UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UMBERTO XAVIER DA SILVA NETO

CONSTRUÇÃO E CONTROLE DE UM MODELO HELICÓPTERO TANDEM

DISSERTAÇÃO

CORNÉLIO PROCÓPIO
2021

UMBERTO XAVIER DA SILVA NETO

CONSTRUÇÃO E CONTROLE DE UM MODELO HELICÓPTERO TANDEM

Construction and Control of a Tandem Helicopter Model

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná – Campus Cornélio Procópio, como requisito para obtenção do título de “Mestre em Engenharia Elétrica”. Área de Concentração: Sistemas Eletrônicos Industriais.

Orientador: Prof. Dr. Alessandro do Nascimento Vargas.

CORNÉLIO PROCÓPIO

2021

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio

UMBERTO XAVIER DA SILVA NETO

CONSTRUÇÃO E CONTROLE DE UM MODELO HELICÓPTERO TANDEM

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Sistemas Eletrônicos Industriais.

Data de aprovação: 29 de Outubro de 2021

Prof Alessandro Do Nascimento Vargas, Doutorado - Universidade Tecnológica Federal do Paraná

Prof.a Maira Martins Da Silva, Doutorado - Escola de Engenharia de São Carlos (Eesc) - Universidade de São Paulo (Usp)

Prof Marcio Aurelio Furtado Montezuma, Doutorado - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 29/10/2021.

Dedico este trabalho aos meus pais Umberto Xavier da Silva Júnior e Mafalda Albertin Xavier da Silva (*in memoriam*).

AGRADECIMENTOS

Dedico este trabalho à instituição de ensino UTFPR, que desde o ensino técnico em 2010, passando pela graduação, e até hoje no mestrado, fornece o ensino, material e condições para a realização pessoal e profissional.

Aos meus colegas do laboratório LaSisC, Matheus e Eduardo, no auxílio e ensinamentos para o desenvolvimento do trabalho.

Ao meu orientador, Alessandro do Nascimento Vargas, por cada etapa concluída no mestrado.

Ao meu supervisor, Marcio Aurelio Furtado Montezuma, pelo apoio e fornecimento e suporte de materiais essenciais para o desenvolvimento deste projeto.

Ao meu pai, Umberto, por cada palavra de apoio e pela oportunidade de me permitir cursar uma pós-graduação.

RESUMO

SILVA NETO, Umberto Xavier da. **Construção e Controle de um Modelo Helicóptero Tandem**. 2021. Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2021.

Esta dissertação descreve o processo de desenvolvimento de um protótipo para estudos e controle de um Helicóptero Tandem de 3 graus de liberdade. O texto apresenta a modelagem do processo realizada por prototipagem virtual com auxílio de softwares CAD e CAE; o detalhamento físico da planta real; o processo experimental de identificação dos atuadores da planta; e a aplicação do Tracking Loop, técnica para estimativa da velocidade angular a partir de sinais digitais de posição angular obtidos de encoders. A interface entre instrumentação e comando é realizada por meio de placas de aquisição de dados com intermediação de um software interativo numérico. O controle da planta é realizado por um controlador seguidor com ganhos calculados por meio de realimentação de estados com atribuição de autoestrutura completa. Os resultados são apresentados de forma comparativa entre a simulação linear, simulação não linear e ensaio experimental, destacando e justificando as divergências.

Palavras-chave: Tandem; Prototipagem Virtual; Controle Moderno; Atribuição de Autoestrutura Completa.

ABSTRACT

SILVA NETO, Umberto Xavier da. **Construction and Control of a Tandem Helicopter Model**. 2021. Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2021.

This dissertation describes the development of a prototype for studies and control of a Tandem Helicopter with 3 degrees of freedom. The text presents the process modeling, performing it in virtual prototyping with the aid of CAD and CAE softwares; the physical detailing of the real plant; the experimental identification process of the plant actuators; and the Tracking Loop application, technique for estimating angular velocity from digital signals of angular position obtained from encoders. The interface between instrumentation and command is performed through data acquisition boards with the intermediation of interactive numerical software. The plant control is designed by a tracking system where the gains are given by state feedback with the entire eigenstructure assignment, comparing the linear simulation, non-linear simulation, and experimental results, showing and justifying divergences.

Keywords: Tandem; Virtual Prototyping; Modern Control; Entire Eigenstructure Assignment.

LISTA DE ILUSTRAÇÕES

Figura 1.1 – Classificação das aeronaves.....18

Figura 1.2 – Boeing Chinook.....19

Figura 1.3 – Helicóptero 3 GDL da Quanser.....21

Figura 1.4 – 3 GDL referência deste trabalho.....22

Figura 2.1 – Diagrama de Blocos do Controle Seguidor com realimentação de estados.....31

Figura 3.1 – Diagrama de corpo livre do Helicóptero de 3 GDL 32

Figura 3.2 – Os 3 movimentos em um Helicóptero Tandem (HC-1B Chinook) 33

Figura 3.3 – Planta Construída. É possível identificar as 5 partes: base, haste, braço, aeronave e contrapeso.....34

Figura 3.4 – Desenho Técnico da Base.....35

Figura 3.5 – Desenho Técnico da Haste 36

Figura 3.6 – Desenho Técnico do Braço 37

Figura 3.7 – Desenho Técnico da Aeronave 39

Figura 3.8 – Desenho Técnico do Contrapeso.....40

Figura 4.1 – Base no SolidWorks.....44

Figura 4.2 – Haste no SolidWorks.....45

Figura 4.3 – Braço no SolidWorks.....45

Figura 4.4 – Aeronave no SolidWorks.....46

Figura 4.5 – Contrapeso no SolidWorks.....46

Figura 4.6 – Montagem da planta virtual no SolidWorks.....47

Figura 4.7 – Orientação do centro de massa de cada corpo rígido do Helicóptero de 3GDL.....49

Figura 4.8 – Junta fixa da base com o solo (azul); Junta de revolução de deslocamento (verde); Junta de revolução de elevação (magenta); Junta fixa do braço com comtrapeso (branco); Junta de revolução de arfagem (amarelo).....50

Figura 4.9 – Análise estática (esquerda); Posição inicial de elevação (direita).....51

Figura 5.1 – Processo para identificação dos polinômios dos atuadores.....55

Figura 5.2 – Processo de calibração da célula de carga em laboratório.....56

Figura 5.3 – Sinais aquisitados para uma frequência de 100 $\boldsymbol{Hz}$.....57

Figura 5.4 – Sinais aquisitados para uma frequência de 800 $\boldsymbol{Hz}$.....58

Figura 5.5 – Curva de calibração da célula de carga.....59

Figura 5.6 – Processo para identificação dos polinômios dos atuadores.....60

Figura 5.7 – Progressão da razão cíclica para identificação dos atuadores 61

Figura 5.8 – Ajustes polinomiais dos motores dianteiro e traseiro.....61

Figura 5.9 – Desvio padrão das curvas de calibração dos motores dianteiro e traseiro.....62

Figura 5.10 – Processo de identificação dos motores em laboratório.....63

Figura 6.1 – Funcionamento do encoder óptico incremental.....64

Figura 6.2 – Encoders dispostos no Helicóptero de 3 Graus de Liberdade.....65

Figura 6.3 – Tracking Loop com implemetação através de um sistema realimentado com um controle PI e integrador na saída 66

Figura 6.4 – Experimento para validação doTracking Loop.....68

Figura 6.5 – Variação no movimento de arfagem e deslocamento.....70

Figura 6.6 – Experimentos para o encoder de arfagem.....70

Figura 6.7 – Experimentos para o encoder de deslocamento..... 71

Figura 6.8 – Experimento 12 para arfagem e deslocamento ($\boldsymbol{K_p = 40, K_i = 900}$)..... 72

Figura 7.1 – Diagrama de Blocos do Controle Seguidor para os experimentos.....74

Figura 7.2 – Bloco para simulação linear.....74

Figura 7.3 – Elementos dentro do bloco do *ADAMS*.....75

Figura 7.4 – Simulação não linear. $\boldsymbol{t}$ = 30,11 $\boldsymbol{s}$ (esquerda). $\boldsymbol{t}$ = 41,11 $\boldsymbol{s}$ (direita).....75

Figura 7.5 – Diagrama para experimento da planta real.....76

Figura 7.6 – Aquisição dos dados dos encoders e Tracking Loop.....77

Figura 7.7 – Tratamento dos sinais de controle.....77

Figura 7.8 – Ensaio Real. t = 31,5 s (esquerda). t = 40,22 s (direita).....78

Figura 7.9 – Posição e velocidade angular de elevação do Experimento 1 79

Figura 7.10 – Empuxo Dianteiro e Traseiro do Experimento 1.....80

Figura 7.11 – Posição e velocidade angular de deslocamento do Experimento 2 81

Figura 7.12 – Posição e velocidade angular de elevação do Experimento 2 81

Figura 7.13 – Posição e velocidade angular de arfagem do Experimento 2.....82

Figura 7.14 – Empuxo Dianteiro e Traseiro do Experimento 2.....82

Figura 7.15 – Posição e velocidade angular de deslocamento do Experimento 3 83

Figura 7.16 – Posição e velocidade angular de elevação do Experimento 3 84

Figura 7.17 – Posição e velocidade angular de arfagem do Experimento 3.....84

Figura 7.18 – Empuxo Dianteiro e Traseiro do Experimento 3 85

Figura 7.19 – Posição e velocidade angular de deslocamento do Experimento 4 86

Figura 7.20 – Posição e velocidade angular de elevação do Experimento 4 86

Figura 7.17 – Posição e velocidade angular de arfagem do Experimento 4.....87

Figura 7.18 – Empuxo Dianteiro e Traseiro do Experimento 4.....87

Figura 7.20 – Peso para compensar posição angular de arfagem.....89

Figura 7.21 – Geometria do contrapeso complicando posicionamento.....90

Figura 7.22 – Cabos da Haste.....90

LISTA DE TABELAS

Tabela 3.1 – Itens da Base.....35

Tabela 3.2 – Itens da Haste.....36

Tabela 3.3 – Itens do Braço.....38

Tabela 3.4 – Itens da Aeronave.....39

Tabela 3.5 – Itens do Contrapeso 40

Tabela 4.1 – Propriedades dos corpos no SolidWorks.....47

Tabela 4.2 – Orientação dos corpos nos softwares.....48

Tabela 4.3 – Distâncias definidas.....51

Tabela 5.1 – Média dos pontos de calibração para cada experimento 59

Tabela 5.2 – Dados da dispersão entre curvas 62

Tabela 6.1 – Características dos encoders da planta do Helicóptero de 3 Graus de Liberdade.....66

Tabela 6.2 – Experimentos com diferentes tipos de ganhos $\boldsymbol{K_p}$e $\boldsymbol{K_i}$.....69

LISTA DE SÍMBOLOS

$\mathbf{0}$	Matriz de zeros.
$\boldsymbol{A}$	Matriz de estados de um sistema invariante no tempo.
$\boldsymbol{A(t)}$	Matriz de estados de um sistema variante no tempo.
$\overline{\boldsymbol{A}}$	Matriz de estados aumentada.
$\boldsymbol{A_{cl}}$	Matriz de estados do sistema em malha fechada.
$\boldsymbol{A'_{cl}}$	Matriz de estados do sistema aumentado de um controle seguidor em malha fechada.
$\mathbb{A}$	Matriz diagonal.
$\boldsymbol{B}$	Matriz de entrada de um sistema invariante no tempo.
$\boldsymbol{B(t)}$	Matriz de entrada de um sistema variante no tempo.
$\overline{\boldsymbol{B}}$	Matriz de entrada aumentada.
$\overline{\boldsymbol{B}}'$	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.
$\overline{\boldsymbol{C}}$	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 de variáveis controladas.
$\boldsymbol{e(t)}$	Sinal de erro de uma realimentação.
e_p	Erro da posição angular.
$e^{\boldsymbol{At}}$	Matriz de transição de estados.
$\boldsymbol{F}$	Matriz de seleção de variáveis não controladas.
F_t	Força de empuxo do motor traseiro.
F_d	Força de empuxo do motor dianteiro.
$\boldsymbol{I}$	Matriz Identidade.
$\boldsymbol{K}$	Matriz de ganhos de realimentação de estado.

$\bar{\boldsymbol{K}}$	Matriz de ganhos de realimentação de estado do sistema aumentado.
$\boldsymbol{K_1}$	Matriz de ganho do controle seguidor.
$\boldsymbol{K_2}$	Matriz de ganho do controle seguidor relacionada a integração de erros dos estados.
$\boldsymbol{k_p}$	Ganho proporcional.
$\boldsymbol{k_i}$	Ganho integral.
$\boldsymbol{l}$	Número de variáveis de saída.
L_a	Distância do centro dos motores até o eixo de 2 GDL.
L_c	Distância do eixo de 2 GDL até o contrapeso.
L_h	Distância do eixo de arfagem até o centro dos motores.
$\boldsymbol{m}$	Número de variáveis de controle.
$\boldsymbol{M_c}$	Matriz de controlabilidade.
M_t	Massa do motor traseiro.
M_d	Massa do motor dianteiro.
M_h	Massa da haste do braço.
M_c	Massa do contrapeso.
$\boldsymbol{n}$	Número de variáveis de estado.
$\boldsymbol{p}$	Número de varáveis de saída controláveis.
p_e	Posição angular real.
$\hat{p}_e$	Posição angular estimada.
$\boldsymbol{q}$	Matriz dos autovetores $\boldsymbol{q_i}$.
$Q(\lambda)$	Raízes da equação característica.
$\boldsymbol{q_i}$	Autovetores de seleção.
$\boldsymbol{r(t)}$	Vetor de referência.
$\boldsymbol{r_\lambda}$	Referência de deslocamento
$\boldsymbol{r_\varepsilon}$	Referência de elevação
$S(\lambda_i)$	Espaço Nulo.

s	Variável complexa.
t	Instante de tempo.
$\boldsymbol{T}$	Matriz modal.
t_0	Instante de tempo inicial.
$u(t)$	Sinal de controle.
$\boldsymbol{u_{op}}$	Força de operação para linearização.
$\boldsymbol{v}$	Matriz de autovetores associados $\boldsymbol{v_i}$.
$\boldsymbol{v_i}$	Autovetor associado.
$\hat{v}_e$	Velocidade angular estimada.
$\hat{v}_{e2}$	Velocidade angular estimada dada uma integração.
$\boldsymbol{x}(t)$	Vetor de estado.
$\dot{\boldsymbol{x}}(t)$	Primeira derivada temporal do vetor de estado.
$\boldsymbol{y}(t)$	Vetor de saída.
y_c	Polinômio da célula de carga.
y_d	Polinômio do motor dianteiro.
y_t	Polinômio do motor traseiro.
$\boldsymbol{w_i}$	Autovetores recíprocos
λ_i	Autovalor.
$\boldsymbol{\sigma}$	Espectro de autovalores.
ρ	Arfagem.
ε	Elevação.
λ	Deslocamento.

LISTA DE SIGLAS E ABREVIATURAS

ADAMS	Advanced Dynamic Analysis of Mechanical Systems
CAD	Computed Aided Design
CAE	Computed-aided Engineering
GDL	Graus de Liberdade
HTA	Heavier Than Air
LTA	Lighter Than Air
LQR	Linear Quadratic Regulator
LVR	Laboratório Virtual e Remoto
MBS	Multibody Systems
MIMO	Multiple Inputs – Multiple Outputs
PID	Proporcional, Integral e Derivativo
PWM	Pulse Width Modulation
SISO	Single Input – Single Output
VTOL	Vertical Take Off and Landing

SUMÁRIO

1 INTRODUÇÃO 18

1.1 REVISÃO DO CONTROLE EM MODELO HELICÓPTERO TANDEM 20

1.2 OBJETIVOS 23

1.2.1 Objetivos específicos 23

2 PROJETO DE CONTROLE VIA AUTOESTRUTURA 24

2.1 ATRIBUIÇÃO DE AUTOESTRUTURA COMPLETA 25

2.1.1 Controle Seguidor com Realimentação de Estado 29

3 CONSTRUÇÃO E INSTRUMENTAÇÃO DA PLANTA MODELO HELICÓPTERO TANDEM 32

4 MODELAGEM DE SISTEMAS MULTICORPOS 41

4.1 PROTOTIPAGEM VIRTUAL DO SISTEMA 44

5 IDENTIFICAÇÃO DOS ATUADORES 55

5.1 IDENTIFICAÇÃO EMPUXO 60

6 SENSORES E ESTIMAÇÃO 64

6.1 SENSORES UTILIZADOS 65

6.2 TRACKING LOOP 66

6.3 EXPERIMENTO PARA VALIDAÇÃO 67

7 RESULTADOS EXPERIMENTAIS 73

7.1 SIMULAÇÃO LINEAR 74

7.2 SIMULAÇÃO NÃO LINEAR 75

7.3 PLANTA REAL 76

7.4 RESULTADOS EXPERIMENTAIS: CONTROLE 78

7.4.1 Experimento 1 79

7.4.2 Experimento 2 80

7.4.3 Experimento 3 83

7.4.4 Experimento 4 85

7.5 DISCUSSÃO CONSIDERAÇÕES DA CONSTRUÇÃO 88

8 CONCLUSÕES 92

8.1 TRABALHOS FUTUROS 92

REFERÊNCIAS 94

1 INTRODUÇÃO

Segundo Bouabdallah (2007), as aeronaves podem ser divididas em duas categorias principais: mais leves que o ar (LTA, Lighter Than Air) e mais pesadas que o ar (HTA, Heavier Than Air). A Figura 1.1 apresenta a classificação das aeronaves em relação ao princípio de voo e ao modo de propulsão, variando entre não motorizadas e motorizadas.

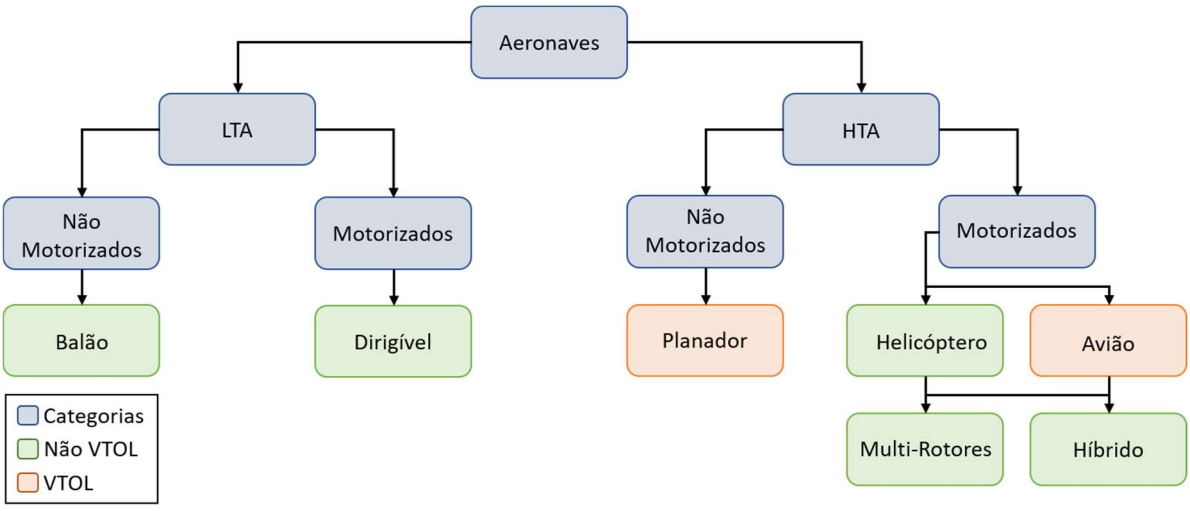


Figura 1.1 – Classificação das Aeronaves
Fonte: Adaptado de Paula (2012)

Durante vistorias de grandes áreas, os aviões mantem uma velocidade específica, denominada de velocidade de cruzerio. Porém, por possuírem propulsores na vertical, necessitam de uma velocidade mínima de sustentação produzida pela diferença de pressão sob suas asas. Essa necessidade de velocidade mínima dificulta vistorias mais detalhadas, limitando o alcance de sua operação (COSTA, 2012).

Entrando na categoria de veículos de pouso e decolagem vertical (do inglês Vertical Take Off and Landing, VTOL), pode-se citar primeiramente os balões e dirigíveis autônomos, que possuem autonomia de várias horas e conseguem manter velocidades mínimas, sendo aplicável em vistorias mais detalhadas. Porém, esses veículos são suscetíveis às irregularidades climáticas (COSTA, 2012; PAULA, 2012).

Ainda na categoria VTOL, têm-se as aeronaves menos suscetíveis a variações climáticas. Tais aeronaves, como os helicópteros e multi-rotores, fornecem grande manobrabilidade e precisão, com autonomia entre dezenas de minutos a

algumas horas. A vantagem dessas aeronaves é a precisão de navegação (COSTA, 2012). A diferença entre helicópteros e multi-rotores é a ausência do rotor de cauda no segundo, dependendo apenas das velocidades individuais das hélices para sua locomoção (BRAGA E SANTANA, 2008; PAULA, 2012). A habilidade do helicóptero planar permite com que ele pouse em qualquer lugar onde há um espaço seguro, lembrando que há também os helicópteros anfíbios, que podem pousar tanto na água quanto em solo firme (WATKINSON, 2004).

O helicóptero modelo Tandem, objeto de estudo deste trabalho, tem dois rotores girando em direções opostas e dispostos nas extremidades de um casco longo. Os rotores geralmente são sincronizados através de um sistema de transmissão. Um exemplo deste tipo de helicóptero é o Chinook, visto na Figura 1.2 (WATKINSON, 2004).

Figura 1.2 – Boeing Chinook
Fonte: (Creative Common)

A principal contribuição desse trabalho é construir um protótipo experimental completo de helicóptero Tandem com 3 graus de liberdade. O protótipo foi projetado e construído a partir do zero, conforme detalhado adiante.

1.1 REVISÃO DO CONTROLE EM MODELO HELICÓPTERO TANDEM

A representação do helicóptero Tandem via sistemas dinâmicos depende diretamente das considerações, restrições e objetivos da pesquisa associada. Por exemplo, quando deseja-se aprofundar na dinâmica do helicóptero Tandem, mantém-se na modelagem considerações físicas fidedignas ao modelo real. Dzul et al. (2002), por exemplo, apresenta a simulação de um controle Lyapunov utilizando técnicas backstepping para um modelo dinâmico de um helicóptero Tandem em condições de voo planado. Nesta modelagem é considerado que os dois rotores são sobrepostos em cerca de 20% do raio do rotor, necessitando elevar o rotor traseiro para minimizar a interferência aerodinâmica criada por este. Os movimentos do helicóptero em planeio dependem da intensidade de empuxo dos rotores e da inclinação lateral de cada um, representando o funcionamento real da aeronave.

