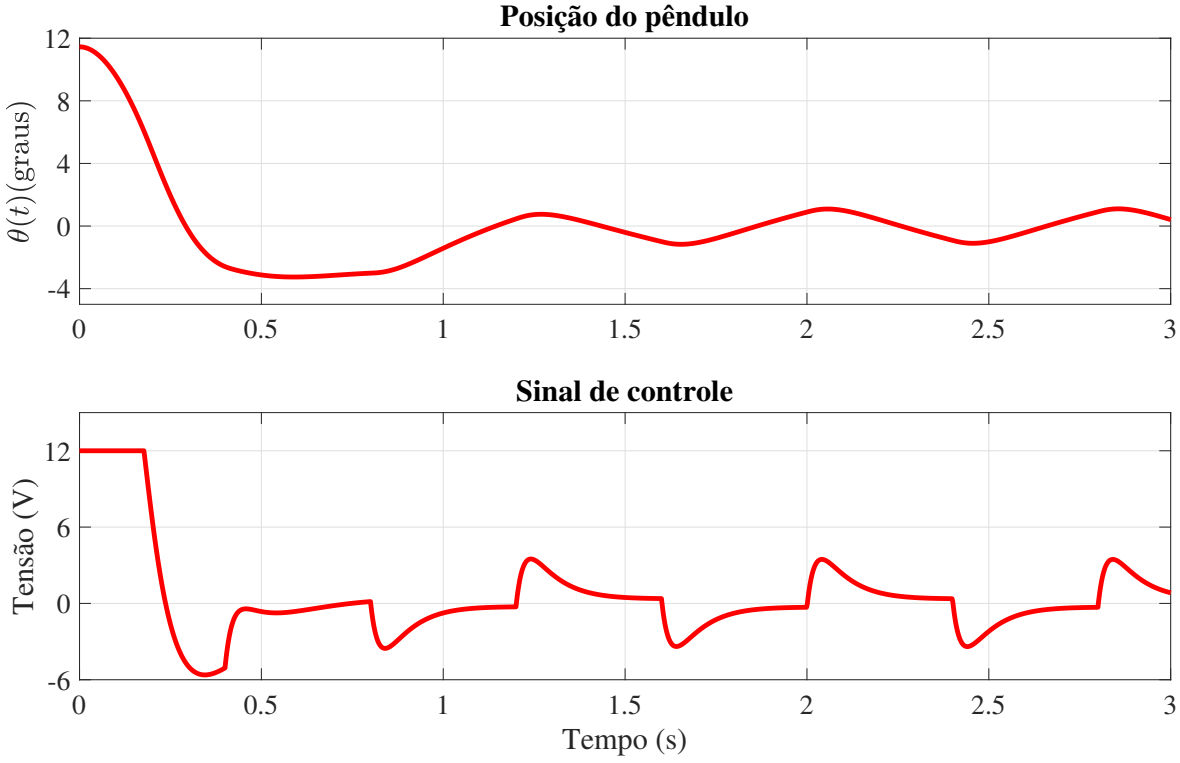
K_p	K_i	K_d
200	786	5

Fonte: próprio autor.

Considerando que o período de amostragem empregado nos experimentos em tempo real é de $5\,ms$ e que o sistema em estudo apresenta dinâmica relativamente lenta, esse intervalo é suficientemente pequeno para que o controlador implementado de forma discreta apresente, na prática, um comportamento equivalente ao de um controlador em tempo contínuo.

Na Figura 26 é apresentado o controle de posição angular e sinal de controle, para uma condição inicial de $C.I. = [11{,}45^\circ\ 0\ 0]^T$, levando em consideração um *setpoint* com onda triangular.

Figura 26 – Simulação do controle PID com posição e sinal de controle.

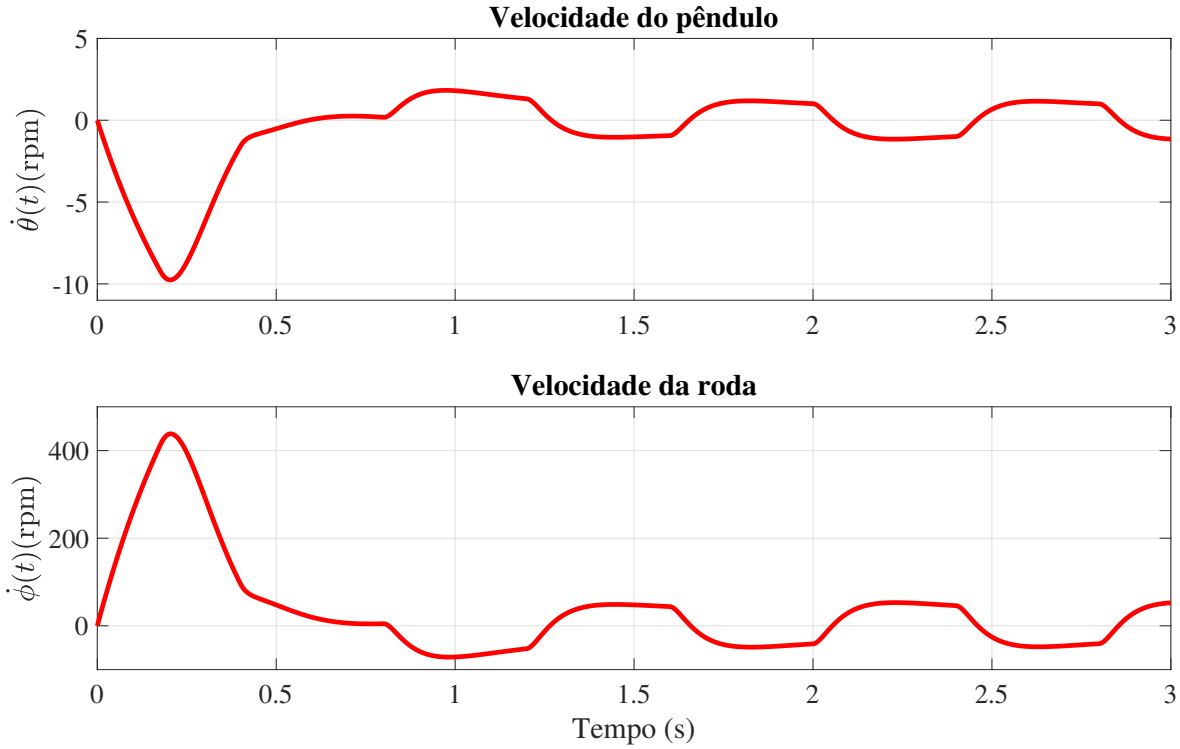


Fonte: próprio autor.

O funcionamento do controlador PID projetado inicia-se com a comparação entre a referência desejada $\theta_d(t)$ e a posição angular atual do pêndulo $\theta(t)$. Com base nessa diferença, o controlador calcula e gera um sinal de controle de tensão por PWM $u(t)$, o qual é aplicado ao motor CC. Um saturador foi implementado para manter a relação de tensão de alimentação máxima em $12\,V$. Observa-se que o controle de estabilização do pêndulo para referência $\theta_d(t) = 0°$ foi obtido, entretanto o controlador atingiu a saturação nos instantes iniciais.

Na Figura 27 são apresentadas as velocidades do pêndulo $\dot{\theta}(t)$ e da roda $\dot{\phi}(t)$, com o objetivo de verificar o comportamento simulado do sistema PIRR.

Figura 27 – Simulação do controle PID com os estados de velocidades.



Fonte: próprio autor.

Este resultado comprova que o controle de posição do pêndulo é consequente da variação do momento angular através da ação da roda de reação instalada na extremidade do pêndulo. Desta forma, o equilíbrio do pêndulo na posição invertida é obtido por meio da aceleração e desaceleração do motor que está acoplado à roda de reação.

No Apêndice A.4 é apresentado o diagrama da planta com controle PID implementado no SIMULINK®.

5.4 CONTROLE *SWING UP*

Um dos principais problemas encontrado no sistema do PIRR é elevar o pêndulo para a região de estabilidade utilizando pouca energia. A força necessária do motor para impulsionar o pêndulo somente com o controle de estabilização, é muito alta, o que torna um processo inviável. Contudo, com a adição de um controle específico como o *swing up*, o pêndulo consegue atingir a região invertida utilizando a própria energia acumulada do movimento oscilatório para proporcionar o movimento do pêndulo.

Dessa forma, o método *swing up* tem o propósito de proporcionar a energia necessária para que o pêndulo alcance a vizinhança da posição de equilíbrio instável, inspirado nos métodos de controle de energia (Åström; Furuta, 1996; Xin; Kaneda, 2002).

$$u_{sw}(t) = \quad -K_{sw}\theta^n(t)\,\mathrm{sgn}(\dot{\theta}(t)\cos(\theta(t))), \tag{99}$$

sendo o ganho $K_{sw} = 18{,}5$.

O termo $\theta^n(t)$ tem um comportamento semelhante de uma função de energia desejada, em que para valores de ângulos próximos à 0° (ponto de equilíbrio instável) sua atuação é baixa, enquanto para ângulos maiores há um aumento. O sinal de controle será positivo enquanto o termo $\dot{\theta}(t)\cos(\theta(t))$ for negativo ou vice-versa. Assim, em conjunto com o termo $\theta^n(t)$, a velocidade do pêndulo é aumentada gradualmente, até alcançar a região próxima a posição desejada (0°).

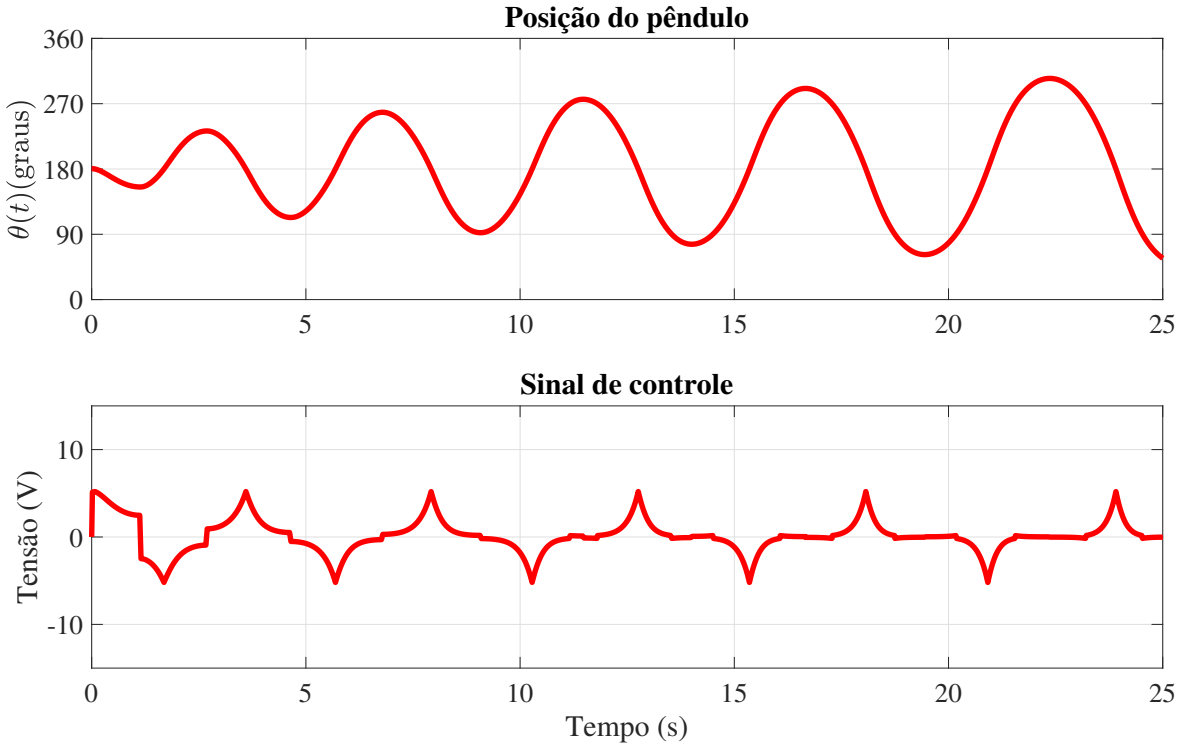
Para que o pêndulo movimente de forma semelhante tanto na região maior que π e menor que π, é necessário criar outra variável para o *swing up*. Como se trata de uma posição que opera em 0° à 360°, isto resulta em diferentes atuações do controlador nas regiões especificadas, o que faz com que o pêndulo tenha uma tendência maior de estabilizar em apenas uma região. Por esse motivo a variável $\delta_s(t)$ é obtida com objetivo de substituir o termo $\theta^n(t)$:

$$\delta_s(t) = \mathrm{Mod}\left(\mathrm{Resto}\left(\frac{(\theta+\pi)}{2\pi}\right) - \pi\right), \tag{100}$$

sendo a operação (Resto) definida como o resto da divisão de $(\theta + \pi)$ por 2π. Em seguida é subtraído π para deslocar o intervalo para $[-\pi, \pi]$. Por fim, é aplicado o módulo desse resultado para garantir que o pêndulo oscile de forma proporcional em ambas as regiões.

Na Figura 28 é apresentado o resultado de simulação do controle *swing up* para elevação do pêndulo.

Figura 28 – Simulação do controle *swing up*.



Fonte: próprio autor.

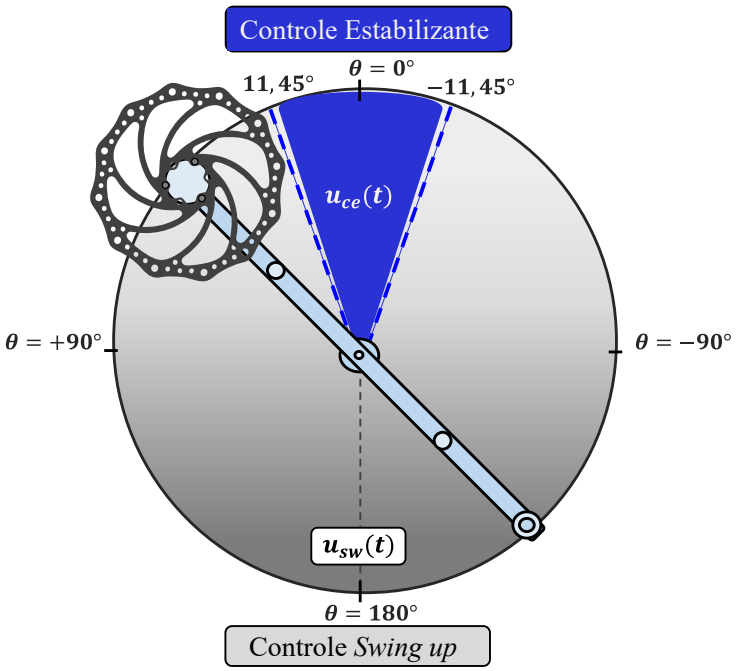
Observa-se que na simulação a oscilação do pêndulo é proporcional, com uma forma de onda suave atuando igualmente em ambos os sentidos, possibilitando que o pêndulo ao se aproximar de 0° (ou 360°), o pêndulo se aproxima da região em que entra em ação o controle de estabilização.

5.5 ESTRATÉGIA DE CONTROLE SWITCH

Nesta seção é descrito o método de controle *switch* considerado no projeto de controle do sistema PIRR. Este método de controle utiliza dois controladores: um para elevar o pêndulo próximo à região invertida (ponto de equilíbrio instável) e outro para o controle estabilizante.

Para os controladores alternarem em seus respectivos pontos de operação, é necessário que uma estratégia de controle seja definida. Na Figura 29 é ilustrado um método de controle *switch* considerado neste trabalho.

Figura 29 – Área de atuação dos controladores selecionados por região.



Fonte: próprio autor.

O problema de controle *swing up* consiste em transferir o pêndulo do ponto de equilíbrio estável inferior para o ponto de equilíbrio instável superior. O controle *swing up* considera as oscilações em torno da posição inferior, e aplica forças adequadas ao sistema de modo a acumular energia suficiente para levar o pêndulo até a vizinhança do ponto de equilíbrio invertido. Durante esse processo, o pêndulo realiza oscilações crescentes em sentido horário ou anti-horário até atingir uma condição favorável para a aplicação de um controlador de estabilização.

É importante mencionar que a variável de estado relacionada à posição do pêndulo, implementada no algoritmo de controle de estabilização $\delta(t)$, é diferente da utilizada para análise de dados $\theta(t)$ conforme discutido na Seção 5.4. As posições angulares do pêndulo, por estarem associadas a um círculo trigonométrico, podem gerar ambiguidades na leitura com o encoder, por exemplo, como identificar se a posição 360° obtida através do encoder, é a mesma de 0°, já que o ponto de equilíbrio considerado é 0°. Isso faz com que o controlador não atue nessa região e dificulta o seu funcionamento em conjunto com o controlador *swing up*.

Dessa forma, para manter o ângulo calculado $\theta(t)$ dentro de um intervalo finito e consistente, foi utilizada uma função de normalização baseada em módulo. O ângulo arbitrário $\theta(t) \in \mathbb{R}$, obtido a partir do encoder incremental, é transformado para o intervalo $[-\pi\,, \pi]$ radianos pela seguinte relação:

$$\delta(t) = \text{Resto}\left(\frac{(\theta + \pi)}{2\pi}\right) - \pi, \tag{101}$$

sendo a operação (Resto), definida como o resto da divisão inteira de um número real por 2π, e este resultado do resto garante que $\delta(t)$ fique entre $[0, 2\pi]$. Por fim, subtrai-se π para deslocar o intervalo entre $[-\pi, \pi]$.

Assim, ambos lados tendem para 0°, ou seja, independentemente do número de voltas completas do encoder, positivas ou negativas, o ângulo $\delta(t)$ permanece limitado no intervalo $[-\pi, \pi]$, garantido o ajuste circular necessário para a aplicação de controle do sistema PIRR.

Dado que o controle de estabilização utiliza a variável de estado $\delta(t)$, então no momento que o delta está localizado entre os ângulos $\theta_{ce} \leq |11{,}45°|$, é acionado o controlador de estabilização $u_{ce}(t)$, e para ângulos localizados externamente a esta faixa o controlador $u_{sw}(t)$ é acionado. Portanto, a estratégia de controle projetada é apresentada por:

$$u(t) = \begin{cases} u_{ce}(t) & \text{se } 11{,}45° \geq \delta(t) \geq -11{,}45°, \\ u_{sw}(t) & \text{caso contrário.} \end{cases} \tag{102}$$

Desse modo, são implementadas as ferramentas necessárias para permitir a estabilização do pêndulo em torno da posição de equilíbrio instável ($\theta(t) = 0°$), com condições iniciais próximas ao ponto de equilíbrio estável ($\theta(t) = 180°$).

6 RESULTADOS EXPERIMENTAIS DOS CONTROLADORES

Neste capítulo são apresentados e discutidos resultados experimentais obtidos para os controladores projetados. Na Seção 6.1 são apresentados os resultados do controle *swing up* para levantar o pêndulo até a região invertida. Na Seção 6.2 são apresentados os resultados do controle *switch* aplicado no sistema PIRR. Na Seção 6.3 são exibidos os resultados de implementação para os controladores lineares. Na Seção 6.4 são apresentados os resultados para o controlador CMD. Na Seção 6.5 são mostrados e comparados os resultados dos controladores lineares através de índices de desempenho. As estratégias de controle projetadas foram implementadas em aplicação experimental com o objetivo de avaliar seu desempenho. Essas leis de controle foram programadas em *script* desenvolvido na IDE Arduino utilizando o microcontrolador ESP32.

6.1 RESULTADO DO CONTROLADOR *SWING UP*

Nesta seção é apresentado o resultado experimental do controle *swing up*. Na região de *swing up* a condição inicial do pêndulo foi estabelecida em 180°. O comportamento do sistema PIRR, com o controlador não linear projetado, foi verificado por meio de simulação utilizando o MATLAB/SIMULINK®, cujo resultado é apresentado na Seção 5.4.

6.1.1 Resultado Experimental do Controlador *Swing Up*

A utilização do controle *swing up* em sistema de pêndulo invertido é necessário, porque o torque requerido do motor para impulsionar o pêndulo somente com controle de estabilização, é muito alto, e isso pode influenciar no custo do motor. Visto que nesta tese foi requisitado para utilizar somente dispositivos e componentes de baixo custo para o controle do sistema PIRR, foi preciso projetar o controlador *swing up*.

