UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

EDUARDO HIDEKI KANEKO

DESENVOLVIMENTO DE UM LABORATÓRIO VIRTUAL E REMOTO PARA CONTROLE DE UM HELICÓPTERO COM TRÊS GRAUS DE LIBERDADE

DISSERTAÇÃO

CORNÉLIO PROCÓPIO
2020

EDUARDO HIDEKI KANEKO

DESENVOLVIMENTO DE UM LABORATÓRIO VIRTUAL E REMOTO PARA CONTROLE DE UM HELICÓPTERO COM TRÊS GRAUS DE LIBERDADE

Dissertação apresentada ao Programa de Pós-graduação da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Mestre.

Área de concentração: Ciência Mecânicas

Linha de pesquisa: Sistemas Dinâmicos

Orientador: Prof. Dr. Adailton Silva Borges
Universidade Tecnológica Federal do Paraná (UTFPR), Câmpus Cornélio Procópio (CP)

Coorientador: Prof. Dr. Marcio Aurelio Furtado Montezuma
Universidade Tecnológica Federal do Paraná (UTFPR), Câmpus Cornélio Procópio (CP)

CORNÉLIO PROCÓPIO
2020

Dados Internacionais de Catalogação na Publicação

K163 Kaneko, Eduardo Hideki

Desenvolvimento de um laboratório virtual e remoto para controle de um helicóptero com três graus de liberdade / Eduardo Hideki Kaneko. – 2020.
154 p. : il. color. ; 31 cm.

Orientador: Adailton Silva Borges.
Coorientador: Marcio Aurelio Furtado Montezuma.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Mecânica. Cornélio Procópio, 2020.
Bibliografia: p. 147-154.

1. Modelagem. 2. Sistemas lineares de controle. 3. Laboratórios. 4. Engenharia Mecânica – Dissertações. I. Borges, Adailton Silva, orient. II. Montezuma, Marcio Aurelio Furtado, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Mecânica. IV. Título.

CDD (22. ed.) 620.1

Biblioteca da UTFPR - Câmpus Cornélio Procópio
Bibliotecário/Documentalista responsável:
Romeu Righetti de Araujo – CRB-9/1676

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Engenharia Mecânica

Título da Dissertação Nº 045:

"Desenvolvimento De Um Laboratório Virtual E Remoto Para Controle De Um Helicóptero Com Três Graus De Liberdade".

Por

Eduardo Hideki Kaneko

Orientador: **Prof. Dr. Adailton Silva Borges**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de **MESTRE EM ENGENHARIA MECÂNICA** – Área de Concentração: **Ciências Mecânicas,** linha de pesquisa: **Sistemas Dinâmicos**, pelo Programa de Pós-Graduação em Engenharia Mecânica – PPGEM – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 10h do dia 03 de ABRIL de 2020. O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:

Prof.Dr. Adailton Silva Borges (Orientador – UTFPR-CP)	Prof. Dr. Sandra Mara Domiciano (UTFPR-CP)

Prof. Dr. Luiz Henrique Geromel
(IFSP-Piracicaba)

Dedico este trabalho aos meus pais e minha irmã, por todo amor e por estarem sempre ao meu lado.

AGRADECIMENTOS

Agradeço primeiramente a Deus por tudo.

Agradeço aos meus pais, minhas fontes de exemplo e inspiração, por todo o amor, amizade, dedicação e esforço, que me permitiram chegar a esta etapa da minha vida.

Agradeço a minha irmã, meu exemplo nos estudos, pelo amor e amizade.

Agradeço ao Prof. Dr. Adailton Silva Borges, pela confiança ao aceitar ser o orientador do trabalho, e por todo conhecimento transmitido.

Agradeço ao Prof. Dr. Marcio Aurelio Furtado Montezuma, por toda orientação, ensinamentos, oportunidades de aprendizado e pela confiança ao me propor a realização deste trabalho.

Agradeço a todos os professores por toda dedicação para auxiliar na construção do conhecimento nestes últimos anos.

Agradeço ao Wanderlei Malaquias, por todos ensinamentos e auxílio na construção de muitos dos equipamentos utilizados na realização dos experimentos.

Agradeço ao Lucas Niro, por todos ensinamentos e pelo grande auxílio na produção dos artigos.

Agradeço ao Bruno Shimada, pelos trabalhos realizados no LaSisC, que possibilitaram a execução deste trabalho.

Agradeço ao Matheus, Wagner e Umberto, por todo apoio e auxílio no desenvolvimento deste trabalho.

Agradeço aos amigos do LaSisC, Matheus, Wagner, Niro, Malaquias, Shimada, Thiago, Umberto, Marcos, Gabriel, David, Blandown, Lillyane e tantos outros que passaram pelo laboratório, por toda diversidade de conversas e troca de experiências.

Agradeço a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de mestrado.

Agradeço a todos que de alguma maneira contribuíram para que este trabalho pudesse ser realizado.

RESUMO

KANEKO, Eduardo Hideki. Desenvolvimento de um Laboratório Virtual e Remoto para Controle de um Helicóptero com Três Graus de Liberdade. 2020. 154 f. Dissertação – Programa de Pós-Graduação em Engenharia Mecânica, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2020.

Esta dissertação apresenta o desenvolvimento de um laboratório virtual e remoto (LVR) para o controle de um Helicóptero com 3 graus de liberdade (GDL). Este é um sistema mecânico para o estudo da dinâmica de uma aeronave de hélices paralelas. Os três movimentos da planta são denominados arfagem, elevação e deslocamento, sendo o objetivo controlar estes dois últimos. O modelo matemático do Helicóptero com 3 GDL foi determinado pelo método de prototipagem virtual. As variáveis de saída do modelo matemático são as posições e velocidades angulares dos movimentos, estimadas por *Tracking Loops*, que utilizam os sinais discretos obtidos por meio de sensores do tipo *encoder*. As variáveis de controle do modelo matemático são as forças de empuxo, a identificação de um polinômio permite relacionar estas forças às tensões aplicadas aos motores de corrente contínua equipados com hélices de passo fixo, que atuam sobre a planta. O LVR, de modo geral, atua como um ambiente de desenvolvimento integrado. Este permite projetar, construir e monitorar controladores, por meio de um sistema de aquisição e controle composto pelas tecnologias Octave, Block Diagram Coder (BDC, um pacote de software desenvolvido neste trabalho) e a placa de microcontrolador NUCLEO-F767ZI, responsável pela leitura dos sensores e acionamento dos atuadores da planta. Para validar todo o desenvolvimento do trabalho, foram executados experimentos de controle seguidor com realimentação de estado, com e sem o observador de Luenberger. Os controladores foram projetados por LQR (*Linear Quadratic Regulator*) e atribuição de autoestrutura completa. Estes experimentos foram executados no LVR e também por meio de um sistema de aquisição e controle tradicional, composto pelo MATLAB/Simulink e uma placa de aquisição da National Instruments PCI-6602. A comparação e análise dos resultados permitiu concluir que o LVR funciona adequadamente para o desenvolvimento de experimentos de controle. Além disso, a reposta dos controladores para o Helicóptero com 3 GDL apresentou bons resultados, validando os processos de modelagem matemática, estimação das variáveis de saída e identificação dos atuadores.

Palavras-chave: Modelagem Matemática. Sistemas de Controle. Laboratório Virtual e Remoto.

ABSTRACT

KANEKO, Eduardo Hideki. Development of a Virtual and Remote Laboratory for the Control of a Three Degrees of Freedom Helicopter. 2020. 154 f. Dissertação – Programa de Pós-Graduação em Engenharia Mecânica, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2020.

This thesis presents the development of a virtual and remote laboratory (VRL) for the control of a 3 degrees of freedom (DOF) Helicopter. This is a mechanical system to study the dynamics of an aircraft with parallel propellers. The three movements of the plant are called pitch, elevation and travel, and the objective is to control the latter two. The mathematical model of the 3 DOF helicopter was determined by the virtual prototyping method. The output variables of the mathematical model are the angular positions and velocities of the movements, estimated by Tracking Loops, which use the discrete signals obtained by encoder sensors. The control variables of the mathematical model are the thrust forces, the identification of a polynomial allows to relate these forces to the voltages applied to the direct current motors equipped with fixed pitch propellers, that actuates on the plant. The VRL, in general, acts as an integrated development environment. It allows the design, construction, and monitoring of controllers, through an acquisition and control system composed of the technologies Octave, Block Diagram Coder (BDC, a software package developed in this work) and the NUCLEO-F767ZI microcontroller board, responsible for reading the sensors and driving the plant actuators. In order to validate all the work development, control experiments of tracking systems with state feedback were performed, with and without the Luenberger observer. The controllers were designed by LQR (Linear Quadratic Regulator) and entire eigenstructure assignment. These experiments were performed at the VRL and also through a traditional acquisition and control system, composed of MATLAB/Simulink and a National Instruments PCI-6602 acquisition board. The comparison and analysis of the results concluded that the VRL works properly for the development of control experiments. Moreover, the response of the controllers for the 3 DOF helicopter showed good results, validating the processes of mathematical modeling, output variables estimation and actuators identification.

Keywords: Mathematical Modeling. Control Systems. Virtual and Remote Laboratory.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

3D	Tridimensional
Adams	*Advanced Dynamic Analysis of Mechanical Systems*
API	*Aplication Programming Interface*
BDC	Block Diagram Coder
bps	*bits per second*
CAD	*Computer Aided Design*
CAE	*Computer Aided Engineering*
CLI	*Command-Line Interface*
CSRF	*Cross Site Request Forgery*
CSS	*Cascading Style Sheets*
FPU	*Floating Point Unit*
GDL	Graus de Liberdade
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IETF	Internet Engineering Task Force
IoT	*Internet of Things*
ITAE	*Integral of the Time-Weighted Absolute Error*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
LQR	*Linear Quadratic Regulator*
LTS	*Long Term Support*
LVR	Laboratório Virtual e Remoto
MBS	*Multibody Systems*

MIMO	*Multiple-Input and Multiple-Output*
MVC	*Model-View-Controller*
MVVM	*Model-View-ViewModel*
MVW	*Model-View-Whatever*
NPM	*Node Package Manager*
NRMSE	*Normalized Root Mean Square Error*
OS	*Operating System*
PID	Proporcional, Integral e Derivativo
PWM	*Pulse Width Modulation*
RTOS	*Real-Time Operating System*
SBC	*Single Board Computer*
SISO	*Single-Input and Single-Output*
SPA	*Single-Page Application*
TL	*Tracking Loop*
UART	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*
VANT	Veículo Aéreo Não Tripulado
W3C	World Wide Web Consortium

LISTA DE SÍMBOLOS

$\mathbf{0}$	Matriz de zeros
$\boldsymbol{A}$	Matriz de estado de um sistema invariante no tempo
$\boldsymbol{A}(t)$	Matriz de estado de um sistema variante no tempo
$\bar{\boldsymbol{A}}$	Matriz de estado aumentada
$\boldsymbol{A_{cl}}$	Matriz de estado do sistema em malha fechada
$\boldsymbol{A'_{cl}}$	Matriz de estado do sistema aumentado do controle seguidor, em malha fechada
$\boldsymbol{B}$	Matriz de entrada de um sistema invariante no tempo
$\boldsymbol{B}(t)$	Matriz de entrada de um sistema variante no tempo
$\boldsymbol{B'}$	Matriz de entrada do sistema aumentado do controle seguidor, em malha fechada
$\bar{\boldsymbol{B}}$	Matriz de entrada aumentada
$\bar{\boldsymbol{B}}'$	Matriz de entrada de referência aumentada
$\boldsymbol{C}$	Matriz de saída de um sistema invariante no tempo
$\boldsymbol{C}(t)$	Matriz de saída de um sistema variante no tempo
$\bar{\boldsymbol{C}}$	Matriz de saída aumentada
$\boldsymbol{D}$	Matriz de transmissão direta de um sistema invariante no tempo
$\boldsymbol{D}(t)$	Matriz de transmissão direta de um sistema variante no tempo
$\boldsymbol{E}$	Matriz de seleção das variáveis de saída controladas
$\boldsymbol{e}(t)$	Vetor de erro de reconstrução do observador
$\dot{\boldsymbol{e}}(t)$	Derivada temporal do vetor de erro de reconstrução do observador
$e(t)$	Sinal de erro do controle de um processo SISO
e_p	Erro entre posição angular real e posição angular estimada pelo *Tracking Loop*
$\boldsymbol{F}$	Matriz de seleção das variáveis de saída não controladas

$\boldsymbol{f}$	Função matricial da equação de estado não-linear e variante no tempo
$G(s)$	Representação de uma função transferência
$\boldsymbol{g}$	Função matricial da equação de saída não-linear e variante no tempo
$\boldsymbol{I}$	Matriz identidade
J	Índice de desempenho quadrático
$\boldsymbol{K}$	Matriz de ganho de realimentação de estado para um sistema linear MIMO em malha aberta
$\bar{\boldsymbol{K}}$	Matriz de ganho da realimentação de estado do sistema aumentado do controle seguidor
$\boldsymbol{K_1}$	Matriz de ganho de realimentação de estado para um sistema linear MIMO em malha aberta
$\boldsymbol{K_2}$	Matriz de ganho relacionada a integral do erro da realimentação das variáveis de saída controladas
K	Ganho proporcional do controlador PID
K_I	Ganho integral do controlador PI do *Tracking Loop*
K_P	Ganho proporcional do controlador PI do *Tracking Loop*
$ker\boldsymbol{S}(\lambda_i)$	*Kernel* ou espaço nulo de $\boldsymbol{S}(\lambda_i)$
$\boldsymbol{L}$	Matriz do observador
l	Número de variáveis de saída
m	Número de variáveis de controle
n	Número de variáveis de estado
$\boldsymbol{P}$	Matriz auxiliar
p	Número de variáveis de saída controladas
p_e	Posição angular real obtida do sensor *encoder*
$\hat{p}_e$	Posição angular estimada pelo *Tracking Loop*
$\boldsymbol{Q}$	Na atribuição de autoestrutura completa é a matriz formada pelos vetores $\boldsymbol{q_i}$, no LQR é a matriz de ponderação dos estados

$\boldsymbol{q_i}$	Vetor obtido da relação entre o ganho de realimentação de estado $\boldsymbol{K}$ e o autovetor $\boldsymbol{v_i}$
$\boldsymbol{R}$	Matriz de ponderação do controle
$\boldsymbol{r}(t)$	Vetor de referência
$r(t)$	Sinal de referência do controle de um processo SISO
r_1	Referência de controle do movimento de deslocamento
r_2	Referência de controle do movimento de elevação
$\boldsymbol{S}$	Matriz para selecionar as variáveis de saída controladas pelo seguidor com realimentação de estado, partindo do vetor de saída
$\boldsymbol{S1}$	Matriz para selecionar as variáveis de saída controladas pelo seguidor com realimentação de estado, partindo do vetor de saída
$\boldsymbol{S2}$	Matriz para selecionar as variáveis de saída comparadas pelo observador de Luenberger, partindo do vetor de saída
$\boldsymbol{S}(\lambda_i)$	Matriz obtida da relação entre os autovalores de malha fechada λ_i e os autovetores associados $\boldsymbol{v_i}$
s	Variável complexa
T_d	Tempo derivativo do controlador PID
T_i	Tempo integral do controlador PID
T_s	Tempo de amostragem fixado para um diagrama de blocos do Block Diagram Coder
t	Instante de tempo
t_0	Instante de tempo inicial
t_n	Passo de tempo atual
$\boldsymbol{u}(t)$	Vetor de controle
$\boldsymbol{u}(t_n)$	Entrada atual dos blocos Discrete Derivative e Discrete-Time Integrator
$\boldsymbol{u}(t_{n-1})$	Entrada anterior dos blocos Discrete Derivative e Discrete-Time Integrator
$u(t)$	Sinal de entrada ou controle de um processo SISO
u_b	Sinal de entrada ou controle de um processo SISO quando $e = 0$

$\boldsymbol{V}$	Matriz formada pelos autovetores associados $\boldsymbol{v_i}$
$\boldsymbol{v}$	Conjunto de autovetores associados
$\boldsymbol{v_i}$	Autovetor associado
$\hat{v}_e$	Primeira estimativa de velocidade angular do *Tracking Loop*
$\hat{v}_{e2}$	Segunda estimativa de velocidade angular do *Tracking Loop*
$\boldsymbol{w}(t)$	Vetor de saída controlada
$\boldsymbol{x}(t)$	Vetor de estado
$\dot{\boldsymbol{x}}(t)$	Derivada temporal do vetor de estado
$\boldsymbol{x}'(t)$	Vetor de estado aumentado
$\dot{\boldsymbol{x}}'(t)$	Derivada temporal do vetor de estado aumentado
$\hat{\boldsymbol{x}}(t)$	Vetor de estado estimado
$\dot{\hat{\boldsymbol{x}}}(t)$	Derivada temporal do vetor de estado estimado
x	Sinal comparado pelo NRMSE
$xref$	Sinal de referência do NRMSE
$\boldsymbol{y}(t)$	Vetor de saída
$\boldsymbol{y}(t_n)$	Saída atual dos blocos Discrete Derivative e Discrete-Time Integrator
$\boldsymbol{y}(t_{n-1})$	Saída anterior do bloco Discrete-Time Integrator
$\hat{\boldsymbol{y}}(t)$	Vetor de saída estimado
$y(t)$	Sinal de saída de um processo SISO
$\boldsymbol{z}(t)$	Integral do vetor de erro da realimentação das variáveis de saída controladas
$\dot{\boldsymbol{z}}(t)$	Vetor de erro da realimentação das variáveis de saída controladas
λ_i	Autovalor de malha fechada
$\boldsymbol{\sigma}$	Espectro de autovalores de malha fechada
τ	Variável de integração do sinal de erro do controle de um processo SISO

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de controle automático são essenciais para a engenharia e ciência, sendo parte integrante da sociedade moderna e tendo papel relevante no progresso e desenvolvimento desta (OGATA, 2010; NISE, 2012; D'AZZO; HOUPIS; SHELDON, 2003). Um sistema de controle consiste na interconexão de subsistemas e processos a fim de formar uma configuração de sistema que proporcione, para uma entrada específica, uma saída desejada com um desempenho desejado (DORF; BISHOP, 2013; NISE, 2012).

De acordo com Nise (2012), quatro são as razões principais para a construção de sistemas de controle:

1. Amplificação de potência;
2. Conveniência da forma de entrada;
3. Controle remoto;
4. Compensação de perturbações.

Por meio dos sistemas de controle é possível mover equipamento pesado com precisão, sendo esta uma tarefa difícil de ser executada apenas pelo homem. O sistema de controle de um elevador, por exemplo, tem como entrada a posição de um andar desejado. O sistema de controle desloca o elevador até a posição desejada utilizando motores, que fornecem a potência necessária para a carga do elevador. A conveniência, proporcionada pelos sistemas de controle, por meio da alteração da forma de entrada é exemplificada pelo uso da posição de um termostato como entrada para um sistema de controle de temperatura (NISE, 2012).

O controle de sistemas em localidades remotas ou perigosas pode ter sua utilidade exemplificada por um braço robótico, controlado remotamente, para coleta de materiais em ambiente radioativo. Outra vantagem dos sistemas de controle é a capacidade de fornecer a saída desejada, mesmo na presença de perturbações externas ao sistema. O sistema de controle de direção de uma antena, por exemplo, tem como entrada a direção em que a antena deve captar algum sinal. Como saída o sistema de controle deve garantir que a antena mantenha-se na direção comandada, mesmo que exista algum fator externo (por exemplo, rajadas de vento) agindo sobre a antena (NISE, 2012).

Atualmente, um amplo campo para a aplicação de sistemas de controle é a orientação, navegação e controle de veículos espacias e aviões, bem como veículos aéreos não tripulados (VANTs). Estes normalmente são controlados por operadores em solo, constituindo assim uma forma de controle remoto (NISE, 2012; DORF; BISHOP, 2013). Por não apresentarem o mesmo nível de segurança de aeronaves tripuladas, os VANTs não tem permissão de voar livremente no espaço aéreo comercial. Desta forma, os VANTs são aplicados em trabalhos de levantamento aéreo em projetos de construção e monitoramentos meteorológicos e de

plantações. Militarmente, os VANTs são empregados em missões de inteligência, vigilância e reconhecimento (DORF; BISHOP, 2013).

Considerando a importância dos sistemas de controle na sociedade moderna, Kheir *et al.* (1996) e Ogata (2010) afirmam ser desejável que engenheiros e cientistas tenham o conhecimento dos conceitos matemáticos por trás da teoria de sistemas de controle, assim como a experiência na implementação das soluções teóricas em sistemas reais. A teoria de sistemas de controle depende de quatro conceitos abstratos principais, os sistemas dinâmicos, a estabilidade, a realimentação e a compensação dinâmica, sendo estes melhor representados de forma matemática. A experiência prática em sistemas de controle se desenvolve sobre a ideia de que algum sistema real deve ser controlado. Deste modo, um dos desafios do ensino de sistemas de controle é encontrar o equilíbrio entre a teoria e a prática (KHEIR *et al.*, 1996; HERADIO; TORRE; DORMIDO, 2016).

Na solução deste desafio, a experimentação em laboratório cumpre o importante papel de conectar a teoria com a prática, por meio dos seguintes objetivos (ANTSAKLIS *et al.*, 1999):

- Demonstrar, validar e motivar a análise de conceitos;
- Introduzir problemas reais da modelagem e controle, como saturação, ruído, dinâmica de sensores e atuadores, incertezas e outros;
- Proporcionar experiência no uso de ferramentas de instrumentação e medição;
- Proporcionar uma visão mais ampla dos problemas de projeto, como a necessidade de especificar a implementação de hardware e fazer considerações de economia;
- Expor estudantes a prática profissional de documentação de projeto e escrita de relatórios;
- Permitir a solução de problemas pelo trabalho em equipe;
- Permitir a comparação de resultados teóricos e experimentais, como validação dos conceitos teóricos.

Embora a experimentação em laboratório apresente todos estes benefícios, existe um alto custo associado a necessidade de equipamentos, espaço físico e equipe de manutenção. Neste contexto, no ano de 1998 a National Science Foundation em conjunto com a Control Systems Society, no Workshop sobre "Novas Direções na Educação em Engenharia de Controle", concluíram que o uso da Internet representa uma grande oportunidade na disseminação da educação em sistemas de controle. Deste modo, os laboratórios remotos foram considerados uma ferramenta valiosa por permitirem a experimentação em sistemas reais por meio da Internet, e aproveitarem ao máximo os custosos laboratórios de controle. Por conta disto, nos dias atuais as universidades tem demonstrado interesse na integração de laboratórios virtuais e remotos (LVRs) como forma de complementação do ensino tradicional (ANTSAKLIS *et al.*, 1999; GOMES; BOGOSYAN, 2009; HERADIO; TORRE; DORMIDO, 2016; HERADIO *et al.*, 2016; SÁENZ *et al.*, 2015; CHACÓN *et al.*, 2015b).

1.1 LABORATÓRIOS VIRTUAIS E REMOTOS

Em Dormido (2004) é sugerido um critério de classificação dos tipos de laboratório, de acordo com a natureza do recurso e a forma de acesso ao recurso, como apresentado no Quadro 1. A combinação destes critérios permite descrever de forma mais precisa os LVRs:

Quadro 1 – Critério de classificação dos tipos de laboratório.

		Natureza do Recurso	
		Real	**Simulado**
Acesso ao Recurso	**Local**	Laboratório Tradicional	Laboratório Virtual Mono-usuário
	Remoto	Laboratório Remoto	Laboratório Virtual Multi-usuário

Fonte: Adaptado de Dormido (2004).

- **Acesso local - Recurso real**: Nesta configuração um usuário, dentro da universidade, realiza experimentos em um sistema real conectado a um computador. Esta configuração representa os laboratórios tradicionais;
- **Acesso local - Recurso simulado**: Nesta configuração um usuário, dentro da universidade, realiza experimentos em um ambiente totalmente virtual, simulado em computador. Esta configuração representa um laboratório virtual acessado apenas por um usuário;
- **Acesso remoto - Recurso real**: Nesta configuração um usuário, em qualquer localidade remota, realiza experimentos em um sistema real utilizando uma interface Web, disponibilizada na Internet. Esta configuração representa os laboratórios remotos;
- **Acesso remoto - Recurso simulado**: Nesta configuração um usuário, em qualquer localidade remota, realiza a simulação do modelo matemático de um sistema utilizando uma interface Web, disponibilizada na Internet. Esta configuração representa um laboratório virtual, que por ser disponível na Internet, pode ser acessado por múltiplos usuários.

Os LVRs apresentam outros benefícios além de aproveitar ao máximo recursos de custo elevado. Um dos benefícios é a disponibilidade, os LVRs são capazes de atender igualmente usuários em diferentes localidades geográficas e fusos horários. Os LVRs também são acessíveis a

usuários com necessidades físicas especiais. Outro benefício é a observabilidade, os experimentos realizados podem ser transmitidos para múltiplos usuários, que por sua vez podem utilizar recursos para captura de tela no formato de vídeo. A segurança também é uma vantagem quando os experimentos realizados forem de natureza perigosa (GRAVIER *et al.*, 2008; HERADIO *et al.*, 2016).

Como os benefícios dos LVRs se estendem à diferentes áreas como engenharia de controle, robótica, mecatrônica, eletrônica, física e química, existe uma vasta produção acadêmica envolvendo os temas de educação e LVRs desde seus primórdios no ano de 1993 (HERADIO *et al.*, 2016; GOMES; BOGOSYAN, 2009). Trabalhos como Heradio *et al.* (2016) e Heradio, Torre e Dormido (2016) fazem uma revisão da extensa bibliografia de LVRs com o objetivo de identificar o seu papel educacional e detectar as tendências de pesquisa para o futuro.

A revisão bibliográfica de Heradio, Torre e Dormido (2016) avalia exclusivamente os trabalhos de LVRs em engenharia de controle, enquanto Heradio *et al.* (2016) avalia os LVRs independente de sua área de aplicação. Dentre as tendências de pesquisa para o futuro, pode-se destacar o desenvolvimento de formas eficientes de combinar laboratórios virtuais com laboratórios remotos, o aumento da reusabilidade dos LVRs e a criação de LVRs que possibilitem ao usuário a liberdade de implementar a própria técnica de controle.

1.1.1 Papel educacional

Embora existam trabalhos, como Brinson (2015), que provem empiricamente a capacidade dos LVRs em atingir resultados comparáveis aos laboratórios tradicionais, quanto a avaliação do aprendizado dos alunos, ambos não são alternativas exclusivas (HERADIO *et al.*, 2016; HERADIO; TORRE; DORMIDO, 2016). Segundo Heradio *et al.* (2016), os laboratórios tradicionais, virtuais e remotos podem ser integrados, formando uma unidade de aprendizado complementar e integral. Em Zacharia (2007) concluiu-se por experimentos, que o entendimento conceitual dos estudantes é maior quando se combina laboratórios virtuais com remotos, do que apenas utilizando laboratórios remotos. Neste contexto, Abdulwahed e Nagy (2011) propõe o uso integrado dos LVRs com laboratórios tradicionais na seguinte ordem:

1. Empregar os laboratórios virtuais em sessões de preparação;
2. Empregar os laboratórios tradicionais em aulas interativas com experimentação;
3. Empregar os laboratórios remotos na repetição da execução dos experimentos.

1.1.2 Combinação eficiente de laboratórios virtuais com laboratórios remotos

Tanto os laboratórios virtuais quanto os laboratórios remotos apresentam qualidades intrínsecas. Algumas das qualidades únicas dos laboratórios virtuais são as capacidades de (HERADIO *et al.*, 2016):

- Modificar ou simplificar sistemas do mundo real, tornando os fenômenos mais visíveis e adaptáveis para o aprendizado por diferentes níveis cognitivos;
- Executar uma grande quantidade de experimentos com maior rapidez e facilidade, proporcionando aos estudantes a análise imediata de erros e permitindo a reexecução dos experimentos;
- Ajudar estudantes na visualização de processos de difícil percepção;
- Realizar experimentos de grande dificuldade ou custo, quanto a execução em plantas reais.

Quanto aos laboratórios remotos, pode-se apresentar como algumas de suas qualidades únicas as capacidades de (HERADIO *et al.*, 2016):

- Lidar com equipamentos físicos para aquisição de informações reais;
- Aprender sobre a existência e como lidar com diferentes tipos de erros de medida.

Porém, dependendo dos diferentes objetivos de aprendizado, as qualidades dos laboratórios virtuais e remotos podem gerar desvantagens. Por exemplo, os erros de medição presentes nos laboratórios remotos são importantes para ensinar os estudantes, já familiarizados com a aquisição e controle de sistemas, a identificar e lidar com tais erros no mundo real. Para o caso de estudantes sendo introduzidos ao controle de sistemas, a presença destes erros de medição pode gerar dúvidas e distrações (KHEIR *et al.*, 1996; HERADIO *et al.*, 2016). Portanto, a eficiente combinação do uso de laboratórios virtuais com remotos depende do objetivo de aprendizado por trás de cada experimento de controle. Em alguns casos, sendo necessária a mistura eficiente destes dois tipos de laboratórios (HERADIO *et al.*, 2016).

1.1.3 Reusabilidade

Reconhecidamente a criação de LVRs é uma tarefa multidisciplinar e repetitiva, envolvendo o trabalho com sistemas de aquisição e controle, em alguns casos sistemas embarcados, desenvolvimento de servidores e interfaces de aplicações Web e projeto arquitetural de software. Por anos, a tarefa de criação dos LVRs para sistemas de controle complexos demandava um grande esforço por parte dos professores, que embora dominem os conhecimentos de controle, o mesmo não pode ser generalizado para os conhecimentos de desenvolvimento Web e de software. Porém, com o avanço tecnológico e especialmente após a criação de ferramentas que auxiliam na reusabilidade dos LVRs, o processo tornou-se mais simples, permitindo aos professores dar mais atenção aos aspectos pedagógicos dos LVRs (HERADIO *et al.*, 2016; HERADIO; TORRE; DORMIDO, 2016; VARGAS *et al.*, 2011; VARGAS *et al.*, 2009).

Um requisito básico dos LVRs é ser dinâmico e interativo, de modo que qualquer mudança que ocorra nas variáveis do sistema sejam imediatamente refletidas na interface gráfica. Isto permite aos usuários acompanhar instantaneamente a evolução do controle do sistema ao longo do tempo. Este requisito deve ser cumprido por todos os LVRs e seu desenvolvimento

acaba se tornando um processo complexo e repetitivo para diferentes sistemas de controle. Deste modo, ferramentas como o EjsS (Easy Java/Javascript Simulations) podem ser utilizadas para aumentar a produtividade e reusabilidade na criação dos LVRs. O EjsS é uma ferramenta de geração de código, por meio de uma interface gráfica é possível construir a interface dos LVRs contendo até mesmo a simulação discreta de sistemas de controle (HERADIO; TORRE; DORMIDO, 2016; SALZMANN; GILLET, 2013; CHACÓN *et al.*, 2015b; EJS Wiki, 2018).

Para proporcionar uma experiência mais realista aos usuários, os LVRs apresentam recursos de gráficos, vídeo, áudio e outros. Para tanto, a tecnologia de Java Applets é muito utilizada na criação das interfaces dos LVRs, principalmente pela existência de bibliotecas em Java, como o EjsS, que auxiliam no processo de desenvolvimento. Porém, os Java Applets estão atualmente deixando de ter suporte nos navegadores Web. Deste modo, o desenvolvimento da interface de LVRs passou a migrar para o uso da tecnologia JavaScript. Embora, o JavaScript apresente todos os recursos necessários para tornar a experiência do usuário mais realista, existe um menor número de bibliotecas para o auxílio no desenvolvimento de LVRs (HERADIO; TORRE; DORMIDO, 2016; SALZMANN; GILLET, 2013).

Tradicionalmente, os recursos de servidor dos laboratórios remotos são implementados com auxílio de programas sendo executados nos ambientes do MATLAB/Simulink ou LabVIEW, instalados em um computador (HERADIO; TORRE; DORMIDO, 2016). Para proporcionar a reusabilidade, trabalhos como Farias *et al.* (2010), Vargas *et al.* (2009) e Chacón *et al.* (2015b) apresentam metodologias completas para o desenvolvimento de laboratórios remotos por meio destes softwares, utilizando recursos prontos.

Porém, esta abordagem de laboratório remoto tem um alto custo de implementação, relacionado as necessidades de ter um computador dedicado e compra da licença do MATLAB/ Simulink ou LabVIEW. Como forma de reduzir este custo, trabalhos como Sáenz *et al.* (2015) e Bermúdez-Ortega *et al.* (2015) apresentam metodologias de desenvolvimento de laboratórios remotos, nas quais o computador é substituído por um computador de placa única, por exemplo, BeagleBone Black e Raspberry Pi, conhecidas em inglês por *Single Board Computer* ou SBC, e o MATLAB/Simulink e LabVIEW são substituídos pelo Node.js, que é um ambiente *open source* para execução de programas em linguagem de programação JavaScript.

1.1.4 Liberdade no projeto de controladores

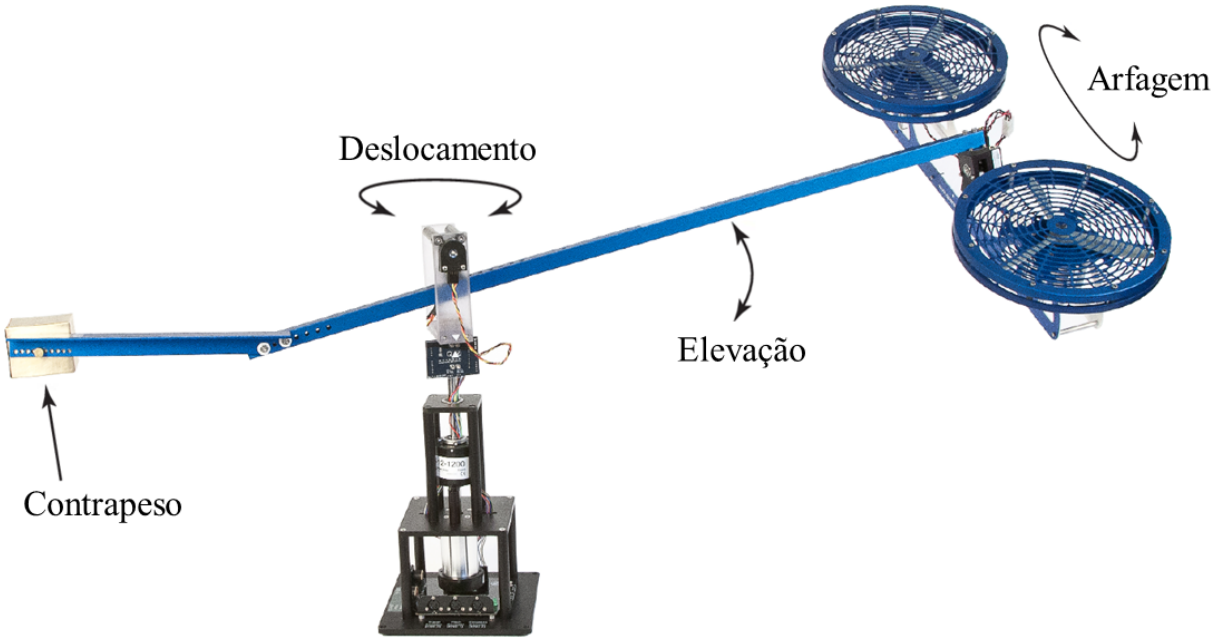
Tradicionalmente os LVRs são implementados com um forte acoplamento ao controle do sistema em malha fechada. Ou seja, o LVR apresenta uma implementação de controlador fixa, os usuários tem a capacidade de apenas alterar os ganhos associados ao controlador e configurar diferentes sinais de referência. No máximo os usuários podem escolher entre algumas opções de controle pré-implementadas. Esta abordagem de desenvolvimento é muito utilizada devido a maior facilidade de implementação. Porém, tal configuração é insuficiente para atender usuários mais avançados, como pesquisadores e alunos de pós-graduação, que precisam desenvolver suas próprias leis de controle (HERADIO; TORRE; DORMIDO, 2016).

Neste contexto, surgem algumas implementações com o objetivo de flexibilizar o projeto de controladores em LVRs. Em Chacón *et al.* (2015a) é implementado um interpretador de linguagem dentro do LVR, deste modo o usuário pode implementar o código do controlador que deseja executar. Em Galan *et al.* (2016) foi implementada uma aplicação *desktop*, denominada editor de experimentos, que permite o carregamento de LVRs. Esta aplicação permite aos usuários modificar o comportamento do laboratório, por meio da adição de novas funcionalidades ou da reconstrução de parte do LVR (HERADIO; TORRE; DORMIDO, 2016).

1.2 HELICÓPTERO COM 3 GDL

Para desenvolver um LVR é necessário ter um sistema de controle pronto para execução (SÁENZ *et al.*, 2015). Um exemplo de sistema de controle disponibilizado no formato de laboratório remoto é o Helicóptero com 3 graus de liberdade (GDL), como pode ser visto em Apkarian e Dawes (2000), Ishutkina (2004) e Ishutkina *et al.* (2004). O Helicóptero com 3 GDL, representado na Figura 1, é comercializado pela empresa Quanser. Os 3 GDL que compõem o sistema são denominados arfagem, deslocamento e elevação, sendo o controle destes dois últimos o objetivo deste sistema (APKARIAN; LÉVIS; FULFORD, 2012).

Figura 1 – Helicóptero com 3 GDL comercializado pela Quanser.



Fonte: Adaptada de Apkarian, Lévis e Fulford (2012).

Em Apkarian e Dawes (2000) é apresentada a solução de laboratório remoto desenvolvida pela Quanser, tendo como exemplo a aplicação no Helicóptero com 3 GDL. No servidor o controlador do sistema é projetado por meio do MATLAB e implementado por meio de um diagrama de blocos do Simulink. Por meio da *toolbox* Real-Time Workshop (nas versões mais recentes do MATLAB é conhecida como Simulink Coder), o controlador do Simulink é

convertido em código na linguagem C. A compilação deste código gera um programa que é utilizado pelo WinCon, que por sua vez é um produto comercializado pela Quanser para o controle de plantas de sua autoria (APKARIAN; DAWES, 2000).

O controle realizado pelo WinCon pode ser tanto remoto, quanto local. Para o controle remoto a Quanser desenvolveu o WebLab, que atua como uma interface gráfica para o WinCon. Tudo isto permitiu construir um laboratório remoto, no qual o usuário pode ajustar os ganhos de um controlador e modificar o sinal de referência. Na época as tecnologias de transmissão de vídeo ainda não apresentavam um bom desempenho, portanto como forma de acompanhar a execução do controle foi utilizado um projeto 3D da planta, movimentando-se de acordo com a informação do sistema real (APKARIAN; DAWES, 2000).

Em Ishutkina (2004) e Ishutkina *et al.* (2004) também é apresentado o desenvolvimento de um laboratório remoto para o Helicóptero com 3 GDL, comercializado pela Quanser. Nesta metodologia a finalidade da planta, do WinCon, do MATLAB/Simulink e do Real-Time Workshop é a mesma da metodologia apresentada em Apkarian e Dawes (2000). Porém, Ishutkina (2004) desenvolve a própria interface gráfica para o WinCon, por meio da tecnologia de Java Applets, além de incluir a transmissão de vídeo da planta, em tempo real. Quanto as funcionalidades, o laboratório remoto desenvolvido por Ishutkina (2004) permite ao usuário desenvolver o próprio controlador, ajustar os ganhos deste e também modificar o sinal de referência.

Para configurar os diferentes tipos de controle o usuário deve realizar o *upload* de um diagrama de blocos do Simulink, respeitando um conjunto de convenções. Como a planta sendo disponibilizada por laboratório remoto está sujeita a acidentes por conta de controladores mal projetados, foi desenvolvido um sistema para garantir a integridade física da planta. Tal sistema realiza o controle supervisório do Helicóptero com 3 GDL, com a finalidade de mantê-lo atuando apenas dentro de uma região segura (ISHUTKINA, 2004).

O desenvolvimento de um LVR que permite ao usuário criar o próprio controlador é uma abordagem interessante para o sistema do Helicóptero com 3 GDL. Uma vez que, possuir uma dinâmica não-linear de sexta ordem torna este sistema atrativo para a validação de diferentes metodologias de controle. Em Lopes (2007) o Helicóptero com 3 GDL foi utilizado no estudo de um controle preditivo baseado em modelo (*Model Predictive Control*). No trabalho de Maia (2008), o Helicóptero com 3 GDL, com as condições de espaço de manobra restrito, saturação nos atuadores e perturbações limitadas, foi utilizado no estudo de um controlador preditivo robusto.

O Helicóptero com 3 GDL também é utilizado para o estudo de metodologias que consideram o atraso do sinal de controle uma falha a ser detectada (APOLINÁRIO, 2009; PAULA, 2011). Em Apolinário (2009) e Pereyra (2013) são utilizados controles com modos deslizantes discreto em conjunto com observadores, que além de estimar os estados é fundamental para detecção das falhas. Tal metodologia é capaz de melhorar o desempenho dos sistemas na presença de atraso do sinal de controle.

No trabalho de Buzachero (2014) o Helicóptero com 3 GDL foi utilizado na obtenção de resultados sobre a estabilidade robusta de sistemas lineares sujeitos à incertezas paramétricas, variantes no tempo, do tipo politópicas. Para o projeto destes controladores robustos foi explorado o projeto de controladores dinâmicos chaveados. Em Pereyra (2013) o Helicóptero com 3 GDL foi utilizado para o estudo da detecção de falhas por meio de filtros H_∞. Dois filtros foram projetados, um para detecção de falhas no atuador traseiro do helicóptero e outro para detecção de falhas no sensor do grau de liberdade de deslocamento.

1.3 JUSTIFICATIVA

Nas universidades, o ensino de novos conceitos e teorias é realizado tradicionalmente em salas de aula. Além disso, no currículo de engenharia existem áreas, como a engenharia de controle, na qual a experimentação em laboratórios é de grande importância. Por conta disto, as universidades de engenharia tem demonstrado interesse em complementar o ensino tradicional com ferramentas de experimentação por meio da Internet (SÁENZ *et al.*, 2015; KARAKASIDIS, 2013; CHACÓN *et al.*, 2015b; IONESCU *et al.*, 2013).

Nos cenários nacional (SCHAF, 2006) e internacional (VARGAS *et al.*, 2011), a comunidade de engenharia de controle, envolvendo diversas universidades, trabalha em conjunto para desenvolver e compartilhar LVRs. Isto comprova a importância e valor dos LVRs como ferramentas complementares de ensino e pesquisa na engenharia de controle. Neste contexto, a realização deste trabalho visa desenvolver um LVR, que contribua com a comunidade acadêmica e as tendências atuais de pesquisa na área de LVRs da seguinte forma:

- A experiência de desenvolvimento dos experimentos de controle dentro do LVR, deve ser semelhante a experiência do desenvolvimento em laboratório tradicional. Isto contribui para que a inclusão do LVR na metodologia de ensino respeite seu papel educacional, apresentado na Subseção 1.1.1;
- As interfaces dos laboratórios virtual e remoto, tradicionalmente desenvolvidas separadamente, devem ser integradas em uma única interface. Isto contribui para a combinação eficiente de laboratório virtual com laboratório remoto, facilitando assim a utilização dos LVRs para diferentes objetivos de aprendizado, como discutido na Subseção 1.1.2;
- O LVR deve ser reutilizável para diferentes sistemas de controle. Isto contribui para facilitar o desenvolvimento de LVRs, permitindo aos docentes focar nos aspectos pedagógicos dos LVRs, como discutido na Subseção 1.1.3;
- O LVR deve permitir ao usuário o desenvolvimento da própria técnica de controle e o projeto dos ganhos do controlador, por meio de técnicas como LQR (*Linear Quadratic Regulator* ou Regulador Linear Quadrático, em português) e atribuição de autoestrutura completa. Isto contribui para que o LVR seja aplicado no ensino e pesquisa de uma maior diversidade de técnicas de controle, como discutido na Subseção 1.1.4.

Na engenharia de controle os experimentos em laboratório conectam a teoria com a prática, melhorando assim o entendimento dos conceitos e o desenvolvimento do pensamento crítico (KHEIR *et al.*, 1996; KARAKASIDIS, 2013). Deste modo, o uso de ferramentas como LVRs contribui para formação de profissionais capacitados para o trabalho com sistemas de controle automático, que são hoje parte integrante do progresso e desenvolvimento da sociedade moderna (OGATA, 2010; NISE, 2012; D'AZZO; HOUPIS; SHELDON, 2003).

Quanto a contribuição tecnológica, a organização e funcionamento do LVR se assemelha ao de Apkarian e Dawes (2000) e Ishutkina (2004), porém com um conjunto *open source* de tecnologias. A CLI (*Command-Line Interface* ou Interface de Linha de Comando, em português) do Octave substitui o MATLAB, no projeto dos controladores. A placa de microcontrolador NUCLEO-F767ZI em conjunto com o pacote de funções Block Diagram Coder (BDC), desenvolvido neste trabalho, substituem o Simulink, o Real-Time Workshop e o WinCon, na configuração, geração e execução do controle. A aplicação Web do LVR atua como uma IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, em português) para a CLI do Octave. O desenvolvimento da aplicação Web utiliza as tecnologias em JavaScript, como Node.js, Angular.js e outros *frameworks* e bibliotecas *open source*.

O sistema de controle integrado ao LVR é o Helicóptero com 3 GDL, desenvolvido por Shimada (2015). Para tanto, o desenvolvimento deste trabalho contribui com o estudo dos Helicópteros com 3 GDL ao realizar as seguintes melhorias na resposta de controle do sistema, sugeridas na conclusão de Shimada (2015):

- Melhorar a perda de eficiência apresentada pelo aquecimento dos atuadores;
- Reduzir o torque sobre o movimento de deslocamento, produzido pelo fato das hélices girarem no mesmo sentido;
- Reduzir o ruído presente no sinal da ação de controle;
- Reduzir a oscilação na resposta do movimento de arfagem.

1.4 OBJETIVOS

O objetivo deste trabalho é selecionar as tecnologias, determinar as funcionalidades e planejar a arquitetura para o desenvolvimento de um LVR aplicado em sistemas de controle. O LVR desenvolvido é integrado ao sistema de um Helicóptero com 3 GDL, devidamente modelado e instrumentado para aplicação de técnicas de controle.

1.4.1 Objetivos específicos

O sucesso do desenvolvimento deste trabalho depende da conclusão dos seguintes objetivos específicos:

- Determinar o modelo matemático do Helicóptero com 3 GDL pelo método de prototipagem virtual, utilizando os softwares SolidWorks e Adams;
- Instalar um par de hélices com rotação em sentidos opostos e identificar a relação matemática para acionamento dos atuadores;
- Estimar as posições e velocidades angulares dos graus de liberdade do sistema a partir da leitura dos sensores do tipo *encoder* incremental;
- Desenvolver o pacote denominado Block Diagram Coder, para construir, gerar e executar o controle virtual, por meio do Octave CLI, e o controle remoto, em uma placa de microcontrolador NUCLEO-F767ZI;
- Desenvolver a aplicação Web do LVR por meio das tecnologias em JavaScript;
- Executar experimentos de controle seguidor com realimentação de estado, utilizando as técnicas LQR, atribuição de autoestrutura completa e observador de Luenberger, em laboratório tradicional e LVR, como forma de validar o desenvolvimento deste.

1.5 ORGANIZAÇÃO DO TRABALHO

O desenvolvimento deste trabalho está organizado nos seguintes capítulos:

- **Capítulo 2:** Revisão da literatura dos tópicos relacionados a Helicópteros com 3 GDL e LVRs, importantes para o desenvolvimento deste trabalho;
- **Capítulo 3:** Apresentação dos conceitos relacionados a teoria de variáveis de estado, projeto de controladores e estimadores de estado;
- **Capítulo 4:** Apresentação dos conteúdos relacionados a descrição, modelagem e instrumentação do Helicóptero com 3 GDL, bem como a construção das técnicas de controle em laboratório tradicional;
- **Capítulo 5:** Apresentação dos conteúdos relacionados ao desenvolvimento do LVR, bem como a construção das técnicas de controle em LVR;
- **Capítulo 6:** Execução e análise dos resultados de diferentes experimentos de controle aplicados ao Helicóptero com 3 GDL, em laboratório tradicional e LVR;
- **Capítulo 7:** Conclusões sobre o desenvolvimento do trabalho e considerações sobre trabalhos futuros.

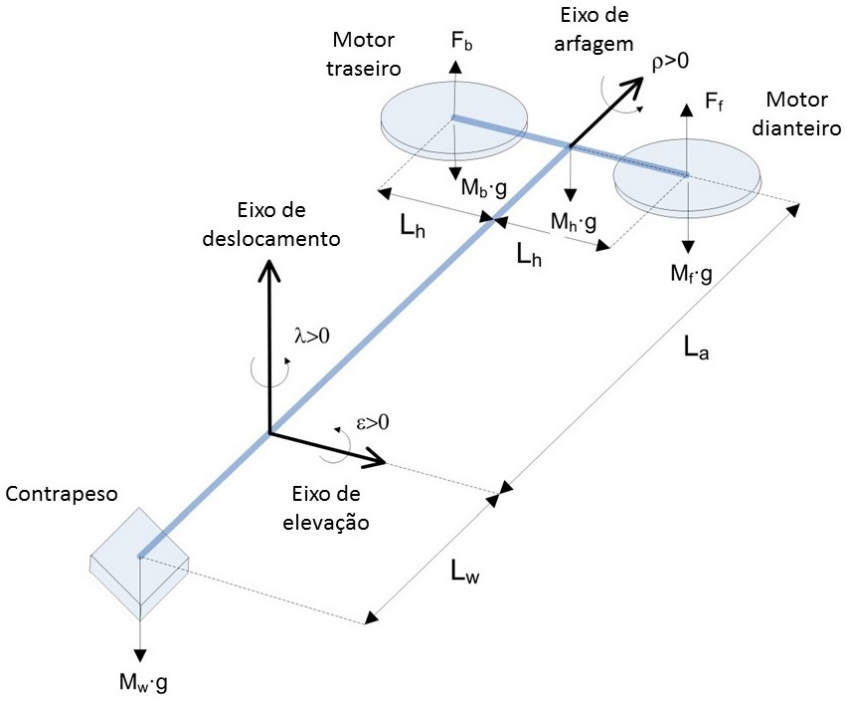
2 REVISÃO DA LITERATURA

Este capítulo apresenta a revisão da literatura dos tópicos relacionados a Helicópteros com 3 GDL e LVRs, importantes para o desenvolvimento deste trabalho. Quanto aos Helicópteros com 3 GDL são apresentadas diferentes considerações, convenções e métodos para determinar o modelo dinâmico do sistema. Para os LVRs utilizados na engenharia de controle são apresentadas as formas de organização, quanto a arquitetura, divisão das funcionalidades e localização da malha fechada de controle.

2.1 MODELOS MATEMÁTICOS DO HELICÓPTERO COM 3 GDL

O Helicóptero com 3 GDL, mostrado na Figura 1, é um sistema análogo a um helicóptero Tandem. Desta forma, apresenta um par de motores dianteiro e traseiro com eixos paralelos, que produzem forças de empuxo normais ao corpo do helicóptero. Para determinar o modelo matemático do Helicóptero com 3 GDL diferentes convenções podem ser adotadas, uma destas é proposta pela Quanser no diagrama de corpo livre da Figura 2 (APKARIAN; LÉVIS; FULFORD, 2012).

Figura 2 – Diagrama de corpo livre proposto pela Quanser.

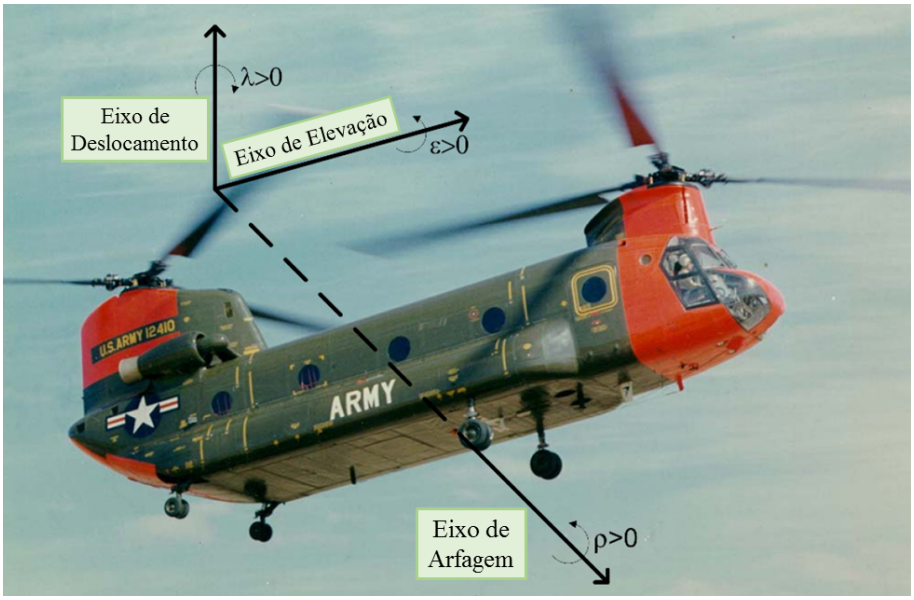


Fonte: Adaptada de Apkarian, Lévis e Fulford (2012).

No diagrama de corpo livre da Figura 2 são indicados os eixos e sentidos adotados para os GDL de deslocamento, elevação e arfagem, bem como os parâmetros de interesse para modelagem do Helicóptero com 3 GDL. É importante destacar que o contrapeso, presente

apenas no sistema do Helicóptero com 3 GDL, tem a finalidade de reduzir a massa efetiva do helicóptero, de modo que as forças de empuxo produzidas pelos motores sejam suficientes para atuar sobre o sistema. Para um melhor entendimento da relação entre o Helicóptero com 3 GDL e um helicóptero Tandem, o diagrama de corpo livre da Figura 2 é adaptado para o helicóptero Tandem do modelo Boeing HC-1B Chinook na Figura 3.

Figura 3 – Diagrama de corpo livre proposto pela Quanser adaptado para o Boeing HC-1B Chinook.



Fonte: Adaptada de Apkarian, Lévis e Fulford (2012).