Quando o foco da pesquisa é o controle do sistema, pode-se considerar simplificações no modelo, facilitando o uso de protótipos em testes. A Figura 1.3 apresenta a planta kit didático de um Helicóptero de 3 GDL (Graus de Liberdade) desenvolvida pela Quanser (Markham, Canadá). Esse kit didático permite estudos de métodos de controle porque apresenta muitas peculiaridades de cunho prático, tal como dinâmica com parâmetros não modeláveis, não linearidades, incertezas, acoplamentos, perturbações e ruídos na medição. Note que o Helicóptero de 3 GDL é uma plataforma experimental desafiadora em termos de avaliação de técnicas de controle (Yu et al., 2015). Aqui neste trabalho, apresentamos o detalhamento completo para a construção e implementação de um kit protótipo Helicóptero de 3 GDL, semelhante àquele construído pela Quanser. Tal construção e implementação mostra-se como a principal contribuição dessa dissertação.

Há diversos trabalhos que estudam o Helicóptero de 3 GDL. Por exemplo, Karaman et al. (2015) emprega 3 controladores PID Fuzzy para controlar os 3 graus de liberdade da planta. Esse tipo de controlador proporciona soluções efetivas no

controle de plantas complexas e com incertezas paramétricas. As entradas da malha de controle são os erros de posição e velocidade angular dos movimentos, e as saídas dos controladores são as tensões aplicadas diretamente nos motores. A planta é representada por meio de uma modelagem matemática não linear. Para a validação do controle, simulou-se e comparou-se os resultados com um controlador LQR (Linear Quadratic Regulator).

Figura 1.3 – Helicóptero 3 GDL da Quanser
Fonte: Quanser (2017)

Outras abordagens usando a planta da Quanser são: Controlador adaptativo por referência de modelo, na qual aplica-se em plantas com dinâmicas incertas e desconhecidas (KOCAGIL et al., 2018); Um controlador preditivo robusto, onde as incertezas são dadas por erros no modelo e perturbações externas (MAIA, 2008); Controle robusto por H_∞ também aplicando incertezas do modelo (POSTLETHWAITE et al., 2005).

Em Shimada (2015) é apresentada uma abordagem baseada em técnicas de modelagem multicorpos, facilitando o desenvolvimento da modelagem matemática de um Helicóptero de 3 GDL baseado na planta da Quanser. Um protótipo virtual desenvolvido por meio de softwares CAD e CAE foi proposto baseado em uma planta real, apresentando um modelo linear e não linear, na qual o procedimento adotado para identificação das características físicas do modelo real foi por meio de um cavalete desenvolvido por Montezuma (2010), utilizando o método de reações estáticas, meio na qual coloca-se os corpos do sistema em suspensão para a medição

de sua reação. Este método depende de alguns ensaios experimentais e ainda necessita de certas técnicas matemáticas. Para a validação do protótipo foi proposto um controle seguidor por meio da atribuição de autoestrutura completa, comparando os resultados por meio de simulações e ensaios reais.

Kaneko (2020) utilizou a mesma planta de Shimada (2015) para a elaboração de um laboratório virtual e remoto (LVR), aplicando melhorias em alguns processos, como a anulação do efeito binário da planta, a aplicação da técnica de Tracking Loop para a estimação das velocidades angulares do sistema e a consideração das massas do sistema de instrumentação. Para a identificação das características físicas foi usado a densidade específica de cada peça do protótipo real como entrada ao software CAD, técnica apresentada em Malaquias (2017) e Mollon (2020). O software CAD facilita a descoberta do modelo linear, porém pode gerar um erro de identificação, pois negligencia as características físicas não lineares das peças. Por fim, projetou-se um controle seguidor com e sem observador de Luenberger, aplicando o método de atribuição de autoestrutura completa e LQR. Os resultados foram comparados em ambiente de simulação, real e remoto.

A Figura 1.4 mostra o protótipo virtual do Helicóptero de 3 GDL desenvolvido por Shimada (2015) e Kaneko (2020).

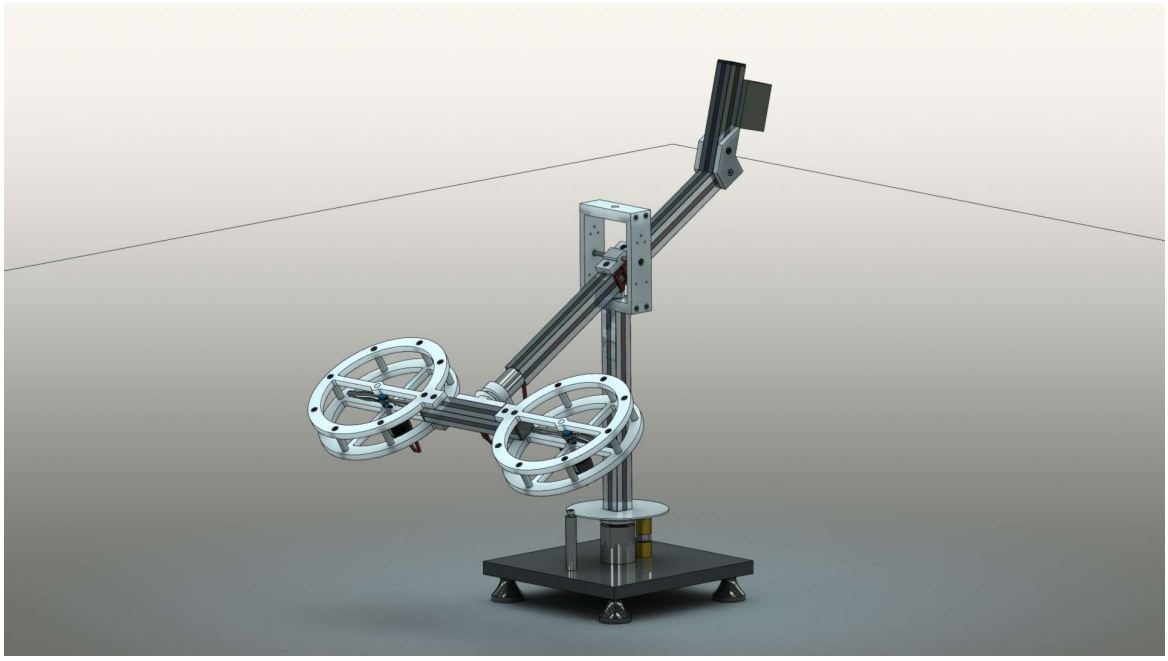


Figura 1.4 – 3 GDL referência deste trabalho
Fonte: Disponibilizado pelos autores

1.2 OBJETIVOS

Este trabalho tem dois objetivos principais:

1. Apresentar o detalhamento completo de construção de um protótipo experimental de helicóptero Tandem com 3 graus de liberdade;
2. Apresentar um projeto de controle para o protótipo experimental de helicóptero Tandem.

1.2.1 Objetivos específicos

- Construir um protótipo virtual, utilizando uma técnica de modelagem de sistemas multicorpos, conhecida como prototipagem virtual;
- Obter o modelo matemático linear e não linear do protótipo virtual e realizar simulações para estes modelos;
- Desenvolver um controle seguidor pelo método de realimentação de estados com atribuição de autoestrutura completa;
- Simular e comparar o resultado dos controladores através dos modelos extraídos;
- Construir a planta do helicóptero e instrumentá-la;
- Realizar a identificação dos atuadores;
- Aplicar a técnica Tracking Loop nos sinais dos encoders;
- Instrumentar a planta real através de um sistema de aquisição de dados e controle;
- Analisar as características da resposta temporal da planta mais o sistema de controle sob diversos tipos de entrada.

O trabalho desenvolvido neste documento reproduz partes do trabalho de Kaneko (2020) e Shimada (2015), utilizando a técnica de modelagem de sistemas multicorpos (MBS), criando um protótipo virtual para desenvolver um controle seguidor com realimentação de estados a partir da atribuição de autoestrutura completa, validando o processo em uma nova planta real de um helicóptero de 3 GDL.

2 PROJETO DE CONTROLE: USANDO ATRIBUIÇÃO DE AUTO ESTRUTURA

O objetivo desse capítulo é relembrar o projeto de sistemas de controle sob a técnica de autoestrutura. Para isso, introduz-se alguns conceitos básicos sobre a teoria de sistemas lineares.

O estado de um sistema é um conjunto de n variáveis $x_1(t), \dots, x_n(t)$, tal que os valores iniciais $x_i(t_o)$ deste conjunto e as entradas $u_j(t)$ são suficientes para descrever unicamente a resposta futura do sistema, onde $t \geq t_o$. O conjunto de variáveis de estado $x_i(t)$ representa os elementos/componentes de um vetor de estados $\boldsymbol{x}(t)$ de n dimensão, e o conjunto de entradas $u_j(t)$ contém os elementos do vetor de entradas $\boldsymbol{u}(t)$ de m dimensão, onde (D'AZZO; HOUPIS, 1995):

$$\boldsymbol{x} = \boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} \quad ; \quad \boldsymbol{u} = \boldsymbol{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \tag{2.1}$$

As equações de estado diferenciais podem ser definidas em formato matricial como (D'AZZO; HOUPIS, 1995):

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{u} \tag{2.2}$$

E a equação de saída como (D'AZZO; HOUPIS, 1995):

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \tag{2.3}$$

Assim podemos definir cada componente da equação de estado e saída como:

- $\boldsymbol{x}(t)$ = vetor de estado com dimensão $n \times 1$;
- $\dot{\boldsymbol{x}}(t)$ = derivada do vetor de estado com dimensão $n \times 1$;
- $\boldsymbol{u}(t)$ = vetor de entradas com dimensão $m \times 1$;
- $\boldsymbol{y}(t)$ = vetor de saída com dimensão $l \times 1$;

- $\boldsymbol{A}$ = matriz de estado com dimensão $n \times n$;
- $\boldsymbol{B}$ = matriz de entrada com dimensão $n \times m$;
- $\boldsymbol{C}$ = matriz de saída com dimensão $l \times 1$;
- $\boldsymbol{D}$ = matriz de transmissão direta com dimensão $l \times m$.

2.1 ATRIBUIÇÃO DE AUTOESTRUTURA COMPLETA

Para entender a aplicação do método de atribuição de autoestrutura completa, é preciso apresentar a influência dos autovalores e autovetores associados sobre a resposta temporal. Nos sistemas MIMO, há um número infinito de matrizes de realimentação que podem atribuir um conjunto específico de autovalores de malha fechada. Para cada matriz de realimentação, há um conjunto diferente de autovetores associados. Tanto os autovalores quanto os autovetores afetam a resposta temporal do sistema (D'AZZO; HOUPIS, 1995).

A atribuição de autoestrutura completa por realimentação de estados são demonstradas neste capítulo através da sistemática de sistemas seguidores (Tracking Systems). Os sistemas seguidores se fazem necessários quando há a necessidade de rastrear um vetor de entrada de comando contendo sinais polinomiais. Neste vetor, são necessárias as informações de saídas específicas y_i, dadas por realimentação, para seguir entradas correspondentes r_i. Um vetor integrador é necessário na topologia do controle seguidor, pois deve ser requerida cada entrada de um sistema MIMO para controlar apenas uma saída correspondente sem afetar as outras saídas do sistema (D'AZZO; HOUPIS, 1995).

Considere o sistema invariante no tempo em malha aberta representado pelas equações de estado e saída:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}(\boldsymbol{t}) + \boldsymbol{B}\boldsymbol{u}(\boldsymbol{t}) \qquad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}(\boldsymbol{t}) \tag{2.4}$$

O espectro de autovalores $\boldsymbol{\sigma}(\boldsymbol{A})$ é o conjunto de raízes da equação característica, que é formada a partir de:

$$Q(\lambda) = |\lambda \boldsymbol{I} - \boldsymbol{A}| = \lambda^n + a_{n-1}\lambda^{n-1} + \cdots + a_1\lambda + a_0 = 0 \tag{2.5}$$

Sabe-se que os autovalores são invariantes em uma transformação linear, como a de equivalência, e quando todos os autovalores de $\boldsymbol{A}$ são distintos, uma matriz modal $\boldsymbol{T}$ pode ser determinada, tal que a matriz $\boldsymbol{A}$ é transformada em uma matriz diagonal $\mathbb{A}$ com os autovalores compondo-a (CHEN, 1999):

$$\boldsymbol{T^{-1}AT} = \mathbb{A} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}. \tag{2.6}$$

Sempre quando os autovalores λ são distintos, temos que os autovetores $\boldsymbol{v_i}$ são as colunas de $\boldsymbol{T}$ e satisfazem a equação:

$$[\lambda_i \boldsymbol{I} - \boldsymbol{A}]\boldsymbol{v_i} = 0 \quad ; \quad \boldsymbol{v_i} \neq 0. \tag{2.7}$$

Os autovetores recíprocos $\boldsymbol{w_i^T}$ são as linhas de $\boldsymbol{T^{-1}}$ e satisfazem a equação:

$$\boldsymbol{w_i^T}[\lambda_i \boldsymbol{I} - \boldsymbol{A}] = 0 \quad ; \quad \boldsymbol{w_i^T} \neq 0. \tag{2.8}$$

Portanto:

$$\boldsymbol{T} = [\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \cdots \quad \boldsymbol{v_n}] \qquad \boldsymbol{T^{-1}} = \begin{bmatrix} \boldsymbol{w_1^T} \\ \boldsymbol{w_2^T} \\ \vdots \\ \boldsymbol{w_n^T} \end{bmatrix}. \tag{2.9}$$

Pode-se mostrar que:

$$x(t) = e^{\boldsymbol{A}t}\boldsymbol{x(0)} + \int_0^t e^{\boldsymbol{A}\beta}\boldsymbol{Bu}(\boldsymbol{t} - \boldsymbol{\beta})d\beta. \tag{2.10}$$

No qual:

$$e^{\boldsymbol{A}t} = \begin{bmatrix} \boldsymbol{v_1} & \boldsymbol{v_2} & \cdots & \boldsymbol{v_n} \end{bmatrix} \begin{bmatrix} e^{\lambda_1 t} & 0 & \cdots & 0 \\ 0 & e^{\lambda_2 t} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & e^{\lambda_n t} \end{bmatrix} \begin{bmatrix} \boldsymbol{w_1^T} \\ \boldsymbol{w_2^T} \\ \vdots \\ \boldsymbol{w_n^T} \end{bmatrix}, \tag{2.11}$$

$$\boldsymbol{v_1} e^{\lambda_1 t} \boldsymbol{w_1^T} + \boldsymbol{v_2} e^{\lambda_2 t} \boldsymbol{w_2^T} + \cdots + \boldsymbol{v_n} e^{\lambda_n t} \boldsymbol{w_n^T} = \sum_{i=1}^{n} \boldsymbol{v_i} e^{\lambda_i t} \boldsymbol{w_i^T}. \tag{2.12}$$

Utilizando as equações (2.4), (2.10) e (2.12), chega-se a:

$$y(t) = \sum_{i=1}^{n} (\boldsymbol{C v_i})\, e^{\lambda_i t} \boldsymbol{w_i^T x(0)} + \sum_{j=1}^{m} \sum_{i=1}^{n} (\boldsymbol{C v_i})(\boldsymbol{w_i^T} b_j) \int_0^t e^{\lambda_i \beta} u_j(t-\beta) d\beta. \tag{2.13}$$

Nota-se que a resposta transiente do sistema é, portanto, determinado pelo autovetor $\boldsymbol{v_i}$, e sua característica de domínio no tempo é determinada pelo autovalor λ_i.

$$\boldsymbol{v_i} e^{\lambda_i t}, \quad i = 1,2, \dots, n. \tag{2.14}$$

A Equação (2.13) mostra que a atribuição de autoestrutura completa determina a resposta temporal do sistema, ou seja, os autovalores λ_i, os autovetores associados $\boldsymbol{v_i}$ e os autovetores recíprocos $\boldsymbol{w_i}$ contribuem na resposta temporal.

O propósito de aplicar a realimentação de estados é atribuir um espectro de autovalores de malha fechada.

$$\boldsymbol{\sigma(A + BK)} = \{\lambda_1, \lambda_2, \dots, \lambda_n\}, \tag{2.15}$$

e um conjunto associado de autovetores:

$$\boldsymbol{v(A + BK)} = \{v_1, v_2, \dots, v_n\}, \tag{2.16}$$

que são selecionados com o objetivo de alcançar as características de reposta temporal desejadas. Os autovalores e autovetores de malha fechada são relacionados com a Equação (2.17):

$$[\boldsymbol{A} + \boldsymbol{BK}]\boldsymbol{v_i} = \lambda_i \boldsymbol{v_i}. \tag{2.17}$$

Esta equação pode ser também apresentada na forma (2.18):

$$[\boldsymbol{A} - \lambda_i \boldsymbol{I} \quad \boldsymbol{B}] \begin{bmatrix} \boldsymbol{v}_i \\ \boldsymbol{q}_i \end{bmatrix} = 0\,, \qquad i = 1,2,\dots,n. \tag{2.18}$$

Onde:

$$\boldsymbol{q}_i = \boldsymbol{K}\boldsymbol{v}_i. \tag{2.19}$$

Com o objetivo de satisfazer a Equação (2.18), o vetor $[\boldsymbol{v}_i^T \quad \boldsymbol{q}_i^T]^T$ deve pertencer ao espaço nulo da matriz:

$$S(\lambda_i) = [\boldsymbol{A} - \lambda_i \boldsymbol{I} \quad \boldsymbol{B}]. \tag{2.20}$$

A partir de (2.19), chega-se a:

$$\begin{gathered} [\boldsymbol{q}_1 \quad \boldsymbol{q}_2 \quad \cdots \quad \boldsymbol{q}_n] = [\boldsymbol{K}\boldsymbol{v}_1 \quad \boldsymbol{K}\boldsymbol{v}_2 \quad \dots \quad \boldsymbol{K}\boldsymbol{v}_n] \\ \boldsymbol{K} = [\boldsymbol{q}_1 \quad \boldsymbol{q}_2 \quad \cdots \quad \boldsymbol{q}_n][\boldsymbol{v}_1 \quad \boldsymbol{v}_2 \quad \dots \quad \boldsymbol{v}_n]^{-1} \\ \boldsymbol{K} = \boldsymbol{Q}\boldsymbol{V}^{-1} \end{gathered} \tag{2.21}$$

Portanto, se um espectro de autovalores desejados é especificado em (2.15) e os autovetores associados a estes são selecionados para satisfazer a Equação (2.18), então, a Equação (2.21) nos fornece a matriz $\boldsymbol{K}$ de realimentação de estados, ou seja, os ganhos do sistema.

Também, para $\boldsymbol{K}$ ser sempre uma matriz real, o conjunto de autovetores selecionados devem ser auto conjugados, ou seja, eles devem ser complexo conjugados para os pares de autovalores que são complexo conjugados.

Por exemplo, dado um sistema com $A_{3\times3}$ e $B_{3\times2}$, e utilizando o controle por realimentação de estados para alocar o espectro de autovalores, onde $\lambda_1 \in \mathbb{R}$ e $(\lambda_1, \lambda_2) \in \mathbb{C}$, sendo estes dois últimos complexos conjugados. O cálculo de $\boldsymbol{K}$, dado pela Equação (2.14), se estende como:

$$\boldsymbol{K} = [\boldsymbol{q}_1 \quad Re(\boldsymbol{q}_2) \quad Im(\boldsymbol{q}_2)][\boldsymbol{v}_1 \quad Re(\boldsymbol{v}_2) \quad Im(\boldsymbol{v}_2)]^{-1} \tag{2.22}$$

2.1.1 Controle Seguidor com Realimentação de Estado

Um sistema controlável MIMO é representado pelas equações de estado e saída, respectivamente:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.23}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} = \begin{bmatrix}\boldsymbol{E}\\ \boldsymbol{F}\end{bmatrix}\boldsymbol{x} \tag{2.24}$$

Dependendo dos parâmetros do sistema, há variáveis de saída que participam da realimentação de estados e variáveis que não participam. Onde a matriz $\boldsymbol{E}_{p\times n}$ é responsável pela seleção das variáveis de saída controladas, ou seja, que geram realimentação para o controle do sistema, sendo p o número de variáveis de saída controláveis, e $\boldsymbol{F}_{(l-p)\times n}$ responsável pela seleção das variáveis de saída não controladas, ou seja, as que não participam da realimentação, com l definido pela dimensão do vetor de saída, $\boldsymbol{y}_{l\times1}$. Com isso, pode-se dizer que as saídas y_i controladas, ou seja $i = 1, 2, \dots, p$, devem ser os primeiros p elementos da saída $\boldsymbol{y}$, devendo este ser reorganizado, se necessário, para cumprir este requerimento.

Para se manter a controlabilidade, o número de saídas controláveis p nunca devem exceder o número de entradas a serem controladas m. Sendo $\boldsymbol{w} = \boldsymbol{E}\boldsymbol{x}$, representando todas as saídas controláveis requeridas para seguir um vetor de referência $\boldsymbol{r}$ constante em partes de dimensão $p \times 1$, tem-se a seguinte formulação matemática, em regime permanente:

$$\lim_{t\to\infty} \boldsymbol{w}(t) = \boldsymbol{r}(t) \tag{2.25}$$

O método do controle seguidor, onde há a realimentação das variáveis de saída controladas, consiste na adição de um vetor comparador e um integrador, que satisfazem a seguinte equação:

$$\dot{\boldsymbol{z}} = \boldsymbol{r} - \boldsymbol{w} = \boldsymbol{r} - \boldsymbol{E}\boldsymbol{x} \tag{2.26}$$

O sistema em malha aberta é então governado por equações de estado e saída aumentadas, originadas das equações (2.23) e (2.24):

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{z}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{E} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{r} = \bar{\boldsymbol{A}}\boldsymbol{x}' + \bar{\boldsymbol{B}}\boldsymbol{u} + \bar{\boldsymbol{B}}'\boldsymbol{r} \tag{2.27}$$

$$\boldsymbol{y} = [\boldsymbol{C} \quad \boldsymbol{0}] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} = \bar{\boldsymbol{C}}\boldsymbol{x}' \tag{2.28}$$

Onde:

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ -\boldsymbol{E} & \boldsymbol{0} \end{bmatrix};\ \bar{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix};\ \boldsymbol{B}' = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix};\ \bar{\boldsymbol{C}} = [\boldsymbol{C} \quad \boldsymbol{0}] \tag{2.29}$$

A lei de controle a ser utilizada na realimentação de estado é:

$$\boldsymbol{u} = \boldsymbol{K_1}\boldsymbol{x} + \boldsymbol{K_2}\boldsymbol{z} = [\boldsymbol{K_1} \quad \boldsymbol{K_2}] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} = \bar{\boldsymbol{K}} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} \tag{2.30}$$

Um diagrama de blocos que representa o sistema de controle realimentado, contendo as equações de estado (Equação (2.23)), saída (Equação (2.24)) e lei de controle (Equação (2.26)), é apresentado na Figura 2.1. Nela, pode-se ver o ganho $\boldsymbol{K_1}$ atuando na realimentação dos estados e o ganho $\boldsymbol{K_2}$ atuando na integral do erro entre referência e saída das variáveis controladas.