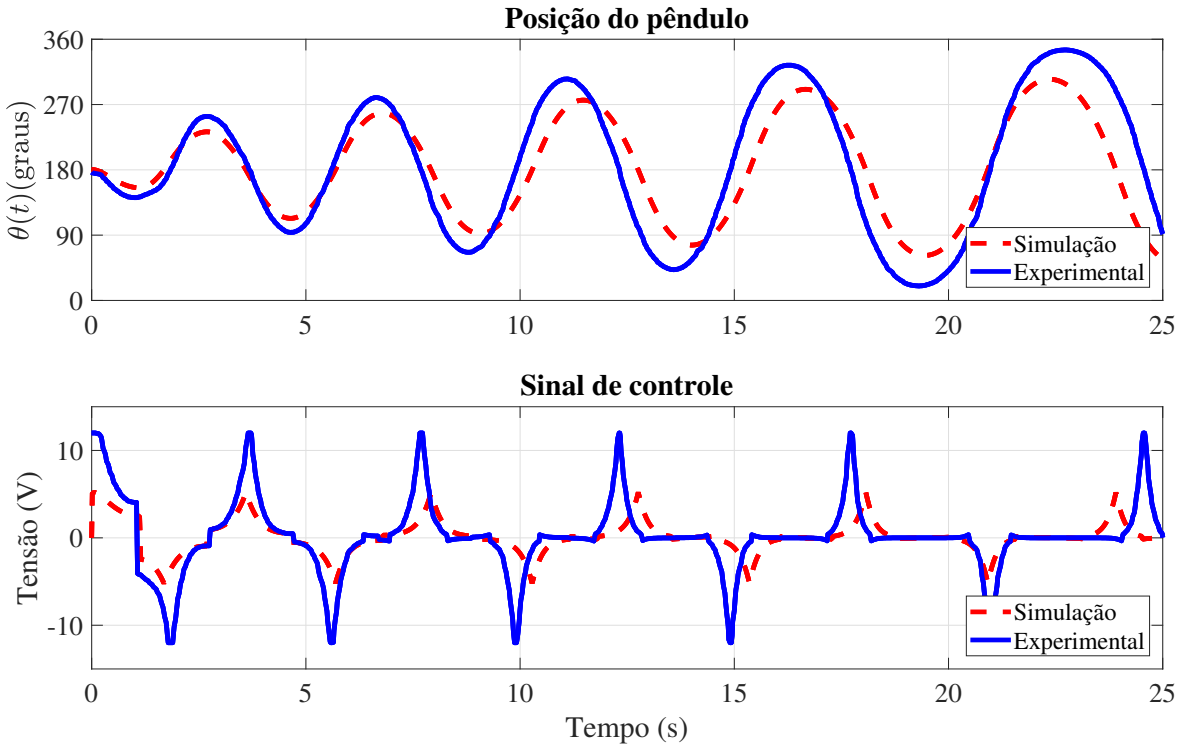
Inicialmente foram realizados testes com o sistema PIRR, a fim de determinar o ganho K_{sw} adequado para o pêndulo atingir a região invertida utilizando a própria energia acumulada do movimento oscilatório. Esse ganho foi determinado por meio de ensaios em bancada, e o ganho que apresentou o melhor desempenho para oscilar o pêndulo foi $K_{sw} = 18{,}5$.

Foi verificado nos ensaios em bancada que o pêndulo tende a estabilizar em apenas uma região, pois a faixa especificada para o movimento do pêndulo é de 0° à 360°. Desse modo, foi considerado uma outra variável para leitura da posição $\theta(t)$, na qual é aplicado o módulo nessa leitura, para garantir que o pêndulo oscile de forma proporcional em ambas

as regiões, conforme equação (100).

Na Figura 30 é mostrado o controle de posição e sinal de controle para o controlador *swing up*. Pode-se verificar que o pêndulo começa o movimento com condição inicial próxima de 180°, e a posição é aumentada gradualmente até alcançar a região próxima do ponto de operação (0° ou 360°).

Figura 30 – Resultado experimental do controle *Swing Up* com posição e sinal de controle.



Fonte: próprio autor.

6.2 RESULTADOS DO CONTROLE *SWITCH* DO SISTEMA

Nesta seção são apresentados os resultados experimentais para os controladores projetados considerando a estratégia de controle *switch*. O objetivo é conduzir o pêndulo situado inicialmente em $\theta(t) = 180°$, até próximo à região invertida com o controlador *swing up* e, em seguida, controlá-lo no ponto de equilíbrio instável $\theta(t) = 0°$ com controlador de estabilização.

6.2.1 Resultados do Controlador por Alocação de Autovalores e *Swing Up*

Nesta subseção são apresentados os resultados experimentais obtidos para o controlador *swing up* com controlador por alocação de autovalores, em que se considera o

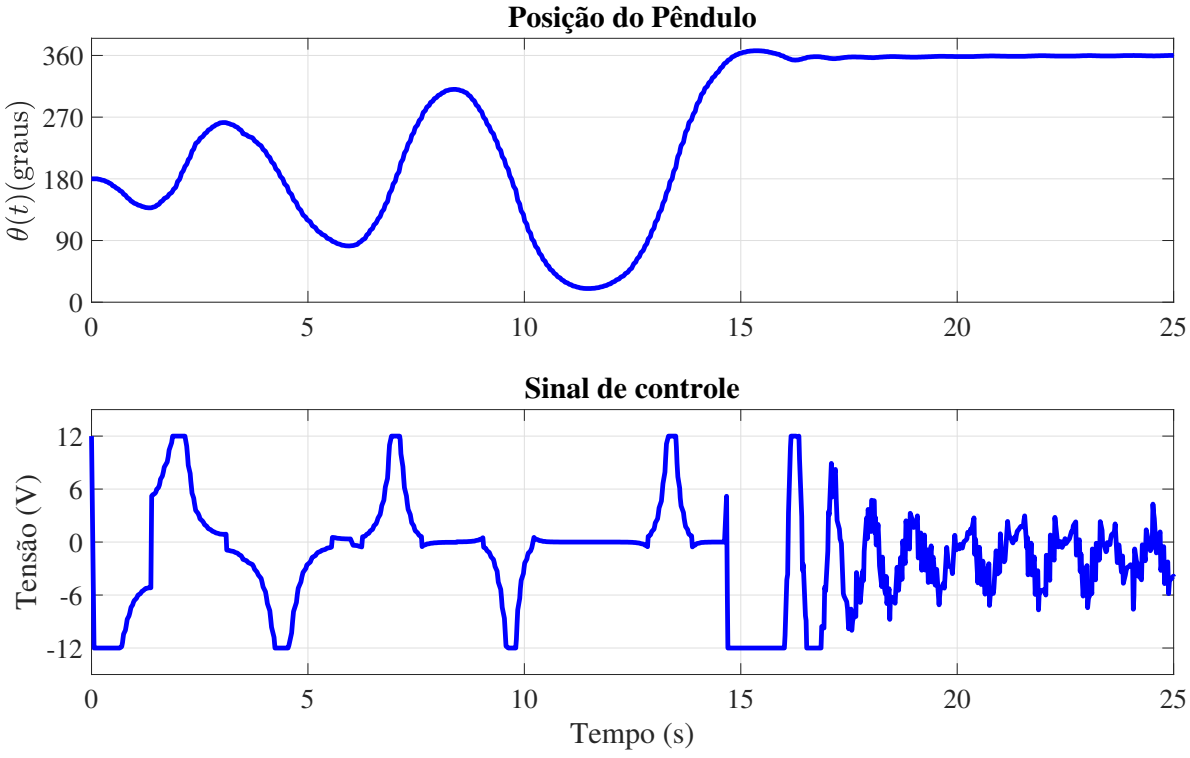
método de controle *switch* proposto na equação (102).

A condição de comutação entre os controladores de estabilização e de elevação foi determinada de acordo com o ângulo de inclinação do pêndulo. Nesse critério deve-se garantir que o pêndulo não acumule energia cinética excessiva ao cruzar o ângulo de captura. Dessa forma, foram realizados ensaios em bancada para identificar o ângulo máximo de inclinação do pêndulo, a partir do qual o controlador de estabilização foi capaz de estabilizar o pêndulo mesmo sem velocidade inicial.

Assim, as regiões de operação dos controladores desenvolvidos considera a condição de que se o módulo de $\theta(t)$ for menor ou igual a 11,45°, é ativado o controle de estabilização, caso contrário, é ativado o controle *swing up*.

Na Figura 31 são apresentadas as respostas de posição do pêndulo, e sinal de controle para o controlador *swing up*, bem como para o controlador por alocação de autovalores.

Figura 31 – Resultado experimental do controle *swing up* e por alocação de autovalores com posição e sinal de controle.



Fonte: próprio autor.

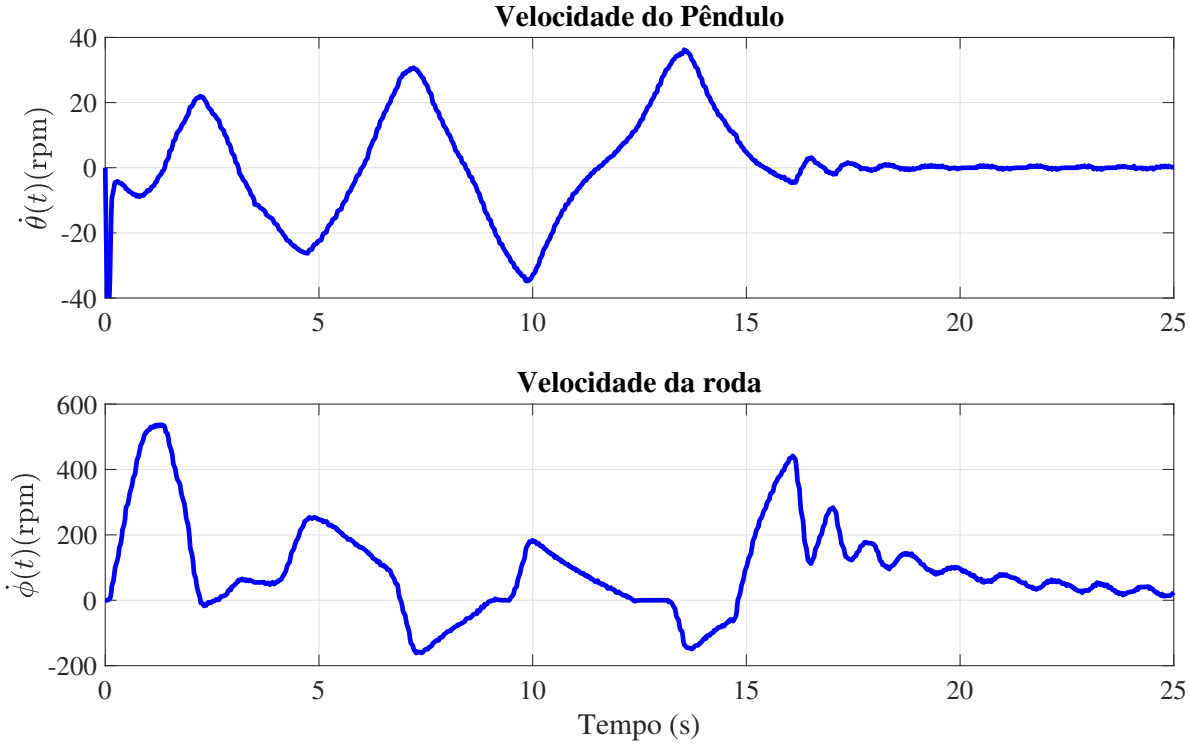
Observa-se que nos instantes iniciais o controle *swing up* começa atuar e com condição inicial próximo de 180°, elevando o pêndulo até próximo à região invertida, ou região de estabilização. Logo, o método de controle *switch* troca o controlador *swing up* pelo controlador por alocação de autovalores em $t = 14{,}5$ segundos. Dessa forma, o

controle foi capaz de alcançar a posição invertida com apenas quatro oscilações do controle *swing up*.

O método de controle *switch* determina qual controle é ativado. Nota-se que o sinal de tensão do controle *swing up* atua até $t = 14{,}5$ segundos. Após esse instante de tempo, o controlador de estabilização é acionado por meio da comutação do controle *switch*, que passa a aplicar o sinal de tensão proveniente do controle por alocação de autovalores.

Na Figura 32 são apresentadas as respostas das velocidades do pêndulo e da roda. Observa-se que a velocidade do pêndulo $\dot{\theta}(t)$ possui velocidade próxima de zero em regime permanente. A velocidade da roda reação $\dot{\phi}(t)$, atua nos instantes iniciais com velocidade máxima próxima de $550\,rpm$. Para o controle estabilizante, a velocidade da roda $\dot{\phi}(t)$ decresce, contudo $\dot{\phi}(t)$ não zera devido ao torque de distúrbio resultante do cabo de conexão do motor.

Figura 32 – Resultado experimental do controle *swing up* e por alocação de autovalores com os estados de velocidades do pêndulo e da roda.



Fonte: próprio autor.

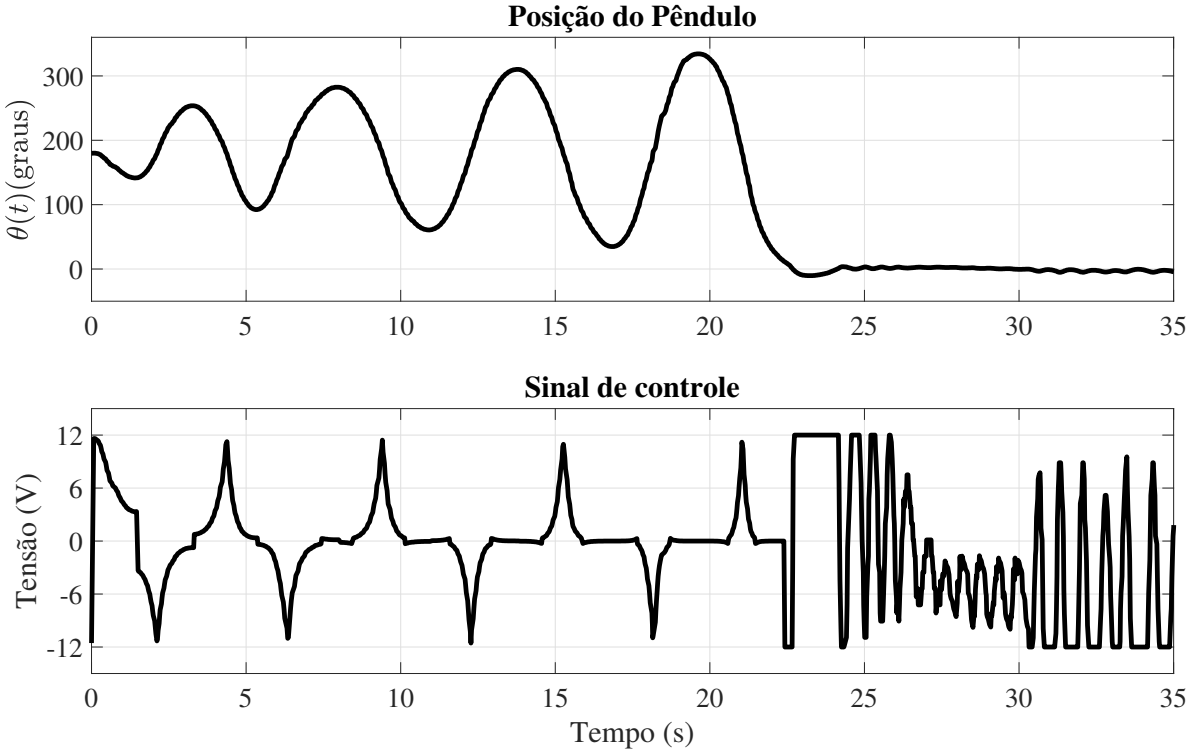
No Apêndice B.1 é apresentado o código que considera a condicional para o controle do PIRR em ambas as regiões.

6.2.2 Resultados do Controlador PID e *Swing Up*

Nesta subseção são apresentados os resultados experimentais obtidos do controlador *swing up* juntamente com o PID.

Na Figura 33 são apresentadas as respostas de posição do pêndulo e o sinal de controle correspondentes aos controladores *swing up* e PID, evidenciando o comportamento dinâmico obtido em cada estratégia de controle.

Figura 33 – Resultado experimental do controle *swing up* e PID com posição e sinal de controle.



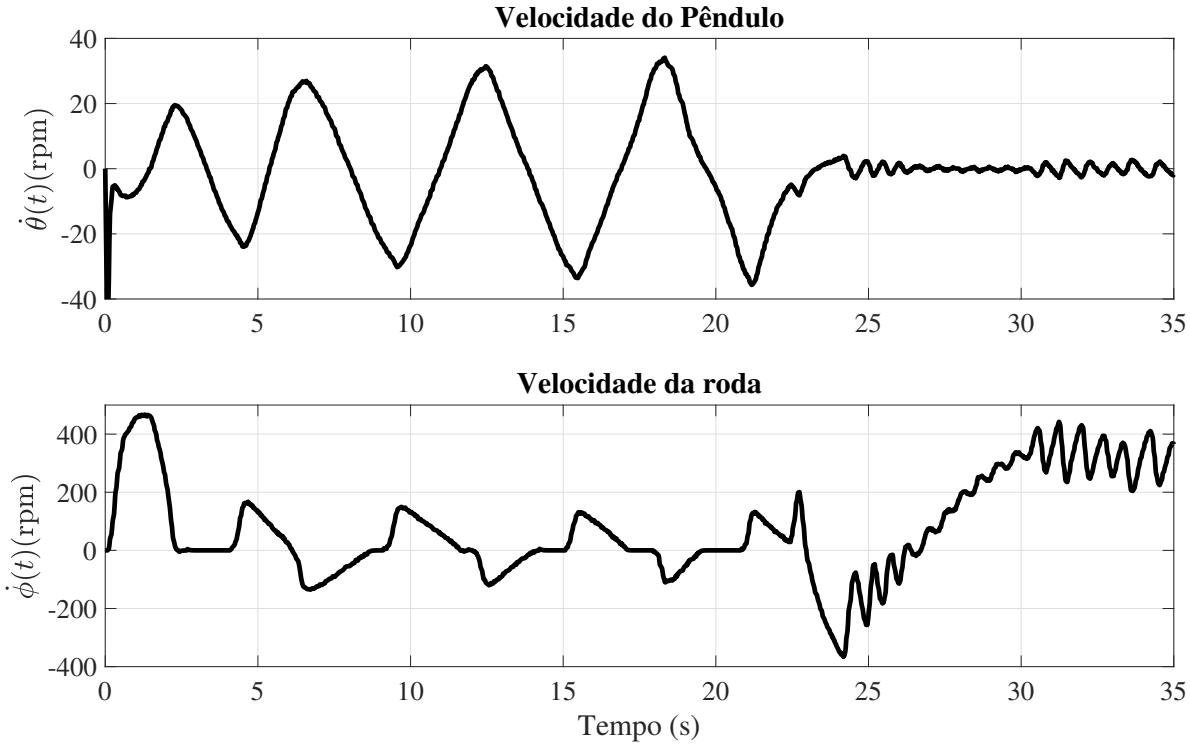
Fonte: próprio autor.

Observa-se que nos instantes iniciais, o controlador *swing up* entra em ação a partir de uma condição inicial próxima de 180°, conduzindo o pêndulo até a região próxima do ponto de operação, também denominada região de estabilização. Após 23 segundos, o controle *switch* troca o controlador *swing up* pelo controlador PID. Portanto, o controle *swing up* foi capaz de alcançar a posição invertida com apenas cinco oscilações do sistema PIRR.

Nota-se que o sinal de tensão gerado pelo controlador *swing up* permanece ativo até cerca de $t = 23$ segundos. Após esse período, o controlador PID é acionado por meio da comutação do método *switch*, que passa a aplicar o sinal de tensão proveniente do controle PID.

Na Figura 34 são apresentadas as respostas das velocidades do pêndulo e da roda para o controle PID. Observa-se que a velocidade do pêndulo aproxima-se de zero em regime permanente. Já a velocidade da roda de reação atinge valores máximos próximos $450\,rpm$ nos instantes iniciais. Durante a atuação do controle de estabilização, a velocidade da roda fica oscilando mas não satura.

Figura 34 – Resultado experimental do controle *swing up* e PID com os estados de velocidades do pêndulo e da roda.



Fonte: próprio autor.

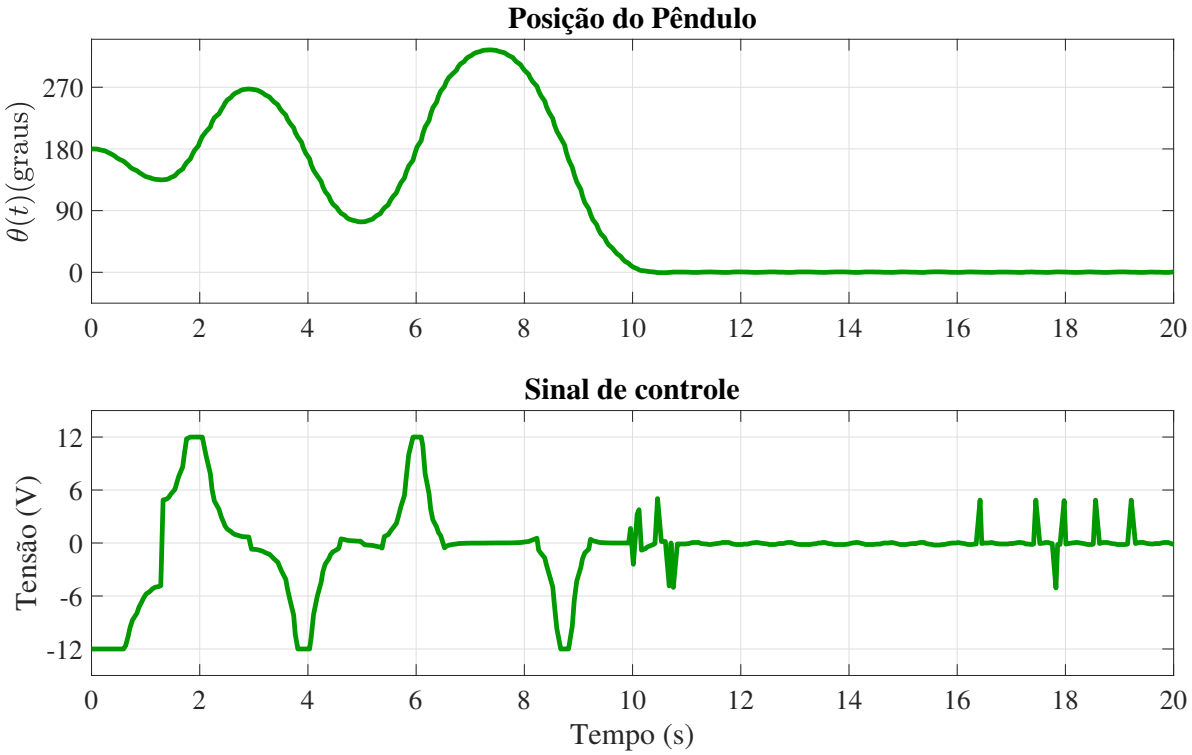
No Apêndice B.2 é apresentado o código que considera a condicional para o controle do PIRR com PID e *swing up* em ambas as regiões.