O modelo matemático proposto pela Quanser é descrito em Apkarian, Lévis e Fulford (2012), sendo todo o equacionamento apresentado em Quanser (2017). Este modelo é obtido pelo método de Euler-Lagrange, para isto a inércia do sistema foi simplificada assumindo-se massas pontuais associadas à dianteira do helicóptero, à traseira do helicóptero e ao contrapeso. Como visto na Figura 2, considera-se a simplificação de que os centros de massa do contrapeso e do helicóptero, e o ponto de revolução da elevação estão alinhados sobre o eixo de arfagem. Desta forma, considera-se que o centro de massa do helicóptero coincide com o ponto de revolução da arfagem. A equação de Euler-Lagrange foi aplicada para as coordenadas generalizadas de deslocamento, elevação e arfagem e os torques foram descritos em função das forças de empuxo dianteiro e traseiro, das distâncias medidas da planta e do ângulo de arfagem.

As variáveis de estado do modelo matemático de Apkarian, Lévis e Fulford (2012) são as posições e velocidades angulares das coordenadas generalizadas, e as variáveis de controle são as tensões dos motores que produzem as forças de empuxo dianteiro e traseiro. Este modelo matemático, por ser encontrado no manual do Helicóptero com 3 GDL comercializado pela Quanser, é utilizado em vários trabalhos. Por exemplo, em Buzachero (2014), Manesco (2013), Pereyra (2013), Silva (2013), Buzachero (2010), Apolinário (2009) e Rodrigues (2009).

Em Lopes (2007) é apresentado um modelo matemático para o Helicóptero com 3 GDL, também obtido pelo método de Euler-Lagrange, porém com menos simplificações que o modelo de Apkarian, Lévis e Fulford (2012). No modelo de Lopes (2007) a inércia é simplificada por massas pontuais associadas ao contrapeso, ao braço de sustentação, ao helicóptero e também ao sistema de perturbação ativa presente na planta da Quanser. Neste modelo não existe a simplificação assumindo que os centros de massa do contrapeso, braço de sustentação, sistema de perturbação ativa, helicóptero e o ponto de revolução da elevação estão alinhados sobre o eixo de arfagem.

Porém, Lopes (2007) desacoplou a dinâmica de arfagem das dinâmicas de deslocamento e elevação, ainda mantendo a simplificação de que o centro de massa do helicóptero coincide com o ponto de revolução da arfagem. Deste modo, a equação de Euler-Lagrange foi aplicada apenas para as coordenadas generalizadas de deslocamento e elevação, a dinâmica do movimento de arfagem foi obtida pelo método de Newton-Euler. Os torques foram descritos em função das forças de empuxo dianteiro e traseiro, das distâncias medidas da planta e do ângulo de arfagem. As variáveis de estado do modelo matemático de Lopes (2007) são as posições e velocidades angulares do deslocamento, elevação e arfagem, e as variáveis de controle são as tensões dos motores que produzem as forças de empuxo dianteiro e traseiro.

Tanto o modelo de Apkarian, Lévis e Fulford (2012), quanto de Lopes (2007) desprezam os efeitos de atritos nas juntas, arrasto aerodinâmico e momento angular das hélices, que na planta da Quanser giram no mesmo sentido. No trabalho de Maia (2008), o modelo matemático determinado em Lopes (2007) foi avaliado experimentalmente, por meio de um controle em malha fechada, a fim de determinar o quão boa era a representação da dinâmica do Helicóptero com 3 GDL. Desta forma, Maia (2008), constatou que a simulação das dinâmicas de arfagem e deslocamento não correspondia com o resultado experimental e aplicou duas modificações ao modelo matemático, tornando a representação da dinâmica do sistema mais fiel:

1. Como na planta da Quanser as hélices dianteira e traseira giram no mesmo sentido, surge um torque adicional atuando sobre o helicóptero e produzindo um movimento de deslocamento, deste modo a posição de arfagem não se mantém em zero no regime para compensar tal movimento. Sendo assim, Maia (2008) adicionou um termo constante à dinâmica do deslocamento;
2. Como o modelo de Lopes (2007) despreza os efeitos de atrito nas juntas, Maia (2008) assumiu ser esta a causa das respostas experimentais do deslocamento e arfagem serem mais lentas e terem menor amplitude que as respostas simuladas. Desta forma, foi adicionado o efeito do atrito viscoso sobre às dinâmicas do deslocamento e arfagem.

Este modelo matemático determinado por Lopes (2007) e posteriormente complementado por Maia (2008) foi adotado nos trabalhos de Paula (2011) e Silva (2011) sem a necessidade de alterações, uma vez que todos estes trabalhos foram realizados sobre o

Helicóptero com 3 GDL comercializado pela Quanser.

Nos trabalhos de Oliveira e Angélico (2018) e Choudhary (2014) o modelo matemático do Helicóptero com 3 GDL é determinado pelo método de Newton-Euler. As dinâmicas dos movimentos de deslocamento, elevação e arfagem são descritas com relação, respectivamente, aos momentos de inércia e acelerações angulares em torno dos eixos de deslocamento, elevação e arfagem. Os torques são descritos em função das forças de empuxo dianteiro e traseiro, das distâncias medidas da planta e do ângulo de arfagem. Ao assumir as mesmas simplificações, os modelos de Apkarian, Lévis e Fulford (2012), Oliveira e Angélico (2018) e Choudhary (2014) se tornam equivalentes.

A fim de obter um modelo mais rigoroso do sistema, Oliveira e Angélico (2018) propôs um segundo modelo matemático determinado pelo método de Newton-Euler. Nesta abordagem foi considerado que o centro de massa do helicóptero está deslocado abaixo do ponto de revolução do movimento de arfagem. Desta forma, embora o deslocamento do centro de massa seja pequeno, a arfagem se comporta mais como um movimento pendular do que inercial. Para o segundo modelo matemático proposto por Oliveira e Angélico (2018) as variáveis de estado continuam sendo as posições e velocidades angulares do deslocamento, elevação e arfagem, mas as variáveis de controle do sistema são as velocidades de rotação dos motores dianteiro e traseiro elevadas ao quadrado.

Nos trabalhos de Castañeda *et al.* (2016) e Yang e Zheng (2019) o modelo matemático do Helicóptero com 3 GDL é determinado pelo método de Newton-Euler, assim como o modelo de Oliveira e Angélico (2018) e Choudhary (2014). As variáveis de estado continuam sendo as posições e velocidades angulares do deslocamento, elevação e arfagem. Porém, para todos os modelos apresentados até então o torque atuando sobre o movimento de deslocamento dependia apenas do ângulo de arfagem ser diferente de zero para existir uma componente de força que produz o movimento de deslocamento. Enquanto no modelo matemático de Castañeda *et al.* (2016) e Yang e Zheng (2019), o torque atuando sobre o deslocamento depende também do ângulo de elevação, que altera a distância da componente de força até o eixo de deslocamento. Esta distância tem seu valor máximo quando o braço de sustentação se encontra na posição horizontal. Como variáveis de controle do modelo matemático, Castañeda *et al.* (2016) e Yang e Zheng (2019) utilizam a soma e a diferença das tensões que produzem as forças de empuxo dianteiro e traseiro.

Nos trabalhos de Shan, Liu e Nowotny (2005), Zhou *et al.* (2015) e Zheng e Zhong (2011) o modelo matemático é determinado pelo método de Newton-Euler, por meio de considerações semelhantes aos trabalhos de Castañeda *et al.* (2016) e Yang e Zheng (2019). Porém, considera-se que o movimento de deslocamento pode ser obtido por meio de um controle preciso do movimento de arfagem. Deste modo, a dinâmica de 3 GDL é simplificada como uma dinâmica de 2 GDL com os estados do modelo sendo as posições e velocidades angulares da elevação e arfagem.

Em Breganon (2009) e Shimada (2015) o modelo matemático do Helicóptero com 3

GDL é determinado por uma técnica de sistemas multicorpos (em inglês, *Multibody Systems* ou MBS). A técnica MBS empregada é denominada prototipagem virtual, na qual são utilizadas ferramentas computacionais para obter o modelo matemático do sistema. Em Breganon (2009) foi utilizado o software Adams (*Advanced Dynamic Analysis of Mechanical Systems*), enquanto em Shimada (2015) foram utilizados os softwares SolidWorks e Adams. Como na prototipagem virtual são utilizadas informações mais fiéis de geometria e inércia dos corpos, o modelo matemático obtido não apresenta tantas simplificações quanto nas abordagens de Apkarian, Lévis e Fulford (2012), Choudhary (2014), Castañeda *et al.* (2016) e Zheng e Zhong (2011).

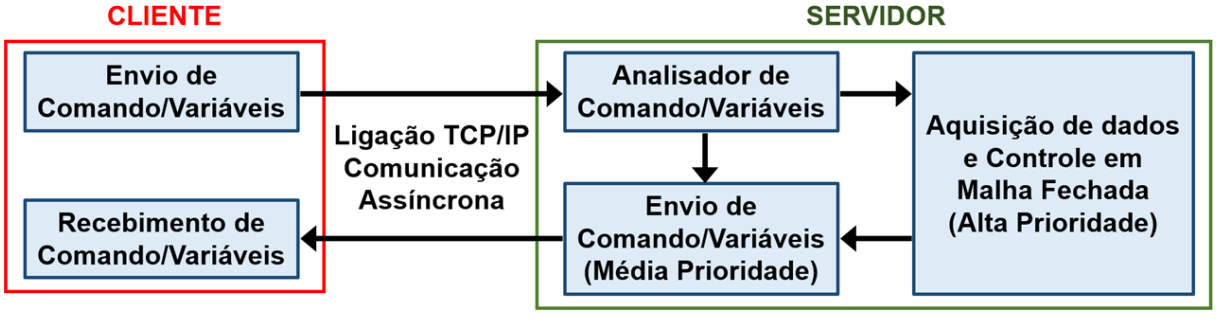
Na prototipagem virtual apresentada nos trabalhos de Breganon (2009) e Shimada (2015) são escolhidas as variáveis de saída do modelo matemático e não as variáveis de estado. Desta forma, as variáveis de saída do modelo matemático são as posições e velocidades angulares dos movimentos de deslocamento, elevação e arfagem, e as variáveis de controle do modelo são as forças de empuxo dianteiro e traseiro.

2.2 ORGANIZAÇÃO DE LABORATÓRIOS VIRTUAIS E REMOTOS PARA ENGENHARIA DE CONTROLE

Os experimentos de sistemas de controle, disponibilizados em LVRs, envolvem principalmente o controle de malha fechada em tempo real, como visto em Vargas *et al.* (2008), Vargas *et al.* (2009), Vargas *et al.* (2011), Chacón *et al.* (2015b) e Sáenz *et al.* (2015). Uma necessidade dos laboratórios remotos, para tais experimentos, é possuir bom desempenho e adaptabilidade quanto a continuidade da interação entre a interface do laboratório remoto e a planta de controle localizada remotamente (VARGAS *et al.*, 2008).

Neste contexto, a utilização da arquitetura cliente-servidor se apresenta como uma boa abordagem para o desenvolvimento de laboratórios remotos. Isto ocorre por conta da grande flexibilidade, desempenho e adaptabilidade desta arquitetura, em situações que demandam qualidade de serviço e continuidade da comunicação (VARGAS *et al.*, 2008). Na Figura 4, como visto em Vargas *et al.* (2009) e Vargas *et al.* (2011), está representada uma arquitetura cliente-servidor utilizada na organização das funcionalidades de laboratórios remotos.

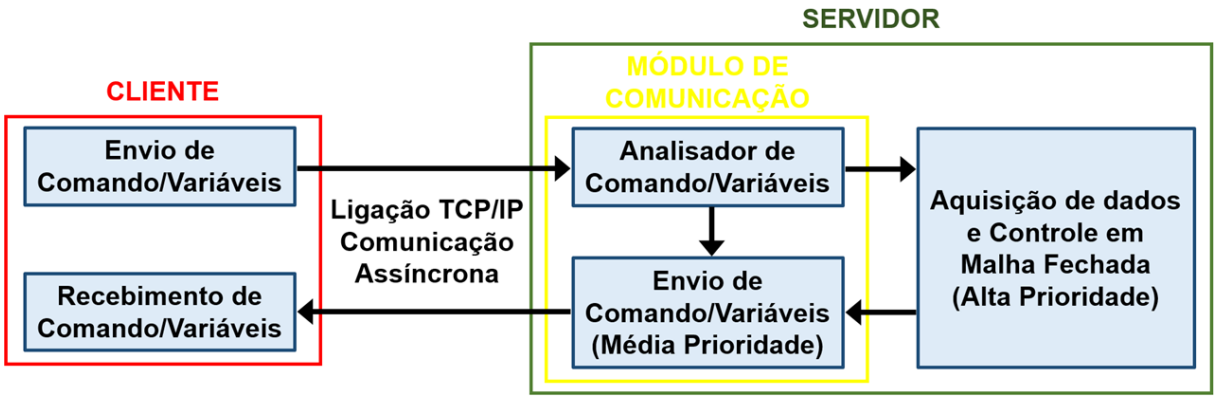
Figura 4 – Arquitetura cliente-servidor utilizada em laboratórios remotos.



Fonte: Adaptada de Vargas *et al.* (2009).

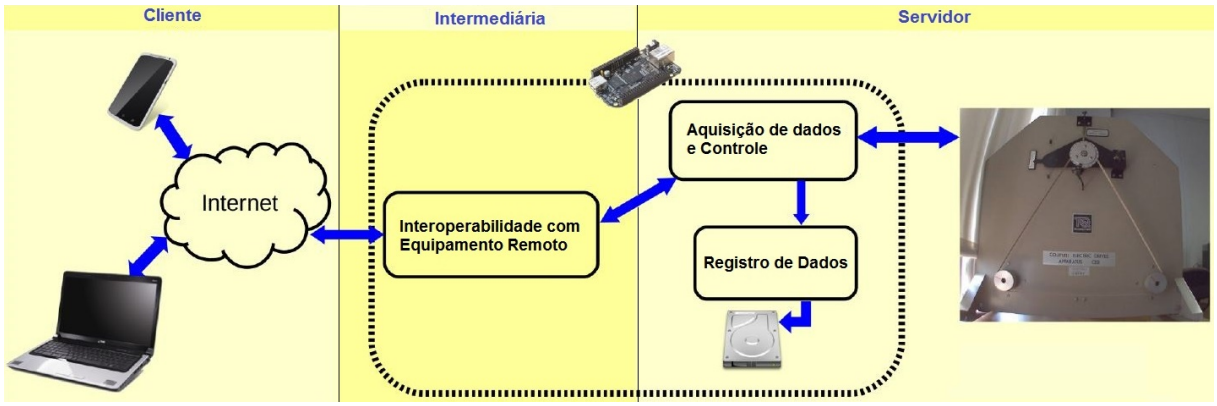
Mais recentemente, as abordagens de construção de laboratórios remotos evoluíram da arquitetura cliente-servidor para uma arquitetura de três camadas. Nesta, as funções de análise e envio de comando/variáveis, que constitui o módulo de comunicação representado na Figura 5, foram substituídas por um módulo de interoperabilidade, como apresentado na Figura 6, incluído entre as camadas do cliente e do servidor. Deste modo, a camada intermediária trata exclusivamente da comunicação entre cliente e servidor, reduzindo a dependência entre estas camadas (CHACÓN *et al.*, 2015b; SÁENZ *et al.*, 2015).

Figura 5 – Módulo de comunicação na arquitetura cliente-servidor.



Fonte: Adaptada de Vargas *et al.* (2011).

Figura 6 – Arquitetura cliente-servidor com camada intermediária utilizada em laboratórios remotos.



Fonte: Adaptada de Sáenz *et al.* (2015).

A aplicação da arquitetura cliente-servidor permite uma divisão das funcionalidades que compõem os laboratórios remotos. O lado do servidor realiza, entre outras atividades, a aquisição e controle da planta localizada remotamente, a transmissão de vídeo em tempo real, e a implementação da comunicação entre cliente e servidor, podendo ser com ou sem uma camada intermediária (CHACÓN *et al.*, 2015b; SÁENZ *et al.*, 2015).

O lado do cliente tem a funcionalidade de atender o usuário por meio de uma interface gráfica. Isto permite realizar, entre outras atividades, a configuração de ganhos dos controladores,

a mudança dos sinais de referência, a implementação de diferentes leis de controle e também o monitoramento do controle da planta, por meio de gráficos e vídeos atualizados em tempo de execução (CHACÓN *et al.*, 2015b; SÁENZ *et al.*, 2015; CHACÓN *et al.*, 2015a).

Além da divisão de funcionalidades entre as camadas de cliente e servidor existe duas diferentes abordagens de laboratórios remotos quanto a execução do controlador. Na primeira abordagem, a malha fechada de controle encontra-se exclusivamente no lado do servidor, como nos trabalhos de Vargas *et al.* (2008), Vargas *et al.* (2009), Vargas *et al.* (2011). Na segunda abordagem, a planta é disponibilizada remotamente para um controlador executado no lado do cliente, deste modo a malha de controle se fecha por meio da rede, de modo transparente ao usuário, como nos trabalhos de Guinaldo, Sánchez e Dormido (2010), Chacón *et al.* (2015b), Sáenz *et al.* (2015).

Os atrasos da rede de comunicação entre as camadas de cliente e servidor afetam estas duas abordagens de maneiras diferentes. No caso em que o controlador está no lado do cliente e a malha se fecha pela rede, o usuário monitora o sistema de controle sem sofrer com atrasos de comunicação, porém a resposta de controle pode perder desempenho devido a atrasos na aplicação da ação de controle. No caso da malha fechada de controle estar exclusivamente no lado do servidor não existe perda de desempenho no controle do processo, porém a interação e monitoramento do controle pelo usuário sofre com atrasos de comunicação (VARGAS *et al.*, 2011).

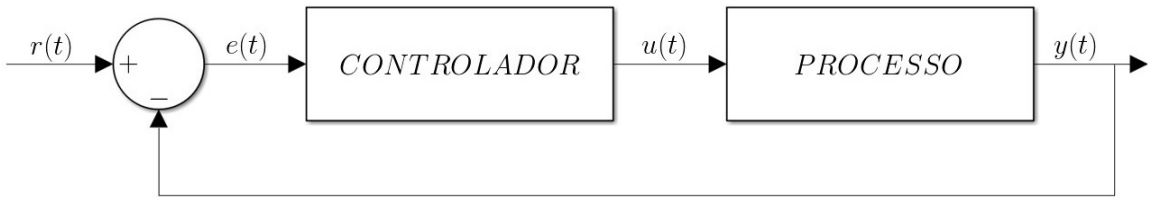
3 TEORIA DE CONTROLE

Neste capítulo são apresentados os conceitos relacionados a teoria de variáveis de estado, projeto de controladores e estimadores de estado. Os controles apresentados são o PID (Proporcional, Integral e Derivativo) e o seguidor com realimentação de estado. Os métodos utilizados no projeto de controle são a atribuição de autoestrutura completa e o LQR. O método apresentado para estimação de estado é o observador de Luenberger.

3.1 CONTROLE PROPORCIONAL, INTEGRAL E DERIVATIVO

A Figura 7 apresenta um diagrama simplificado do controle de um processo SISO (*Single-Input and Single-Output* ou Entrada Única e Saída Única, em português) por realimentação da saída. A referência de controle do processo (*setpoint*) é dada por $r(t)$. A entrada do controlador é o sinal de erro $e(t)$ obtido pela diferença entre a referência $r(t)$ e a saída do processo $y(t)$. A entrada do processo corresponde a saída do controlador $u(t)$.

Figura 7 – Diagrama simples do controle de um processo SISO por realimentação da saída.



Fonte: Autoria própria.

Segundo Åström e Hägglund (1995), partindo destas considerações, é possível definir a lei do controlador PID na Equação (1). A variável de controle $u(t)$ depende do erro $e(t)$ e resulta da soma das ações proporcional, integral e derivativa. Os parâmetros do controlador são o ganho proporcional K, o tempo integral T_i e o tempo derivativo T_d. O objetivo do controle é determinar a variável de controle $u(t)$, tal que a saída do processo $y(t)$ corresponda a referência $r(t)$ quando o processo estiver em regime.

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

3.1.1 Ação proporcional

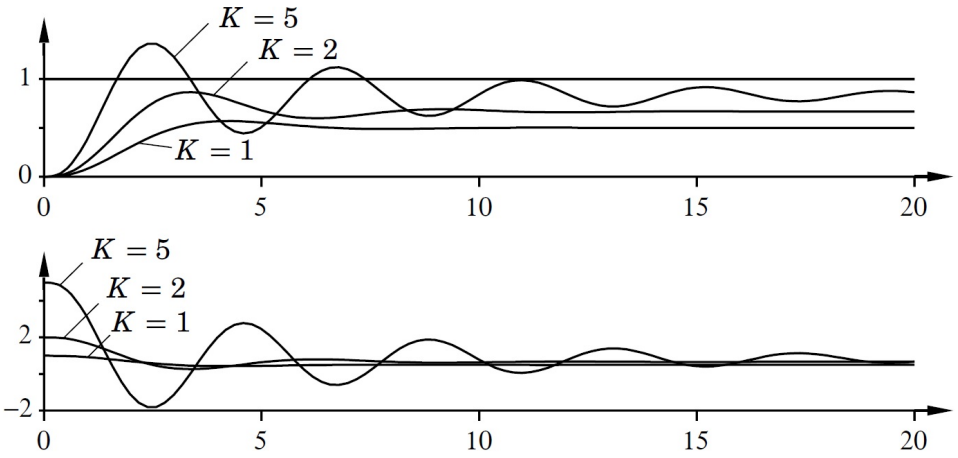
Para um controlador apenas com ação proporcional, a lei de controle na Equação (1) é reduzida para Equação (2). A variável de controle $u(t)$ assume um valor u_b quando o erro

$e(t)$ é zero (ÅSTRÖM; HÄGGLUND, 1995).

$$u(t) = Ke(t) + u_b \tag{2}$$

A Figura 8 apresenta um exemplo típico de controle apenas com ação proporcional, para diferentes valores de K. O processo simulado na Figura 8 é representado pela função transferência $G(s) = (s+1)^{-3}$, o gráfico superior mostra o sinal de saída do processo $y(t)$ para uma referência $r = 1$, o gráfico inferior apresenta o sinal de controle $u(t)$ oscilando em torno do valor de u_b. Deste modo, a Figura 8 mostra que o aumento de K reduz o erro $e(t)$ em regime, porém torna o sinal de saída do processo $y(t)$ oscilatório (ÅSTRÖM; HÄGGLUND, 1995).

Figura 8 – Simulação do controle de um processo SISO por realimentação da saída, apenas com ação proporcional.



Fonte: Åström e Hägglund (1995).

3.1.2 Ação integral

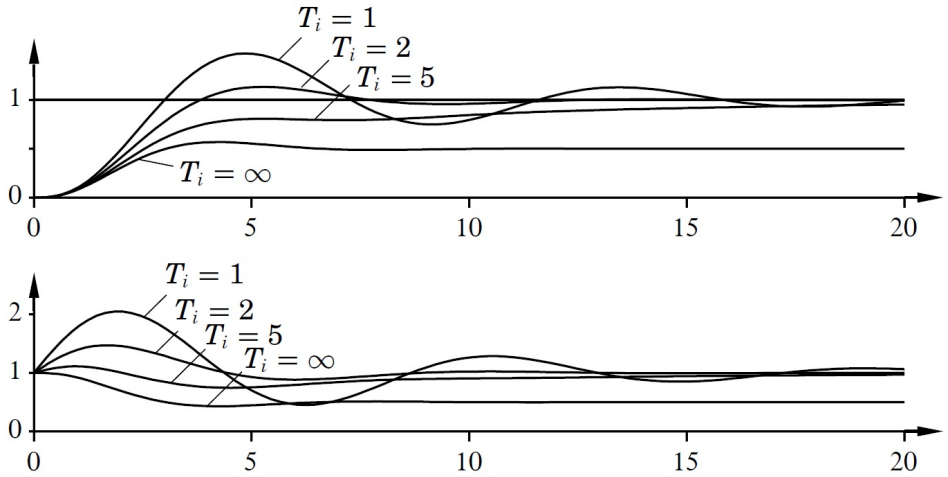
Para um controlador com ações proporcional e integral, a lei de controle na Equação (1) é reduzida para Equação (3). Analisando esta equação, a ação integral tem a função de reduzir o erro $e(t)$ em regime, desempenhada pelo termo u_b na lei de controle proporcional apresentada pela Equação (2) (ÅSTRÖM; HÄGGLUND, 1995; OGATA, 2010).

$$u(t) = Ke(t) + \frac{K}{T_i}\int_0^t e(\tau)d\tau \tag{3}$$

A Figura 9 apresenta as respostas de um controlador com ações proporcional e integral, para $K = 1$ e diferentes valores de T_i. O processo simulado na Figura 9 é representado pela função transferência $G(s) = (s+1)^{-3}$, o gráfico superior mostra a saída do processo $y(t)$ para uma referência $r = 1$, o gráfico inferior apresenta a variável de controle $u(t)$. Na Figura 9 a resposta de $y(t)$ para $T_i = \infty$ equivale a resposta de um controle apenas com ação proporcional $K = 1$, como pode ser visto na Figura 8 (ÅSTRÖM; HÄGGLUND, 1995).

Para todas as respostas de $y(t)$, na Figura 9, em que a ação integral é válida, o erro $e(t)$ em regime é removido. O aumento do efeito da ação integral, pela diminuição do valor de T_i, faz com que $y(t)$ se aproxime mais rápido de $r(t)$, ao mesmo passo que torna $y(t)$ mais oscilatória (ÅSTRÖM; HÄGGLUND, 1995).

Figura 9 – Simulação do controle de um processo SISO por realimentação da saída, com ações proporcional e integral.



Fonte: Åström e Hägglund (1995).

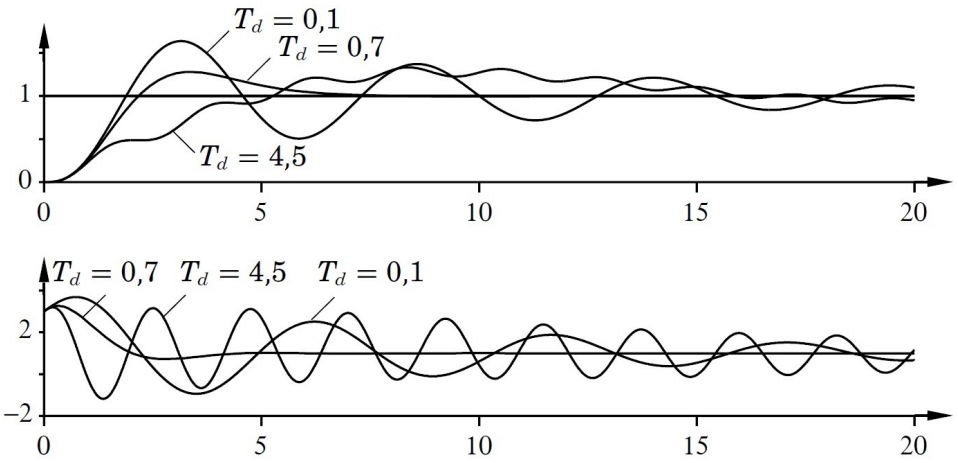
3.1.3 Ação derivativa

O objetivo da ação derivativa é promover a estabilidade de um controle com realimentação da saída, ou seja, em malha fechada. A ação derivativa permite ao controlador prever erros futuros e iniciar uma ação corretiva antecipada, tendendo a tornar o sistema mais estável (ÅSTRÖM; HÄGGLUND, 1995; OGATA, 2010).

A Figura 10 apresenta as respostas do controlador PID representado pela Equação (1), para $K = 3$, $T_i = 2$ e diferentes valores de T_d. O processo simulado na Figura 10 é representado pela função transferência $G(s) = (s + 1)^{-3}$, o gráfico superior mostra a saída do processo $y(t)$ para uma referência $r = 1$, o gráfico inferior apresenta a variável de controle $u(t)$ (ÅSTRÖM; HÄGGLUND, 1995).

As configurações de K e T_i fazem com que o controlador na Figura 10 tenha um comportamento oscilatório. Conforme o valor de T_d é incrementado (por exemplo, para valores de 0,1 e 0,7), o sinal de saída do processo $y(t)$ passa a apresentar um maior amortecimento. Porém, um aumento grande do valor de T_d (por exemplo, um valor de 4,5) torna a saída $y(t)$ novamente oscilatória (ÅSTRÖM; HÄGGLUND, 1995).

Figura 10 – Simulação do controle de um processo SISO por realimentação da saída, com ações proporcional, integral e derivativa.



Fonte: Åström e Hägglund (1995).

3.2 ESPAÇO DE ESTADO

Considerando um vetor $\boldsymbol{x}(t)$, denominado vetor de estado, composto por n variáveis de estado, um vetor $\boldsymbol{u}(t)$, que recebe o nome de vetor de controle, contendo m variáveis de controle, e o instante de tempo inicial sendo t_0. O estado de um sistema dinâmico é definido como o menor conjunto de n variáveis de estado, tal que o conhecimento de $\boldsymbol{x}(t_0)$ e $\boldsymbol{u}(t \geq t_0)$ determina completamente o comportamento do sistema para qualquer instante $t \geq t_0$. O espaço de estado é definido como o espaço n-dimensional, no qual os eixos coordenados são as componentes do vetor de estado $\boldsymbol{x}(t)$ (OGATA, 2010; D'AZZO; HOUPIS, 1995).

A análise no espaço de estado envolve as variáveis de controle, de saída e de estado, presentes na modelagem de sistemas dinâmicos. Considerando o vetor $\boldsymbol{y}(t)$, denominado vetor de saída, com l variáveis de saída, e a derivada temporal do vetor de estado $\boldsymbol{x}(t)$ sendo $\dot{\boldsymbol{x}}(t)$. Um sistema dinâmico é representado na notação de espaço de estado pelas equações de estado e saída, respectivamente, nas Equações (4) e (5), ambas na forma matricial (OGATA, 2010).

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},t) \tag{4}$$

$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},t) \tag{5}$$

A linearização das Equações (4) e (5), em torno de um ponto de operação, resulta nas equações de estado e saída, respectivamente, nas Equações (6) e (7) (OGATA, 2010):

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) \tag{6}$$

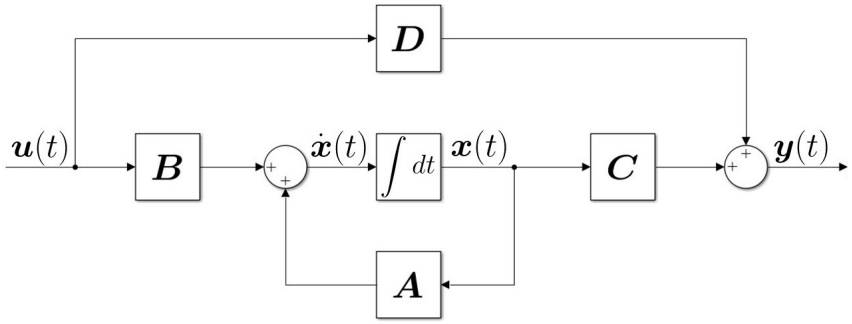
$$\boldsymbol{y}(t) = \boldsymbol{C}(t)\boldsymbol{x}(t) + \boldsymbol{D}(t)\boldsymbol{u}(t) \tag{7}$$

Para um sistema invariante no tempo as equações de estado e saída são, respectivamente, as Equações (8) e (9) (OGATA, 2010; D'AZZO; HOUPIS, 1995). A Figura 11 apresenta o diagrama de blocos da derivação destas equações.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{8}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \tag{9}$$

Figura 11 – Diagrama de blocos da derivação das equações de estado e saída de um sistema linear e invariante no tempo.



Fonte: Autoria própria.

De maneira resumida, os componentes da representação de um sistema dinâmico linear e invariante no tempo, na notação de espaço de estado, são (OGATA, 2010; D'AZZO; HOUPIS, 1995):

- O vetor de estado $\boldsymbol{x}(t)$ e sua derivada temporal $\dot{\boldsymbol{x}}(t)$, ambos com dimensão $n \times 1$;
- O vetor de controle $\boldsymbol{u}(t)$ com dimensão $m \times 1$;
- O vetor de saída $\boldsymbol{y}(t)$ com dimensão $l \times 1$;
- A matriz de estado $\boldsymbol{A}$ com dimensão $n \times n$;
- A matriz de entrada $\boldsymbol{B}$ com dimensão $n \times m$;
- A matriz de saída $\boldsymbol{C}$ com dimensão $l \times n$;
- A matriz de transmissão direta $\boldsymbol{D}$ com dimensão $l \times m$;

3.3 CONTROLE SEGUIDOR COM REALIMENTAÇÃO DE ESTADO

Ao partir da definição de espaço de estado, apresentada na Seção 3.2, e das equações de estado e saída, respectivamente, nas Equações (8) e (9). Considerando p o número de variáveis de saída controladas, uma matriz $\boldsymbol{E}$ com dimensão $p \times n$, responsável pela seleção das variáveis de saída controladas, e uma matriz $\boldsymbol{F}$ com dimensão $(l-p) \times n$, responsável pela seleção das variáveis de saída não controladas. Um sistema controlável MIMO (*Multiple-Input and Multiple-Output* ou Entradas Múltiplas e Saídas Múltiplas, em português), em malha

aberta, é representado pelas equações de estado e saída, respectivamente, nas Equações (10) e (11). Por conveniência de notação, a dependência no tempo dos vetores $\dot{\boldsymbol{x}}$, $\boldsymbol{x}$ e $\boldsymbol{u}$ não está explicitada (D'AZZO; HOUPIS, 1995).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{10}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{E} \\ \boldsymbol{F} \end{bmatrix} \boldsymbol{x} \tag{11}$$

As variáveis de saída controladas devem corresponder aos primeiros p componentes de $\boldsymbol{y}$, de modo que um vetor de saída controlada, com dimensão $p \times 1$, possa ser definido por $\boldsymbol{w} = \boldsymbol{E}\boldsymbol{x}$. Caso necessário o vetor de saída $\boldsymbol{y}$ deve ter a ordem alterada. No controle seguidor com realimentação de estado, as variáveis de saída controladas são realimentadas com o objetivo de seguir um vetor de referência $\boldsymbol{r}$, com dimensão $p \times 1$ e constante por partes, em regime. A definição matemática deste objetivo é representada pela Equação (12) (D'AZZO; HOUPIS, 1995).

$$\lim_{t \to \infty} \boldsymbol{w}(t) = \boldsymbol{r}(t) \tag{12}$$

O método de realimentação das variáveis de saída controladas consiste da adição de um comparador de vetores e um integrador, que satisfaça a Equação (13) (D'AZZO; HOUPIS, 1995).

$$\dot{\boldsymbol{z}} = \boldsymbol{r} - \boldsymbol{w} = \boldsymbol{r} - \boldsymbol{E}\boldsymbol{x} \tag{13}$$

Deste modo, é determinado um sistema aumentado, em malha aberta, governado pela equação de estado aumentada, na Equação (14), e pela equação de saída aumentada, na Equação (15). Estas são formadas pelas Equações de (10) a (13) (D'AZZO; HOUPIS, 1995).

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{z}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{E} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{r} = \bar{\boldsymbol{A}}\boldsymbol{x'} + \bar{\boldsymbol{B}}\boldsymbol{u} + \bar{\boldsymbol{B}}'\boldsymbol{r} \tag{14}$$

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} = \bar{\boldsymbol{C}}\boldsymbol{x'} \tag{15}$$

As matrizes $\boldsymbol{0}$ e $\boldsymbol{I}$ são uma forma conveniente de expressar, respectivamente, matrizes de zeros e identidade. Das Equações (14) e (15) são obtidos o vetor de estado aumentado $\boldsymbol{x}'$ e as matrizes aumentadas $\bar{\boldsymbol{A}}$, $\bar{\boldsymbol{B}}$, $\bar{\boldsymbol{B}}'$ e $\bar{\boldsymbol{C}}$, apresentados na Equação (16). A dimensão do vetor $\boldsymbol{x}'$ é $(n+p) \times 1$. A dimensão da matriz $\bar{\boldsymbol{A}}$ é $(n+p) \times (n+p)$. A matriz $\bar{\boldsymbol{B}}$ tem dimensão $(n+p) \times m$. A dimensão da matriz $\bar{\boldsymbol{B}}'$ é $(n+p) \times p$. E a matriz $\bar{\boldsymbol{C}}$ tem dimensão $l \times (n+p)$ (D'AZZO; HOUPIS, 1995).

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix};\ \bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{E} & \boldsymbol{0} \end{bmatrix};\ \bar{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix};\ \bar{\boldsymbol{B}}' = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix};\ \bar{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \tag{16}$$

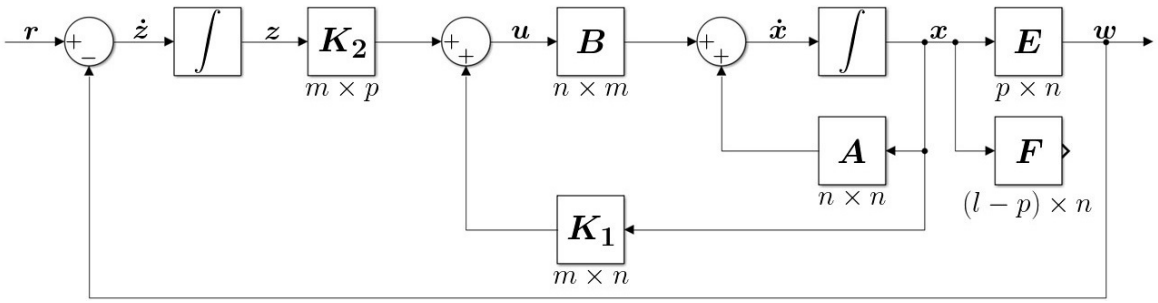
Considerando a matriz $\bar{\boldsymbol{K}}$, com dimensão $m \times (n+p)$, o ganho da realimentação de estado do sistema aumentado, apresentado nas Equações (14) e (15). A lei de controle a ser utilizada para uma realimentação de estado positiva é representada na Equação (17) (D'AZZO; HOUPIS, 1995). Para o caso de uma realimentação de estado negativa, a lei de controle é apresentada na Equação (18) (YOUNG; WILLEMS, 1972).

$$\boldsymbol{u} = \bar{\boldsymbol{K}}\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{K_1} & \boldsymbol{K_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} = \boldsymbol{K_1}\boldsymbol{x} + \boldsymbol{K_2}\boldsymbol{z} \tag{17}$$

$$\boldsymbol{u} = -\bar{\boldsymbol{K}}\boldsymbol{x}' = -\begin{bmatrix} \boldsymbol{K_1} & \boldsymbol{K_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} = -\boldsymbol{K_1}\boldsymbol{x} - \boldsymbol{K_2}\boldsymbol{z} \tag{18}$$

Nas Equações (17) e (18) é indicado que $\bar{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K_1} & \boldsymbol{K_2} \end{bmatrix}$. A matriz $\boldsymbol{K_1}$, com dimensão $m \times n$, é o ganho da realimentação de estado do sistema representado pelas Equações (10) e (11). A matriz $\boldsymbol{K_2}$, com dimensão $m \times p$, é o ganho atribuído a integral do erro de realimentação das variáveis de saída controladas. A Figura 12 apresenta o diagrama de blocos do controle seguidor com realimentação de estado para lei de controle representada na Equação (17) (D'AZZO; HOUPIS, 1995).

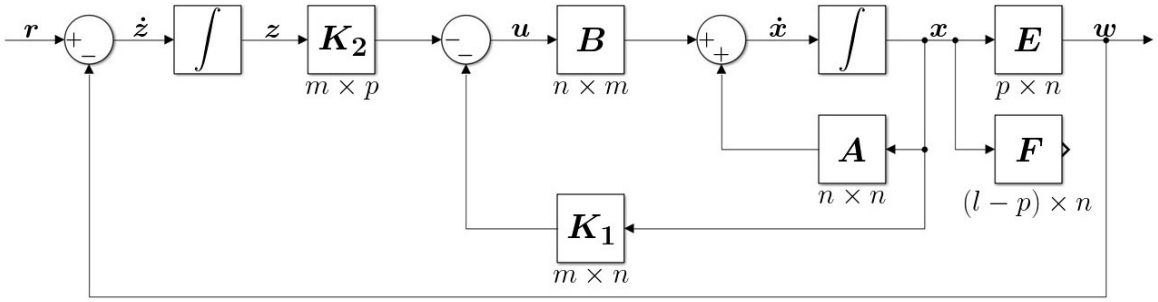
Figura 12 – Diagrama de blocos do controle seguidor com realimentação de estado positiva.



Fonte: Autoria própria.

Na Figura 13 está representado o diagrama de blocos do controle seguidor com realimentação de estado para lei de controle apresentada na Equação (18).

Figura 13 – Diagrama de blocos do controle seguidor com realimentação de estado negativa.



Fonte: Autoria própria.

As leis de controle nas Equações (17) e (18) são capazes de atribuir o espectro desejado de autovalores de malha fechada, se e somente se, o par de matrizes aumentadas de estado e entrada $(\bar{\boldsymbol{A}},\bar{\boldsymbol{B}})$ for controlável. Segundo D'azzo e Houpis (1995) e Young e Willems (1972), esta condição é verdadeira se as condições nas Equações (19) e (20) forem verdadeiras.

$$posto\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A^2B} & \cdots & \boldsymbol{A^{n-1}B}\end{bmatrix} = n \tag{19}$$

$$posto\begin{bmatrix}\boldsymbol{B} & \boldsymbol{A} \\ \boldsymbol{0} & \boldsymbol{-E}\end{bmatrix} = n + m \tag{20}$$

A condição na Equação (19) indica que o par de matrizes de estado e entrada $(\boldsymbol{A},\boldsymbol{B})$ é controlável. A condição na Equação (20) é satisfeita apenas se o número de variáveis de saída controladas p, necessárias para seguir o vetor de referência $\boldsymbol{r}$, for menor ou igual ao número de variáveis de controle m. Satisfazer a condição na Equação (20), garante que as leis de controle nas Equações (17) e (18) podem ser sintetizadas de modo que a saída do sistema em malha fechada $\boldsymbol{w}$ siga a referência $\boldsymbol{r}$. A equação de estado do sistema em malha fechada, para a lei de controle na Equação (17), está representada na Equação (21) (D'AZZO; HOUPIS, 1995).

$$\dot{\boldsymbol{x}}' = \begin{bmatrix}\dot{\boldsymbol{x}} \\ \dot{\boldsymbol{z}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A} + \boldsymbol{BK_1} & \boldsymbol{BK_2} \\ \boldsymbol{-E} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ \boldsymbol{z}\end{bmatrix} + \begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{I}\end{bmatrix}\boldsymbol{r} \tag{21}$$

A equação de estado do sistema em malha fechada, para a lei de controle na Equação (18), está representada na Equação (22) (YOUNG; WILLEMS, 1972).

$$\dot{\boldsymbol{x}}' = \begin{bmatrix}\dot{\boldsymbol{x}} \\ \dot{\boldsymbol{z}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A} - \boldsymbol{BK_1} & \boldsymbol{-BK_2} \\ \boldsymbol{-E} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{x} \\ \boldsymbol{z}\end{bmatrix} + \begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{I}\end{bmatrix}\boldsymbol{r} \tag{22}$$

De forma simplificada, as Equações (21) e (22) podem ser representadas como a Equação (23). Em que $\boldsymbol{A'_{cl}}$ e $\boldsymbol{B'}$ são, respectivamente, as matrizes de estado e entrada do sistema em malha fechada (D'AZZO; HOUPIS, 1995).

$$\dot{\boldsymbol{x}}' = \boldsymbol{A'_{cl}}\boldsymbol{x}' + \boldsymbol{B'}\boldsymbol{r} \tag{23}$$

A matriz de realimentação $\bar{\boldsymbol{K}}$ pode ser selecionada de modo que o espectro de autovalores de $\boldsymbol{A'_{cl}}$ esteja no semiplano complexo esquerdo. Como a referência $\boldsymbol{r}$ é constante por partes, os valores dos estados $\boldsymbol{x}$ e $\boldsymbol{z}$, em regime, também são constantes. Deste modo, o erro $\dot{\boldsymbol{z}}$ é igual a zero e a Equação (13) indica que a Equação (12) é satisfeita. Isto garante que o vetor de saída controlada $\boldsymbol{w}$ segue o vetor de referência $\boldsymbol{r}$ (D'AZZO; HOUPIS, 1995).

3.4 ATRIBUIÇÃO DE AUTOESTRUTURA COMPLETA

Ao considerar a definição de espaço de estado, apresentada na Seção 3.2, e a equação de estado, apresentada na Equação (8). Um sistema MIMO linear, invariante no tempo e

em malha aberta é representado, no domínio contínuo do tempo, pela equação de estado apresentada na Equação (24). Por conveniência de notação, a dependência no tempo dos vetores $\dot{\boldsymbol{x}}$, $\boldsymbol{x}$ e $\boldsymbol{u}$ não está explicitada (D'AZZO; HOUPIS, 1995).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{24}$$

Sabendo que n é o número de variáveis de estado em $\boldsymbol{x}$ e m o é número de variáveis de controle em $\boldsymbol{u}$. Considerando o vetor de referência de entrada $\boldsymbol{r}$, com dimensão $m \times 1$, e a matriz de ganho da realimentação de estado $\boldsymbol{K}$, com dimensão $m \times n$. A lei de controle de realimentação de estado é definida pela Equação (25) (D'AZZO; HOUPIS, 1995).

$$\boldsymbol{u} = \boldsymbol{r} + \boldsymbol{K}\boldsymbol{x} \tag{25}$$

Ao substituir a Equação (25) na Equação (24) é determinada a equação de estado, em malha fechada, representada pela Equação (26) (D'AZZO; HOUPIS, 1995).

$$\dot{\boldsymbol{x}} = \Big[\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\Big]\,\boldsymbol{x} + \boldsymbol{B}\boldsymbol{r} = \boldsymbol{A_{cl}}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{r} \tag{26}$$

A matriz de estado em malha fechada $\boldsymbol{A_{cl}}$ tem dimensão $n \times n$. O propósito de aplicar a realimentação de estado é poder atribuir tanto um espectro de autovalores de malha fechada, quanto um conjunto de autovetores associados. Ambos são escolhidos a fim de obter as características desejadas de resposta no tempo. O espectro de n autovalores de malha fechada está representado na Equação (27). O conjunto de n autovetores associados é apresentado na Equação (28) (D'AZZO; HOUPIS, 1995).

$$\boldsymbol{\sigma}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) = \Big[\lambda_1, \lambda_2, \ldots, \lambda_n\Big] \tag{27}$$

$$\boldsymbol{v}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) = \Big[\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_{n+m}}\Big] \tag{28}$$

A relação entre os autovalores de malha fechada e os autovetores é apresentada na Equação (29), que pode ser reescrita na forma da Equação (30) (D'AZZO; HOUPIS, 1995).

$$\Big[\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\Big]\,\boldsymbol{v_i} = \lambda_i \boldsymbol{v_i} \quad \textit{para } i = 1,2, \ldots, n \tag{29}$$

$$\Big[\boldsymbol{A} - \boldsymbol{\lambda_i}\boldsymbol{I} \quad \boldsymbol{B}\Big] \begin{bmatrix} \boldsymbol{v_i} \\ \boldsymbol{q_i} \end{bmatrix} = \boldsymbol{S}(\lambda_i) \begin{bmatrix} \boldsymbol{v_i} \\ \boldsymbol{q_i} \end{bmatrix} = \boldsymbol{0} \quad \textit{para } i = 1,2, \ldots, n \tag{30}$$

As matrizes $\boldsymbol{0}$ e $\boldsymbol{I}$ são uma forma conveniente de expressar, respectivamente, matrizes de zeros e identidade. A matriz $\boldsymbol{S}(\lambda_i)$ tem dimensão $n \times (n+m)$. O autovetor $\boldsymbol{v_i}$ tem dimensão $n \times 1$. O vetor $\boldsymbol{q_i}$, definido na Equação (31), tem dimensão $m \times 1$ (D'AZZO; HOUPIS, 1995).

$$\boldsymbol{q_i} = \boldsymbol{K}\boldsymbol{v_i} \tag{31}$$

Para satisfazer a Equação (30), o vetor $\begin{bmatrix} \boldsymbol{v_i^T} & \boldsymbol{q_i^T} \end{bmatrix}^T$ deve pertencer ao *kernel* ou espaço nulo de $\boldsymbol{S}(\lambda_i)$, denotado por $ker\boldsymbol{S}(\lambda_i)$. A partir da resolução da Equação (30), para todo o espectro de n autovalores de malha fechada, e da relação na Equação (31) pode ser obtida a igualdade matricial na Equação (32) (D'AZZO; HOUPIS, 1995).

$$\begin{bmatrix} \boldsymbol{q_1} & \boldsymbol{q_2} & ... & \boldsymbol{q_n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Kv_1} & \boldsymbol{Kv_2} & ... & \boldsymbol{Kv_n} \end{bmatrix} \tag{32}$$

Ao fatorar a Equação (32) é possível determinar a matriz do ganho de realimentação de estado $\boldsymbol{K}$ na Equação (33). A dimensão da matriz $\boldsymbol{Q}$ é $m \times n$. A matriz $\boldsymbol{V}$ tem dimensão $n \times n$ (D'AZZO; HOUPIS, 1995).

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{q_1} & \boldsymbol{q_2} & ... & \boldsymbol{q_n} \end{bmatrix} \begin{bmatrix} \boldsymbol{v_1} & \boldsymbol{v_2} & ... & \boldsymbol{v_n} \end{bmatrix}^{-1} = \boldsymbol{QV^{-1}} \tag{33}$$

Os autovetores $\boldsymbol{v_i}$ selecionados devem ser linearmente independentes tal que $\boldsymbol{V}^{-1}$ exista. Tanto os autovalores λ_i, quanto os autovetores associados $\boldsymbol{v_i}$ e os vetores $\boldsymbol{q_i}$ são reais ou pares complexo-conjugados, sendo descritos como auto-conjugados. Portanto, a matriz de ganho da realimentação de estado $\boldsymbol{K}$ é uma matriz real (D'AZZO; HOUPIS, 1995).

O problema do projeto de um controle seguidor com realimentação de estado, apresentado na Seção 3.3, corresponde a determinação de uma matriz de ganho de realimentação de estado $\bar{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K_1} & \boldsymbol{K_2} \end{bmatrix}$, a fim de atribuir as características de resposta no tempo sobre a equação de estado em malha fechada, apresentada na Equação (21). A aplicação do método de atribuição de autoestrutura completa sobre as matrizes aumentadas de estado $\bar{\boldsymbol{A}}$ e de entrada $\bar{\boldsymbol{B}}$, apresentadas na Equação (16), permite determinar a matriz do ganho de realimentação de estado $\bar{\boldsymbol{K}}$.

3.5 REGULADOR LINEAR QUADRÁTICO

Ao considerar a definição de espaço de estado, apresentada na Seção 3.2, e a equação de estado, apresentada na Equação (8). Um sistema MIMO linear, invariante no tempo e em malha aberta é representado, no domínio contínuo do tempo, pela equação de estado apresentada na Equação (34). Por conveniência de notação, a dependência no tempo dos vetores $\dot{\boldsymbol{x}}$, $\boldsymbol{x}$ e $\boldsymbol{u}$ não está explicitada (LEWIS, 1998; LEWIS; VRABIE; SYRMOS, 2012).

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu} \tag{34}$$

O vetor de estado $\boldsymbol{x} \in R^n$, o vetor de controle $\boldsymbol{u} \in R^m$ e $\boldsymbol{x(0)}$ é a condição inicial. Assumindo que todos as variáveis de estado são mensuráveis, o objetivo é determinar para a lei de controle na Equação (35) a matriz do ganho de realimentação de estado $\boldsymbol{K}$, que atribua as propriedades desejadas em malha fechada (LEWIS, 1998; LEWIS; VRABIE; SYRMOS, 2012).

$$\boldsymbol{u} = -\boldsymbol{Kx} \tag{35}$$

Ao substituir a Equação (35) na Equação (34) obtém-se a equação de estado em malha fechada apresentada na Equação (36), na qual a matriz de estado em malha fechada $\boldsymbol{A_{cl}}$ tem dimensão $n \times n$ (LEWIS, 1998; LEWIS; VRABIE; SYRMOS, 2012).

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{BK})\boldsymbol{x} = \boldsymbol{A_{cl}}\, \boldsymbol{x} \tag{36}$$

No projeto do controle por realimentação de estado ótimo é necessário definir o índice de desempenho quadrático J na Equação (37) (LEWIS, 1998; FRIEDLAND, 1986).

$$J = \frac{1}{2} \int_0^\infty \boldsymbol{x^T Q x} + \boldsymbol{u^T R u}\, dt \tag{37}$$

A matriz de ponderação dos estados $\boldsymbol{Q}$, com dimensão $(n \times n)$, deve ser positiva semi-definida. Se $\boldsymbol{Q}$ for uma matriz diagonal, todos seus termos devem ser positivos, com possíveis zeros na diagonal principal. A matriz de ponderação do controle $\boldsymbol{R}$, com dimensão $(m \times m)$, deve ser positiva definida e inversível. Se $\boldsymbol{R}$ for uma matriz diagonal, todos seus termos devem ser positivos. Desta forma, a quantidade escalar $\boldsymbol{x^T Q x}$ é sempre positiva ou zero, em cada instante t, para todas as funções $\boldsymbol{x}(t)$. A quantidade escalar $\boldsymbol{u^T R u}$ é sempre positiva, em cada instante t, para todos os valores de $\boldsymbol{u}(t)$. Tudo isto garante ao índice de desempenho J um valor definido (LEWIS, 1998; FRIEDLAND, 1986).

A substituição da Equação (35) na Equação (37) resulta no índice de desempenho quadrático J, apresentado na Equação (38), em função da matriz do ganho de realimentação de estado $\boldsymbol{K}$. O problema da determinação de uma matriz $\boldsymbol{K}$ para minimizar J é conhecido por Regulador Linear Quadrático (LQR), uma vez que o sistema na Equação (34) é linear e o índice de desempenho J é quadrático. O termo "regulador" indica que o controle por realimentação de estado regula o vetor de estado $\boldsymbol{x}$ para zero (LEWIS, 1998).

$$J = \frac{1}{2} \int_0^\infty \boldsymbol{x^T}(\boldsymbol{Q} + \boldsymbol{K^T R K})\boldsymbol{x}\, dt \tag{38}$$

O índice de desempenho quadrático J pode ser interpretado como uma função de energia, torná-lo pequeno mantém a energia total do sistema em malha fechada pequena. O vetor de estado $\boldsymbol{x}$ e o vetor de controle $\boldsymbol{u}$ são pesos de J. Então, para um valor pequeno de J, $\boldsymbol{x}$ e $\boldsymbol{u}$ não podem ser muito grandes. O índice de desempenho J é finito, se minimizado, e por ser uma integral infinita de $\boldsymbol{x}(t)$ indica que quando t tende a infinito $\boldsymbol{x}(t)$ vai para zero. Isto garante a estabilidade do sistema em malha fechada (LEWIS, 1998).