Esta lei de controle pode alocar o espectro de autovalores desejados em malha fechada, se e somente se, a planta aumentada e o par de matriz de controle $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{B}})$ forem controláveis. Portanto, esta condição é satisfeita se $(\boldsymbol{A}, \boldsymbol{B})$ for um par controlável, algo verificável com a definição de um sistema completamente controlável, onde a matriz de controlabilidade $\boldsymbol{M}_c$ deve possuir a característica da Equação (2.31), e também se for verdadeira a Equação (2.32):

$$posto\ \boldsymbol{M}_c = posto[\boldsymbol{B} \quad \boldsymbol{AB} \quad \boldsymbol{A}^2\boldsymbol{B} \quad \cdots \quad \boldsymbol{A}^{n-m}\boldsymbol{B}] = n \tag{2.31}$$

$$posto \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A} \\ \boldsymbol{0} & -\boldsymbol{E} \end{bmatrix} = n + m \tag{2.32}$$

Satisfazendo as Equações (2.31) e (2.32), pode-se sintetizar a lei de controle. Logo, a Equação (2.33) demonstra as equações de estados em malha fechada:

$$\dot{\boldsymbol{x}}' = \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{z}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} + \boldsymbol{BK_1} & \boldsymbol{BK_2} \\ -\boldsymbol{E} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{r} = \boldsymbol{A'}_{\boldsymbol{cl}}\boldsymbol{x}' + \boldsymbol{B}'\boldsymbol{r} \tag{2.33}$$

A matriz de realimentação deve ser selecionada de modo que os autovalores da planta estejam todos no semiplano complexo esquerdo. Portanto, a matriz $\overline{\boldsymbol{K}}$ é obtida ao selecionar os autovalores desejados ao sistema em malha fechada apresentado na Equação (2.33).

Pode-se utilizar a técnica de atribuição por autoestrutura completa, onde o espectro de autovalores desejados (Equação (2.15)), é dado por:

$$\boldsymbol{\sigma}(\overline{\boldsymbol{A}} + \overline{\boldsymbol{B}}\overline{\boldsymbol{K}}) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\} \tag{2.34}$$

E os autovetores associados devem pertencer ao espaço nulo (Equação (2.20)), onde:

$$\overline{\boldsymbol{S}}(\lambda_i) = [\overline{\boldsymbol{A}} - \lambda_i \boldsymbol{I} \quad \overline{\boldsymbol{B}}] \tag{2.35}$$

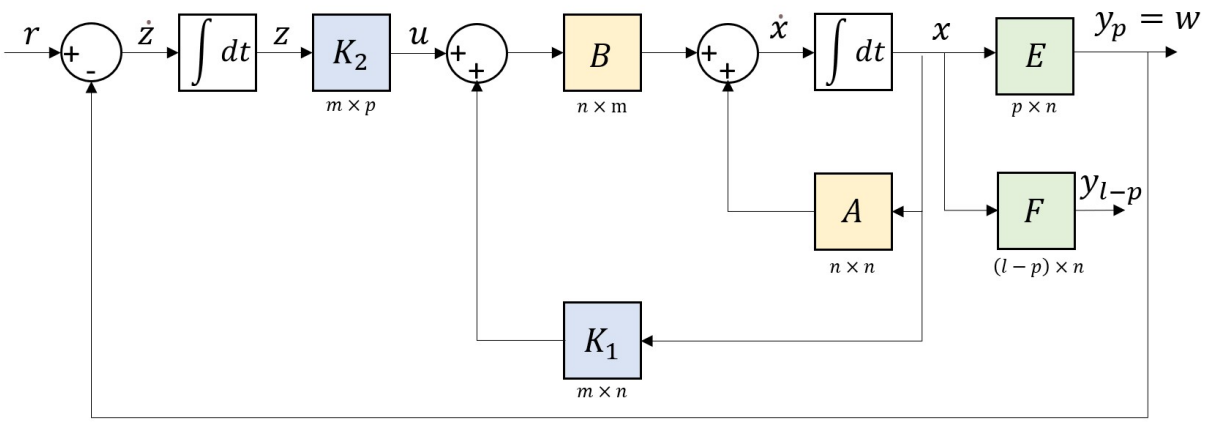


Figura 2.1 – Diagrama de Blocos do Controle Seguidor com realimentação de estados
Fonte: Autoria própria

3 CONSTRUÇÃO E INSTRUMENTAÇÃO DA PLANTA DO HELICÓPTERO TANDEM

Este capítulo descreve a construção de uma planta que tem dinâmica semelhante a um helicóptero do tipo Tandem. O modelo de referência para a construção é a planta do Helicóptero de 3 graus de liberdade (GDL) da *Quanser*[1].

A planta representa a aeronave helicóptero Tandem através de uma estrutura retangular com dois motores de corrente contínua (DC) e imã permanente alocados em cada extremidade, acoplados às hélices de passo fixo, gerando carga no motor não proporcional à velocidade aplicada, resultando em uma força de empuxo em direção ao passo definido (HOMA, 2010; QUANSER, 2012). Como os motores estão presos na mesma estrutura, possuem eixos e orientações semelhantes, criando uma força de empuxo normal à estrutura da aeronave (QUANSER, 2012).

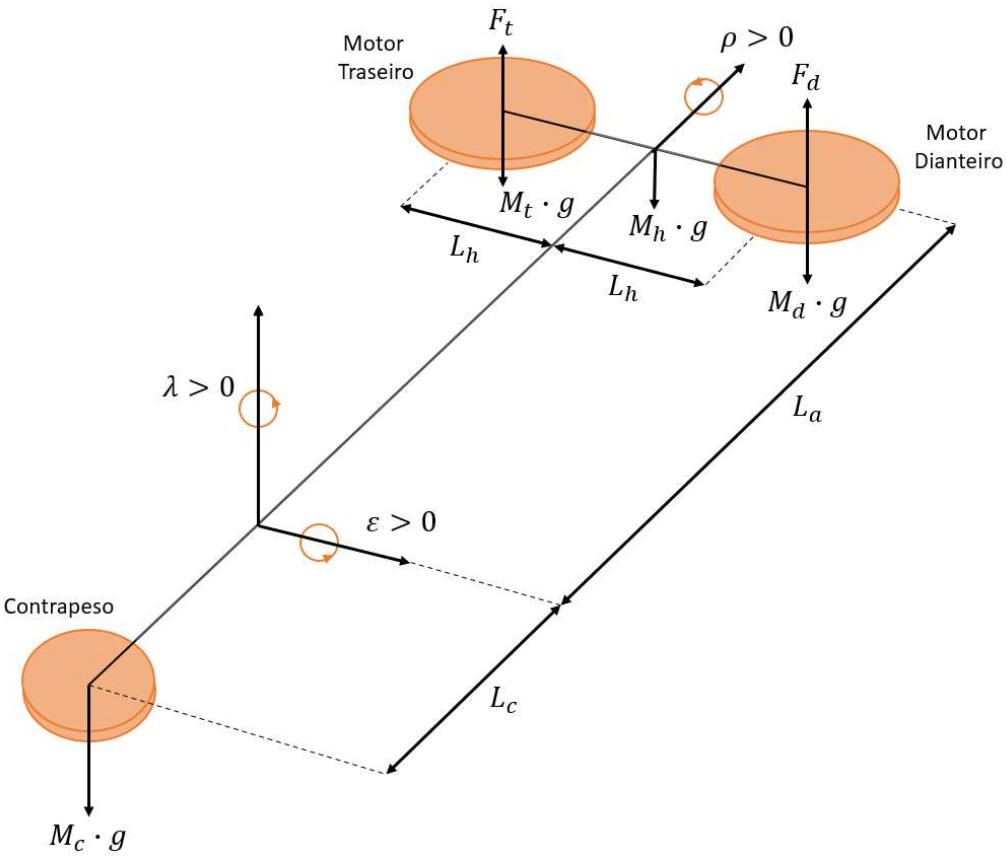


Figura 3.1 – Diagrama de corpo livre do Helicóptero de 3 GDL
Fonte: Autoria Própria

[1] <https://www.quanser.com/>. Acesso em: 25 de nov. de 2021.

Analisando a representação do diagrama de corpo livre de um helicóptero de 3 GDL, dada pela Figura 3.1, é possível fazer a identificação de 3 movimentos, assim como suas orientações. Como a aeronave é fixa por uma junta de revolução localizada na extremidade de um braço, possibilita-se o movimento de arfagem ρ. O mesmo braço é montado em cima de uma junta de revolução de 2 GDL, gerando os movimentos de deslocamento λ e elevação ε. Na outra extremidade do braço é instalado um contrapeso com o objetivo de diminuir a força de empuxo necessária para a elevação da aeronave (QUANSER, 2012).

Um conjunto de distâncias ditam uma configuração do helicóptero. A Figura 3.1 mostra a distância do eixo de arfagem até o centro dos motores L_h, a distância do centro dos motores até o eixo de 2 GDL L_a e a distância do eixo de 2 GDL até o contrapeso L_c. Por fim, o diagrama ainda representa as forças significantes para uma modelagem simplificada, destacando as forças de empuxo do motor traseiro F_t e dianteiro F_d e os pesos dados pelas massas do conjunto do motor traseiro M_t, do conjunto do motor dianteiro M_d, da haste do braço M_h e do contrapeso M_c.

A Figura 3.2 faz a analogia entre os 3 movimentos da planta e um helicóptero do tipo tandem, o HC-1B Chinook da Boeing.

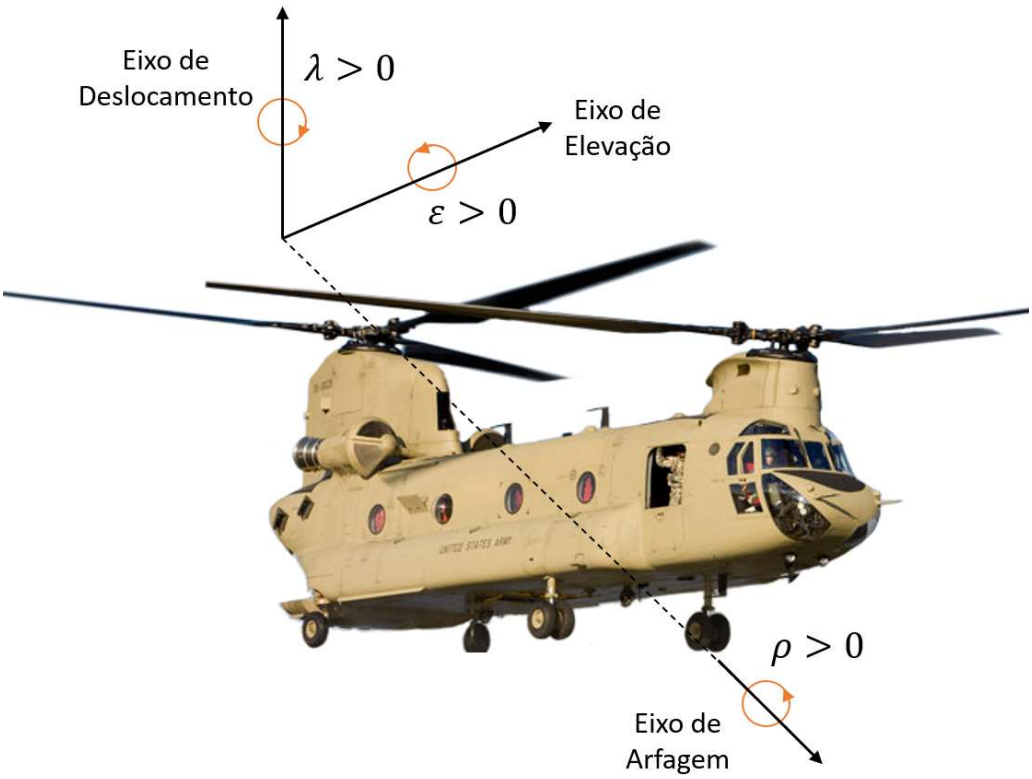


Figura 3.2 – Os 3 movimentos em um Helicóptero Tandem (HC-1B Chinook)
Fonte: Autoria Própria (Creative Common)

A construção da planta foi realizada a partir da usinagem de materiais do tipo acrílico, alumínio e aço-inox. Para a montagem, a planta foi separada em 5 partes: base, haste, braço, aeronave e contrapeso. A Figura 3.3 mostra a planta construída, assim como a indicação de cada parte do projeto mecânico.

As Figuras 3.4, 3.5, 3.6, 3.7 e 3.8 mostram o desenho técnico da base, haste, braço, aeronave e contrapeso, respectivamente, destacando dimensões e detalhes da construção. Já as Tabelas 3.1, 3.2, 3.3, 3.4 e 3.5 informam os pesos e densidades de cada item utilizado na base, haste, braço, aeronave e contrapeso, respectivamente.

A instrumentação da planta conta com 2 motores DC A380/PL24-21600S da *Akiyama* (Joinville, Brasil), 2 drivers de potência SX8847 da *Sonxun* (Pequim, China), 2 encoders RCML15 da *RENCO* (Traunreut, Alemanha) para os movimentos de arfagem e elevação, e 1 encoder Q9863 da *HP* (Palo Alto, EUA) para o movimento de deslocamento. A interface entre computador e planta é realizada através da placa de aquisição *PCI-6602* da *National Instruments* (Austin, EUA).

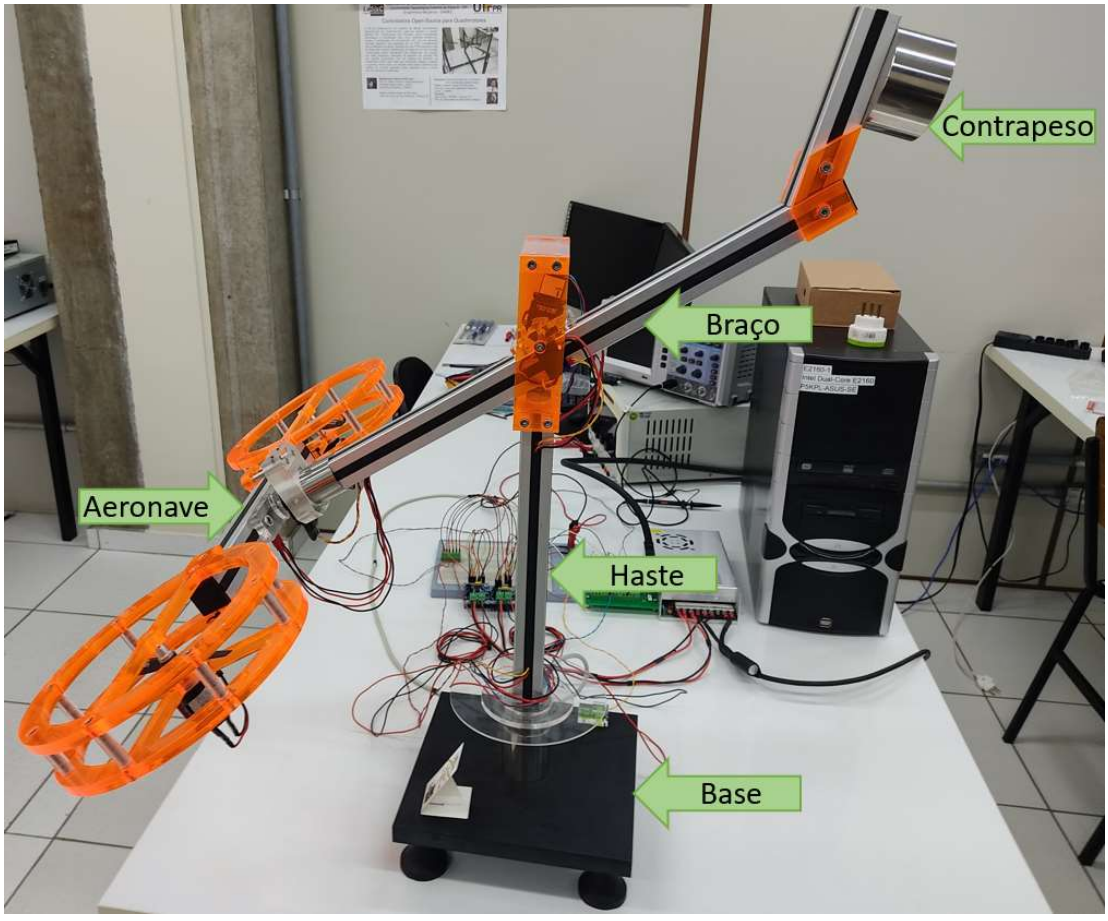


Figura 3.3 – Planta Construída. É possível identificar as 5 partes: base, haste, braço, aeronave e contrapeso

Fonte: Autoria Própria

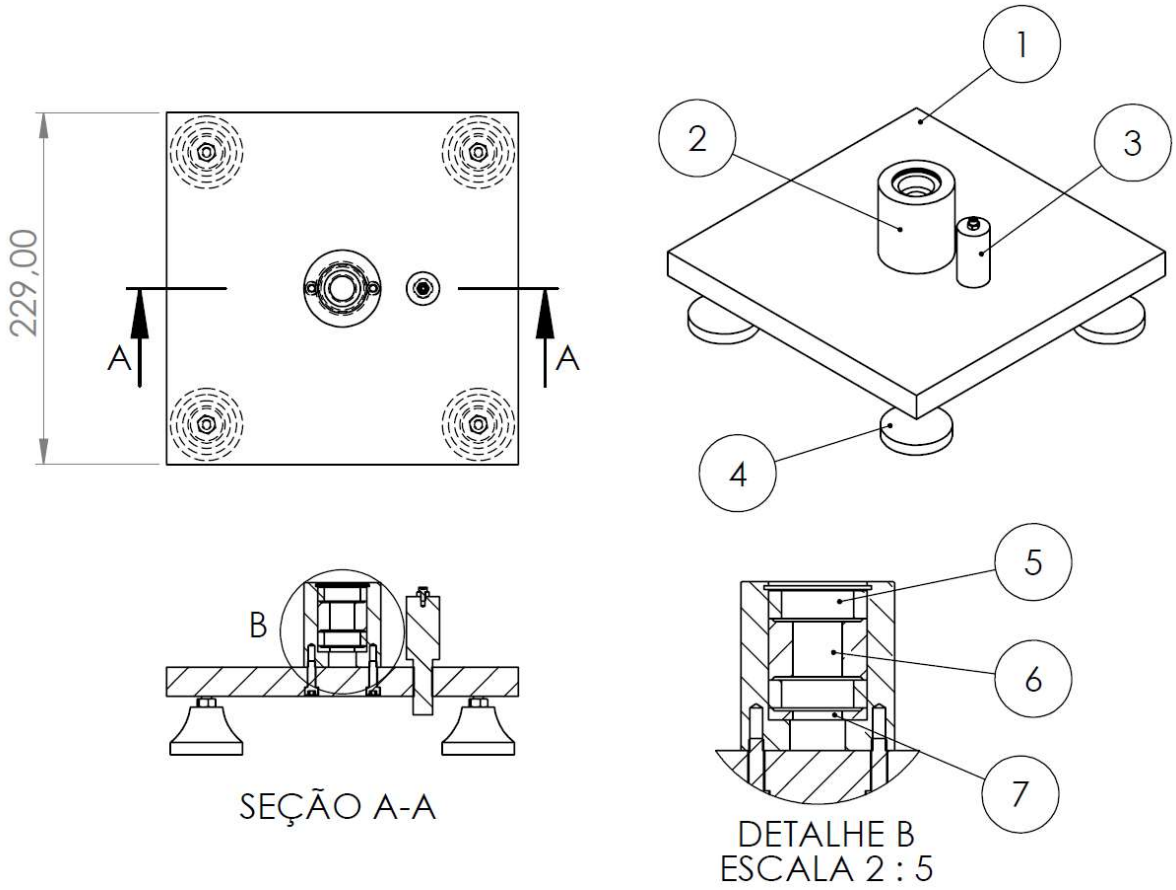


Figura 3.4 – Desenho Técnico da Base
Fonte: Autoria Própria

Tabela 3.1 – Itens da Base

ID	Nome	Peso (g)	Densidade (Kg/m^3)
1	Base	8100,00	8168,04
2	Mancal	557,10	8114,88
3	Suporte encoder	170,24	8768,59
4	Pé industrial	52,33	1790,71
5	Anel externo	17,41	5223,26
6	Espaçador intermediário	29,40	2638,12
7	Espaçador inferior	4,60	2355,32
-	Alen M3x6	0,65	6360,70
-	Alen M5x17	3,60	8114,88
-	Porca M3	0,71	6348,92
-	Porca 1/4	3,15	6835,04

Fonte: Autoria Própria

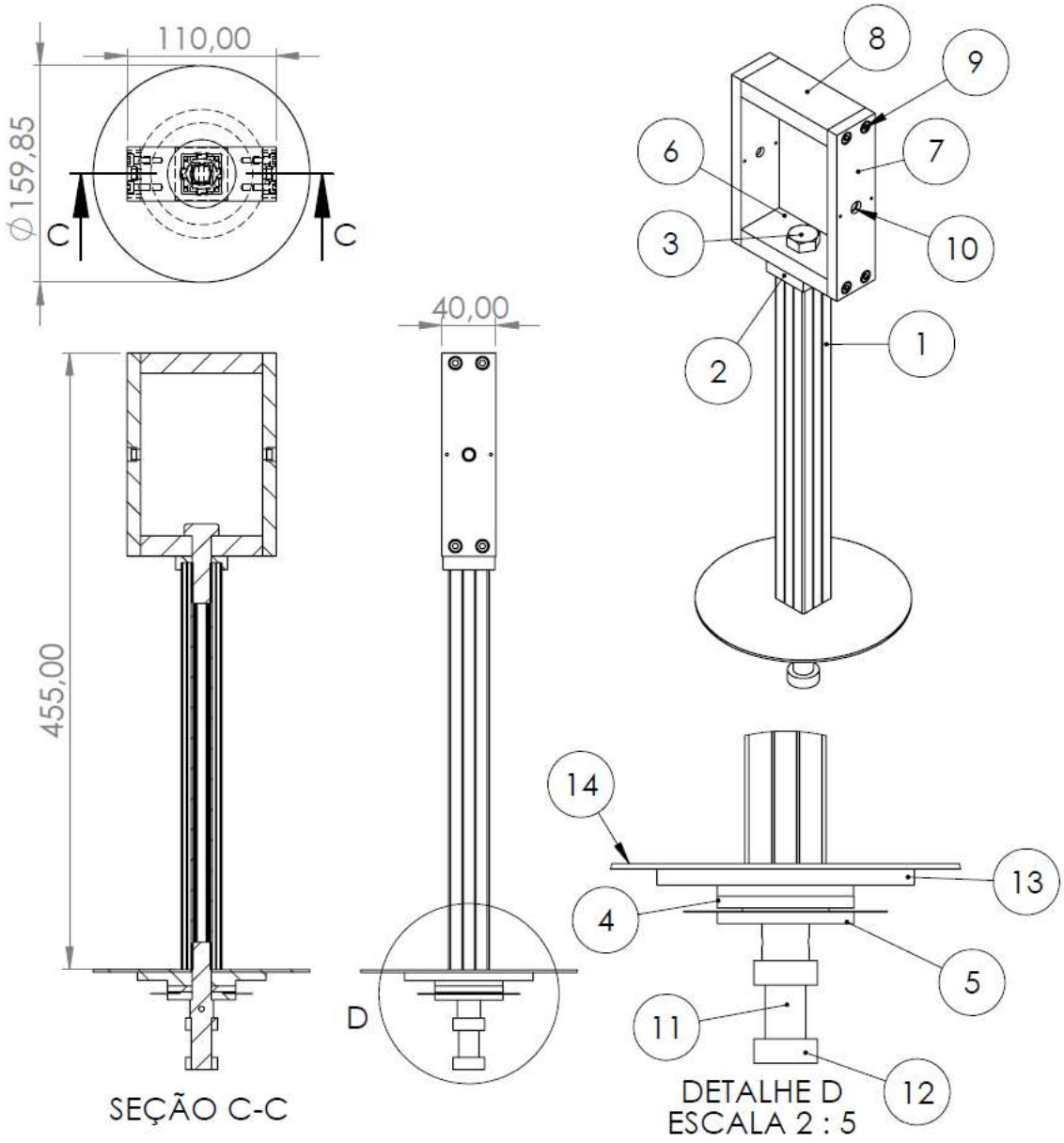


Figura 3.5 – Desenho Técnico da Haste
Fonte: Autoria Própria

Tabela 3.2 – Itens da Haste

(continua)