6.2.3 Resultados do Controlador CMD e *Swing Up*

Nesta subseção são apresentados os resultados experimentais obtidos para o controlador *swing up* com controlador CMD.

Na Figura 35 são apresentadas as respostas de posição do pêndulo, bem como o sinal de controle associado aos controladores *swing up* e CMD, evidenciando o comportamento dinâmico resultante de cada estratégia de controle.

Figura 35 – Resultado experimental do controle *swing up* e CMD com posição e sinal de controle.



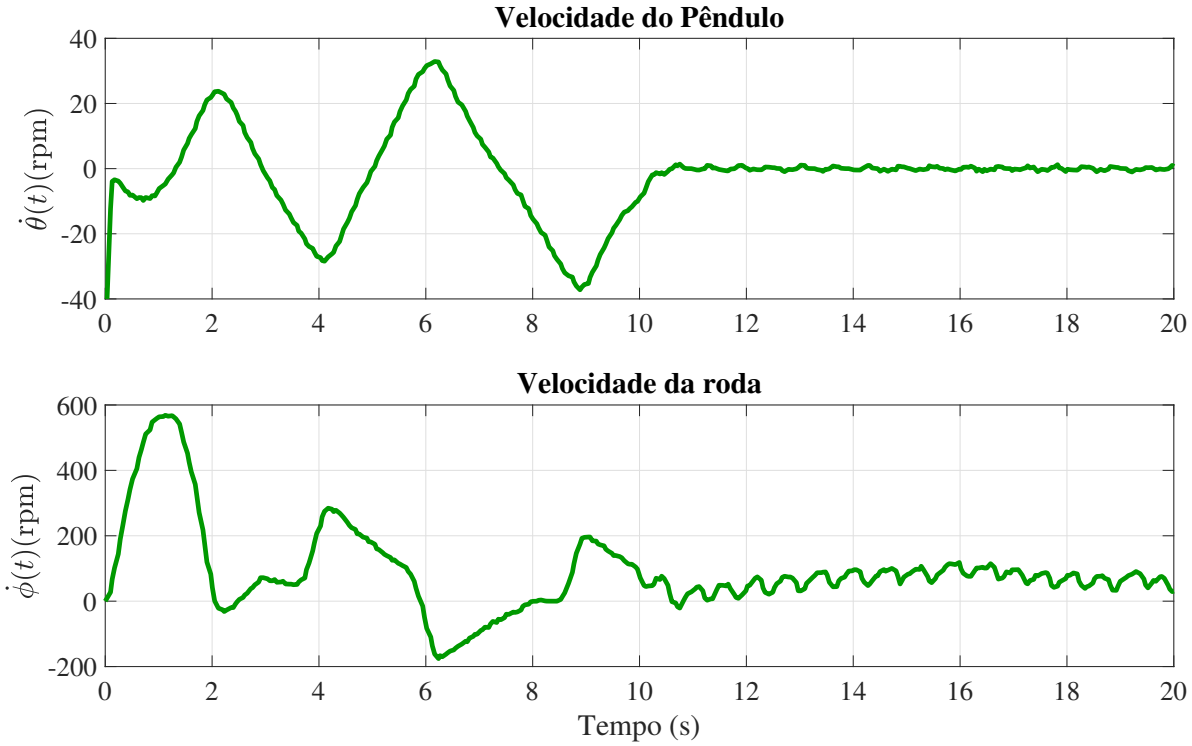
Fonte: próprio autor.

Nota-se que após 10 segundos, o controle *switch* troca o controlador *swing up* pelo controlador CMD. Portanto, o controle *swing up* foi capaz de alcançar a posição invertida com apenas três oscilações do sistema PIRR.

Observa-se que o sinal de tensão correspondente ao controlador *swing up* atua até aproximadamente $t = 10$ segundos. A partir desse instante, ocorre a comutação por meio do controle *switch*, acionando o controlador de estabilização, o qual passa a aplicar o sinal de tensão resultante da lei de controle CMD.

Na Figura 36 são apresentadas as respostas das velocidades do pêndulo e da roda. Observa-se que a velocidade do pêndulo possui velocidade próxima de zero em regime permanente. Nos instantes iniciais, a roda de reação opera com velocidade angular elevada, atingindo valores próximos de 600 rpm. Durante a atuação do controlador de estabilização, a velocidade da roda apresenta redução significativa; entretanto, não atinge valores nulos.

Figura 36 – Resultado experimental do controle *swing up* e CMD com os estados de velocidades do pêndulo e da roda.



Fonte: próprio autor.

No Apêndice B.3 é apresentado o código que considera a condicional para o controle do PIRR com CMD e *swing up* em ambas as regiões.

6.3 RESULTADOS DOS CONTROLADORES LINEARES

Nesta seção são apresentados os resultados experimentais para os controladores lineares projetados, que considera a condição inicial $C.I. = [\theta\ \dot{\theta}\ \dot{\phi}]^T = [11{,}45°\ 0\ 0]^T$, ou seja o controle opera na região de estabilização. O comportamento do sistema PIRR, com os controladores projetados, foi verificado por meio de simulações realizadas no MATLAB/SIMULINK®, cujos resultados são apresentados nas Seções 5.2 e 5.3.

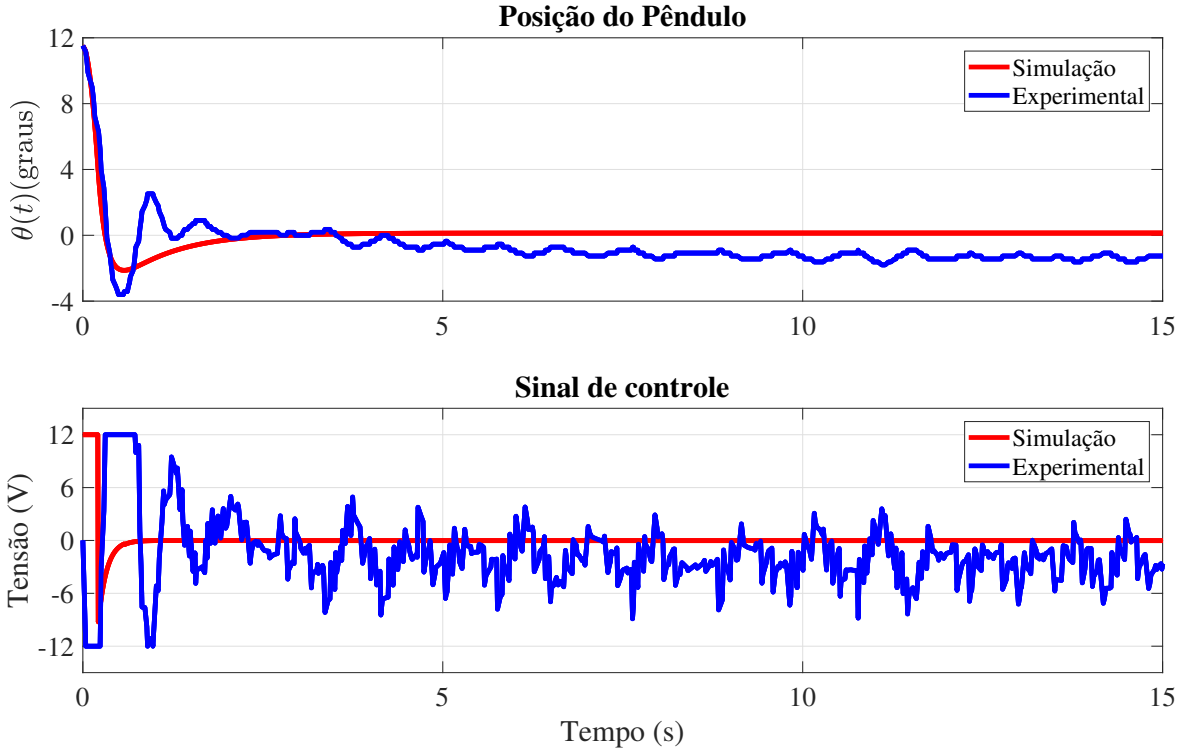
6.3.1 Resultados do Controlador por Alocação de Autovalores

Nesta subseção são apresentados os resultados experimentais obtidos para o controlador por alocação de autovalores, utilizando *hardware* de baixo custo. O controlador foi implementado na IDE Arduino empregando o microcontrolador ESP32, e a coleta de dados foi realizada através da comunicação serial entre o MATLAB® e ESP32.

Na Figura 37 são apresentadas as respostas de posição do pêndulo e sinal de controle para o controle de estabilização. Pode-se verificar que o controle foi capaz de

estabilizar a posição do pêndulo em torno do ponto de operação. Entretanto, o sinal de controle satura nos instantes iniciais, e logo se mantêm distantes do limite de saturação e com pequenas oscilações.

Figura 37 – Resultado experimental do controle por alocação de autovalores com posição e sinal de controle.

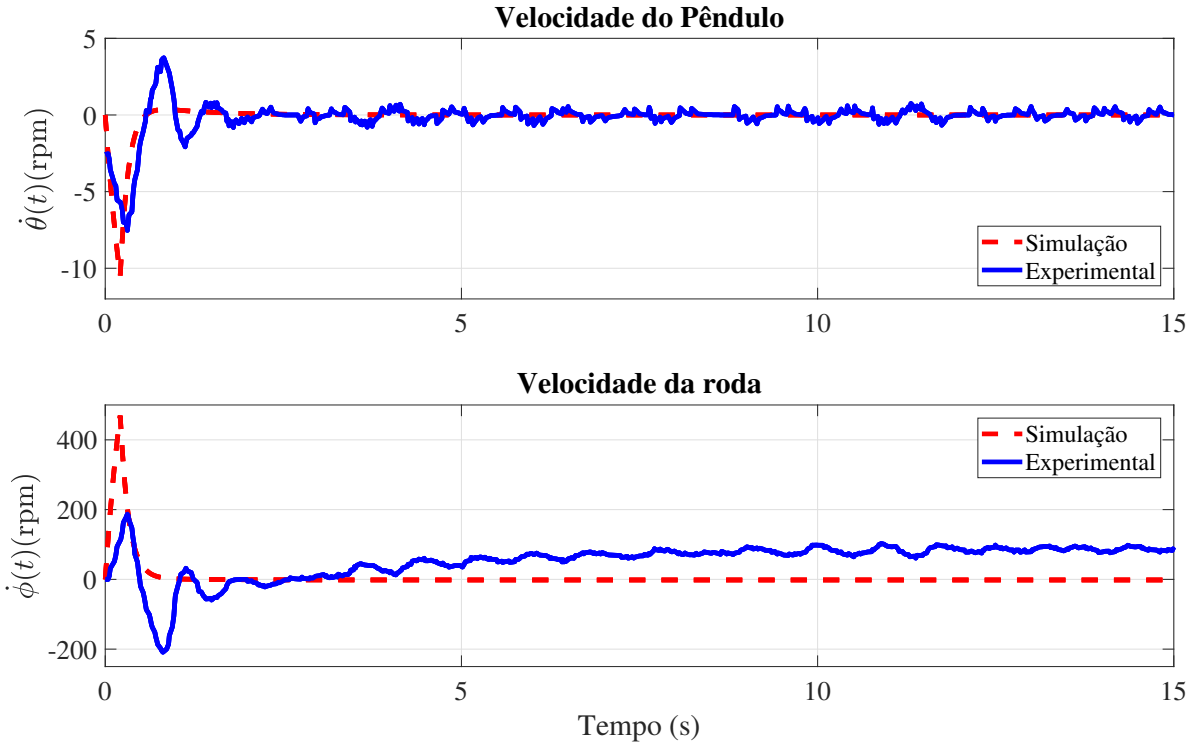


Fonte: próprio autor.

Observa-se também que na Figura 37 existe um erro de estado estacionário para a posição do pêndulo $\theta(t)$. Isto é devido ao torque de distúrbio aplicado ao pêndulo pelo cabo de conexão do motor CC. Além disso, o fenômeno de atrito de Coulomb e viscoso foram obtidos por estimativa, o que contribui para imperfeições no modelo matemático do sistema PIRR.

Na Figura 38 são apresentadas as repostas de velocidades do pêndulo e da roda de reação. Nota-se que devido ao torque de distúrbio resultante do cabo de conexão, a velocidade da roda $\dot{\phi}(t)$ possui erro de regime de aproximadamente $90\,rpm$, e a velocidade do pêndulo $\dot{\theta}(t)$ com valor perto de $0\,rpm$.

Figura 38 – Resultado experimental do controle por alocação de autovalores de velocidades do pêndulo e da roda.



Fonte: próprio autor.

O tempo de amostragem de controle foi definido para $5\,ms$. Este tempo de amostragem é suficientemente rápido para os ganhos por alocação de autovalores contínuo projetados. No Apêndice B.1 é apresentado o código implementado na IDE Arduino.

6.3.2 Resultados do Controlador PID

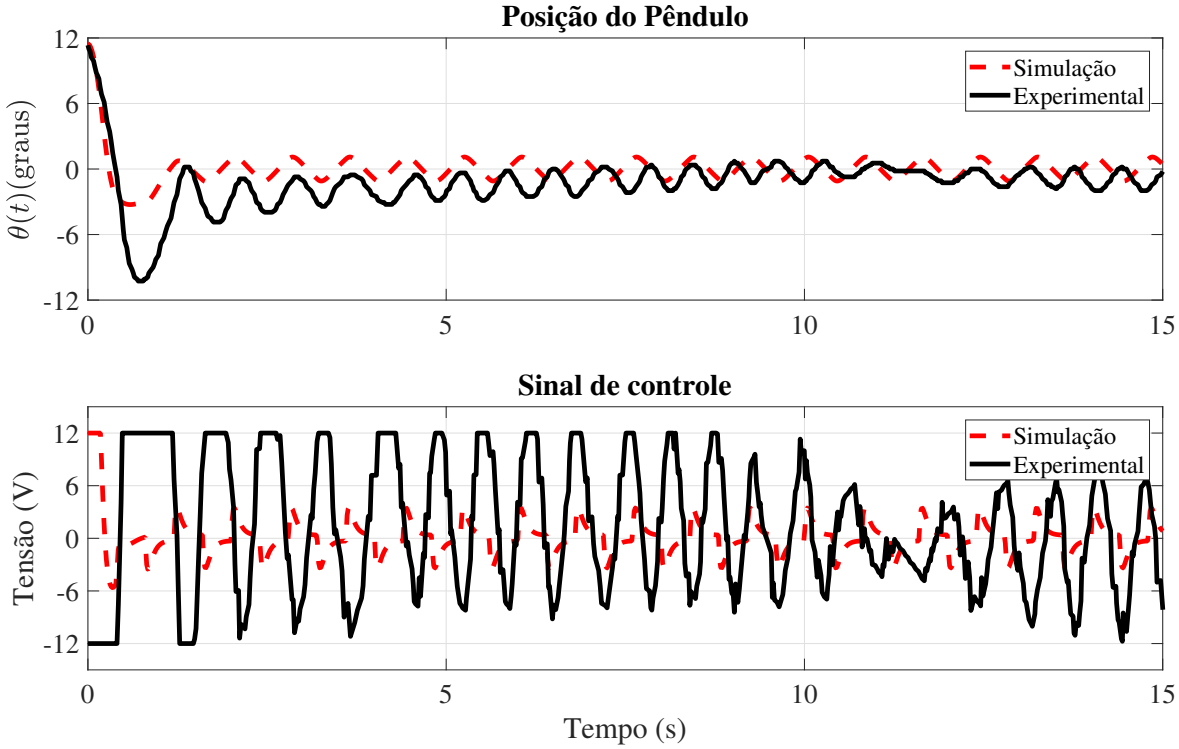
Nesta subseção são mostrados os resultados experimentais referentes à aplicação do controlador PID que utiliza dispositivos de *hardware* de baixo custo. Inicialmente foram realizados experimentos em bancada com o controle PID, e foi levado em consideração a variação do valor de referência $\theta_d(t)$.

A determinação de um valor de referência estático, em $\theta_d(t) = 0°$ por exemplo, leva a ação integradora do PID a acumular erro $e(t)$ durante instantes de tempo em que o sinal de controle já está saturado. Para exemplificar essa instabilidade, considere que o pêndulo apresente uma pequena defasagem em relação ao ângulo desejado $\theta_d(t) = 0°$. Nesta condição, o termo integral do controlador tende a provocar um aumento gradual na velocidade de rotação da roda de reação. Contudo, essa variação lenta na velocidade angular gera um torque de magnitude reduzida, o qual é insuficiente para produzir a correção necessária, e que leva o PIRR a instabilidade.

Por isso foi considerada uma onda triangular para o valor de *setpoint*, a fim de evitar o acúmulo estático do integrador, e saturação do torque da roda de reação durante a estabilização.

Na Figura 39 são mostradas as respostas de posição do pêndulo com condição inicial de 11,45°, e sinal de controle PID para o controle de estabilização. Pode-se verificar que o controle foi capaz de levar o ângulo para o valor desejado $\theta_d(t)$. No entanto, nos instantes iniciais o motor atingiu a saturação.

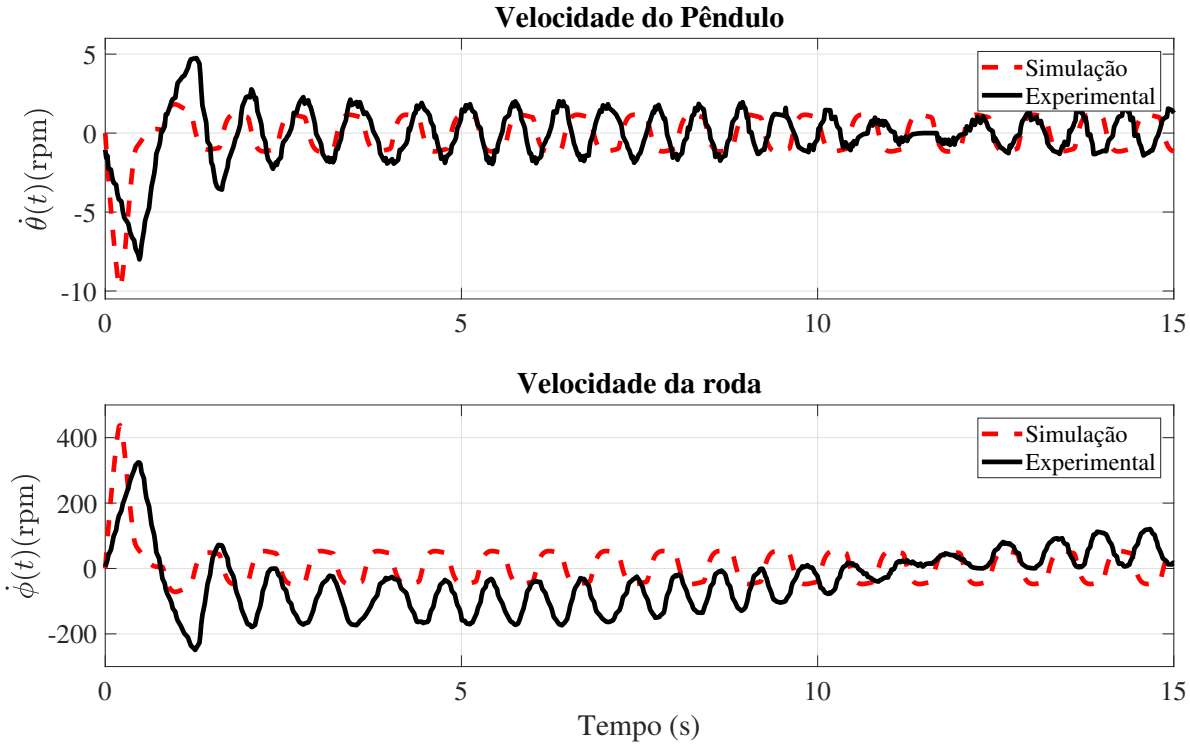
Figura 39 – Resultado experimental do controle PID com posição e sinal de controle.



Fonte: próprio autor.

Na Figura 40 são apresentadas as respostas de velocidades do pêndulo e da roda de reação. Observa-se que a velocidade do pêndulo varia em todo instante de tempo devido ao *setpoint* variável. Nota-se que velocidade da roda apresenta velocidade máxima de $350\,rpm$ e mínima de $300\,rpm$, e não apresentou saturação. Além disso, a velocidade da roda no controle PID varia muito quando comparada com a do controle por alocação de autovalores, em que apresenta uma velocidade na roda de reação mais constante.

Figura 40 – Resultado experimental do controle PID com os estados de velocidades do pêndulo e da roda.