A seleção das matrizes $\boldsymbol{Q}$ e $\boldsymbol{R}$ é tarefa do projetista do controlador, podendo obter sistemas em malha fechada com diferentes tipos de resposta. O termo $\boldsymbol{x^T Q x}$ representa a penalidade no desvio do vetor de estado $\boldsymbol{x}$, com relação a sua condição inicial $\boldsymbol{x}(0)$. Na seleção de um $\boldsymbol{Q}$ grande, o vetor de estado $\boldsymbol{x}$ deve ser pequeno a fim de manter o índice de desempenho J pequeno. Isto significa que os polos da matriz de estado em malha fechada $\boldsymbol{A_{cl}}$ são alocados mais à esquerda do semiplano complexo esquerdo. Deste modo, os estados decaem mais rápido para zero (LEWIS, 1998; FRIEDLAND, 1986).

O termo $\boldsymbol{u^T R u}$ representa o "custo do controle". Na seleção de um $\boldsymbol{R}$ grande, o vetor de controle $\boldsymbol{u}$ deve ser pequeno a fim de manter o índice de desempenho J pequeno. Isto significa uma redução no esforço de controle. Ou seja, os polos da matriz de estado em malha fechada $\boldsymbol{A_{cl}}$ são mais lentos, em geral, resultando em maiores valores para o vetor de estado $\boldsymbol{x}$ (LEWIS, 1998; FRIEDLAND, 1986).

Para determinar a matriz do ganho de realimentação de estado $\boldsymbol{K}$ ótima, supõe-se a existência de uma matriz constante $\boldsymbol{P}$, presente na Equação (39) (LEWIS, 1998).

$$\frac{d}{dt}(\boldsymbol{x^T P x}) = -\boldsymbol{x^T}(\boldsymbol{Q} + \boldsymbol{K^T R K})\boldsymbol{x} \tag{39}$$

A substituição da Equação (39) na Equação (38) resulta no índice de desempenho quadrático J, apresentado na Equação (40), dependente apenas da matriz auxiliar $\boldsymbol{P}$ e da condição inicial $\boldsymbol{x}(0)$. Além disso, assume-se que o sistema em malha fechada é estável, então $\boldsymbol{x}(t)$ vai para zero conforme t tende ao infinito (LEWIS, 1998).

$$J = -\frac{1}{2}\int_0^\infty \frac{d}{dt}(\boldsymbol{x^T P x})\, dt = \frac{1}{2}\boldsymbol{x^T}(0)\boldsymbol{P x}(0) \tag{40}$$

Ao resolver a derivada à esquerda da igualdade, a relação na Equação (39) pode ser reescrita como na Equação (41) (LEWIS, 1998).

$$\boldsymbol{\dot{x}^T P x} \; + \; \boldsymbol{x^T P \dot{x}} \; + \; \boldsymbol{x^T Q x} \; + \boldsymbol{x^T K^T R K x} = 0 \tag{41}$$

Na determinação da matriz do ganho de realimentação de estado $\boldsymbol{K}$, de modo que a suposição na Equação (39) se cumpra. A equação de estado em malha fechada, apresentada na Equação (36), é substituída na Equação (41), resultando na Equação (42) (LEWIS, 1998).

$$\boldsymbol{x^T}(\boldsymbol{A_{cl}^T P + P A_{cl} + Q + K^T R K})\boldsymbol{x} = 0 \tag{42}$$

Como a relação na Equação (42) deve se manter para qualquer $\boldsymbol{x}$, é possível determinar que o termo entre parêntesis é igual a zero, como apresentado na Equação (43) (LEWIS, 1998).

$$\boldsymbol{A_{cl}^T P + P A_{cl} + Q + K^T R K} = 0 \tag{43}$$

A substituição do termo $\boldsymbol{A_{cl}}$, na Equação (43), por $(\boldsymbol{A - BK})$ resulta em uma equação quadrática matricial representada na Equação (44) (LEWIS, 1998).

$$\boldsymbol{A^T P + P A + Q + K^T R K - K^T B^T P - P B K} = 0 \tag{44}$$

Como não é trivial o processo de determinação da matriz do ganho de realimentação de estado $\boldsymbol{K}$, que minimiza o índice de desempenho quadrático J na Equação (38), é suposta a solução na Equação (45) (LEWIS, 1998; LEWIS; VRABIE; SYRMOS, 2012).

$$\boldsymbol{K = R^{-1} B^T P} \tag{45}$$

Ao substituir a Equação (45) na Equação (44) e simplificar, é obtida a equação quadrática matricial representada pela Equação (46), denominada equação algébrica de Riccati (LEWIS, 1998; LEWIS; VRABIE; SYRMOS, 2012).

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} = 0 \tag{46}$$

Dadas as matrizes de estado $\boldsymbol{A}$, de entrada $\boldsymbol{B}$, de ponderação dos estados $\boldsymbol{Q}$ e de ponderação do controle $\boldsymbol{R}$, a equação algébrica de Riccati pode ser resolvida para determinar a matriz auxiliar $\boldsymbol{P}$. Uma solução $\boldsymbol{P} > 0$ existe, se o par de matrizes $(\boldsymbol{A},\boldsymbol{B})$ respeitar a condição de controlabilidade na Equação (47), e o par de matrizes $(\boldsymbol{A},\sqrt{\boldsymbol{Q}})$ respeitar a condição de observabilidade na Equação (48). A matriz do ganho de realimentação de estado $\boldsymbol{K}$ ótimo é, então, obtida pela Equação (45). O valor mínimo do índice de desempenho quadrático J é dado pela Equação (40), dependente apenas da matriz auxiliar $\boldsymbol{P}$ e da condição inicial $\boldsymbol{x}(0)$. Deste modo, o custo do controle por realimentação de estado para $\boldsymbol{K}$ pode ser calculado, antes mesmo da aplicação no sistema, por meio da condição inicial de $\boldsymbol{x}$ (LEWIS, 1998; LEWIS; VRABIE; SYRMOS, 2012).

$$posto \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^{\boldsymbol{2}}\boldsymbol{B} & \cdots & \boldsymbol{A}^{\boldsymbol{n-1}}\boldsymbol{B} \end{bmatrix} = n \tag{47}$$

$$posto \begin{bmatrix} \sqrt{\boldsymbol{Q}} \\ \sqrt{\boldsymbol{Q}}\boldsymbol{A} \\ \sqrt{\boldsymbol{Q}}\boldsymbol{A}^{\boldsymbol{2}} \\ \vdots \\ \sqrt{\boldsymbol{Q}}\boldsymbol{A}^{\boldsymbol{n-1}} \end{bmatrix} = n \tag{48}$$

O problema do projeto de um controle seguidor com realimentação de estado, apresentado na Seção 3.3, corresponde a determinação de uma matriz de ganho de realimentação de estado $\bar{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K_1} & \boldsymbol{K_2} \end{bmatrix}$, a fim de atribuir as características de resposta no tempo sobre a equação de estado em malha fechada, apresentada na Equação (22). A aplicação do método LQR sobre as matrizes aumentadas de estado $\bar{\boldsymbol{A}}$ e de entrada $\bar{\boldsymbol{B}}$, apresentadas na Equação (16), permite determinar a matriz do ganho de realimentação de estado $\bar{\boldsymbol{K}}$.

3.6 OBSERVADOR DE LUENBERGER

Ao considerar a definição de espaço de estado, apresentada na Seção 3.2, e as equações de estado e saída, respectivamente, nas Equações (8) e (9). Uma planta MIMO é representada pelas equações de estado e saída apresentadas, respectivamente, nas Equações (49) e (50). Por conveniência de notação, a dependência no tempo dos vetores $\dot{\boldsymbol{x}}$, $\boldsymbol{x}$, $\boldsymbol{u}$ e $\boldsymbol{y}$ não está explicitada (D'AZZO; HOUPIS; SHELDON, 2003).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{49}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} \tag{50}$$

No projeto do controle por realimentação de estado pelos métodos de atribuição de autoestrutura completa, apresentado na Seção 3.4, e LQR, apresentado na Seção 3.5. Assume-se que todas as n componentes do vetor de estado $\boldsymbol{x}$ da planta são mensuráveis. Nos casos em que não é possível mensurar o vetor de estado $\boldsymbol{x}$, é possível estimá-lo por meio das l componentes do vetor de saída $\boldsymbol{y}$ da planta. Esta capacidade depende de todas as variáveis de estado da planta, representada pelas Equações (49) e (50), serem observáveis. Isto ocorre quando a condição na Equação (51) é verdadeira (D'AZZO; HOUPIS; SHELDON, 2003).

$$posto \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA^2} \\ \vdots \\ \boldsymbol{CA^{n-1}} \end{bmatrix} = n \tag{51}$$

Considerando um vetor $\hat{\boldsymbol{x}}$, denominado vetor de estado estimado, e sua derivada temporal $\dot{\hat{\boldsymbol{x}}}$, ambos com dimensão $n \times 1$. Um vetor $\hat{\boldsymbol{y}}$, denominado vetor de saída estimado, com dimensão $l \times 1$, e uma matriz $\boldsymbol{L}$, denominada matriz do observador, com dimensão $n \times l$. As equações de estado e de saída do observador de Luenberger são definidas, respectivamente, pelas Equações (52) e (53). A Figura 14 apresenta o diagrama de blocos da planta em conjunto com o observador de Luenberger (D'AZZO; HOUPIS; SHELDON, 2003).

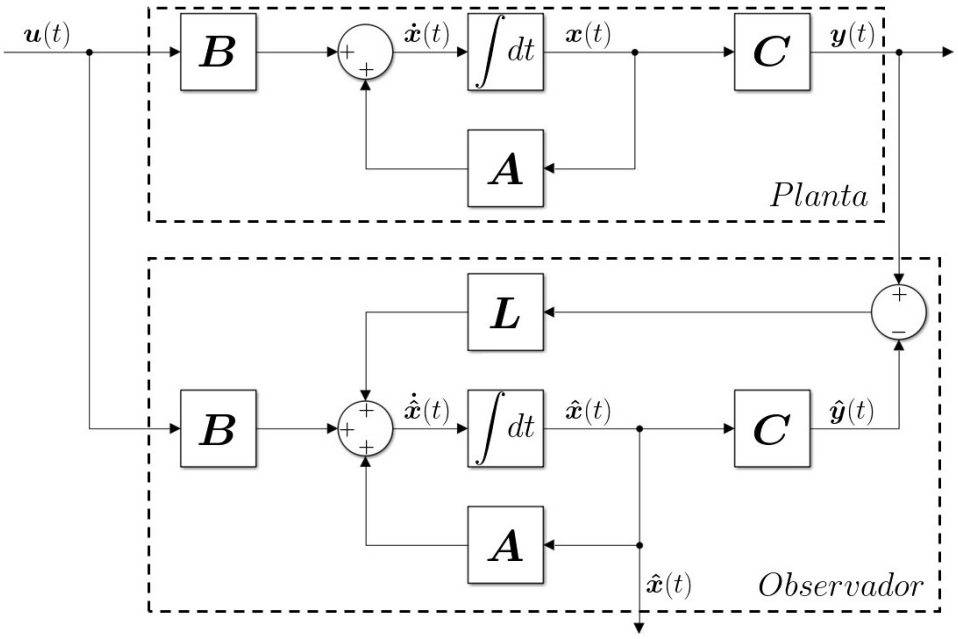
$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{52}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{C}\hat{\boldsymbol{x}} \tag{53}$$

O objetivo do observador de Luenberger, representado pelas Equações (52) e (53), é obter uma estimativa $\hat{\boldsymbol{x}}$ do vetor de estado $\boldsymbol{x}$ da planta, representada pelas Equações (49) e (50). O método do observador de Luenberger consiste em simular a dinâmica da planta, por meio do mesmo vetor de controle $\boldsymbol{u}$ sendo aplicada sobre a planta. Sendo assim, o vetor de controle $\boldsymbol{u}$ e a dinâmica, representada pelas matrizes da notação de espaço de estado $\boldsymbol{A}$, $\boldsymbol{B}$ e $\boldsymbol{C}$ são iguais no sistema do observador, dado pelas Equações (52) e (53), e na planta, apresentada nas Equações (49) e (50) (D'AZZO; HOUPIS; SHELDON, 2003).

Para os vetores de estado da planta $\boldsymbol{x}$ e do observador $\hat{\boldsymbol{x}}$ serem iguais, as condições iniciais $\boldsymbol{x}(0)$ e $\hat{\boldsymbol{x}}(0)$ devem ser iguais. Porém, a planta pode estar sujeita a perturbações inexistentes na simulação do observador. Por conta disto, o termo $\boldsymbol{L}(\boldsymbol{y} - \hat{\boldsymbol{y}})$ é adicionado na equação de estado do observador, apresentada pela Equação (52). Desta forma, a diferença entre os vetores de saída da planta $\boldsymbol{y}$ e do observador $\hat{\boldsymbol{y}}$ multiplicada pela matriz $\boldsymbol{L}$, projetada para o sistema do observador, permite compensar as diferenças entre o processo real da planta e a simulação de sua dinâmica (D'AZZO; HOUPIS; SHELDON, 2003).

Figura 14 – Diagrama de blocos de uma planta com variáveis de estado inacessíveis em conjunto com um observador de Luenberger.



Fonte: Autoria própria.

A sintetização da matriz do observador $\boldsymbol{L}$ depende do vetor $\boldsymbol{e}$, denominado erro de reconstrução do observador, e de sua derivada temporal $\dot{\boldsymbol{e}}$, ambos com dimensão $n \times 1$. A definição dos vetores $\boldsymbol{e}$ e $\dot{\boldsymbol{e}}$ está representada, respectivamente, nas Equações (54) e (55) (D'AZZO; HOUPIS; SHELDON, 2003).

$$\boldsymbol{e} \equiv \boldsymbol{x} - \hat{\boldsymbol{x}} \tag{54}$$

$$\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}} - \dot{\hat{\boldsymbol{x}}} \tag{55}$$

A partir da planta, representada nas Equações (49) e (50), do observador, dado pelas Equações (52) e (53), e das definições do erro de reconstrução do observador, nas Equações (54) e (55). É possível obter a equação de estado do erro do observador, apresentada na Equação (56) (D'AZZO; HOUPIS; SHELDON, 2003).

$$\dot{\boldsymbol{e}} = (\boldsymbol{A} - \boldsymbol{LC})\boldsymbol{e} \tag{56}$$

A equação de estado do erro do observador não possui entrada, sendo excitada apenas pela condição inicial. A escolha apropriada da matriz do observador $\boldsymbol{L}$ permite que todos os autovalores de $\boldsymbol{A} - \boldsymbol{LC}$ sejam atribuídos ao semiplano complexo esquerdo, usualmente mais à esquerda dos autovalores da matriz de estado $\boldsymbol{A}$. Deste modo, o resultado da Equação (57) é verdadeiro para qualquer condição inicial (D'AZZO; HOUPIS; SHELDON, 2003).

$$\lim_{t \to \infty} \boldsymbol{e}(t) = 0 \tag{57}$$

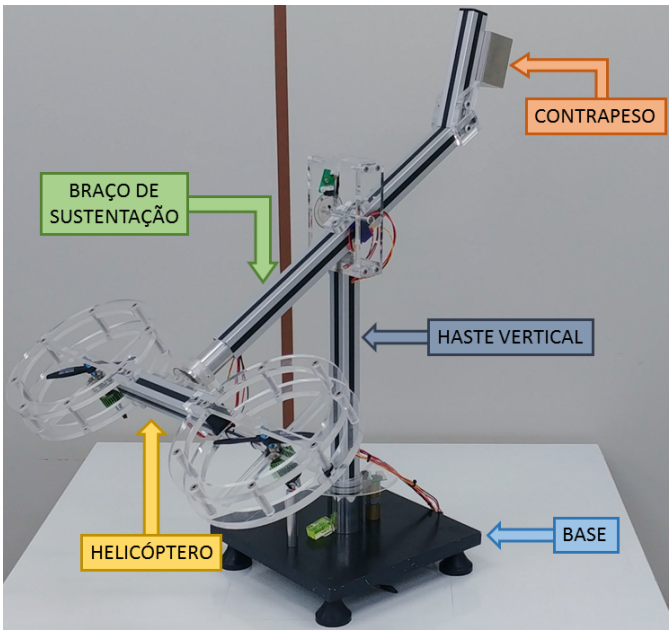
4 HELICÓPTERO COM 3 GRAUS DE LIBERDADE

Neste capítulo são apresentados os conteúdos relacionados a descrição, modelagem e instrumentação do Helicóptero com 3 GDL utilizado neste trabalho. Na descrição do sistema é explicado, de forma geral, o conceito e objetivo do Helicóptero com 3 GDL. Quanto a modelagem matemática é apresentado o desenvolvimento da prototipagem virtual do sistema. Com relação a instrumentação da planta, são apresentados os programas e equipamentos responsáveis pela aquisição e controle, a identificação dos atuadores, e a configuração dos sensores.

4.1 DESCRIÇÃO DO SISTEMA

A Figura 15 apresenta o Helicóptero com 3 GDL, utilizado neste trabalho, e destaca cinco partes importantes, a base, a haste vertical, o braço de sustentação, o helicóptero e o contrapeso.

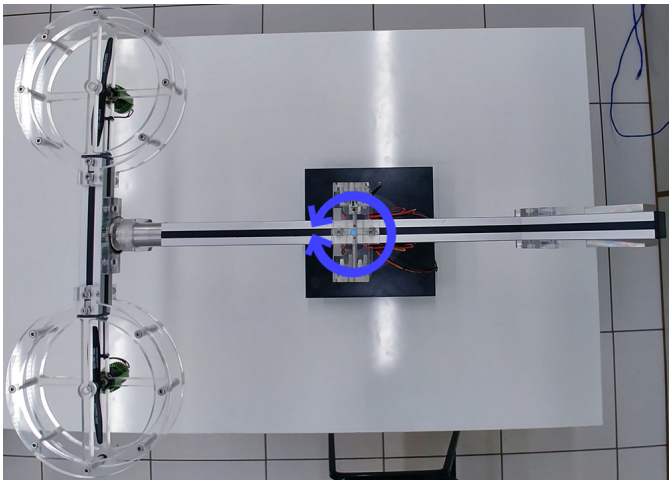
Figura 15 – Helicóptero com 3 GDL.



Fonte: Autoria própria.

A haste vertical se conecta a base por meio de uma junta de revolução, o movimento desta representa o grau de liberdade denominado deslocamento, apresentado na Figura 16. O ângulo formado entre a haste vertical e a base é a posição angular do movimento de deslocamento. Adotando a mesma convenção de Apkarian, Lévis e Fulford (2012), se o movimento ilustrado na Figura 16 for no sentido anti-horário a posição angular do deslocamento é incrementada, caso contrário é decrementada.

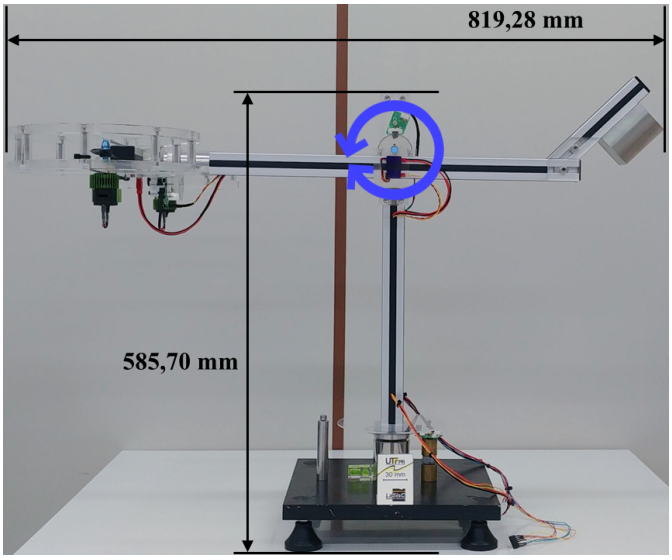
Figura 16 – Movimento de deslocamento.



Fonte: Autoria própria.

O grau de liberdade denominado elevação, representado na Figura 17, é proveniente do movimento da junta de revolução conectando o braço de sustentação à haste vertical. A posição angular da elevação é dada pelo ângulo formado entre o braço de sustentação e a haste vertical. Segundo a convenção de Apkarian, Lévis e Fulford (2012), se o movimento ilustrado na Figura 17 for no sentido horário, a posição angular da elevação é incrementada, caso contrário é decrementada. Quanto a posição inicial do braço de sustentação, foi considerado que a posição angular da elevação é zero quando o braço de sustentação está apoiado na haste vertical (como visto na Figura 15), no caso ilustrado na Figura 17 a posição angular da elevação é aproximadamente 0,5458 radianos (31,27°). A Figura 17 também apresenta algumas das dimensões da planta real.

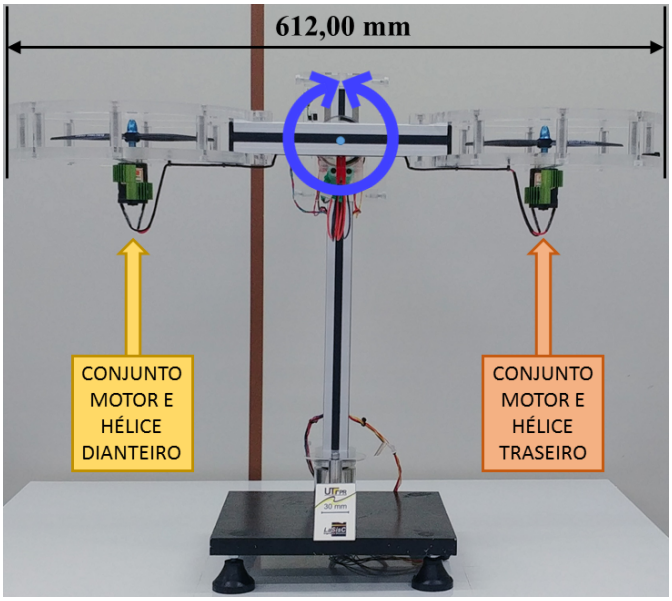
Figura 17 – Movimento de elevação.



Fonte: Autoria própria.

A Figura 18 apresenta o grau de liberdade denominado arfagem, resultante do movimento da junta de revolução conectando o helicóptero ao braço de sustentação. A posição angular da arfagem é dada pelo ângulo entre o helicóptero e o braço de sustentação, sendo igual a zero para a posição representada na Figura 18. Para a mesma convenção de Apkarian, Lévis e Fulford (2012), se o movimento ilustrado na Figura 18 for no sentido horário a posição angular da arfagem é incrementada, caso contrário é decrementada. A Figura 18 também apresenta o comprimento do helicóptero.

Figura 18 – Movimento de arfagem.



Fonte: Autoria própria.

Na Figura 18, segundo a convenção de Apkarian, Lévis e Fulford (2012), são indicados os conjuntos de motor e hélice dianteiro e traseiro responsáveis por produzir as forças de empuxo que atuam sobre o helicóptero. Para esta planta, foi instalado um novo par de hélices, com diâmetro de 6" e passo de 3", que giram em sentidos contrários e apenas no sentido que produz forças de empuxo para elevação do helicóptero. Variar igualmente as forças de empuxo do helicóptero produz o movimento de elevação. Causar um desequilíbrio entre as forças de empuxo produz um movimento de arfagem, este, por sua vez, produz uma componente de força que proporciona ao sistema o movimento de deslocamento. O contrapeso fixado no braço de sustentação tem o propósito de reduzir a força necessária na atuação do sistema.

Resumidamente, uma força de empuxo dianteiro maior que a força de empuxo traseiro resulta em um ângulo de arfagem positivo, que, por sua vez, resulta em um ângulo de deslocamento positivo. Uma força de empuxo traseiro maior que uma força de empuxo dianteiro produz um ângulo de arfagem negativo, que, por sua vez, produz um ângulo de deslocamento negativo. Um aumento igual das forças de empuxo dianteiro e traseiro incrementa o ângulo de elevação, assim como uma redução igual das forças de empuxo dianteiro e traseiro decrementa o ângulo de elevação.

4.2 TÉCNICA DE MODELAGEM DE SISTEMAS MULTICORPOS

A definição de MBS é um conjunto de corpos rígidos cinematicamente restringidos por diferentes tipos de juntas, sendo cada corpo rígido capaz de realizar grandes deslocamentos translacionais e rotacionais. O termo corpo rígido implica que a deformação do corpo não apresenta efeito sobre seu movimento bruto, ou seja, a distância entre quaisquer duas partículas do corpo se mantém constante em qualquer instante e configuração (SHABANA, 2013).

Um MBS é também definido como um sistema mecânico que apresenta muitos GDL, tal que os movimentos são governados por equações dinâmicas. A resolução do problema de modelagem permite determinar as equações dinâmicas do movimento de um MBS, porém existem vários níveis de complexidade envolvidos. A capacidade de executar as seguintes etapas de um processo de modelagem é uma qualidade necessária aos cientistas e engenheiros (MONTEZUMA, 2003; BREGANON, 2009):

1. Descrever o modelo físico de um sistema contendo os aspectos relevantes e hipóteses simplificadoras;
2. Obter as equações que descrevem matematicamente a dinâmica do sistema;
3. Resolver as equações analítica ou numericamente a fim de estimar o comportamento do sistema;
4. Verificar os resultados do modelo por meio da comparação com o sistema real;
5. Verificar a necessidade de modificações no sistema físico, ou usá-lo para projeto.

A técnica MBS denominada prototipagem virtual pode ser aplicada na análise e projeto de qualquer sistema mecânico, desde que possa ser modelado como um conjunto de corpos rígidos interconectado por juntas, sob a influência de forças, dirigido por movimentos prescritos e limitado por vínculos. A prototipagem virtual auxilia na execução das etapas 2 e 3 do processo de modelagem de um sistema. Deste modo, os cientistas e engenheiros podem se concentrar nas etapas 1, 4 e 5 do processo (MONTEZUMA, 2003; BREGANON, 2009).

Em uma abordagem tradicional, as empresas constroem protótipos a fim de aprender sobre sistemas complexos, isto possibilita tomar decisões críticas sobre o projeto do sistema. O aprendizado de um sistema a partir de um protótipo real é composto das seguintes fases (MONTEZUMA, 2003; BREGANON, 2009):

1. Projeto, construção e montagem do sistema completo;
2. Instrumentação do sistema;
3. Execução de testes;
4. Coleta de dados;
5. Interpretação dos dados;
6. Por fim, tomada de decisões do projeto, as quais envolvem frequentemente mudanças no protótipo real e repetição das fases anteriores.

No caso do aprendizado sobre o sistema pela abordagem da prototipagem virtual, o processo é composto pelas seguintes fases (MONTEZUMA, 2003; BREGANON, 2009):

1. Construção (modelagem) do sistema;
2. Instrumentação, por meio da solicitação de determinadas variáveis de saída ao software MBS;
3. Execução de testes, por meio de simulações;
4. Comparação visual do desempenho de várias alternativas de projeto por meio de animações gráficas;
5. Interpretação dos dados coletados automaticamente, organizados e exibidos em gráficos;
6. Tomada de decisões de projeto com maior rapidez.

O processo da prototipagem virtual permite focar nos aspectos criativos do projeto, interpretação de dados e tomada de novas decisões. Em geral, os softwares MBS são adaptados para a necessidade de alterar o protótipo, gerar novos dados de saída e repetir a execução de testes. Para a prototipagem real, o processo de reconstrução e instrumentação do protótipo consome muito do tempo de desenvolvimento do sistema. Porém, é importante destacar que a prototipagem virtual não descarta a necessidade de construção de um protótipo real. Após o processo da prototipagem virtual, pode-se construir um protótipo real com a necessidade de um número reduzido de testes. Os dados coletados do protótipo real permitem validar o processo da prototipagem virtual (MONTEZUMA, 2003; BREGANON, 2009).

4.3 SOLIDWORKS

O software SolidWorks comercializado pela Dassault Systèmes SolidWorks Corporation é uma ferramenta CAD (*Computer Aided Design*) tridimensional (3D) para automação de projetos mecânicos, permitindo a rápida produção de modelos e desenhos detalhados, a análise destes em diferentes vistas e a experimentação com dimensões. Por meio do software SolidWorks Simulation, o SolidWorks também pode ser considerado uma ferramenta CAE (*Computer Aided Engineering*) capaz de, por exemplo, realizar análises estáticas e dinâmicas, sendo lineares ou não-lineares (SOUZA *et al.*, 2003; DASSAULT SYSTÈMES, 2019).

A utilização do SolidWorks permite construir um protótipo digital 3D facilitando a visualização, simulação e análise do comportamento por meio da simulação de condições reais. A realização destas etapas antes da construção do sistema real permite uma maior rapidez de desenvolvimento, com a necessidade de construir uma quantidade menor de protótipos reais (SILVA, 2011).

4.4 ADAMS

O software Adams comercializado pela MSC Software Corporation é uma ferramenta CAE para análise da cinemática e dinâmica de sistemas multicorpos complexos, assim como para determinação da distribuição de esforços, gerada pela ação de um determinado sistema de forças sob determinadas condições de funcionamento. O Adams é um sistema modular com *plug-ins* que atendem as diferentes necessidades do usuário. O módulo padrão é o Adams/View, que permite realizar o estudo estático, cinemático e dinâmico, e o refinamento dos protótipos virtuais mecânicos por meio de simulações obtidas pelos módulos Adams/Solver e Adams/Postprocessor (NUNES; SILVA, 2014).

A base das técnicas MBS na determinação das equações dinâmicas para simulação são as leis da física, no caso do Adams é utilizado método de Euler-Lagrange. Na determinação do modelo mecânico para uma simulação, o Adams depende dos seguintes requisitos (MONTEZUMA, 2003; BREGANON, 2009):

1. Definição das características inerciais dos corpos;
2. Definição da interação entre os corpos;
3. Definição dos movimentos e forças atuando sobre o sistema.

Estes requisitos são atendidos por meio da entrada, no Adams, dos seguintes parâmetros e configurações (MONTEZUMA, 2003; BREGANON, 2009):

- Propriedades de massa e inércia dos corpos;
- Configuração dos aspectos geométricos do sistema, ou seja, posicionamento dos corpos e de seus respectivos centros de massa e a especificação dos pontos onde se aplicam os movimentos e forças;
- Configuração das juntas mecânicas que conectam os corpos, além de outros vínculos e elementos elásticos;
- Configuração das forças externas e excitações agindo sobre o sistema;
- Configuração de atributos gráficos para visualização do comportamento do sistema, por meio de animações.

Este processo de entrada de parâmetros e configurações é simplificado pelo módulo Adams/View por uma série de bibliotecas padrão para inserção de sistemas de coordenadas, juntas mecânicas, movimentos e forças. Antes de executar a simulação o Adams analisa os dados de entrada a fim de verificar se a composição do sistema mecânico está completa, correta e consistente (MONTEZUMA, 2003; BREGANON, 2009).

4.4.1 Tipos de análise

O Adams permite a realização de seis tipos diferentes de análise: a de condições iniciais, a cinemática, a de equilíbrio estático, a quase-estática, a dinâmica e a linear. Cada uma destas análises tem sua utilidade, dependendo do contexto e do interesse no desenvolvimento do sistema (MONTEZUMA, 2003; BREGANON, 2009).

A análise de equilíbrio estático permite determinar um estado para o sistema balancear todas as forças internas e externas na ausência de quaisquer movimentos ou forças inerciais. Frequentemente, esta análise é aplicada para determinar o ponto de partida de uma análise dinâmica, removendo assim transitórios indesejados no início da simulação (MONTEZUMA, 2003; BREGANON, 2009).

A análise dinâmica é a mais complexa dentre as análises. Uma série de integradores diferentes está disponível para encontrar a solução de um sistema completo de equações diferenciais e algébricas. Esta análise determina a solução no tempo de todos os deslocamentos, velocidades, acelerações e forças de reação internas do sistema mecânico, estando este sob a ação de um conjunto de forças e excitações externas (MONTEZUMA, 2003; BREGANON, 2009).

Por meio da análise linear, o Adams lineariza o sistema de equações não-lineares, que descreve o sistema mecânico, em um ponto de operação. Isto resulta em um sistema de equações diferenciais de primeira ordem e invariantes no tempo fornecido na notação de espaço de estado (apresentada na Seção 3.2), além do cálculo dos modos e autovalores do modelo (MONTEZUMA, 2003; BREGANON, 2009).

4.5 SISTEMAS DE AQUISIÇÃO E CONTROLE

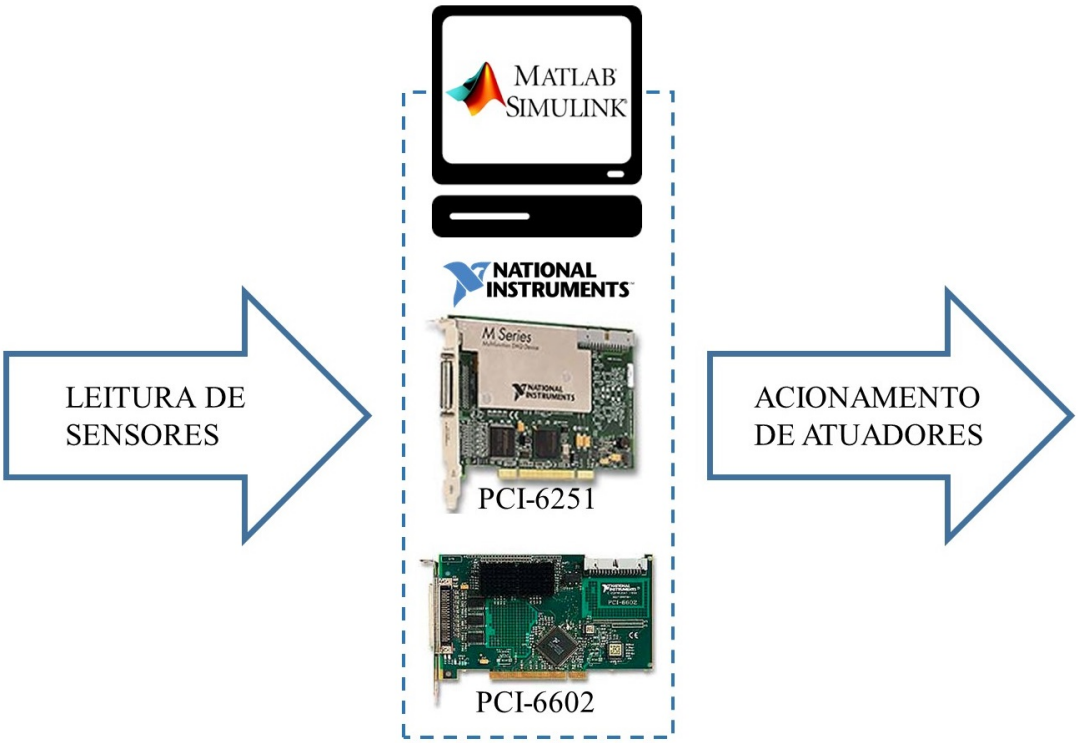
A área de aquisição de dados e controle envolve um amplo conjunto de atividades com o objetivo de construir uma interface entre um computador e o mundo real. A aquisição de dados trata do processo da coleta automatizada de informações proveniente de sensores, por meio da leitura de sinais analógicos ou digitais. Estes sinais podem representar as variáveis de estado de um sistema físico, como posição ou orientação. Uma vez que estes sinais estão dentro de um computador é possível processá-los, por exemplo, em um algoritmo de controle, a fim de produzir sinais para acionamento de processos físicos (JAMES, 2000; AUSTERLITZ, 2003; SHIMADA, 2015).

Os sistemas de aquisição e controle são formados por tecnologias de hardware e software. A parte de hardware permite a ligação física com o mundo exterior, dependendo das necessidades dos ensaios ou experimentos de controle é necessário alterar o hardware utilizado. A parte de software permite o processamento, análise, visualização e armazenamento dos sinais coletados pelo hardware (JAMES, 2000; AUSTERLITZ, 2003; NATIONAL INSTRUMENTS, 2019; SHIMADA, 2015).

No ensaio de identificação dos atuadores, foi necessário um hardware com capacidade simultânea de leitura de sensor analógico e acionamento por PWM (*Pulse Width Modulation* ou Modulação de Largura de Pulso, em português), deste modo o hardware utilizado foi a PCI-6251 da National Instruments. Para os experimentos de controle, foi necessário um hardware com capacidade simultânea de decodificação de três sensores do tipo *encoder* e acionamento de dois atuadores por PWM, sendo assim o hardware utilizado foi a PCI-6602 da National Instruments (NATIONAL INSTRUMENTS, 2016; NATIONAL INSTRUMENTS, 2009).

O computador utilizado possui uma placa-mãe Asus P5QL SE, processador Core 2 Quad Q9650, 4GB de memória RAM, 240 GB de armazenamento SSD Kingston A400 e sistema operacional Windows 8.1. O software utilizado para o processamento, análise, visualização e armazenamento dos sinais coletados pelos hardwares da National Instruments foi o MATLAB/Simulink R2016b. Este software também foi utilizado para fechamento das malhas de controle do Helicóptero com 3 GDL, tanto do sistema físico, quanto das simulações linear e não linear, possíveis devido ao protótipo virtual do Adams. A Figura 19 apresenta um diagrama com as tecnologias de hardware e software, que constituem o sistema tradicional de aquisição e controle utilizado neste trabalho.

Figura 19 – Diagrama das tecnologias tradicionais de sistemas de aquisição e controle.

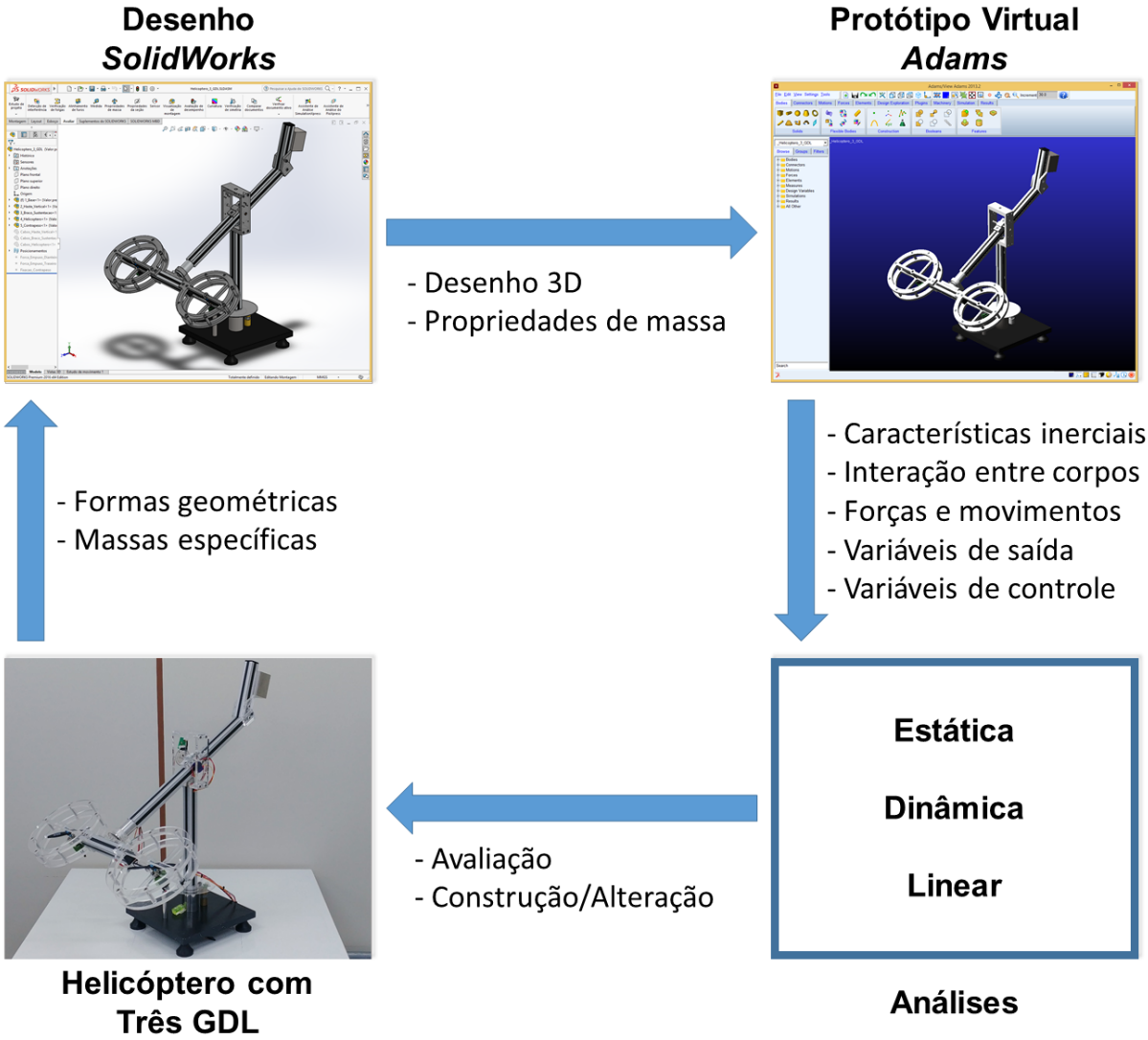


Fonte: Autoria própria.

4.6 PROTOTIPAGEM VIRTUAL DO HELICÓPTERO COM 3 GRAUS DE LIBERDADE

A metodologia de prototipagem virtual utilizada neste trabalho é a mesma apresentada em Shimada (2015), na qual o SolidWorks é utilizado como uma ferramenta CAD para o desenho completo do sistema e o Adams é o software responsável pela análise e modelagem MBS. A visão geral do processo de desenvolvimento do protótipo virtual do Helicóptero com 3 GDL está representada no diagrama da Figura 20.

Figura 20 – Diagrama do processo de prototipagem virtual do Helicóptero com 3 GDL.



Fonte: Autoria própria.

Partindo do Helicóptero com 3 GDL desmontado, cada uma das peças tem suas características geométricas medidas, por meio de um paquímetro com resolução de 0,02 mm e uma trena com resolução de 1 mm, de modo que seja possível construí-las de forma simplificada no SolidWorks. A massa específica de cada peça é calculada por meio de seu volume, determinado pelo SolidWorks, e de sua massa, medida por uma balança digital com resolução de 0,1 g. Com o desenho 3D de todas as peças, configuradas com suas respectivas

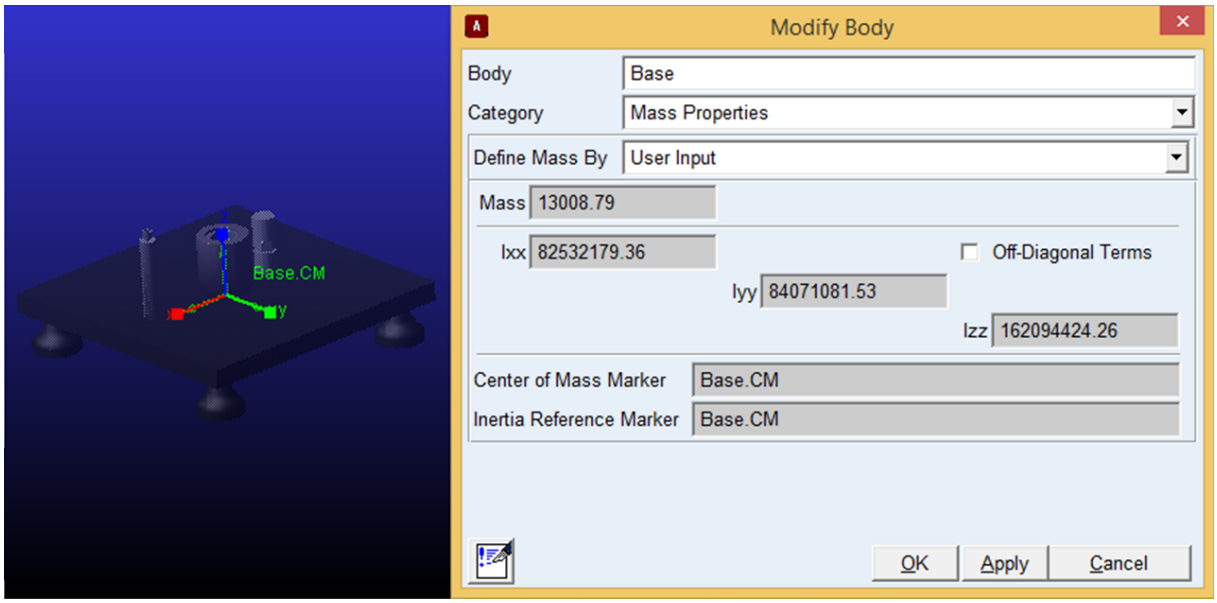
massas específicas, foi possível construir uma montagem completa do Helicóptero com 3 GDL no SolidWorks.

A montagem completa do Helicóptero com 3 GDL foi dividida em múltiplos corpos rígidos, cada um destes representando as partes principais do sistema, descritas na Figura 15, sendo a base, a haste vertical, o braço de sustentação, o helicóptero e o contrapeso. Para cada um dos corpos rígidos foi determinado as propriedades de centro de massa, massa e momento de inércia, por meio do SolidWorks. Além disso, o desenho 3D de cada um dos corpos foi exportado em um formato comum ao SolidWorks e o Adams.

Assim que cada um dos corpos rígidos foi importado no protótipo virtual do Adams, foram realizadas as configurações das características inerciais. As Figuras de 21 até 25 apresentam, respectivamente, as características inerciais da base, haste vertical, braço de sustentação, helicóptero e contrapeso. À esquerda das Figuras de 21 até 25 está representado o sistema de coordenadas do centro de massa, que indica os eixos principais de inércia de cada corpo rígido. À direita das Figuras de 21 até 25 está indicado a massa (g) e os momentos de inércia principais ($g.mm^2$) com relação ao sistema de coordenadas do centro de massa.

O primeiro corpo rígido importado no protótipo virtual do Adams foi a base, apresentada na Figura 21. Por meio de uma junta fixa, a base foi presa ao solo. Isto mantém todo o Helicóptero com 3 GDL fixo no espaço.

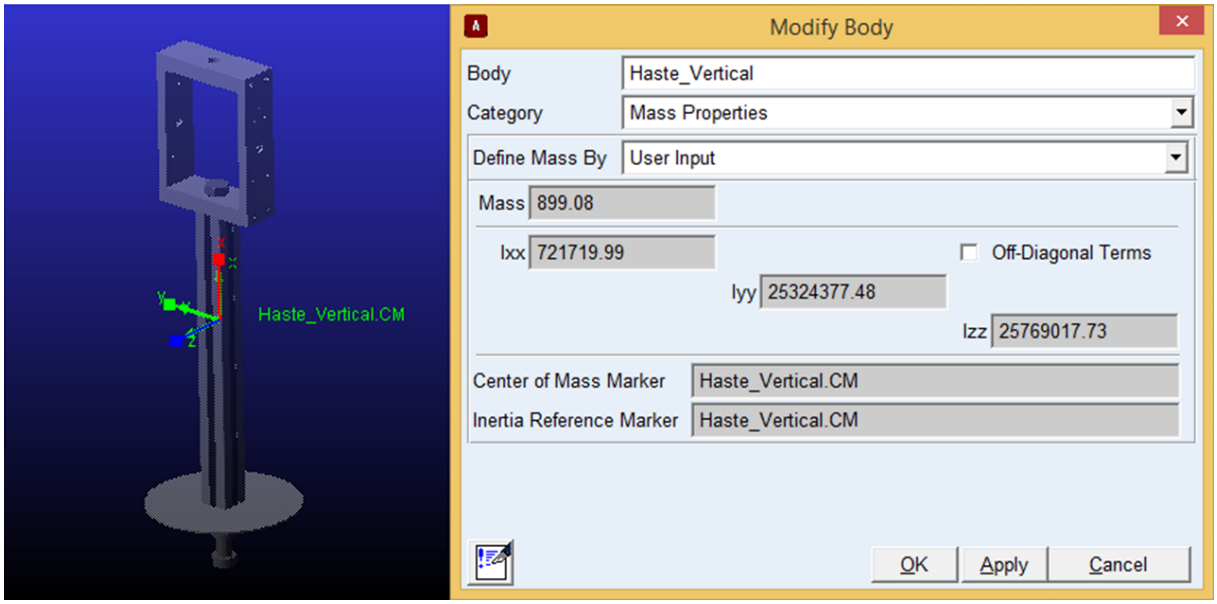
Figura 21 – Configuração das características inerciais da base.



Fonte: Autoria própria.

O segundo corpo rígido importado no protótipo virtual do Adams foi a haste vertical, apresentada na Figura 22. A haste vertical se relaciona com a base por meio de uma junta de revolução, isto permite que o Helicóptero tenha o GDL de deslocamento.

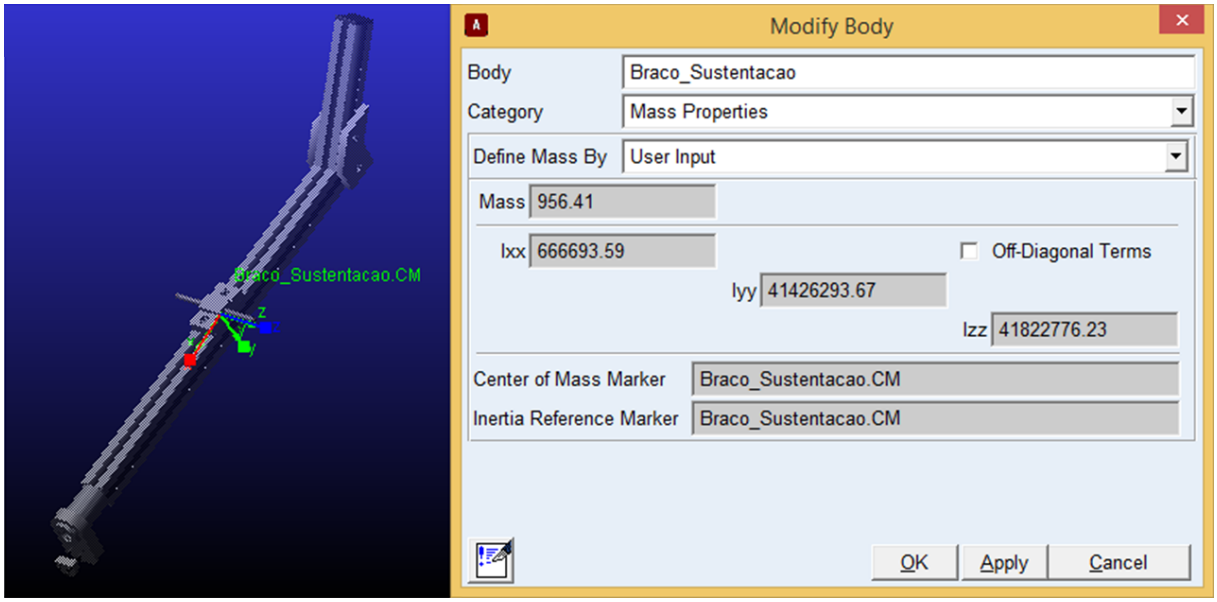
Figura 22 – Configuração das características inerciais da haste vertical.



Fonte: Autoria própria.

O terceiro corpo rígido importado no protótipo virtual do Adams foi o braço de sustentação, apresentado na Figura 23. O braço de sustentação se relaciona com a haste vertical por meio de uma junta de revolução, isto permite que o Helicóptero tenha o GDL de elevação.

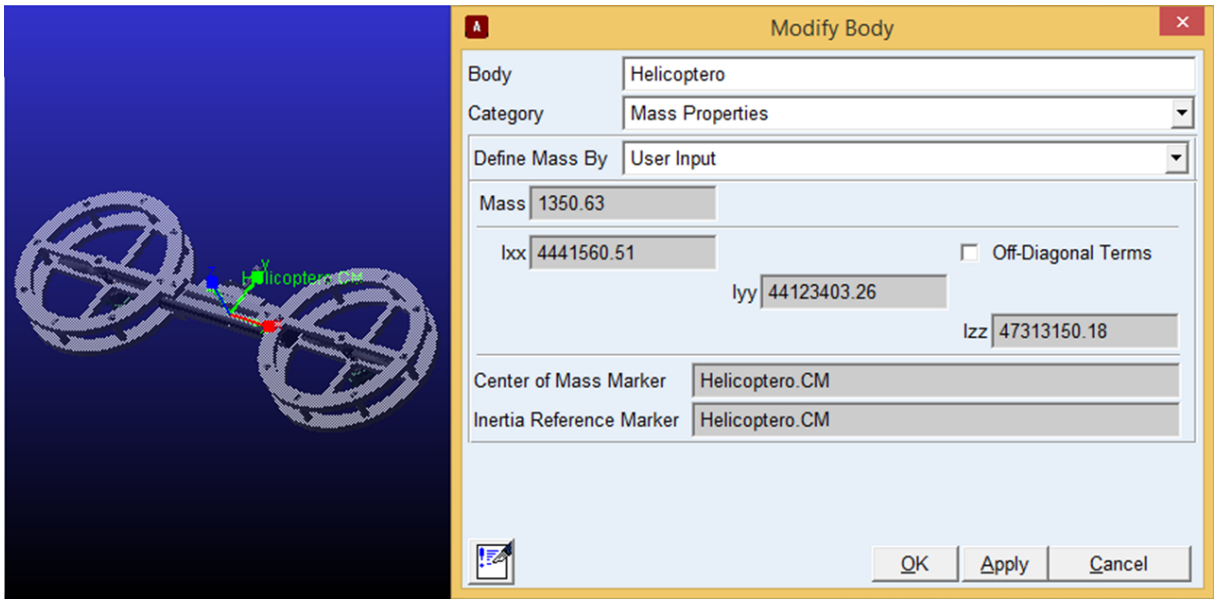
Figura 23 – Configuração das características inerciais do braço de sustentação.



Fonte: Autoria própria.