ID	Nome	Peso (g)	Densidade (Kg/m^3)
1	Perfil de alumínio	231,12	2572,30
2	Placa de proteção	10,66	1114,77
3	M14x50	67,06	5906,82
4	Arruela inox superior	64,36	7922,98
5	Arruela inox inferior	68,78	7049,13
6	Placa inferior	57,64	1133,83
7	Placa lateral	65,30	1140,21

Tabela 3.2 – Itens da Haste

(conclusão)

ID	Nome	Peso (g)	Densidade (Kg/m^3)
8	Placa superior	60,48	1138,00
9	M5x12	3,04	6526,72
10	Anel externo	0,70	5122,07
11	Eixo	117,00	7219,09
12	Anel interno	12,08	5223,28
13	Disco de proteção	47,40	1191,44
14	Disco de proteção	57,50	1175,47

Fonte: Autoria Própria

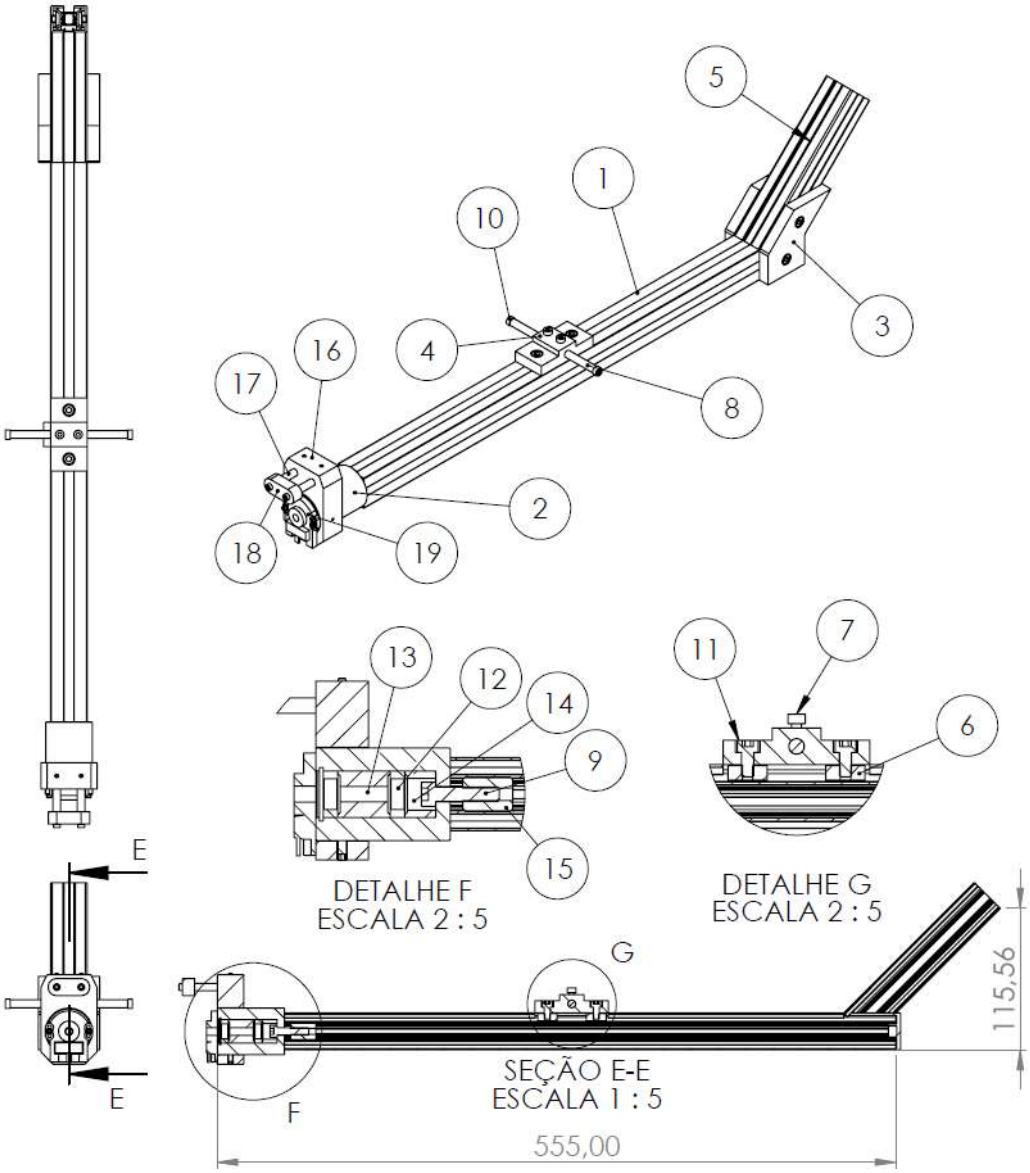


Figura 3.6 – Desenho Técnico do Braço

Fonte: Autoria Própria

Tabela 3.3 – Itens do Braço

ID	Nome	Peso (g)	Densidade (Kg/m^3)
1	Perfil de alumínio	408,68	2560,21
2	Mancal	377,90	7862,69
3	Fixadores do perfil	26,68	1090,30
4	Suporte do eixo de elevação	22,20	1166,88
5	Perfil de alumínio do contrapeso	105,50	2590,08
6	Porcas fixadoras	8,25	7241,83
7	M4x6	0,65	6360,70
8	Eixo	23,92	7787,44
9	M6x26	6,80	6149,33
10	Anel Interno	0,50	5122,07
11	M5x10	2,94	6869,17
12	Anel externo	4,85	5047,65
13	Espaçador intermediário	13,50	2685,51
14	Espaçador inferior	4,90	2708,33
15	Bucha roscada	10,90	4020,16
16	Suporte Limitador	48,28	1132,61
17	Extensor limitador	11,42	8171,32
18	Limitador da arfagem	4,08	1078,10
19	Encoder	11,1	1543,17
-	Tampa 30mm	2,94	6869,17
-	M4x8 SC	0,46	5391,46
-	M3x8 SC	0,25	4727,68
-	M3x8	0,79	4727,68
-	M2x10	0,20	3862,49
-	Arruela M2	0,07	4283,96
-	Suporte cabos	2,11	1161,02

Fonte: Autoria Própria

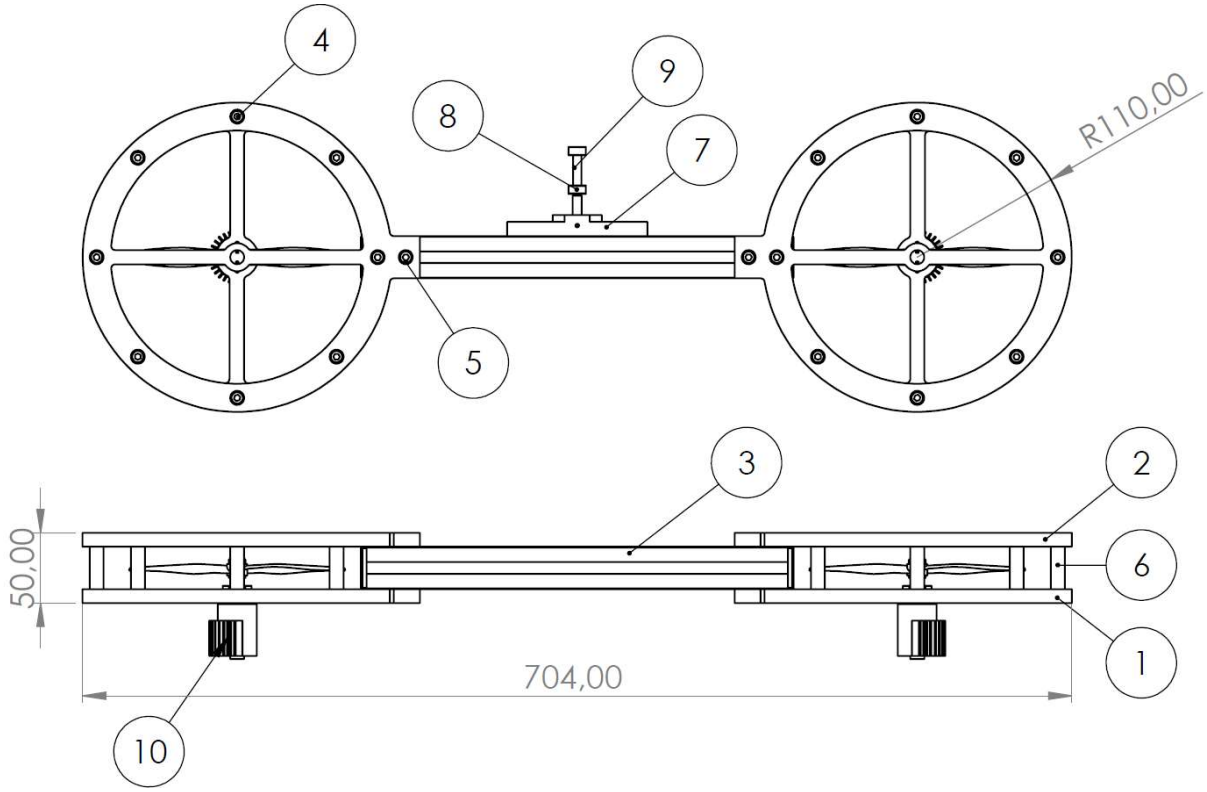


Figura 3.7 – Desenho Técnico da Aeronave
Fonte: Autoria Própria

Tabela 3.4 – Itens da Aeronave

ID	Nome	Peso (g)	Densidade (Kg/m^3)
1	Proteção inferiror	174,76	1085,56
2	Proteção superior	164,43	1126,88
3	Perfil de alumínio	232,20	2584,32
4	M5x45	7,02	6439,04
5	M5x10	2,94	6869,17
6	Espaçador	2,08	1195,97
7	Suporte	38,44	1165,18
8	Anel interno	2,84	5047,48
9	Eixo	16,40	7821,33
10	Conjunto Motor	86,84	25387,43
-	Tampa 30mm	2,87	891,56
-	Porca fixadora	8,25	7241,83
-	Porca M5	1,05	6357,08
-	M3x10 SC	0,38	5669,94

Fonte: Autoria Própria

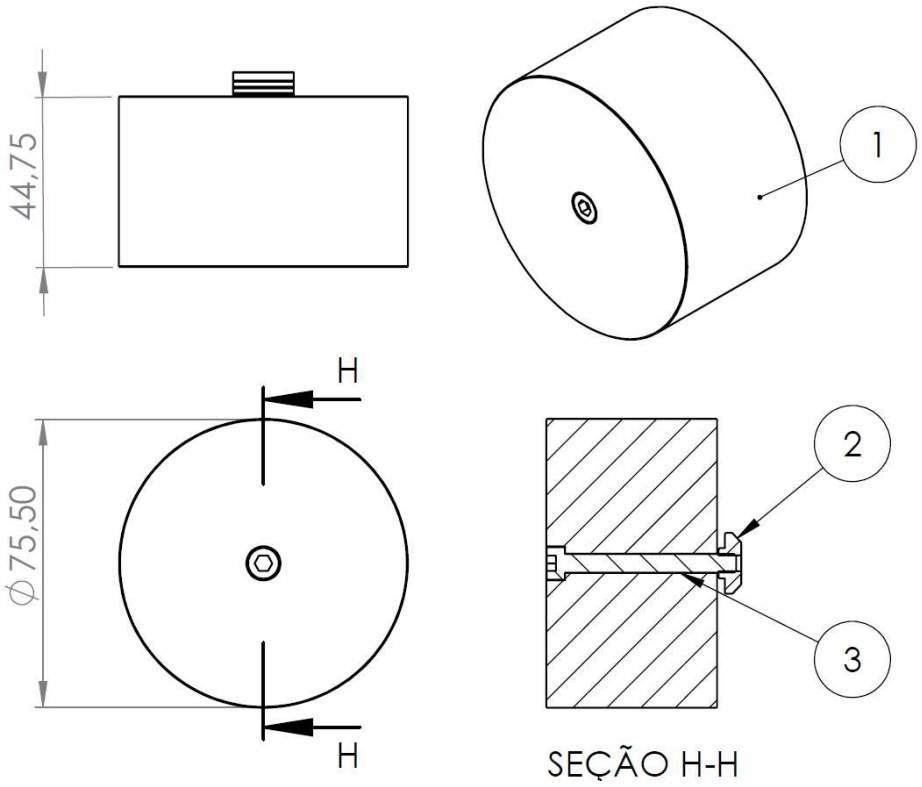


Figura 3.8 – Desenho Técnico do Contrapeso
Fonte: Autoria Própria

Tabela 3.5 – Itens do Contrapeso

ID	Nome	Peso (g)	Densidade (Kg/m^3)
1	Contrapeso	1603,10	7896,17
2	Porca fixadora	8,25	7241,83
3	M5x45	7,38	6764,43

Fonte: Autoria Própria

4 MODELAGEM DE SISTEMAS MULTICORPOS

Esta seção descreve a metodologia aplicada para a modelagem do sistema do helicóptero de 3 GDL, utilizando técnicas de modelagem de Sistemas Multicorpos (do inglês, Multibody Systems – MBS). Segundo Montezuma (2003), uma modelagem deve seguir os seguintes passos:

1. Descrever o modelo físico de um sistema, quando possível, e obter equações que descrevem a dinâmica do sistema;
2. Resolver as equações resultantes de maneira analítica ou numérica, a fim de estimar o comportamento do sistema;
3. Verificar os resultados do modelo, comparando-o com o sistema real;

A “Prototipagem Virtual” é uma técnica de modelagem MBS, que através de programas computacionais, auxilia na geração de equações de movimento de um sistema mecânico que possa ser modelado em um conjunto de corpo rígidos interconectados por juntas, influenciado por forças, dirigido por movimentos prescritos e limitado por vínculos, apenas baseando sobre a geometria e inércia dos corpos e suas interconexões. Tal técnica permite desenvolver, testar, reajustar e otimizar o desempenho de sistemas mecânicos submetidos a qualquer movimento. Programas computacionais para geração de equações MBS, como o software *ADAMS* (Irvine, EUA), auxiliam nas tarefas 2 e 3 (MONTEZUMA, 2003).

O *ADAMS*, desenvolvido pela empresa *MSC Software*, é um software MBS do tipo Engenharia Assistida por Computador (CAE). A tecnologia CAE (do inglês, Computer-aided Engineering - CAE) tem o objetivo de analisar o desenho construído em um software do tipo Desenho Assistido por Computador (do inglês, Computed Aided Design – CAD). Ele auxilia no estudo dinâmico de corpos móveis e como as cargas e forças são distribuídas através do sistema mecânico, buscando a eficiência e redução de custos de desenvolvimento de projetos a partir de uma validação prévia do design do sistema (MSC SOFTWARE, 2021). Com o auxílio do *ADAMS*, e com resultados de sua simulação, pode-se ajustar os parâmetros de projeto para melhorar o desempenho do sistema (MONTEZUMA, 2003).

Na aplicação tradicional, através de protótipos reais no aprendizado sobre sistemas complexos, tem-se as seguintes fases de aplicação (MONTEZUMA, 2003):

3. Projetar, construir e montar o sistema mecânico;
4. Instrumentar o sistema;
5. Executar testes;
6. Coletar dados e organizá-los;
7. Interpretar dados;
8. Tomada de decisões (modificar o protótipo e repetir as fases anteriores).

Já no aprendizado utilizando a técnica de prototipagem virtual, o processo é dividido em (MONTEZUMA, 2003):

1. Construir/Modelar o sistema;
2. Instrumentar o sistema, declarando saídas de interesse no programa de MBS;
3. Executar testes, com simulações;
4. Comparar o desempenho de várias configurações através de animações gráficas;
5. Interpretar os dados gerados pelo software;
6. Tomada de decisões.

Deve-se atentar que o uso da prototipagem virtual não elimina a necessidade de construção de um protótipo real. O seu uso motivado pela redução de tempo e custo em testes em protótipos reais (MONTEZUMA, 2003).

Como a base para o campo de MBS são as leis da física, a implementação de equações dinâmicas no ADAMS é realizada pelo sistema Euler-Lagrange, sendo este sistema composto por equações diferenciais de segunda ordem, primeira ordem e equações algébricas. Para realizar uma simulação, o *ADAMS* necessita das características inerciais dos corpos, da relação de interação entre os corpos e dos movimentos e forças presentes no sistema. Portanto, deve-se ter como dados de entrada no módulo *Adams/View*: (MONTEZUMA, 2003):

- Massa e inércia dos corpos rígidos;
- Definição dos centros de massa das partes, da posição de união das juntas, e dos pontos de forças aplicadas ao sistema e movimentos;
- Conectividade do sistema através de juntas mecânicas, definindo assim, os graus de liberdade do sistema;
- Descrição de forças internas e externas do sistema;
- Atributos gráficos para animação e visualização da dinâmica.

O software CAD utilizado no projeto é o *SolidWorks* da *Dassault Systèmes SolidWorks Corporation* (Waltham, EUA). A adaptação do sistema de coordenadas do *SolidWorks* para o *ADAMS* na estimativa de orientação do centro de massa é realizada por matrizes de rotação.

Das análises realizáveis pelo *ADAMS*, é de interesse do projeto, a análise de equilíbrio estático, a dinâmica e a linear. Montezuma (2003), as descrevem como:

1. **Análise de equilíbrio estático**: Determinar um estado em que o sistema entra em equilíbrio na ausência de qualquer movimento ou força inercial, ou seja, todas as velocidades e acelerações são nulas. Usada para determinar o ponto de partida para uma análise dinâmica, removendo os transitórios indesejados no início da simulação;
2. **Análise dinâmica**: Utiliza-se vários integradores diferentes para determinar a solução de um sistema de equações diferenciais e algébricas. Esta fornece a solução em relação ao tempo para os deslocamentos, velocidades, acelerações e forças de reação internas em um sistema mecânico passível de forças e excitações externas;
3. **Análise linear**: Lineariza o sistema de equações não-lineares em cima de um ponto de operação, fornecendo equações lineares invariantes no tempo.

4.1 PROTOTIPAGEM VIRTUAL DO SISTEMA

Utiliza-se como ferramenta CAD, o software *SolidWorks*. O processo utilizado para a criação de um modelo virtual 3D CAD segue:

1. Medir as características geométricas das peças reais;
2. Determinação das massas específicas de cada peça;
3. Realizar o desenho 3D, contendo os dados das etapas 1 e 2;
4. Montagem dos múltiplos corpos, divididos em: base, haste, braço, aeronave e contrapeso;
5. Identificação das propriedades de centro de massa, massa e momento de inércia para cada um dos corpos rígidos.

Os corpos rígidos base, haste, braço, aeronave e contrapeso são apresentados respectivamente da Figura 4.1 até a Figura 4.5.

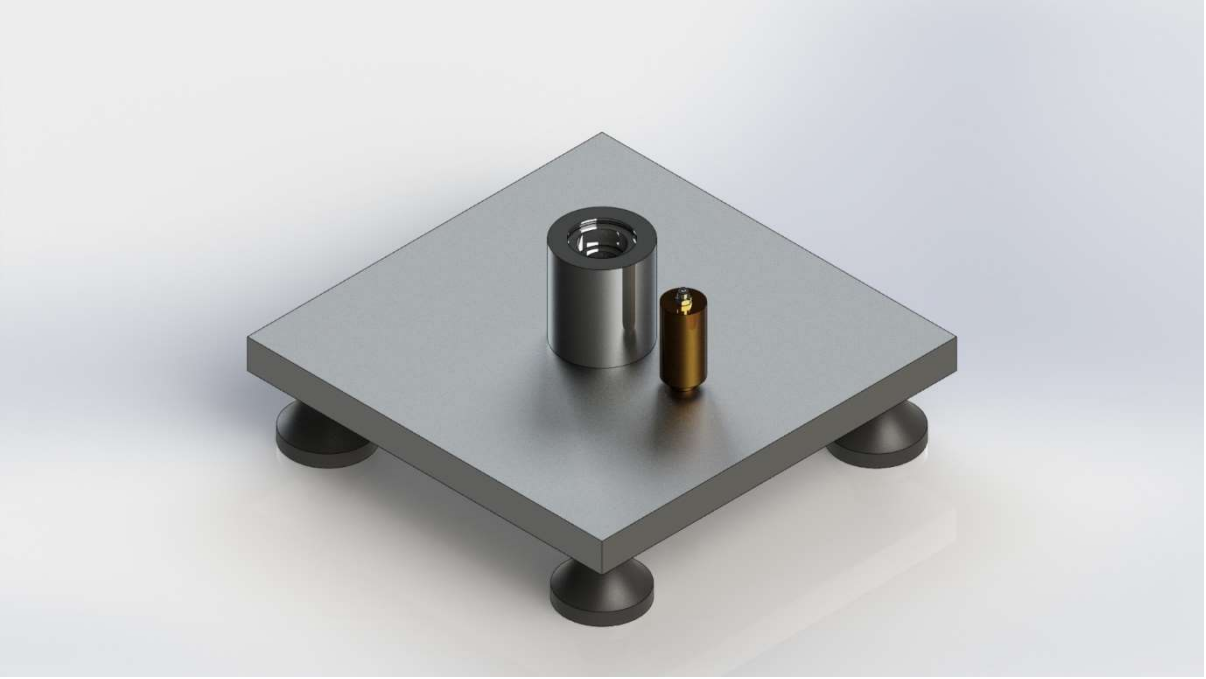


Figura 4.1 – Base no SolidWorks
Fonte: Autoria Própria

Figura 4.2 – Haste no SolidWorks
Fonte: Autoria Própria

Figura 4.3 – Braço no SolidWorks
Fonte: Autoria Própria

Figura 4.4 – Aeronave no SolidWorks
Fonte: Autoria Própria

Figura 4.5 – Contrapeso no SolidWorks
Fonte: Autoria Própria

Figura 4.6 – Montagem da planta virtual no SolidWorks
Fonte: Autoria Própria

A Figura 4.6 ilustra todos os corpos rígidos montados pelas juntas de revolução. Para maior fidelidade com o modelo real, foram consideradas qualquer massa relevante no sistema, ou seja, todos os elementos de eletrônica embarcada, inclusive conectores e cabos, foram aplicados ao modelo virtual.