Fonte: próprio autor.

O controlador PID foi implementado na IDE Arduino utilizando o microcontrolador ESP32, e o tempo de amostragem de controle foi definido para $5\,ms$. No Apêndice B.2 é apresentado o código implementado.

6.4 RESULTADOS DO CONTROLADOR CMD

Nesta seção são apresentados os resultados experimentais para o controle com modos deslizantes. A condição inicial para o controle CMD foi definida em $C.I. = [\theta\ \dot{\theta}\ \dot{\phi}]^T = [11{,}45°\ 0\ 0]^T$, pois esse controle opera na região de estabilização. O comportamento do sistema PIRR, com o controlador projetado, foi verificado por meio de simulações com o MATLAB/SIMULINK®, cujos resultados são apresentados na Seção 5.1.

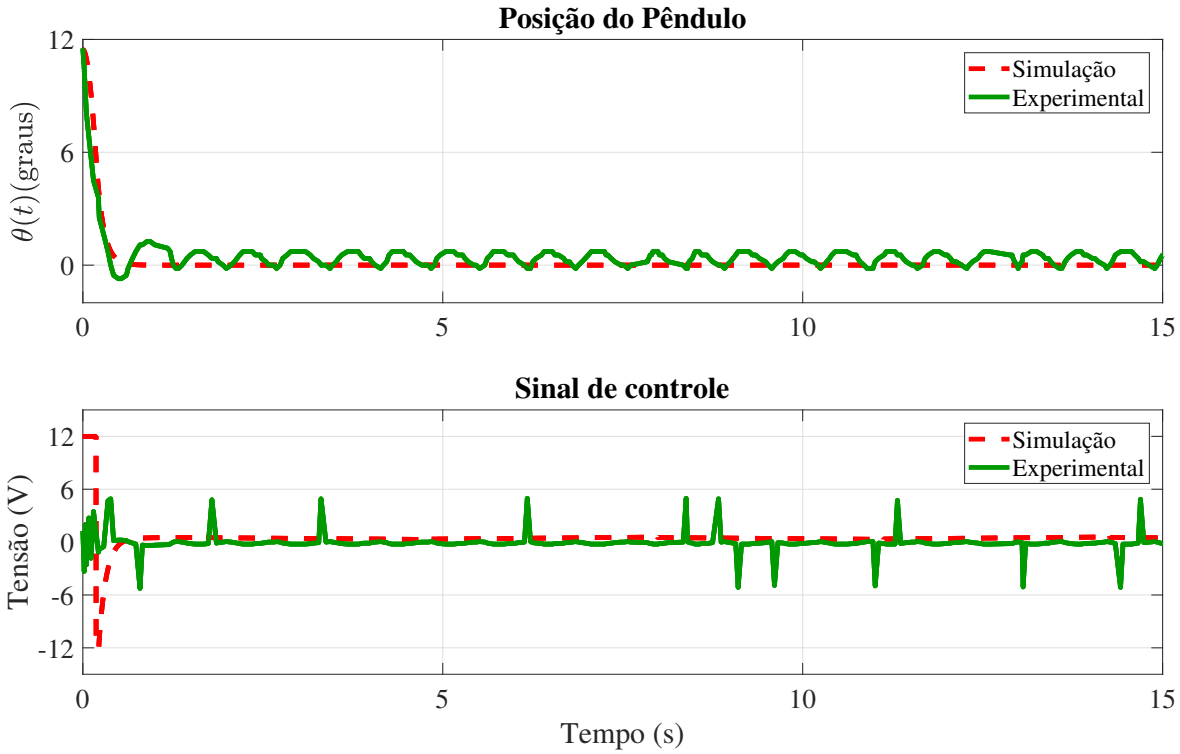
6.4.1 Resultados experimentais do Controlador CMD

Nesta subseção são apresentados os resultados experimentais referentes à implementação do controlador CMD, utilizando dispositivos de *hardware* de baixo custo para a validação prática do sistema de controle projetado.

Na Figura 41 são apresentadas as respostas de posição do pêndulo e sinal de

controle. Inicialmente, o controle CMD foi validado via simulação com o modelo não linear, e assumindo condição inicial próxima do ponto de equilíbrio instável igual à 11,45°.

Figura 41 – Resultado experimental do controle CMD com posição e sinal de controle.

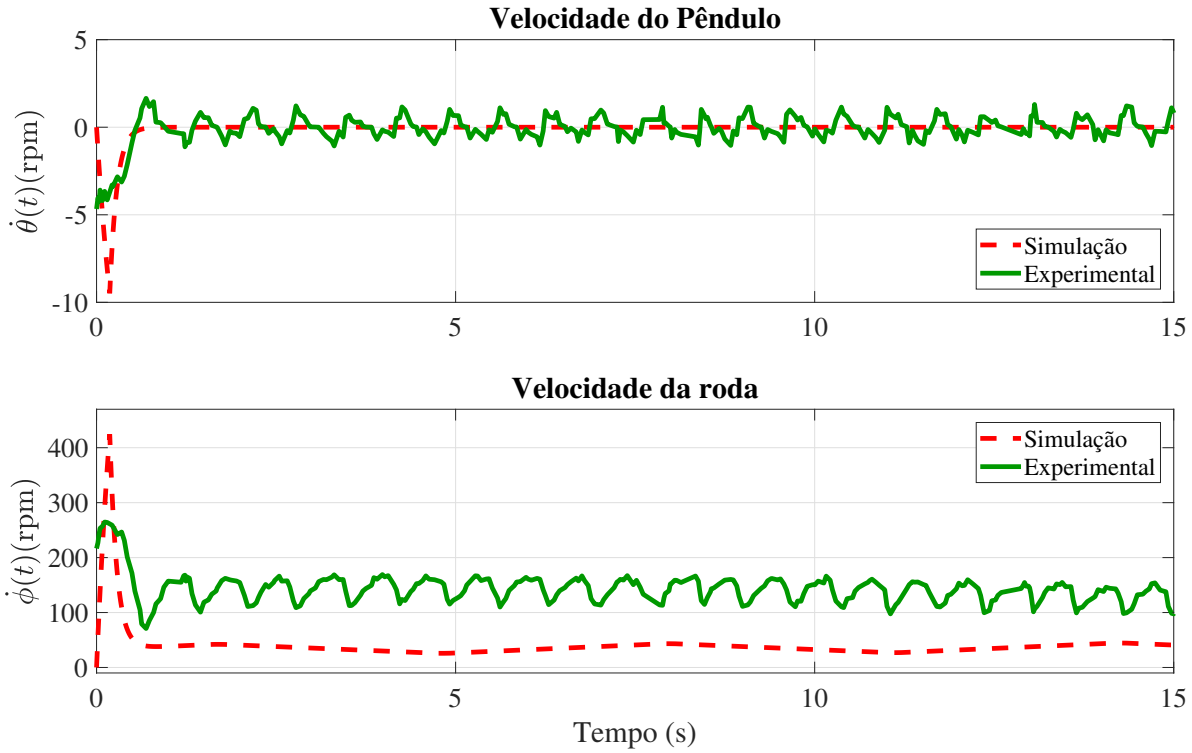


Fonte: próprio autor.

Observa-se que o controle foi capaz de levar a posição do pêndulo em torno do valor de referência $\theta_d(t) = 0°$. O valor de posição do pêndulo oscila com valor máximo de 0,5° em torno do ponto de operação. Pode-se verificar que a tensão de entrada apresenta valores próximos de $0{,}5\,V$, e em determinados instantes de tempo essa tensão aumenta rapidamente, contudo o sinal de controle não satura.

Na Figura 42 são apresentadas as repostas de velocidades do pêndulo e da roda de reação. Observa-se que a velocidade do pêndulo varia em baixa velocidade. Pode-se notar que a velocidade da roda apresenta valor de velocidade máxima de $250\,rpm$, e após 1 segundo essa velocidade diminui para aproximadamente $150\,rpm$.

Figura 42 – Resultado experimental do controle CMD com os estados de velocidades do pêndulo e da roda.



Fonte: próprio autor.

No algoritmo de controle foi considerado um tempo de amostragem de $3\,ms$. No Apêndice B.3 é apresentado o código implementado.

6.5 RESULTADOS COMPARATIVOS DOS CONTROLADORES DE ESTABILIZAÇÃO

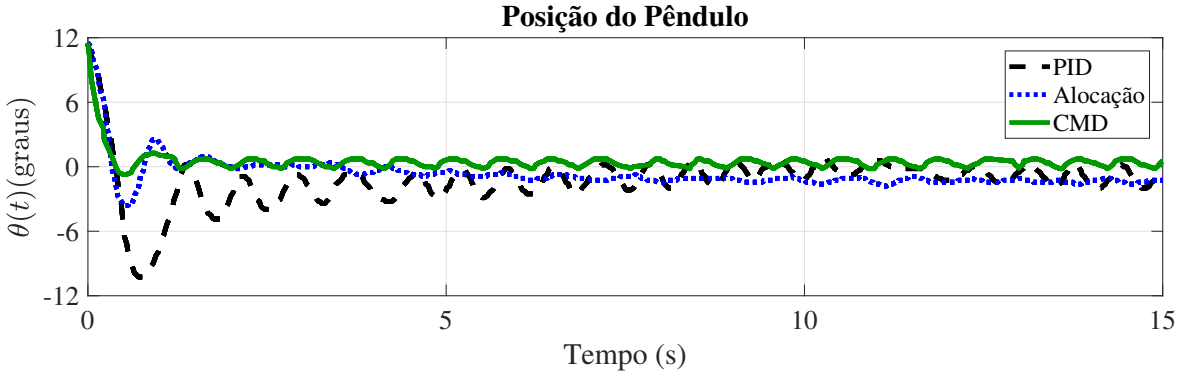
Nesta seção são apresentados índices de desempenho para avaliar a resposta da variável de estado controlada $\theta(t)$. Tais índices constituem métricas apropriadas para estabelecer uma comparação objetiva entre dois métodos de controle de estabilização para o pêndulo invertido por roda de reação.

Na Figura 43 é apresentada a posição angular controlada para o sistema PIRR, em que se considera os controladores PID, alocação de autovalores e controle CMD.

Observa-se que a técnica de controle CMD apresenta o melhor desempenho, pois esse método apresenta menor tempo de subida e tempo de pico, além de atingir o regime permanente com o menor tempo de acomodação. Além disso, o método praticamente não exibe sobressinal, mantendo oscilações reduzidas em torno do ponto de equilíbrio. Em contraste, os controladores PID e o controlador por alocação de autovalores apresentam tempos de resposta superiores e maior variação transitória, resultando em uma

estabilização mais lenta do sistema.

Figura 43 – Resultado comparativo do controle de posição angular.



Fonte: próprio autor.

O controle CMD alcançou o valor de referência $\theta_d(t) = 0°$ em aproximadamente meio segundo, enquanto o controle por alocação de autovalores levou três segundos, e o controle PID demorou cerca de seis segundos para se estabilizar em torno do ponto de operação.

Apesar da proximidade entre os resultados experimentais, a utilização de uma medida quantitativa de desempenho mostra-se útil para uma análise mais precisa. Essa avaliação pode ser realizada por meio do cálculo de índices de desempenho, que permitem quantificar e comparar o comportamento obtido sob diferentes condições de controle.

Um critério de desempenho rapidamente instrumentado é o da integral do valor absoluto do erro (do inglês, *Integral of Absolute Error* - IAE), que é a soma total do erro ao longo do tempo (Dorf; Bishop, 2009):

$$\text{IAE} = \int_0^T |e(t)|\, dt, \tag{103}$$

sendo $|e(t)|$ o erro absoluto. Outro índice de desempenho muito utilizado é a integral do quadrado do erro (do inglês, *Integral of Squared Error* - ISE) (Dorf; Bishop, 2009). A ISE é descrita da seguinte forma:

$$\text{ISE} = \int_0^T e(t)^2\, dt. \tag{104}$$

A integral do tempo multiplicado pelo valor absoluto do erro (do inglês, *Integral of Time multiplied by Absolute Error* - ITAE) é um índice de desempenho amplamente conhecido, que visa diminuir a influência dos erros iniciais e enfatizar os que surgem nas etapas finais da resposta (Dorf; Bishop, 2009). Este índice pode ser escrito como:

$$\text{ITAE} = \int_0^T t|e(t)|\, dt. \tag{105}$$

Outro índice de desempenho é o da integral do tempo multiplicado pelo quadrado do erro (do inglês, *Integral of Time multiplied by the Squared Error* - ITSE) (Dorf; Bishop, 2009):

$$\text{ITSE} = \int_0^T te^2(t)\,dt. \tag{106}$$

Para que um índice de desempenho seja útil, ele deve assumir apenas valores positivos ou nulos. Portanto o melhor sistema é definido como o sistema que minimiza este índice (Dorf; Bishop, 2009).

Na Tabela 5 são apresentados os valores obtidos de índices de desempenho para o resultado experimental do controle de posição.

Tabela 5 – Índices de desempenho do controle de posição.

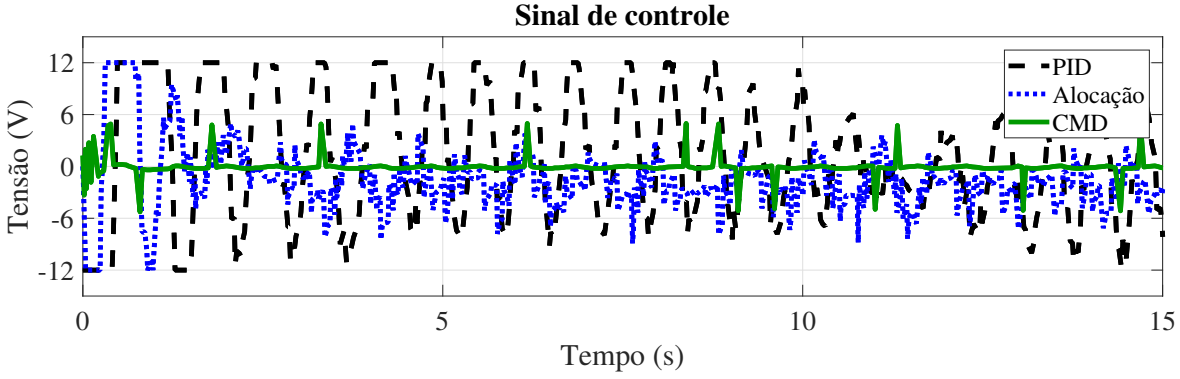
Controladores	**IAE**	**ISE**	**ITAE**	**ITSE**
CMD	0,1041	0,0040	0,8240	0,0208
Alocação	0,2571	0,0112	2,1170	0,0564
PID	0,3056	0,0289	3,4841	0,3099

Fonte: próprio autor.

Analisando-se os resultados, pode-se verificar que o controlador CMD apresentou melhor resultado de controle de posição do pêndulo, pois os índices de desempenho desse controlador apresentam menores valores. Para os cálculos dos índices de desempenho, foi considerado um tempo de 15 segundos para as análises.

Na Figura 44 são apresentadas as tensões aplicadas ao motor CC da roda de reação por cada controlador. Observa-se que a tensão aplicada no motor CC com controle PID, apresenta um valor de tensão mais elevado nos instantes iniciais, no entanto, após 10 segundos as tensões aplicadas nos controladores PID e realimentação são muito próximas; o sinal de controle CMD não apresenta saturação.

Figura 44 – Comparação da tensão aplicada no motor CC da roda de reação.



Fonte: próprio autor.

Dessa forma, para uma melhor avaliação desses resultados foi calculado o valor eficaz das tensões, conforme apresentado na Tabela 6.

Tabela 6 – Valor eficaz das tensões aplicadas no motor CC.

Controladores	Valor eficaz (**RMS**)
CMD	1,1628
Alocação	4,4815
PID	9,7199

Fonte: próprio autor.

O valor eficaz (do inglês, *Root Mean Square* - RMS), ou raiz do valor médio quadrático dos controles CMD e realimentação apresentaram menor valor RMS, e que indica um menor consumo de energia. Isso significa que o controlador CMD aplicou em média, uma tensão eficaz de $1{,}1628\,V$ nos instantes de tempo avaliado, enquanto o controlador por alocação de autovalores aplicou tensão eficaz de $4{,}4815\,V$, conforme Tabela 6.

No cálculo do valor eficaz para os sinais de tensões, foi considerado um tempo de 15 segundos para as análises.

Foi disponibilizado um vídeo que demonstra o desempenho dos controladores implementados em tempo real:

https://www.youtube.com/watch?v=zbQyZXtpmvA.

7 CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as conclusões e propostas para trabalhos futuros.

7.1 CONCLUSÕES

Nesta tese foram feitas implementações práticas de controladores clássicos e com modos deslizantes para o controle de posição do sistema PIRR, utilizando componentes de *hardware* de baixo custo. Também foi considerado o método de controle *switch* que envolve dois tipos de controladores: um para elevar o pêndulo até a região invertida, e outro para controlar o pêndulo em seu ponto de equilíbrio instável na posição invertida.

Primeiramente, foi realizada a modelagem matemática do PIRR para descrever o sistema em espaço de estados. Posteriormente, foi identificado o atrito do pêndulo experimentalmente em bancada a fim de ser comparado com o resultado de simulação da equação de movimento do pêndulo que inclui o modelo de atrito.

Foram projetados os controladores PID, por alocação de autovalores e o controlador CMD com o objetivo de estabilizar o pêndulo no ponto de equilíbrio instável, definido por $\theta(t) = 0°$, possibilitando a realização de uma análise comparativa de desempenho entre os métodos empregados. A avaliação dos controladores foi realizada por meio do cálculo de índices de desempenho, permitindo a quantificação e a comparação objetiva dos resultados obtidos.

Também, foi considerada uma estratégia de controle *switch* para atuação dos controladores *swing up* e de estabilização. Para alcançar o objetivo proposto, foi projetado um controlador *swing up*, responsável por elevar o pêndulo do ponto de equilíbrio naturalmente estável até a posição invertida. Adicionalmente, foram considerados controladores não lineares e lineares para a realização do controle de estabilização.

O comportamento do sistema PIRR, com os controladores projetados, foi verificado por meio de simulações realizadas no MATLAB/SIMULINK®. Após, foram implementados os códigos na IDE Arduino considerando as leis de controles projetadas, utilizando o microcontrolador ESP32. A coleta de dados foi realizada através da comunicação serial entre o MATLAB® e ESP32.

Os resultados experimentais indicaram que o controlador CMD apresentou desempenho superior em comparação aos controladores clássicos. O controlador CMD evidenciou elevada robustez em relação às incertezas paramétricas do sistema, obtendo os melhores índices de desempenho, mesmo quando parte dos parâmetros da planta é conhecida com precisão limitada. Essa constatação foi confirmada pela análise dos índices de desempenho, nos quais o CMD apresentou os menores valores.

7.2 PROPOSTAS DE CONTINUIDADE DO TRABALHO

Os próximos passos para a pesquisa são:

a) Implementação do controle digital;

b) Melhorar os componentes de *hardware* da planta;

c) Projetar e implementar controle com LMIs no sistema PIRR;

d) Projetar e implementar o controle por modos deslizantes integral;

e) Propor uma nova superfície de deslizamento não linear.

REFERÊNCIAS

ABDULLAH, M.; AMIN, A. A.; IQBAL, S.; MAHMOOD-UL-HASAN, K. Swing up and stabilization control of rotary inverted pendulum based on energy balance, fuzzy logic, and LQR controllers. **Measurement and Control**, London, v. 54, n. 9–10, p. 1356–1370, 2021.

ABRAMOVITCH, D. Y. A unified framework for analog and digital PID controllers. *In:* IEEE CONFERENCE ON CONTROL APPLICATIONS (CCA), 2015, Sydney. **Proceedings of the IEEE Conference on Control Applications**. Piscataway: IEEE, 2015.

ABRAMOVITCH, D. Y. Thoughts on furthering the control education of practicing engineers. **IEEE Control Systems Magazine**, Piscataway, v. 43, n. 1, p. 64–88, 2023.

ALVES, J. L. L. **Instrumentação, controle e automação de processos**. Rio de Janeiro: LTC, 2013.

ALVES, R. M. R.; NEVES, G. P. d.; ANGÉLICO, B. A. Modelagem, construção e controle de um pêndulo invertido com roda de reação. *In:* CONGRESSO BRASILEIRO DE AUTOMÁTICA (CBA), XXII., 2019, Belo Horizonte. **Anais do XXII Congresso Brasileiro de Automática**. São Paulo: SBA, 2019.

ANDREASSA, M. M. **Construção, modelagem e controle de um pêndulo com roda de reação utilizando técnicas modernas**. 2022. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual Paulista, 2022.

ÅSTRÖM, K. J. Hybrid control of inverted pendulums. *In:* Yamamoto, Y., Hara, S. (ed) Learning, control and hybrid systems. **Lecture Notes in Control and Information Sciences**. London: Springer, v. 241, 1999.

ÅSTRÖM, K. J.; FURUTA, K. Swinging up a pendulum by energy control. **IFAC Proceedings Volumes**, Amsterdam, v. 29, n. 1, p. 1919–1924, 1996.

ÅSTRÖM, K. J.; FURUTA, K. Swinging up a pendulum by energy control. **Automatica**, Amsterdam, v. 36, n. 2, p. 287–295, 2000.

ÅSTRÖM, K. J.; HÄGGLUND, T. **Advanced PID control**. North Carolina: ISA, 2006.