O quarto corpo rígido importado no protótipo virtual do Adams foi o helicóptero, apresentado na Figura 24. O helicóptero se relaciona com o braço de sustentação por meio de uma junta de revolução, isto permite que o Helicóptero tenha o GDL de arfagem.

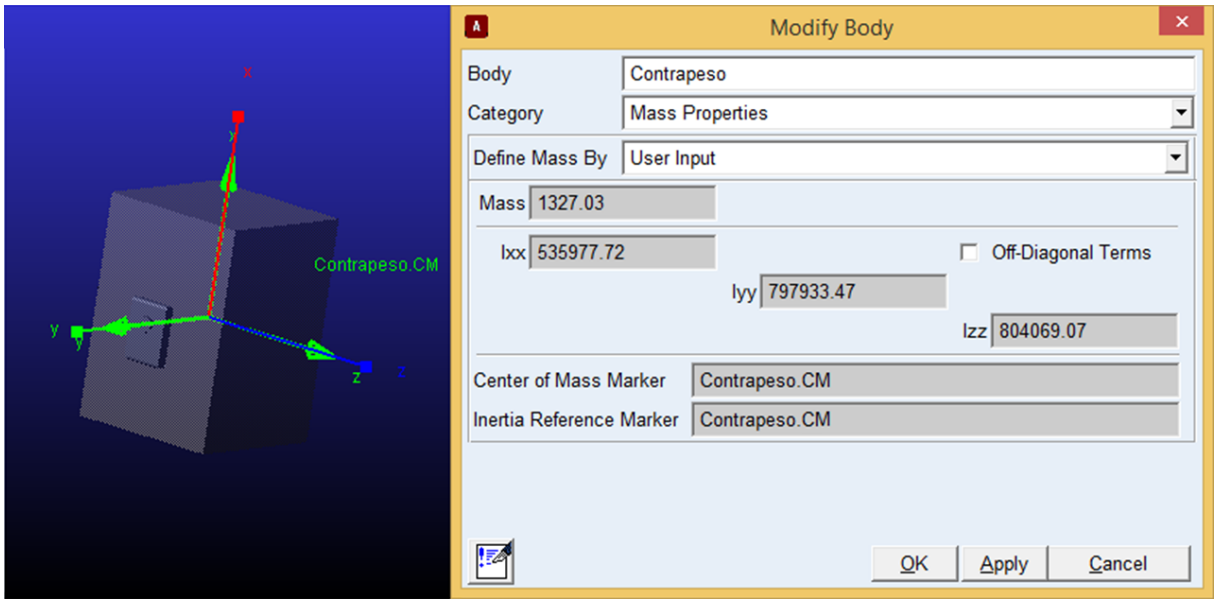
Figura 24 – Configuração das características inerciais do helicóptero.



Fonte: Autoria própria.

O quinto corpo rígido importado no protótipo virtual do Adams foi o contrapeso, apresentado na Figura 25. Por meio de uma junta fixa, o contrapeso foi preso ao braço de sustentação.

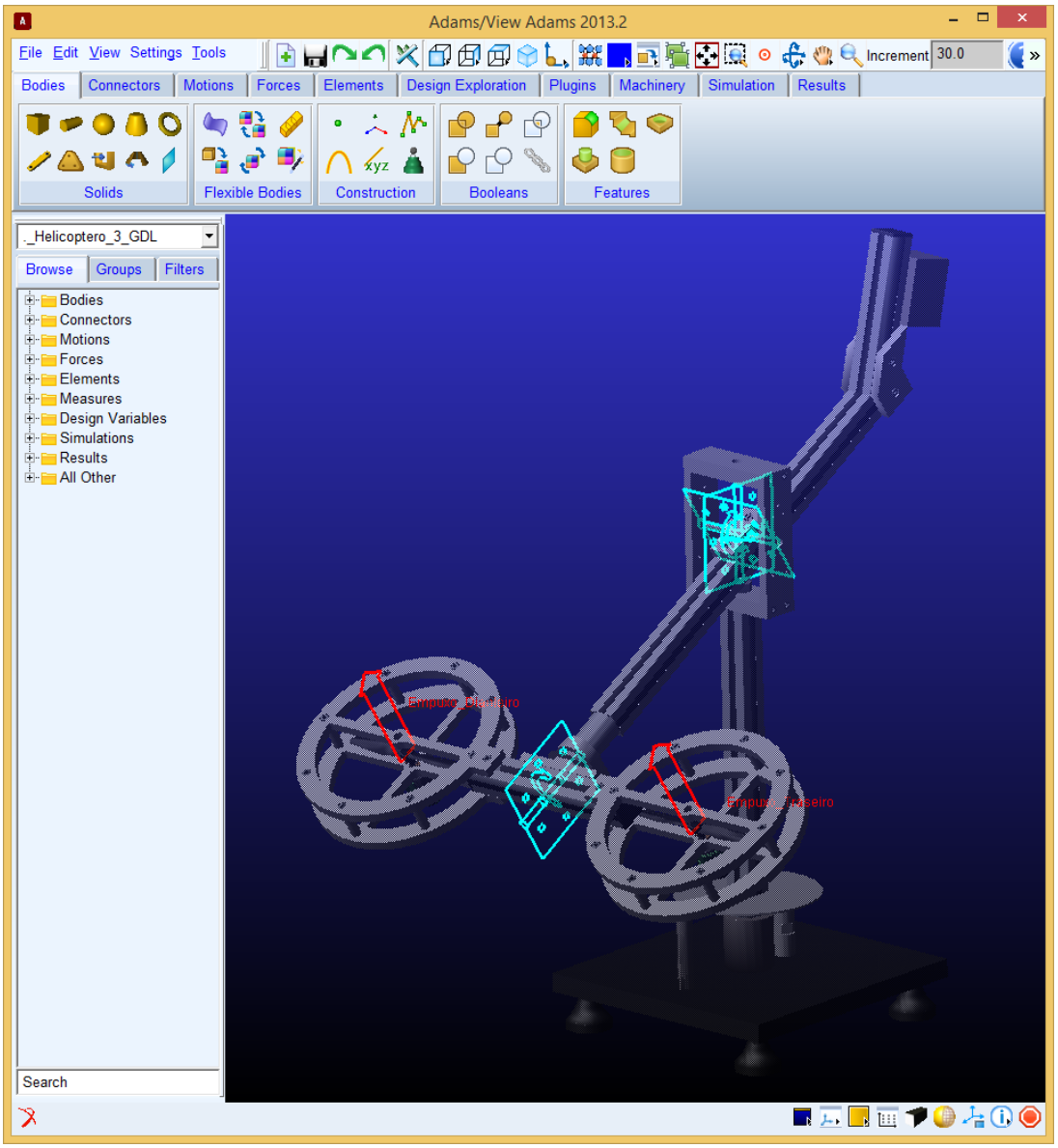
Figura 25 – Configuração das características inerciais do contrapeso.



Fonte: Autoria própria.

Após a importação de todos os corpos rígidos, configuração de suas características inercias e definição das juntas relacionando cada um dos corpos, foram definidas as forças de empuxo atuando sobre o corpo rígido do helicóptero. A Figura 26 apresenta o posicionamento das juntas de revolução de deslocamento, elevação e arfagem, e as forças de empuxo dianteiro e traseiro, respeitando as convenções apresentadas na Seção 4.1.

Figura 26 – Configuração das juntas de revolução e das forças de empuxo.



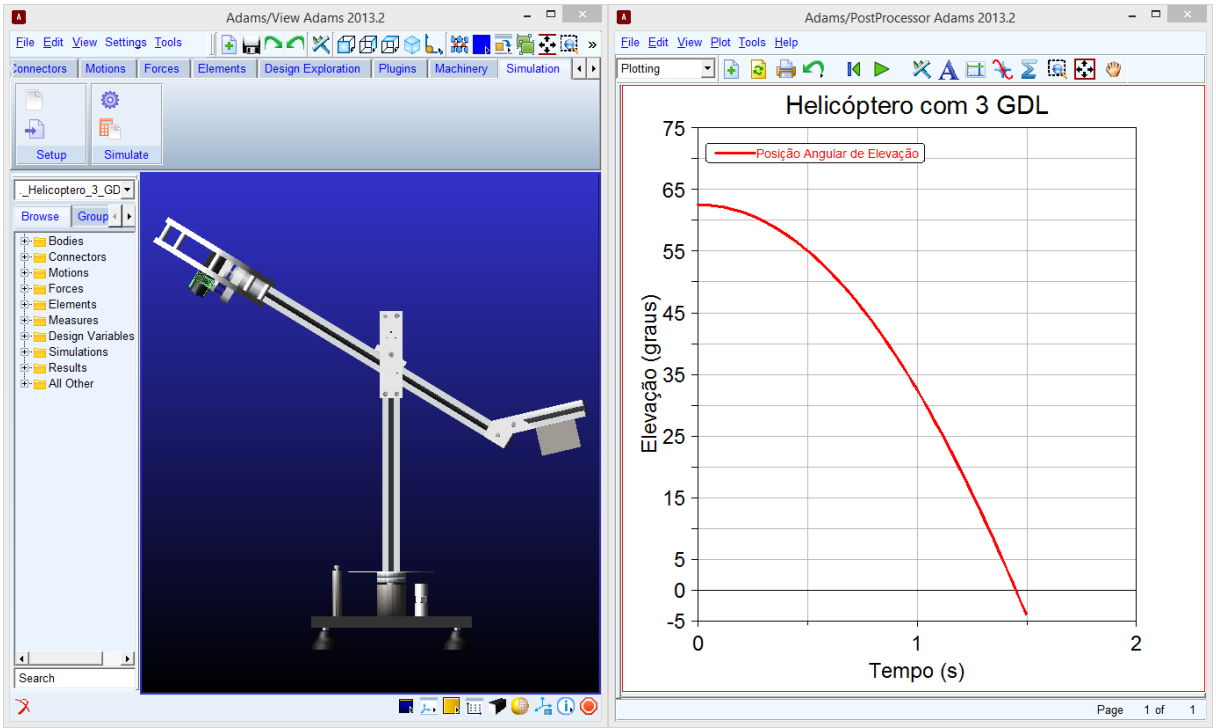
Fonte: Autoria própria.

Para concluir a construção do protótipo virtual foram criadas as variáveis de saída e controle do Helicóptero com 3 GDL. As variáveis de saída são as posições angulares do movimento de deslocamento (y_1), elevação (y_2) e arfagem (y_3), seguidas das velocidades angulares de deslocamento (y_4), elevação (y_5) e arfagem (y_6). As variáveis de controle são as forças de empuxo dianteiro (u_1) e traseiro (u_2).

Após concluir a construção do protótipo virtual do Helicóptero com 3 GDL foi necessário avaliá-lo por meio das análises estática e dinâmica, a fim de determinar se o sistema tem um comportamento adequado. Sabendo que o sistema produz forças de empuxo apenas para elevação do helicóptero, e que quanto mais distante o contrapeso estiver do helicóptero menor será a força necessária na atuação do sistema. O protótipo virtual foi construído e avaliado para diferentes posições do contrapeso, a fim de determinar uma configuração controlável e com menor exigência dos atuadores. A posição determinada para o contrapeso foi a uma distância de 20 mm da extremidade do braço de sustentação.

A análise dinâmica permite visualizar o comportamento do Helicóptero com 3 GDL no tempo. A posição do contrapeso foi escolhida de modo que partindo da posição angular de elevação máxima, aproximadamente 1,0919 radianos (62,56°), o helicóptero fosse capaz de voltar para a posição de elevação zero, apenas sob a ação da gravidade, sem a necessidade de forças de empuxo. À esquerda da Figura 27 o Helicóptero com 3 GDL está na condição inicial da análise dinâmica. À direita da Figura 27 está a resposta no tempo da posição angular de elevação.

Figura 27 – Análise dinâmica do protótipo virtual do Helicóptero com 3 GDL.

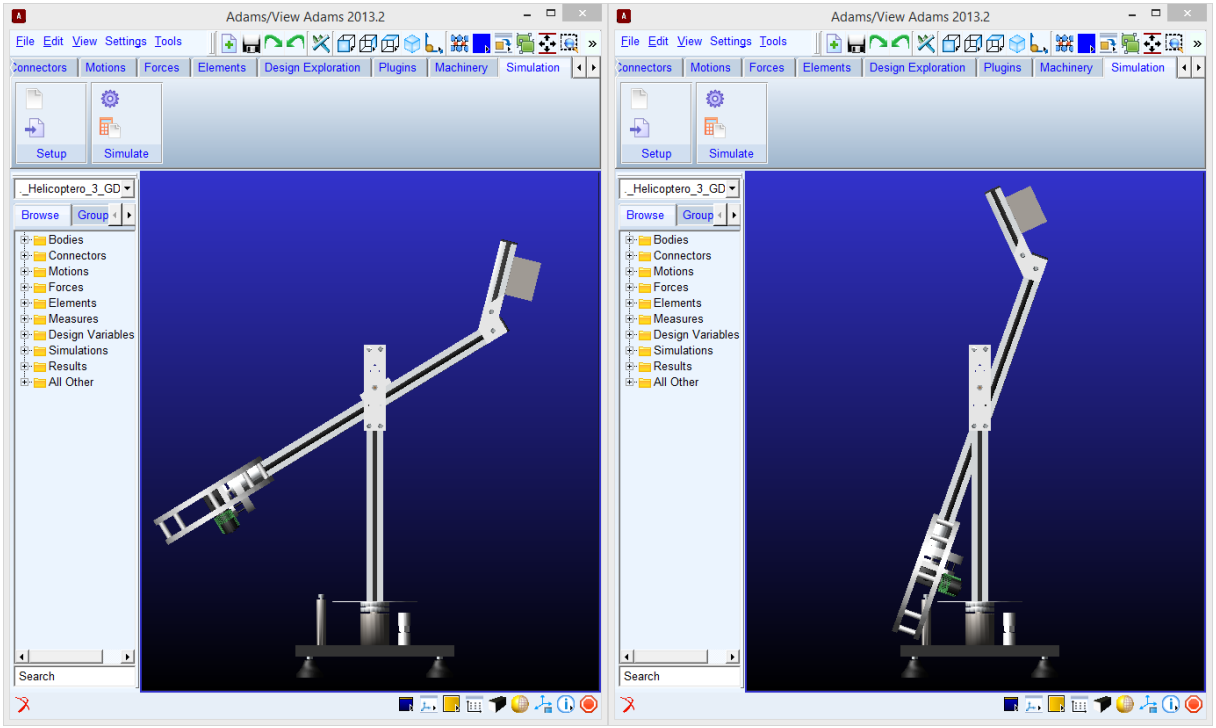


Fonte: Autoria própria.

A análise estática permite encontrar um ponto de equilíbrio do sistema, no qual as forças de empuxo dianteiro e traseiro são zero. A posição do contrapeso foi escolhida de modo que o ponto de equilíbrio do sistema se encontrasse abaixo da elevação em sua posição angular zero. Deste modo, para controlar o helicóptero em qualquer posição angular de elevação igual ou maior que zero é necessário aplicar forças de empuxo positivas. À esquerda da Figura 28 o

Helicóptero com 3 GDL está em sua posição inicial, na qual todos as posições angulares são zero. À direita da Figura 28 o sistema está em seu ponto de equilíbrio, no qual as posições angulares de deslocamento e arfagem são zero, e a posição angular de elevação é aproximadamente -0,6828 radianos (-39,1191°). É importante destacar, que nenhuma força de contato foi definida entre os corpos do protótipo virtual construído.

Figura 28 – Análise estática do protótipo virtual do Helicóptero com 3 GDL.



Fonte: Autoria própria.

A análise linear permite determinar o modelo matemático linear do Helicóptero com 3 GDL na notação de espaço de estado apresentada na Seção 3.2, as equações de estado e saída são representadas, respectivamente, nas Equações (58) e (59).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{58}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \tag{59}$$

O vetor de estado $\boldsymbol{x}$ é selecionado internamente pelo Adams, contendo as coordenadas generalizadas com as mais rápidas variações dentro do modelo (MONTEZUMA, 2003; BREGANON, 2009). As variáveis escolhidas para compor o vetor de saída $\boldsymbol{y}$ são, em ordem, as posições angulares de deslocamento (y_1), elevação (y_2) e arfagem (y_3), seguidas das velocidades angulares de deslocamento (y_4), elevação (y_5) e arfagem (y_6). As variáveis escolhidas para compor o vetor de controle $\boldsymbol{u}$ são, em ordem, as forças de empuxo dianteiro (u_1) e traseiro (u_2).

A linearização do Helicóptero com 3 GDL em torno de um ponto de operação diferente do ponto de equilíbrio estático, apresentado à direita da Figura 28, depende da aplicação de forças de empuxo dianteiro e traseiro, que compensem constantemente a força gravitacional sobre o helicóptero. Deste modo, é necessário determinar as forças de empuxo de operação para linearizar o sistema em torno da condição inicial, na qual todas as posições angulares são zero.

A identificação das forças de operação no ponto de condição inicial foi realizada por meio de um controle do modelo não linear do Adams, em malha fechada. Este método é possível devido a execução de uma cossimulação entre MATLAB/Simulink e Adams, que permite controlar o modelo não linear do Helicóptero com 3 GDL sob ação da gravidade. O controle parte do ponto de equilíbrio estático e estabiliza o helicóptero na condição em que as posições angulares são zero. Deste modo, a força de operação no ponto de condição inicial equivale as forças de empuxo dianteiro e traseiro, que estabilizam o controle do helicóptero.

O controle aplicado foi o seguidor com realimentação de estado, apresentado na Seção 3.3, projetado pelo método de atribuição de autoestrutura completa, apresentado na Seção 3.4. Para tanto, utilizou-se o modelo matemático do Helicóptero com 3 GDL linearizado em torno do ponto de equilíbrio estático, representado pelas matrizes de estado $\boldsymbol{A}$, de entrada $\boldsymbol{B}$, de saída $\boldsymbol{C}$ e de transmissão direta $\boldsymbol{D}$, respectivamente, nas Equações (60), (61), (62) e (63).

$$\boldsymbol{A} = \begin{bmatrix} 0 & -0{,}5773 & 0 & -3{,}797e-16 & 0 & -0{,}915 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1{,}08e-16 & 0 & -1{,}062 & 0 & 2{,}989e-16 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -0{,}4646 & 0 & 1{,}078e-15 & 0 & -0{,}7364 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{60}$$

$$\boldsymbol{B} = \begin{bmatrix} 0{,}1077 & -0{,}1077 \\ 0 & 0 \\ -0{,}9958 & -0{,}9958 \\ 0 & 0 \\ -3{,}775 & 3{,}775 \\ 0 & 0 \end{bmatrix} \tag{61}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -2{,}12e-16 & 0 & -1 & 0 & 0 \\ 0 & -0{,}7382 & 0 & -5{,}404e-16 & 0 & -1{,}17 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -2{,}12e-16 & 0 & -1 & 0 & 0 & 0 \\ -0{,}7382 & 0 & -5{,}404e-16 & 0 & -1{,}17 & 0 \end{bmatrix} \tag{62}$$

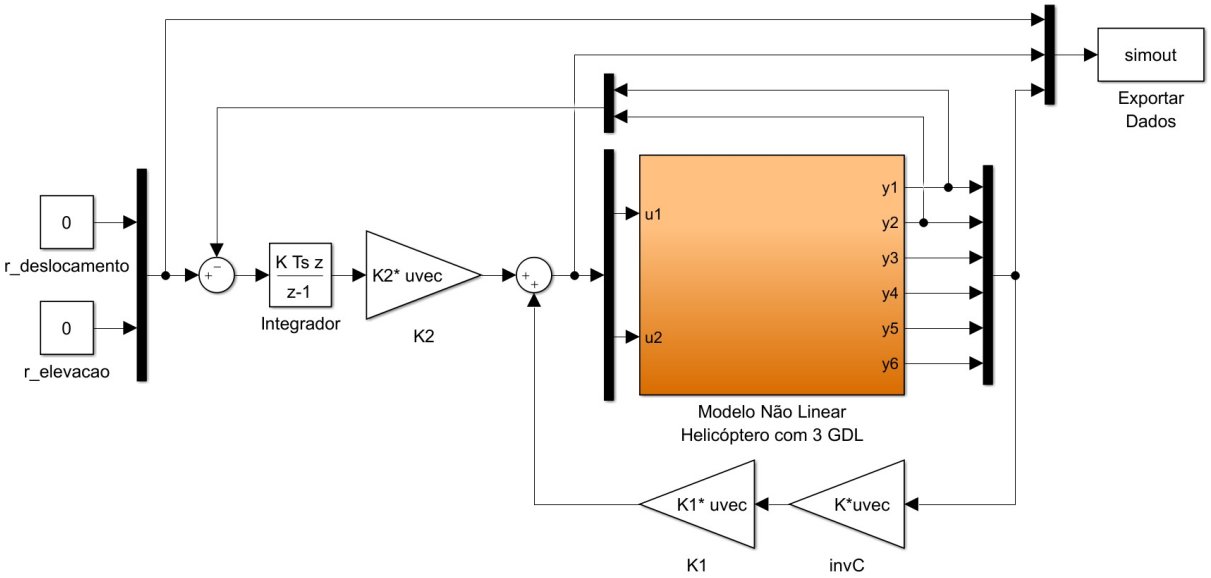
$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{63}$$

Para as matrizes de estado $\boldsymbol{A}$, na Equação (60), e entrada $\boldsymbol{B}$, na Equação (61), a condição de controlabilidade na Equação (64) é satisfeita, sendo n o número de variáveis de estado.

$$posto\begin{bmatrix}\boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A^2B} & \cdots & \boldsymbol{A^{n-1}B}\end{bmatrix} = n \tag{64}$$

O diagrama de blocos, construído no Simulink, do controle seguidor com realimentação de estado, para identificação das forças de operação, está representado na Figura 29. O bloco em cor laranja é gerado pelo Adams para realizar a cossimulação com o Simulink. As únicas saídas controladas são as posições angulares de deslocamento (y_1) e elevação (y_2). A referência de controle destas é zero, correspondendo ao ponto de operação em torno do qual se deseja linearizar o sistema.

Figura 29 – Controle em malha fechada para identificação das forças de operação.



Fonte: Autoria própria.

As matrizes de ganhos $\boldsymbol{K_1}$ e $\boldsymbol{K_2}$ representadas, respectivamente nas Equações (65) e (66), foram obtidas pela atribuição do espectro de autovalores de malha fechada na Equação (67) e da matriz de autovetores associados, subdividida nas Equações (68) e (69).

$$\boldsymbol{K_1} = \begin{bmatrix} -0{,}6093 & 0{,}1122 & 1{,}3306 & 0{,}6467 & 0{,}5654 & 0{,}8562 \\ 0{,}6093 & -0{,}1122 & 1{,}3306 & 0{,}6467 & -0{,}5654 & -0{,}8562 \end{bmatrix} \tag{65}$$

$$\boldsymbol{K_2} = \begin{bmatrix} 0{,}0758 & 0{,}3504 \\ -0{,}0758 & 0{,}3504 \end{bmatrix} \tag{66}$$

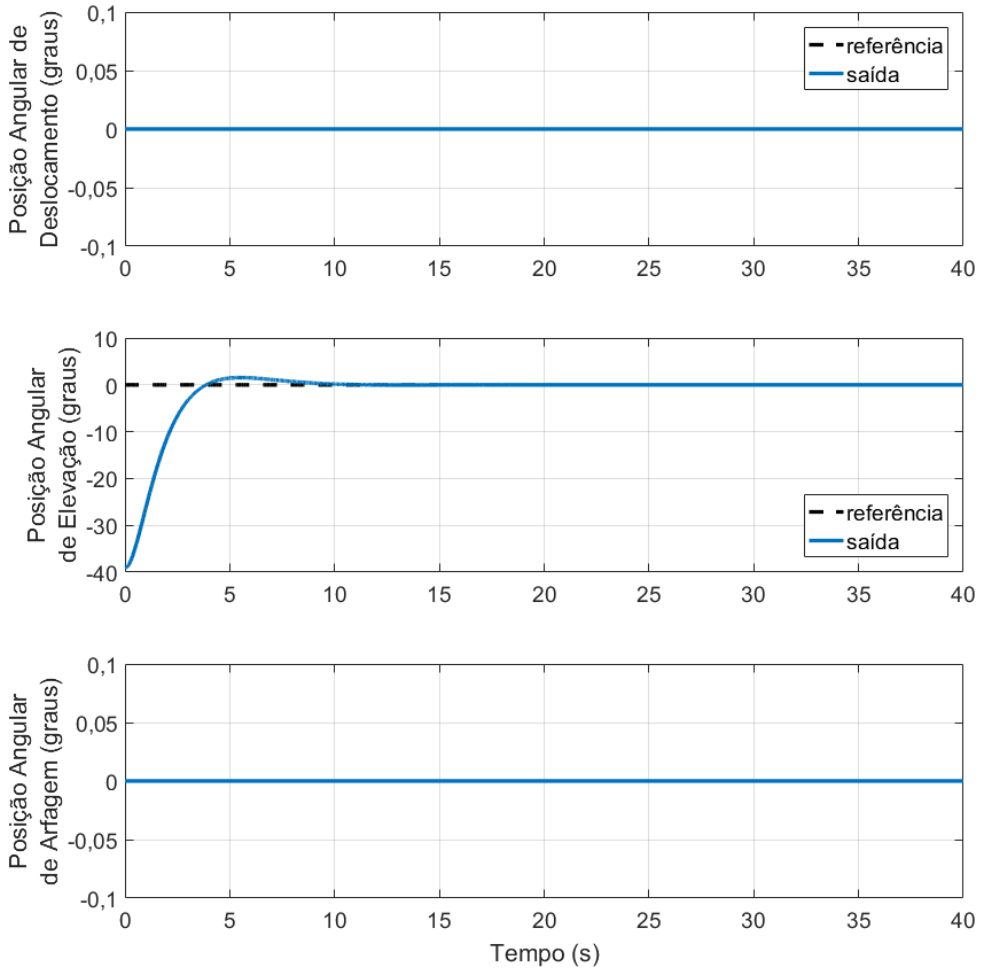
$$\boldsymbol{\sigma(A + BK)} = [-0{,}90 + 0{,}12i \quad -0{,}90 - 0{,}12i \quad -0{,}85 + 0{,}11i \quad -0{,}85 - 0{,}11i \quad \cdots \\ -0{,}80 + 0{,}10i \quad -0{,}80 - 0{,}10i \quad -1{,}00 \quad -0{,}95] \tag{67}$$

$$V = \begin{bmatrix} 3{,}6625 & 0{,}8505 & -0{,}0000 & -0{,}0000 & 4{,}4880 & 0{,}9177 & 3{,}1475 & -0{,}0000 \\ -3{,}8745 & -1{,}4616 & 0{,}0000 & 0{,}0000 & -5{,}3825 & -1{,}8199 & -3{,}1475 & 0{,}0000 \\ -0{,}0000 & -0{,}0000 & -0{,}9577 & 0{,}0226 & -0{,}0000 & -0{,}0000 & -0{,}0000 & -0{,}9632 \\ 0{,}0000 & 0{,}0000 & 1{,}1115 & 0{,}1173 & 0{,}0000 & 0{,}0000 & 0{,}0000 & 1{,}0139 \\ -5{,}4843 & -0{,}4398 & -0{,}0000 & 0{,}0000 & -5{,}8939 & -0{,}4496 & -5{,}1904 & 0 \\ 5{,}9232 & 1{,}2784 & 0{,}0000 & -0{,}0000 & 7{,}1849 & 1{,}4602 & 5{,}1904 & 0 \\ -4{,}0171 & -2{,}1596 & 0{,}0000 & 0{,}0000 & -6{,}3446 & -3{,}0680 & -3{,}1475 & 0{,}0000 \\ -0{,}0000 & -0{,}0000 & -1{,}2685 & -0{,}3021 & -0{,}0000 & -0{,}0000 & -0{,}0000 & -1{,}0673 \end{bmatrix} \tag{68}$$

$$Q = \begin{bmatrix} -1 & 0 & -1 & 0 & -1 & 0 & -1 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 & -1 \end{bmatrix} \tag{69}$$

A Figura 30 apresenta a resposta do controlador, apresentado na Figura 29, para as posições angulares de deslocamento, elevação e arfagem. Em regime o helicóptero permanece estabilizado na posição inicial.

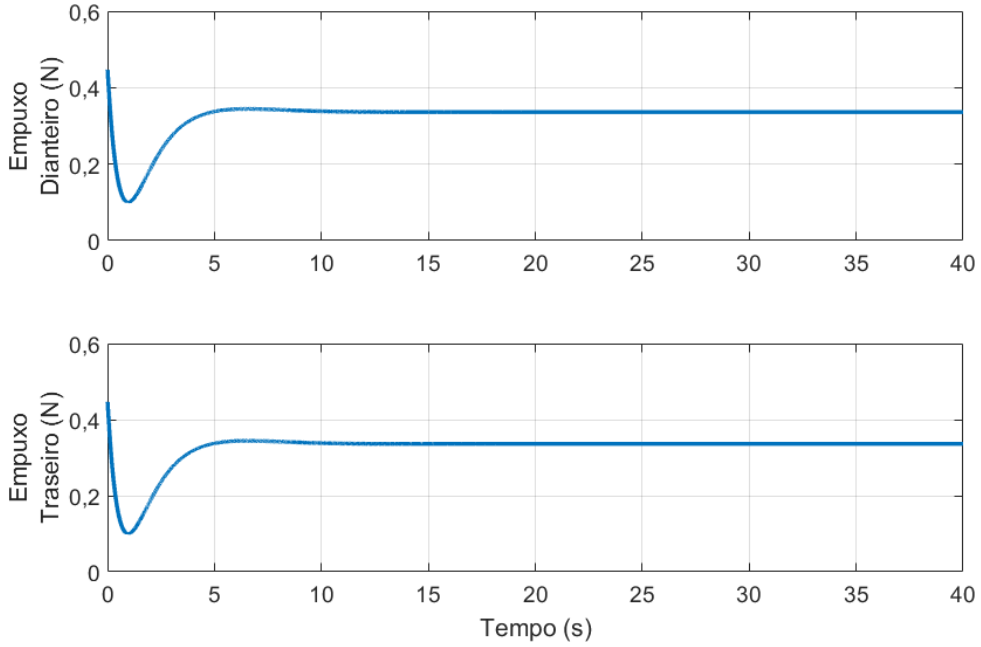
Figura 30 – Resposta de controle das posições angulares na identificação das forças de operação.



Fonte: Autoria própria.

A Figura 31 apresenta as forças de empuxo dianteiro e traseiro produzidas pelo controlador, apresentado na Figura 29. Os valores das forças de empuxo dianteiro e traseiro no regime formam o vetor das forças de empuxo de operação $\boldsymbol{u_{op}}$, apresentado na Equação (70).

Figura 31 – Resposta das forças de empuxo dianteiro e traseiro na identificação das forças de operação.



Fonte: Autoria própria.

$$\boldsymbol{u_{op}} = \begin{bmatrix} u_{op1} \\ u_{op2} \end{bmatrix} \cong \begin{bmatrix} 0{,}3363 \\ 0{,}3363 \end{bmatrix} \tag{70}$$

As forças de operação dianteira (u_{op1}) e traseira (u_{op2}), na Equação (70), foram somadas, respectivamente, às forças de empuxo dianteiro (u_1) e traseiro (u_2) no protótipo virtual do Adams. Isto permitiu obter o modelo matemático do Helicóptero com 3 GDL, linearizado em torno do ponto de condição inicial, representado pelas matrizes de estado $\boldsymbol{A}$, de entrada $\boldsymbol{B}$, de saída $\boldsymbol{C}$ e de transmissão direta $\boldsymbol{D}$, respectivamente, nas Equações (71), (72), (73) e (74).

$$\boldsymbol{A} = \begin{bmatrix} 0 & -2{,}935e-17 & 0 & -0{,}995 & 0 & 1{,}714e-14 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2{,}297e-16 & 0 & -1{,}902 & 0 & -1{,}455e-15 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1{,}497e-17 & 0 & -8{,}114e-15 & 0 & -0{,}8237 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{71}$$

$$\boldsymbol{B} = \begin{bmatrix} 0{,}03819 & -0{,}03819 \\ 0 & 0 \\ -3{,}777 & 3{,}777 \\ 0 & 0 \\ -0{,}9958 & -0{,}9958 \\ 0 & 0 \end{bmatrix} \tag{72}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1{,}225e-16 & 0 & 0 & 0 & -1 \\ 0 & -1{,}11e-16 & 0 & -1{,}17 & 0 & 7{,}225e-15 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1{,}225e-16 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1{,}17 & 0 & 7{,}225e-15 & 0 \end{bmatrix} \tag{73}$$

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{74}$$

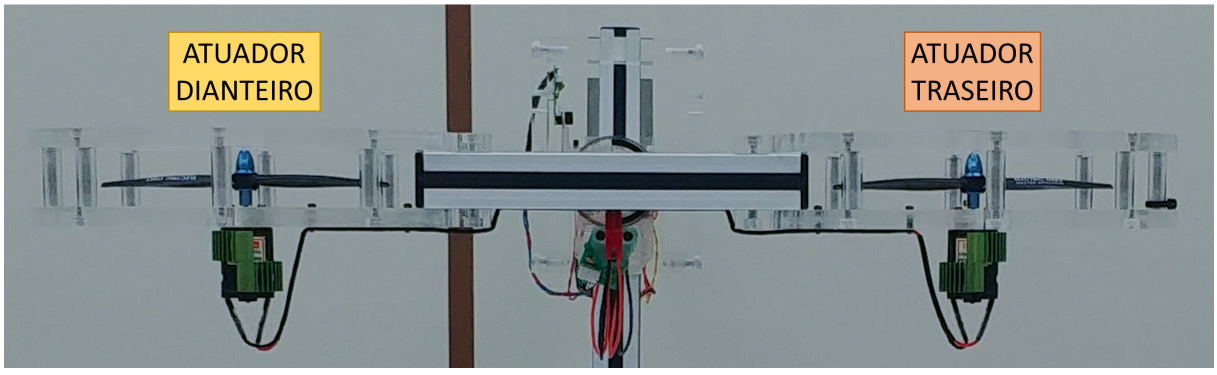
Para as matrizes de estado $\boldsymbol{A}$, na Equação (71), e entrada $\boldsymbol{B}$, na Equação (72), a condição de controlabilidade na Equação (75) é satisfeita.

$$posto \begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A^2B} & \cdots & \boldsymbol{A^{n-1}B} \end{bmatrix} = n \tag{75}$$

4.7 ATUADORES

Como apresentado na Seção 4.1 o Helicóptero com 3 GDL possui como atuadores dois motores de corrente contínua equipados com hélices de passo fixo, apresentados na Figura 32. Este novo par de hélices, agora, gira em sentidos opostos e produz forças de empuxo apenas para elevar o helicóptero.

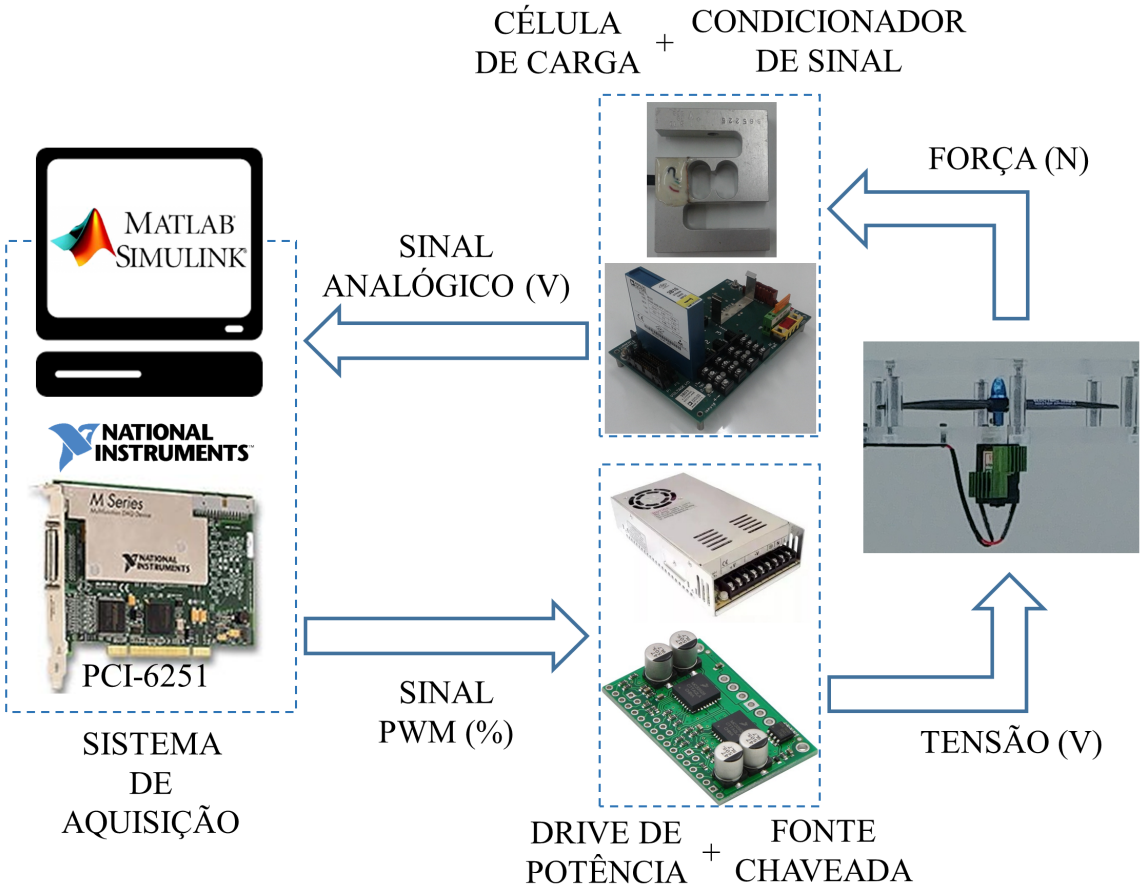
Figura 32 – Atuadores dianteiro e traseiro do Helicóptero com 3 GDL.



Fonte: Autoria própria.

Os motores de corrente contínua são acionados por um *drive* de potência Dual MC33926. Ao variar a razão cíclica do sinal de PWM aplicado ao *drive* de potência entre 0 e 100%, a tensão aplicada aos motores varia entre 0 e 24 V. Como o modelo matemático do Helicóptero com 3 GDL tem como variáveis de controle as forças de empuxo produzidas pelos atuadores dianteiro e traseiro, foi necessário determinar um polinômio matemático para o cálculo da razão cíclica do PWM a partir de uma determinada força de empuxo. O processo de identificação desta relação matemática dos atuadores é representado na Figura 33.

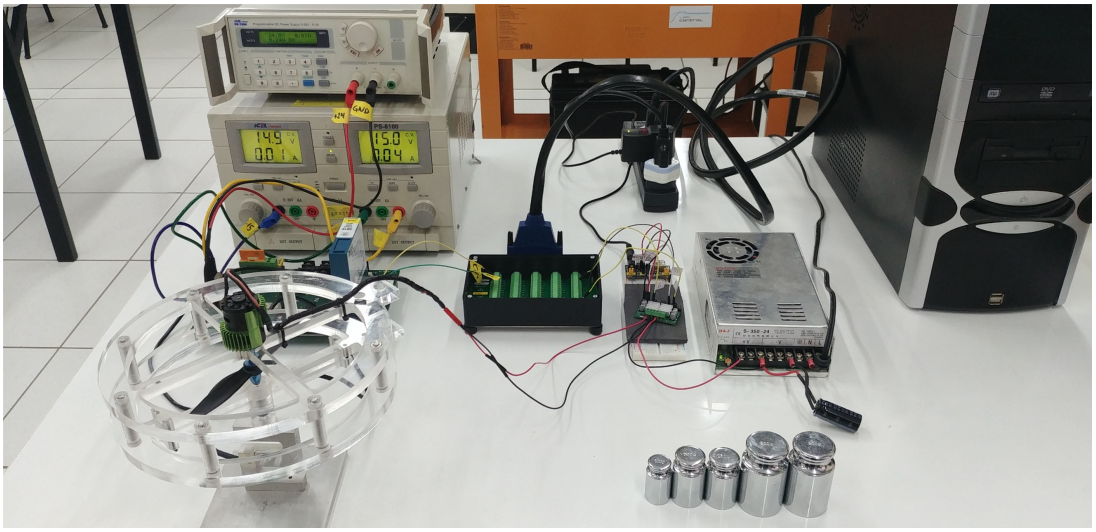
Figura 33 – Processo de identificação do polinômio matemático dos atuadores.



Fonte: Autoria própria.

O sistema de aquisição, por meio da placa PCI-6251, aplica ao *drive* de potência Dual MC33926 um sinal de PWM. O *drive* de potência, alimentado por uma fonte chaveada de 24 V, aplica a tensão respectiva ao sinal de PWM no atuador. Para a dada tensão, o atuador produz uma força de empuxo, que causa uma deformação na célula de carga. Este sensor, do modelo S10 da Alfa Instrumentos, em conjunto com um condicionador de sinal 3B16 e uma placa modular 3B03, ambos da Analog Device, produzem um sinal analógico correspondente a deformação sofrida pela célula de carga. Por fim, este sinal analógico é lido pelo sistema de aquisição por meio da PCI-6251. Este processo foi aplicado aos atuadores individualmente, como apresentado na Figura 34.

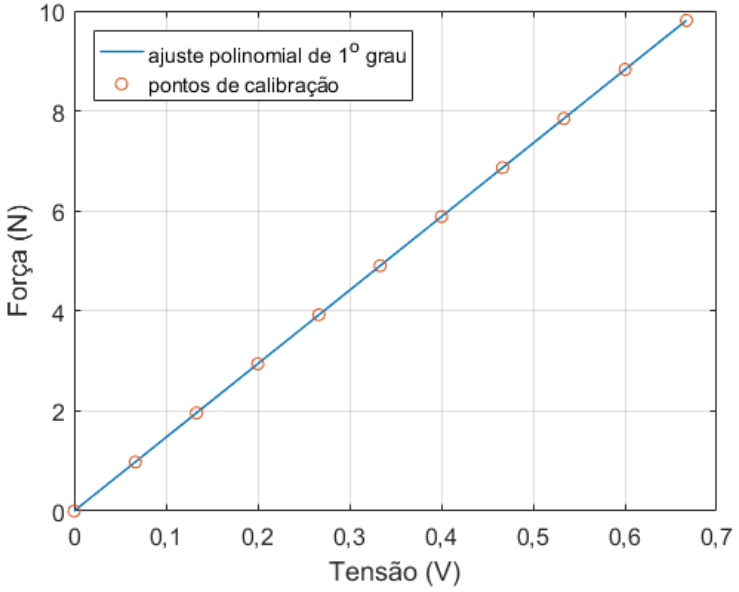
Figura 34 – Identificação do polinômio matemático de um dos atuadores.



Fonte: Autoria própria.

Por meio de uma prévia calibração da célula de carga para massas de 0 até 1000 g, com incrementos de 100 g, foi obtida uma relação matemática que permite interpretar o sinal analógico da célula de carga diretamente como força (N). O gráfico da Figura 35 mostra os pontos de calibração e o ajuste polinomial de 1° grau obtido na calibração da célula de carga. Considerando o sinal da célula de carga como x e a força como y, o polinômio de calibração da célula de carga é dado na Equação (76).

Figura 35 – Ajuste polinomial de calibração da célula de carga.



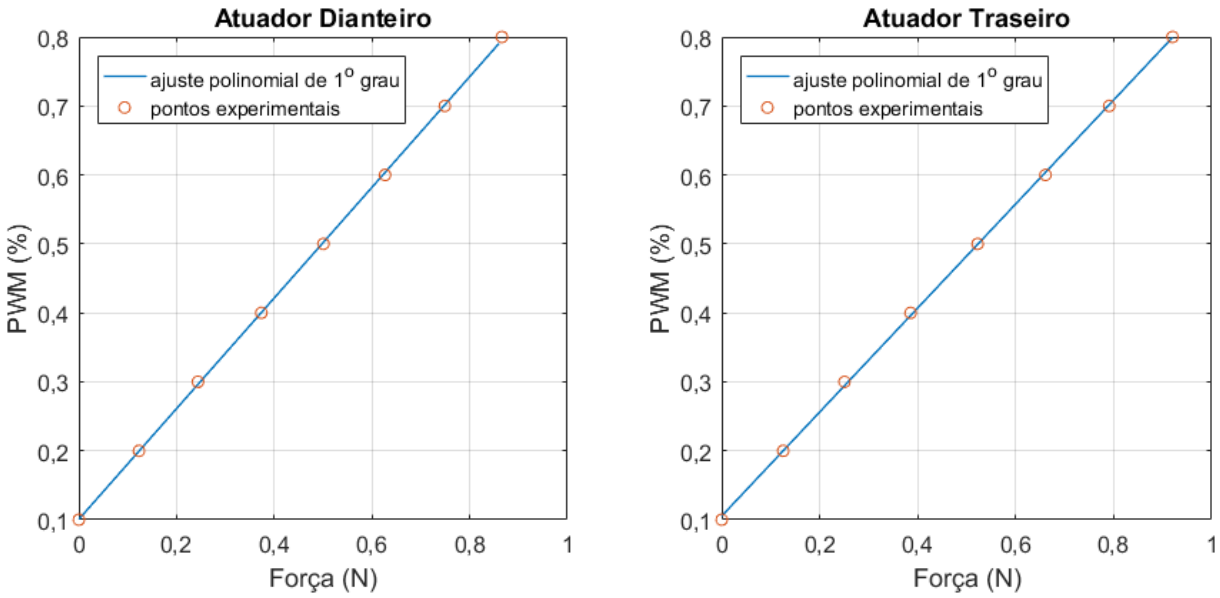
Fonte: Autoria própria.

$$y = 14{,}7076x + 0{,}0007 \tag{76}$$

Como os atuadores começam a responder com uma razão cíclica de 10% e se mantêm com uma relação linear até uma razão cíclica de 80%, este foi o intervalo de identificação da relação matemática dos atuadores. A razão cíclica variou em incrementos de 10% até atingir o limite máximo, em seguida sofreu decrementos de 10% até atingir o limite mínimo. Para cada variação da razão cíclica do sinal de PWM, foi aquisitado o sinal de força da célula de carga por cinco segundos. O valor de força associado com cada valor de razão cíclica corresponde a média do sinal neste período de tempo, descartando-se os dois primeiros segundos de transição.

A Figura 36 apresenta o ajuste polinomial de 1° grau dos atuadores dianteiro e traseiro. Os pontos experimentais associados a cada valor de razão cíclica correspondem a média entre os valores aferidos para a força de empuxo no incremento e no decremento. Considerando o valor de força como x e a razão cíclica do sinal de PWM como y, os polinômios identificados para os atuadores dianteiro e traseiro são apresentados, respectivamente, nas Equações (77) e (78).

Figura 36 – Ajuste polinomial dos atuadores dianteiro e traseiro.



Fonte: Autoria própria.

$$y = 0{,}8023x + 0{,}1007 \tag{77}$$

$$y = 0{,}7542x + 0{,}1054 \tag{78}$$

4.8 SENSORES

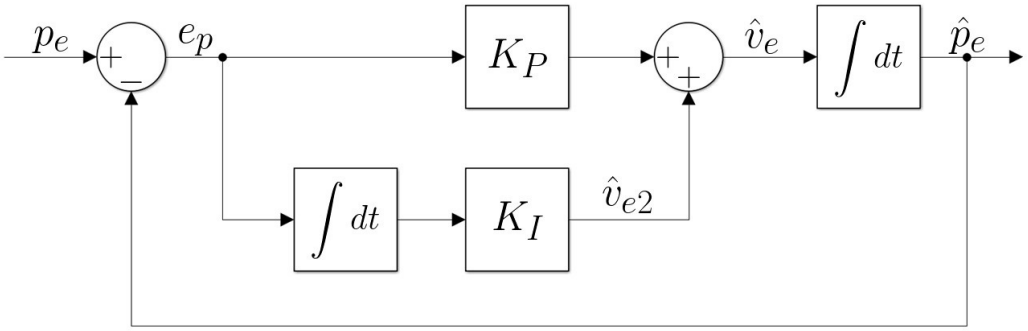
A determinação das posições angulares do Helicóptero com 3 GDL é feita por meio de sensores digitais do tipo *encoder* incremental. O modelo utilizado é o HEDS-9721, da fabricante Hewlett-Packard. Para os movimentos de elevação e arfagem a resolução do *encoder*

é 1200 pulsos por revolução, enquanto para o movimento de deslocamento é de 1800 pulsos por revolução. Sendo assim, a acuracidade da posição medida pelo *encoder* depende de sua quantização. Em quadratura 4x as posições angulares de elevação e arfagem apresentam uma resolução de 0,075°, e a posição angular de deslocamento uma resolução de 0,05° (MERRY; MOLENGRAFT; STEINBUCH, 2010; SHIMADA, 2015).

No sistema do Helicóptero com 3 GDL, utilizado neste trabalho, não existe nenhum sensor específico para a medição das velocidades angulares dos movimentos. Desta forma, é necessário determinar as velocidades angulares a partir das posições angulares obtidas pelos *encoders*. Como os sinais de posição angular para estes sensores são discretos e quantizados a derivação direta do sinal não representa uma boa opção, uma vez que produziria um ruído de quantização (TILLI; MONTANARI, 2001; MERRY; MOLENGRAFT; STEINBUCH, 2010).

Neste contexto, um método que apresenta melhores resultados na estimação da velocidade angular a partir da posição angular do *encoder*, para movimentos de baixa velocidade, é o *Tracking Loop* (TL). Considerando a posição angular real obtida do encoder p_e, uma posição angular estimada dada por $\hat{p}_e$, o erro entre posição angular real e estimada e_p, a primeira estimativa de velocidade $\hat{v}_e$ e a segunda estimativa de velocidade $\hat{v}_{e2}$. A estimação por TL é dada por um controlador PI seguido de um integrador, como apresentado no diagrama de blocos da Figura 37 (SACHS, 2013; LEE; SONG, 2001).

Figura 37 – Diagrama de blocos do *Tracking Loop*.



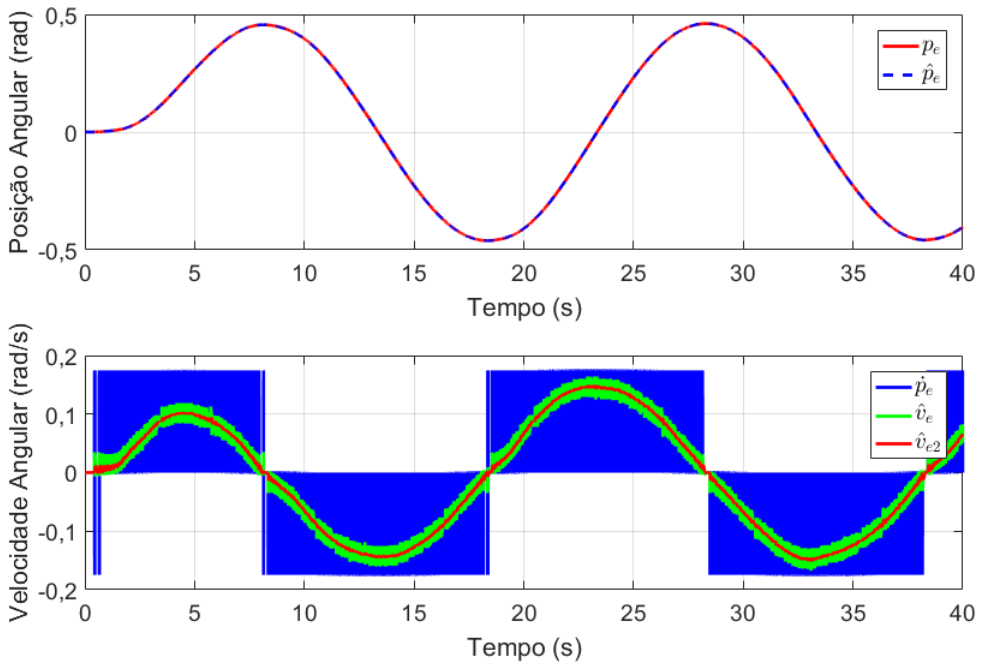
Fonte: Autoria própria.

Como visto na Figura 37, o erro entre posição angular real e estimada $e_p = p_e - \hat{p}_e$ é a entrada do controlador PI, que por sua vez tem como saída a estimativa de velocidade $\hat{v}_e$. A integração no tempo de $\hat{v}_e$ resulta na estimativa de posição $\hat{p}_e$. O ganho proporcional do TL é dado por K_P, enquanto o ganho integral é dado por K_I. Algo interessante da estimação por TL é que o método resulta em duas estimativas de velocidade, dadas por $\hat{v}_e$ e $\hat{v}_{e2}$, sendo esta menos ruidosa por estar na saída do integrador (SACHS, 2013).

A Figura 38 apresenta o resultado da estimação da velocidade angular por TL, semelhante ao apresentado em Sachs (2013), com ganho proporcional $K_P = 40$ e integral $K_I = 900$, aplicada sobre o sinal de posição angular obtido do *encoder* do movimento de deslocamento. O gráfico superior mostra que a estimativa de posição $\hat{p}_e$ está bem adequada

a posição angular real p_e. O gráfico inferior mostra as duas estimativas de velocidade do TL, dadas por $\hat{v}_e$ e $\hat{v}_{e2}$, comprovando que a segunda estimativa é menos ruidosa. Além disso, também está representada a velocidade angular $\dot{p}_e$, obtida diretamente pela derivação do sinal de posição angular real p_e. Comprovando que o TL apresenta melhores resultados para estimação de velocidade angular a partir de um sinal quantizado.

Figura 38 – Resultados do *Tracking Loop* sobre o sinal de posição angular de um *encoder*.