As propriedades de cada corpo rígido estão apresentadas na Tabela 4.1. Nela constam os valores de centro de massa, massa e momento de inércia destes.

Tabela 4.1 – Propriedades dos corpos no SolidWorks

(continua)

Corpo		Centro de Massa (mm)	Massa (g)	Momento de Inércia (gmm^2)
Base	**x**	0,89	9126,68	39001054,84
	y	49,63		39420624,80
	z	0,00		75153755,80
Haste	**x**	0,00	989,52	826242.16
	y	305,96		30378637.05
	z	0,00		30810039.63
Braço	**x**	0,00	1282,51	812353.63
	y	480,52		59004364.90
	z	43,64		59477260.07

Tabela 4.1 – Propriedades dos corpos no SolidWorks

(conclusão)

Corpo		Centro de Massa (mm)	Massa (g)	Momento de Inércia (gmm^2)
Aeronave	x	0,00	1453,37	4419642.11
	y	325,23		66679650.27
	z	265,04		69780236.70
Contrapeso	x	0,00	1618,73	859362.16
	y	730,41		859408.12
	z	-267,52		1163964.18

Fonte: Autoria Própria

A transferência dos corpos rígidos do software CAD *Solidworks* para o software CAE *ADAMS* deve ser feita no formato *parasolid*, e a orientação do centro de massa é calculada por matrizes de rotação para ângulos Z-X-Z de Euler. A Tabela 4.2 mostra os eixos principais de inércia e a orientação do centro de massa de cada corpo em ângulos Z-X-Z de Euler.

Tabela 4.2 – Orientação dos corpos nos softwares

Corpo		SolidWorks		Adams
Base	Ix	(1.00, 0.01, 0.00)	Z	-179,42°
	Iy	(0.00, 0.00, -1.00)	X	90,00°
	Iz	(-0.01, 1.00, 0.00)	Z	180,00°
Haste	Ix	(0.00, 1.00, 0.00)	Z	0,00°
	Iy	(-1.00, 0.00, 0.00)	X	0,00°
	Iz	(0.00, 0.00, 1.00)	Z	90,00°
Braço	Ix	(0.00, -0.58, 0.82)	Z	90,00°
	Iy	(0.00, -0.82, -0.58)	X	90,00°
	Iz	(1.00, 0.00, 0.00)	Z	125,27°
Aeronave	Ix	(1.00, 0.00, 0.00)	Z	180,00°
	Iy	(0.00, 0.54, -0.84)	X	57,26°
	Iz	(0.00, 0.84, 0.54)	Z	180,00°
Contrapeso	Ix	(0.00, 0.97, -0.22)	Z	180,00°
	Iy	(-1.00, 0.00, 0.00)	X	12,77°
	Iz	(0.00, 0.22, 0.97)	Z	-90,00°

Fonte: Autoria Própria

A Figura 4.7 apresenta a orientação do centro de massa de cada corpo rígido no software *ADAMS*. A cor vermelha representa a componente X, verde a componente Y e azul a componente Z.

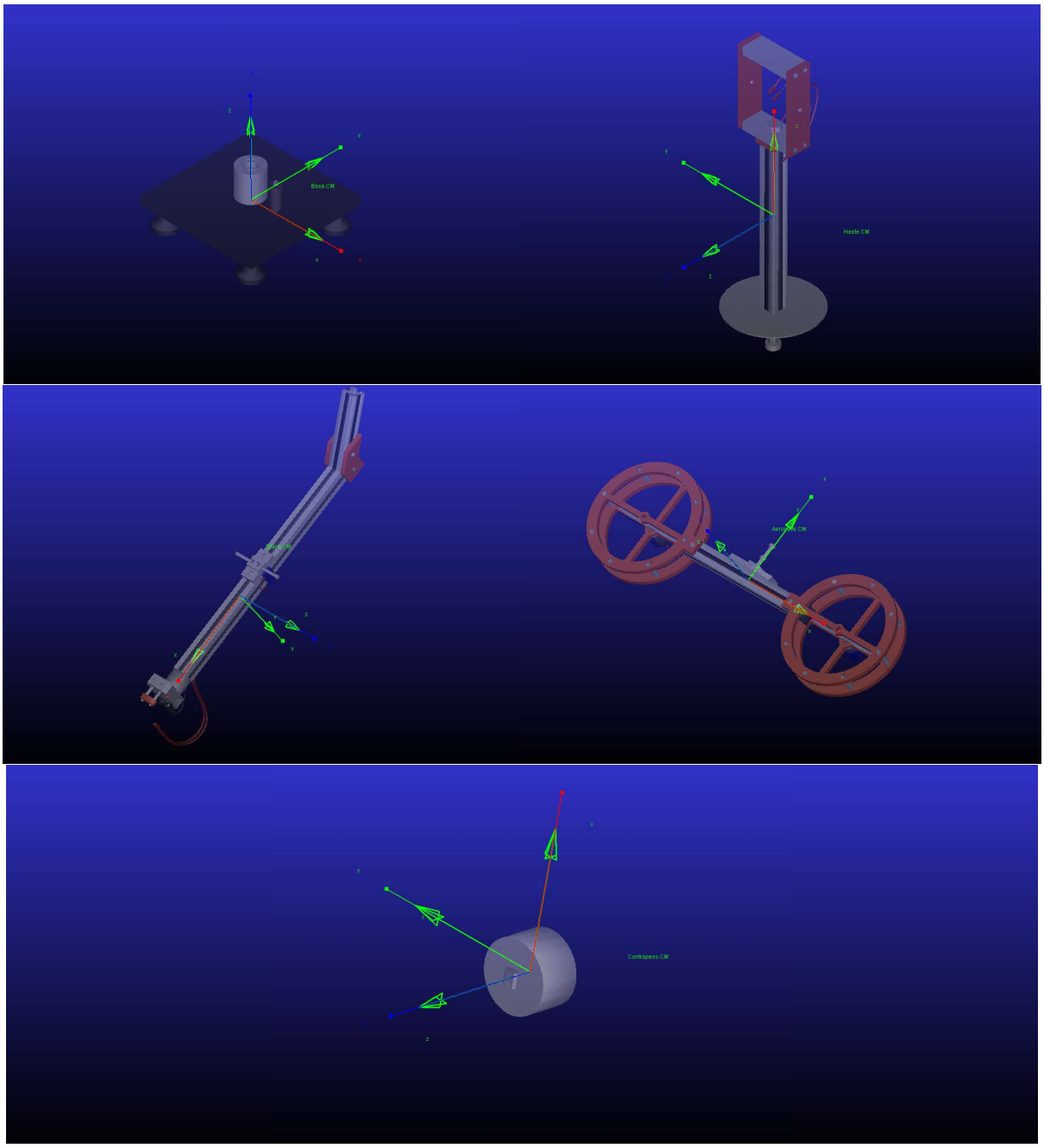


Figura 4.7 – Orientação do centro de massa de cada corpo rígido do Helicóptero de 3GDL
Fonte: Autoria Própria

Ainda no *ADAMS*, para cada corpo rígido deve-se inserir seu movimento ou não de liberdade, e isto é feito com a junta de revolução e junta fixa através do plugin

Adams View. A Figura 4.8 demonstra todas as 5 juntas citadas na lista abaixo, assim como a entrada das forças de empuxo em cada motor.

1. Junta fixa da base com o solo;
2. Junta de revolução da haste com a base (deslocamento);
3. Junta de revolução do braço com a haste (elevação);
4. Junta fixa do braço com o contrapeso;
5. Junta de revolução do braço coma aeronave (arfagem).

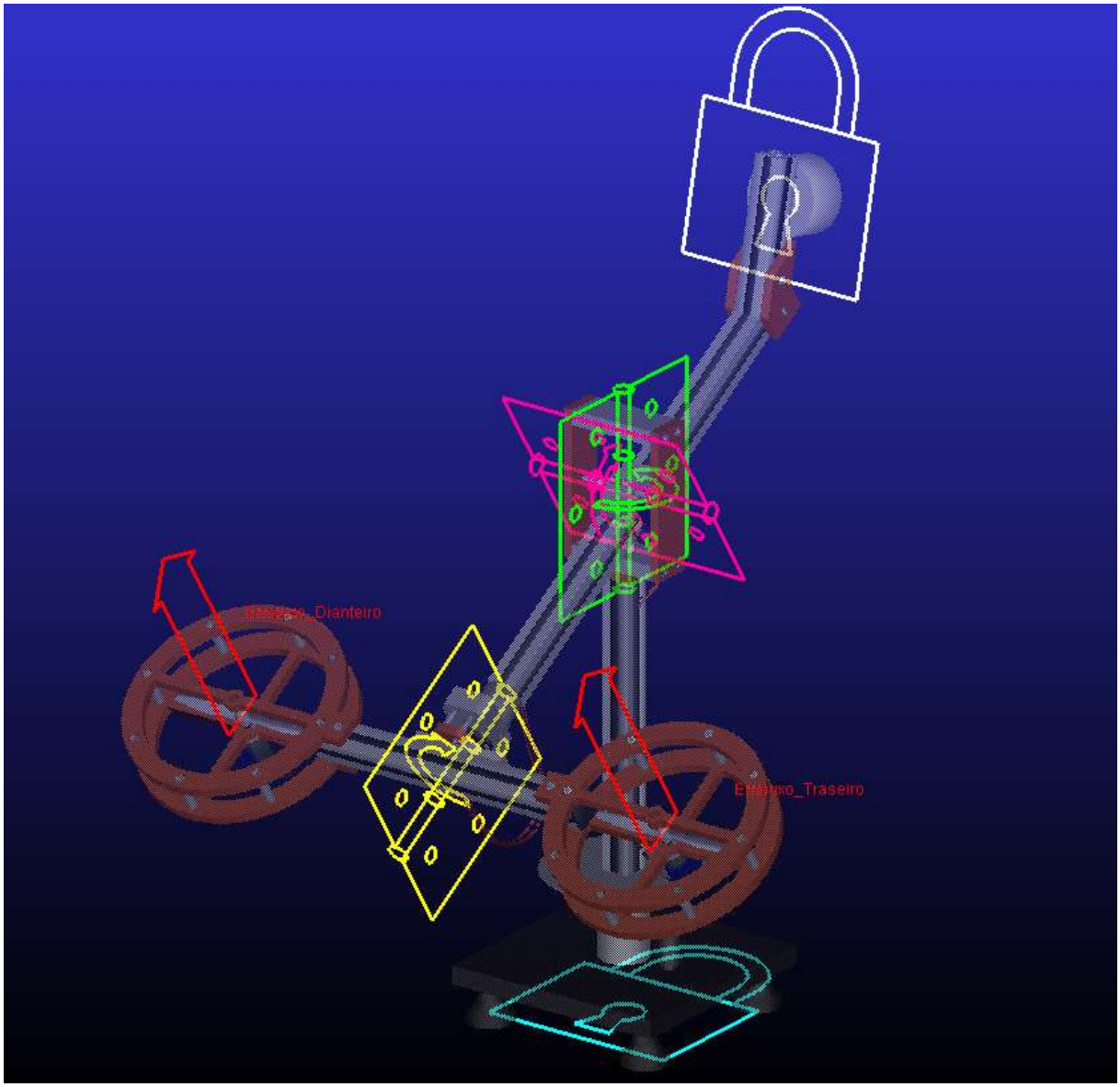


Figura 4.8 – Junta fixa da base com o solo (azul); Junta de revolução de deslocamento (verde); Junta de revolução de elevação (magenta); Junta fixa do braço com comtrapeso (branco); Junta de revolução de arfagem (amarelo)

Fonte: Autoria Própria

Todas as juntas de revolução devem respeitar as convenções de movimento, para que assim sejam geradas as variáveis de entrada: deslocamento y_1, elevação y_2, arfagem y_3, velocidade angular de deslocamento y_4, velocidade angular de elevação y_5 e velocidade angular de arfagem y_6. Assim como as variáveis de controle: empuxo dianteiro u_1 e empuxo traseiro u_2.

Tendo realizados os passos até aqui, deve-se verificar o comportamento do sistema através das análises estática e dinâmica, definindo assim a posição do eixo do braço e contrapeso para menor uso possível dos atuadores. A Tabela 4.3 mostra a configuração do Helicóptero de 3 GDL com as distâncias físicas definidas ao final das análises. Pode-se referenciar cada componente com o diagrama de corpo livre da Figura 3.1.

Tabela 4.3 – Distâncias definidas

Componente	Distância (mm)
L_h	242
L_a	332
L_c.	[0 ; 210,68 ; 245,73]

Fonte: Autoria Própria

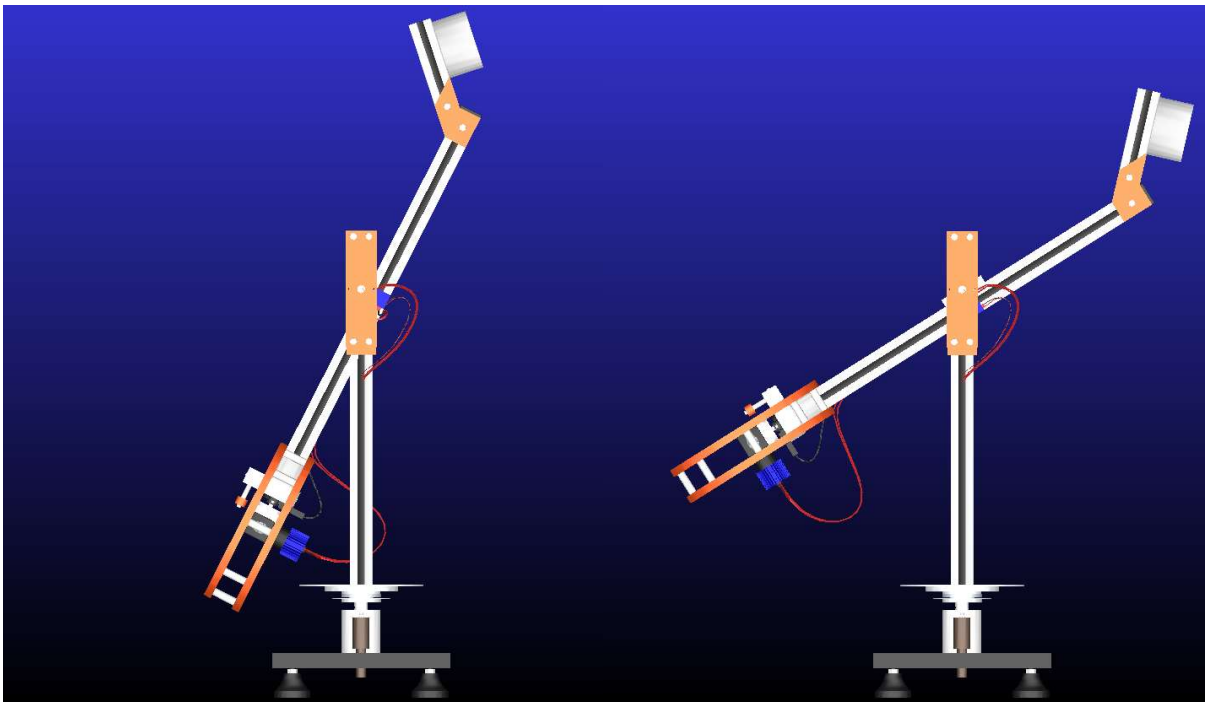


Figura 4.9 – Análise estática (esquerda); Posição inicial de elevação (direita)
Fonte: Autoria Própria

Como não foi aplicada nenhuma força de contato entre os corpos no software *ADAMS*, gera-se uma angulação de elevação fisicamente impossível ($\varepsilon = -30{,}9067°$) na análise estática, conforme visto na Figura 4.9. Deve-se então, linearizar o sistema sob a condição de elevação inicial, ou seja, o ângulo formado pelo contato físico do braço e haste, utilizando as forças dos atuadores. Para isso, seguiu os passos descritos:

1. Gerar o modelo linear em espaço de estados através da análise linear do *ADAMS*;

$$A = \begin{bmatrix} 0 & -0.2433 & 0 & 3.434e^{-15} & 0 & -0.4737 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.05e^{-16} & 0 & -0.3748 & 0 & 19.782e^{-16} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -0.3231 & 0 & 2.802e^{-1} & 0 & -0.6291 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.04918 & -0.04918 \\ 0 & 0 \\ -0.802 & -0.802 \\ 0 & 0 \\ -3.06 & 3.06 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -5.205e^{-16} & 0 & -1 & 0 & -7.459e^{-17} \\ 0 & -0.6066 & 0 & 1.317e^{-16} & 0 & -1.181 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -5.205e^{-16} & 0 & -1 & 0 & -7.459e^{-17} & 0 \\ -0.6066 & 0 & 1.317e^{-16} & 0 & -1.181 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

No qual os estados são definidos como: $x_1 =$ velocidade de deslocamento, $x_2 =$ posição angular de deslocamento, $x_3 =$ velocidade de elevação, $x_4 =$ posição angular de elevação, $x_5 =$ velocidade de arfagem e $x_6 =$ posição angular de arfagem.

2. Aplicar um controle seguidor com realimentação de estado pelo método de atribuição de autoestrutura completa através do *MATLAB/Simulink,* utilizando as matrizes geradas em 1 com ângulo de elevação inicial $\varepsilon_0 = -30{,}9067°$ e referência de elevação $r_\varepsilon = 0°$;

$$\sigma(\boldsymbol{A}+\boldsymbol{BK}) = [-0.90+0.12\mathrm{i} \quad -0.90-0.12\mathrm{i} \quad -0.85+0.11\mathrm{i} \quad -0.85-0.11\mathrm{i} \quad \ldots \\ -0.80+0.10\mathrm{i} \quad -0.80-0.10\mathrm{i} \quad -1.00 \quad -0.95]$$

$$\boldsymbol{K_1} = \begin{bmatrix} -1.9520 & -0.4477 & 1.6521 & 1.2312 & 0.6876 & 1.1173 \\ 1.9520 & 0.4477 & 1.6521 & 1.2312 & -0.6876 & -1.1173 \end{bmatrix}$$

$$\boldsymbol{K_2} = \begin{bmatrix} 0.1810 & 0.4351 \\ -0.1810 & 0.4351 \end{bmatrix}$$

$$\boldsymbol{v} = \begin{bmatrix} 1.9261 & 0.5007 & 0 & 0 & 2.4189 & 0.5543 & 1.6339 & 0 \\ -2.0299 & -0.8270 & 0 & 0 & 2.8919 & -1.0543 & -1.6339 & 0 \\ 0 & 0 & -1.2473 & -0.0523 & 0 & 0 & 0 & -1.1929 \\ 0 & 0 & 1.4354 & 0.2473 & 0 & 0 & 0 & 1.2557 \\ -4.2657 & -0.2448 & 0 & 0 & -4.4805 & -0.2255 & 4.0806 & 0 \\ 4.6212 & 0.8882 & 0 & 0 & 5.4797 & 0.9669 & 4.0806 & 0 \\ -2.0957 & -1.1983 & 0 & 0 & -3.3970 & -1.7425 & -1.6339 & 0 \\ 0 & 0 & -1.6238 & -0.5011 & 0 & 0 & 0 & -1.3218 \end{bmatrix}$$

$$\boldsymbol{q} = \begin{bmatrix} -1 & 0 & -1 & 0 & -1 & 0 & -1 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 & -1 \end{bmatrix}$$

Onde $\sigma(\boldsymbol{A}+\boldsymbol{BK})$ é o espectro de autovalores de malha fechada, $\boldsymbol{K_1}$ ganho de realimentação de estados do controle seguidor, $\boldsymbol{K_2}$ ganho integral do controle seguidor, $\boldsymbol{v}$ e $\boldsymbol{q}$ são os autovetores associados.

3. Gerar o bloco de simulação não linear do *ADAMS* para o *MATLAB/Simulink*;
4. Com os ganhos calculados em 2, verificar o controle sobre o sistema não linear de 3. Caso resultado seja não satisfatório, resultando em instabilidade, voltar em 2 e alocar diferentes autovalores;
5. Identificar as forças de empuxo para equilíbrio na posição inicial de elevação (u_{op});

$$u_{op} = \begin{bmatrix} u_{op1} \\ u_{op2} \end{bmatrix} = \begin{bmatrix} 0.12 \\ 0.12 \end{bmatrix}$$

6. Aplicar as intensidades de empuxo identificadas sobre uma nova análise linear no ADAMS, gerando o modelo matemático linear final.

$$A = \begin{bmatrix} 0 & -1.085e^{-16} & 0 & 9.095e^{-14} & 0 & -0.3653 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -9.281e^{-18} & 0 & -0.3216 & 0 & 1.098e^{-1} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -3.085e^{-16} & 0 & -3.447e^{-14} & 0 & -1.128 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.02453 & -0.02453 \\ 0 & 0 \\ -0.802 & -0.802 \\ 0 & 0 \\ -3.061 & 3.061 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1.225e^{-1} & 0 & -1 & 0 & -7.317e^{-14} \\ 0 & 1.11e^{-16} & 0 & 8.625e^{-14} & 0 & -1.181 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1.225e^{-16} & 0 & -1 & 0 & -7.317e^{-14} & 0 \\ 0 & 0 & 8.625e^{-14} & 0 & -1.181 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

5 IDENTIFICAÇÃO DOS ATUADORES

Neste trabalho, cada haste do Helicóptero de 3 GDL contém um motor de corrente contínua A380/PL24-21600S da fabricante *Akiyama* (Joinville, Brasil). Estes motores aumentam sua velocidade com o aumento de tensão aplicada, variando entre 0V e 24V. Para o controle desta tensão, é utilizado um driver de potência SX8847 da *Sonxun* (Pequim, China), através de um sinal PWM, onde a tensão enviada aos motores é proporcional à razão cíclica enviada a este driver. Deve-se então relacionar este sinal PWM com o valor de empuxo das variáveis de controle do modelo com um polinômio. Para isso, antes deve-se calcular um polinômio de calibração da célula de carga S10 da *Alfa Instrumentos* (São Paulo, Brasil). A Figura 5.1 representa o processo de calibração da curva da célula de carga.