ÅSTRÖM, K. J.; HÄGGLUND, T. **PID controllers: theory, design, and tuning**. 2. ed. New York: ISA, 1995.

BAPIRAJU, B.; SRINIVAS, K. N.; KUMAR, P. P.; BEHERA, L. On balancing control strategies for a reaction wheel pendulum. *In:* IEEE INDIA ANNUAL CONFERENCE

(INDICON), XXII., 2004, Kharagpur. **Proceedings of the XXII IEEE India Annual Conference**. Piscataway: IEEE, 2004.

BARBASHIN, E. A.; GERASCHENKO, E. I. On speeding up sliding modes in automatic control systems. **Differentzialnye Uravneniya**, Moscow, v. 1, p. 25–32, 1965.

BARTOSZEWICZ, A. A new reaching law for sliding mode control of continuous time systems with constraints. **Transactions of the Institute of Measurement and Control**, London, v. 37, n. 4, p. 515–521, 2015.

BLOCK, D. J.; ÅSTRÖM, K. J.; SPONG, M. W. **The reaction wheel pendulum**. San Rafael: Morgan & Claypool, 2007.

BORSOTTO, B.; GODOY, E.; BEAUVOIS, D.; DEVAUD, E. An identification method for static and coulomb friction coefficients. **International Journal of Control, Automation and Systems**, Berlin, v. 7, p. 305–310, 2009.

BROWN, T. L.; SCHMIEDELER, J. P. Energetic effects of reaction wheel actuation on underactuated biped robot walking. *In:* IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), XXXI., 2014, Hong Kong. **Proceedings of the XXXI IEEE International Conference on Robotics and Automation**. Piscataway: IEEE, 2014.

CASTRUCCI, P. B. d. L.; BITTAR, A.; SALES, R. M. **Controle automático**. 2. ed. Rio de Janeiro: LTC, 2018.

CHEN, C.-T. **Analog and digital control system design: transfer-function, state-space, and algebraic methods**. New York: Oxford University Press, 1993.

CHEN, C.-T. **Linear system theory and design**. 3. ed. Fort Worth: Saunders College Publishing, 1999.

COLAGROSSI, A. Integrated magnetic management of stored angular momentum in autonomous attitude control systems. **Aerospace**, Basel, v. 10, n. 2, p. 103, 2023.

DECARLO, R. A.; ZAK, S. H.; MATTHEWS, G. P. Variable structure control of nonlinear multivariable systems: a tutorial. **Proceedings of the IEEE**, Piscataway, v. 76, n. 3, p. 212–232, 1988.

DENNEHY, C. J. A survey of reaction wheel disturbance modeling approaches for spacecraft line-of-sight jitter performance analysis. *In:* EUROPEAN SPACE MECHANISMS AND TRIBOLOGY SYMPOSIUM (ESMATS), XVIII., 2019, Munich. **Proceedings of the XVIII European Space Mechanisms and Tribology Symposium**. Noordwijk: European Space Agency, 2019.

DEVIKA, K. B.; THOMAS, S. Sliding mode controller design for MIMO nonlinear systems: a novel power rate reaching law approach for improved performance. **Journal of the Franklin Institute**, Oxford, v. 355, n. 12, p. 5082–5098, 2018.

DORF, R. C.; BISHOP, R. H. **Sistemas de controle moderno**. 11. ed. Rio de Janeiro: LTC, 2009.

DURAND, S.; GUERRERO-CASTELLANOS, J. F. Event-based digital PID control. *In:* INTERNATIONAL CONFERENCE ON EVENT-BASED CONTROL, COMMUNICATION, AND SIGNAL PROCESSING (EBCCSP), I., 2015, Krakow. **Proceedings of the I International Conference on Event-Based Control, Communication, and Signal Processing**. Piscataway: IEEE, 2015.

EDWARDS, C.; SPURGEON, S. **Sliding mode control: theory and applications**. London: Taylor & Francis, 1998.

EKER, I. Second-order sliding mode control with experimental application. **ISA Transactions**, Amsterdam, v. 49, n. 3, p. 394–405, 2010.

EMELYANOV, S.; UTKIN, V.; TARIN, V.; KOSTYLEVA, N.; SHUBLADZE, A.; EZEROV, V.; DUBROVSKY, E. **Theory of variable structure control systems (in Russian)**. Moscow: Nauka, 1970.

EMELYANOV, S. V. Binary control systems. **International Research Institute for Management Sciences**, [*S.l.*], v. 1, 1985.

ESTEVES, L. M. **Acionamento de motor de indução trifásico, sujeito a falhas, utilizando controle com estrutura variável e modos deslizantes**. 2016. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual Paulista, 2016.

FALLAHA, C. J.; SAAD, M.; KANAAN, H. Y.; AL-HADDAD, K. Sliding-mode robot control with exponential reaching law. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 58, n. 2, p. 600–610, 2010.

FANTONI, I.; LOZANO, R. **The reaction wheel pendulum: nonlinear control for underactuated mechanical systems**. London: Springer, 2002.

FANTONI, I.; LOZANO, R.; SPONG, M. W. Energy based control of the pendubot. **IEEE Transactions on Automatic Control**, Piscataway, v. 45, n. 4, p. 725–729, 2000.

FANTONI, I.; LOZANO, R.; SPONG, M. W. Stabilization of the reaction wheel pendulum using an energy approach. *In:* EUROPEAN CONTROL CONFERENCE (ECC), VI., 2001, Porto. **Proceedings of the VI European Control Conference**. *[S.l.]*: IEEE, 2001.

FERRARA, A. **Sliding mode control of vehicle dynamics**. London: Institution of Engineering e Technology, 2017.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. **Sistemas de controle para engenharia**. 6. ed. Porto Alegre: Bookman, 2013.

FRANKLIN, G. F.; POWELL, J. D.; WORKMAN, M. L. **Digital control of dynamic systems**. 3. ed. Menlo Park: Addison-Wesley, 1998.

FURUTA, K.; PAN, Y. Sliding sector for variable structure system. *In:* Young, K. D. Özguner, Ü. (ed) Variable structure systems, sliding mode and nonlinear control. **Lecture Notes in Control and Information Sciences**. London: Springer, v. 247, 1999.

FURUTA, K.; YAMAKITA, M.; KOBAYASHI, S. Swing-up control of inverted pendulum using pseudo-state feedback. **Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering**, London, v. 206, n. 4, p. 263–269, 1992.

GAMBHIRE, S. J.; KISHORE, D. R.; LONDHE, P. S.; PAWAR, S. N. Review of sliding mode based control techniques for control system applications. **International Journal of Dynamics and Control**, London, v. 9, n. 1, p. 363–378, 2021.

GAO, W.; HUNG, J. C. Variable structure control of nonlinear systems: a new approach. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 40, n. 1, p. 45–55, 1993.

GAO, W.; WANG, Y.; HOMAIFA, A. Discrete-time variable structure control systems. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 42, n. 2, p. 117–122, 1995.

GOGOUSSIS, A.; DONATH, M. Coulomb friction joint and drive effects in robot mechanisms. *In:* IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), IV., 1987, Raleigh. **Proceedings of the IV IEEE International Conference on Robotics and Automation**. Piscataway: IEEE, 1987.

GOLDSTEIN, H.; POOLE, C.; SAFKO, J. **Classical mechanics**. 3. ed. [*S. l.*]: American Association of Physics Teachers, 2002.

HE, Y.; XU, W.; CHENG, Y. A novel scheme for sliding-mode control of DC–DC converters with a constant frequency based on the averaging model. **Journal of Power Electronics**, Seoul, v. 10, n. 1, p. 1–8, 2010.

HERNANDEZ, V. M.; SIRA-RAMIREZ, H. Generalized PI control for swinging up and balancing the inertia wheel pendulum. *In:* AMERICAN CONTROL CONFERENCE

(ACC), XXII, 2003, Denver. **Proceedings of the XXII American Control Conference**. Piscataway: IEEE, 2003.

HU, C.; WAN, F. Parameter identification of a model with coulomb friction for a real inverted pendulum system. *In:* CHINESE CONTROL AND DECISION CONFERENCE (CCDC), XXI., 2009, Guilin. **Proceedings of the XXI Chinese Control and Decision Conference**. Piscataway: IEEE, 2009.

HU, Q. Robust adaptive sliding-mode fault-tolerant control with L2-gain performance for flexible spacecraft using redundant reaction wheels. **IET Control Theory & Applications**, London, v. 4, n. 6, p. 1055–1070, 2010.

HUNG, J. Y.; GAO, W.; HUNG, J. C. Variable structure control: a survey. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 40, n. 1, p. 2–22, 1993.

JEERANANTASIN, N.; NUNGAM, S. Sliding mode control of three-phase AC/DC converters using exponential rate reaching law. **Journal of Systems Engineering and Electronics**, Beijing, v. 33, n. 1, p. 210–221, 2022.

JUNEJO, A. K.; XU, W.; MU, C.; ISMAIL, M. M.; LIU, Y. Adaptive speed control of PMSM drive system based a new sliding-mode reaching law. **IEEE Transactions on Power Electronics**, Piscataway, v. 35, n. 11, p. 12110–12121, 2020.

KELLY, R.; LLAMAS, J. Determination of viscous and Coulomb friction by using velocity responses to torque ramp inputs. *In:* IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), XVI, 1999, Detroit. **Proceedings of the XVI IEEE International Conference on Robotics and Automation**. Piscataway: IEEE, 1999.

KOMURCUGIL, H.; BAYHAN, S.; GULER, N.; ABU-RUB, H. A new exponential reaching law approach to the sliding mode control: a multilevel multifunction converter application. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 70, n. 8, p. 7557–7568, 2023.

KRIM, Y.; ABBES, D.; KRIM, S.; MIMOUNI, M. F. Classical vector, first-order sliding-mode and high-order sliding-mode control for a grid-connected variable-speed wind energy conversion system: a comparative study. **Wind Engineering**, London, v. 42, n. 1, p. 16–37, 2018.

KUMAR, K. V.; KUMAR, T. A.; GANESH, V. Chattering free sliding mode controller for load frequency control of multi area power system in deregulated environment. *In:* POWER INDIA INTERNATIONAL CONFERENCE (PIICON), VII., 2016, Bikaner. **Proceedings of the Power India International Conference**. Piscataway: IEEE, 2016.

KUMAR, K. D.; ABREU, N.; SINHA, M. Fault-tolerant attitude control of miniature satellites using reaction wheels. **Acta Astronautica**, Amsterdam, v. 151, p. 206–216, 2018.

LEE, C.-Y.; YANG, S.; BOKSER, B.; MANCHESTER, Z. Enhanced balance for legged robots using reaction wheels. *In:* IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), XL., 2023, London. **Proceedings of the XL IEEE International Conference on Robotics and Automation**. Piscataway: IEEE, 2023.

LIAO, K.; XU, Y. A robust load frequency control scheme for power systems based on second-order sliding mode and extended disturbance observer. **IEEE Transactions on Industrial Informatics**, Piscataway, v. 14, n. 7, p. 3076–3086, 2017.

LIRA, J. A. How to stabilize a satellite using the principle of conservation of angular momentum. **Physics Education**, Bristol, v. 60, n. 3, p. 035015, 2025.

LOGHIS, E. K.; XIROS, N. I. Development of discrete-time waterjet control systems used in surface vehicle thrust vectoring. **Journal of Marine Science and Engineering**, Basel, v. 10, n. 12, p. 1844, 2022.

MATHEW, N. J.; RAO, K. K.; SIVAKUMARAN, N. Swing up and stabilization control of a rotary inverted pendulum. **IFAC Proceedings Volumes**, Amsterdam, v. 46, n. 32, p. 654–659, 2013.

MEHRJARDI, M. F.; SANUSI, H.; ALI, M. A. M. Developing a proposed satellite reaction wheel model with current mode control. *In:* INTERNATIONAL CONFERENCE ON SPACE SCIENCE AND COMMUNICATION (ICONSPACE), II., 2015, Langkawi. **Proceedings of the II International Conference on Space Science and Communication**. [*S. l.*]: IEEE, 2015.

MERIAM, J. L.; KRAIGE, G. **Engineering mechanics: dynamics**. 6. ed. [*S. l.*]: John Wiley & Sons, 2006.

MOHD ZAIHIDEE, F.; MEKHILEF, S.; MUBIN, M. Robust speed control of PMSM using sliding mode control (SMC)— a review. **Energies**, Basel, v. 12, n. 9, p. 1669, 2019.

MOLDOVEANU, F. Sliding mode controller design for robot manipulators. **Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I**, Braşov, v. 7, n. 2, p. 97, 2014.

MONK, S. **Programming Arduino: getting started with sketches**. New York: McGraw-Hill Education, 2016.

MOZAYAN, S. M.; SAAD, M.; VAHEDI, H.; FORTIN-BLANCHETTE, H.; SOLTANI, M. Sliding mode control of PMSG wind turbine based on enhanced exponential reaching law. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 63, n. 10, p. 6148–6159, 2016.

MUSKINJA, N.; TOVORNIK, B. Swinging up and stabilization of a real inverted pendulum. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 53, n. 2, p. 631–639, 2006.

NISE, S. N. **Engenharia de sistemas de controle**. 6. ed. Rio de Janeiro: LTC, 2012.

NOACK, D.; BRIESS, K. Laboratory investigation of a fluid-dynamic actuator designed for cubeSats. **Acta Astronautica**, Amsterdam, v. 96, p. 78–82, 2014.

OGATA, K. **Engenharia de controle moderno**. 5. ed. São Paulo: Prentice Hall do Brasil, 2010.

OHSUMI, A.; IZUMIKAWA, T. Nonlinear control of swing-up and stabilization of an inverted pendulum. *In:* IEEE CONFERENCE ON DECISION AND CONTROL (CDC), XXXIV., 1995, New Orleans. **Proceedings of the XXXIV IEEE Conference on Decision and Control**. Piscataway: IEEE, 1995.

OLIVARES, M.; ALBERTOS, P. On the linear control of underactuated systems: the flywheel inverted pendulum. *In:* IEEE INTERNATIONAL CONFERENCE ON CONTROL AND AUTOMATION (ICCA), X., 2013, Hangzhou. **Proceedings of the X IEEE International Conference on Control and Automation**. Piscataway: IEEE, 2013.

PARK, D.; CHWA, D.; HONG, S.-K. An estimation and compensation of the friction in an inverted pendulum. *In:* SICE-ICASE INTERNATIONAL JOINT CONFERENCE, [*S. l.*], 2006, Busan. **Proceedings of the International Joint Conference**. [*S. l.*]: IEEE, 2006.

RAHIMI, A.; KUMAR, K. D.; ALIGHANBARI, H. Fault isolation of reaction wheels for satellite attitude control. **IEEE Transactions on Aerospace and Electronic Systems**, Piscataway, v. 56, n. 1, p. 610–629, 2019.

RAMEZANI-AL, M. R.; TAVANAEI-SERESHKI, Z. An adaptive sliding mode controller with a new reaching law for tracking problem of an autonomous underwater vehicles. **Transactions of the Institute of Measurement and Control**, London, v. 41, n. 6, p. 1772–1787, 2019.

RIBEIRO, J. M. d. S. **Controle discreto com modos deslizantes em sistemas incertos com atraso no sinal de controle**. 2006. Tese (Doutorado em Engenharia Elétrica) – Universidade Estadual Paulista, 2006.

RIZAL, Y.; MANTALA, R.; RACHMAN, S.; NURMAHALUDIN, N. Balance control of reaction wheel pendulum based on second-order sliding mode control. *In:* INTERNATIONAL CONFERENCE ON APPLIED SCIENCE AND TECHNOLOGY (ICAST), I., 2018, Manado. **Proceedings of the I International Conference on Applied Science and Technology**. [*S. l.*]: IEEE, 2018.

SCHAUB, H.; JUNKINS, J. L. **Analytical mechanics of space systems**. 2. ed. Virginia: AIAA, 2003.

SHIYOU, Y.; ALI, A.; RAO, S.; FAHAD, S.; JING, W.; TONG, J.; TAHIR, M. Active attitude control for microspacecraft: a survey and new embedded designs. **Advances in Space Research**, Amsterdam, v. 69, n. 10, p. 3741–3769, 2022.

SHTESSEL, Y.; EDWARDS, C.; FRIDMAN, L.; LEVANT, A. **Sliding mode control and observation**. New York: Springer, 2014.

SINGH, G. K.; HOLÉ, K. E. Guaranteed performance in reaching mode of sliding mode controlled systems. **Sādhanā**, New Delhi, v. 29, p. 129–141, 2004.

SLOTINE, J.-J. E.; LI, W. **Applied nonlinear control**. Englewood Cliffs, NJ: Prentice Hall, 1991.

SPONG, M. W. The swing up control problem for the acrobot. **IEEE Control Systems Magazine**, Piscataway, v. 15, n. 1, p. 49–55, 1995.

SPONG, M. W.; CORKE, P.; LOZANO, R. Nonlinear control of the reaction wheel pendulum. **Automatica**, Amsterdam, v. 37, n. 11, p. 1845–1851, 2001.

SUN, X.; CAO, J.; LEI, G.; GUO, Y.; ZHU, J. A composite sliding mode control for SPMSM drives based on a new hybrid reaching law with disturbance compensation. **IEEE Transactions on Transportation Electrification**, Piscataway, v. 7, n. 3, p. 1427–1436, 2021.

TRENTIN, J. F. S.; CENALE, T. P.; SILVA, S. da; RIBEIRO, J. M. d. S. Attitude control of inverted pendulums using reaction wheels: comparison between using one and two actuators. **Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering**, London, v. 234, n. 3, p. 420–429, 2020.

TRENTIN, J. F. S.; SILVA, S. da; RIBEIRO, J. M. d. S.; SCHAUB, H. Inverted pendulum nonlinear controllers using two reaction wheels: design and implementation. **IEEE Access**, Piscataway, v. 8, p. 74922–74932, 2020.

UTKIN, V.; GULDNER, J.; SHI, J. **Sliding mode control in electro-mechanical systems**. Boca Raton: CRC Press, 2017.

UTKIN, V.; POZNYAK, A.; ORLOV, Y. V.; POLYAKOV, A. **Road map for sliding mode control design**. Cham: Springer, 2020.

UTKIN, V. I. Variable structure systems: present and future. **Automation and Remote Control**, New York, n. 9, p. 1105–1120, 1983.

UTKIN, V. I. **Sliding modes in control and optimization**. London: Springer-Verlag, 1992.

WANG, A.; JIA, X.; DONG, S. A new exponential reaching law of sliding mode control to improve performance of permanent magnet synchronous motor. **IEEE Transactions on Magnetics**, Piscataway, v. 49, n. 5, p. 2409–2412, 2013.

WANG, Y.; FENG, Y.; ZHANG, X.; LIANG, J. A new reaching law for antidisturbance sliding-mode control of PMSM speed regulation system. **IEEE Transactions on Power Electronics**, Piscataway, v. 35, n. 4, p. 4117–4126, 2019.

WEBER, X.; CUVILLON, L.; GANGLOFF, J. Active vibration canceling of a cable-driven parallel robot using reaction wheels. *In:* IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), XXVII., 2014, Chicago. **Proceedings of the XXVII International Conference on Intelligent Robots and Systems**. Piscataway: IEEE, 2014.

WU, L.; LIU, J.; VAZQUEZ, S.; MAZUMDER, S. K. Sliding mode control in power converters and drives: a review. **IEEE/CAA Journal of Automatica Sinica**, Beijing, v. 9, n. 3, p. 392–406, 2021.

XIN, X.; KANEDA, M. The swing up control for the acrobot based on energy control approach. *In:* IEEE CONFERENCE ON DECISION AND CONTROL (CDC), XLI., 2002, Las Vegas. **Proceedings of the XLI Conference on Decision and Control**. Piscataway: IEEE, 2002.

XU, Y.; KANADE, T. **Space robotics: dynamics and control**. 1. ed. New York: Springer, 1992.

YAMAKITA, M.; HOSHINO, T.; FURUTA, K. Control practice using pendulum. *In:* AMERICAN CONTROL CONFERENCE (ACC), XVIII., 1999, San Diego. **Proceedings of the XVIII American Control Conference**. Piscataway: IEEE, 1999.

YANG, Y. Spacecraft attitude and reaction wheel desaturation combined control method. **IEEE Transactions on Aerospace and Electronic Systems**, Piscataway, v. 53, n. 1, p. 286–295, 2017.

YIN, S.; XIAO, B.; DING, S. X.; ZHOU, D. A review on recent development of spacecraft attitude fault tolerant control system. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 63, n. 5, p. 3311–3320, 2016.

YU, X.; KAYNAK, O. Sliding-mode control with soft computing: a survey. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 56, n. 9, p. 3275–3285, 2009.