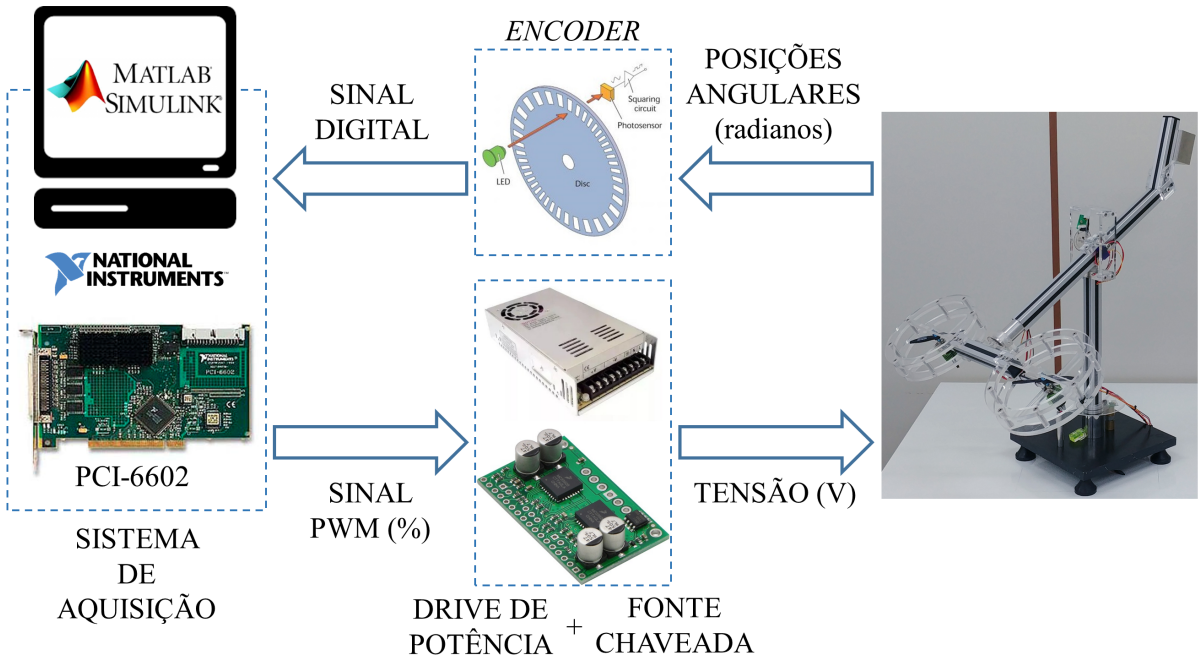
Fonte: Autoria própria.

4.9 CONSTRUÇÃO DOS EXPERIMENTOS DE CONTROLE TRADICIONAIS

Os experimentos de controle tradicionais, realizados presencialmente em laboratório, dependem da configuração do sistema de aquisição e controle apresentado na Figura 39. Os controles em malha fechada são projetados no MATLAB e sua construção e execução é realizada no Simulink. Por meio do hardware de aquisição PCI-6602, o Simulink é capaz de interagir com a planta de controle. A leitura do sinal digital do *encoder* permite determinar as variáveis de saída do Helicóptero com 3 GDL. A transmissão do sinal de PWM ao *drive* de potência permite atuar sobre a planta de controle.

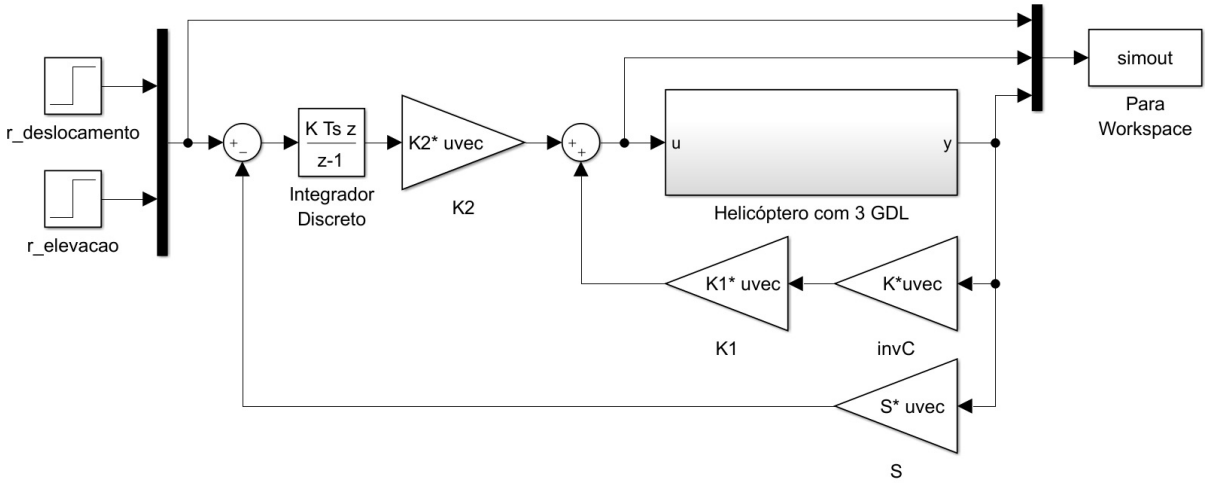
A Figura 40 exemplifica a construção de um diagrama de blocos do Simulink, correspondente a um controle seguidor com realimentação de estado, apresentado na Seção 3.3. A execução dos experimentos de controle no Helicóptero com 3 GDL depende do conteúdo do bloco "Helicóptero com 3 GDL". Os sinais de entrada e saída deste bloco são, respectivamente, o vetor de controle $\boldsymbol{u}$ e o vetor de saída $\boldsymbol{y}$, conforme definido pela modelagem do sistema na Seção 4.6. Para realizar os controles do sistema real, e dos modelos linear e não linear, basta apenas alterar o conteúdo do bloco "Helicóptero com 3 GDL".

Figura 39 – Sistema de aquisição e controle tradicional.



Fonte: Autoria própria.

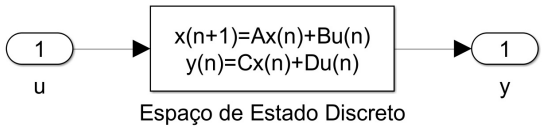
Figura 40 – Exemplo de diagrama de blocos para o controle seguidor com realimentação de estado.



Fonte: Autoria própria.

Para o controle do modelo linear foi configurado, dentro do bloco "Helicóptero com 3 GDL", o bloco de espaço de estado discreto, como representado na Figura 41. A discretização do modelo contínuo do Helicóptero com 3 GDL, dado nas Equações (71), (72), (73) e (74), é feita pelo método *zero-order hold*, implementado na função "c2d" do MATLAB.

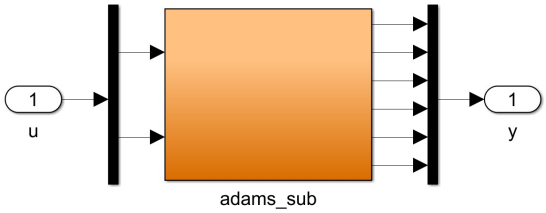
Figura 41 – Bloco de espaço de estado discreto.



Fonte: Autoria própria.

Para o controle do modelo não linear foi configurado, dentro do bloco "Helicóptero com 3 GDL", o bloco de cossimulação exportado pelo Adams, como representado na Figura 42.

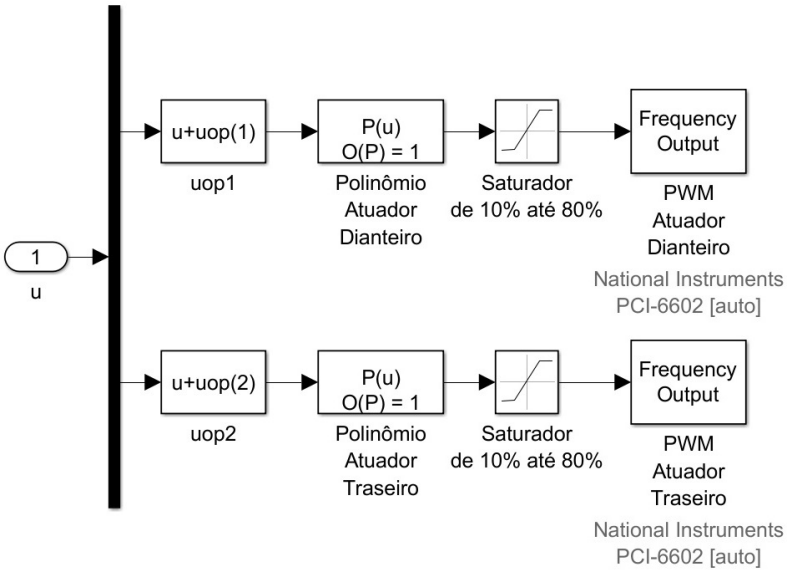
Figura 42 – Bloco de configuração da cossimulação entre Adams e Simulink.



Fonte: Autoria própria.

Para o controle do sistema real foi necessário configurar, dentro do bloco "Helicóptero com 3 GDL", a aplicação das variáveis de controle nos atuadores, apresentada na Figura 43, e a determinação das variáveis de saída a partir da leitura dos sensores, apresentada na Figura 44.

Figura 43 – Diagrama de blocos da aplicação das variáveis de controle.

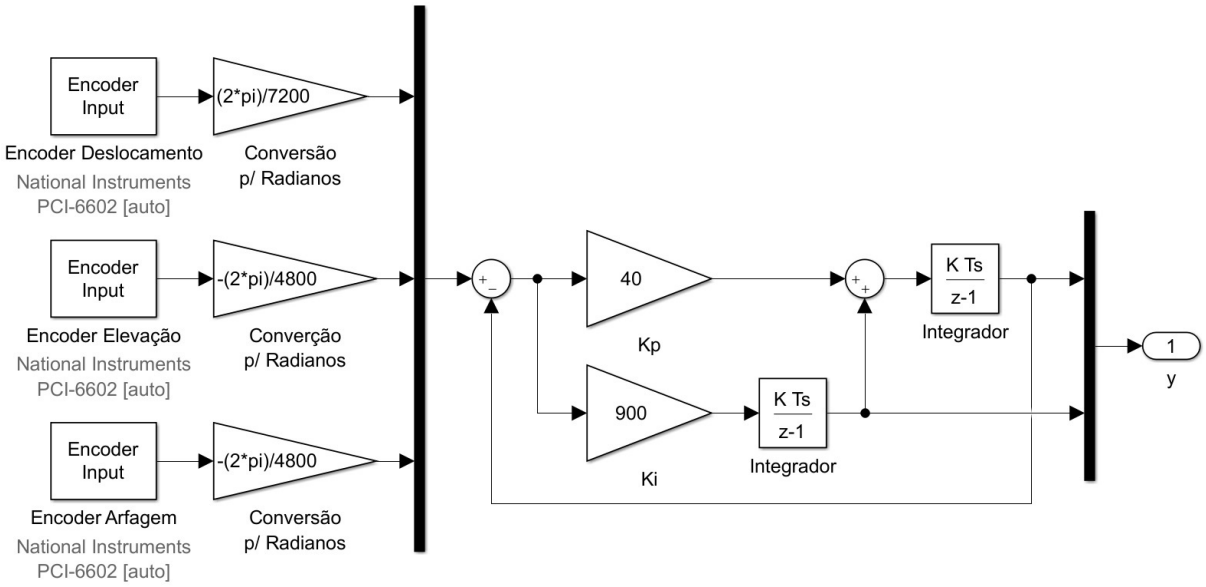


Fonte: Autoria própria.

No processo de aplicação das variáveis de controle (Figura 43), as forças de empuxo de operação, na Equação (70), foram somadas as forças de empuxo dianteiro e traseiro, calculadas pelo controlador. Em seguida, por meio dos polinômios identificados para os atuadores,

apresentados nas Equações (77) e (78), foi obtido o sinal de PWM correspondente as forças de empuxo. Os saturadores garantem que a ação de controle do sistema se mantenha no intervalo de 10% até 80%, determinado na Seção 4.7 para o funcionamento dos atuadores.

Figura 44 – Diagrama de blocos da estimação das variáveis de estado.



Fonte: Autoria própria.

No processo de determinação das variáveis de saída (Figura 44), a leitura da contagem de pulsos de cada um dos *encoders* é primeiro convertida para a posição angular em radianos. Em seguida, por meio do TL com ganho proporcional $K_P = 40$ e integral $K_I = 900$, são estimadas as variáveis de saída do sistema.

Quanto a construção do controle seguidor, exemplificada na Figura 40, são feitas as seguintes considerações. Os sinais de referência de deslocamento e elevação devem ser informados em radianos. O vetor de saída $\boldsymbol{y}$ é multiplicado pela matriz $\boldsymbol{C}^{-1}$ para determinar o vetor de estado $\boldsymbol{x}$ realimentado pelo controle. As variáveis de saída controladas pelo sistema são as posições angulares de deslocamento (y_1) e de elevação (y_2), selecionadas por meio da multiplicação do vetor de saída $\boldsymbol{y}$ por uma matriz $\boldsymbol{S}$, apresentada na Equação (79). As matrizes de ganhos $\boldsymbol{K_1}$ e $\boldsymbol{K_2}$ podem ser projetadas pelos métodos de atribuição de autoestrutura completa (Seção 3.4) ou LQR (Seção 3.5).

$$\boldsymbol{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{79}$$

4.10 OBSERVAÇÕES

Como apresentado nas Seções 2.1 e 4.1, o contrapeso presente no Helicóptero com 3 GDL tem o objetivo de reduzir a massa efetiva do helicóptero de modo que as forças de

empuxo, produzidas pelos atuadores dianteiro e traseiro, sejam suficientes para atuar sobre o sistema. As análises apresentadas na Seção 4.6 permitiram posicionar o contrapeso, deste modo determinou-se que as forças de empuxo necessárias para estabilizar o sistema na condição inicial, na qual as posições e velocidades angulares são zero, é de aproximadamente 0,3363 N para cada atuador, como apresentado na Equação (70). A identificação das forças produzidas pelos atuadores, apresentada na Seção 4.7, mostra que cada atuador é capaz de produzir forças de empuxo de até aproximadamente 0,85 N. Deste modo, os atuadores produzem forças de empuxo suficientes para controlar o Helicóptero com 3 GDL.

Na planta do Helicóptero com 3 GDL são utilizados cabos para realizar a instrumentação da parte elétrica dos atuadores e sensores. Num primeiro momento, na construção do protótipo virtual do Helicóptero com 3 GDL, descrita na Seção 4.6, desprezavam-se as massas destes cabos, por serem pequenas quando comparadas a massa dos corpos. Por exemplo, dos 956,41 g de massa do braço de sustentação (Figura 23) 26 g pertencem aos cabos, dos 1350,63 g de massa do helicóptero (Figura 24) 15 g são dos cabos. Embora as massas dos cabos sejam pequenas, ao construir o protótipo virtual sem os cabos e mantendo o contrapeso na mesma posição determinada na Seção 4.6, as forças de empuxo necessárias para estabilizar o sistema na condição inicial reduzem para aproximadamente 0,1987 N para cada atuador.

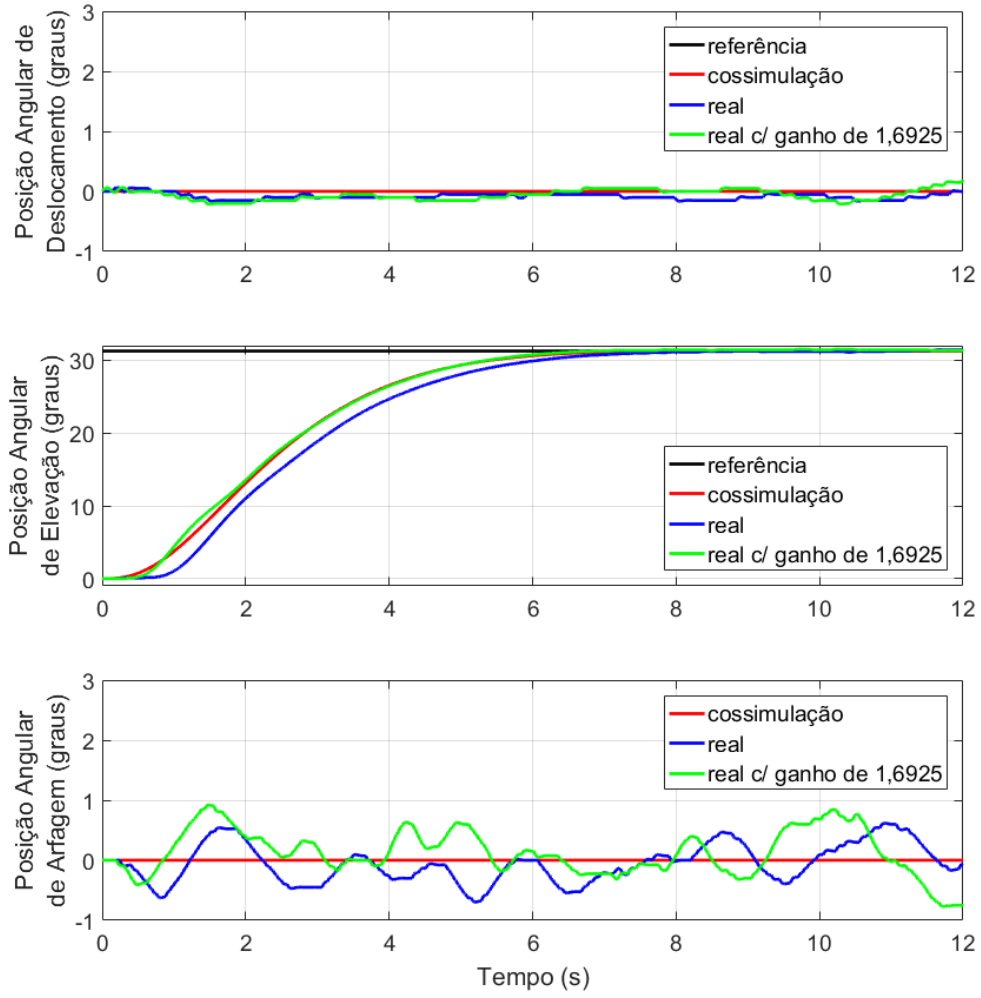
Portanto, ao desprezar os cabos o protótipo virtual do Helicóptero com 3 GDL passa a não representar adequadamente o comportamento do sistema real, pois as forças de empuxo dianteiro e traseiro necessárias para controlar o modelo linear e a cossimulação, produzidos pelo protótipo virtual do Adams, são menores que as necessárias para controlar o sistema real do Helicóptero com 3 GDL. Tal problema se reflete nas respostas de controle da posição angular de elevação e das forças de empuxo dianteiro e traseiro. Para representar isto graficamente, as Figuras 45 e 46 apresentam um exemplo de comparação das respostas de controle real e simulado obtidas para a prototipagem virtual sem cabos, o controle aplicado foi o seguidor com realimentação de estado, como descrito na Seção 4.9, projetado pelo método de atribuição de autoestrutura completa.

A Figura 45 apresenta a comparação das respostas de controle do sistema real e da cossimulação com o Adams para as posições angulares dos GDL de deslocamento, elevação e arfagem. As referências dos GDL controláveis de deslocamento e elevação são, respectivamente, de 0º e 31,27º. As respostas do controle real das posições angulares de deslocamento e arfagem correspondem as respostas obtidas pela cossimulação, embora apresentem um comportamento oscilatório. Tal oscilação é uma característica intrínseca do controle do sistema real, como analisado no Capítulo 6. A resposta do controle real da posição angular de elevação apresenta um atraso em comparação com a resposta da cossimulação, isto acontece pois as forças de empuxo determinadas pelo controlador não foram suficientes para elevar o helicóptero.

Considerando que as forças de empuxo necessárias para estabilizar o helicóptero na condição inicial são conhecidas, é possível calcular que um aumento de aproximadamente 69,25% é suficiente para que os 0,1987 N obtidos pelo protótipo virtual sem cabos correspondam aos

0,3363 N obtidos pelo protótipo virtual com cabos. Deste modo, ao controlar o Helicóptero com 3 GDL aplicando um ganho de 1,6925 sobre as forças de empuxo calculadas, o atraso na resposta de controle real da posição angular de elevação é corrigido, como mostrado na Figura 45.

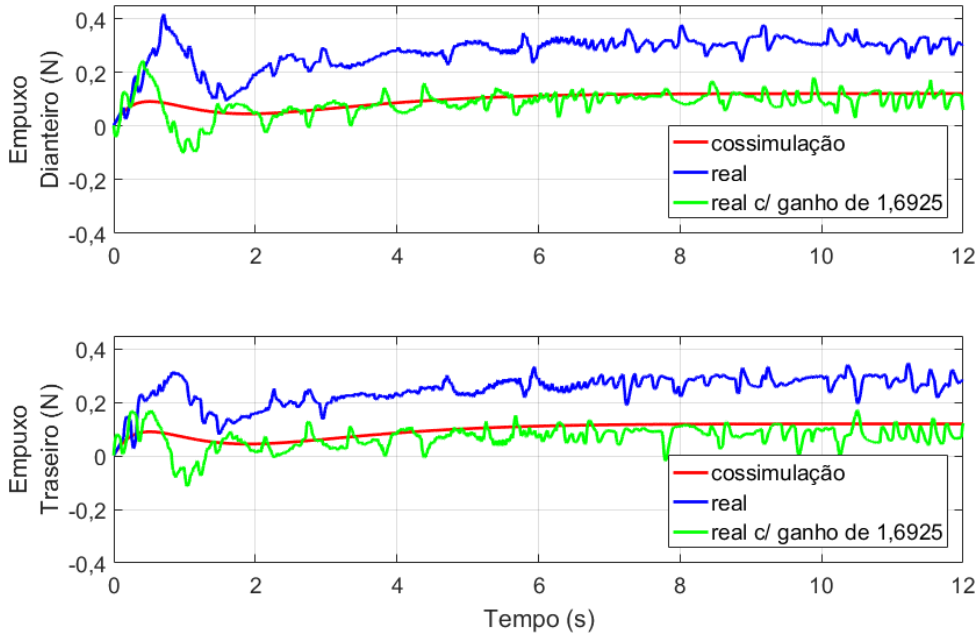
Figura 45 – Resposta de controle das posições angulares para a prototipagem virtual sem cabos.



Fonte: Autoria própria.

A Figura 46 apresenta a comparação entre as respostas das forças de empuxo para os controles por meio da cossimulação com o Adams e do sistema real, com e sem o ganho de 1,6925. Comparando as respostas de controle da cossimulação e do sistema real é possível perceber que a força necessária para controlar o sistema real é maior. Ao aplicar o ganho de 1,6925 as forças de empuxo previstas pelo protótipo virtual sem cabos passam a ser suficientes para controlar o sistema real. Neste contexto, toda a construção do protótipo virtual, apresentada na Seção 4.6, considera as propriedades de massa dos cabos. Deste modo, os resultados analisados no Capítulo 6 não apresentam os problemas discutidos nesta Seção.

Figura 46 – Resposta das forças de empuxo para a prototipagem virtual sem cabos.



Fonte: Autoria própria.

5 LABORATÓRIO VIRTUAL E REMOTO

Este capítulo apresenta a visão geral do LVR desenvolvido neste trabalho, incluindo a descrição das tecnologias utilizadas, a apresentação da arquitetura, a localização da malha fechada de controle e as funcionalidades desenvolvidas.

5.1 TECNOLOGIAS UTILIZADAS NO LABORATÓRIO VIRTUAL E REMOTO

O LVR desenvolvido é essencialmente uma aplicação Web composta por duas partes, a interface Web (*front-end*), que compõe a camada do cliente, e um serviço (*back-end*), que compõe a camada do servidor. De modo geral, a aplicação Web desenvolvida para o LVR está integrada a um sistema de aquisição e controle, em execução no lado do servidor, que permite ao usuário projetar e implementar diferentes técnicas de controle. Esta seção define brevemente as tecnologias utilizadas no desenvolvimento da aplicação Web e do sistema de aquisição e controle que compõem o LVR.

5.1.1 AngularJS v1.7.8

O AngularJS[1] é um *framework* estrutural, em JavaScript, utilizado na construção da interface de aplicações Web dinâmicas. Tais interfaces são denominadas *Single-Page Applications* (SPA ou, em português, Aplicações de Página Única), podendo ser organizadas segundo os padrões arquiteturais MVC (*Model-View-Controller*) ou MVVM (*Model-View-ViewModel*). Esta flexibilidade faz com que o AngularJS seja considerado um *framework* MVW (*Model-View-Whatever*), que em português significa "Modelo-Visão-Aquilo que servir melhor". O AngularJS estende a estrutura do HTML (*HyperText Markup Language*), por meio de atributos personalizados, permitindo expressar os componentes da interface de maneira sucinta e clara. O uso de todas estas características implica em uma redução do volume de código escrito no desenvolvimento com o AngularJS (HAVIV, 2014; ANGULARJS, 2019a; ANGULARJS, 2019b).

Existem módulos e *frameworks* externos, desenvolvidos para adicionar funcionalidades ao AngularJS. A seguir estão listados os recursos externos utilizados:

- **Angular Loading Bar v0.9.0**[2]**:** Este módulo exibe uma barra de carregamento sempre que uma requisição assíncrona por meio do objeto XmlHttpRequest é realizada;
- **ngStorage v0.3.11**[3]**:** Este módulo adapta as funcionalidades do Web Storage para seguirem o modo de trabalho do AngularJS;

[1] Página do AngularJS: https://angularjs.org/
[2] Página no GitHub do Angular Loading Bar: https://github.com/chieffancypants/angular-loading-bar
[3] Página no GitHub do ngStorage: https://github.com/gsklee/ngStorage

- **AngularJS Material v1.1.12[4]:** Este é um *framework* de recursos de interface prontos, como caixas de texto e abas de navegação.

5.1.2 Bootstrap v4.1.3

O Bootstrap[5] é um conjunto de ferramentas em HTML, CSS (*Cascading Style Sheets*) e JavaScript para desenvolvimento de interfaces responsivas de aplicações Web (BOOTSTRAP, 2019). Os vários recursos prontos do Bootstrap são de grande auxílio na construção do *design* de interfaces Web.

5.1.3 CodeMirror v5.47.0

O CodeMirror[6] é um editor de texto, para navegadores Web, versátil e desenvolvido em JavaScript. A especialidade desta ferramenta é a edição de códigos, tendo suporte a sintaxe de diversas linguagens de programação e sendo, inclusive, utilizado nas ferramentas de desenvolvimento dentro de navegadores como Mozilla Firefox e Google Chrome (CODEMIRROR, 2019).

5.1.4 Xterm.js v3.14.2

O Xterm.js[7] é um componente de interface, implementado em linguagem TypeScript, para construção de aplicações de terminal, ricas em recursos, dentro de navegadores Web. Este componente é utilizado em projetos populares como a IDE Visual Studio Code da Microsoft e a IDE Eclipse Theia, ambas desenvolvidas por meio de tecnologias Web (XTERM, 2019).

5.1.5 D3.js v1.6.2 e Rickshaw v1.6.2

O D3.js[8] é uma biblioteca, implementada em JavaScript, para construir a visualização de dados de maneira gráfica, por meio de HTML, SVG (*Scalable Vector Graphics*) e CSS (BOSTOCK, 2019). O Rickshaw[9] é um conjunto de ferramentas, construído sobre o D3.js, para construção de gráficos interativos de sinais evoluindo no tempo (SHUTTERSTOCK INC., 2019).

5.1.6 *HyperText Transfer Protocol*

O HTTP[10] (*HyperText Transfer Protocol*) é um protocolo de solicitação/resposta sem armazenamento de estado, aplicado em nível de sistemas distribuídos, colaborativos e

[4]Página do AngularJS Material: https://material.angularjs.org/
[5]Página do Bootstrap: http://getbootstrap.com/
[6]Página do CodeMirror: https://codemirror.net/
[7]Página do Xterm.js: https://xtermjs.org/
[8]Página do D3.js: https://d3js.org/
[9]Página do Rickshaw: https://tech.shutterstock.com/rickshaw/
[10]A especificação completa do HTTP/1.1 é disponibilizada nos documentos de RFC7230 até RFC7235 da Internet Engineering Task Force (IETF)

com informações em hipertexto. O HTTP foi projetado a fim de esconder os detalhes de implementação de um serviço por meio da apresentação de uma interface uniforme (FIELDING; RESCHKE, 2019). Simplificadamente, o HTTP age como um envelope estruturado para o envio de documentos. Ao realizar uma solicitação, um cliente Web insere um documento dentro de um envelope HTTP e envia para um servidor web. Este retorna uma resposta também em um envelope HTTP (RICHARDSON; RUBY, 2007).

5.1.7 WebSockets

O WebSockets permite estabelecer uma forma de comunicação persistente e bidirecional entre servidores e clientes web, de forma que ambos transmitam dados ao mesmo tempo. A padronização do protocolo[11] WebSockets é realizada pela Internet Engineering Task Force (IETF), enquanto a padronização da API[12] (*Aplication Programming Interface*) WebSockets é responsabilidade da World Wide Web Consortium (W3C) (PTERNEAS, 2013).

5.1.8 *JavaScript Object Notation*

O JSON[13] (*JavaScript Object Notation*) é uma sintaxe textual para serializar dados (números, vetores, vetores de caracteres e outros) de maneira estruturada, facilitando o envio e recebimento de dados entre diferentes linguagens de programação. A sintaxe do JSON teve inspiração na sintaxe dos objetos literais (*object literals*) da linguagem JavaScript e faz uso de caracteres como colchetes, dois pontos, chaves e vírgulas na estruturação dos dados (ECMA INTERNATIONAL, 2017; MOZILLA DEVELOPER NETWORK, 2019).

5.1.9 Servidor Ubuntu 18.04.3 LTS

Todos os programas no lado do servidor são executados em um computador com placa-mãe Asus P5QL SE, processador Core 2 Quad Q9650, 4GB de memória RAM, 240 GB de armazenamento SSD Kingston A400 e sistema operacional Linux da distribuição Ubuntu[14] 18.04.3 LTS (*Long Term Support*).

5.1.10 Node.js v10.15.3

O Node.js[15] é um ambiente para desenvolvimento em linguagem JavaScript, executado no lado do servidor. Este ambiente é construído sobre o Chrome V8[16], o motor de linguagem JavaScript utilizado no navegador Chrome, ambos desenvolvidos pela Google. A finalidade do Node.js é a execução de processos de longa duração sem a necessidade de *multithreading*, em

[11] Protocolo WebSockets: https://tools.ietf.org/html/rfc6455
[12] API WebSockets: https://www.w3.org/TR/websockets/
[13] Padrão ECMA-404: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[14] Página do Ubuntu: https://ubuntu.com/
[15] Página do Node.js: https://nodejs.org/
[16] Página do Chrome V8: https://v8.dev/

servidor. Para isto, é implementada uma estrutura orientada a eventos, de *thread* única e sem bloqueio de entrada e saída (em inglês, *non-blocking I/O*) (TILKOV; VINOSKI, 2010). A filosofia de trabalho do Node.js, segundo Casciaro (2014), é resumida no seguinte conjunto de princípios:

- **Núcleo pequeno:** O Node.js apresenta um núcleo de funcionalidades de trabalho reduzido e bem documentado[17];
- **Módulos pequenos:** O Node.js estrutura a criação de aplicações e bibliotecas reutilizáveis por meio do conceito de módulos;
- **Área superficial pequena:** Como forma de aumentar a clareza e reusabilidade dos códigos implementados, os módulos em Node.js expõem um conjunto reduzido de funcionalidades;
- **Simplicidade e pragmatismo:** A combinação destes, com os princípios anteriores é que torna o desenvolvimento em Node.js adaptável, rápido e de fácil manutenção.

Por conta desta filosofia, a implementação de funcionalidades mais complexas depende da importação de módulos externos. O NPM (*Node Package Manager*) é a ferramenta que gerencia a importação destes módulos. A seguir estão listados os módulos externos do Node.js utilizados no desenvolvimento do LVR:

- **body-parser v1.19.0**[18]**:** Este módulo extrai as informações do corpo da solicitação HTTP, a fim de facilitar o uso destas;
- **chokidar v3.0.1**[19]**:** Este módulo permite observar um diretório, detectando quando um arquivo é, entre outros eventos, criado, removido ou atualizado;
- **cookie-parser v1.4.4**[20]**:** Este módulo extrai as informações dos *cookies*, que acompanham a solicitação HTTP, a fim de facilitar o uso destas;
- **csurf v1.10.0**[21]**:** Este módulo auxilia na proteção contra ataques do tipo CSRF (*Cross Site Request Forgery*), sofridos por aplicações Web;
- **express v4.17.0**[22]**:** Este módulo constitui o *framework* Express[23], que possui um conjunto minimalista de funcionalidades para a estruturação e desenvolvimento de serviços para aplicações Web (HAVIV, 2014);
- **express-ws v4.0.0**[24]**:** Este módulo acrescenta os recursos necessários no desenvolvimento de serviços utilizando WebSockets, por meio do *framework* Express;
- **glob v7.1.4**[25]**:** Este módulo consulta os arquivos presentes em um diretório, por meio

[17]Documentação da API do Node.js v10.x: https://nodejs.org/docs/latest-v10.x/api/
[18]Página do body-parser no NPM: https://www.npmjs.com/package/body-parser
[19]Página do chokidar no NPM: https://www.npmjs.com/package/chokidar
[20]Página do cookie-parser no NPM: https://www.npmjs.com/package/cookie-parser
[21]Página do csurf no NPM: https://www.npmjs.com/package/csurf
[22]Página do express no NPM: https://www.npmjs.com/package/express
[23]Página do Express: http://expressjs.com/
[24]Página do express-ws no NPM: https://www.npmjs.com/package/express-ws
[25]Página do glob no NPM: https://www.npmjs.com/package/glob

de sintaxe especial. Por exemplo, para obter todos os arquivos de texto basta pesquisar pelo termo "*.txt";

- **jsonwebtoken v8.5.1**[26]**:** Este módulo codifica e decodifica JWTs[27] (*JSON Web Tokens*). Estes são *tokens* de formato compacto, utilizados pelo servidor para verificar a autorização de usuários (JONES; BRADLEY; SAKIMURA, 2015);
- **morgan v1.9.1**[28]**:** Este módulo lista as requisições enviadas para o servidor web;
- **node-pty v0.8.1**[29]**:** Este módulo integra aplicações em Node.js com processos de CLIs;
- **npmlog v4.1.2**[30]**:** Este módulo formata as mensagens exibidas na saída de texto padrão;
- **opencv4nodejs v5.1.0**[31]**:** Este módulo adiciona as capacidades de visão computacional à aplicações em Node.js.

5.1.11 *Universal Asynchronous Receiver/Transmitter*

Os dispositivos presentes em sistemas embarcados com a finalidade de permitir o envio e recebimento de dados de modo serial são os periféricos de comunicação, como a serial UART (*Universal Asynchronous Receiver/Transmitter*). Esta forma de comunicação é *full-duplex*, ou seja é possível enviar e receber dados ao mesmo tempo, por meio de vias exclusivas de transmissão e recepção. Além disso, a comunicação UART é assíncrona, deste modo o lado transmissor não precisa enviar um sinal de *clock* para que o receptor sincronize a leitura. Para o envio e recebimento de dados ser trabalhado da mesma forma, os dispositivos de transmissão e recepção devem estar de acordo quanto as seguintes configurações (MOLLOY, 2014):

- **Taxa de transmissão:** Esta configuração é dada em *bits* por segundo (bps), podendo assumir diferentes valores como 9600 bps, 38400 bps, 57600 bps, 115200 bps e outras;
- **Bit de paridade:** Esta configuração determina se o *bit* de paridade, para verificação de erros, está presente na transmissão de dados;
- **Bits de parada:** Esta configuração determina se o número de *bits* de parada é 1 ou 2.

5.1.12 Octave v5.1.0

O Octave[32] é uma linguagem de alto nível, sendo a computação numérica a sua finalidade primária. Por meio de uma CLI, o Octave pode ser utilizado convenientemente para solução de problemas numéricos lineares e não lineares, integração de funções ordinárias, manipulação de polinômios, entre outras finalidades. Uma característica interessante do Octave é sua ampla compatibilidade com a linguagem do MATLAB (EATON, 2019).

[26]Página do jsonwebtoken no NPM: https://www.npmjs.com/package/jsonwebtoken
[27]Padronização do JWT: https://tools.ietf.org/html/rfc7519
[28]Página do morgan no NPM: https://www.npmjs.com/package/morgan
[29]Página do node-pty no NPM: https://www.npmjs.com/package/node-pty
[30]Página do npmlog no NPM: https://www.npmjs.com/package/npmlog
[31]Página do opencv4nodejs no NPM: https://www.npmjs.com/package/opencv4nodejs
[32]Página do Octave: https://www.gnu.org/software/octave/

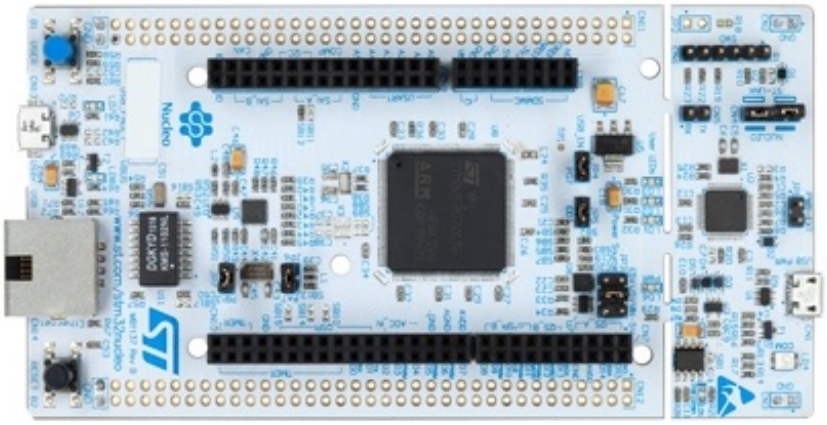
Parte das funcionalidades do Octave são disponibilizadas por pacotes externos, gerenciados pelo Octave Forge[33], sendo comparáveis as *toolboxes* do MATLAB. Os pacotes externos utilizados neste trabalho são:

- **control v3.2.0[34]:** Este pacote adiciona as funcionalidade de computação em teoria de sistemas de controle;
- **signal v1.4.1[35]:** Este pacote adiciona as funcionalidades de processamento de sinais;
- **instrument-control v0.4.0[36]:** Este pacote adiciona as funcionalidades de integração com hardware em baixo nível.

5.1.13 NUCLEO-F767ZI

O NUCLEO-F767ZI[37], apresentado na Figura 47, é uma placa de desenvolvimento embarcado, comercializada pela STMicroelectronics. Esta placa é equipada com o microcontrolador Arm Cortex-M7 STM32F767ZI[38], que suporta decodificação de *encoders* de quadratura, geração de sinais de PWM, comunicação serial UART, além de possuir uma FPU (*Floating Point Unit* ou Unidade de Ponto Flutuante, em português) com suporte a processamento de instruções de ponto flutuante com precisão única ou dupla. O NUCLEO-F767ZI também vem equipado com a ferramenta ST-LINK/V2-1 para programação das aplicações embarcadas (STMICROELECTRONICS, 2019).

Figura 47 – Placa de desenvolvimento NUCLEO-F767ZI.



Fonte: Adaptada de STMicroelectronics (2020).

[33]Página do Octave Forge: https://octave.sourceforge.io/
[34]Página do pacote control no Octave Forge: https://octave.sourceforge.io/control/
[35]Página do pacote signal no Octave Forge: https://octave.sourceforge.io/signal/
[36]Página do pacote instrument-control no Octave Forge: https://octave.sourceforge.io/instrument-control/
[37]Página oficial da STMicroelectronics para o NUCLEO-F767ZI: https://www.st.com/en/evaluation-tools/nucleo-f767zi.html
[38]Página oficial da STMicroelectronics para o STM32F767ZI: https://www.st.com/en/microcontrollers-microprocessors/stm32f767zi.html

Como o NUCLEO-F767ZI é um dispositivo "Arm Mbed Enabled", as aplicações embarcadas podem ser desenvolvidas por meio do Arm Mbed OS[39] (*Operating System* ou Sistema Operacional, em português). Este é um RTOS (*Real-Time Operating System* ou Sistema Operacional de Tempo Real, em português) para o desenvolvimento de aplicações em IoT (*Internet of Things* ou Internet das Coisas, em português). O Arm Mbed OS, por meio de uma camada de abstração, disponibiliza funções para uma variedade de hardwares de diferentes fabricantes, permitindo assim focar no desenvolvimento das aplicações em linguagem C/C++. A compilação das aplicações para o Arm Mbed OS pode ser feita *online*, por meio de uma IDE, ou *offline*, no caso desta são necessárias as seguintes ferramentas (ARM, 2019a; ARM, 2019b):

- **Arm Mbed CLI v1.9.0[40]:** Esta é uma interface de linha de comando para construir e compilar as aplicações em Arm Mbed OS para diferentes placas embarcadas, por meio de diferentes *toolchains*;
- **GNU Arm Embedded Toolchain v8.2.1[41]:** Este é um conjunto de ferramentas *open source* para compilação de aplicações em C/C++ e Assembly, para dispositivos Arm Cortex-M e Cortex-R.

5.1.14 PlayStation Eye

O PlayStation Eye é a câmera digital do PlayStation 3, tendo capacidade de geração de vídeo em resolução 640x480 *pixels* e taxa de atualização de até 60 Hz, ou em resolução 320x240 *pixels* e taxa de atualização de até 120 Hz. A interface de comunicação do PlayStation Eye é a USB (*Universal Serial Bus*).

5.2 BLOCK DIAGRAM CODER

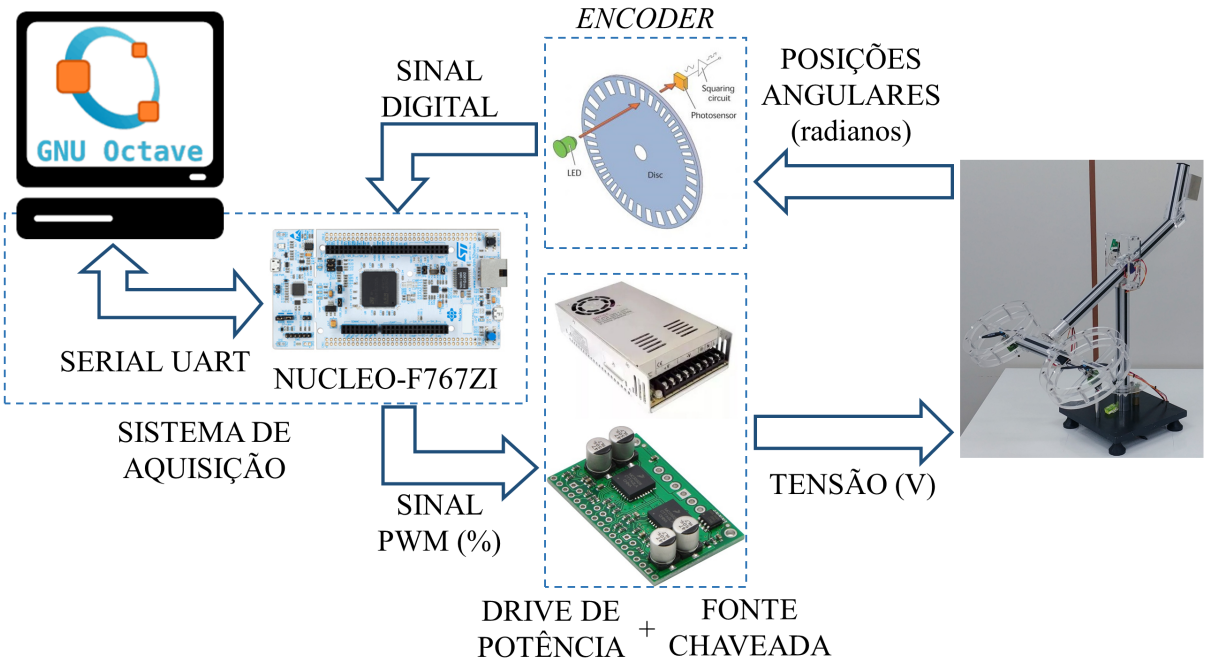
O BDC é um pacote de código desenvolvido para permitir ao usuário implementar e executar técnicas de controle tanto em simulação linear discreta, quanto em plantas reais. Por meio do pacote BDC, do Octave CLI e da placa NUCLEO-F767ZI, é possível alterar o sistema de aquisição e controle tradicional, apresentada pela Seção 4.9 na Figura 39, pelo sistema de aquisição e controle utilizado no LVR, representado na Figura 48. Nesta configuração, o Octave assume a função de projetar e construir os controles em malha fechada. Para o caso do controle do modelo linear discreto, a execução ocorre no próprio Octave. No caso do controle real da planta, a execução é realizada no NUCLEO-F767ZI, sendo este o responsável por realizar a leitura dos sensores e acionamento dos atuadores.

[39] Página oficial do Arm Mbed OS: https://os.mbed.com/

[40] Página no GitHub da Arm Mbed CLI: https://github.com/ARMmbed/mbed-cli

[41] Página oficial do GNU Arm Embedded Toolchain: https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm

Figura 48 – Sistema de aquisição e controle do LVR.



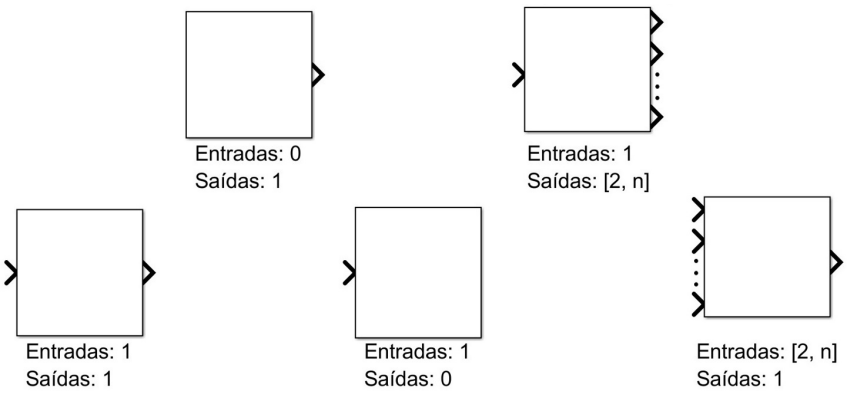
Fonte: Autoria própria.

O BDC foi inspirado na capacidade do Simulink de construir técnicas de controle por meio de diagramas de blocos. No BDC, porém, o diagrama de blocos é construído por *script*, com auxílio de um conjunto de funções prontas. O BDC foi desenvolvido em linguagem MATLAB, sendo totalmente suportado pelo Octave. A implementação do BDC utilizou os recursos de orientação a objetos disponibilizados pelo MATLAB, permitindo organizar os dados e funcionalidades em estruturas lógicas, ou seja, objetos. Tal abordagem é importante no gerenciamento da complexidade do pacote de software desenvolvido. Dentre os recursos da orientação a objetos disponíveis, foram utilizados a criação e herança de classes (MATHWORKS, 2019b).

5.2.1 Considerações gerais

Os elementos principais de um diagrama são os blocos, sendo cada bloco responsável pela implementação de uma funcionalidade específica. Os blocos recebem sinais pelas portas de entrada, e transmitem sinais pelas portas de saída. No BDC, quanto à quantidade de portas, os blocos assumem uma das configurações na Figura 49.

Figura 49 – Diferentes configurações de portas dos blocos.



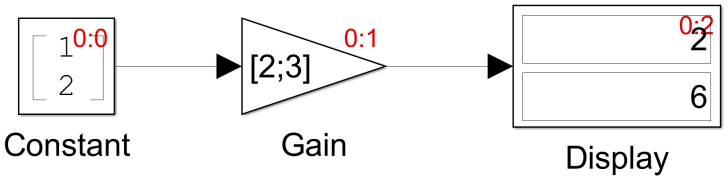
Fonte: Autoria própria.

Quanto à conexão entre as portas de saída e entrada, o BDC respeita as seguintes convenções:

- Não é possível a conexão entre duas portas de entrada;
- Não é possível a conexão entre duas portas de saída;
- Uma porta de saída pode ser conectada a várias portas de entrada diferentes;
- Uma porta de entrada recebe a conexão de apenas uma porta de saída.

A inserção e conexão dos blocos em um diagrama é arbitrária. Deste modo, é necessário determinar a ordem de execução das funcionalidades dos blocos. A Figura 50 apresenta um diagrama de blocos executado no Simulink, no canto superior direito de cada bloco está representada sua respectiva ordem de execução. Neste exemplo, a execução parte do bloco Constant, seguindo para o Gain e por fim o Display.

Figura 50 – Ordem de execução dos blocos no Simulink.



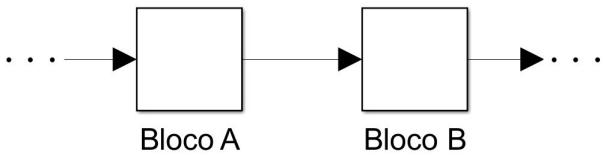
Fonte: Autoria própria.

O algoritmo implementado no BDC para determinar a ordem de execução dos blocos segue as convenções utilizadas pelo Simulink. Este define que os blocos apresentam ou não portas de passagem direta. Na execução de um bloco com porta de passagem direta, em um mesmo instante de amostragem, o sinal existente na porta de entrada determina diretamente o sinal transmitido na porta de saída. Partindo desta definição e considerando o exemplo de conexão entre blocos na Figura 51, as seguintes convenções são respeitadas na determinação

da ordem de execução dos blocos (MATHWORKS, 2016b):

- Se a porta de entrada do Bloco B for de passagem direta, então o Bloco A deve ser executado antes do Bloco B. Isto garante que a cada instante de amostragem o sinal de entrada do Bloco B é o mais atual;
- Blocos que não possuem porta de entrada com passagem direta podem ser executados em qualquer ordem, desde que seja respeitada a convenção anterior.

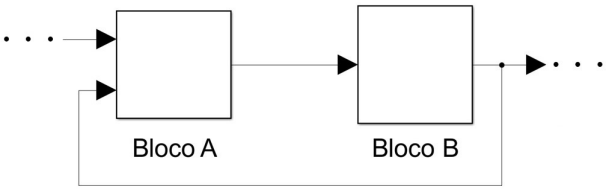
Figura 51 – Exemplo de conexão entre blocos.



Fonte: Autoria própria.

Em um diagrama pode ocorrer a situação representada na Figura 52, na qual todos os blocos envolvidos possuem portas de entrada de passagem direta. Nesta situação, denominada laço algébrico, existe uma dependência circular entre as portas de entrada e saída dos blocos, resultando em um problema que deve ser resolvido a cada instante de amostragem. O Simulink possui algoritmos, que podem ou não resolver a execução de diagramas com laço algébrico (MATHWORKS, 2016b). O BDC, porém, impede a construção e execução de diagramas que apresentem laço algébrico.

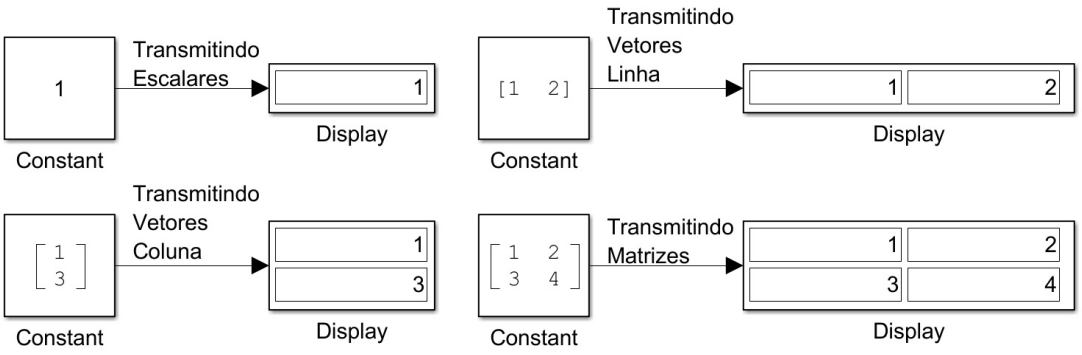
Figura 52 – Exemplo de um diagrama com laço algébrico.



Fonte: Autoria própria.

Os diagramas de blocos do BDC são executados em tempo discreto e fixo. Deste modo, a transmissão dos sinais de uma porta de saída para uma porta de entrada é realizada amostra por amostra. O Simulink transmite amostras na forma de escalar, vetor linha, vetor coluna e matriz, como apresentado na Figura 53. A implementação do BDC, porém, limita-se a amostras em escalar e vetor coluna.

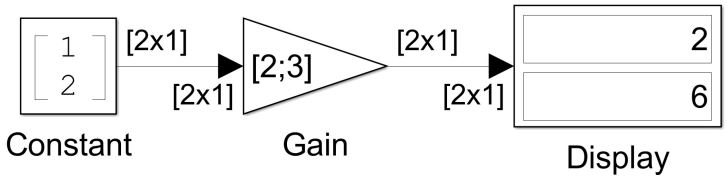
Figura 53 – Formas de transmissão de cada amostra de um sinal no Simulink.



Fonte: Autoria própria.

Voltando a considerar o diagrama representado na Figura 50, o vetor coluna transmitido pelo bloco Constant é multiplicado, elemento por elemento, pelo vetor coluna do bloco Gain. A execução deste diagrama depende dos vetores coluna, dos blocos Constant e Gain, terem a mesma dimensão. A Figura 54 apresenta a verificação das dimensões realizada pelo Simulink. No BDC a verificação das dimensões é realizada após a determinação da ordem de execução dos blocos do diagrama.

Figura 54 – Dimensão das amostras dos sinais transmitidos pelos blocos no Simulink.



Fonte: Autoria própria.

5.2.2 Desenvolvimento das técnicas de controle

A utilização de diagramas de blocos permite representar as funções executadas por cada componente do sistema de controle e também o fluxo de suas informações (D'AZZO; HOUPIS, 1995). Sendo assim, os diagramas de blocos constituem uma forma conveniente para definir o funcionamento das técnicas de controle. O desenvolvimento destas, por meio do BDC, depende da execução sequencial das seguintes etapas:

- **Definição da técnica de controle:** Nesta etapa, um diagrama é configurado, todos os blocos das funcionalidades necessárias são adicionados ao diagrama e, por fim, os blocos são interconectados para representar o fluxo de informações entre as funcionalidades executadas pela técnica de controle;
- **Construção do controlador:** Partindo do diagrama de blocos definido, deve-se determinar a ordem de execução dos blocos e verificar se a dimensão dos sinais transmitidos

entre estes está correta. Em seguida, a representação do diagrama de blocos é convertida em um código equivalente. Para o caso da simulação de um controle linear, o código é construído em linguagem MATLAB/Octave. Já no caso do controle do sistema real, o código é construído na linguagem C/C++ para o Arm Mbed OS, compilado pelo Arm Mbed CLI e programado na placa NUCLEO-F767ZI. O código gerado é simples, correspondendo apenas a lógica da técnica de controle. Toda configuração necessária para execução do controlador, seja a leitura dos sensores, acionamento dos atuadores ou comunicação serial entre Octave CLI e NUCLEO-F767ZI são pré-programadas. Por fim, é configurado um *script* para chamada da execução do controlador, isto garante que todos os dados obtidos da execução do controle sejam acessíveis dentro do Octave CLI;
- **Execução do controlador:** Nesta etapa, apenas executa-se o *script* de chamada do controlador configurado pela etapa anterior. Para terminar a execução do controlador basta sinalizar ao terminal com o comando "CTRL + C", por meio deste sinal o BDC é capaz de finalizar corretamente o controle executado no NUCLEO-F767ZI.