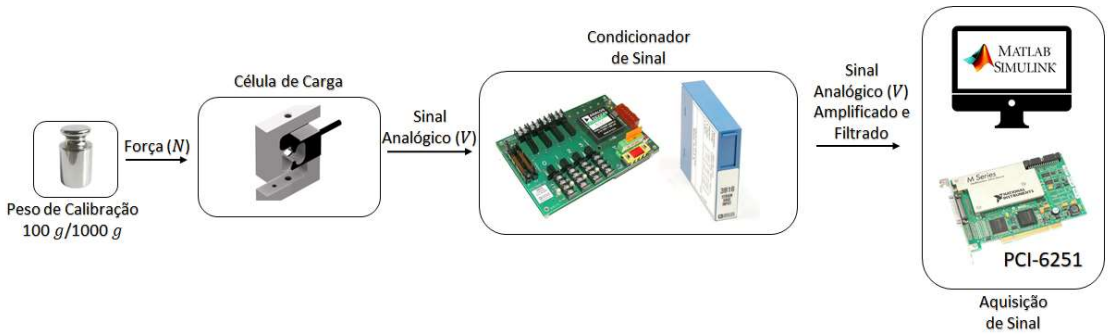


Figura 5.1 – Processo para identificação dos polinômios dos atuadores
Fonte: Autoria própria

De acordo com a Figura 5.1, são utilizados pesos de calibração com massas de 100 g a 1000 g, fazendo incrementos de 100 g. Cada uma dessas massas gera uma perturbação diferente na célula de carga, na qual responde com um sinal analógico de tensão correspondente. Como este sinal é composto por valores de pequena amplitude contendo ruído, deve-se realizar um condicionamento de sinal, filtrando e amplificando o sinal da célula de carga. Para isso, utiliza-se um módulo de condicionamento de sinais 3B16 conectado na placa 3B03, ambos da *Analog Devices* (Norwood, EUA). Este sinal de tensão condicionado é recebido pelo *MATLAB/Simulink* por meio da placa PCI-6251 da *National Instruments* (Austin, EUA), e através de um algoritmo é realizada a relação deste sinal com a força (N). A Figura 5.2 mostra o experimento correspondente sendo realizado em laboratório.

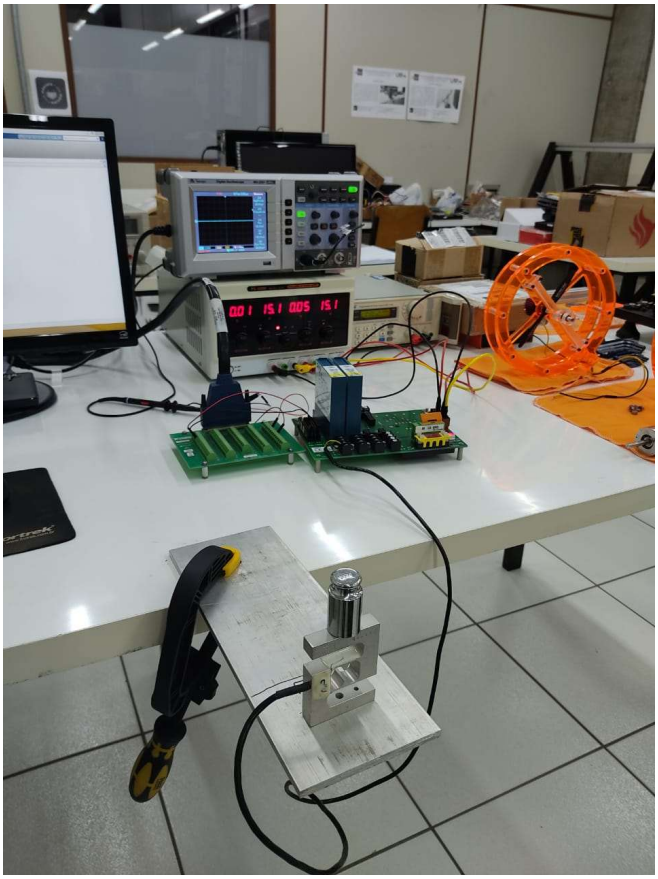


Figura 5.2 – Processo de calibração da célula de carga em laboratório
Fonte: Autoria própria

A fim de analisar a captação dos sinais de tensão provenientes das cargas estáticas (100 g a 1000 g), foram realizados dois experimentos, onde cada ponto de calibração, ou ponto de incremento de massa, foram coletados com frequência de 100 Hz e 800 Hz por um período de 5 s, dados estes programados pelo *MATLAB/Simulink.* A Figura 5.3 mostra alguns dos sinais obtidos para a frequência de 100 Hz e a Figura 5.4 alguns dos resultados para a frequência de 800 Hz. Importante citar que o sinal de tensão para 0 N foi aquisitado, e todos os valores de tensão subsequentes são subtraídos por este, a fim de remover a polarização (offset) da leitura proveniente da célula de carga. Nenhum filtro foi utilizado nesta etapa de aquisição.

A Tabela 5.1 apresenta as médias de cada ponto de calibração dos dois experimentos. Foi identificado um erro nas medidas na ordem de 10^{-4}. No restante deste trabalho serão considerados apenas dados provenientes do experimento na frequência de 800 Hz.

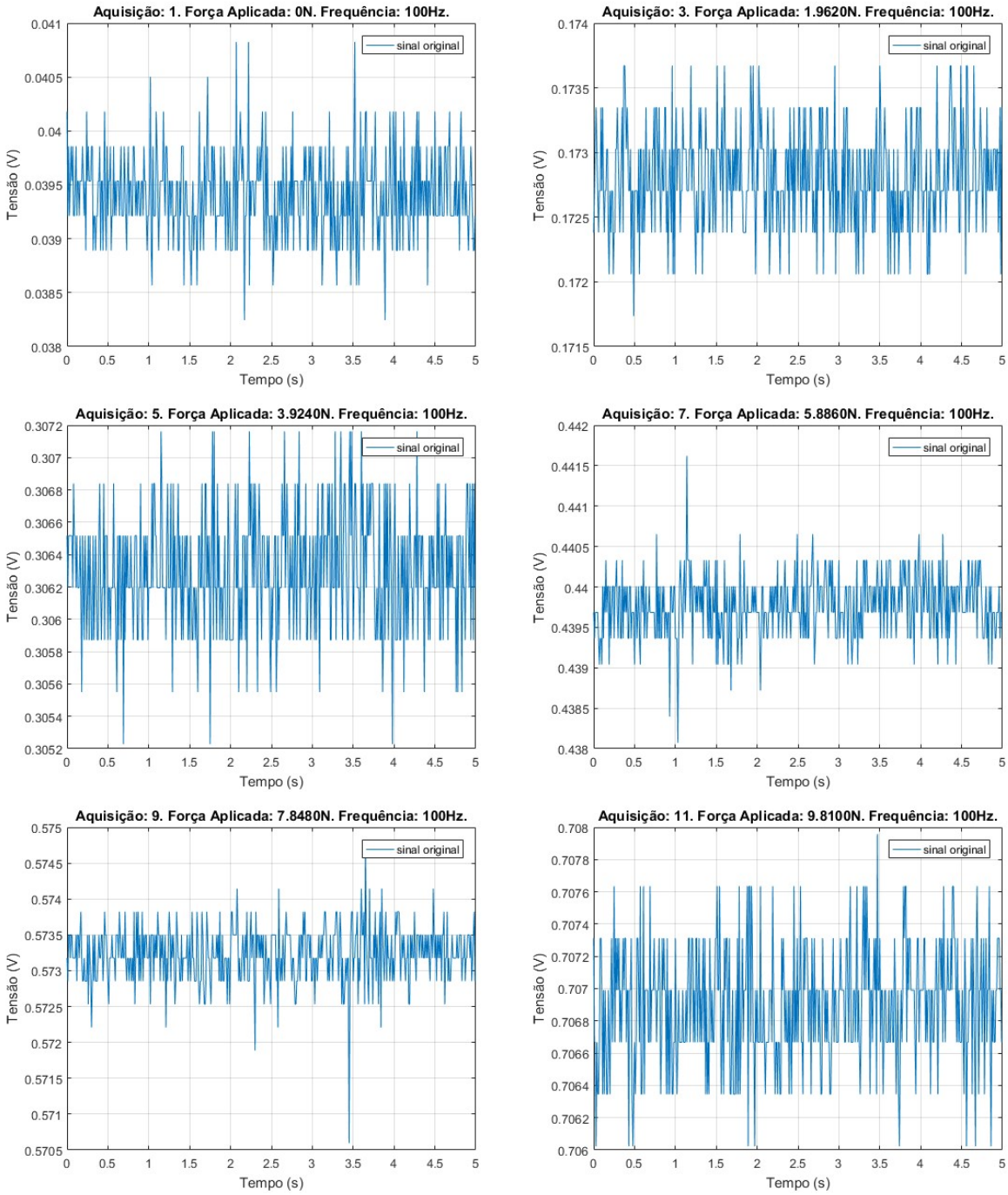


Figura 5.3 – Sinais aquisitados para uma frequência de 100 Hz
Fonte: Autoria própria

A Figura 5.5 mostra o gráfico da curva de calibração da célula de carga, onde estão dispostos os pontos de calibração para cada incremento de massa, sendo estes uma média dos valores apresentados na Tabela 5.1 para a frequência de 800 Hz. A figura mostra que esses pontos podem ser representados por uma reta, calculando o polinômio de calibração y_c dado pela Equação (5.1).

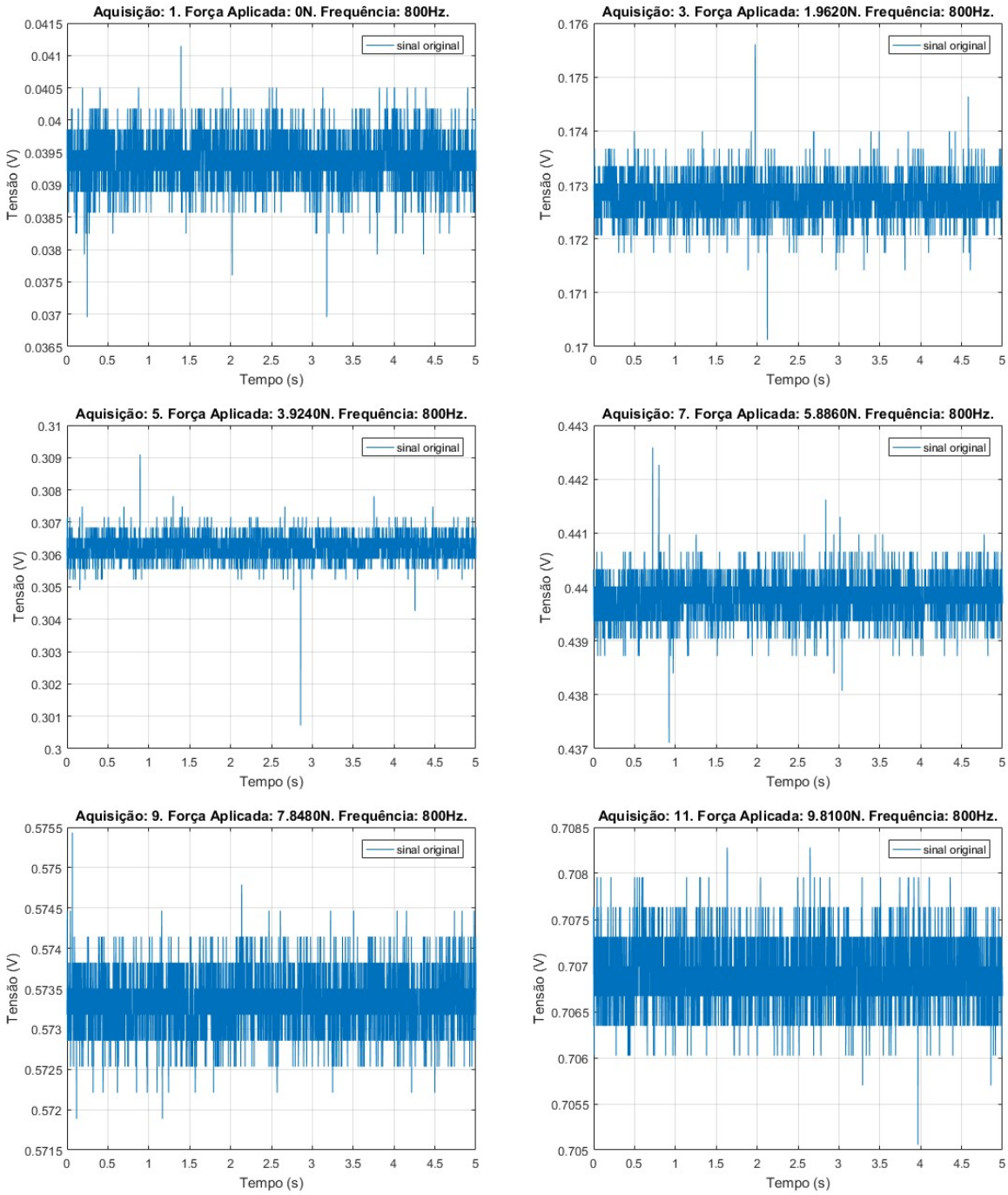


Figura 5.4 – Sinais aquisitados para uma frequência de 800 Hz
Fonte: Autoria própria

Tabela 5.1 - Média dos pontos de calibração para cada experimento

Força Aplicada (N)	Média (V) (100 Hz)	Média (V) (800 Hz)	Erro (V)
0	0	0	0
0.9810	0.0667	0.0667	0
1.9620	0.1334	0.1334	0
2.9430	0.2002	0.2002	0
3.9240	0.2669	0.2668	0.0001
4.9050	0.3337	0.3337	0
5.8860	0.4004	0.4004	0
6.8670	0.4671	0.4672	0.0001
7.8480	0.5338	0.5340	0.0002
8.8290	0.6006	0.6007	0.0001
9.8100	0.6675	0.6676	0.0001

Fonte: Autoria Própria

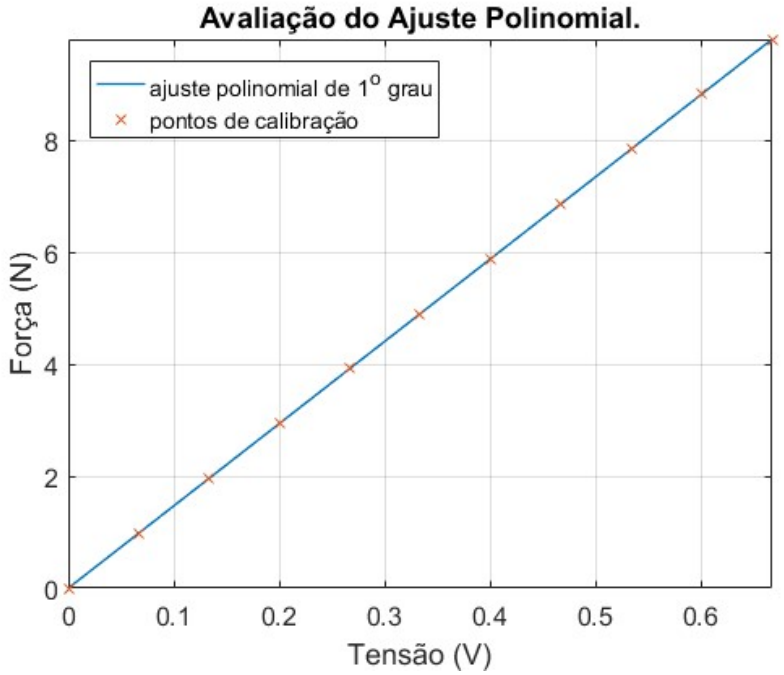


Figura 5.5 – Curva de calibração da célula de carga
Fonte: Autoria própria

$$y_c = 14.6960x + 0.0010 \quad (5.1)$$

5.1 IDENTIFICAÇÃO EMPUXO[2]

De acordo com a Figura 5.6, usando o *MATLAB/Simulink* configura-se um sinal PWM com a razão cíclica desejada, e pelo barramento PCI envia-se este sinal com a placa de aquisição PCI-6251 para o driver de potência. O driver gera em sua saída uma tensão proporcional ao valor do PWM recebido. Dependendo da carga recebida pelos motores, deforma-se a célula de carga, enviando um sinal analógico de tensão, que deve também ser condicionado. Por fim, aquisita-se o valor de tensão, podendo realizar no *MATLAB/Simulink* o cálculo da razão cíclica do PWM com o empuxo gerado pelo motor.

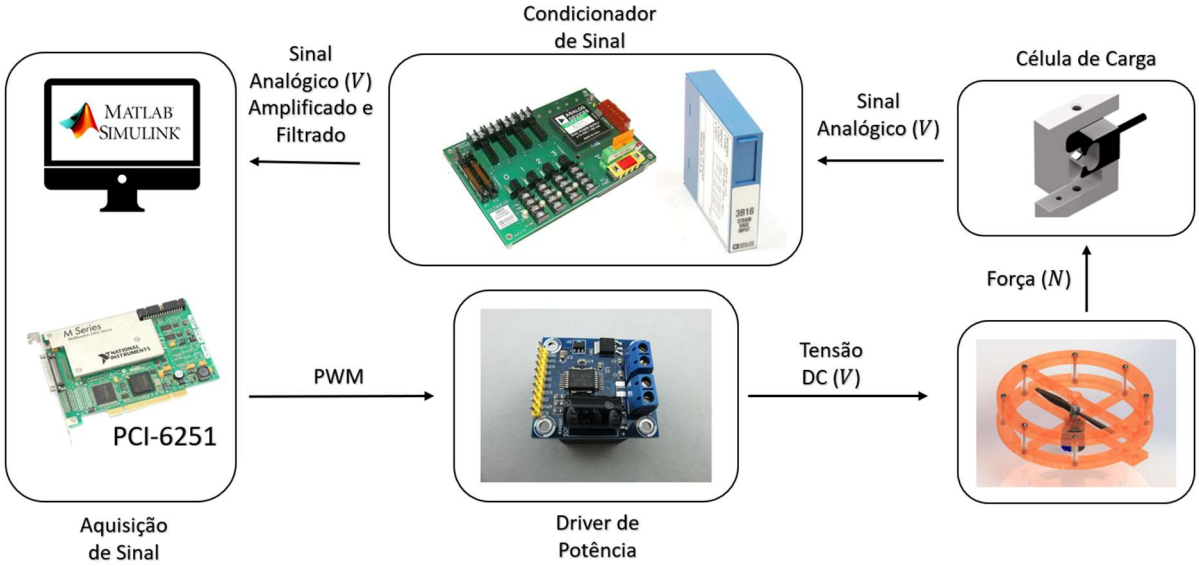


Figura 5.6 – Processo para identificação dos polinômios dos atuadores.
Fonte: Autoria própria.

O algoritmo criado para excitar o motor gera um sinal PWM de 200 Hz, realizando incrementos de 10% na razão cíclica, num intervalo entre 10% e 80%, protegendo o motor de superaquecimento, e logo após, os decrementos, gerando uma onda quadrada, onde cada degrau tem a duração de 5 s. Este sinal para controle da razão cíclica pode ser visto na Figura 5.7.

A aquisição da média dos valores correspondentes a cada razão cíclica, tanto na subida, quanto na descida, gera informação para o ajuste polinomial de $3°$ grau.

[2] Vídeo do experimento: <https://www.youtube.com/watch?v=k0BO-jfB0S0>. Acessado em 21 de setembro de 2021.

Para maior confiabilidade na aproximação, foram realizados 20 ajustes polinomiais para cada motor. A Figura 5.8 mostra as curvas de ajuste obtidas para os motores dianteiro e traseiro.

Verificando um grau de dispersão entre as curvas vistas na Figura 5.8, calcula-se a estatística da média e desvio padrão. Esta análise pode ser vista na Figura 5.9 para ambos os motores. Os dados desta etapa podem ser visualizados na Tabela 5.2.

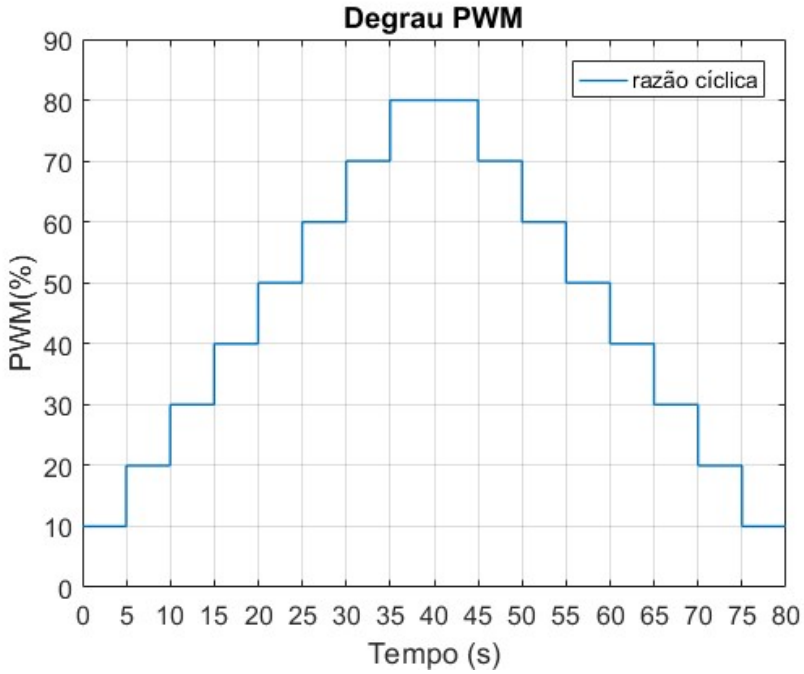


Figura 5.7 – Progressão da razão cíclica para identificação dos atuadores
Fonte: Autoria própria.