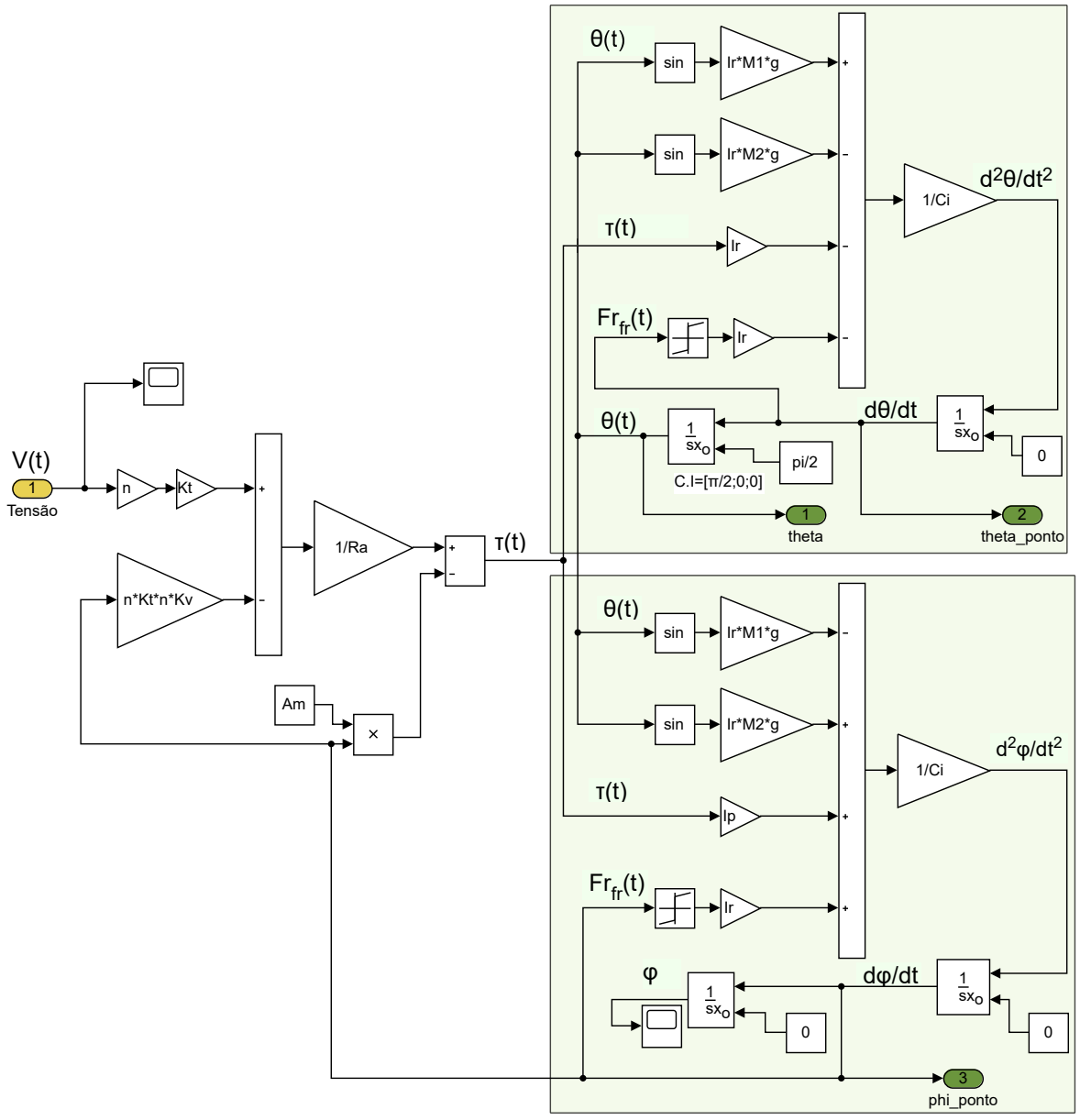
APÊNDICE A – SISTEMA PIRR NO MATLAB/SIMULINK®

Neste apêndice é apresentado o diagrama de blocos para simulação do sistema PIRR e controladores no MATLAB/SIMULINK®.

A.1 MODELO DO PIRR NO MATLAB/SIMULINK®

O modelo não linear da planta implementado no SIMULINK®, é apresentado a seguir:

Figura 45 – Diagrama de blocos do sistema não linear PIRR.



Fonte: próprio autor.

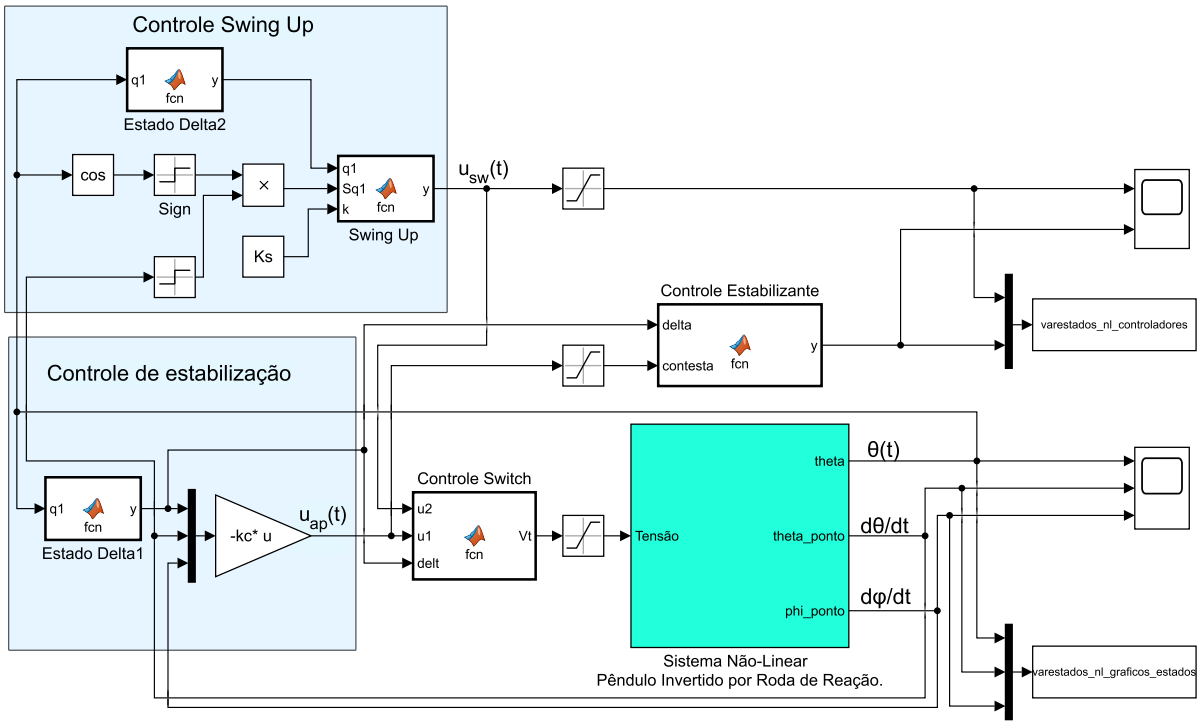
A seguir é apresentado o código com os parâmetros do PIRR no MATLAB® para ser utilizado no diagrama da Figura 45, e implementado no SIMULINK®.

```
%%Parâmetros do sistema Pêndulo Invertido por Roda de Reação    ...

clc;clear;close all

%-- Parâmetros da roda
Ir = 11.18*1e-4;      %kg m^2 - Momento de inércia da roda
Ma = 0.065;           %kg     - Massa do acoplamento da roda
Mm = 0.390;           %kg     - Massa do motor
Mr = 0.190;           %kg     - Massa da roda
Mtr = Ma + Mm + Mr;   %kg     - Massa Total da roda

%-- Parâmetros do pêndulo
Mh1 = 0.08;  %kg-Massa de meia haste
Mh2 = 0.08;
Lh1 = 0.25;  %m-Comprimento da haste em relação ao pivo
Lh2 = 0.25;
Lh3 = 0.2278;%m-Distância do pivo ao centro de massa da roda
Lh4 = 0.2272;%m-Distância do pivo ao centro de massa do contrapeso
Lch1 = 0.125;%m-Distância do pivo ao centro de massa da haste
Lch2 = 0.125;
Mc  = 0.350; %kg - Massa do contrapeso

Ip = ((1/3) * Mh1 * Lh1^2) + ((1/3) * Mh2 * Lh2^2) + Mtr*Lh3^2 + ...
   Mc*Lh4^2 + Ir; %-- kg m^2 - Momento de inércia do pêndulo

%-- simplificação
M1 = Mh1*Lch1 + Mtr*Lh3;
M2 = Mh2*Lch2 + Mc *Lh4;
%-- simplificação constante de inércia
Ci = (Ir*Ip)-(Ir^2);
g = 9.81;    %m/(s^2)

%-- motor
Am = 0.0014; %(Nm)/(rad/s)- coeficiente de atrito do motor
Kt = 0.0181; %(Nm)/A      - constante de torque
n = 5.2;     %            - redução do motor
Kv = 0.0181; %V/(rad/s)   - constante de campo
Ra = 1.6632; %ohms        - resistência de armadura
```

A.2 DIAGRAMA DE BLOCOS DO SISTEMA DE CONTROLE *SWING UP* E ALOCAÇÃO DE AUTOVALORES NO SIMULINK®

Para simular os controladores *swing up* e de controle por alocação de autovalores para estabilização, foi considerado o seguinte diagrama no SIMULINK®:

Figura 46 – Diagrama com os controladores *swing up*, alocação de autovalores, e o controle *switch* no SIMULINK®.



Fonte: próprio autor.

Para a identificação das informações internas dos blocos *MATLAB Function*, os quais permitem a inserção de código MATLAB® por meio de funções personalizadas (fcn), apresentam-se, a seguir, os respectivos códigos implementados em cada bloco.

A seguir é apresentado o código do bloco *MATLAB function* identificado como *Controle Estabilizante*.

```
1  %Controle Estabilizante
2  function y = fcn(delta,contesta)
3  %#codegen
4  if abs(delta) <= 0.20
5  u = contesta;
6  else
7  u = 0;
8  end
9  y = u;
```

A seguir é apresentado o código do bloco *MATLAB function* nomeado *Swing Up*.

```
%Controle Swing Up
function y = fcn(q1,Sq1,k)
%#codegen
u2 = k * q1^5 * Sq1;
y = u2;
```

A seguir é apresentado o código do bloco *MATLAB function* caracterizado como *Controle Switch*.

```
%Controle Switch
function Vt = fcn(u2,u1,delt)
%#codegen
if abs(delt) <= 0.20
u = u1;
else
u = u2;
end
Vt = u;
```

A seguir é apresentado o código do bloco *MATLAB function* nomeado *Estado Delta1*.

```
%Estado Delta1
function y = fcn(q1)
%#codegen
q2 = mod(q1 + pi, 2*pi);

if q2 < 0
q2 = q2 + 2*pi;
end

y = q2 - pi;
```

A seguir é apresentado o código do bloco *MATLAB function* identificado como *Estado Delta2*.

```
%Estado Delta2
function y = fcn(q1)
%#codegen
```

```
    q2 = mod(q1 + pi, 2*pi);

    if q2 < 0
    q2 = q2 + 2*pi;
    end

    y = abs(q2 - pi);
```

Por fim, é apresentado o código do modelo do PIRR linearizado, e projeto por alocação de autovalores no MATLAB® para ser utilizado no diagrama da Figura 46.

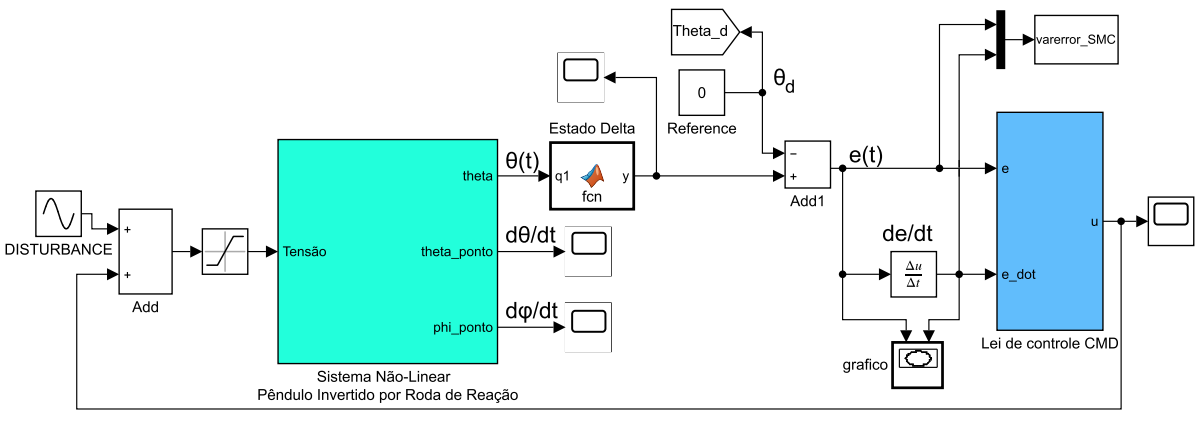
```
%%Modelo do Sistema PIRR Linearizado
clc;clear;close all
%-- Parâmetros da roda
Ir = 11.18*1e-4;   %kg m^2-Momento de inércia da roda
Ma = 0.065;        %kg-Massa do acoplamento da roda
Mm = 0.390;        %kg-Massa do motor
Mr = 0.190;        %kg-Massa da roda
Mtr = Ma + Mm + Mr;%kg-Massa Total da roda
%-- Parâmetros do pêndulo
Mh1 = 0.08;  %kg-Massa de meia haste
Mh2 = 0.08;
Lh1 = 0.25;  %m-Comprimento da haste em relação ao pivo
Lh2 = 0.25;
Lh3 = 0.2278;%m-Distância do pivo ao centro de massa da roda
Lh4 = 0.2272;%m-Distância do pivo ao centro de massa do contrapeso
Lch1 = 0.125;%m-Distância do pivo ao centro de massa da haste
Lch2 = 0.125;
Mc  = 0.350; %kg-Massa do contrapeso
Ip = ((1/3) * Mh1 * Lh1^2) + ((1/3) * Mh2 * Lh2^2) + Mtr*Lh3^2 + ...
   Mc*Lh4^2 + Ir; %-- kg m^2 - Momento de inércia do pêndulo

%-- simplificação
M1 = Mh1*Lch1 + Mtr*Lh3;
M2 = Mh2*Lch2 + Mc *Lh4;
%-- simplificação constante de inércia
Ci = (Ir*Ip)-(Ir^2);
g = 9.81;     %m/(s^2)
%-- motor
Am = 0.0014; %(Nm)/(rad/s)-coeficiente de atrito do motor
Kt = 0.0181; %(Nm)/A-constante de torque
n = 5.2;     %-redução do motor
Kv = 0.0181; %V/(rad/s)-constante de campo
Ra = 1.6632; %ohms-resistência de armadura

%% Modelo do Sistema Linearizado - considerando
```

```
%  xe=[0; 0; 0] ponto de equilíbrio
A = [ 0                    1                      0              ;
     (Ir*M1*g-Ir*M2*g)/Ci  0   (Ir*(n^2)*Kt*Kv+Ir*Am*Ra)/(Ci*Ra);
     (Ir*M2*g-Ir*M1*g)/Ci  0   (-Ip*(n^2)*Kt*Kv-Ip*Am*Ra)/(Ci*Ra)]

B = [0; -(Ir*n*Kt)/(Ci*Ra); (Ip*n*Kt)/(Ci*Ra)]

C = [1 0 0;
     0 1 0;
     0 0 1]
D = [zeros(3,1)]
%% Pólos em malha aberta
p_ma = eig(A)

%% Controlabilidade e Observabilidade
Contr = ctrb(A, B);
%det diferente de zero para que o par(A, B) seja controlável
if(rank(Contr) == size(A,1) && (det(Contr)~= 0))
    disp('--> O Modelo é Controlável')
else
    disp('Não é Controlável')
end
%% Pólos do Controlador (desejado)
pc = [-2210.37  -8.67  -1.301];
kc = place(A,B,pc)
```

A.3 DIAGRAMA DE BLOCOS DO SISTEMA COM CONTROLE CMD

A simulação do controlador CMD para estabilização do pêndulo PIRR, foi realizada considerando o diagrama da Figura 47.

Figura 47 – Diagrama do PIRR com controle CMD no SIMULINK®.



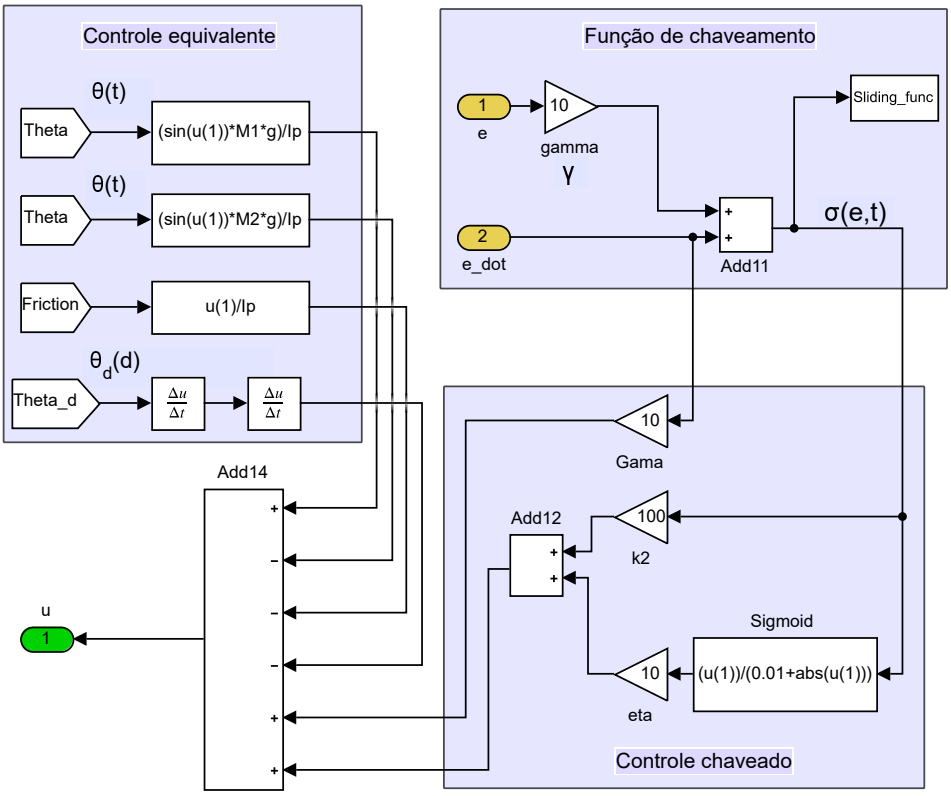
Fonte: próprio autor.

A seguir é apresentado o código do bloco *MATLAB function* identificado como *Estado Delta*.

```
%Estado Delta
function y = fcn(q1)
%#codegen
q2 = mod(q1 + pi, 2*pi);
if q2 < 0
q2 = q2 + 2*pi;
end

y = q2 - pi;
```

Na Figura 48, é apresentado o diagrama da lei de controle CMD considerado no diagrama da Figura 47.

Figura 48 – Diagrama da lei de controle CMD.

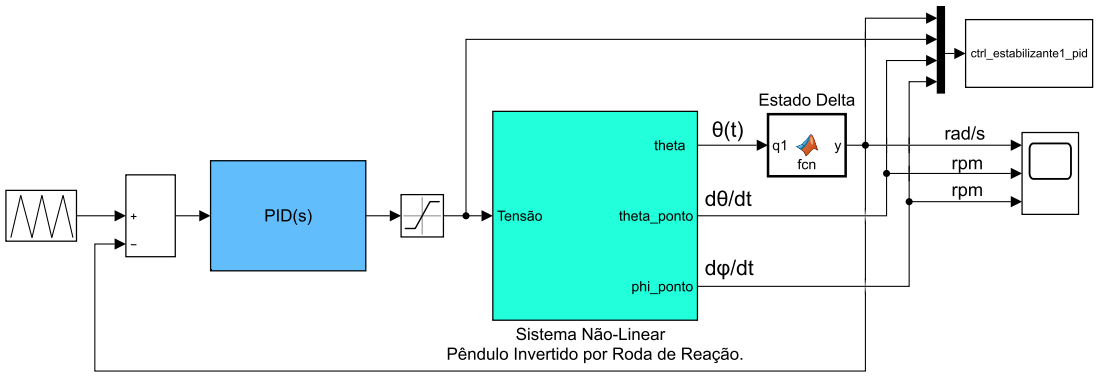


Fonte: próprio autor.

A.4 DIAGRAMA DE BLOCOS DO SISTEMA COM CONTROLE PID

A simulação do controlador PID para estabilização do pêndulo, foi realizada considerando o diagrama da Figura 49.

Figura 49 – Diagrama do PIRR com controle PID no SIMULINK®.



Fonte: próprio autor.

A seguir é apresentado o código do bloco *MATLAB function* nomeado *Estado Delta*.

```
1  %Estado Delta
2  function y = fcn(q1)
3  %#codegen
4  q2 = mod(q1 + pi, 2*pi);
5  if q2 < 0
6  q2 = q2 + 2*pi;
7  end
8
9  y = q2 - pi;
```

APÊNDICE B – CÓDIGOS IMPLEMENTADOS

Neste apêndice são apresentados os códigos dos controladores projetados para controle do sistema PIRR.