As funcionalidades prontas para execução destas etapas, por meio do BDC, estão descritas no restante desta seção.

5.2.3 Propriedades gerais dos blocos

Embora cada tipo de bloco em um diagrama realize uma funcionalidade diferente, existem partes da implementação comuns a todos os blocos desenvolvidos. No BDC, a lógica comum à todos os blocos é implementada por meio da classe BlockClass, de modo a ser transmitida a cada bloco específico por herança. A classe BlockClass é responsável pela configuração das seguintes propriedades dos blocos:

- **name:** Esta propriedade é o identificador único de cada bloco, deve ser um nome de variável válido segundo a sintaxe da linguagem MATLAB/Octave e apresentar no máximo 30 caracteres;
- **inputPorts:** Cada porta de entrada de um bloco é representada por um objeto da classe InputPortClass. Caso o bloco apresente mais de uma porta de entrada, a propriedade inputPorts assume um vetor coluna de objetos de InputPortClass. Caso o bloco não possua porta de entrada, a propriedade inputPorts é nula;
- **outputPorts:** Cada porta de saída de um bloco é representada por um objeto da classe OutputPortClass. Caso o bloco apresente mais de uma porta de saída, a propriedade outputPorts assume um vetor coluna de objetos de OutputPortClass. Caso o bloco não possua porta de saída, a propriedade outputPorts é nula;
- **priority:** Esta propriedade pode assumir um valor inteiro no intervalo [0,10], quanto maior o valor, maior a prioridade na execução do bloco. Tal propriedade classifica os blocos de acordo com o tipo de porta de entrada, permitindo determinar a ordem de

execução dos blocos do diagrama;

- **sorted:** Esta propriedade assume um valor lógico verdadeiro ou falso. Tal propriedade é utilizada na determinação da ordem de execução dos blocos do diagrama. Caso o bloco já esteja ordenado o valor é verdadeiro, caso contrário é falso;
- **dimensioned:** Esta propriedade assume um valor lógico verdadeiro ou falso. Tal propriedade é utilizada na verificação das dimensões dos sinais em um diagrama. Caso o bloco já esteja verificado o valor é verdadeiro, caso contrário é falso.

5.2.4 Propriedades das portas

As portas de entrada e saída definidas, respectivamente, pelas classes InpuPortClass e OutputPortClass possuem propriedades em comum, herdadas da classe PortClass. As propriedades desta classe são:

- **name:** Esta propriedade é o identificador único do bloco, definido pela classe BlockClass, ao qual a porta pertence;
- **port:** Esta propriedade é um valor inteiro maior ou igual à 1, identificando unicamente cada uma das portas do bloco. Este valor é interpretado separadamente para as portas de entrada e saída. Para um bloco com uma porta de entrada e uma porta de saída, por exemplo, ambas portas são identificados pelo valor 1.

Além das propriedades herdadas de PortClass, a classe OutputPortClass possui a seguinte propriedade:

- **dimension:** Esta propriedade indica a dimensão do sinal sendo transmitido pela respectiva porta de saída. Como o BDC limita-se a transmissão de amostras em escalar e vetor coluna, a propriedade dimension assume um valor inteiro maior ou igual à 1.

Além das propriedades herdadas de PortClass, a classe InputPortClass possui a seguinte propriedade:

- **source:** Esta propriedade corresponde a um objeto da classe OutputPortClass. Tal propriedade permite definir a ligação da porta de saída de um bloco na porta de entrada de outro bloco, por meio da cópia do respectivo objeto OutputPortClass para a propriedade source.

5.2.5 Configurando um diagrama de blocos

No BDC, toda a lógica de configuração de um diagrama, assim como as funcionalidades de adicionar e realizar a conexão entre blocos é implementada na classe Diagram. As

propriedades desta classe são:

- **name:** Esta propriedade é o identificador único de cada diagrama construído, deve ser um nome de variável válido segundo a sintaxe da linguagem MATLAB/Octave e apresentar no máximo 30 caracteres;
- **sampleTime:** Esta propriedade é o tempo fixo de amostragem definido para a execução discreta do diagrama de blocos. O sampleTime assume um valor numérico maior que zero e múltiplo de 0,001 segundo, ou seja, o menor tempo de amostragem possível é de 1 milissegundo;
- **stopTime:** Esta propriedade define o tempo total de execução do diagrama. O stopTime deve ser um valor numérico maior que zero e múltiplo do valor associado a propriedade sampleTime;
- **solver:** Para a versão do BDC apresentada neste trabalho esta propriedade limita-se a configuração 'discrete', uma vez que a execução dos diagramas é discreta;
- **mode:** Esta propriedade pode assumir as configurações 'simulation' ou 'external'. Caso o mode seja configurado como 'simulation' o BDC executa a simulação discreta do diagrama de blocos, em linguagem MATLAB/Octave. Caso o mode seja configurado como 'external' o BDC executa o controle da planta real;
- **target:** Esta propriedade permite selecionar o dispositivo embarcado, no qual é executado o controle da planta real. Para a versão atual do BDC a única configuração de target disponível é 'NUCLEO-F767ZI', que corresponde a placa embarcada apresentada na Subseção 5.1.13;
- **BLOCKS:** Esta propriedade corresponde a um vetor de células (cell array) da linguagem MATLAB, para armazenar todos os blocos adicionados ao diagrama. Neste vetor de células são armazenados, em coluna, todos os objetos que herdam a classe BlockClass.

Para configurar um diagrama de blocos, em um *script* da linguagem MATLAB/Octave, basta criar um objeto da classe Diagram. O Algoritmo 1 exemplifica este processo, deve-se substituir os parâmetros dentro dos parênteses pelos valores desejados.

Algoritmo 1 – Configurando um objeto da classe Diagram.

```
1 diagram = Diagram(name,sampleTime,stopTime,solver,mode,target);
```

Fonte: Autoria própria.

5.2.6 Adicionando um bloco Constant

A funcionalidade de um bloco Constant é gerar um sinal constante (MATHWORKS, 2016a). No BDC, um bloco Constant é definido pela classe ConstantBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Constant também apresenta

a seguinte propriedade:

- **constant:** Este é o valor constante do sinal gerado, deve ser um escalar ou vetor coluna de valores numéricos.

As características do bloco Constant são listadas abaixo:

- Possui apenas uma porta de saída;
- Como não possui porta de entrada, associa-se o valor 10 à propriedade priority herdada de BlockClass;
- A dimensão do sinal gerado na porta de saída deve corresponder a dimensão da propriedade constant.

Para adicionar um bloco Constant basta executar o método addConstant da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de ConstantBlock. O Algoritmo 2 exemplifica este processo.

Algoritmo 2 – Adicionando um bloco Constant.

```
1 addConstant(diagram,name,constant);
```

Fonte: Autoria própria.

5.2.7 Adicionando um bloco Signal Generator

A funcionalidade de um bloco Signal Generator é gerar sinais em diferentes formas de onda (MATHWORKS, 2016a). No BDC, um bloco Signal Generator é definido pela classe SignalGeneratorBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Signal Generator também apresenta as seguintes propriedades:

- **waveForm:** Esta propriedade determina a forma de onda, podendo ser 'square' (onda quadrada), 'sine' (onda senoidal) ou 'triangle' (onda triangular);
- **offset:** Este é o valor de *offset* somado ao sinal gerado, deve ser um escalar numérico;
- **amplitude:** Este é o valor de amplitude do sinal gerado, deve ser um escalar numérico;
- **period:** Este é o período do sinal gerado, deve ser um escalar numérico;
- **dutyCycle:** Esta é a razão cíclica do sinal gerado, deve ser um escalar inteiro no intervalo [0,100]. Esta propriedade é configurada apenas para a waveForm 'square'.

As características do bloco Signal Generator são listadas abaixo:

- Possui apenas uma porta de saída;
- Como não possui porta de entrada, associa-se o valor 10 à propriedade priority herdada de BlockClass;
- O sinal gerado na porta de saída tem dimensão escalar.

Para adicionar um bloco Signal Generator basta executar o método addSignalGenerator da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de SignalGeneratorBlock. O Algoritmo 3 exemplifica este processo.

Algoritmo 3 – Adicionando um bloco Signal Generator.

```
1 addSignalGenerator(diagram,name,waveForm,offset,amplitude,period,dutyCycle);
```

Fonte: Autoria própria.

5.2.8 Adicionando um bloco Scope

A funcionalidade de um bloco Scope é exibir graficamente os sinais em sua porta de entrada, durante a execução do diagrama (MATHWORKS, 2016a). No BDC, o bloco Scope é definido pela classe ScopeBlock, que herda a classe BlockClass. As características do bloco Scope são listadas abaixo:

- Possui apenas uma porta de entrada;
- A porta de entrada é de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- No BDC, os sinais na porta de entrada do bloco Scope são estruturados em JSON e armazenados em um diretório predefinido, em um arquivo com a extensão arbitrária ".scplog";
- No caso da execução do controle da planta real, por meio do NUCLEO-F767ZI, os sinais exibidos pelo bloco Scope são recebidos pelo Octave por comunicação serial UART;
- No BDC, cada diagrama pode ter no máximo 3 blocos Scope, e cada um destes pode receber no máximo um sinal de dimensão 5, na porta de entrada.

Para adicionar um bloco Scope basta executar o método addScope da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco e a propriedade name de BlockClass. O Algoritmo 4 exemplifica este processo.

Algoritmo 4 – Adicionando um bloco Scope.

```
1 addScope(diagram,name);
```

Fonte: Autoria própria.

5.2.9 Adicionando um bloco To Workspace

A funcionalidade de um bloco To Workspace é disponibilizar, no workspace, os sinais em sua porta de entrada (MATHWORKS, 2016a). No BDC, um bloco To Workspace é definido pela classe ToWorkspaceBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco To Workspace também apresenta as seguintes propriedades:

- **variableName:** Este é o nome da variável utilizado para salvar os dados no workspace, deve ser válido segundo a sintaxe da linguagem MATLAB/Octave e apresentar no máximo 30 caracteres;
- **saveFormat:** Esta propriedade indica o formato no qual os dados serão salvos, podendo ser 'Structure' ou 'Structure with time'. Em ambos casos os dados são salvos como uma estrutura (structure array). Porém no caso do formato 'Structure with time', além dos sinais também é armazenado o vetor de tempo da execução do diagrama.

As características do bloco To Workspace são listadas abaixo:

- Possui apenas uma porta de entrada;
- A porta de entrada é de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- No caso da execução do controle da planta real, por meio do NUCLEO-F767ZI, os sinais armazenados pelo bloco To Workspace são recebidos pelo Octave por comunicação serial UART;
- No BDC, cada diagrama pode ter no máximo 2 blocos To Workspace, e cada um destes pode receber no máximo um sinal de dimensão 10, na porta de entrada.

Para adicionar um bloco To Workspace basta executar o método addToWorkspace da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de ToWorkspaceBlock. O Algoritmo 5 exemplifica este processo.

Algoritmo 5 – Adicionando um bloco To Workspace.

```
1 addToWorkspace(diagram,name,variableName,saveFormat);
```

Fonte: Autoria própria.

5.2.10 Adicionando um bloco Sum

A funcionalidade de um bloco Sum é fazer a adição ou subtração dos sinais em suas portas de entrada (MATHWORKS, 2016a). No BDC, um bloco Sum é definido pela classe SumBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Sum também apresenta a seguinte propriedade:

- **signs:** É um vetor linha composto por no mínimo 2 caracteres '+' e/ou '-'. Caso esta propriedade seja '++-', por exemplo, o bloco Sum apresentará 3 portas de entrada, os sinais das duas primeiras portas de entrada são somados e o sinal da última porta de entrada é subtraido.

As características do bloco Sum são listadas abaixo:

- Possui o número de portas de entrada equivalente ao número de caracteres na propriedade signs;
- Possui uma porta de saída;
- As portas de entrada são de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- Todos os sinais nas portas de entrada devem apresentar a mesma dimensão, que por sua vez é a dimensão do sinal transmitido na porta de saída.

Para adicionar um bloco Sum basta executar o método addSum da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de SumBlock. O Algoritmo 6 exemplifica este processo.

Algoritmo 6 – Adicionando um bloco Sum.

```
1 addSum(diagram,name,signs);
```

Fonte: Autoria própria.

5.2.11 Adicionando um bloco Mux

A funcionalidade de um bloco Mux é combinar os sinais em suas portas de entrada em um único sinal, em vetor coluna, transmitido pela porta de saída (MATHWORKS, 2016a). No BDC, um bloco Mux é definido pela classe MuxBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Mux também apresenta a seguinte propriedade:

- **numberInputs:** Este valor indica a quantidade de portas de entrada do bloco, deve ser um escalar inteiro maior ou igual à 2.

As características do bloco Mux são listadas abaixo:

- Possui o número de portas de entrada associado a propriedade numberInputs;
- Possui uma porta de saída;
- As portas de entrada são de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- A dimensão do sinal transmitido na porta de saída equivale a soma das dimensões de todos os sinais nas portas de entrada do bloco Mux.

Para adicionar um bloco Mux basta executar o método addMux da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de MuxBlock. O Algoritmo 7 exemplifica este processo.

Algoritmo 7 – Adicionando um bloco Mux.

```
1 addMux(diagram,name,numberInputs);
```

Fonte: Autoria própria.

5.2.12 Adicionando um bloco Demux

A funcionalidade de um bloco Demux é separar igualmente o sinal, em vetor coluna, na porta de entrada através das portas de saída (MATHWORKS, 2016a). No BDC, um bloco Demux é definido pela classe DemuxBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Demux também apresenta a seguinte propriedade:

- **numberOutputs:** Este valor indica a quantidade de portas de saída do bloco, deve ser um escalar inteiro maior ou igual à 2.

As características do bloco Demux são listadas abaixo:

- Possui apenas uma porta de entrada;
- Possui o número de portas de saída associado à propriedade numberOutputs;
- A porta de entrada é de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- O número de sinais na porta de entrada deve ser múltiplo do valor associado à propriedade numberOutputs, para ser possível uma divisão igual.

Para adicionar um bloco Demux basta executar o método addDemux da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de DemuxBlock. O Algoritmo 8 exemplifica este processo.

Algoritmo 8 – Adicionando um bloco Demux.

```
1 addDemux(diagram,name,numberOutputs);
```

Fonte: Autoria própria.

5.2.13 Adicionando um bloco Bias

A funcionalidade de um bloco Bias é somar um valor de *offset* ao sinal em sua porta de entrada (MATHWORKS, 2016a). No BDC, um bloco Bias é definido pela classe BiasBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Bias também apresenta a seguinte propriedade:

- **bias:** Este é o valor adicionado ao sinal de entrada do bloco, deve ser um escalar ou vetor coluna de valores numéricos.

As características do bloco Bias são listadas abaixo:

- Possui uma porta de entrada;
- Possui uma porta de saída;
- A porta de entrada é de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- A dimensão do sinal na porta de entrada deve corresponder a dimensão da propriedade bias, resultando em um sinal de saída com mesma dimensão.

Para adicionar um bloco Bias basta executar o método addBias da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a

propriedade name de BlockClass, e na sequência as propriedades específicas de BiasBlock. O Algoritmo 9 exemplifica este processo.

Algoritmo 9 – Adicionando um bloco Bias.

```
1 addBias(diagram,name,bias);
```

Fonte: Autoria própria.

5.2.14 Adicionando um bloco Gain

A funcionalidade de um bloco Gain é multiplicar o sinal em sua porta de entrada por um valor constante (MATHWORKS, 2016a). No BDC, um bloco Gain é definido pela classe GainBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Gain também apresenta as seguintes propriedades:

- **gain:** Este é o valor que multiplica o sinal de entrada do bloco, deve ser uma matriz de qualquer dimensão e apenas com valores numéricos;
- **multiplication:** Este é o método de multiplicação do sinal de entrada do bloco. Caso o método seja 'Element-wise(K.*u)', o sinal de entrada é multiplicado elemento a elemento. Caso o método seja 'Matrix(K*u) (u vector)', o valor da propriedade gain multiplica o sinal de entrada pela esquerda.

As características do bloco Gain são listadas abaixo:

- Possui uma porta de entrada;
- Possui uma porta de saída;
- A porta de entrada é de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- No método de multiplicação 'Element-wise(K.*u)', a dimensão do sinal de entrada e da propriedade gain devem ser iguais, resultando em um sinal de saída com a mesma dimensão;
- No método de multiplicação 'Matrix(K*u) (u vector)' o número de colunas da matriz associada à propriedade gain deve ser igual a dimensão do sinal de entrada. A dimensão do sinal de saída é igual ao número de linhas da matriz associada à propriedade gain.

Para adicionar um bloco Gain basta executar o método addGain da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de GainBlock. O Algoritmo 10 exemplifica este processo.

Algoritmo 10 – Adicionando um bloco Gain.

```
1 addGain(diagram,name,gain,multiplication);
```

Fonte: Autoria própria.

5.2.15 Adicionando um bloco Discrete Derivative

A funcionalidade de um bloco Discrete Derivative é realizar a derivada discreta do sinal em sua porta de entrada. Considerando o passo de tempo atual t_n, a entrada atual do bloco $\boldsymbol{u}(t_n)$, a entrada anterior do bloco $\boldsymbol{u}(t_{n-1})$, a saída atual do bloco $\boldsymbol{y}(t_n)$ e o tempo de amostragem fixado para o diagrama T_s. A saída do bloco Discrete Derivative, em cada instante de amostragem, é dada pela Equação (80) (MATHWORKS, 2016a).

$$\boldsymbol{y}(t_n) = \frac{(\boldsymbol{u}(t_n) - \boldsymbol{u}(t_{n-1}))}{T_s} \tag{80}$$

No BDC, um bloco Discrete Derivative é definido pela classe DiscreteDerivativeBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Discrete Derivative também apresenta a seguinte propriedade:

- **initialCondition:** Este é o valor que inicializa a entrada anterior do bloco, deve ser um escalar ou vetor coluna de valores numéricos.

As características do bloco Discrete Derivative são listadas abaixo:

- Possui uma porta de entrada;
- Possui uma porta de saída;
- A porta de entrada é de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- A dimensão do sinal na porta de entrada deve corresponder a dimensão da propriedade initialCondition, resultando em um sinal de saída com mesma dimensão.

Para adicionar um bloco Discrete Derivative basta executar o método addDiscreteDerivative da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de DiscreteDerivativeBlock. O Algoritmo 11 exemplifica este processo.

Algoritmo 11 – Adicionando um bloco Discrete Derivative.

```
1 addDiscreteDerivative(diagram,name,initialCondition);
```

Fonte: Autoria própria.

5.2.16 Adicionando um bloco Discrete-Time Integrator

A funcionalidade de um bloco Discrete-Time Integrator é realizar a integração em tempo discreto do sinal em sua porta de entrada. Considerando o passo de tempo atual t_n, a entrada atual do bloco $\boldsymbol{u}(t_n)$, a saída atual do bloco $\boldsymbol{y}(t_n)$, a saída anterior do bloco $\boldsymbol{y}(t_{n-1})$ e o tempo de amostragem fixado para o diagrama T_s. A saída do bloco Discrete-Time Integrator, em cada instante de amostragem, é dada pelo método Backward Euler na Equação (81) (MATHWORKS, 2016a).

$$\boldsymbol{y}(t_n) = \boldsymbol{y}(t_{n-1}) + \boldsymbol{u}(t_n)T_s \tag{81}$$

No BDC, um bloco Discrete-Time Integrator é definido pela classe DiscreteTimeIntegratorBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Discrete-Time Integrator também apresenta as seguintes propriedades:

- **integrationMethod:** Este propriedade indica o método de integração utilizado. Para a versão atual do BDC o único método disponível é 'Backward Euler';
- **initialCondition:** Este é o valor que inicializa a saída anterior do bloco, deve ser um escalar ou vetor coluna de valores numéricos.

As características do bloco Discrete-Time Integrator são listadas abaixo:

- Possui uma porta de entrada;
- Possui uma porta de saída;
- A porta de entrada é de passagem direta, neste caso associa-se o valor 8 à propriedade priority herdada de BlockClass;
- A dimensão do sinal na porta de entrada deve corresponder a dimensão da propriedade initialCondition, resultando em um sinal de saída com mesma dimensão.

Para adicionar um bloco Discrete-Time Integrator basta executar o método addDiscreteTimeIntegrator da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de DiscreteTimeIntegratorBlock. O Algoritmo 12 exemplifica este processo.

Algoritmo 12 – Adicionando um bloco Discrete-Time Integrator.

```
addDiscreteTimeIntegrator(diagram,name,integrationMethod,initialCondition);
```

Fonte: Autoria própria.

5.2.17 Adicionando um bloco Unit Delay

A funcionalidade de um bloco Unit Delay é atrasar o sinal em sua porta de entrada por um instante de amostragem (MATHWORKS, 2016a). No BDC, um bloco Unit Delay é definido pela classe UnitDelayBlock, que herda a classe BlockClass. Além de todas as propriedades de BlockClass, o bloco Unit Delay também apresenta a seguinte propriedade:

- **initialCondition:** Este é o valor que inicializa a saída do bloco no primeiro instante de amostragem, deve ser um escalar ou vetor coluna de valores numéricos.

As características do bloco Unit Delay são listadas abaixo:

- Possui uma porta de entrada;
- Possui uma porta de saída;
- A porta de entrada não é de passagem direta, neste caso associa-se o valor 9 à propriedade priority herdada de BlockClass;
- A dimensão do sinal na porta de entrada deve corresponder a dimensão da propriedade initialCondition, resultando em um sinal de saída com mesma dimensão.

Para adicionar um bloco Unit Delay basta executar o método addUnitDelay da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco, a propriedade name de BlockClass, e na sequência as propriedades específicas de UnitDelayBlock. O Algoritmo 13 exemplifica este processo.

Algoritmo 13 – Adicionando um bloco Unit Delay.

```
addUnitDelay(diagram,name,initialCondition);
```

Fonte: Autoria própria.

5.2.18 Adicionando um bloco Helicopter 3DOF

A funcionalidade de um bloco Helicopter 3DOF é permitir o controle do modelo linear e da planta do Helicóptero com 3 GDL. A configuração interna do bloco Helicopter 3DOF é a mesma do bloco "Helicóptero com 3 GDL", apresentado pela Seção 4.9 na Figura 40. A

diferença, porém, é que o bloco Helicopter 3DOF não suporta a cossimulação com o Adams e o controle da planta, por meio do BDC, é realizado no NUCLEO-F767ZI. No BDC, um bloco Helicopter 3DOF é definido pela classe Helicopter3DOFBlock, que herda a classe BlockClass. As características do bloco Helicopter 3DOF são listadas abaixo:

- Possui uma porta de entrada;
- Possui uma porta de saída;
- A porta de entrada não é de passagem direta, neste caso associa-se o valor 9 à propriedade priority herdada de BlockClass;
- A dimensão do sinal na porta de entrada corresponde a dimensão do vetor de controle $\boldsymbol{u}$ e a dimensão do sinal de saída corresponde a dimensão do vetor de saída $\boldsymbol{y}$, tudo conforme definido na modelagem matemática apresentada na Seção 4.6.

Para adicionar um bloco Helicopter 3DOF basta executar o método addHelicopter3DOF da classe Diagram. Os parâmetros deste método são o objeto de Diagram, no qual deseja-se adicionar o bloco e a propriedade name de BlockClass. O Algoritmo 14 exemplifica este processo.

Algoritmo 14 – Adicionando um bloco Helicopter 3DOF.

```
1 addHelicopter3DOF(diagram,name);
```

Fonte: Autoria própria.

5.2.19 Conectando blocos

Para realizar a conexão entre blocos basta executar o método linkBlocks da classe Diagram. Este método cópia os objetos de OutputPortClass para a propriedade source dos objetos de InputPortClass. Os parâmetros do método linkBlocks são o objeto de Diagram, no qual deseja-se realizar a conexão dos blocos, a propriedade name do bloco de origem, a propriedade port da porta de saída do bloco de origem, a propriedade name do bloco de destino e a propriedade port da porta de entrada do bloco de destino. O Algoritmo 15 exemplifica este processo.

Algoritmo 15 – Conectando blocos.

```
1 linkBlocks(diagram,nameOrigem,portOrigem,nameDestino,portDestino);
```

Fonte: Autoria própria.

5.2.20 Construindo um controlador

Para construir o controlador definido por um diagrama de blocos, basta executar a função bdcBuild. O parâmetro deste método é o objeto de Diagram. O Algoritmo 16 exemplifica este processo.

Algoritmo 16 – Construindo um controlador definido por um diagrama de blocos.

```
1 bdcBuild(diagram);
```

Fonte: Autoria própria.

5.2.21 Executando um controlador

O processo de contrução define a chamada de execução do controlador como bdcRun. Ao chamar a execução deste *script*, o BDC executa apenas o último controlador construído pela função bdcBuild. O Algoritmo 17 exemplifica este processo.

Algoritmo 17 – Executando um controlador.

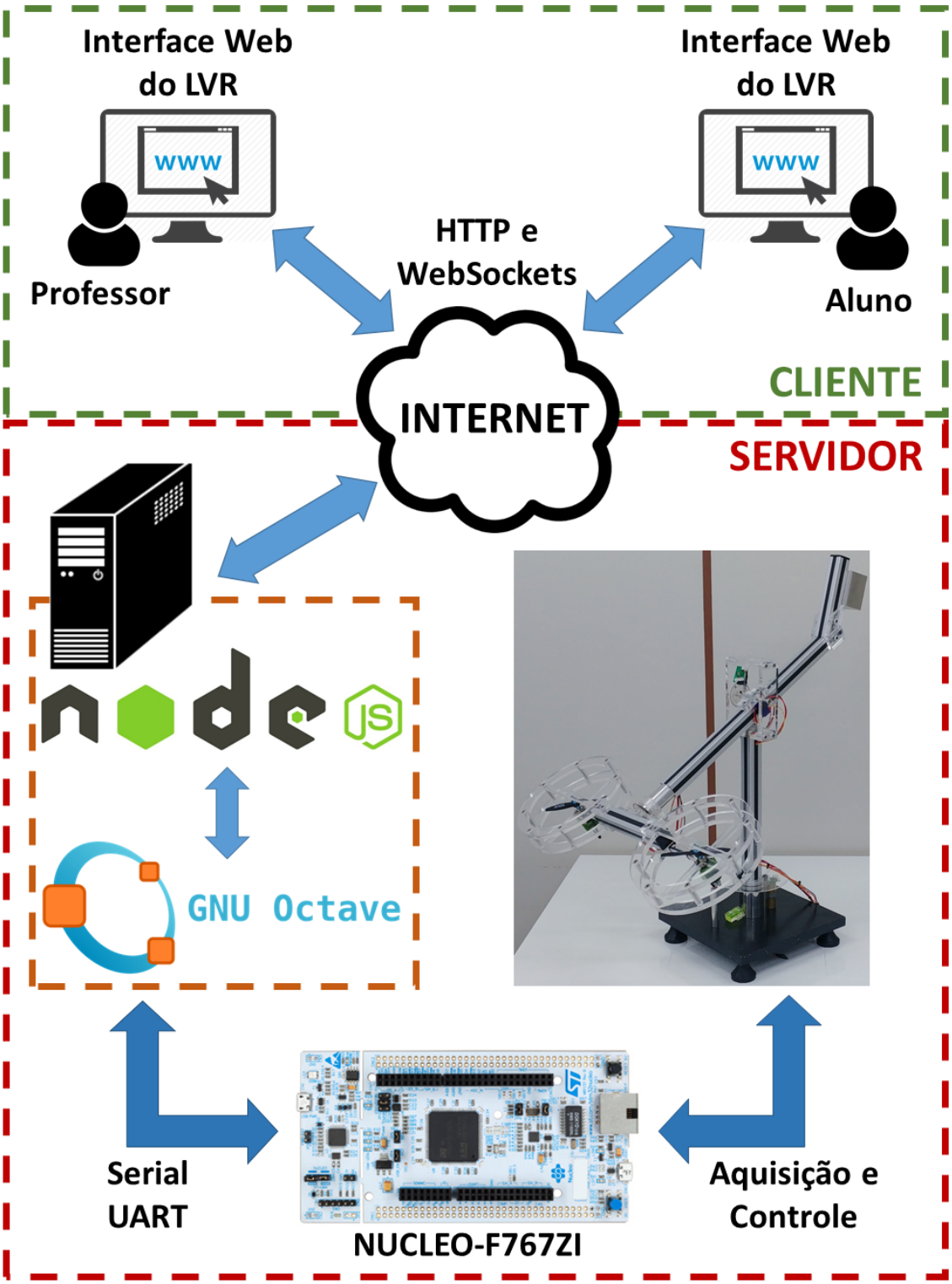
```
1 bdcRun;
```

Fonte: Autoria própria.

5.3 APLICAÇÃO WEB DO LABORATÓRIO VIRTUAL E REMOTO

Uma vez que os experimentos de controle virtual (simulação linear) e remoto são desenvolvidos exclusivamente por *scripts* na linguagem MATLAB/Octave, a aplicação Web do LVR foi desenvolvida para funcionar como uma IDE remota para o Octave CLI, em execução no lado do servidor. Deste modo, a arquitetura cliente-servidor da aplicação Web foi combinada com o sistema de aquisição e controle do LVR, representado na Seção 5.2 pela Figura 48, permitindo ao usuário projetar e implementar diferentes técnicas de controle, executadas exclusivamente do lado do servidor. Esta combinação de arquiteturas está representada na Figura 55.

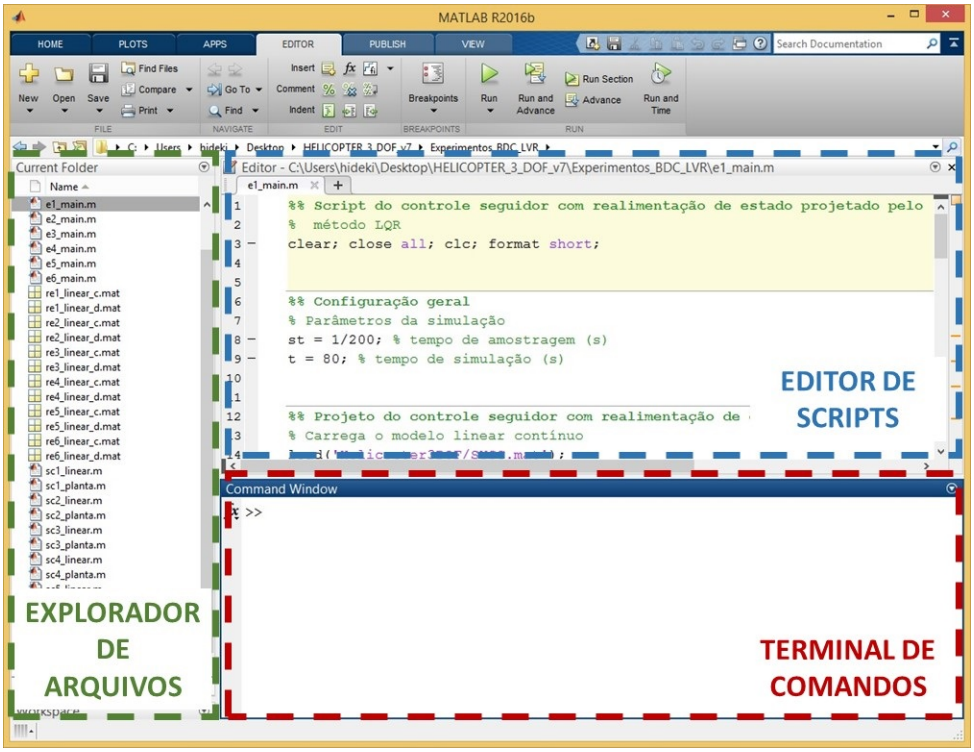
Figura 55 – Visão geral da arquitetura do LVR.



Fonte: Autoria própria.

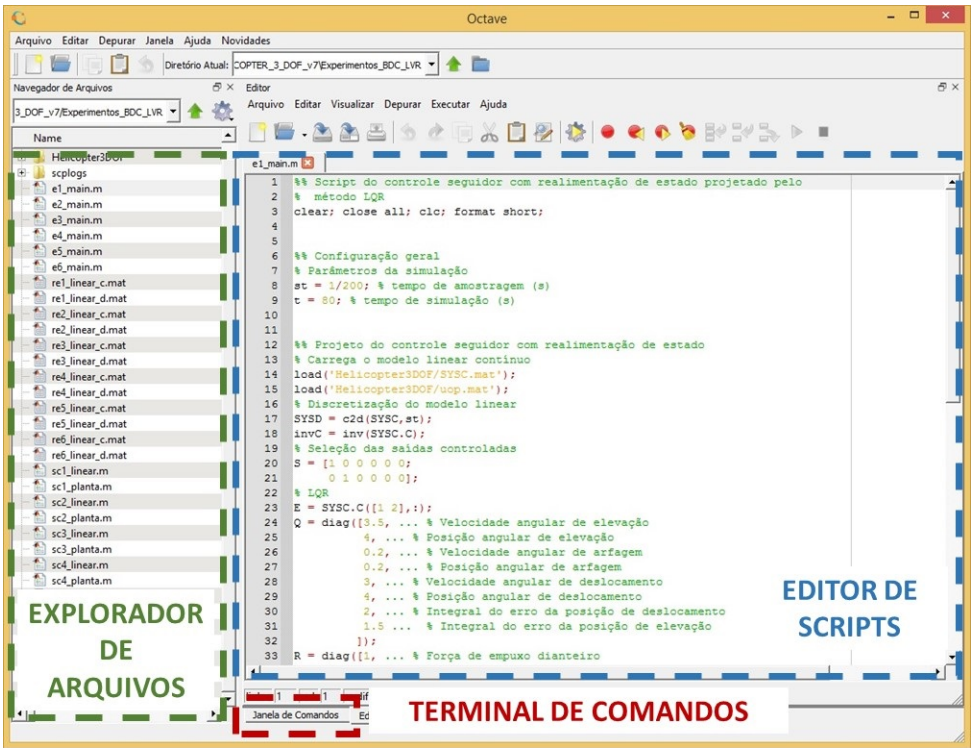
A interface da aplicação Web é construída com inspiração nos recursos de exploração de arquivos, editor de *scripts* e terminal de comandos, disponíveis nas IDEs *desktop* do MATLAB e do próprio Octave representadas, respectivamente, nas Figuras 56 e 57.

Figura 56 – Recursos da IDE do MATLAB.



Fonte: Autoria própria.

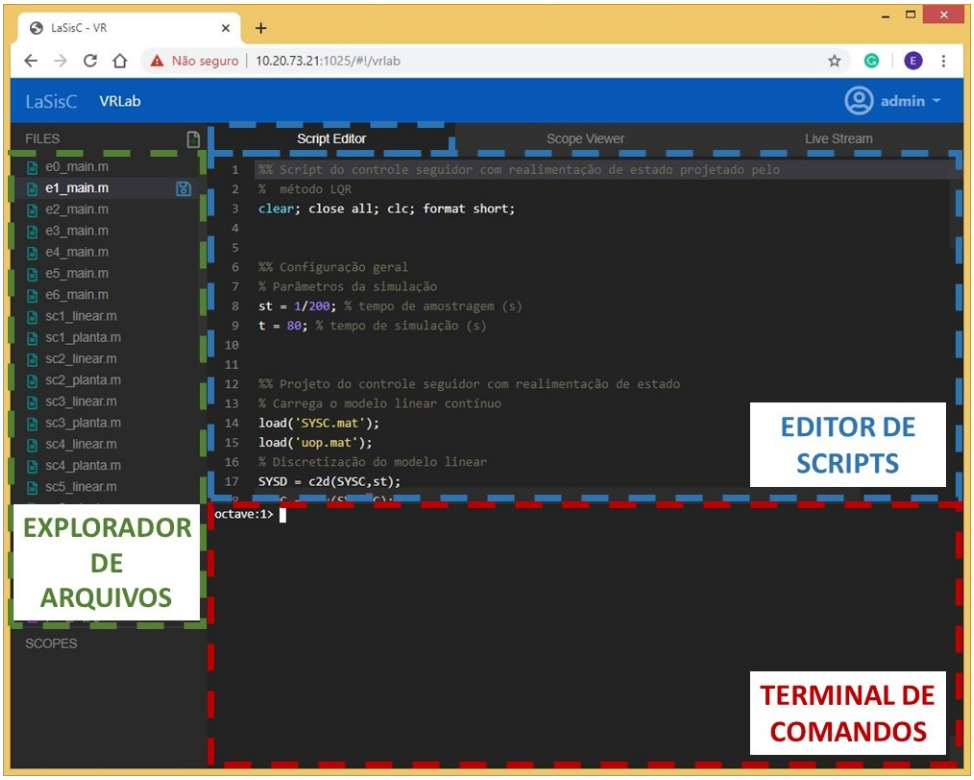
Figura 57 – Recursos da IDE do Octave.



Fonte: Autoria própria.

Tais recursos da interface Web são programados por meio da linguagem JavaScript, utilizando a tecnologia AngularJS. A aparência da interface Web é programada por meio das tecnologias Bootstrap e CSS. O *framework* AngularJS Material é utilizado na implementação de caixas de mensagem, que aparecem na interface Web. A Figura 58 apresenta a interface Web desenvolvida para o LVR, com os mesmos recursos destacados nas Figuras 56 e 57. Na sequência está descrito o funcionamento dos recursos do LVR na interface Web.

Figura 58 – Recursos da IDE do LVR.



Fonte: Autoria própria.

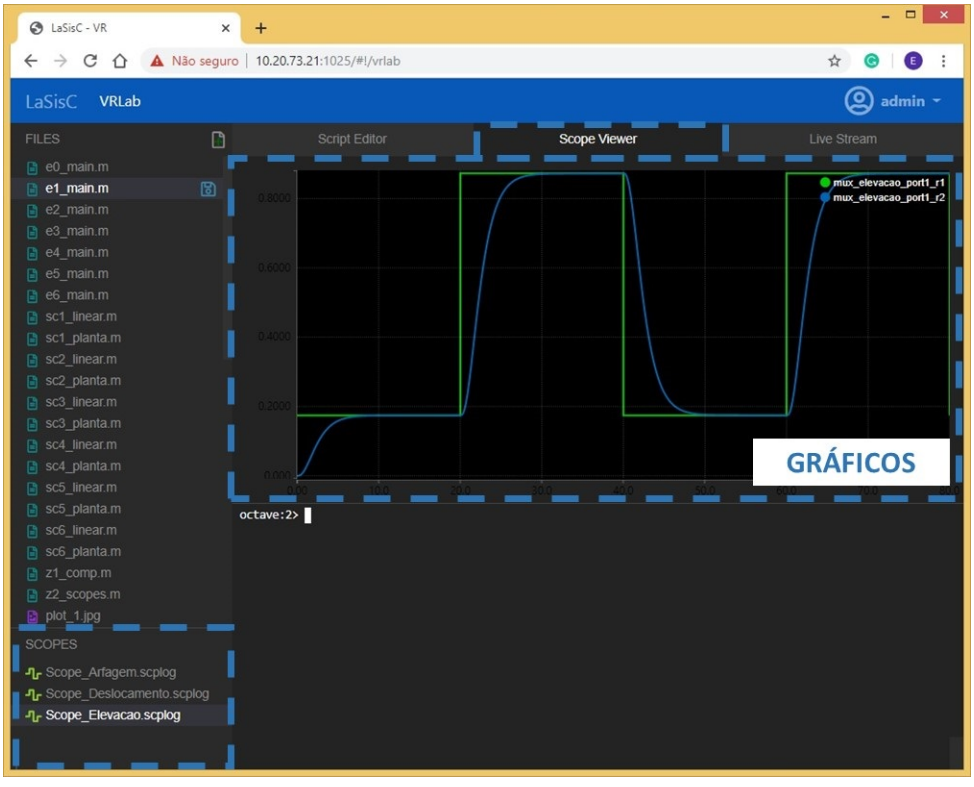
- **Interface do explorador de arquivos:** Este recurso permite ao usuário criar, salvar alterações, renomear, excluir e solicitar a visualização ou *download* de arquivos. Os tipos de arquivo suportados pelo explorador são *scripts* da linguagem MATLAB/Octave (com extensão ".m"), imagens (com extensão ".jpg" e ".png") e dados exportados para leitura em MATLAB ou Octave (com extensão ".mat"). A capacidade de criar e salvar alterações, por meio do explorador de arquivos, é restrita a arquivos com extensão ".m". A solicitação de visualização, por meio do explorador de arquivos, é permitida apenas para arquivos com extensão ".m", ".jpg" e ".png". A renomeação, exclusão e solicitação de *download* se aplica a todos as extensões suportadas pelo explorador de arquivos;
- **Interface do editor de scripts:** Este recurso permite ao usuário programar *scripts* em linguagem MATLAB/Octave. O desenvolvimento deste recurso depende da integração da aplicação em AngularJS com a tecnologia CodeMirror, que apresenta suporte à sintaxe da linguagem MATLAB/Octave;

- **Interface do terminal de comandos:** Este recurso permite ao usuário executar comandos na CLI do Octave, em execução no servidor. Deste modo, o usuário também pode solicitar a execução dos *scripts* construídos em linguagem MATLAB/Octave. O desenvolvimento deste recurso depende da integração da aplicação em AngularJS com a tecnologia Xterm.js.

Além destes recursos, a interface do LVR permite ao usuário monitorar os controles virtual e remoto, por meio das seguintes capacidades:

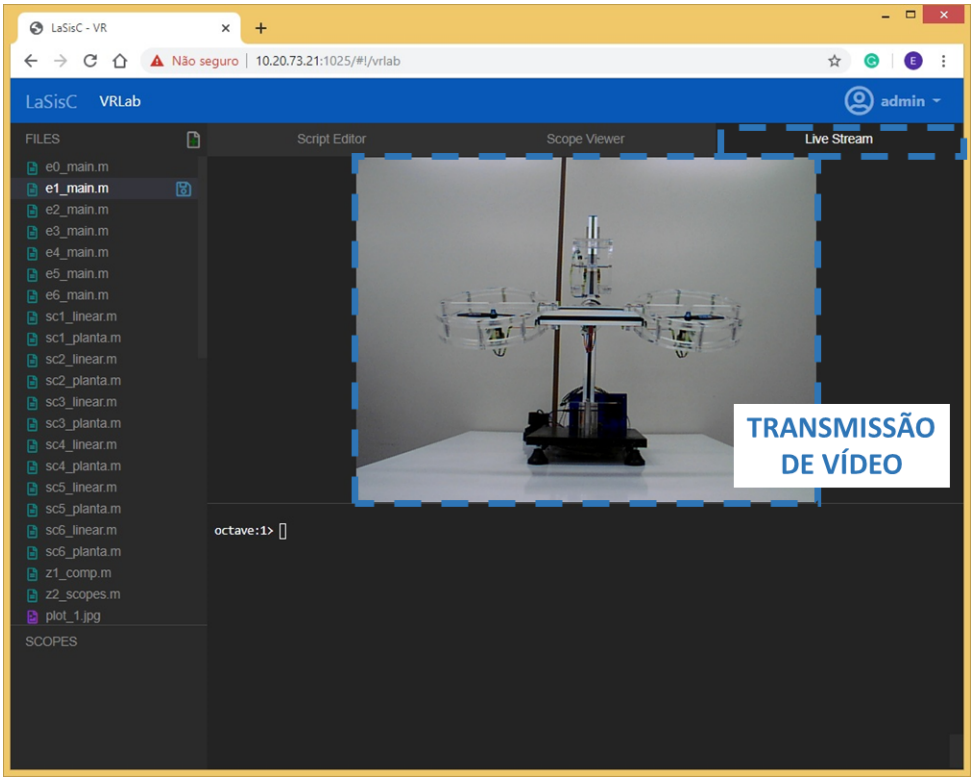
- **Interface de visualização de gráficos construídos em tempo de execução:** Este recurso permite que o usuário visualize, por meio de gráficos, a evolução dos sinais de modo simultâneo a execução dos experimentos de controle virtual e remoto. O desenvolvimento deste recurso depende da integração da aplicação em AngularJS com as tecnologias D3.js e Rickshaw. A Figura 59 apresenta a interface do LVR com este recurso;
- **Interface de visualização de vídeo transmitido em tempo de execução:** Este recurso permite que o usuário acompanhe a execução do controle no sistema real, por meio de *streaming* de vídeo. A Figura 60 apresenta a interface do LVR com este recurso.

Figura 59 – Recurso de gráficos atualizados em tempo de execução na interface do LVR.



Fonte: Autoria própria.

Figura 60 – Recurso de vídeo atualizado em tempo de execução na interface do LVR.



Fonte: Autoria própria.

O funcionamento dos recursos na interface Web dependem da troca de informações com a aplicação do servidor, desenvolvida em linguagem JavaScript e executada no ambiente do Node.js instalado no sistema operacional Ubuntu 18.04.3 LTS. Esta troca de informações é estruturada por meio do padrão JSON e realizada por meio do HTTP e WebSockets. Na sequência está descrito o funcionamento dos recursos do LVR, no servidor Web.

- **Serviço do explorador de arquivos:** Os arquivos gerenciados por este recurso são mantidos em um diretório do servidor. Desta forma, a interface do LVR solicita/envia para a aplicação do Node.js as informações de nome e conteúdo dos arquivos, dependendo da funcionalidade sendo executada pelo explorador de arquivos (criar, salvar, excluir, solicitar visualização ou *download*). Caso alguma operação realizada no servidor, por algum programa em linguagem MATLAB/Octave, crie ou exclua arquivos, a aplicação do Node.js envia as informações para a interface do LVR atualizar a exibição do explorador de arquivos. O gerenciamento dos arquivos por meio do Node.js depende dos módulos chokidar e glob. O módulo chokidar é utilizado para detectar quando um arquivo é criado ou excluído. O módulo glob é utilizado para obter o nome dos arquivos dentro do diretório de gerenciamento;
- **Serviço do editor de scripts:** A edição de *scripts* é realizada exclusivamente no Code-Mirror integrado a aplicação em Angular.js da interface Web, enquanto o gerenciamento

do arquivo ".m" é realizado exclusivamente pelo recurso de explorador de arquivos;

- **Serviço do terminal de comandos:** O desenvolvimento deste recurso utiliza o módulo node-pty para iniciar a execução de um processo do Octave CLI. Deste modo, a aplicação do Node.js recebe um comando, enviado pelo Xterm.js, integrado ao Angular.js, e o executa na CLI do Octave. A aplicação do Node.js captura a resposta do comando executado pela CLI do Octave e envia para o Xterm.js, integrado ao Angular.js, para exibição na interface Web;
- **Serviço de visualização de gráficos construídos em tempo de execução:** Todos os sinais exibidos por este recurso são calculados ou obtidos pelo BDC. Deste modo, a aplicação do Node.js monitora um diretório do servidor, onde o bloco Scope armazena os sinais obtidos do controlador. Os dados dos sinais são estruturados em JSON e armazenados em um arquivo com a extensão arbitrária ".scplog". Cada arquivo representa, individualmente, o gráfico de um bloco Scope. Por meio do módulo chokidar, a aplicação em Node.js pode determinar quando um arquivo ".scplog" é criado, removido ou atualizado. Ao detectar a criação e remoção, a aplicação do Node.js envia para a interface Web quais os Scopes estão disponíveis para exibição. Ao detectar a atualização, a aplicação em Node.js envia para a interface Web os sinais do controlador para exibição em tempo de execução;
- **Serviço de visualização de vídeo transmitido em tempo de execução:** A aplicação do Node.js, por meio do módulo opencv4nodejs, captura os *frames* da câmera no formato JPEG e envia para a interface Web. Esta é responsável pela exibição sequencial dos *frames*.

6 ANÁLISE DOS RESULTADOS EXPERIMENTAIS

Neste capítulo são apresentados quatro experimentos com o objetivo de controlar os movimentos de deslocamento e elevação do Helicóptero com 3 GDL. Cada experimento foi executado em laboratório tradicional e também em LVR, para isto foram considerados os conceitos apresentados, respectivamente, nas Seções 4.9 e 5.2. A análise e comparação dos resultados experimentais permitiu avaliar as características deste trabalho.

Nos Experimentos 1 e 2, utilizou-se a técnica de controle seguidor com realimentação de estado, apresentado na Seção 3.3. Nos Experimentos 3 e 4, o controle seguidor com realimentação de estado foi combinado ao observador de Luenberger, apresentado na Seção 3.6. O projeto do controlador nos Experimento 1 e 3 foi realizado pelo método LQR, apresentado na Seção 3.5. Nos Experimento 2 e 4, o projeto do controlador foi realizado pelo método de atribuição de autoestrutra completa, apresentado na Seção 3.4. A alocação dos polos do observador de Luenberger, nos Experimentos 3 e 4, foi realizada pela função "place" disponível no MATLAB/Octave.

No projeto dos controladores discretos, pelos métodos LQR e atribuição de autoestrutura completa, assim como no projeto do observador de Luenberger, pela função "place", foi utilizado o modelo discretizado do Helicóptero com 3 GDL. O modelo contínuo foi determinado na Seção 4.6 e apresentado nas Equações (71), (72), (73) e (74), a discretização foi realizada pela função "c2d" do MATLAB, por meio do método padrão *zero-order hold*. O tempo de amostragem utilizado foi 0,005 segundos. O tempo total da execução de cada experimento foi de 80 segundos. Para possibilitar a comparação dos resultados, todos os controladores atuaram sobre o mesmo vetor de referência $\boldsymbol{r}$:

- **Referência do deslocamento (r_1):** É um sinal, em radianos, com forma de onda quadrada, *offset* equivalente a 25°, amplitude equivalente a -25°, período de 40 segundos e razão cíclica de 50%;
- **Referência da elevação (r_2):** É um sinal, em radianos, com forma de onda quadrada, *offset* equivalente a 30°, amplitude equivalente a -20°, período de 40 segundos e razão cíclica de 50%.

Os sinais de interesse, resultantes da execução dos experimentos, são o vetor de referência $\boldsymbol{r}$, e os vetores de controle $\boldsymbol{u}$ e saída $\boldsymbol{y}$ do Helicóptero com 3 GDL, conforme definido pela Seção 4.6. A comparação destes resultados foi feita por meio de gráficos e também do critério NRMSE (*Normalized Root Mean Square Error* ou Raiz do Erro Quadrático Médio Normalizado, em português). Tal critério foi utilizado para avaliar o quanto um sinal está ajustado com relação a outro. Em trabalhos como Chaves *et al.* (2017), por exemplo, o critério do NRMSE é utilizado para avaliar a proximidade das respostas de controle real e simulada,

como forma de validar o processo de identificação de parâmetros de um motor de corrente contínua.

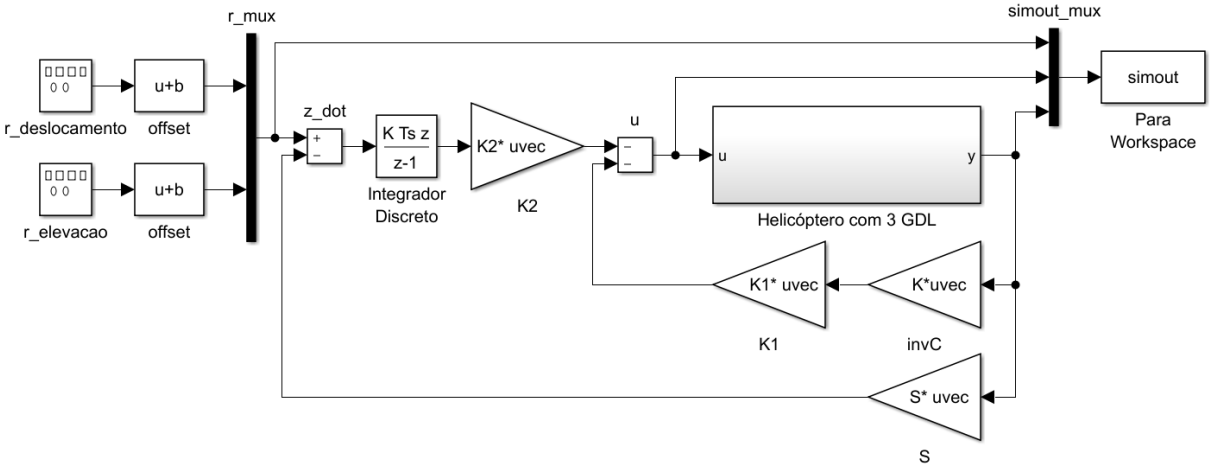
O NRMSE é implementado pela função "goodnessOfFit" do MATLAB. Considerando que $\|$ indica a norma de um vetor, $xref$ indica os dados de referência e x os dados comparados, o NRMSE é calculado pela Equação (82). O resultado do NRMSE para comparação de dois sinais iguais é 1, deste modo, deseja-se que a comparação dos resultados obtidos esteja próxima deste valor ideal (MATHWORKS, 2019a).

$$NRMSE = 1 - \frac{\|xref - x\|}{\|xref - média(xref)\|} \tag{82}$$

6.1 CONSTRUÇÃO DOS EXPERIMENTOS 1 E 2

A Figura 61 apresenta o controle seguidor com realimentação de estado definido para o Experimento 1. Este foi desenvolvido para o laboratório tradicional por meio do Simulink. Na definição do controlador para o Experimento 2, basta alterar os sinais de "$-$" para "$+$" no bloco Sum com nome "u".

Figura 61 – Controlador do Experimento 1 definido em um diagrama de blocos do Simulink.



Fonte: Autoria própria.

O Algoritmo 18 apresenta o controlador do Experimento 1, equivalente ao apresentado na Figura 61, desenvolvido para o LVR utilizando o pacote BDC. Na definição do controlador para o Experimento 2, basta alterar na linha de código 12 o parâmetro '$--$' para '$++$'.