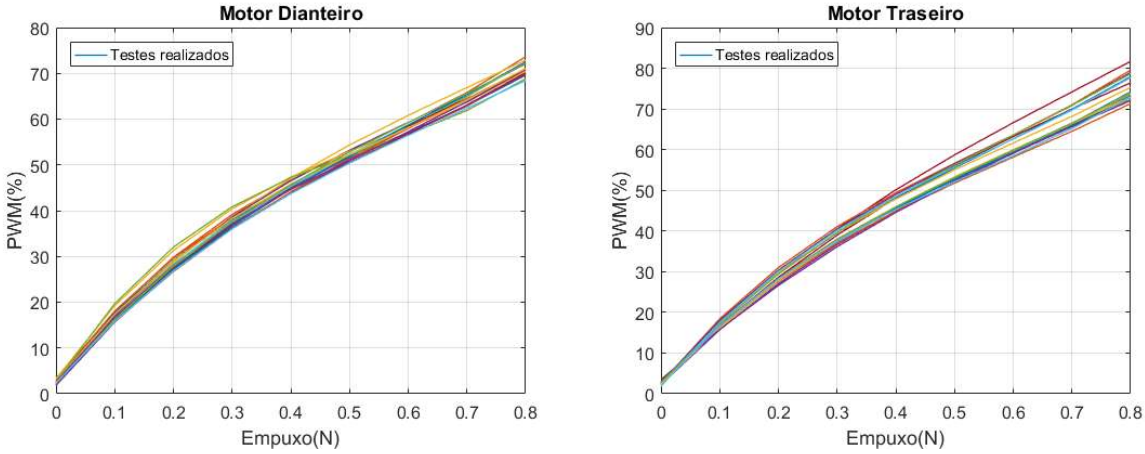


Figura 5.8 – Ajustes polinomiais dos motores dianteiro e traseiro
Fonte: Autoria própria

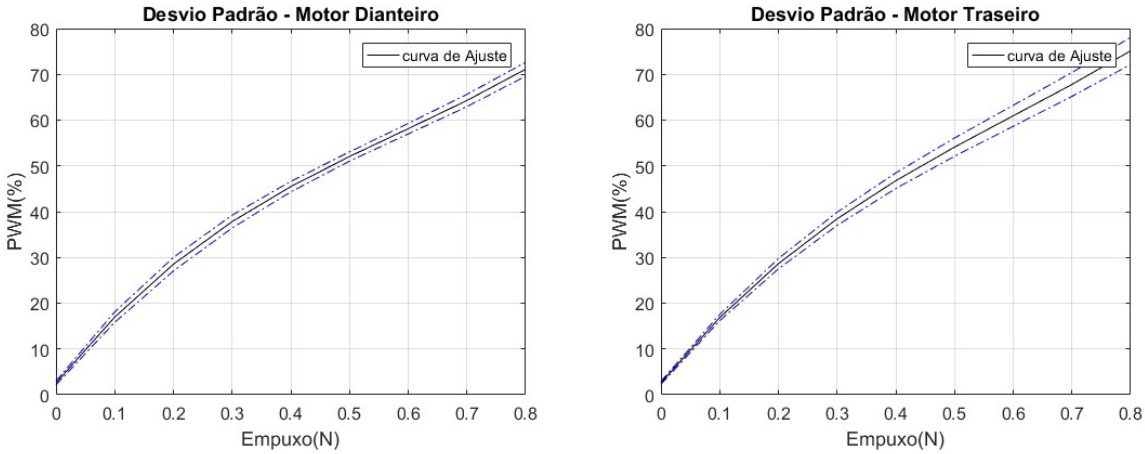


Figura 5.9 – Desvio padrão das curvas de calibração dos motores dianteiro e traseiro
Fonte: Autoria própria

Tabela 5.2 – Dados da dispersão entre curvas

	Motor Dianteiro		Motor Traseiro	
Empuxo (N)	**Média Aritmética - PWM (%)**	**Desvio Padrão (%)**	**Média Aritmética - PWM (%)**	**Desvio Padrão (%)**
0	2.5774	0.4200	2.6150	0.3276
0.1	16.9586	1.1438	16.8572	0.6867
0.2	28.5144	1.4793	28.6347	1.1390
0.3	37.8272	1.4029	38.4401	1.4371
0.4	45.4792	1.1474	46.7659	1.7010
0.5	52.0527	1.0116	54.1048	1.9905
0.6	58.1297	1.1410	60.9493	2.2978
0.7	64.2927	1.3413	67.7920	2.6054
0.8	71.1238	1.5183	75.1255	2.9679

Fonte: Autoria Própria

Realizando o ajuste polinomial, respeitando os limites do desvio padrão e procurando a melhor correspondência com a curvas de ajuste da Figura 5.9, obtém-se o polinômio de identificação do motor dianteiro y_d e do motor traseiro y_t.

$$y_d = 97.04x^3 - 170.38x^2 + 159.88x + 2.5774 \quad (5.2)$$

$$y_t = 82.099x^3 - 147.87x^2 + 156.39x + 2.615 \quad (5.3)$$

Uma imagem ilustrando a identificação dos motores no laboratório pode ser vista na Figura 5.10. Na imagem pode-se observar um suporte desenvolvido para fixar o motor sobre a célula de carga, e nesta configuração, todos os valores de tensão gerados pela célula de carga, devido ao empuxo dos motores, são negativos.

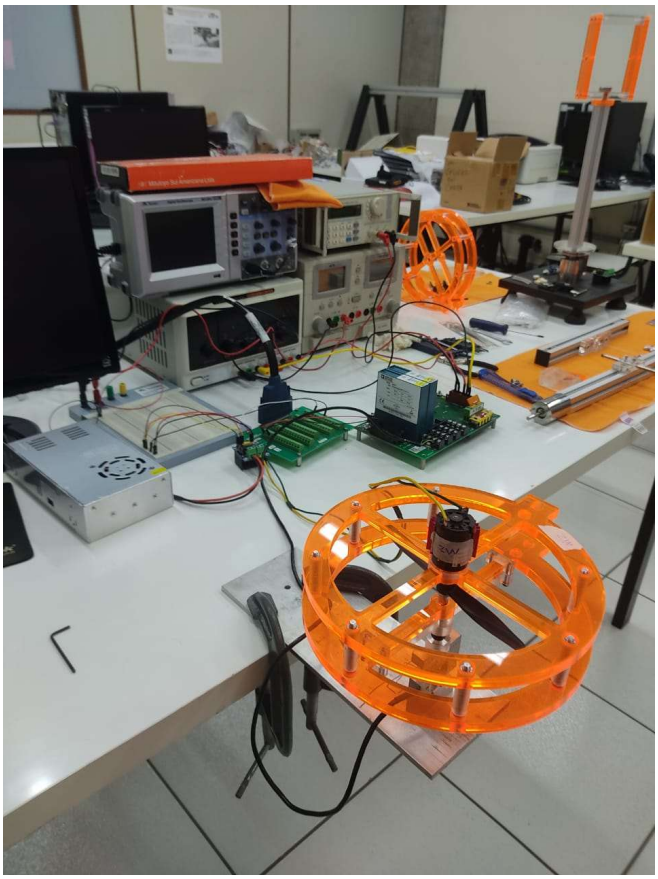


Figura 5.10 – Processo de identificação dos motores em laboratório
Fonte: Autoria própria

6 SENSORES E ESTIMAÇÃO

Para a determinação das posições angulares do Helicóptero de 3 Graus de Liberdade, foram utilizados encoders ópticos do tipo incremental. O encoder óptico possui um disco de vidro ou plástico, contendo uma sequência de marcações transparentes e opacas no decorrer de sua circunferência. Essas marcações são utilizadas para detectar a rotação do disco, dada a utilização de LEDs emissores de luz numa face e sensores ópticos na face oposta. Assim, quando o disco está em rotação, os sensores ópticos respondem com pulsos elétricos provenientes da excitação da luz emissora através de uma marcação transparente (MATOS, 2012; ROSÁRIO, 2006). Quanto maior o número de marcações, maior será a resolução por volta.

O encoder óptico incremental possui o mesmo princípio de funcionamento, utilizando 3 canais, conforme pode ser visto na Figura 6.1. Os sensores dos canais A e B são defasados em 90°, permitindo o aumento de resolução e a determinação da direção (quadratura). O sensor do canal Z é responsável por determinar um ponto de início/fim de uma volta (ROSÁRIO, 2006).

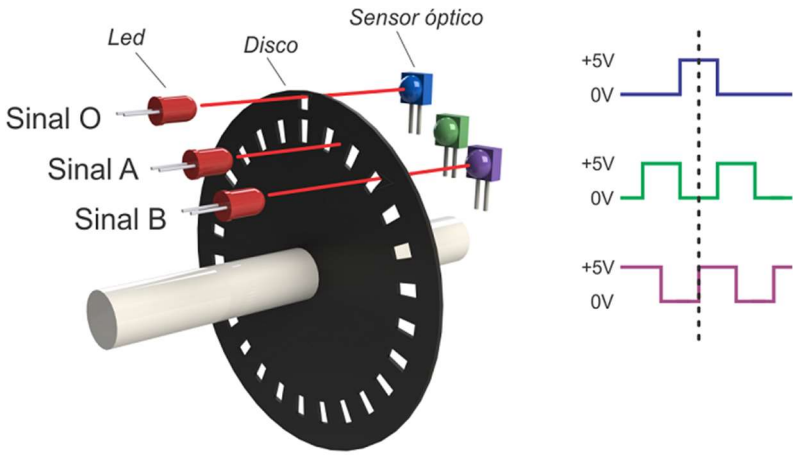


Figura 6.1 – Funcionamento do encoder óptico incremental
Fonte: Adaptado de HI Tecnologia

Encoders de quadratura são do tipo incrementais, ou seja, informam apenas variações relativas à posição através de um par de trens de pulso dados pelos canais A e B. O canal Z é um indexador, e gera um pulso a cada revolução completa. Na

instrumentação, um contador de pulsos pode ser incrementado ou decrementado com a leitura dos canais A e B, tendo seu valor inicial de referência “0” ao pulso do canal indexador (ROSÁRIO, 2006; SACHS, 2013). Codificadores são encontrados na maioria dos microcontroladores atuais, porém para este trabalho, sendo uma bancada de controle, foi utilizado o contador existente na própria placa PCI-6602 da *National Instruments* (Austin, EUA).

6.1 SENSORES UTILIZADOS

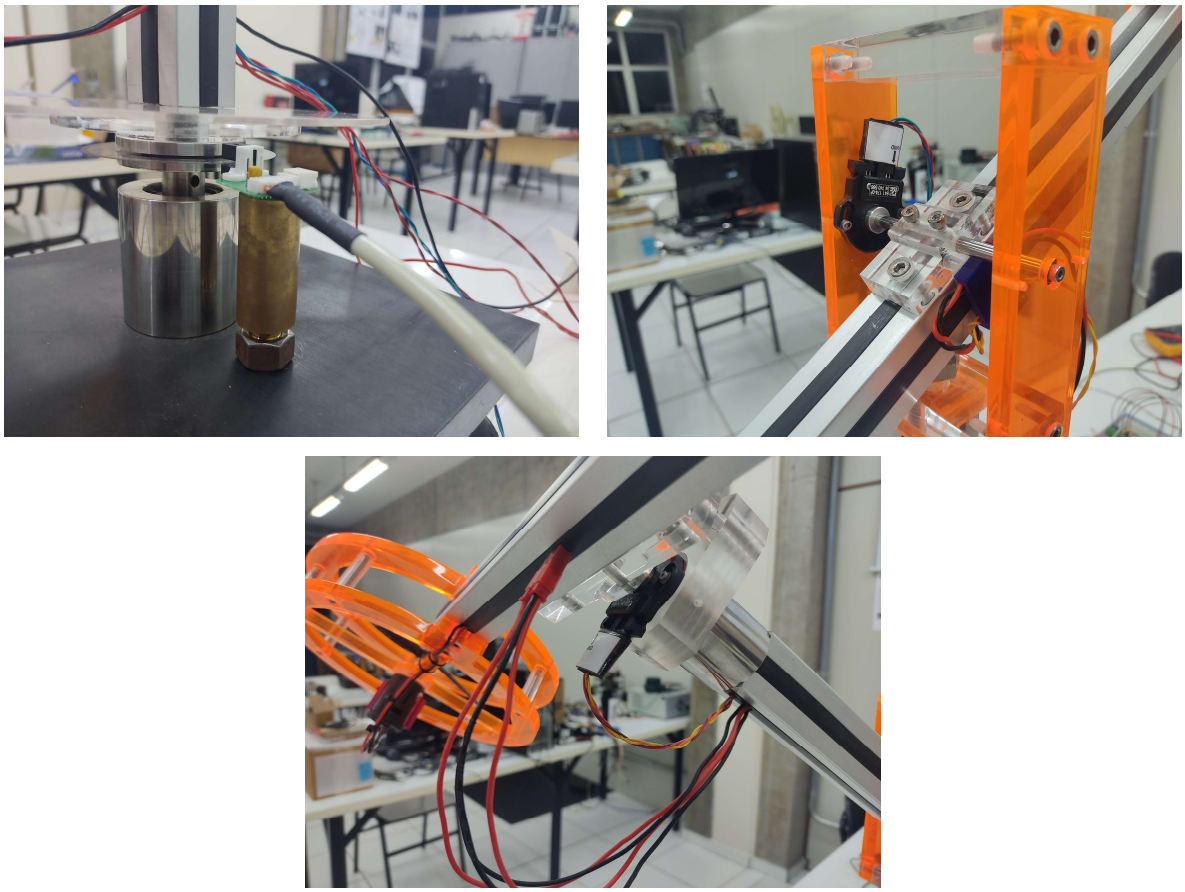


Figura 6.2 – Encoders dispostos no Helicóptero de 3 Graus de Liberdade
Fonte: Autoria própria

Os encoders ópticos incrementais utilizados no Helicóptero de 3 Graus de Liberdade foram dois. O RCML15 da fabricante *RENCO* (Traunreut, Alemanha) e o Q9863 da fabricante *HP* (Palo Alto, EUA). A Tabela 6.1. apresenta características

adicionais destes encoders e a Figura 6.2 mostra a instalação em cada um dos eixos de liberdade do helicóptero.

Tabela 6.1 – Características dos encoders da planta do Helicóptero de 3 Graus de Liberdade

Modelo	Marca	Tensão de Operação	Pulos Por Revolução	GDL	Resolução Quadratura
RCML15	RENCO	5 V	2048	Elevação/Arfagem	0,044°
Q9863	HP	5 V	1800	Deslocamento	0,05°

Fonte: Autoria Própria

6.2 TRACKING LOOP

O encoder de quadratura usado neste trabalho mede somente a posição angular. Para obter a medida da velocidade angular, deve-se utilizar técnicas de estimação de velocidade.

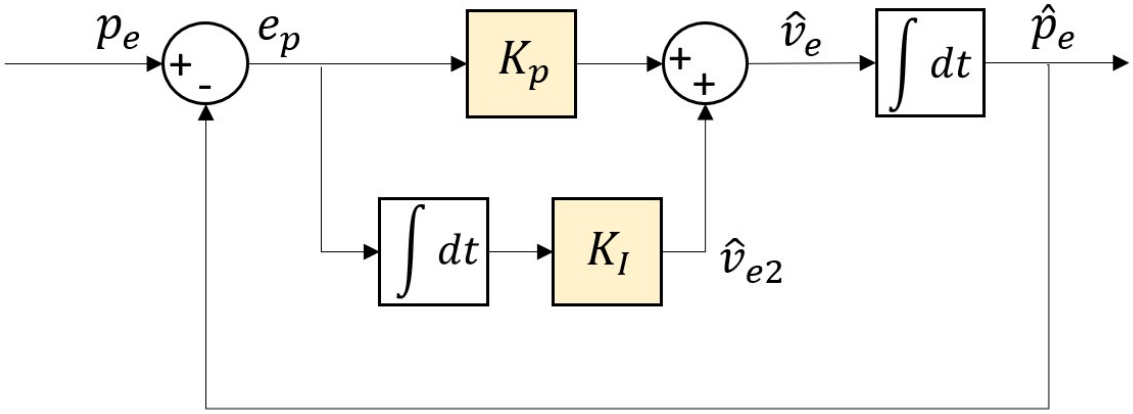


Figura 6.3 – Tracking Loop com implemetação através de um sistema realimentado com um controle PI e integrador na saída
Fonte: Autoria própria

Nota-se, porém, que os codificadores incrementais são digitais e possuem um efeito de erro de quantização. Erros de quantização podem gerar ruídos na estimação da velocidade, além de possivelmente incrementar ruído na leitura em torno da

velocidade nula. Para contornar tais problemas, um método de estimação utilizado na área de controle é o Tracking Loop. O seu esquemático de funcionamento pode ser visto na Figura 6.3: percebe-se um controlador PI com ganhos proporcional (K_p) e integral (K_i) e um integrador na saída. Sendo p_e a posição angular real dada pelo encoder, $\hat{p}_e$ a posição angular estimada, e_p o erro entre posição angular real e estimada, $\hat{v}_e$a velocidade angular estimada e $\hat{v}_{e2}$ a velocidade angular estimada dada uma integração (LEE; SONG, 2001; HIDEKI, 2020).

Através da calibração dos ganhos proporcional (K_p) e integral (K_i), tem-se duas velocidades estimadas na saída do controlador, sendo $\hat{v}_{e2}$ menos ruidosa que $\hat{v}_e$ devido a presença do integrador.

6.3 EXPERIMENTO PARA VALIDAÇÃO

Para a validação do método Tracking Loop, foi utilizado o esquemático da Figura 6.4. Como são utilizados dois tipos de encoders diferentes, foi aplicada uma variação na posição angular de arfagem para o RCML15 e uma variação na posição angular de deslocamento para o Q9863. A codificação do sinal é realizada com a placa de aquisição PCI-6602, e a interface MATLAB/Simulink é utilizada para o tratamento dos dados.

Hideki (2020) com uma planta similar, e utilizando o Tracking Loop para estimação da velocidade angular do Helicóptero de 3 Graus de Liberdade, adotou os ganhos $K_p = 40$ e $K_i = 900$. Tomando como base estes valores, foram realizados os experimentos dispostos na Tabela 6.2 para ambos os encoders.

Na aquisição foi utilizada uma taxa de 200 Hz, mantendo assim, a frequência configurada no PWM dos motores, o qual é a variável de entrada de controle, e criando uma resolução máxima de 5 ms na quantificação dos encoders. A variação nos movimentos de arfagem e deslocamento foram realizados através de um controlador seguidor de posição, garantido o mesmo movimento para cada configuração de teste. Na Figura 6.5 (a), encontra-se a variação realizada para o movimento angular de arfagem, e na Figura 6.5 (b) para o movimento de deslocamento, ambos com duração de 10 s.

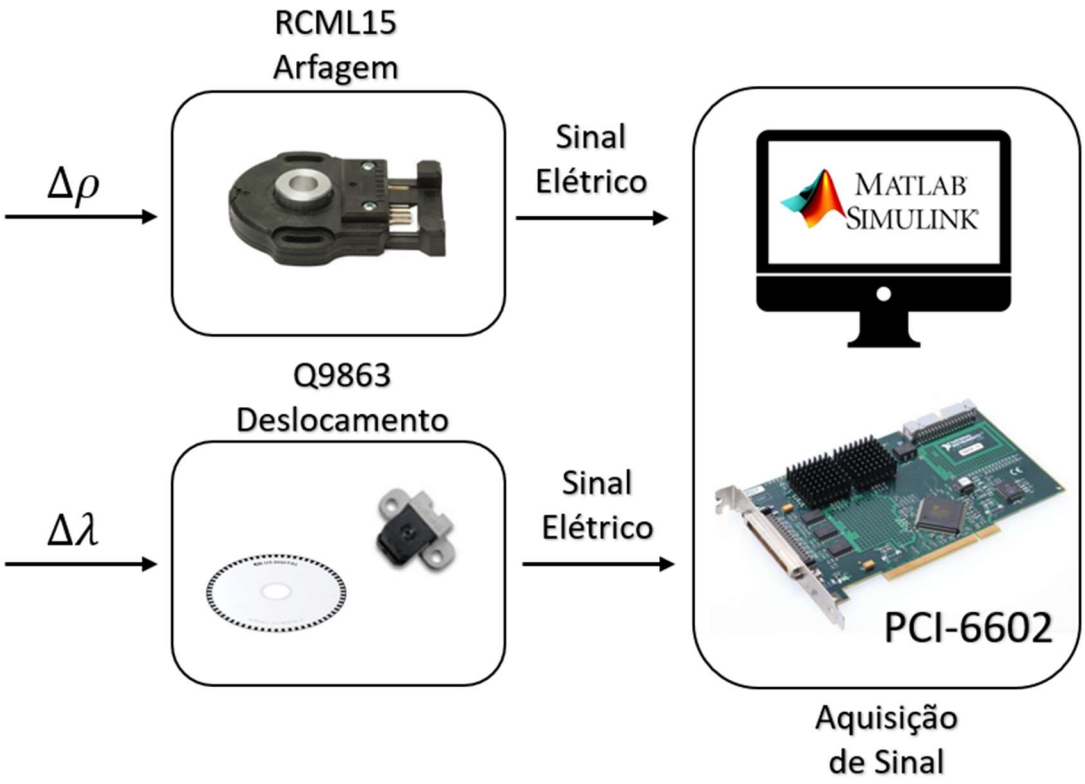


Figura 6.4 – Experimento para validação doTracking Loop
Fonte: Autoria própria

Na Figura 6.6 (a) é demonstrado o resultado da aplicação do Tracking Loop para um ganho proporcional $K_p = 20$ e ganho integral $K_i = 800$, em um intervalo de 1 s dentro da variação total de 10 s do movimento angular de arfagem. Para comparação dos resultados e validação do método, foram apresentadas as velocidades angulares por derivação $\dot{p}_e$ e as duas saídas do Tracking Loop $\dot{v}_{\mathrm{e}}$e $\dot{v}_{\mathrm{e2}}$.

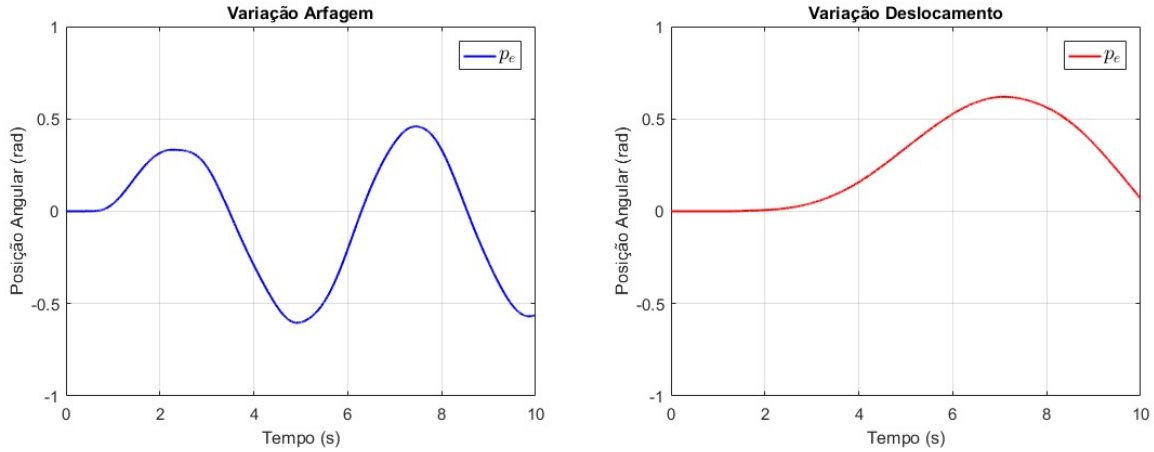
As Figuras 6.6 (b), (c) e (d) fazem a mesma análise dentro do mesmo intervalo de variação do movimento angular de arfagem, aplicando, respectivamente, as configurações: ($K_p = 20$ e $K_i = 1200$), ($K_p = 60$ e $K_i = 800$) e ($K_p = 60$ e $K_i = 1200$).