B.1 CÓDIGO DO CONTROLADOR POR ALOCAÇÃO DE AUTOVALORES

Nesta subseção é apresentado o algoritmo de controle por alocação de autovalores implementado na IDE Arduino utilizando o microcontrolador ESP32.

```
/*
--Pêndulo Invertido por Roda de Reação
--Discente: Rafael Máximo
*/
#include <ESP32Encoder.h>
//--- definindo os canais dos Encoder ---
#define chA_P 16  // canal A do Pêndulo
#define chB_P 17  // canal B do Pêndulo
#define chA_W 22  // canal A da Roda
#define chB_W 23  // canal B da Roda

//--- Encoder do PÊNDULO ------------------
float finalPositionPend_rad = 0.0;
float PositionPend_rad = 0.0;
float PositionPendDeg_mat;
float initialPositionPend_rad = 0.0;
long finalPositionPend_ppr;
long initialPositionPend_ppr;
float speedPendulum_rad = 0.0;
float velFiltered_P = 0.0;
float velFiltered_Pn = 0.0;
float velFiltered_P_rpm = 0.0;

unsigned long dt_P = 0;//velocidade do pêndulo
unsigned long prevTime_P = 0;
int elapsedTimeVel_P = 5;//5 ms
float dtheta;

//--- Encoder da RODA ---------------------
```

```
30 float finalPositionWheel_rad = 0.0;
31 float initialPositionWheel_rad = 0.0;
32 long finalPositionWheel_ppr;
33 long initialPositionWheel_ppr;
34 float speedWheel_rad;
35 float velFiltered_W = 0.0;
36 float velFiltered_Wn = 0.0;
37 float velFiltered_W_rpm = 0.0;
38
39 unsigned long dt_W;//velocidade do pêndulo
40 unsigned long prevTime_W;
41 int elapsedTimeVel_W = 5;//5 ms
42 float dphi;
43
44 //--- Objeto p/ obter rotações do eixo do Pêndulo ---
45 ESP32Encoder encoder_P;
46 ESP32Encoder encoder_W;
47
48 //--- Variáveis para o Timer do loop de controle ---
49 int elapsedTime_ctrl = 5;
50 unsigned long currentTime = 0, prevTime_ctrl = 0, ...
      timeDiff_ctrl = 0;
51
52 //--- Configurando as propriedades do PWM ---
53 const int RPWM_out = 26;
54 const int LPWM_out = 27;
55 const int pwmChannel_R = 0;
56 const int pwmChannel_L = 1;
57 const int pwmFreq = 19500;
58 const int pwmResolution = 12;
59
60 //--- Controlador por realimentação de estados
61 float K_pos_P, K_vel_P, K_vel_W;
62 float u_ctr, u_ctr1;
63 int deadBand = 1000;//zona morta do motor
64
65 //--- Controlador swing up ---
66 float K_sw;
67 float u_sw, u_sw1;
68 float u_sw_mat;
69 float u_ctr_mat;
70 float volt_pwm_mat;
```

```
71
72 //--- Timer para coleta de dados no Matlab ---
73 unsigned long dt_mat = 0;
74 unsigned long prevTime_mat = 0;
75 int elapsedTime_mat = 25;
76
77 //###### void setup() ##############
78 void setup()
79 {
80   //--- Uso dos pinos para canal A e B ---
81   encoder_P.attachHalfQuad(chA_P, chB_P);
82   encoder_W.attachHalfQuad(chA_W, chB_W);
83
84   //--- Posições iniciais para obter a velocidade ---
85   encoder_P.attachHalfQuad(chA_P, chB_P);
86   initialPositionPend_ppr = encoder_P.getCount();
87
88   encoder_W.attachHalfQuad ( chA_W, chB_W );
89   initialPositionWheel_ppr = encoder_W.getCount();
90
91   //--- Pinos GPIO PWM como saída ---
92   pinMode(RPWM_out, OUTPUT);
93   pinMode(LPWM_out, OUTPUT);
94
95   //--- Configura o canal para operar com frequência e ...
       resolução estabelecidas
96   ledcSetup(pwmChannel_R, pwmFreq, pwmResolution);
97   ledcSetup(pwmChannel_L, pwmFreq, pwmResolution);
98
99   //--- Anexa o pino GPIO PWM especificado ao canal PWM
100  ledcAttachPin(RPWM_out, pwmChannel_R);
101  ledcAttachPin(LPWM_out, pwmChannel_L);
102
103  Serial.begin(115200);
104 }
105 //###### void setup() //##############
106
107
108 //########## void loop() //##########
109 void loop()
110 {
111   //--- Timer para o loop de controle -------
```

```
112   currentTime = millis();
113   timeDiff_ctrl = (currentTime - prevTime_ctrl);
114
115
116   if (timeDiff_ctrl >= elapsedTime_ctrl)
117   {
118
119     //--- Posição angular do pêndulo e roda em PPR- Pulsos Por ...
          Revolução ---
120     finalPositionPend_ppr = encoder_P.getCount();
121     finalPositionWheel_ppr = encoder_W.getCount();
122
123     PositionPend_rad = ((float)finalPositionPend_ppr / 2000.0) ...
          * (2*PI ) + PI;//rad
124     finalPositionPend_rad = constrainAngle_P(PositionPend_rad);
125
126     //converte para graus
127     PositionPendDeg_mat = PositionPend_rad * 180 / PI;//graus
128
129     //converte a posição da roda para radianos
130     finalPositionWheel_rad = ((float)finalPositionWheel_ppr / ...
          177.0) * (2*PI );
131
132     velFiltered_P = getVelocity_P(finalPositionPend_rad, ...
          currentTime);//rad/s
133     velFiltered_W = getVelocity_W(finalPositionWheel_rad, ...
          currentTime);//rad/s
134
135     velFiltered_P_rpm = velFiltered_P * (60/(2*PI));//RPM
136     velFiltered_W_rpm = velFiltered_W * (60/(2*PI));//RPM
137
138     //--- Controle de estabilização ----------------
139     if(abs(finalPositionPend_rad) < 0.20)
140     {
141       K_pos_P = - 14.4744 *10000.0;
142       K_vel_P = - 2.3056  *10000.0;
143       K_vel_W = - 0.0465  *10000.0;
144
145       //--- Realimentação de estados (u = - [-k*x])
146       u_ctr  = - (K_pos_P * finalPositionPend_rad + K_vel_P * ...
            velFiltered_P + K_vel_W * velFiltered_W);
147       u_ctr1 = u_ctr;
```

```
148
149
150       u_ctr = constrain(abs(u_ctr), 0, 4095);
151       u_ctr_mat = u_ctr;//Matlab
152       u_ctr = map(u_ctr, 0, 4095, deadBand, 4095);//pwm
153
154       //converte em Volts para supervisão no Matlab
155       u_ctr_mat = u_ctr_mat *  (12.0 / 4095.0);//Matlab
156
157       if( u_ctr1 < 0)
158       {
159         digitalWrite(LPWM_out, LOW);
160         ledcWrite(pwmChannel_R, u_ctr);
161         volt_pwm_mat = u_ctr_mat;//Matlab
162       }
163
164       if(u_ctr1 > 0)
165       {
166         digitalWrite(RPWM_out, LOW);
167         ledcWrite(pwmChannel_L, u_ctr);
168         volt_pwm_mat = - u_ctr_mat;//Matlab
169       }
170
171     }
172
173     //--- Controle swing up -----------------------
174     if(abs(finalPositionPend_rad) > 0.20)
175     {
176
177       K_sw = 18.5;
178       u_sw = - K_sw * pow(deltaSwing(PositionPend_rad), 5.2) * ...
              sgnSpeedPend(velFiltered_P) * ...
              sgnPosPend(PositionPend_rad) ;
179       u_sw1 = u_sw;
180
181
182       u_sw = constrain(abs(u_sw), 0, 4095);     //pwm
183       u_sw_mat = u_sw;//Matlab
184       u_sw = map(u_sw, 0, 4095, deadBand, 4095);//pwm
185
186       //converte em Volts para supervisão no Matlab
187       u_sw_mat = u_sw_mat *  (12.0 / 4095.0);//Matlab
```

```
188
189       if( u_sw1 > 0)
190       {
191         digitalWrite(LPWM_out, LOW);
192         ledcWrite(pwmChannel_R, u_sw);
193         volt_pwm_mat = u_sw_mat;//Matlab
194       }
195
196       if(u_sw1 < 0)
197       {
198         digitalWrite(RPWM_out, LOW);
199         ledcWrite(pwmChannel_L, u_sw);
200         volt_pwm_mat = - u_sw_mat;//Matlab
201       }
202
203     }
204
205     prevTime_ctrl = currentTime;// reset do timer de loop de ...
          controle
206   }
207   //--- Timer para o loop de controle -----------------------
208
209
210   //--- Timer para coleta de dados no MATLAB----------------
211   dt_mat = (currentTime - prevTime_mat);
212   if (dt_mat >= elapsedTime_mat)
213   {
214     //========= Aquisição de Dados no MATLAB //==========
215     Serial.print(PositionPendDeg_mat);
216     Serial.print(",");
217     Serial.print(volt_pwm_mat);
218     Serial.print(",");
219     Serial.print(velFiltered_P_rpm);
220     Serial.print(",");
221     Serial.println(velFiltered_W_rpm);
222
223     prevTime_mat = currentTime;// reset do timer de Dados
224   }
225   //--- Timer para coleta de dados no MATLAB----------------
226
227 }
228 //################ void loop() #################
```

```
229
230 //---Funções
231
232 //--- Mantém o ângulo sempre dentro do intervalo [-PI PI]
233 float constrainAngle_P(float theta)
234 {
235   //soma PI ao ângulo de entrada theta e pega o resto da ...
        divisão por 2PI
236   theta = fmod(theta + PI, 2*PI);
237   if (theta < 0)
238   {
239     theta += 2*PI;
240   }
241   //ajustando o resultado para a faixa final de  [-pi, pi] ...
        radianos
242   return theta - PI;
243 }
244
245
246
247 float getVelocity_P(float finalPositionPend_rad, unsigned long ...
      currentTime)
248 {
249   //calcula dt em milisegundos do Pêndulo
250   dt_P = (currentTime - prevTime_P);
251   if (dt_P >= elapsedTimeVel_P)
252   {
253     //a menor diferença no círculo para evitar salto entre +Pi ...
          e -PI
254     dtheta= angleDiff(finalPositionPend_rad, ...
          initialPositionPend_rad);
255     initialPositionPend_rad = finalPositionPend_rad;
256     speedPendulum_rad = (dtheta) /(dt_P/1000.0);
257
258     //filtro p/ reduzir as frequências elevadas na vel. do pêndulo
259     velFiltered_P = 0.90909 * velFiltered_Pn + 0.09090 * ...
          speedPendulum_rad;
260     velFiltered_Pn = velFiltered_P;
261     prevTime_P = currentTime;//reset do timer p/ velocidade ...
          pêndulo
262   }
263   return velFiltered_P;
```

```
264 }
265
266 float getVelocity_W(float finalPositionWheel_rad, unsigned ...
      long currentTime )
267 {
268   //calcula dt em milisegundos do Pêndulo
269   dt_W = (currentTime - prevTime_W);
270   if (dt_W >= elapsedTimeVel_W)
271   {
272     dphi = (finalPositionWheel_rad - initialPositionWheel_rad);
273     speedWheel_rad =    (dphi) /(dt_W/1000.0);
274     initialPositionWheel_rad = finalPositionWheel_rad;
275
276     //filtro p/ reduzir as frequências elevadas na vel. da roda
277     velFiltered_W = 0.90909 * velFiltered_Wn + 0.09090 * ...
          speedWheel_rad;
278     velFiltered_Wn = velFiltered_W;
279     prevTime_W = currentTime;//reset do timer p/ velocidade da ...
          roda
280   }
281   return velFiltered_W;
282 }
283
284 //---diferença angular mínima no círculo--> resultado em [-pi, pi]
285 float angleDiff(float finalPositionPend_rad, float ...
      initialPositionPend_rad)
286 {
287   //menor diferença (no círculo) - dentro do intervalo  [-pi, pi]
288   float d = constrainAngle_P(finalPositionPend_rad - ...
        initialPositionPend_rad);
289   return d;
290 }
291
292 //---- Função sgn(tetha') para o Swing up ---
293 int sgnSpeedPend(float velFiltered_P)
294 {
295   if (velFiltered_P > 0.00000001)
296   {
297     return 1;
298   }
299   else if (velFiltered_P < -0.00000001)
300   {
```

```
301     return -1;
302   }
303   else
304   {
305     return 0;
306   }
307 }
308
309 //---- Função sgn(cos(tetha)) para o Swing up ---
310 int sgnPosPend(float finalPositionPend_rad)
311 {
312   if(cos(finalPositionPend_rad) > 0)
313   {
314     return 1;
315   }
316   else if(cos(finalPositionPend_rad) < 0)
317   {
318     return -1;
319   }
320   else
321   {
322     return 0;
323   }
324 }
325
326 float deltaSwing(float x)
327 {
328   x = fmod(x + PI, 2*PI);
329   if (x < 0)
330   {
331     x += 2*PI;
332   }
333
334   return abs(x - PI);
335 }
```

B.2 CÓDIGO DO CONTROLADOR PID

Nesta subseção é apresentado o algoritmo do controlador PID implementado na IDE Arduino.

```
1
2 /*
3 --Pêndulo Invertido por Roda de Reação
4 --Controlador PID
5 --Discente: Rafael Máximo da Silva
6 */
7 #include <ESP32Encoder.h>
8
9 //--- definindo os canais dos Encoder ---
10 #define chA_P 16  // canal A do Pêndulo
11 #define chB_P 17  // canal B do Pêndulo
12
13 #define chA_W 22  // canal A da Roda
14 #define chB_W 23  // canal B da Roda
15
16 //--- PÊNDULO ------------------
17 float finalPositionPend_rad = 0.0;
18 float PositionPend_rad = 0.0;
19 float PositionPendDeg_mat;
20 float initialPositionPend_rad = 0.0;
21 long finalPositionPend_ppr;
22 long initialPositionPend_ppr;
23 long zero_offset;
24
25 float speedPendulum_rad = 0.0;
26 float velFiltered_P = 0.0;
27 float velFiltered_Pn = 0.0;
28 float velFiltered_P_rpm;
29
30 unsigned long dt_P = 0;//velocidade do pêndulo
31 unsigned long prevTime_P = 0;
32 int elapsedTimeVel_P = 5;
33 float dtheta;
34
35 //--- RODA ---------------------
36 float finalPositionWheel_rad = 0.0;
37 float initialPositionWheel_rad = 0.0;
38 long finalPositionWheel_ppr;
39 long initialPositionWheel_ppr;
40
41 float speedWheel_rad;
```

```
42 float velFiltered_W = 0.0;
43 float velFiltered_Wn = 0.0;
44 float velFiltered_W_rpm;
45
46 unsigned long dt_W;//velocidade do pêndulo
47 unsigned long prevTime_W;
48 int elapsedTimeVel_W = 5;
49 float dphi;
50
51 //--- Objeto p/ obter rotações do eixo do Pêndulo ---
52 ESP32Encoder encoder_P;
53 ESP32Encoder encoder_W;
54
55 //--- Variáveis para o Timer do loop de controle ---
56 int elapsedTime_ctrl = 5;
57 unsigned long currentTime = 0, prevTime_ctrl = 0, ...
      timeDiff_ctrl = 0;
58
59 //--- Configurando as propriedades do PWM ---
60 const int RPWM_out  = 26;
61 const int LPWM_out  = 27;
62 const int pwmChannel_R = 0;
63 const int pwmChannel_L = 1;
64 const int pwmFreq = 19500;
65 const int pwmResolution = 12;
66
67 //--- Controlador PID
68 float K_p, K_i, K_d;
69 float erro_p;
70 float Setpoint = 0.0;
71 float accum = 0.0, motorVel = 0.0;
72
73 float sum_integral=0.0, integral=0.0;
74 float derivada=0.0, erro_anterior=0.0;
75
76 float u_ctr, u_ctr1;
77 int deadBand = 1000;//zona morta do motor
78
79 bool antiClockwise, Clockwise;
80
81 //--- Controlador swing up ---
82 float K_sw;
```

```
83 float u_sw, u_sw1;
84 float u_sw_mat;
85 float u_ctr_mat;
86 float volt_pwm_mat;
87
88 //--- Timer para coleta de dados no Matlab ---
89 unsigned long dt_mat = 0;
90 unsigned long prevTime_mat = 0;
91 int elapsedTime_mat = 25;
92
93
94 //############ void setup() //############
95 void setup()
96 {
97   //--- Uso dos pinos para canal A e B ---
98   encoder_P.attachHalfQuad(chA_P, chB_P);  //Pêndulo GPIO 16 17
99   encoder_W.attachHalfQuad(chA_W, chB_W);  //Roda GPIO 22 e 23
100
101   //--- Posições iniciais para obter a velocidade ---
102   encoder_P.attachHalfQuad(chA_P, chB_P);  //Pêndulo
103   initialPositionPend_ppr = encoder_P.getCount();//PPR
104
105   encoder_W.attachHalfQuad ( chA_W, chB_W ); //Roda
106   initialPositionWheel_ppr = encoder_W.getCount();//PPR
107
108   //--- Pinos GPIO PWM como saída ---
109   pinMode(RPWM_out, OUTPUT);
110   pinMode(LPWM_out, OUTPUT);
111
112   // -- Configura o canal para operar com frequência e ...
        resolução estabelecidas
113   ledcSetup(pwmChannel_R, pwmFreq, pwmResolution);
114   ledcSetup(pwmChannel_L, pwmFreq, pwmResolution);
115
116   //--- Anexa o pino GPIO PWM especificado ao canal PWM ...
        configurado
117   ledcAttachPin(RPWM_out, pwmChannel_R);
118   ledcAttachPin(LPWM_out, pwmChannel_L);
119
120   Serial.begin(115200);
121
122 }
```

```
123 //############ void setup() //############
124
125
126 //############ void loop() //#############
127 void loop()
128 {
129   //--- Timer para o loop de controle ---
130   currentTime = millis();
131   timeDiff_ctrl = (currentTime - prevTime_ctrl);
132
133   if (timeDiff_ctrl >= elapsedTime_ctrl)
134   {
135     //--- posição angular do pêndulo e roda em PPR- Pulsos Por ...
          Revolução ---
136     finalPositionPend_ppr = encoder_P.getCount();
137     finalPositionWheel_ppr = encoder_W.getCount();
138
139     PositionPend_rad = ((float)finalPositionPend_ppr / 2000.0) ...
          * (2*PI ) + PI;//rad
140     finalPositionPend_rad = constrainAngle_P(PositionPend_rad);
141
142     //converte para graus
143     PositionPendDeg_mat = PositionPend_rad * 180 / PI;
144
145     //converte a posição da roda para radianos
146     finalPositionWheel_rad = ((float)finalPositionWheel_ppr / ...
          177.0) * (2*PI );
147
148     velFiltered_P = getVelocity_P(finalPositionPend_rad, ...
          currentTime);//rad/s
149     velFiltered_W = getVelocity_W(finalPositionWheel_rad, ...
          currentTime);
150
151     velFiltered_P_rpm = velFiltered_P * (60/(2*PI));//RPM
152     velFiltered_W_rpm = velFiltered_W * (60/(2*PI));//RPM
153
154     //--- Controle de estabilização --------------------
155     if(abs(finalPositionPend_rad) < 0.20)
156     {
157
158       K_p = 200   * 1000.0;
159       K_i = 786   * 1000.0;
```

```
160       K_d = 0.5    * 1000.0;
161
162       if(Clockwise == true)
163       {
164         motorVel = motorVel + (velFiltered_W / 1000.0);
165         motorVel = constrain(motorVel, -1, 1);
166         accum = accum + motorVel / 10000.0;
167         accum = constrain(accum, 0, 0.0698132);//0 à 4graus ...
              --> ditherampli=2graus
168         Setpoint = 0.0174533 -accum;
169       }
170
171       if(antiClockwise == true)
172       {
173         motorVel = motorVel + (velFiltered_W / 1000.0);
174         motorVel = constrain(motorVel, -1, 1);
175         accum = accum + motorVel / 100000.0;
176         accum = constrain(accum, 0, 0.0872665);//0 à 5graus ...
              --> ditherampli=2graus
177         Setpoint = -0.00872665 +accum;
178       }
179
180       //sinal de erro
181       erro_p = Setpoint - finalPositionPend_rad;
182
183       //Componente da integral
184       sum_integral = erro_p * (timeDiff_ctrl/1000.0)  + ...
            sum_integral;
185       integral = sum_integral * K_i;
186       if(integral > 4095)
187       {
188         integral =  4095;
189       }
190       if(integral < - 4095)
191       {
192         integral = - 4095;
193       }
194
195       //Componente derivativa
196       derivada = (erro_p - erro_anterior ) / ...
            (timeDiff_ctrl/1000.0) * K_d;
197       erro_anterior = erro_p;
```

```
198       if(derivada > 4095)
199       {
200         derivada = 4095;
201       }
202       if(derivada < -4095)
203       {
204         derivada = -4095;
205       }
206
207       //---Controlador PID (Proporcional Integral Derivativo)
208       u_ctr  = erro_p * K_p + integral + derivada;;
209       u_ctr1 = u_ctr;
210
211
212       u_ctr = constrain(abs(u_ctr), 0, 4095);     //pwm
213       u_ctr_mat = u_ctr;//Matlab
214       u_ctr = map(u_ctr, 0, 4095, deadBand, 4095);//pwm
215
216       //converte em Volts para supervisão no Matlab
217       u_ctr_mat = u_ctr_mat *  (12.0 / 4095.0);//Matlab
218
219       if( u_ctr1 > 0)
220       {
221         digitalWrite(LPWM_out, LOW);
222         ledcWrite(pwmChannel_R, u_ctr);
223         volt_pwm_mat = u_ctr_mat;//Matlab
224       }
225
226       if(u_ctr1 < 0)
227       {
228         digitalWrite(RPWM_out, LOW);
229         ledcWrite(pwmChannel_L, u_ctr);
230         volt_pwm_mat = - u_ctr_mat;//Matlab
231       }
232
233     }
234
235     //--- Controle swing up -----------------------
236     if(abs(finalPositionPend_rad) > 0.20)
237     {
238       //---desligar swing u
239       // digitalWrite(LPWM_out, LOW);
```

```
240       // ledcWrite(pwmChannel_R, 0);
241       // digitalWrite(RPWM_out, LOW);
242       // ledcWrite(pwmChannel_L, 0);
243
244       if(velFiltered_P_rpm > 0)
245       {
246         antiClockwise = true;
247         Clockwise = false;
248       }
249       else
250       {
251         Clockwise = true;
252         antiClockwise = false;
253       }
254
255       K_sw = 10.3;
256       u_sw = - K_sw * pow(deltaSwing(PositionPend_rad), 5.2) * ...
             sgnSpeedPend(velFiltered_P) * ...
             sgnPosPend(finalPositionPend_rad);
257       u_sw1 = u_sw;
258
259       u_sw = constrain(abs(u_sw), 0, 4095);     //pwm
260       u_sw_mat = u_sw;//Matlab
261       u_sw = map(u_sw, 0, 4095, deadBand, 4095);//pwm
262
263       //converte em Volts para supervisão no Matlab
264       u_sw_mat = u_sw_mat *  (12.0 / 4095.0);//Matlab
265
266       if( u_sw1 > 0)
267       {
268         digitalWrite(LPWM_out, LOW);
269         ledcWrite(pwmChannel_R, u_sw);
270         volt_pwm_mat = - u_sw_mat;//Matlab - tem sinal ...
               invertido no mat
271       }
272
273       if(u_sw1 < 0)
274       {
275         digitalWrite(RPWM_out, LOW);
276         ledcWrite(pwmChannel_L, u_sw);
277         volt_pwm_mat =  u_sw_mat;//Matlab
278       }
```

```
279
280     }
281
282     prevTime_ctrl = currentTime;// reset do timer de loop de ...
            controle
283   }
284   //--- Timer para o loop de controle ------------
285
286
287   //--- Timer para coleta de dados no MATLAB-------
288   //calcula dt em milisegundos do
289   dt_mat = (currentTime - prevTime_mat);
290
291
292   if (dt_mat >= elapsedTime_mat)
293   {
294
295     //============ Aquisição de Dados no MATLAB //============
296     Serial.print(PositionPendDeg_mat);
297     Serial.print(",");
298     Serial.print(volt_pwm_mat);
299     Serial.print(",");
300     Serial.print(velFiltered_P_rpm);
301     Serial.print(",");
302     Serial.println(velFiltered_W_rpm);
303     prevTime_mat = currentTime;// reset do timer de Dados
304   }
305   //--- Timer para coleta de dados no MATLAB-----------------
306
307
308
309 }
310 //############ void loop() //#############
311
312
313 //--- Funções
314 //--- Garante que o ângulo calculado fique entre  -180 e 180 graus
315 float constrainAngle_P(float x)
316 {
317   x = fmod(x + PI, 2*PI);//soma PI ao ângulo de entrada x e ...
        pega o resto da divisão por 2PI
318   if (x < 0)
```

```
319   {
320     x += 2*PI;        //mantendo ângulo dentro da faixa [0, 2PI]
321   }
322   return x - PI;     //ajusta o resultado para a faixa final ...
        (-180graus a + 180graus)
323 }
324
325
326 float getVelocity_P(float finalPositionPend_rad, unsigned long ...
     currentTime)
327 {
328   //calcula dt em milisegundos do Pêndulo
329   dt_P = (currentTime - prevTime_P);
330   if (dt_P >= elapsedTimeVel_P)
331   {
332     //angleDiff() usa a menor diferença no círculo para evitar ...
        salto de +-PI
333     dtheta= angleDiff(finalPositionPend_rad, ...
        initialPositionPend_rad);
334     initialPositionPend_rad = finalPositionPend_rad;
335
336     speedPendulum_rad = (dtheta) /(dt_P/1000.0);
337
338     //filtro de primeira ordem p/ reduzir as frequências ...
        elevadas na vel. do pêndulo
339     velFiltered_P = 0.90909 * velFiltered_Pn + 0.09090 * ...
        speedPendulum_rad;
340     velFiltered_Pn = velFiltered_P;
341     prevTime_P = currentTime;//reset do timer
342   }
343   return velFiltered_P;
344 }
345
346 float getVelocity_W(float finalPositionWheel_rad, unsigned ...
     long currentTime )
347 {
348   //calcula dt em milisegundos do Pêndulo
349   dt_W = (currentTime - prevTime_W);
350
351   if (dt_W >= elapsedTimeVel_W)
352   {
353     dphi = (finalPositionWheel_rad - initialPositionWheel_rad);
```

```
354     speedWheel_rad =    (dphi) /(dt_W/1000.0);
355     initialPositionWheel_rad = finalPositionWheel_rad;
356
357     //filtro de primeira ordem p/ reduzir as frequências ...
           elevadas na vel. da roda
358     velFiltered_W = 0.90909 * velFiltered_Wn + 0.09090 * ...
           speedWheel_rad;
359     velFiltered_Wn = velFiltered_W;
360     prevTime_W = currentTime;//reset do timer
361   }
362   return velFiltered_W;
363 }
364
365 //---diferença angular mínima no círculo--> resultado em [-PI, ...
       +PI]
366 float angleDiff(float finalPositionPend_rad, float ...
       initialPositionPend_rad)
367 {
368   //menor diferença a - b (no círculo) - dentro do intervalo ...
         [-PI a +PI]
369   float d = constrainAngle_P(finalPositionPend_rad - ...
         initialPositionPend_rad);
370   return d;
371 }
372
373
374 //---- Função sgn(tetha') para o Swing up ---
375 int sgnSpeedPend(float velFiltered_P)
376 {
377   if (velFiltered_P > 0.00000001)
378   {
379     return 1;
380   }
381   else if (velFiltered_P < -0.00000001)
382   {
383     return -1;
384   }
385   else
386   {
387     return 0;
388   }
389 }
```

```
390
391 //---- Função sgn(cos(tetha)) para o Swing up ---
392 int sgnPosPend(float finalPositionPend_rad)
393 {
394   if(cos(finalPositionPend_rad) > 0)
395   {
396     return 1;
397   }
398   else if(cos(finalPositionPend_rad) < 0)
399   {
400     return -1;
401   }
402   else
403   {
404     return 0;
405   }
406 }
407
408 float deltaSwing(float x)
409 {
410   x = fmod(x + PI, 2*PI);
411   if (x < 0)
412   {
413     x += 2*PI;
414   }
415   //ajusta o resultado para a faixa final de |(+/-PI)| radianos
416   return abs(x - PI);
417 }
```