Algoritmo 18 – Controlador do Experimento 1 definido por meio do BDC.

```
% Para o controle do modelo linear executar apenas a linha 3
diagram = Diagram('diagram',st,t,'discrete','simulation');
% Para o controle da sistema real executar apenas a linha 5
diagram = Diagram('diagram',st,t,'discrete','external','NUCLEO_F767ZI');
% Adicionando os blocos
addSignalGenerator(diagram,'r_deslocamento','square',deg2rad(25),deg2rad(-25),40,50);
addSignalGenerator(diagram,'r_elevacao','square',deg2rad(30),deg2rad(-20),40,50);
addMux(diagram,'r_mux',2);
addSum(diagram,'z_dot','+-');
addDiscreteTimeIntegrator(diagram,'integrador_discreto','Backward Euler',zeros(2,1));
addGain(diagram,'K2',K2,'Matrix(K*u) (u vector)');
addSum(diagram,'u','--');
addHelicopter3DOF(diagram,'Helicopter3GDL');
addGain(diagram,'invC',invC,'Matrix(K*u) (u vector)');
addGain(diagram,'K1',K1,'Matrix(K*u) (u vector)');
addGain(diagram,'S',S,'Matrix(K*u) (u vector)');
addMux(diagram,'simout_mux',3);
addToWorkspace(diagram,'para_workspace','simout','Structure with time');
% Realizando as conexões entre os blocos
linkBlocks(diagram,'r_deslocamento',1,'r_mux',1);
linkBlocks(diagram,'r_elevacao',1,'r_mux',2);
linkBlocks(diagram,'r_mux',1,'z_dot',1);
linkBlocks(diagram,'z_dot',1,'integrador_discreto',1);
linkBlocks(diagram,'integrador_discreto',1,'K2',1);
linkBlocks(diagram,'K2',1,'u',1);
linkBlocks(diagram,'u',1,'Helicopter3GDL',1);
linkBlocks(diagram,'Helicopter3GDL',1,'invC',1);
linkBlocks(diagram,'invC',1,'K1',1);
linkBlocks(diagram,'K1',1,'u',2);
linkBlocks(diagram,'Helicopter3GDL',1,'S',1);
linkBlocks(diagram,'S',1,'z_dot',2);
linkBlocks(diagram,'r_mux',1,'simout_mux',1);
linkBlocks(diagram,'u',1,'simout_mux',2);
linkBlocks(diagram,'Helicopter3GDL',1',simout_mux',3);
linkBlocks(diagram,'simout_mux',1,'para_workspace',1);
```

Fonte: Autoria própria.

O bloco Gain com a matriz $\boldsymbol{S}$, presente em todas estas definições do controlador, foi utilizado para selecionar as variáveis de saída controladas, ou seja, as posições angulares de deslocamento e elevação. A matriz $\boldsymbol{S}$ é apresentada na Equação (83).

$$\boldsymbol{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{83}$$

O projeto das matrizes de ganhos discretos $\boldsymbol{K_1}$ e $\boldsymbol{K_2}$ foi realizado por *script*, em linguagem MATLAB/Octave. Para o Experimento 1, as matrizes de ganhos discretos $\boldsymbol{K_1}$ e $\boldsymbol{K_2}$ são apresentadas nas Equações (84) e (85). Estas foram determinadas empiricamente pelo método LQR, por meio das matrizes de ponderação dos estados $\boldsymbol{Q}$ e ponderação do controle

$\boldsymbol{R}$, respectivamente, nas Equações (86) e (87).

$$\boldsymbol{K_1} = \begin{bmatrix} 3{,}5779 & 2{,}7339 & -0{,}8528 & -2{,}2136 & -2{,}1690 & -2{,}6376 \\ -3{,}5779 & -2{,}7339 & 0{,}8528 & 2{,}2136 & -2{,}1690 & -2{,}6376 \end{bmatrix} \tag{84}$$

$$\boldsymbol{K_2} = \begin{bmatrix} -0{,}8655 & -1{,}4076 \\ 0{,}8655 & -1{,}4076 \end{bmatrix} \tag{85}$$

$$\boldsymbol{Q} = \begin{bmatrix} 2{,}00 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2{,}65 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0{,}41 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0{,}90 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4{,}20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6{,}20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1{,}55 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4{,}05 \end{bmatrix} \tag{86}$$

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{87}$$

As matrizes de ganhos discretos $\boldsymbol{K_1}$ e $\boldsymbol{K_2}$, para o Experimento 2, são apresentadas nas Equações (88) e (89). Estas foram determinadas empiricamente pelo método de atribuição de autoestrutura completa. Para tanto, o espectro de autovalores de malha fechada, na Equação (90), foi primeiro mapeado para o plano z, em seguida foram determinados os autovetores associados, subdivididos nas Equações (91) e (92).

$$\boldsymbol{K_1} = \begin{bmatrix} -3{,}5754 & -2{,}7327 & 0{,}8545 & 2{,}2102 & 2{,}1673 & 2{,}6357 \\ 3{,}5754 & 2{,}7327 & -0{,}8545 & -2{,}2102 & 2{,}1673 & 2{,}6357 \end{bmatrix} \tag{88}$$

$$\boldsymbol{K_2} = \begin{bmatrix} 0{,}8630 & 1{,}4065 \\ -0{,}8630 & 1{,}4065 \end{bmatrix} \tag{89}$$

$$\begin{aligned} \boldsymbol{\sigma}(\boldsymbol{A} + \boldsymbol{BK}) = [&-1{,}25 + 0{,}12i \quad -1{,}25 - 0{,}12i \quad -1{,}15 + 0{,}90i \quad -1{,}15 - 0{,}90i \quad \cdots \\ &-1{,}20 + 0{,}11i \quad -1{,}20 - 0{,}11i \quad -2{,}00 \quad -1{,}95] \end{aligned} \tag{90}$$

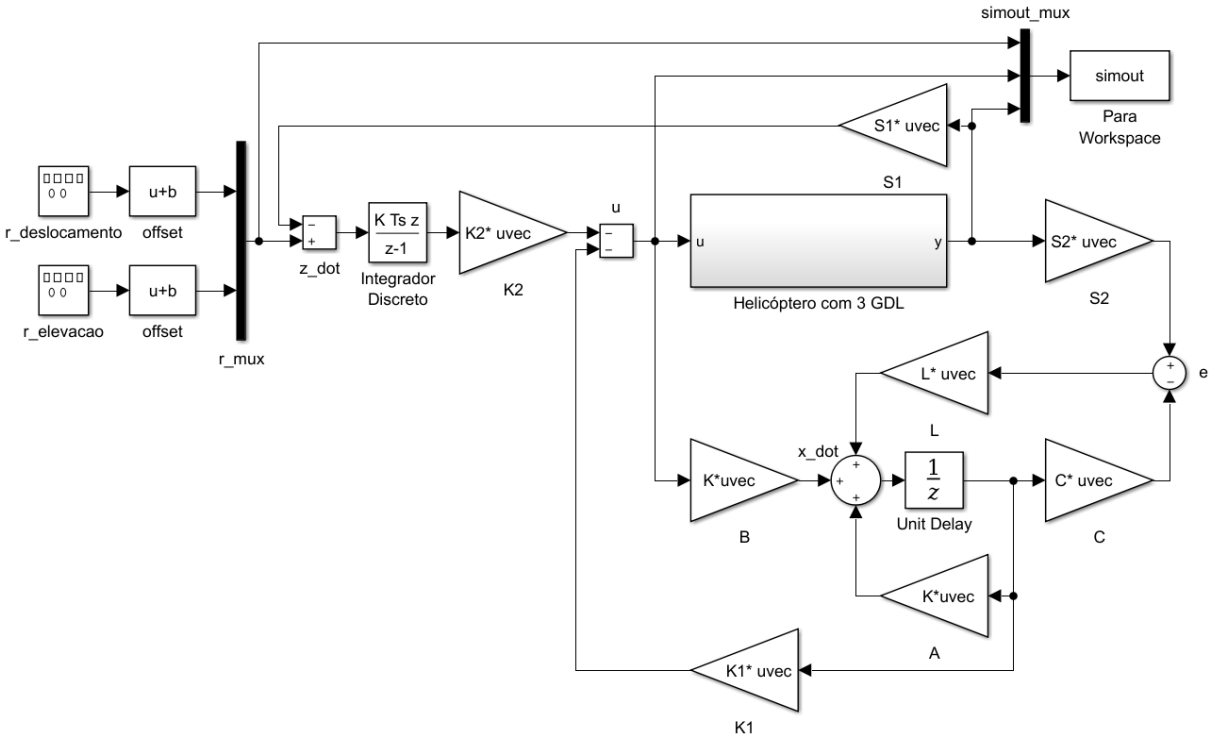
$$\boldsymbol{V} = \begin{bmatrix} 1{,}7658 & 0{,}3198 & 0{,}3643 & 1{,}6228 & 0{,}0000 & -0{,}0000 & 0{,}6783 & 0{,}0000 \\ -1{,}3754 & -0{,}3879 & 0{,}4884 & -1{,}0289 & -0{,}0000 & -0{,}0000 & -0{,}3392 & -0{,}0000 \\ -2{,}7474 & 0{,}0255 & -3{,}4753 & -0{,}1474 & 0{,}0000 & -0{,}0000 & -2{,}5726 & 0{,}0000 \\ 2{,}1798 & 0{,}1889 & 1{,}8119 & 1{,}5462 & -0{,}0000 & -0{,}0000 & 1{,}2863 & -0{,}0000 \\ 0{,}0000 & 0{,}0000 & 0{,}0000 & 0{,}0000 & -1{,}0615 & -0{,}0266 & 0{,}0000 & -0{,}8436 \\ -0{,}0000 & -0{,}0000 & 0{,}0000 & -0{,}0000 & 0{,}8752 & 0{,}1024 & -0{,}0000 & 0{,}4326 \\ -1{,}0607 & -0{,}4122 & 0{,}6976 & -0{,}3487 & -0{,}0000 & -0{,}0000 & -0{,}1696 & -0{,}0000 \\ 0{,}0000 & 0{,}0000 & -0{,}0000 & 0{,}0000 & -0{,}7155 & -0{,}1509 & 0{,}0000 & -0{,}2218 \end{bmatrix} \tag{91}$$

$$\boldsymbol{Q} = \begin{bmatrix} -1 & 0 & -1 & 0 & -1 & 0 & -1 & -1 \\ 1 & 0 & 1 & 0 & -1 & 0 & 1 & -1 \end{bmatrix} \tag{92}$$

6.2 CONSTRUÇÃO DOS EXPERIMENTOS 3 E 4

A Figura 62 apresenta o controle seguidor com realimentação de estado combinado ao observador de Luenberger, definido para o Experimento 3. Este foi desenvolvido para o laboratório tradicional por meio do Simulink. Na definição do controlador para o Experimento 4, basta alterar os sinais de "−" para "+" no bloco Sum com nome "u".

Figura 62 – Controlador do Experimento 3 definido em um diagrama de blocos do Simulink.



Fonte: Autoria própria.

O Algoritmo 19 apresenta o controlador do Experimento 3, equivalente ao apresentado na Figura 62, desenvolvido para o LVR utilizando o pacote BDC. Na definição do controlador para o Experimento 4, basta alterar na linha de código 12 o parâmetro '−−' para '++'.

Algoritmo 19 – Controlador do Experimento 3 definido por meio do BDC.

```
% Para o controle do modelo linear executar apenas a linha 3
diagram = Diagram('diagram',st,t,'discrete','simulation');
% Para o controle da sistema real executar apenas a linha 5
diagram = Diagram('diagram',st,t,'discrete','external','NUCLEO_F767ZI');
% Adicionando os blocos
addSignalGenerator(diagram,'r_deslocamento','square',deg2rad(25),deg2rad(-25),40,50);
addSignalGenerator(diagram,'r_elevacao','square',deg2rad(30),deg2rad(-20),40,50);
addMux(diagram,'r_mux',2);
addSum(diagram,'z_dot','+-');
addDiscreteTimeIntegrator(diagram,'integrador_discreto','Backward Euler',zeros(2,1));
addGain(diagram,'K2',K2,'Matrix(K*u) (u vector)');
addSum(diagram,'u','--');
addHelicopter3DOF(diagram,'Helicopter3GDL');
addGain(diagram,'S1',S1,'Matrix(K*u) (u vector)');
addGain(diagram,'B',SYSD.B,'Matrix(K*u) (u vector)');
addSum(diagram,'x_dot','+++');
addUnitDelay(diagram,'unit_delay',zeros(6,1));
addGain(diagram,'A',SYSD.A,'Matrix(K*u) (u vector)');
addGain(diagram,'K1',K1,'Matrix(K*u) (u vector)');
addGain(diagram,'S2',S2,'Matrix(K*u) (u vector)');
addGain(diagram,'C',C,'Matrix(K*u) (u vector)');
addSum(diagram,'e','+-');
addGain(diagram,'L',L,'Matrix(K*u) (u vector)');
addMux(diagram,'simout_mux',3);
addToWorkspace(diagram,'para_workspace','simout','Structure with time');
% Realizando as conexões entre os blocos
linkBlocks(diagram,'r_deslocamento',1,'r_mux',1);
linkBlocks(diagram,'r_elevacao',1,'r_mux',2);
linkBlocks(diagram,'r_mux',1,'z_dot',1);
linkBlocks(diagram,'z_dot',1,'integrador_discreto',1);
linkBlocks(diagram,'integrador_discreto',1,'K2',1);
linkBlocks(diagram,'K2',1,'u',1);
linkBlocks(diagram,'u',1,'Helicopter3GDL',1);
linkBlocks(diagram,'Helicopter3GDL',1,'S1',1);
linkBlocks(diagram,'S1',1,'z_dot',2);
linkBlocks(diagram,'u',1,'B',1);
linkBlocks(diagram,'B',1,'x_dot',1);
linkBlocks(diagram,'x_dot',1,'unit_delay',1);
linkBlocks(diagram,'unit_delay',1,'A',1);
linkBlocks(diagram,'A',1,'x_dot',2);
linkBlocks(diagram,'unit_delay',1,'K1',1);
linkBlocks(diagram,'K1',1,'u',2);
linkBlocks(diagram,'Helicopter3GDL',1,'S2',1);
linkBlocks(diagram,'unit_delay',1,'C',1);
linkBlocks(diagram,'S2',1,'e',1);
linkBlocks(diagram,'C',1,'e',2);
linkBlocks(diagram,'e',1,'L',1);
linkBlocks(diagram,'L',1,'x_dot',3);
linkBlocks(diagram,'r_mux',1,'simout_mux',1);
linkBlocks(diagram,'u',1,'simout_mux',2);
linkBlocks(diagram,'Helicopter3GDL',1','simout_mux',3);
linkBlocks(diagram,'simout_mux',1,'para_workspace',1);
```

Fonte: Autoria própria.

O bloco Gain com a matriz $\boldsymbol{S1}$, presente em todas definições do controlador, foi utilizado para selecionar as variáveis de saída controladas, ou seja, as posições angulares de deslocamento e elevação. A matriz $\boldsymbol{S1}$ é apresentada na Equação (93).

$$\boldsymbol{S1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{93}$$

O bloco Gain com a matriz $\boldsymbol{S2}$, presente em todas definições do controlador, foi utilizado para selecionar as variáveis de saída comparadas pelo observador de Luenberger, ou seja, as posições angulares dos 3 GDL. A matriz $\boldsymbol{S2}$ é apresentada na Equação (94).

$$\boldsymbol{S2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{94}$$

As matrizes de ganhos $\boldsymbol{K_1}$ e $\boldsymbol{K_2}$, para o Experimento 3, foram as mesmas determinadas nas Equações (84) e (85). Para o Experimento 4, as matrizes de ganhos $\boldsymbol{K_1}$ e $\boldsymbol{K_2}$ foram as mesmas determinadas nas Equações (88) e (89). A matriz do observador discreto $\boldsymbol{L}$, para os Experimentos 3 e 4, é apresentada na Equação (95). Esta foi obtida para os autovalores da matriz de estado $\boldsymbol{A}$, dada na Equação (71), deslocados cinquenta unidades para a esquerda do semiplano complexo. Para determinação da matriz discreta $\boldsymbol{L}$ foi necessário mapear estes polos para o plano z.

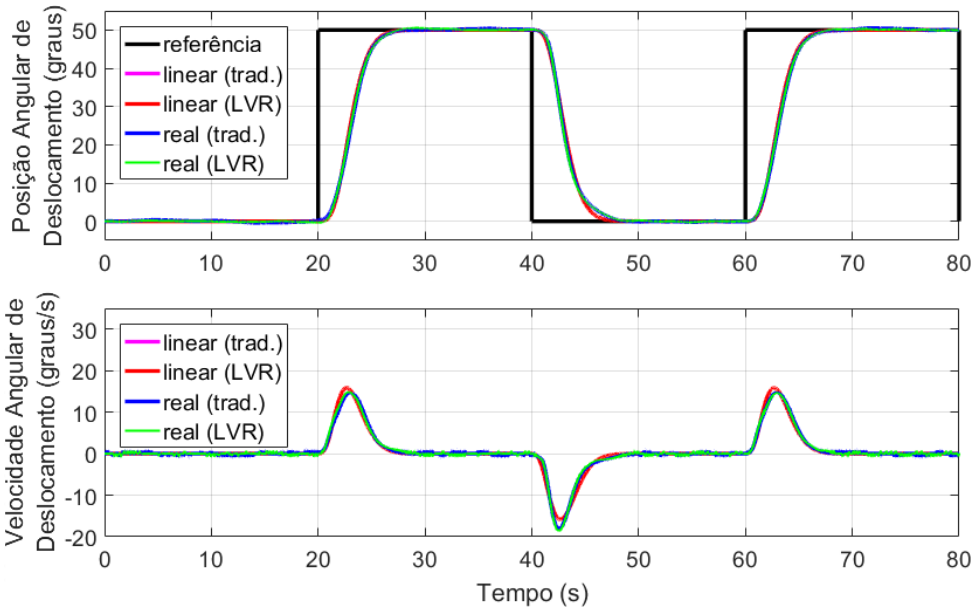
$$\boldsymbol{L} = \begin{bmatrix} 9{,}8421 & 0{,}2202 & -0{,}1588 \\ 0{,}4437 & 0{,}0050 & -0{,}0036 \\ -0{,}0322 & -0{,}1449 & -8{,}3614 \\ -0{,}0007 & -0{,}0033 & -0{,}3782 \\ 0{,}1465 & -9{,}7202 & 0{,}1239 \\ 0{,}0033 & -0{,}4410 & 0{,}0029 \end{bmatrix} \tag{95}$$

6.3 RESULTADOS DO EXPERIMENTO 1

O controlador do Experimento 1, apresentado na Seção 6.1, é o seguidor com realimentação de estado, projetado por LQR, sem o observador de Luenberger. Este controlador foi aplicado no modelo linear discreto e na planta do Helicóptero com 3 GDL, em laboratório tradicional e LVR.

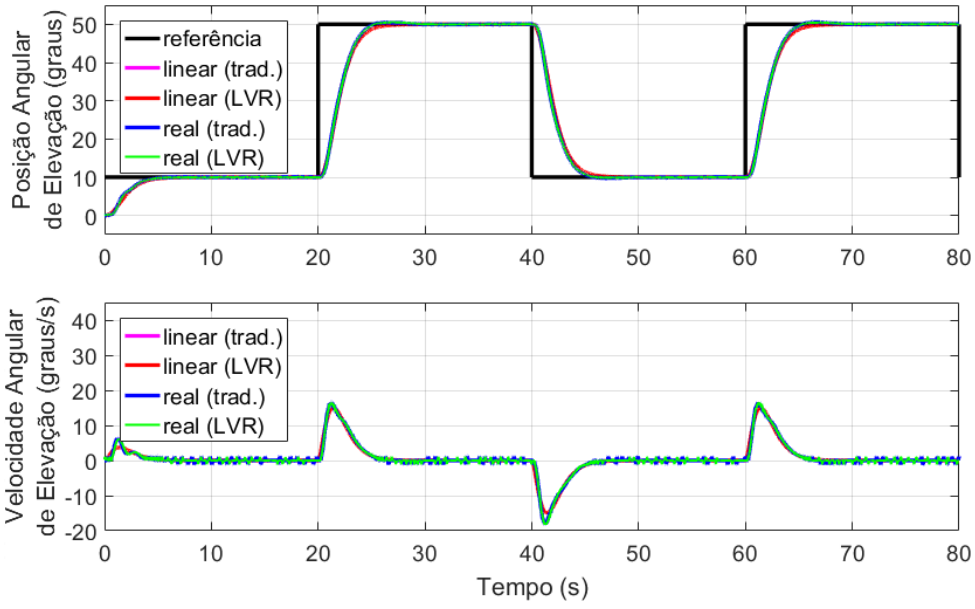
A comparação gráfica dos resultados, do Experimento 1, para os GDL controlados de deslocamento e elevação são apresentadas, respectivamente, nas Figuras 63 e 64.

Figura 63 – Experimento 1 - Resultados do movimento de deslocamento.



Fonte: Autoria própria.

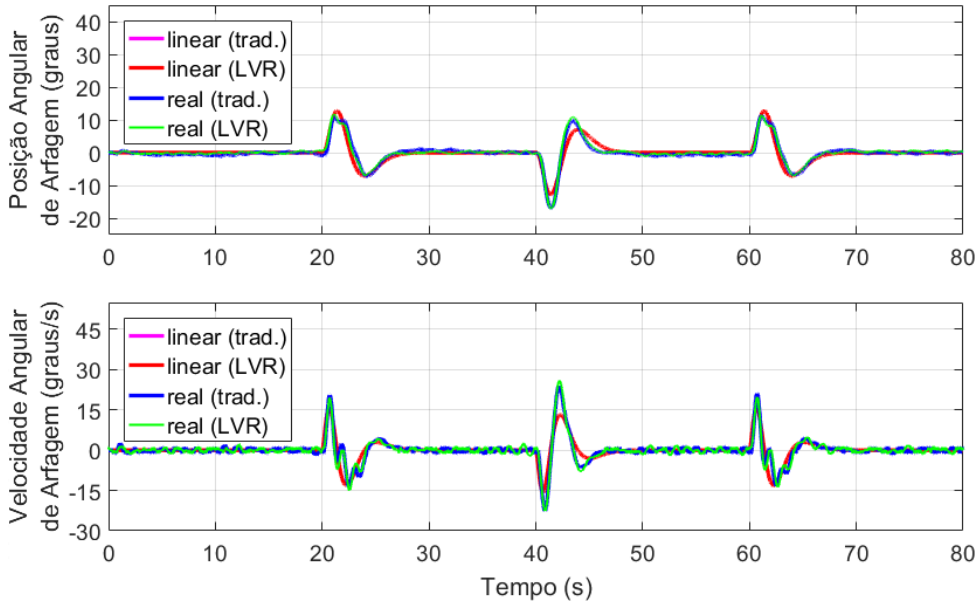
Figura 64 – Experimento 1 - Resultados do movimento de elevação.



Fonte: Autoria própria.

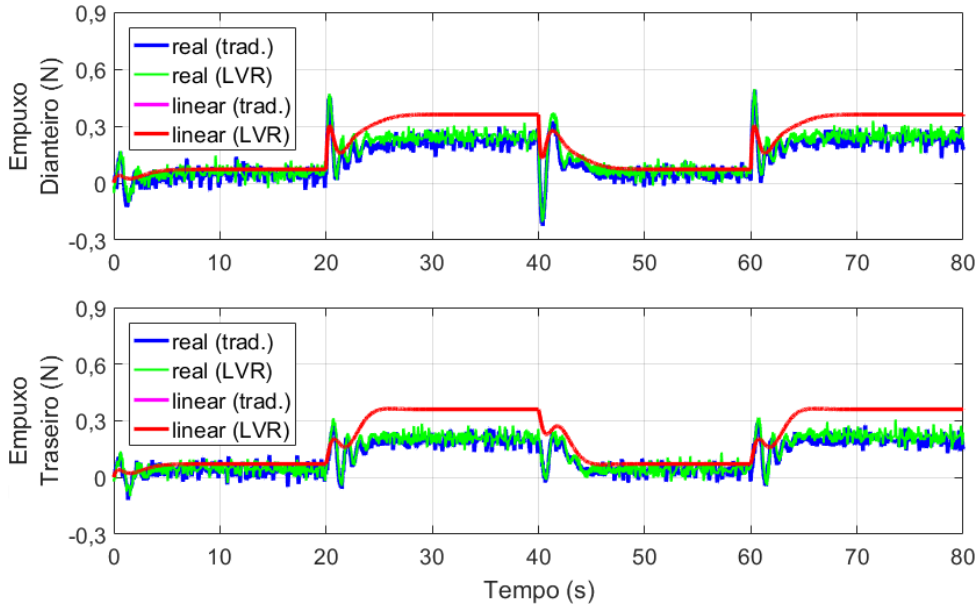
A comparação gráfica dos resultados, do Experimento 1, para o GDL de arfagem é apresentada na Figura 65. Os sinais de controle, obtidos no Experimento 1, são comparados graficamente na Figura 66.

Figura 65 – Experimento 1 - Resultados do movimento de arfagem.



Fonte: Autoria própria.

Figura 66 – Experimento 1 - Variáveis de controle.



Fonte: Autoria própria.

Nas Figuras 63, 64, 65 e 66 não é possível visualizar as respostas do controle do modelo linear discreto, realizado em laboratório tradicional. Isto ocorre, pois estas respostas são muito próximas às respostas do respectivo experimento em LVR. Para quantificar esta proximidade, por meio do NRMSE, foram comparados os resultados obtidos em LVR e laboratório tradicional, para o controle do modelo linear discreto. A Tabela 1 apresenta os valores do NRMSE para esta comparação.

Tabela 1 – Experimento 1 - NRMSE entre os dados obtidos em LVR e laboratório tradicional, quanto ao controle do modelo linear discreto.

Sinal	NRMSE
Referência do deslocamento (r_1)	1,000000000000000
Referência da elevação (r_2)	1,000000000000000
Força de empuxo dianteiro (u_1)	0,999999999998242
Força de empuxo traseiro (u_2)	0,999999999998314
Posição angular do deslocamento (y_1)	0,999999999998679
Posição angular da elevação (y_2)	0,999999999998167
Posição angular da arfagem (y_3)	0,999999999995600
Velocidade angular do deslocamento (y_4)	0,999999999996874
Velocidade angular da elevação (y_5)	0,999999999995757
Velocidade angular da arfagem (y_6)	0,999999999995414

Fonte: Autoria própria

Os resultados da Tabela 1, mostram que as respostas do controle do modelo linear, em laboratório tradicional e LVR, são aproximadamente iguais para os sinais de referência e para as variáveis de controle e saída do modelo matemático. Para executar o controle no LVR, o BDC constrói o código do controlador. Isto implica na necessidade de inicializar variáveis com todos os valores numéricos necessários para execução do controlador. Ao realizar este processo no Octave, não é possível obter com precisão os mesmos valores numéricos utilizados no controle executado no Simulink. Por conta disto, a diferença presente nos resultados da Tabela 1, é considerada proveniente da propagação do erro de inicialização das variáveis, no controle executado em LVR.

A execução do controle da planta real, seja em laboratório tradicional ou LVR, não produz sempre a mesma resposta, como no caso do controle do modelo linear discreto. Isto ocorre pelo fato da planta sofrer influências externas. Por exemplo, sistemas com hélices podem variar conforme a movimentação do ar ao seu redor. Embora sejam diferentes, as respostas do controle da planta devem apresentar o mesmo comportamento previsto pela simulação dos controladores por meio de modelos lineares ou não lineares.

Em todos os experimentos deste trabalho, o controlador do Helicóptero com 3 GDL tem o objetivo de garantir que as variáveis de saída controladas (posições angulares de deslocamento e elevação) tenham sempre o mesmo comportamento. Isto é comprovado ao comparar as respostas do controle do modelo linear discreto e da planta, exibidas nas Figura 63 e 64. A influência externa sobre o Helicóptero com 3 GDL, porém, é refletida no movimento de arfagem e nos sinais de controle. Isto é comprovado ao comparar os resultados do controle da planta e da simulação linear discreta, apresentados nas Figuras 65 e 66.

Na Tabela 2, por meio do NRMSE, foi apresentado o quanto a resposta do controle real está próxima da simulação linear, separadamente para os resultados obtidos em laboratório tradicional e LVR. Além disso, a Tabela 2 apresenta a diferença absoluta entre estes valores de NRMSE, obtidos para os resultados do laboratório tradicional e do LVR. Nesta comparação,

foram consideradas apenas as posições angulares do Helicóptero com 3 GDL.

Tabela 2 – Experimento 1 - NRMSE entre os dados do controle da planta e do modelo linear discreto, para os resultados em laboratório tradicional e LVR.

Posição Angular	Laboratório Tradicional (NRMSE)	LVR (NRMSE)	Diferença Absoluta
Deslocamento	0,9784	0,9786	0,0002
Elevação	0,9785	0,9794	0,0009
Arfagem	0,6454	0,6545	0,0091

Fonte: Autoria própria

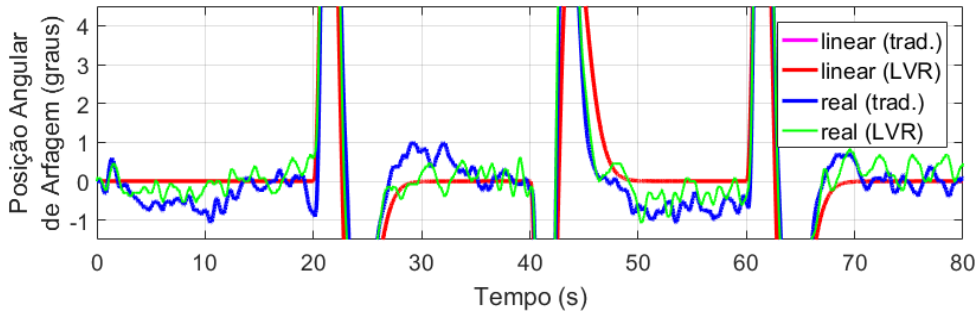
Os valores de NRMSE, na Tabela 2, confirmam que as melhores respostas de controle foram as das variáveis controladas, como já era visível nos gráficos das Figuras 63, 64 e 65. Os valores de NRMSE e da diferença absoluta podem ser interpretados como percentual, deste modo o NRMSE dos GDL controlados indica que a resposta real tem entre 97% e 98% de proximidade com a resposta simulada. As diferenças absolutas entre os valores de NRMSE para as posições angulares de deslocamento e elevação foram, respectivamente, 0,02% e 0,09%, tais diferenças são pequenas por se tratar dos GDL controlados. Para a posição angular de arfagem a diferença absoluta foi de 0,91%, por se tratar de um GDL não controlado é aceitável que a diferença obtida para arfagem seja maior que a dos GDL controlados. As pequenas diferenças analisadas para os valores de NRMSE, nas Tabelas 1 e 2, provam que os experimentos de controle realizados em LVR estão corretos, pois estão de acordo com os resultados dos experimentos em laboratório tradicional.

Embora o comportamento da arfagem no controle real seja graficamente semelhante ao da simulação linear discreta, como visto na Figura 65, os valores de NRMSE, na Tabela 2, não indicam um bom ajuste entre as respostas real e simulada da arfagem. Um dos motivos disto é que a resposta real do movimento de arfagem apresenta algumas diferenças quanto a amplitude, em comparação com a resposta da simulação linear discreta, como analisado nos gráficos da Figura 65. No trabalho de Maia (2008), esta diferença entre os comportamentos real e simulado foi reduzida ao considerar o atrito das juntas de revolução, na modelagem das dinâmicas de deslocamento e arfagem do Helicóptero com 3 GDL.

Influências externas também reduzem a qualidade do ajuste do movimento de arfagem. No Helicóptero com 3 GDL, as conexões elétricas, para leitura dos sensores e acionamento dos atuadores, são feitas por cabos. Estes produzem uma interferência no movimento de deslocamento da planta, que deve ser compensada pelo movimento de arfagem, fazendo com que a resposta do controle real apresente um desvio quando comparada a resposta da simulação linear. Tal interferência dos cabos é mais perceptível nos instantes em que o movimento de arfagem está em regime, como visto na Figura 65, nos intervalos de 0 até 20 s, de 30 até 40 s, de 50 até 60 s e de 70 até 80 s. Para melhor visualizar estes desvios a Figura 67 amplia a região em que a posição angular de arfagem está em regime, no gráfico da Figura 65. Deste

modo, é possível observar que o desvio na resposta de controle real do movimento de arfagem tem amplitude menor que $\pm$ 1,5°.

Figura 67 – Experimento 1 - Desvios do movimento de arfagem em regime.



Fonte: Autoria própria.

A diferenças entre as respostas reais e simuladas também é influenciada pelo fato da simulação ser realizada com o modelo linearizado do Helicóptero com 3 GDL. Como apresentado na Seção 2.1, os torques que produzem os movimentos da planta dependem das forças de empuxo, das distâncias medidas da planta, e de funções trigonométricas associadas as posições angulares de elevação e arfagem do Helicóptero com 3 GDL. Na obtenção do modelo linear do sistema, estas funções trigonométricas são linearizadas em torno do ponto de operação determinado para o Helicóptero com 3 GDL.

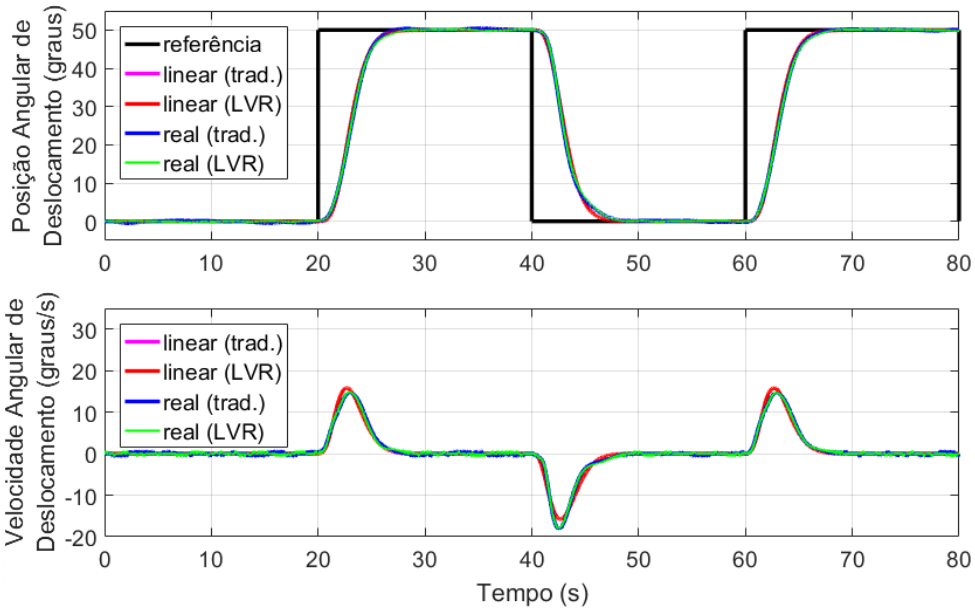
Deste modo, a diferença entre as repostas reais e simuladas, proveniente da linearização do modelo matemático, são mais visíveis nos sinais de controle. A proximidade entre as forças de empuxo aplicadas na planta real e no modelo linear é maior, quanto mais próximo o controle atua do ponto de operação determinado para linearização do modelo matemático. Como apresentado na Seção 4.6, o ponto de operação é a condição inicial, na qual as posições e velocidades angulares são zero. Isto é comprovado pela análise dos gráficos na Figura 66, nos intervalos de 0 até 20 s e de 45 até 60 s a proximidade entre os sinais de controle reais e simulados é maior, pois o controlador atua sobre as referências de 0° e 10°, respectivamente, para as posições angulares do deslocamento e da elevação.

6.4 RESULTADOS DO EXPERIMENTO 2

O controlador do Experimento 2, apresentado na Seção 6.1, é o seguidor com realimentação de estado, projetado por atribuição de autoestrutura completa, sem o observador de Luenberger. Este controlador foi aplicado no modelo linear discreto e na planta do Helicóptero com 3 GDL, em laboratório tradicional e LVR.

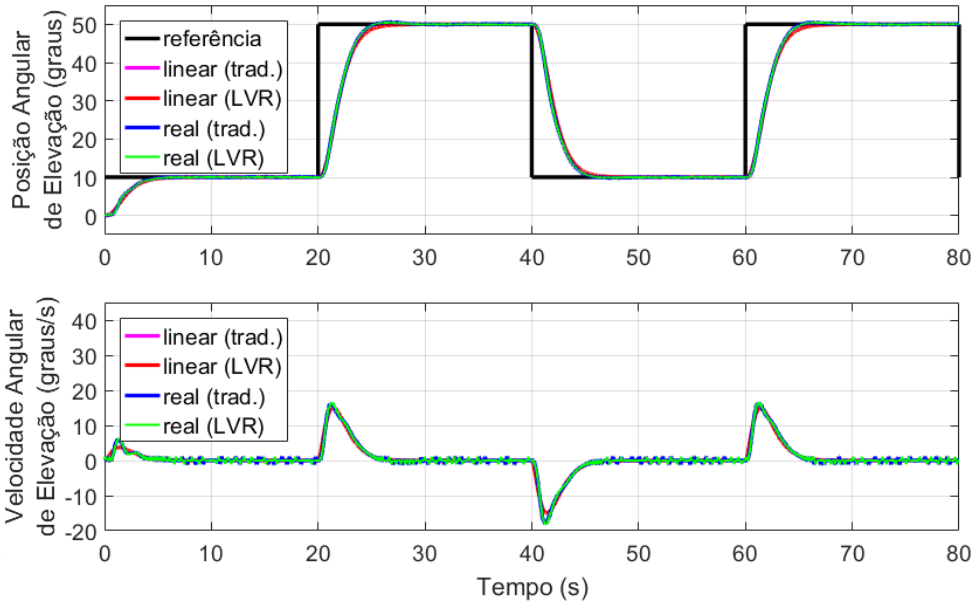
A comparação gráfica dos resultados, do Experimento 2, para os GDL controlados de deslocamento e elevação são apresentadas, respectivamente, nas Figuras 68 e 69.

Figura 68 – Experimento 2 - Resultados do movimento de deslocamento.



Fonte: Autoria própria.

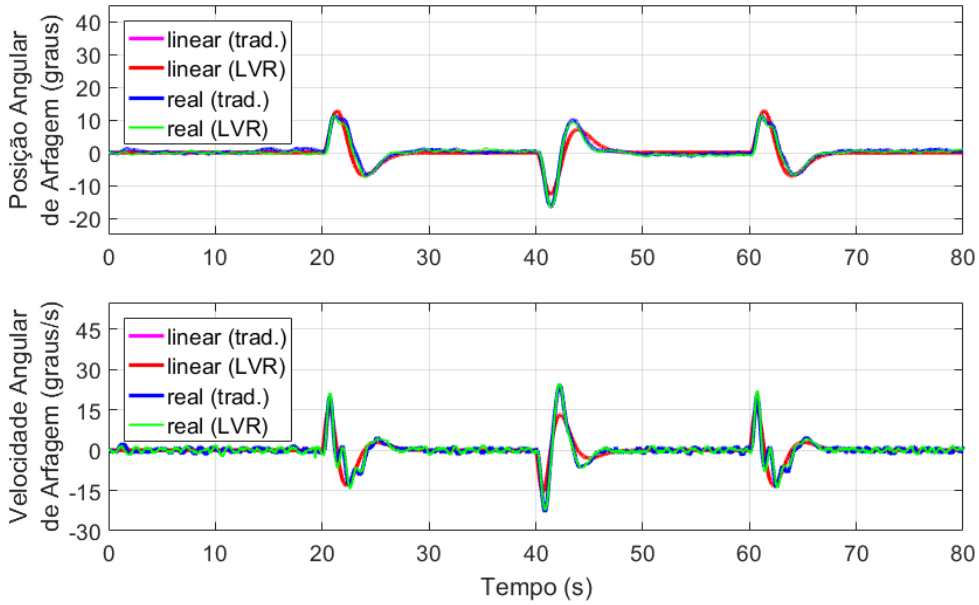
Figura 69 – Experimento 2 - Resultados do movimento de elevação.



Fonte: Autoria própria.

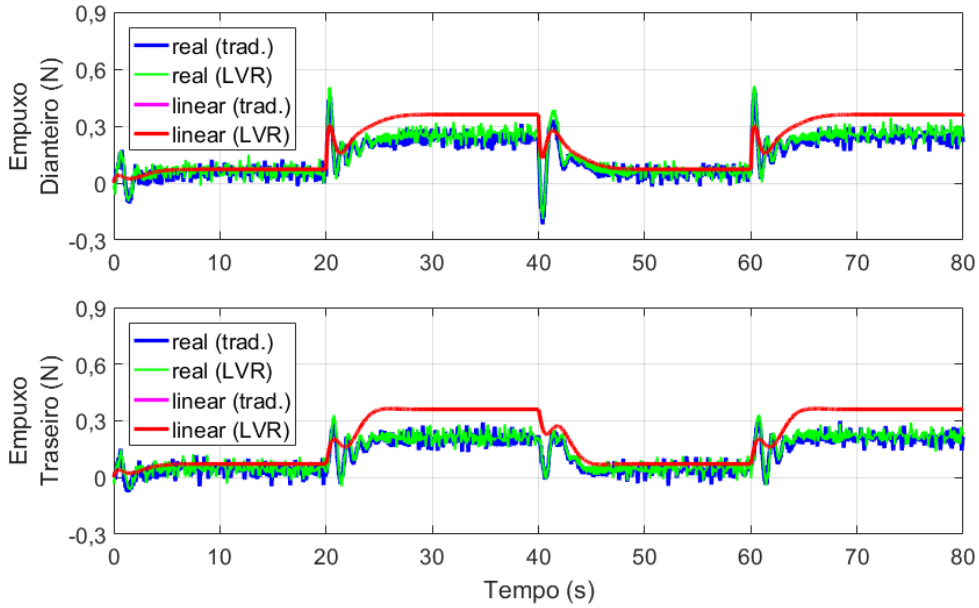
A comparação gráfica dos resultados, do Experimento 2, para o GDL de arfagem é apresentada na Figura 70. Os sinais de controle, obtidos no Experimento 2, são comparados graficamente na Figura 71.

Figura 70 – Experimento 2 - Resultados do movimento de arfagem.



Fonte: Autoria própria.

Figura 71 – Experimento 2 - Variáveis de controle.



Fonte: Autoria própria.

Assim como observado para o Experimento 1, na Seção 6.3, não é possível visualizar nas Figuras 68, 69, 70 e 71 as respostas do controle do modelo linear discreto, realizado em laboratório tradicional. Isto ocorre, pois estas respostas são muito próximas às respostas do respectivo experimento em LVR. A Tabela 3 quantifica esta proximidade, por meio do NRMSE, comparando os resultados obtidos no Experimento 2, em LVR e laboratório tradicional, quanto ao controle do modelo linear discreto.

Tabela 3 – Experimento 2 - NRMSE entre os dados obtidos em LVR e laboratório tradicional, quanto ao controle do modelo linear discreto.

Sinal	NRMSE
Referência do deslocamento (r_1)	1,000000000000000
Referência da elevação (r_2)	1,000000000000000
Força de empuxo dianteiro (u_1)	0,999999999999994
Força de empuxo traseiro (u_2)	0,999999999999996
Posição angular do deslocamento (y_1)	0,999999999999991
Posição angular da elevação (y_2)	0,999999999999999
Posição angular da arfagem (y_3)	0,999999999999973
Velocidade angular do deslocamento (y_4)	0,999999999999980
Velocidade angular da elevação (y_5)	0,999999999999996
Velocidade angular da arfagem (y_6)	0,999999999999969

Fonte: Autoria própria

Para o Experimento 2, a Tabela 4 quantifica a proximidade das respostas do controle real e da simulação linear discreta, por meio do cálculo separado do NRMSE para os resultados obtidos em laboratório tradicional e LVR. Além disso, a Tabela 4 apresenta a diferença absoluta entre estes valores de NRMSE, obtidos para os resultados do laboratório tradicional e do LVR. Nesta comparação, foram consideradas apenas as posições angulares do Helicóptero com 3 GDL.

Tabela 4 – Experimento 2 - NRMSE entre os dados do controle da planta e do modelo linear discreto, para os resultados em laboratório tradicional e LVR.

Posição Angular	Laboratório Tradicional (NRMSE)	LVR (NRMSE)	Diferença Absoluta
Deslocamento	0,9793	0,9795	0,0001
Elevação	0,9790	0,9802	0,0013
Arfagem	0,6352	0,6559	0,0207

Fonte: Autoria própria

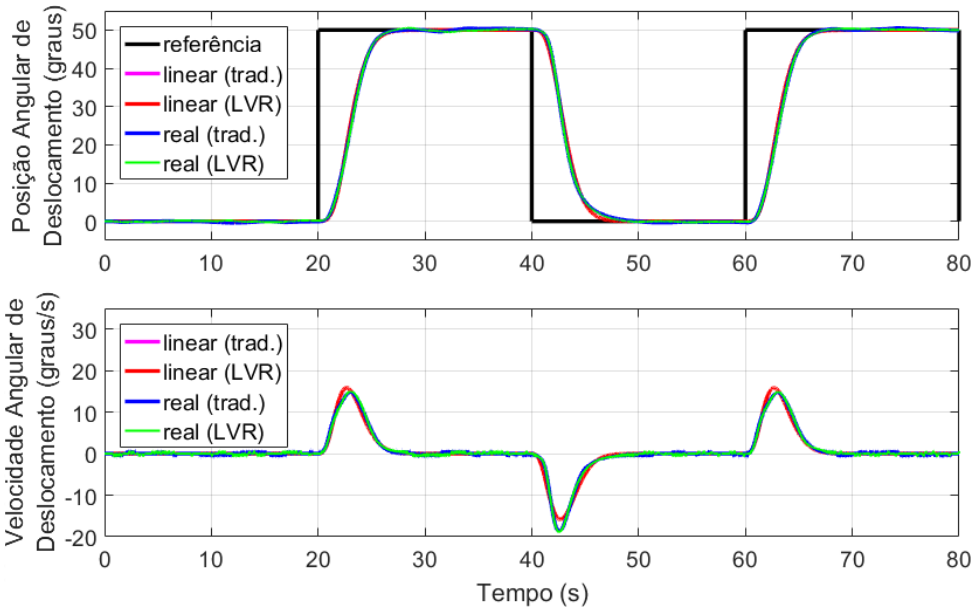
Todos os resultados do Experimento 2, apresentados nas Figuras 68, 69, 70 e 71, e nas Tabelas 3 e 4, estão de acordo e reforçam as análises dos resultados do Experimento 1, apresentada na Seção 6.3.

6.5 RESULTADOS DO EXPERIMENTO 3

O controlador do Experimento 3, apresentado na Seção 6.2, é o seguidor com realimentação de estado, projetado por LQR, com o observador de Luenberger, projetado por alocação de polos. Este controlador foi aplicado no modelo linear discreto e na planta do Helicóptero com 3 GDL, em laboratório tradicional e LVR.

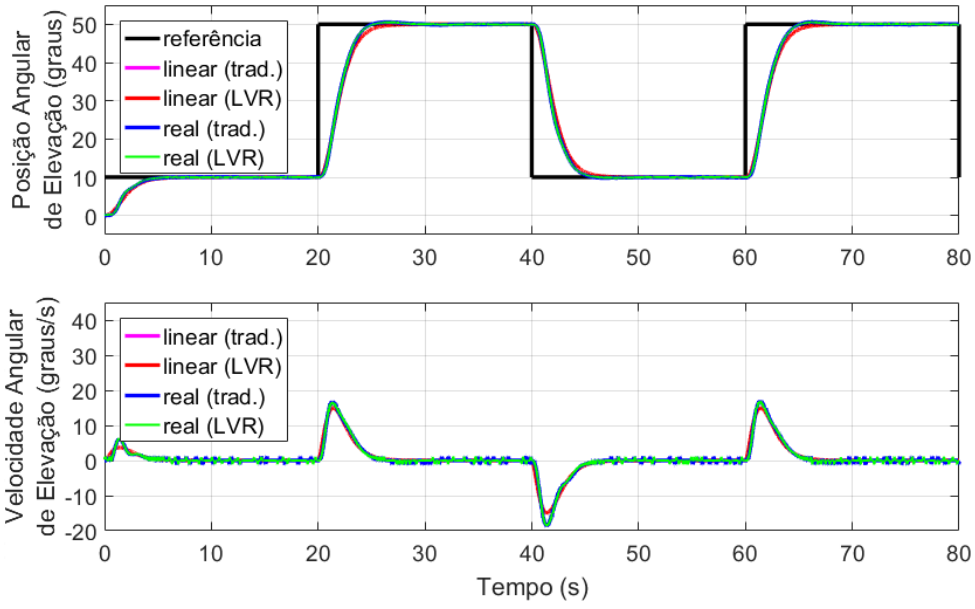
A comparação gráfica dos resultados, do Experimento 3, para os GDL controlados de deslocamento e elevação são apresentadas, respectivamente, nas Figuras 72 e 73.

Figura 72 – Experimento 3 - Resultados do movimento de deslocamento.



Fonte: Autoria própria.

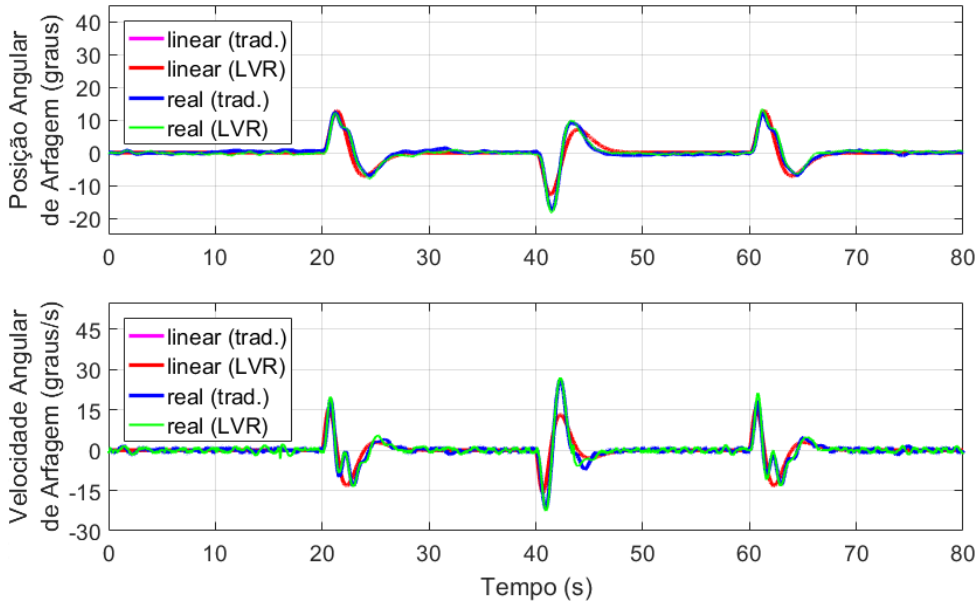
Figura 73 – Experimento 3 - Resultados do movimento de elevação.



Fonte: Autoria própria.

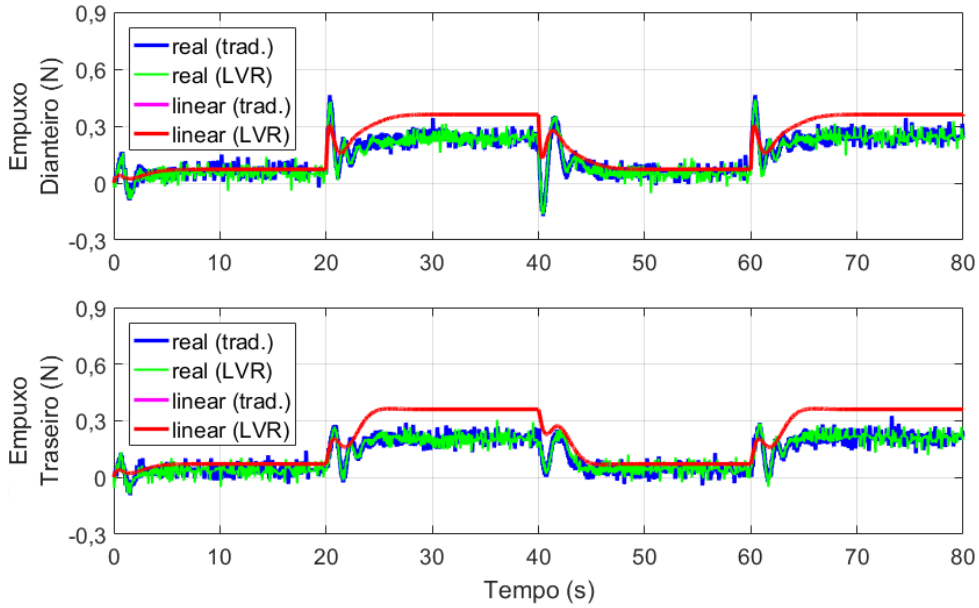
A comparação gráfica dos resultados, do Experimento 3, para o GDL de arfagem é apresentada na Figura 74. Os sinais de controle, obtidos no Experimento 3, são comparados graficamente na Figura 75.

Figura 74 – Experimento 3 - Resultados do movimento de arfagem.



Fonte: Autoria própria.

Figura 75 – Experimento 3 - Variáveis de controle.



Fonte: Autoria própria.

Assim como observado para o Experimento 1, na Seção 6.3, não é possível visualizar nas Figuras 72, 73, 74 e 75 as respostas do controle do modelo linear discreto, realizado em laboratório tradicional. Isto ocorre, pois estas respostas são muito próximas às respostas do respectivo experimento em LVR. A Tabela 5 quantifica esta proximidade, por meio do NRMSE, comparando os resultados obtidos no Experimento 3, em LVR e laboratório tradicional, quanto ao controle do modelo linear discreto.

Tabela 5 – Experimento 3 - NRMSE entre os dados obtidos em LVR e laboratório tradicional, quanto ao controle do modelo linear discreto.