B.3 CÓDIGO DO CONTROLADOR CMD

Nesta seção é apresentado o algoritmo do controlador CMD implementado na IDE Arduino utilizando o microcontrolador ESP32.

```
1
2   /*
3   Pêndulo Invertido por Roda de Reação
4   Controlador CMD
5   Discente: Rafael Máximo da Silva
6   */
```

```
7   #include <ESP32Encoder.h>
8
9   //--- definindo os canais dos Encoder ---
10  #define chA_P 16  // canal A do Pêndulo
11  #define chB_P 17  // canal B do Pêndulo
12
13  #define chA_W 22  // canal A da Roda
14  #define chB_W 23  // canal B da Roda
15
16  //--- PÊNDULO ------------------
17  float finalPositionPend_rad = 0.0;
18  float PositionPend_rad = 0.0;
19  float PositionPendDeg_mat;
20  float initialPositionPend_rad = 0.0;
21  long finalPositionPend_ppr;
22  long initialPositionPend_ppr;
23
24  float speedPendulum_rad = 0.0;
25  float velFiltered_P = 0.0;
26  float velFiltered_Pn = 0.0;
27  float velFiltered_P_rpm;
28
29  unsigned long dt_P = 0;//velocidade do pêndulo
30  unsigned long prevTime_P = 0;
31  int elapsedTimeVel_P = 3;
32  float dtheta;
33
34  //--- RODA ---------------------
35  float finalPositionWheel_rad = 0.0;
36  float initialPositionWheel_rad = 0.0;
37  long finalPositionWheel_ppr;
38  long initialPositionWheel_ppr;
39
40  float speedWheel_rad;
41  float velFiltered_W = 0.0;
42  float velFiltered_Wn = 0.0;
43  float velFiltered_W_rpm;
44
45
46  unsigned long dt_W;//velocidade da roda
47  unsigned long prevTime_W;
48  int elapsedTimeVel_W = 3;
```

```
49  float dphi;
50
51
52  //--- Objeto p/ obter rotações do eixo do Pêndulo ---
53  ESP32Encoder encoder_P;
54  ESP32Encoder encoder_W;
55
56  //--- Variáveis para o Timer do loop de controle ---
57  int elapsedTime_ctrl = 3;
58  unsigned long currentTime = 0, prevTime_ctrl = 0, ...
       timeDiff_ctrl = 0;//loop de controle
59
60
61  //--- Configurando as propriedades do PWM ---
62  const int RPWM_out  = 26;
63  const int LPWM_out  = 27;
64  const int pwmChannel_R = 0;
65  const int pwmChannel_L = 1;
66  const int pwmFreq = 19500;
67  const int pwmResolution = 12;
68
69  //--- Variáveis Controlador CMD
70  float  M1=0.2079582, M2=0.1892608, g=9.81, Ip=0.091326;
71  float friction, fv=0.0012, fc=0.00041;
72  float gammma=10.0, eta=2000.0,  k=10.0;
73  float sigmoid, epsilon=0.1;
74
75  float sigma, erro_smd, dot_erro_smd = 0.0, erro_smd_anterior ...
       = 0.0, Setpoint = 0.0;
76  float u_ctr, u_ctr1;
77  float u_eq, u_n;
78  int deadBand = 1000;//zona morta do motor
79
80  //--- Timer para coleta de dados no Matlab ---
81  unsigned long dt_mat = 0;
82  unsigned long prevTime_mat = 0;
83  int elapsedTime_mat = 50;
84
85
86
87  //################## void setup() //##################
88  void setup()
```

```
89  {
90    //--- Uso dos pinos para canal A e B ---
91    encoder_P.attachHalfQuad(chA_P, chB_P);
92    encoder_W.attachHalfQuad(chA_W, chB_W);
93
94    //--- Posições iniciais para obter a velocidade ---
95    encoder_P.attachHalfQuad(chA_P, chB_P);  //Pêndulo
96    initialPositionPend_ppr = encoder_P.getCount();//PPR
97    encoder_W.attachHalfQuad ( chA_W, chB_W ); //Roda
98    initialPositionWheel_ppr = encoder_W.getCount();//PPR
99
100   //--- Pinos GPIO PWM como saída ---
101   pinMode(RPWM_out, OUTPUT);
102   pinMode(LPWM_out, OUTPUT);
103
104   // -- Configura o canal
105   ledcSetup(pwmChannel_R, pwmFreq, pwmResolution);
106   ledcSetup(pwmChannel_L, pwmFreq, pwmResolution);
107
108   //--- Anexa o pino GPIO PWM especificado ao canal
109   ledcAttachPin(RPWM_out, pwmChannel_R);
110   ledcAttachPin(LPWM_out, pwmChannel_L);
111
112   Serial.begin(115200);
113
114 }
115 //################## void setup() //###################
116
117
118 //################## void loop() //####################
119 void loop()
120 {
121   //--- Timer para o loop de controle ------------
122   currentTime = millis();
123   timeDiff_ctrl = (currentTime - prevTime_ctrl);
124
125   if (timeDiff_ctrl >= elapsedTime_ctrl)
126   {
127     //--- posição angular do pêndulo e roda em PPR- Pulsos ...
          Por Revolução ---
128     finalPositionPend_ppr = encoder_P.getCount();
129     finalPositionWheel_ppr = encoder_W.getCount();
```

```
130
131       PositionPend_rad = ((float)finalPositionPend_ppr / ...
              2000.0) * (2*PI ) + PI;//rad
132       finalPositionPend_rad = constrainAngle_P(PositionPend_rad);
133
134       //converte para graus
135       PositionPendDeg_mat = PositionPend_rad * 180 / PI;
136
137       //converte a posição da roda para radianos
138       finalPositionWheel_rad = ((float)finalPositionWheel_ppr ...
              / 177.0) * (2*PI );
139
140       velFiltered_P = getVelocity_P(finalPositionPend_rad, ...
              currentTime);//rad/s
141       velFiltered_W = getVelocity_W(finalPositionWheel_rad, ...
              currentTime);//rad/s
142
143       velFiltered_P_rpm = velFiltered_P * (60/(2*PI));//RPM
144       velFiltered_W_rpm = velFiltered_W * (60/(2*PI));//RPM
145
146       //--- Controle de estabilização --------------------
147       if(abs(finalPositionPend_rad) < 0.20)//11,45 graus
148       {
149         //erro de rastreamento
150         erro_smd =   finalPositionPend_rad - Setpoint;
151
152         //derivada do erro
153         dot_erro_smd = (erro_smd - erro_smd_anterior ) / ...
                (elapsedTime_ctrl/1000.0);
154         erro_smd_anterior = erro_smd;
155
156         //função de chaveamento
157         sigma = dot_erro_smd + gammma * erro_smd;
158
159         //equação considerando atrito de coloumb e viscoso
160         friction = sgnSpeedPend(velFiltered_P) * (fv * ...
              abs(velFiltered_P) + fc);
161
162         //função sigmóide
163         sigmoid = sigma/(abs(sigma)+epsilon);
164
165         //Controlador por Modos Deslizantes
```

```
166         u_eq = ((M1 * g * sin(finalPositionPend_rad)) / Ip ) - ...
                ((M2 * g * sin(finalPositionPend_rad)) / Ip) - ...
                (friction / Ip) + (gammma * dot_erro_smd);
167         u_n  = (eta * sigmoid) + k * sigma;
168
169         u_ctr = u_eq + u_n;
170         u_ctr1 = u_ctr;
171
172         u_ctr = constrain(abs(u_ctr), 0, 4095);      //pwm
173         u_ctr_mat = u_ctr;//Matlab
174         u_ctr = map(u_ctr, 0, 4095, deadBand, 4095);//pwm
175
176         //converte em Volts para supervisão no Matlab
177         u_ctr_mat = u_ctr_mat *  (12.0 / 4095.0);//Matlab
178
179         if( u_ctr1 < 0)
180         {
181           digitalWrite(LPWM_out, LOW);
182           ledcWrite(pwmChannel_R, u_ctr);
183           volt_pwm_mat = u_ctr_mat;//Matlab
184         }
185
186         if(u_ctr1 > 0)
187         {
188           digitalWrite(RPWM_out, LOW);
189           ledcWrite(pwmChannel_L, u_ctr);
190           volt_pwm_mat = - u_ctr_mat;//Matlab
191         }
192
193       }
194
195       prevTime_ctrl = currentTime;// reset do timer de loop de ...
             controle
196     }
197     //--- Timer para o loop de controle ---------------
198
199
200     //--- Timer para coleta de dados no MATLAB---------
201     dt_mat = (currentTime - prevTime_mat);
202
203     if (dt_mat >= elapsedTime_mat)
204     {
```

```
205       //============ Aquisição de Dados no MATLAB //====
206       Serial.print(PositionPendDeg_mat);
207       Serial.print(",");
208       Serial.print(volt_pwm_mat);
209       Serial.print(",");
210       Serial.print(velFiltered_P_rpm);
211       Serial.print(",");
212       Serial.println(velFiltered_W_rpm);
213
214       prevTime_mat = currentTime;
215     }
216     //--- Timer para coleta de dados no MATLAB---------
217
218   }
219   //################## void loop() //######################
220
221   //--- Funções
222   float constrainAngle_P(float x)
223   {
224     x = fmod(x + PI, 2*PI);
225     if (x < 0)
226     {
227       x += 2*PI;
228     }
229     return x - PI;
230   }
231
232
233   float getVelocity_P(float finalPositionPend_rad, unsigned ...
        long currentTime)
234   {
235     //calcula dt em milisegundos do Pêndulo
236     dt_P = (currentTime - prevTime_P);
237
238     if (dt_P >= elapsedTimeVel_P)
239     {
240       dtheta= angleDiff(finalPositionPend_rad, ...
             initialPositionPend_rad);
241       initialPositionPend_rad = finalPositionPend_rad;
242
243       speedPendulum_rad = (dtheta) /(dt_P/1000.0);
244
```

```
245       //filtro de primeira ordem p/ reduzir as frequências ...
              elevadas na vel. do pêndulo
246       velFiltered_P = 0.90909 * velFiltered_Pn + 0.09090 * ...
              speedPendulum_rad;
247       velFiltered_Pn = velFiltered_P;
248       prevTime_P = currentTime;//reset do timer p/ velocidade ...
              pêndulo
249     }
250     return velFiltered_P;
251   }
252
253
254   float getVelocity_W(float finalPositionWheel_rad, unsigned ...
          long currentTime )
255   {
256     //calcula dt em milisegundos do Pêndulo
257     dt_W = (currentTime - prevTime_W);
258
259     if (dt_W >= elapsedTimeVel_W)
260     {
261       dphi = (finalPositionWheel_rad - initialPositionWheel_rad);
262       speedWheel_rad =   (dphi) /(dt_W/1000.0);
263       initialPositionWheel_rad = finalPositionWheel_rad;
264
265       //filtro de primeira ordem p/ reduzir as frequências ...
              elevadas na vel. da roda
266       velFiltered_W = 0.90909 * velFiltered_Wn + 0.09090 * ...
              speedWheel_rad;
267       velFiltered_Wn = velFiltered_W;
268       prevTime_W = currentTime;
269     }
270     return velFiltered_W;
271   }
272
273   float angleDiff(float finalPositionPend_rad, float ...
          initialPositionPend_rad)
274   {
275     //menor diferença a - b (no círculo) - dentro do intervalo ...
            [-PI a +PI]
276     float d = constrainAngle_P(finalPositionPend_rad - ...
            initialPositionPend_rad);
277
```

```
278       return d;
279     }
```

APÊNDICE C – AQUISIÇÃO E REGISTROS DE DADOS

Neste apêndice é apresentado o código em MATLAB® para aquisição e registro de dados em tempo real, por meio da comunicação serial estabelecida entre o MATLAB® R2017b® e um microcontrolador ESP32.

O microcontrolador ESP32 transmite dados de sensores (encoder) e sinais de tensão (motor) relacionados ao sistema PIRR. Os dados transmitidos através do ESP32, são recebidos pelo MATLAB®, visualizados em tempo real por meio de gráficos dinâmicos e, ao final da execução, armazenados em um arquivo no formato CSV, possibilitando análises posteriores.

```
%%  Código MATLAB para plotagem de posição e tensão em tempo real

% === Configuração ===
port = 'COM6';      % <-- Substitua pela porta COM do seu ESP32
baudRate = 115200;
duration = 180;      % Duração em segundos
plotWindow = 10;    % Janela de plotagem em tempo real em seg


% === Configuração serial ===
s = serial(port, 'BaudRate', baudRate);
fopen(s);

% === Início da plotagem e registro ===
figure;
subplot(2,1,1);
h1 = animatedline('Color', 'b','LineWidth',2);
title('Posição do pêndulo');
xlabel('Tempo (s)'); ylabel('Posição (graus)');
grid on;

subplot(2,1,2);
h2 = animatedline('Color', 'r','LineWidth',2);
title('Sinal de controle Swing-UP');
xlabel('Tempo (s)'); ylabel('Tensão (V)');
grid on;


logT = [];         % Time vector
logPosition = []; % vetor da posição do pêndulo
logVoltage = [];  % vetor da tensão por PWM

startTime = datetime('now');
```

```
34
35      % === Loop de dados ===
36      while seconds(datetime('now') - startTime) < duration
37          try
38              data = fscanf(s); %Leitura do ESP32
39              values = str2num(strtrim(data));
40
41              if length(values) == 2
42                  t = seconds(datetime('now') - startTime);
43                  position = values(1);
44                  voltage = values(2);
45
46                  % Plot  % Add new points
47                  addpoints(h1, t, position);  % Live plot
48                  addpoints(h2, t, voltage);
49                  %drawnow limitrate;
50
51                  % Scroll X-axis (last `plotWindow` seconds)
52                  subplot(2,1,1); xlim([t - plotWindow, t]);
53                  subplot(2,1,2); xlim([t - plotWindow, t]);
54                  drawnow limitrate;
55
56                  % Log
57                  logT(end+1) = t;
58                  logPosition(end+1) = position;
59                  logVoltage(end+1) = voltage;
60
61              end
62          catch
63              %disp('Serial read error.');
64              disp('Error reading/parsing serial data.');
65          end
66      end
67
68      % === Limpeza ===
69      fclose(s);
70      delete(s);
71      clear s;
72
73      %% === Salvando para CSV ===
74      T = table(logT', logPosition', logVoltage', ...
75      'VariableNames', {'Time_sec', 'logPosition', 'logVoltage'});
76
77      filename = ['posicao_voltage_log_' datestr(now, ...
            'yyyymmdd_HHMMSS') '.csv'];
78      writetable(T, filename);
79      disp([' Multi-sensor data saved to: ', filename]);
```