Sinal	NRMSE
Referência do deslocamento (r_1)	1,000000000000000
Referência da elevação (r_2)	1,000000000000000
Força de empuxo dianteiro (u_1)	0,999999999998239
Força de empuxo traseiro (u_2)	0,999999999998310
Posição angular do deslocamento (y_1)	0,999999999998679
Posição angular da elevação (y_2)	0,999999999998167
Posição angular da arfagem (y_3)	0,999999999995601
Velocidade angular do deslocamento (y_4)	0,999999999996875
Velocidade angular da elevação (y_5)	0,999999999995758
Velocidade angular da arfagem (y_6)	0,999999999995409

Fonte: Autoria própria

Para o Experimento 3, a Tabela 6 quantifica a proximidade das respostas do controle real e da simulação linear discreta, por meio do cálculo separado do NRMSE para os resultados obtidos em laboratório tradicional e LVR. Além disso, a Tabela 6 apresenta a diferença absoluta entre estes valores de NRMSE, obtidos para os resultados do laboratório tradicional e do LVR. Nesta comparação, foram consideradas apenas as posições angulares do Helicóptero com 3 GDL.

Tabela 6 – Experimento 3 - NRMSE entre os dados do controle da planta e do modelo linear discreto, para os resultados em laboratório tradicional e LVR.

Posição Angular	Laboratório Tradicional (NRMSE)	LVR (NRMSE)	Diferença Absoluta
Deslocamento	0,9803	0,9793	0,0010
Elevação	0,9728	0,9734	0,0006
Arfagem	0,6470	0,6246	0,0224

Fonte: Autoria própria

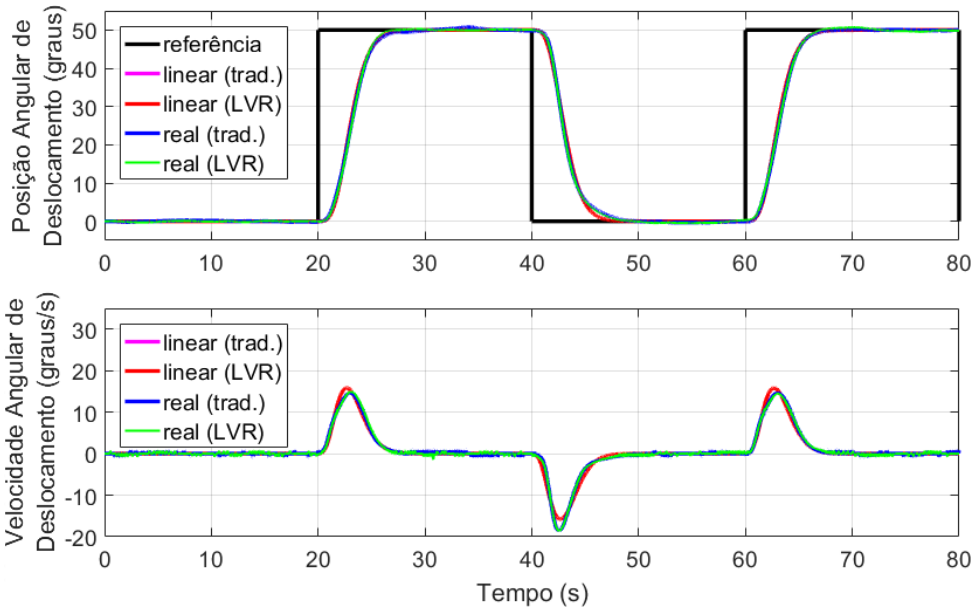
Todos os resultados do Experimento 3, apresentados nas Figuras 72, 73, 74 e 75, e nas Tabelas 5 e 6, estão de acordo e reforçam as análises dos resultados do Experimento 1, apresentada na Seção 6.3.

6.6 RESULTADOS DO EXPERIMENTO 4

O controlador do Experimento 4, apresentado na Seção 6.2, é o seguidor com realimentação de estado, projetado por atribuição de autoestrutura completa, com o observador de Luenberger, projetado por alocação de polos. Este controlador foi aplicado no modelo linear discreto e na planta do Helicóptero com 3 GDL, em laboratório tradicional e LVR.

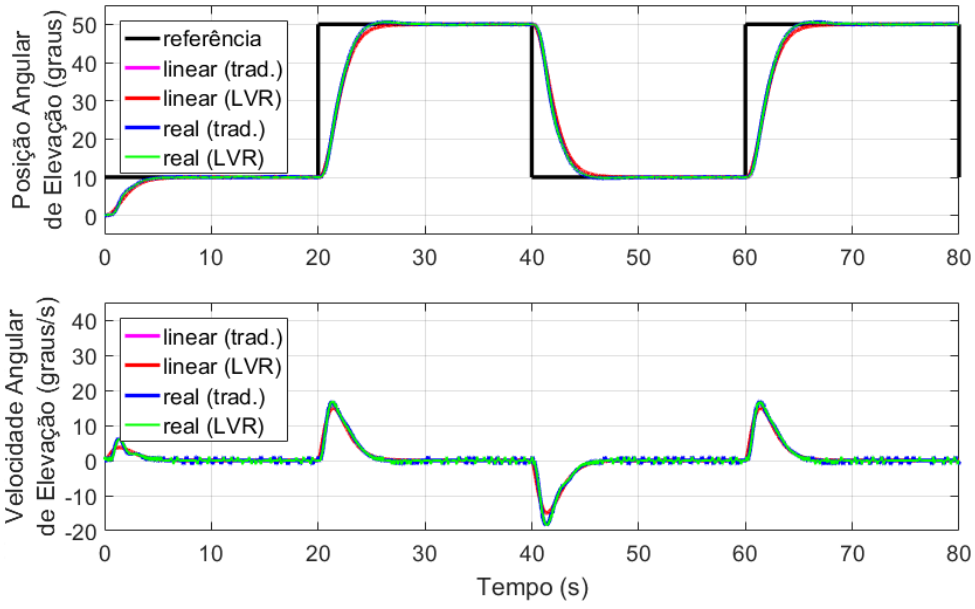
A comparação gráfica dos resultados, do Experimento 4, para os GDL controlados de deslocamento e elevação são apresentadas, respectivamente, nas Figuras 76 e 77.

Figura 76 – Experimento 4 - Resultados do movimento de deslocamento.



Fonte: Autoria própria.

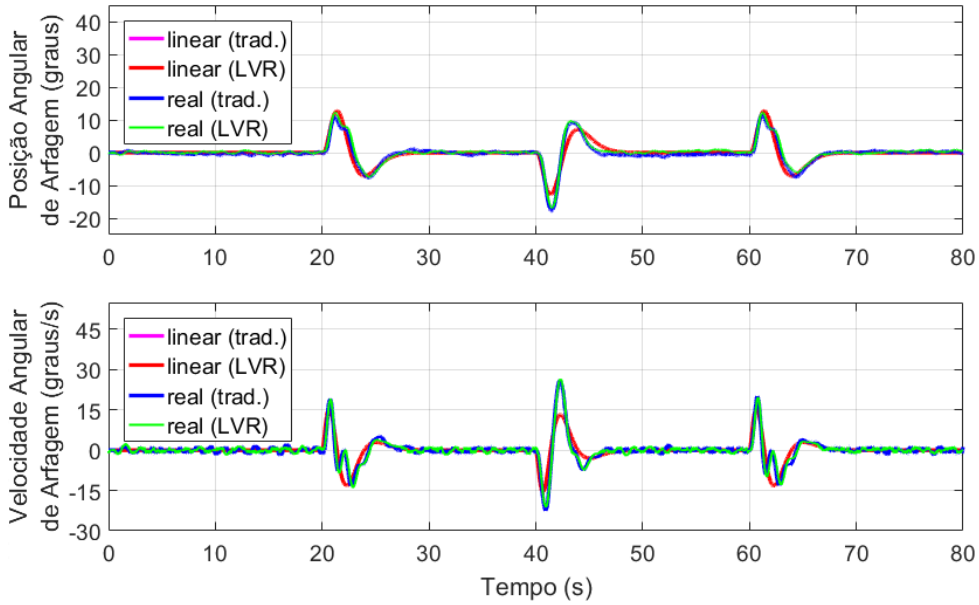
Figura 77 – Experimento 4 - Resultados do movimento de elevação.



Fonte: Autoria própria.

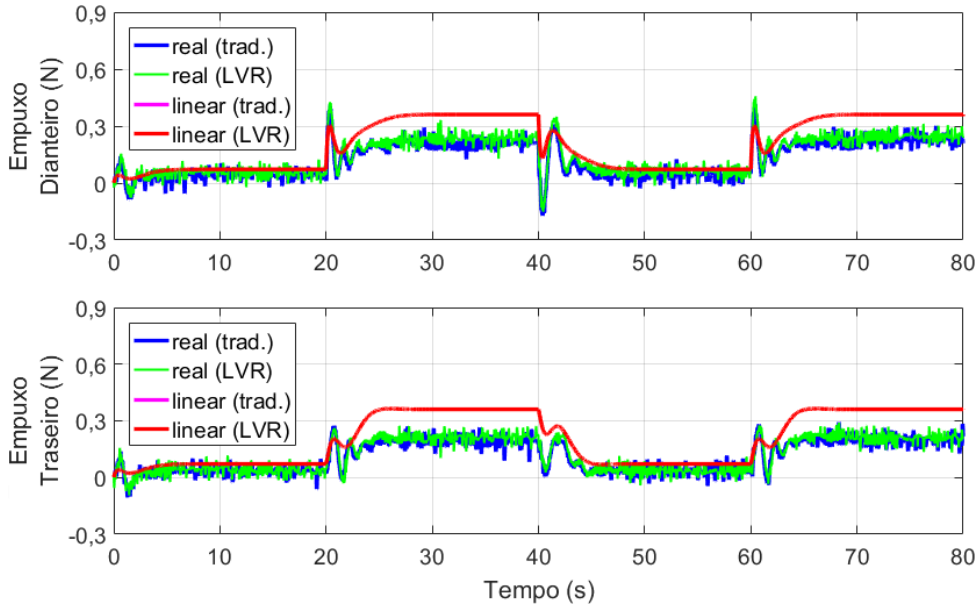
A comparação gráfica dos resultados, do Experimento 4, para o GDL de arfagem é apresentada na Figura 78. Os sinais de controle, obtidos no Experimento 4, são comparados graficamente na Figura 79.

Figura 78 – Experimento 4 - Resultados do movimento de arfagem.



Fonte: Autoria própria.

Figura 79 – Experimento 4 - Variáveis de controle.



Fonte: Autoria própria.

Assim como observado para o Experimento 1, na Seção 6.3, não é possível visualizar nas Figuras 76, 77, 78 e 79 as respostas do controle do modelo linear discreto, realizado em laboratório tradicional. Isto ocorre, pois estas respostas são muito próximas às respostas do respectivo experimento em LVR. A Tabela 7 quantifica esta proximidade, por meio do NRMSE, comparando os resultados obtidos no Experimento 4, em LVR e laboratório tradicional, quanto ao controle do modelo linear discreto.

Tabela 7 – Experimento 4 - NRMSE entre os dados obtidos em LVR e laboratório tradicional, quanto ao controle do modelo linear discreto.

Sinal	NRMSE
Referência do deslocamento (r_1)	1,000000000000000
Referência da elevação (r_2)	1,000000000000000
Força de empuxo dianteiro (u_1)	0,999999999999899
Força de empuxo traseiro (u_2)	0,999999999999910
Posição angular do deslocamento (y_1)	0,999999999999991
Posição angular da elevação (y_2)	0,999999999999998
Posição angular da arfagem (y_3)	0,999999999999958
Velocidade angular do deslocamento (y_4)	0,999999999999977
Velocidade angular da elevação (y_5)	0,999999999999984
Velocidade angular da arfagem (y_6)	0,999999999999893

Fonte: Autoria própria

Para o Experimento 4, a Tabela 8 quantifica a proximidade das respostas do controle real e da simulação linear discreta, por meio do cálculo separado do NRMSE para os resultados obtidos em laboratório tradicional e LVR. Além disso, a Tabela 8 apresenta a diferença absoluta entre estes valores de NRMSE, obtidos para os resultados do laboratório tradicional e do LVR. Nesta comparação, foram consideradas apenas as posições angulares do Helicóptero com 3 GDL.

Tabela 8 – Experimento 4 - NRMSE entre os dados do controle da planta e do modelo linear discreto, para os resultados em laboratório tradicional e LVR.

Posição Angular	Laboratório Tradicional (NRMSE)	LVR (NRMSE)	Diferença Absoluta
Deslocamento	0,9794	0,9784	0,0010
Elevação	0,9728	0,9739	0,0011
Arfagem	0,6354	0,6279	0,0074

Fonte: Autoria própria

Todos os resultados do Experimento 4, apresentados nas Figuras 76, 77, 78 e 79, e nas Tabelas 7 e 8, estão de acordo e reforçam as análises dos resultados do Experimento 1, apresentada na Seção 6.3.

6.7 ANÁLISE GERAL DAS RESPOSTAS DOS EXPERIMENTOS

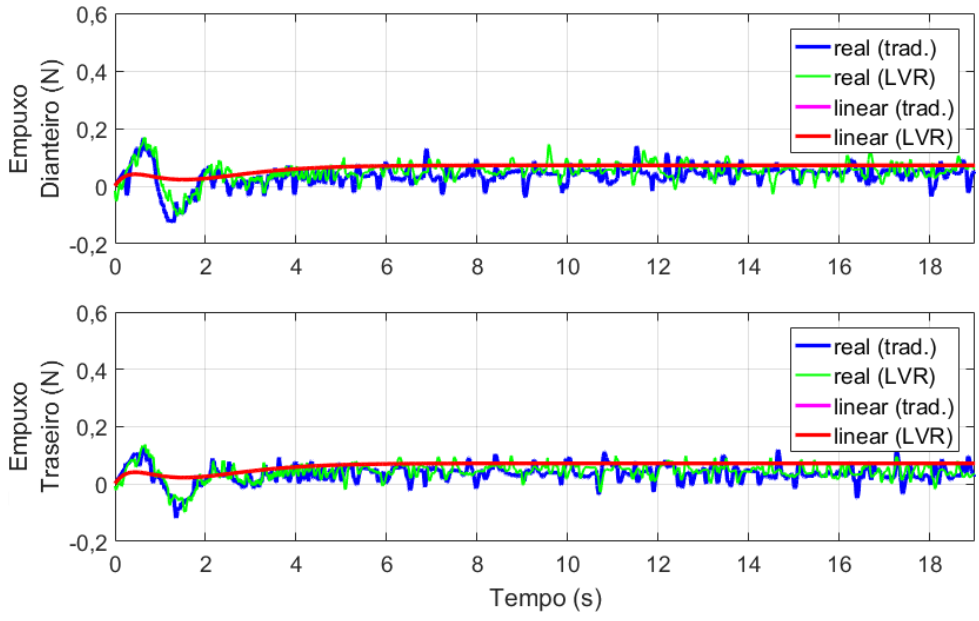
No trabalho de Shimada (2015), a partir da análise das respostas do controle linear, não linear e real foram feitas as seguintes sugestões, quanto a melhoria da resposta do controle real do Helicóptero com 3 GDL.

- **Reduzir o torque sobre o movimento de deslocamento:** Nos experimentos realizados por Shimada (2015), os atuadores do Helicóptero com 3 GDL estavam equipados

com hélices girando no mesmo sentido. Por conta disso, um efeito de torque adicional surgia sobre o movimento de deslocamento. Isto causava uma diferença entre as respostas do controle real e simulado, uma vez que o modelo matemático não contemplava esta condição. Para compensar este efeito, Shimada (2015) sugeriu a instalação de um par de hélices contra rotativas, tal sugestão foi acatada neste trabalho. Deste modo, nos Experimentos 1, 2, 3 e 4 não foi constatada esta tendência de movimento sobre o grau de liberdade de deslocamento.

- **Reduzir o ruído presente no sinal de controle:** A técnica de controle aplicada em Shimada (2015) foi o seguidor com realimentação de estado, neste caso o sinal de controle é determinado a partir da informação das posições e também das velocidades angulares do sistema. Na determinação das velocidades angulares, Shimada (2015) aplicou a derivação discreta sobre o sinal digital dos encoders, isto resulta em um sinal ruidoso, como apresentado na Seção 4.8. Neste caso, o ruído da velocidade angular aparece no sinal de controle calculado. Neste trabalho o ruído no sinal de controle foi reduzido pela estimação das velocidades angulares. Nos Experimentos 1 e 2, a estimação das velocidades angulares foi feita por TL. Nos Experimentos 3 e 4, a estimação das velocidades angulares foi feita pelo observador de Luenberger. Embora as velocidades angulares e os sinais de controle ainda apresentem ruído, ambas técnicas foram capazes de reduzir sua amplitude. O TL, porém, é uma técnica interessante por não depender do conhecimento do modelo matemático do sistema. Nos Experimentos 1, 2 ,3 e 4, a amplitude do ruído no sinal de controle foi menor que 0,2 N de pico a pico. A Figura 80 mostra os gráficos dos sinais de controle do Experimento 1, ampliados para o intervalo de 1 até 19 s;

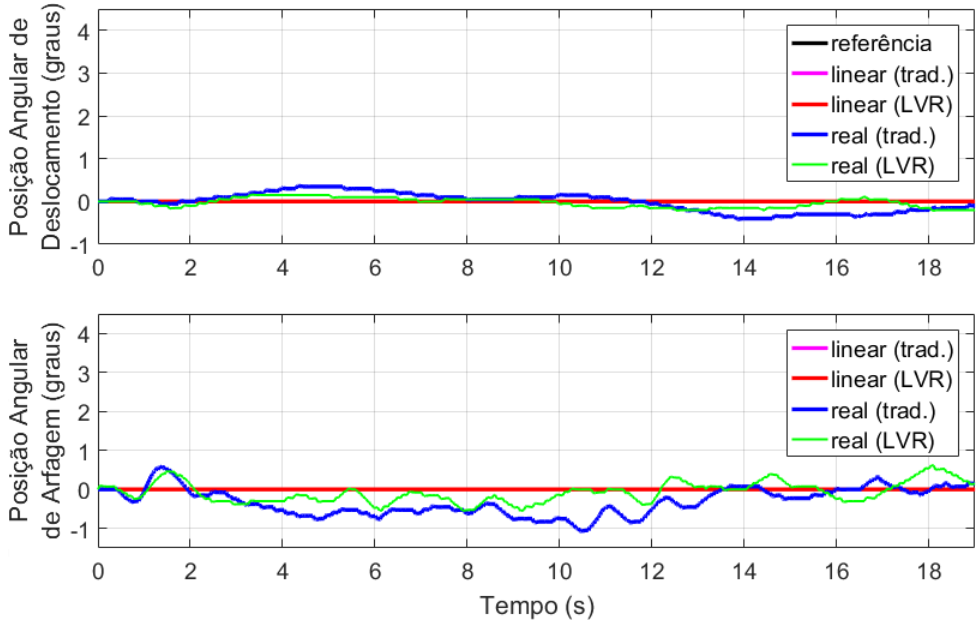
Figura 80 – Experimento 1 - Ruído nos sinais de controle.



Fonte: Autoria própria.

- **Melhorar a perda de eficiência dos atuadores:** Durante os experimentos, Shimada (2015) constatou que os motores aqueciam. Conforme a temperatura dos motores aumentava, a eficiência na produção das forças de empuxo diminuía. Deste modo, os sinais das forças de empuxo, calculados pelo controlador, apresentavam uma tendência de crescimento ao longo da execução do experimento. Neste trabalho, como apresentado pelo processo de prototipagem virtual na Seção 4.6, foi determinado uma posição para o contrapeso, que reduz a exigência sobre os atuadores do Helicóptero com 3 GDL. Deste modo, nos Experimentos 1, 2, 3 e 4 não houve o aquecimento e a consequente perda de eficiência dos atuadores;
- **Reduzir a oscilação na resposta do movimento de arfagem:** Nos experimentos de controle real, Shimada (2015) observou que a resposta da posição angular de arfagem era oscilatória. Este comportamento também foi percebido nos Experimentos 1, 2, 3 e 4. No entanto, esta oscilação foi considerada aceitável, uma vez que no Helicóptero com 3 GDL o movimento de arfagem é que permite a estabilização do movimento de deslocamento, em torno do sinal de referência. Neste trabalho, a oscilação da arfagem se mantém menor que 2° de pico a pico, sendo suficiente para estabilizar a posição angular do deslocamento, em torno da referência, com uma oscilação inferior à 2° de pico a pico. A Figura 81 apresenta os gráficos das posições angulares de deslocamento e arfagem do Experimento 1, ampliados para o intervalo de 1 até 19 s.

Figura 81 – Experimento 1 - Oscilação nas posições angulares de deslocamento e arfagem.



Fonte: Autoria própria.

Nas Seções 6.3, 6.4, 6.5 e 6.6, foi verificado, individualmente, que as abordagens de controle dos Experimentos 1, 2, 3 e 4, respectivamente, apresentaram bons resultados para

comparação entre as respostas do controle real e da simulação linear, tanto em laboratório tradicional quanto em LVR. Porém, ainda é necessário realizar uma comparação de desempenho entre os controladores dos quatro experimentos.

Para tanto, foi utilizado o critério de ITAE (*Integral of the Time-Weighted Absolute Error* ou Integral do Erro Absoluto Ponderado pelo Tempo, em português). Considerando o tempo dado por t, e o sinal de erro entre referência e variável de saída controlada dado por $e(t)$, o cálculo do ITAE é apresentado na Equação (96). Quanto menor o valor obtido para este critério, melhor é o desempenho do controlador em reduzir o erro entre as referências e as respectivas variáveis de saída controladas. Ao ponderar o valor absoluto do erro no tempo, o critério de ITAE penaliza em especial os casos em que o erro persiste ao longo do tempo (SEBORG *et al.*, 2011). Como visto em Martins (2005), a avaliação pelo critério de ITAE também pode ser utilizada na otimização do projeto de controladores.

$$ITAE = \int_0^\infty t\,|e(t)|\,dt \tag{96}$$

O Quadro 2 apresenta a comparação dos valores de ITAE calculados para todos as respostas das variáveis controladas, dos Experimentos 1, 2, 3 e 4.

Quadro 2 – Comparação do desempenho dos controles de deslocamento e elevação dos Experimentos 1, 2, 3 e 4, por meio do critério de ITAE.

		ITAE			
Experimento	**Posição Controlada**	**Linear (trad.)**	**Linear (LVR)**	**Real (trad.)**	**Real (LVR)**
1	Deslocamento	69284,3306	69284,3306	72018,9618	71256,7657
	Elevação	37807,3158	37807,3158	36847,4635	36994,3848
2	Deslocamento	69468,3406	69468,3406	71532,0005	71911,1545
	Elevação	37816,3723	37816,3723	36844,8139	36945,2033
3	Deslocamento	69284,3306	69284,3306	71863,6442	71865,8787
	Elevação	37807,3158	37807,3158	36591,6215	36592,5405
4	Deslocamento	69468,3406	69468,3406	71874,5428	71990,7194
	Elevação	37816,3723	37816,3723	36488,4348	36601,3279

Fonte: Autoria própria.

Os valores de ITAE para as respostas da simulação linear realizadas em laboratório tradicional (terceira coluna do Quadro 2) e LVR (quarta coluna do Quadro 2) são aproximadamente iguais, para todos os experimentos. No caso do controle real, os valores de ITAE calculados para os resultados obtidos em laboratório tradicional (quinta coluna do Quadro 2) e LVR (sexta coluna do Quadro 2) são próximos, para todos os experimentos. Tudo isto reforça a análise, apresentada na Seção 6.3, de que os experimentos realizados em LVR estão corretos, por estarem de acordo com os experimentos realizados em laboratório tradicional.

Considerando as respostas real e linear obtidas em laboratório tradicional e LVR, a comparação entre os valores de ITAE dos Experimentos 1 e 3, assim como a comparação

entre os valores de ITAE dos Experimentos 2 e 4, permitem concluir, que os métodos de TL e observador de Luenberger, utilizados na estimação das velocidades angulares, influenciaram a resposta dos controladores de maneira semelhante.

Para as respostas de controle da planta real e do modelo linear discreto obtidas em laboratório tradicional e LVR, a comparação entre os valores de ITAE dos Experimentos 1 e 2, bem como a comparação entre os valores de ITAE dos Experimentos 3 e 4, mostram que os controladores projetados pela atribuição de autoestrutura completa e por LQR tem aproximadamente o mesmo desempenho. Tal conclusão é valida apenas para comparação entre os controles projetados empiricamente neste trabalho.

7 CONCLUSÕES

Nesta dissertação foi desenvolvido um LVR para aplicação em sistemas de controle. As decisões quanto a escolha das tecnologias, a determinação das funcionalidades, e o projeto da arquitetura, tomadas durante o desenvolvimento, tiveram o objetivo de garantir que o LVR contribuísse com a sua área de pesquisa e a comunidade acadêmica ao apresentar uma série de características. Tradicionalmente, a interface do laboratório remoto é separada da interface do laboratório virtual. Neste trabalho, a interface de ambos é a mesma, facilitando o uso integrado de laboratórios virtuais com remotos, assim como a utilização dos LVRs para diferentes objetivos de aprendizado. Isto é possível pois a interface do LVR atua como uma IDE para o Octave CLI, em execução no lado do servidor. Esta IDE funciona de maneira genérica e dá ao usuário a capacidade de desenvolver o próprio experimento de controle, sendo este virtual ou remoto.

Neste trabalho foi desenvolvido o pacote BDC. Este permite, por meio de um *script* em linguagem MATLAB/Octave, definir o diagrama de blocos de um controlador. O BDC disponibiliza um conjunto de blocos prontos, além das funções para construir, gerar e executar o controlador definido, tanto como simulação quanto no sistema real. A combinação da interface do LVR (IDE), do Octave CLI e do pacote BDC proporciona ao usuário a liberdade de construção das técnicas de controle. A simples combinação da interface do LVR (IDE) com o Octave CLI é suficiente para permitir ao usuário projetar os ganhos dos controladores construídos. Isto contribui para que o LVR seja aplicado no ensino e pesquisa de uma maior diversidade de técnicas de controle.

No laboratório tradicional, os controles real e simulado são definidos, por meio de um diagrama de blocos, e executados no Simulink. No caso do controle real, a integração entre Simulink e planta é realizada pela placa de aquisição PCI-6602 da National Instruments. O projeto dos ganhos dos controladores é feito por meio de *scripts* em linguagem MATLAB. Para garantir esta mesma experiência de trabalho no LVR, o software MATLAB/Simulink foi substituído pelo Octave CLI em conjunto com o BDC, e a PCI-6602 foi substituída pelo NUCLEO-F767ZI. Deste modo, no LVR, os controles real e simulado também são definidos por diagrama de blocos, utilizando o BDC. No caso do controle real, a integração com a planta é realizada pelo NUCLEO-F767ZI. Quanto ao projeto dos ganhos dos controladores, o Octave é capaz de executar exatamente o mesmo *script* utilizado em laboratório tradicional. Isto contribui para que a inclusão do LVR na metodologia de ensino respeite seu papel educacional, ao complementar a experimentação em laboratórios tradicionais.

Os controles virtual e remoto desenvolvidos na IDE do LVR são executados exclusivamente no lado do servidor, por meio do sistema de aquisição e controle definido para o LVR, apresentado na Figura 48. Para integrar o LVR com diferentes plantas de controle é necessário fazer alterações, apenas nas tecnologias que compõem o sistema de aquisição

e controle do LVR, mais especificamente no pacote BDC. A interface Web (IDE) trabalha de maneira genérica, funcionando para qualquer planta de controle da mesma forma. Deste modo, ao desenvolver LVRs para sistemas de controle, é possível focar apenas em aspectos de instrumentação e controle. Todo o desenvolvimento Web é reutilizado, contribuindo assim para facilitar o desenvolvimento de LVRs, permitindo aos docentes focar nos aspectos pedagógicos dos LVRs. Uma importante contribuição tecnológica para o desenvolvimento de LVRs é que todas as tecnologias utilizadas são *open source*.

Para validar o funcionamento do LVR desenvolvido foi utilizado o sistema de um Helicóptero com 3 GDL, desenvolvido por Shimada (2015). A realização dos experimentos de controle sobre o Helicóptero com 3 GDL, tanto em laboratório tradicional, quanto em LVR, necessitou do estudo das questões de modelagem matemática do sistema e também dos aspectos de instrumentação dos atuadores e sensores. Este estudo contribuiu com a solução de alguns problemas na resposta de controle do sistema, levantados por Shimada (2015).

Durante a modelagem matemática do Helicóptero com 3 GDL, pelo método de prototipagem virtual, foram realizadas as análises dinâmica e estática do Helicóptero com 3 GDL, permitindo assim posicionar o contrapeso no sistema. O objetivo deste estudo foi reduzir a exigência na geração das forças de empuxos pelos atuadores. Deste modo, os atuadores deixaram de aquecer e perder eficiência durante os experimentos. A instalação de um par de hélices contra rotativas no Helicóptero com 3 GDL resolveu o problema do surgimento de uma tendência de movimento sobre o GDL de deslocamento, resultante de um torque adicional produzido pelo fato das hélices girarem no mesmo sentido. A estimação das velocidades angulares dos GDL de deslocamente, elevação e arfagem do Helicóptero com 3 GDL, por TL ou observador de Luenberger, reduziu a amplitude do ruído presente no sinal das variáveis de controle.

Quatro diferentes configurações de experimento de controle foram executadas sobre o Helicóptero com 3 GDL, integrado ao laboratório tradicional e também ao LVR. Nestes experimentos foram utilizados o controle seguidor com realimentação de estado, e o controle seguidor com realimentação de estado em conjunto com o observador de Luenberger. Ambos controladores foram projetados empiricamente, tanto por LQR, quanto por atribuição de autoestrutura completa, para apresentarem um desempenho semelhante, deste modo a comparação por ITAE mostrou resultados muito próximos para os quatro experimentos. Os resultados permitiram concluir que os experimentos realizados em LVR estão corretos, pois estão de acordo com os resultados obtidos para os experimentos em laboratório tradicional, como foi avaliado tanto pelo critério do NRMSE quanto por ITAE.

Por meio dos experimentos, também foi possível concluir que o modelo matemático do Helicóptero com 3 GDL, obtido pela prototipagem virtual, representa adequadamente o sistema real. Embora existam diferenças entre as respostas do controle real e simulado para o GDL de arfagem, estas são aceitáveis por se tratar de um GDL não controlado, que sofre com a influência de fatores externos, como a interferência produzida pelos cabos. Além disso, as

oscilações presentes no movimento de arfagem foram consideradas pequenas e aceitáveis por serem necessárias para a estabilização do GDL de deslocamento. Quanto a instrumentação dos atuadores e sensores, os sistemas de aquisição e controle tradicional, apresentado na Figura 39, e do LVR, apresentado na Figura 48, funcionaram de maneira equivalente e com um bom desempenho.

7.1 TRABALHOS FUTUROS

Alguns trabalhos futuros considerados são:

- Como o LVR é essencialmente uma aplicação Web, é necessário finalizar as capacidades de acesso e gerenciamento de usuários, bem como desenvolver algumas medidas de segurança, antes de disponibilizar o acesso por meio da Internet;
- Desenvolver mais blocos no pacote BDC, para proporcionar ao usuário mais opções no desenvolvimento de técnicas de controle;
- Desenvolver os recursos para construção gráfica dos diagramas de blocos, assim como no Simulink;
- Avaliar o LVR em conjunto com outro sistema de controle;
- Avaliar a necessidade da modelagem do atrito viscoso nas juntas de revolução, para aproximar as respostas do controle real e simulado;
- Substituir os cabos de instrumentação da planta, por tecnologias de comunicação sem fio. Para o sistema do Helicóptero com 3 GDL, controlado com tempo de amostragem de 0,005 segundos, é necessário utilizar a comunicação wi-fi;
- Disponibilizar o LVR para utilização em disciplinas e posteriormente a avaliação pelo discente.

Referências

ABDULWAHED, M.; NAGY, Z. K. The trilab, a novel ict based triple access mode laboratory education model. **Computers & Education**, v. 56, n. 1, p. 262–274, 2011. ISSN 0360-1315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360131510002186>.

ANGULARJS. **AngularJS**. 2019. Disponível em: <https://angularjs.org/>. Acesso em: 12 de nov. de 2019.

ANGULARJS. **Developer Guide Introduction**. 2019. Disponível em: <https://docs.angularjs.org/guide/introduction>. Acesso em: 12 de nov. de 2019.

ANTSAKLIS, P. *et al.* Report on the nsf/css workshop on new directions in control engineering education. **IEEE Control Systems Magazine**, v. 19, n. 5, p. 53–58, Oct 1999. Disponível em: <https://ieeexplore.ieee.org/document/793442>.

APKARIAN, J.; DAWES, A. Interactive control education with virtual presence on the web. In: **Proceedings of the 2000 American Control Conference**. Chicago, USA: IEEE, 2000. v. 6, p. 3985–3990. ISSN 0743-1619. Disponível em: <https://ieeexplore.ieee.org/document/876970>.

APKARIAN, J.; LÉVIS, M.; FULFORD, C. **Laboratory guide**: 3 dof helicopter experiment for matlab/simulink users. Markham, Canadá, 2012. 21 p. Disponível em: <https://www.quanser.com/products/3-dof-helicopter/>. Acesso em: 03 de outubro de 2019.

APOLINÁRIO, G. d. C. **Detecção de falhas em sistemas incertos com atraso no sinal de controle**. 2009. 82 f. Dissertação (mestrado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2009. Disponível em: <http://hdl.handle.net/11449/87142>.

ARM. **Mbed OS 5**. 2019. Disponível em: <https://os.mbed.com/>. Acesso em: 12 de nov. de 2019.

ARM. **Mbed OS 5**: Docs. 2019. Disponível em: <https://os.mbed.com/docs/mbed-os/>. Acesso em: 12 de nov. de 2019.

ÅSTRÖM, K. J.; HÄGGLUND, T. **PID controllers**: theory, design, and tuning. 2. ed. Research Triangle Park, NC: Instrument Society of America, 1995.

AUSTERLITZ, H. **Data acquisition techniques using PCs**. 2. ed. San Diego, CA: Academic Press, 2003.

BERMÚDEZ-ORTEGA, J. *et al.* Remote web-based control laboratory for mobile devices based on ejss, raspberry pi and node.js*. **IFAC-PapersOnLine**, v. 48, n. 29, p. 158 – 163, 2015. ISSN 2405-8963. {IFAC} Workshop on Internet Based Control Education {IBCE15Brescia} (Italy), November 4-6, 2015. Disponível em: <http://www.sciencedirect.com/science/article/pii/S240589631502488X>.

BOOTSTRAP. **Bootstrap**. 2019. Disponível em: <http://getbootstrap.com/>. Acesso em: 12 de nov. de 2019.

BOSTOCK, M. **Data-Driven Documents**. 2019. Disponível em: <https://d3js.org/>. Acesso em: 12 de nov. de 2019.

BREGANON, R. **Controle de arfagem e guinada de um sistema de hélices paralelas**. 2009. 82 f. Dissertação (mestrado) — Universidade de São Paulo. Escola de Engenharia de São Carlos, São Carlos - SP, 2009. Disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18148/tde-08062010-184803/pt-br.php>.

BRINSON, J. R. Learning outcome achievement in non-traditional (virtual and remote) versus traditional (hands-on) laboratories: a review of the empirical research. **Computers & Education**, v. 87, p. 218 – 237, 2015. ISSN 0360-1315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360131515300087>.

BUZACHERO, L. F. S. **Otimização de controladores robustos de sistemas dinâmicos sujeitos a falhas estruturais**. 2010. 72 f. Dissertação (mestrado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2010. Disponível em: <http://hdl.handle.net/11449/87054>.

BUZACHERO, L. F. S. **Controle robusto chaveado de sistemas lineares variantes no tempo com aplicação em falhas estruturais**. 2014. 120 f. Tese (doutorado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2014. Disponível em: <http://hdl.handle.net/11449/110512>.

CASCIARO, M. **Node.js Design Patterns**. 1. ed. Birmingham: Packt Publishing Ltd, 2014.

CASTAÑEDA, H. *et al.* Continuous differentiator based on adaptive second-order sliding-mode control for a 3-dof helicopter. **IEEE Transactions on Industrial Electronics**, v. 63, n. 9, p. 5786–5793, Sep 2016. ISSN 0278-0046. Disponível em: <https://ieeexplore.ieee.org/document/7469853>.

CHACÓN, J. *et al.* A new generation of online laboratories for teaching automatic control. **IFAC-PapersOnLine**, v. 48, n. 29, p. 140–145, 2015. ISSN 2405-8963. IFAC Workshop on Internet Based Control Education IBCE15. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405896315024854>.

CHACÓN, J. *et al.* Ejs, jil server, and labview: an architecture for rapid development of remote labs. **IEEE Transactions on Learning Technologies**, v. 8, n. 4, p. 393–401, Oct 2015. ISSN 1939-1382. Disponível em: <https://ieeexplore.ieee.org/document/7004872>.

CHAVES, W. S. *et al.* Parameters identification of a direct current motor using the trust region algorithm. **International Journal of Advanced Engineering Research and Science**, v. 4, n. 12, p. 162–169, 2017. ISSN 2349-6495. Disponível em: <https://dx.doi.org/10.22161/ijaers.4.12.24>.

CHOUDHARY, S. K. Lqr based pid controller design for 3-dof helicopter system. **International Journal of Computer, Information, Systems and Control Engineering**, v. 8, n. 8, p. 1375–1380, 2014.

CODEMIRROR. **CodeMirror**. 2019. Disponível em: <https://codemirror.net/>. Acesso em: 12 de nov. de 2019.

DASSAULT SYSTÈMES. **SOLIDWORKS Simulation**. 2019. Disponível em: <https://www.solidworks.com/product/solidworks-simulation>. Acesso em: 16 de outubro de 2019.

D'AZZO, J. J.; HOUPIS, C. H. **Linear control system analysis and design**: conventional and modern. 4. ed. New York, NY: McGraw-Hill, 1995.

D'AZZO, J. J.; HOUPIS, C. H.; SHELDON, S. N. **Linear control system analysis and design with matlab**. 5. ed. New York, NY: Marcel Dekker, 2003.

DORF, R. C.; BISHOP, R. H. **Sistemas de controle modernos**. 12. ed. Rio de Janeiro, RJ: LTC, 2013.

DORMIDO, S. Control learning: present and future. **Annual Reviews in Control**, v. 28, n. 1, p. 115–136, 2004. ISSN 1367-5788. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1367578804000148>.

EATON, J. W. **GNU Octave**: about. 2019. Disponível em: <https://www.gnu.org/software/octave/about.html>. Acesso em: 12 de nov. de 2019.

ECMA INTERNATIONAL. **The JSON Data Interchange Format**. 2017. Disponível em: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Acesso em: 12 de nov. de 2019.

EJS Wiki. **What is EJS?** 2018. Disponível em: <https://www.um.es/fem/EjsWiki/Main/WhatIsEJS?> Acesso em: 01 de novembro de 2019.

FARIAS, G. *et al.* Developing networked control labs: a matlab and easy java simulations approach. **IEEE Transactions on Industrial Electronics**, v. 57, n. 10, p. 3266–3275, Oct 2010. ISSN 0278-0046. Disponível em: <https://ieeexplore.ieee.org/document/5409651>.

FIELDING, R. T.; RESCHKE, J. F. **Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing**. 2019. Disponível em: <https://tools.ietf.org/html/rfc7230>. Acesso em: 12 de nov. de 2019.

FRIEDLAND, B. **Control system design**: an introduction to state-space methods. Dover ed. New York, NY: McGraw-Hill, 1986.

GALAN, D. *et al.* Automated experiments on ejss laboratories. In: **2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)**. Madrid, Spain: IEEE, 2016. p. 78–85. Disponível em: <https://ieeexplore.ieee.org/document/7444444>.

GOMES, L.; BOGOSYAN, S. Current trends in remote laboratories. **IEEE Transactions on Industrial Electronics**, v. 56, n. 12, p. 4744–4756, Dec 2009. ISSN 0278-0046. Disponível em: <https://ieeexplore.ieee.org/document/5280206>.

GRAVIER, C. *et al.* State of the art about remote laboratories paradigms-foundations of ongoing mutations. **International Journal of Online and Biomedical Engineering**, v. 4, n. 1, p. 19–25, 2008. ISSN 2626-8493. Disponível em: <https://online-journals.org/index.php/i-joe/article/view/480>.

GUINALDO, M.; SÁNCHEZ, J.; DORMIDO, S. Diseño de un sistema de control anticipativo basado en paquetes para control en red. **Proc. Ninth Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI '10)**, 2010.

HAVIV, A. Q. **MEAN Web Development**. Birmingham: Packt Publishing Ltd, 2014.

HERADIO, R.; TORRE, L. de la; DORMIDO, S. Virtual and remote labs in control education: a survey. **Annual Reviews in Control**, v. 42, p. 1–10, 2016. ISSN 1367-5788. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1367578816300281>.

HERADIO, R. *et al.* Virtual and remote labs in education: a bibliometric analysis. **Computers & Education**, v. 98, p. 14–38, 2016. ISSN 0360-1315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360131516300677>.

IONESCU, C. M. *et al.* A remote laboratory as an innovative educational tool for practicing control engineering concepts. **IEEE Transactions on Education**, v. 56, n. 4, p. 436–442, 2013. ISSN 1557-9638. Disponível em: <https://ieeexplore.ieee.org/document/6484200>.

ISHUTKINA, M. *et al.* An internet based laboratory for control of a safety critical system. In: **2004 IEEE International Conference on Systems, Man and Cybernetics**. The Hague, Netherlands: IEEE, 2004. v. 3, p. 2707–2712. ISSN 1062-922X. Disponível em: <https://ieeexplore.ieee.org/document/1400740>.

ISHUTKINA, M. A. **Design and implementation of a supervisory safety controller for a 3DOF helicopter**. 2004. 80 f. Dissertação (mestrado) — Massachusetts Institute of Technology. Department of Aeronautics and Astronautics, Cambridge - MA, 2004. Disponível em: <https://dspace.mit.edu/handle/1721.1/27882>.

JAMES, K. **PC interfacing and data acquisition**: techniques for measurement, instrumentation and control. Oxford: Newnes, 2000.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT)**. 2015. Disponível em: <https://tools.ietf.org/html/rfc7519>. Acesso em: 12 de nov. de 2019.

KARAKASIDIS, T. Virtual and remote labs in higher education distance learning of physical and engineering sciences. In: **2013 IEEE Global Engineering Education Conference (EDUCON)**. Berlin, Germany: IEEE, 2013. p. 798–807. ISSN 2165-9559. Disponível em: <https://ieeexplore.ieee.org/abstract/document/6530198>.

KHEIR, N. *et al.* Control systems engineering education. **Automatica**, v. 32, n. 2, p. 147–166, 1996. ISSN 0005-1098. Disponível em: <https://www.sciencedirect.com/science/article/pii/0005109896855464>.

LEE, S.-H.; SONG, J.-B. Acceleration estimator for low-velocity and low-acceleration regions based on encoder position data. **IEEE/ASME Transactions on Mechatronics**, v. 6, n. 1, p. 58–64, March 2001. ISSN 1083-4435. Disponível em: <https://ieeexplore.ieee.org/document/914392>.

LEWIS, F. L. **Linear quadratic regulator (LQR) state feedback design**. 1998. Disponível em: <http://www.uta.edu/utari/acs/Lectures/lqr.pdf>. Acesso em: 22 de outubro de 2019.

LEWIS, F. L.; VRABIE, D. L.; SYRMOS, V. L. **Optimal control**. 3. ed. Hoboken, NJ: John Wiley & Sons, 2012.

LOPES, R. V. **Modelagem e controle preditivo de um helicóptero com três graus de liberdade**. 2007. 137 f. Dissertação (mestrado) — Instituto Tecnológico de Aeronáutica, São José dos Campos - SP, 2007.

MAIA, M. H. **Controle preditivo robusto de um helicóptero com três graus de liberdade sujeito a perturbações externas**. 2008. 117 f. Dissertação (mestrado) — Instituto Tecnológico de Aeronáutica, São José dos Campos - SP, 2008.

MANESCO, R. M. **Projeto de controladores robustos para sistemas sujeitos a falhas estruturais usando realimentação estática de saída**. 2013. 68 f. Dissertação (mestrado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2013. Disponível em: <http://hdl.handle.net/11449/87072>.

MARTINS, F. G. Tuning pid controllers using the itae criterion. **International Journal of Engineering Education**, v. 21, n. 5, p. 867–873, 2005. ISSN 0949-149X. Disponível em: <https://www.ijee.ie/articles/Vol21-5/Ijee1673.pdf>.

MATHWORKS. **Simulink reference**. Natick, MA: The MathWorks, Inc, 2016.

MATHWORKS. **Simulink user's guide**. Natick, MA: The MathWorks, Inc, 2016.

MATHWORKS. **goodnessOfFit**. 2019. Disponível em: <https://www.mathworks.com/help/ident/ref/goodnessoffit.html>. Acesso em: 12 de nov. de 2019.

MATHWORKS. **Object-Oriented Programming in MATLAB**. 2019. Disponível em: <https://www.mathworks.com/discovery/object-oriented-programming.html>. Acesso em: 12 de nov. de 2019.

MERRY, R.; MOLENGRAFT, M. van de; STEINBUCH, M. Velocity and acceleration estimation for optical incremental encoders. **Mechatronics**, v. 20, n. 1, p. 20–26, 2010. ISSN 0957-4158. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957415809001214>.

MOLLOY, D. **Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux**. Indianapolis: John Wiley & Sons, 2014.

MONTEZUMA, M. A. F. **Modelagem e controle de posição e orientação de uma plataforma de stewart**. 2003. 159 f. Dissertação (mestrado) — Universidade de São Paulo. Escola de Engenharia de São Carlos, São Carlos - SP, 2003. Disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18135/tde-20052016-101448/pt-br.php>.

MOZILLA DEVELOPER NETWORK. **JSON**. 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON>. Acesso em: 12 de nov. de 2019.

NATIONAL INSTRUMENTS. **NI 660X specifications**. 2009. Disponível em: <http://www.ni.com/pdf/manuals/372141b.pdf>. Acesso em: 16 de outubro de 2019.

NATIONAL INSTRUMENTS. **NI 6251**. 2016. Disponível em: <http://www.ni.com/pdf/manuals/375213c.pdf>. Acesso em: 16 de outubro de 2019.

NATIONAL INSTRUMENTS. **O que é aquisição de dados?** 2019. Disponível em: <https://www.ni.com/data-acquisition/what-is/pt/>. Acesso em: 16 de outubro de 2019.

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

NUNES, M. A. d. A.; SILVA, R. d. C. **MSC ADAMS**: guia prático de uiilização. São Paulo, SP: Edgard Blücher, 2014.

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

OLIVEIRA, A. C. B.; ANGÉLICO, B. A. Use of a rigorous model in the control design of a 3-d.o.f. helicopter. In: **XXII Congresso Brasileiro de Automática**. João Pessoa, Brasil: XXII Congresso Brasileiro de Automática, 2018. ISSN 2525-8311. Disponível em: <https://plataforma.swge.com.br/PROCEEDINGS/>.

PAULA, A. L. A. d. **Detecção e acomodação de falhas em sistemas incertos com atraso no sinal de controle utilizando modo deslizante**. 2011. 89 f. Dissertação (mestrado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2011. Disponível em: <http://hdl.handle.net/11449/87149>.

PEREYRA, H. E. S. **Detecção robusta de falhas através de filtros H 'infinito': implementação prática em um helicóptero 3-DOF de bancada e em um servosistema**. 2013. 72 f. Dissertação (mestrado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2013. Disponível em: <http://hdl.handle.net/11449/87076>.

PTERNEAS, V. **Getting Started with HTML5 WebSocket Programming**. Birmingham: Packt Publishing Ltd, 2013.

QUANSER. **3 DOF Helicopter**: simulink courseware. 2017. Disponível em: <https://www.quanser.com/products/3-dof-helicopter/>. Acesso em: 03 de outubro de 2019.

RICHARDSON, L.; RUBY, S. **RESTful Serviços Web**. Rio de Janeiro, RJ: Alta Books, 2007.

RODRIGUES, F. B. **Multicontroladores para melhoria da robustez e acomodação de falhas em sistemas**. 2009. 85 f. Dissertação (mestrado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2009. Disponível em: <http://hdl.handle.net/11449/87137>.

SACHS, J. **How to estimate encoder velocity without making stupid mistakes**: part ii (tracking loops and plls). 2013. Disponível em: <https://www.embeddedrelated.com/showarticle/530.php>. Acesso em: 16 de outubro de 2019.

SÁENZ, J. *et al.* Open and low-cost virtual and remote labs on control engineering. **IEEE Access**, v. 3, p. 805–814, 2015. ISSN 2169-3536. Disponível em: <https://ieeexplore.ieee.org/document/7119567>.

SALZMANN, C.; GILLET, D. Smart device paradigm, standardization for online labs. In: **Global Engineering Education Conference (EDUCON), 2013 IEEE**. Berlin, Germany: IEEE, 2013. p. 1217–1221. ISSN 2165-9559. Disponível em: <https://ieeexplore.ieee.org/abstract/document/6530261>.

SCHAF, F. M. **Arquitetura para ambiente de ensino de controle e automação utilizando experimentos remotos de realidade mista**. 2006. 206 f. Dissertação (mestrado) — Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Programa de Pós-Graduação em Engenharia Elétrica, RS, 2006. Disponível em: <https://lume.ufrgs.br/handle/10183/10320>.

SEBORG, D. E. *et al.* **Process dynamics and controle**. 3. ed. Hoboken, NJ: John Wiley & Sons, 2011.

SHABANA, A. A. **Dynamics of multibody systems**. 4. ed. New York, NY: Cambridge University Press, 2013.

SHAN, J.; LIU, H. T.; NOWOTNY, S. Synchronised trajectory-tracking control of multiple 3-dof experimental helicopters. **IEE Proceedings - Control Theory and Applications**, v. 152, n. 6, p. 683–692, Nov 2005. Disponível em: <https://ieeexplore.ieee.org/document/1512679>.

SHIMADA, B. M. **Identificação, instrumentação e controle de uma aeronave de duas hélices paralelas com atribuição de autoestrutura completa**. 2015. 129 f. Dissertação (mestrado) — Universidade Tecnológica Federal do Parará (UTFPR), Cornélio Procópio - PR, 2015. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/3337>.

SHUTTERSTOCK INC. **Rickshaw**. 2019. Disponível em: <https://tech.shutterstock.com/rickshaw/>. Acesso em: 12 de nov. de 2019.

SILVA, J. C. d. **Desenho técnico auxiliado pelo solidworks**. Florianópolis, SC: Visual Books, 2011.

SILVA, J. H. P. **Controle robusto h-infinito chaveado para sistemas lineares**. 2013. 69 f. Dissertação (mestrado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2013. Disponível em: <http://hdl.handle.net/11449/87077>.

SILVA, J. L. e. **Projeto de controle robusto para acomodação de falhas no módulo do helicóptero 3-DOF**. 2011. 93 f. Dissertação (mestrado) — Universidade Estadual Paulista Júlio de Mesquita Filho. Faculdade de Engenharia de Ilha Solteira, Ilha Solteira - SP, 2011. Disponível em: <http://hdl.handle.net/11449/87099>.

SOUZA, A. C. d. *et al.* **SolidWorks 2003**: modelagem em 3d. Florianópolis, SC: Visual Books, 2003.

STMICROELECTRONICS. **STM32F767ZI**. 2019. Disponível em: <https://www.st.com/en/microcontrollers-microprocessors/stm32f767zi.html>. Acesso em: 12 de nov. de 2019.

STMICROELECTRONICS. **NUCLEO-F767ZI**. 2020. Disponível em: <https://www.st.com/en/evaluation-tools/nucleo-f767zi.html>. Acesso em: 03 de mar. de 2020.

TILKOV, S.; VINOSKI, S. Node.js: using javascript to build high-performance network programs. **IEEE Internet Computing**, v. 14, n. 6, p. 80–83, Nov 2010. ISSN 1089-7801. Disponível em: <https://ieeexplore.ieee.org/document/5617064>.

TILLI, A.; MONTANARI, M. A low-noise estimator of angular speed and acceleration from shaft encoder measurements. **Automatika**, v. 42, n. 3-4, p. 169–176, 2001. ISSN 0005-1144. Disponível em: <https://hrcak.srce.hr/6622>.

VARGAS, H. *et al.* Web-enabled remote scientific environments. **Computing in Science Engineering**, v. 11, n. 3, p. 36–46, May 2009. ISSN 1521-9615. Disponível em: <https://ieeexplore.ieee.org/document/4814981>.

VARGAS, H. *et al.* A systematic two-layer approach to develop web-based experimentation environments for control engineering education. **Intelligent Automation & Soft Computing**, Taylor & Francis, v. 14, n. 4, p. 505–524, 2008.

VARGAS, H. *et al.* A network of automatic control web-based laboratories. **IEEE Transactions on Learning Technologies**, v. 4, n. 3, p. 197–208, July 2011. ISSN 1939-1382. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5654493>.

XTERM. **Xterm.js**. 2019. Disponível em: <https://xtermjs.org/>. Acesso em: 12 de nov. de 2019.

YANG, X.; ZHENG, X. Adaptive nn backstepping control design for a 3-dof helicopter: Theory and experiments. **IEEE Transactions on Industrial Electronics**, p. 1–1, 2019. ISSN 0278-0046. Disponível em: <https://ieeexplore.ieee.org/document/8734872>.

YOUNG, P. C.; WILLEMS, J. C. An approach to the linear multivariable servomechanism problem†. **International Journal of Control**, v. 15, n. 5, p. 961–979, 1972. ISSN 0020-7179. Disponível em: <https://www.tandfonline.com/doi/abs/10.1080/00207177208932211>.

ZACHARIA, Z. C. Comparing and combining real and virtual experimentation: an effort to enhance students' conceptual understanding of electric circuits. **Journal of Computer Assisted Learning**, v. 23, n. 2, p. 120–132, Jan 2007. ISSN 1365-2729. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2729.2006.00215.x>.

ZHENG, B.; ZHONG, Y. Robust attitude regulation of a 3-dof helicopter benchmark: theory and experiments. **IEEE Transactions on Industrial Electronics**, v. 58, n. 2, p. 660–670, Feb 2011. ISSN 0278-0046. Disponível em: <https://ieeexplore.ieee.org/document/5439880>.

ZHOU, Y. *et al.* Robust synchronisation of second-order multi-agent system via pinning control. **IET Control Theory Applications**, v. 9, n. 5, p. 775–783, 2015. Disponível em: <https://ieeexplore.ieee.org/document/7070581>.