As Figuras 6.7 (a), (b), (c) e (d) demonstram também o resultado do Tracking Loop, porém para a variação do movimento de deslocamento dentro de um intervalo de 1 s dos 10 s aquisitados. As configurações, respectivamente, são: ($K_p = 20$ e $K_i = 800$), ($K_p = 20$ e $K_i = 1200$), ($K_p = 60$ e $K_i = 800$) e ($K_p = 60$ e $K_i = 1200$).

Tabela 6.2 – Experimentos com diferentes tipos de ganhos K_pe K_i.

Experimento	K_p	K_i
1	20	800
2	20	900
3	20	1000
4	20	1100
5	20	1200
6	30	800
7	30	900
8	30	1000
9	30	1100
10	30	1200
11	40	800
12	40	900
13	40	1000
14	40	1100
15	40	1200
16	50	800
17	50	900
18	50	1000
19	50	1100
20	50	1200
21	60	800
22	60	900
23	60	1000
24	60	1100
25	60	1200

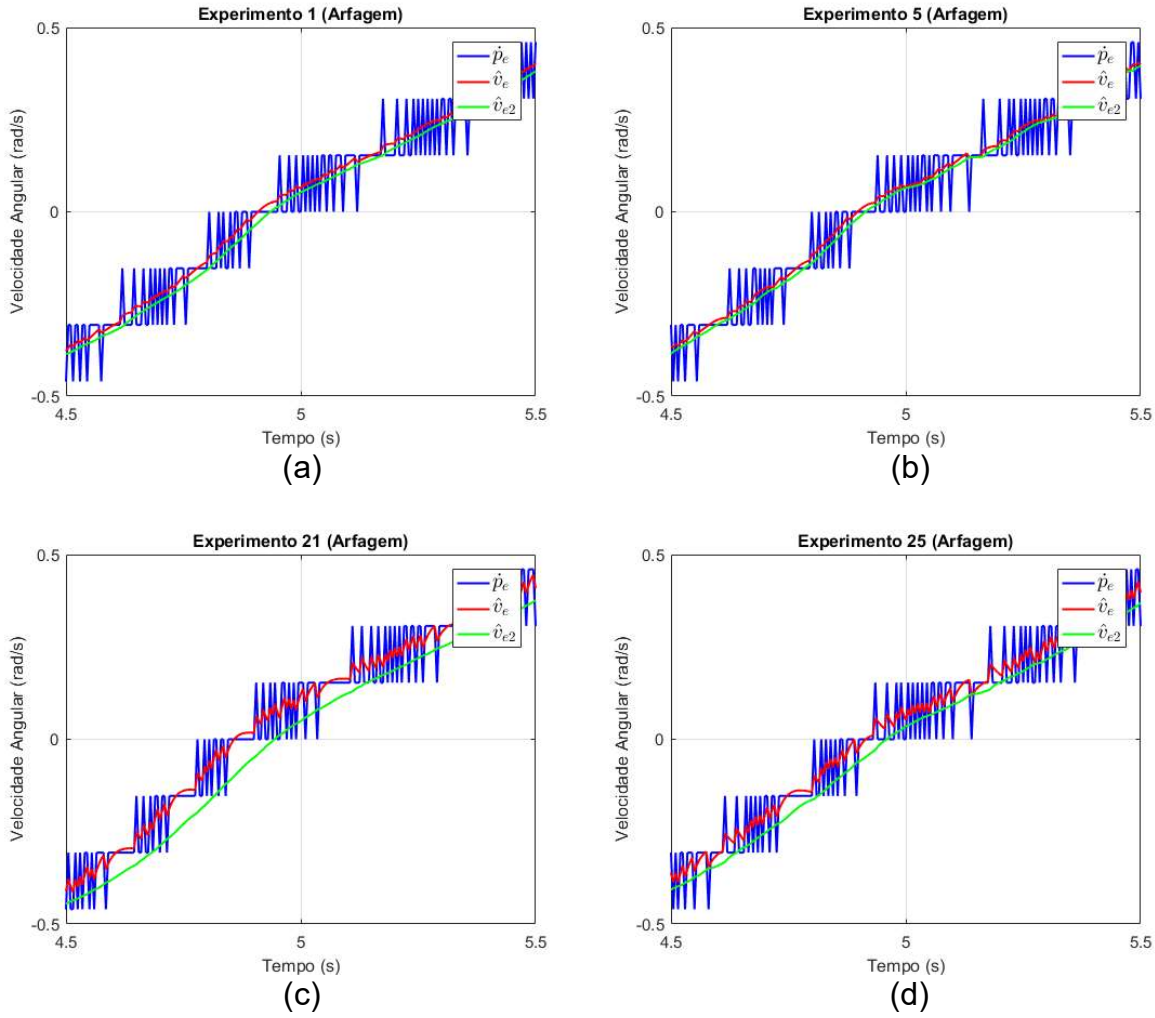
Fonte: Autoria Própria



(a) (b)

Figura 6.5 – Variação no movimento de arfagem (a) e deslocamento (b). Dado real coletado no experimento

Fonte: Autoria própria



(a) (b)

(c) (d)

Figura 6.6 – Experimentos para o encoder de arfagem. (a) Experimento 1 ($K_p = 20, K_i = 800$); (b) Experimento 5 ($K_p = 20, K_i = 1200$); (c) Experimento 21 ($K_p = 60, K_i = 800$); (d) Experimento 25 ($K_p = 60, K_i = 1200$)

Fonte: Autoria própria

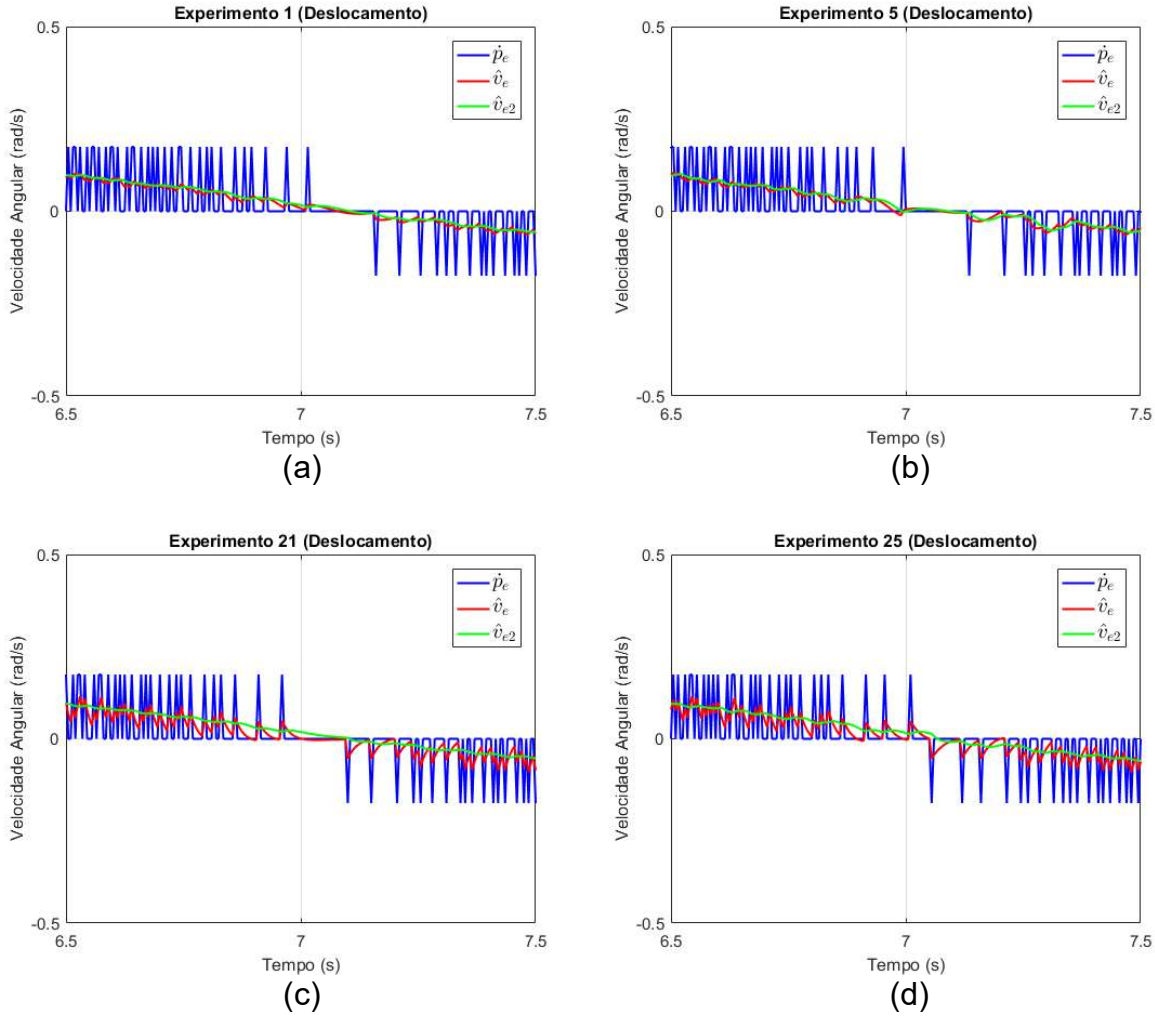


Figura 6.7 – Experimentos para o encoder de deslocamento (a) Experimento 1 ($K_p = 20, K_i = 800$); (b) Experimento 5 ($K_p = 20, K_i = 1200$); (c) Experimento 21 ($K_p = 60, K_i = 800$); (d) Experimento 25 ($K_p = 60, K_i = 1200$)

Fonte: Autoria própria

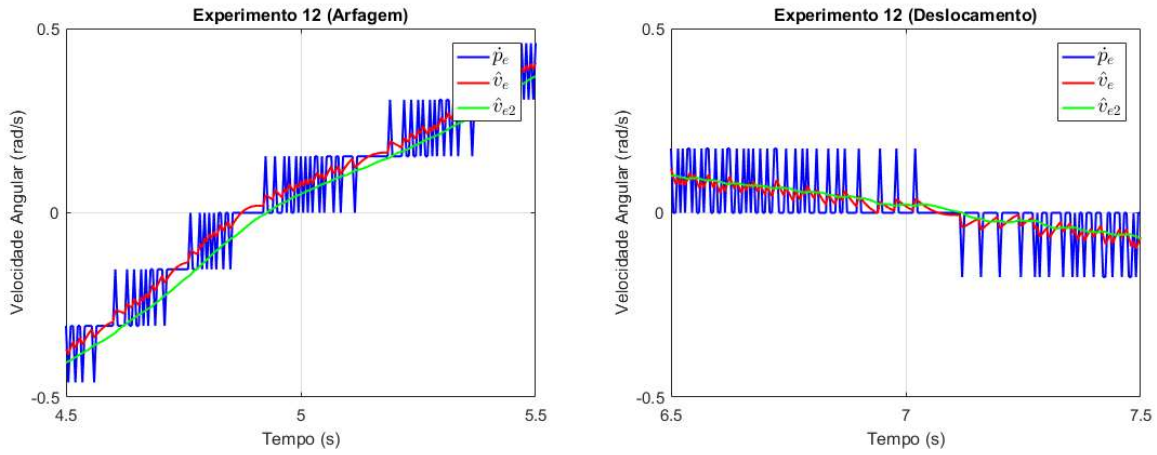
Inspeção imediata dos dados experimentais mostram benefícios da técnica Tracking Loop. Note que a derivação direta do movimento angular $\dot{p}_e$, dada a geração de pulsos pelo encoder somado ao codificador da placa de aquisição, geram picos indesejados no cálculo da velocidade angular. Tais picos podem ocasionar ruídos na ação de controle, e consequentemente nas saídas controladas.

Os dados experimentais ilustram o impacto de cada ganho do controlador PI do Tracking Loop. O ganho K_p tende a gerar resultados mais precisos em $\dot{v}_1$, quanto maior seu valor, porém com o risco de gerar sinais ruidosos em variações bruscas de movimento angular. O ganho K_i tende a gerar resultados menos ruidosos em $\dot{v}_2$, quanto maior seu valor, porém com a contramedida de maior atraso, e, portanto, deslocamento vertical do resultado. É importante citar, que os ganhos K_p e K_i

possuem influência no resultado de ambas as velocidades $\dot{v}_{\mathrm{e}}$ e $\dot{v}_{\mathrm{e2}}$, dado o fato de ser um sistema realimentado.

Este trabalho considerará em seus experimentos de controle $K_p = 40$ e $K_i = 900$ em ambos os sensores de encoder. As análises gráficas para este experimento estão apresentadas na Figura 6.8 para os encoders de arfagem (a) e deslocamento (b).

A velocidade estimada considerada para todos os experimentos reais de controle é a $\dot{v}_{\mathrm{e2}}$. Tal decisão é tomada com o objetivo de diminuir os ruídos nas ações de controle do sistema.



(a) (b)

Figura 6.8 – Experimento 12 ($K_p = 40, K_i = 900$). (a) Arfagem; (b) Deslocamento
Fonte: Autoria própria

7 RESULTADOS EXPERIMENTAIS

Tendo realizado a modelagem por prototipagem virtual do Helicóptero de 3 GDL, obtendo as equações diferenciais do sistema em espaço de estados, assim como seu modelo não linear, tem-se a tarefa de controlar as posições angulares de deslocamento y_1 e elevação y_2 através de um controle seguidor para sistemas MIMO, aplicando o rastreamento de um vetor de entrada com as posições desejadas de cada movimento controlado. Para isso adota-se o método de atribuição de autoestrutura completa para determinar os autovalores e autovetores associados e calcular os ganhos de realimentação e integral do erro, $\boldsymbol{K_1}$ e $\boldsymbol{K_2}$ respectivamente.

Realiza-se a análise dos resultados de controle via três experimentos distintos, conforme a seguir: (1) A simulação do protótipo virtual em um sistema linear, utilizando as matrizes de espaço de estado; (2) A simulação da dinâmica não linear do protótipo virtual, através de um bloco de co-simulação entre *ADAMS* e *MATLAB/Simulink;* (3) O ensaio de controle da planta real, utilizando um sistema de aquisição de dados para a interface com o controle.

A Figura 7.1 mostra o conjunto de blocos criado no *MATLAB/Simulink* para a realização do controle seguidor dos três experimentos citados (simulação linear, simulação não linear e ensaio real). O conjunto de blocos segue a topologia apresentada na Figura 2.1, porém com ajustes para um sistema discreto, como pode ser observado pelo integrador do ganho $\boldsymbol{K_2}$.

O bloco verde que recebe as ações de controle $\boldsymbol{u}$ e fornece as saídas do sistema $\boldsymbol{y}$ é responsável pela descrição do sistema do Helicóptero de 3 GDL, e é alterando o conteúdo interno deste que se seleciona qual dos 3 experimentos é executado.

O vetor de entradas é composto pelas referências das posições angulares de deslocamento r_λ e elevação r_ε em radianos. Estes dois vetores são multiplexados e aplicados na malha de controle comparando-os às saídas das posições angulares de deslocamento y_1 e elevação y_2. A matriz $\boldsymbol{E}_{2\times6}$, apresentada na Equação (7.1), é quem faz a seleção das duas varáveis de saída controláveis.

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.1}$$

Para ter acesso aos dados do vetor de estado $\boldsymbol{x}$ aplica-se a multiplicação da matriz inversa de saída $\boldsymbol{C^{-1}}$ com o vetor de saída $\boldsymbol{y}$.

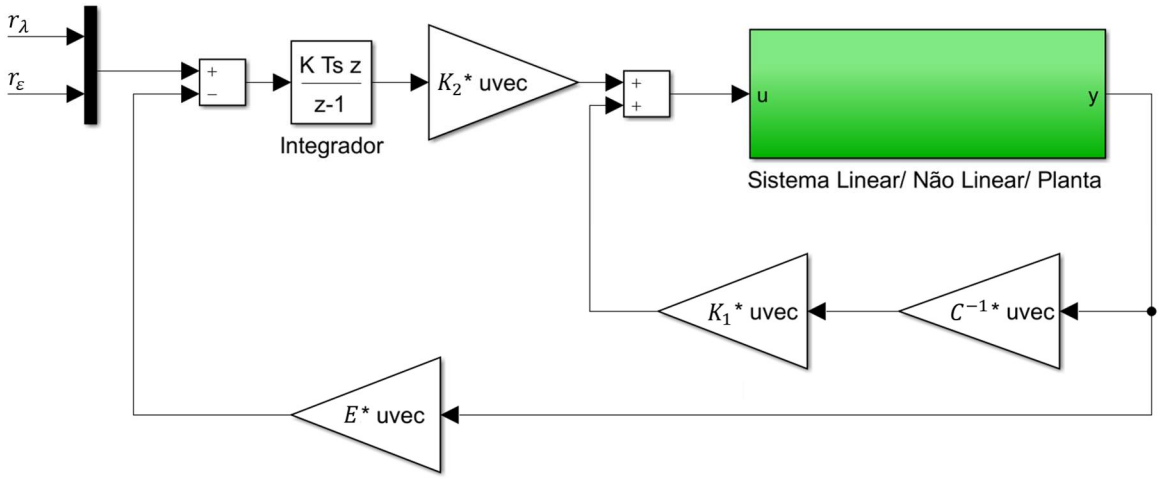


Figura 7.1 – Diagrama de Blocos do Controle Seguidor para os experimentos
Fonte: Autoria própria

7.1 SIMULAÇÃO LINEAR

Para a simulação do protótipo virtual em um sistema linear, utilizando as matrizes de espaço de estado deve-se aplicar ao bloco verde da Figura 7.1 o conteúdo do bloco da Figura 7.2.

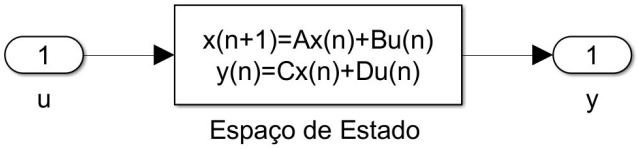


Figura 7.2 – Bloco para simulação linear
Fonte: Autoria própria

As matrizes de espaço de estado do modelo matemático final fornecidas pela análise linear do *ADAMS* no processo de prototipagem virtual são contínuas. Para simular o sistema a tempo discreto é necessário realizar a discretização das matrizes. Utilizou-se a função *c2d* (Zero-Order Hold) do *MATLAB* a uma taxa de 200Hz.

7.2 SIMULAÇÃO NÃO LINEAR

Para a simulação da dinâmica não linear do protótipo virtual, é utilizado um bloco de interação gerado pelo software *ADAMS*. Dentro deste bloco estão os elementos do sistema declarados dentro do software CAE interagindo com a planta não linear, conforme mostrado na Figura 7.3. Com isto, é possível realizar a simulação e ter uma previsão visual da planta em condição das referências de entrada e controle aplicado. A Figura 7.4 demonstra essa interação visual em 2 momentos diferentes de uma simulação onde a referência de deslocamento r_λ é um sinal com forma de onda quadrada positiva de período de 40 s amplitude de 50° e a referência de elevação r_ε é um sinal com forma de onda quadrada positiva de período de 40s amplitude de 40° com um offset de 10° positivos. Os ganhos do controle serão discutidos no próximo capítulo.

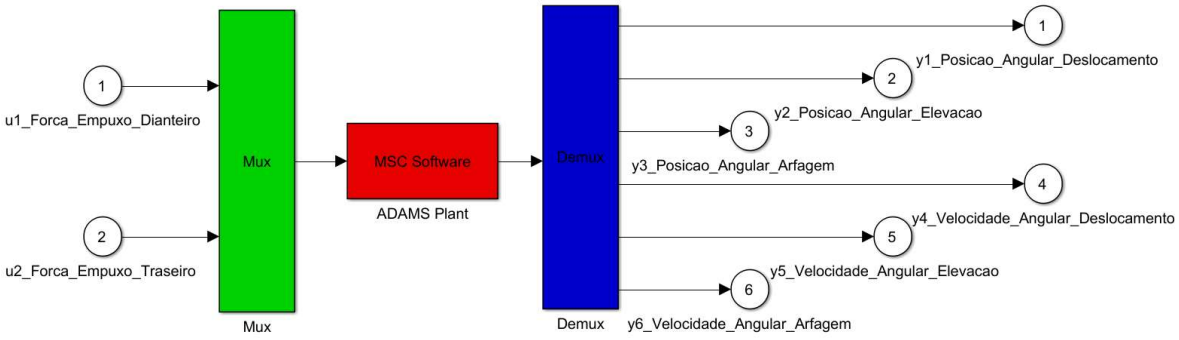


Figura 7.3 – Elementos dentro do bloco do *ADAMS*
Fonte: Autoria própria

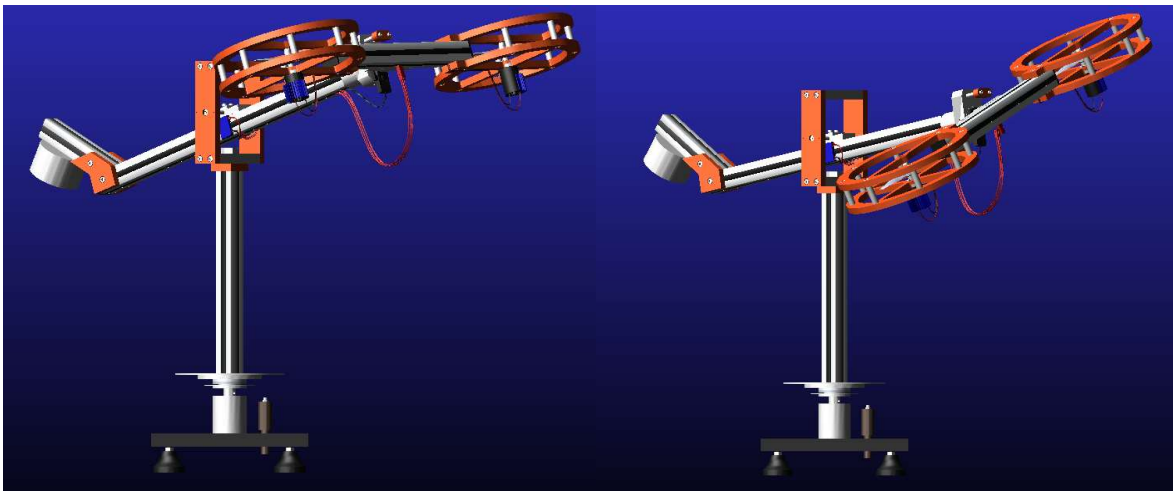


Figura 7.4 – Simulação não linear. t = 30,11 s (esquerda). t = 41,11 s (direita)
Fonte: Autoria própria

7.3 PLANTA REAL

Para o ensaio de controle da planta real, utiliza-se o protótipo modelo Helicóptero Tandem 3 GDL construído neste projeto. A conexão entre o *MATLAB/Simulink* e protótipo ocorre através da placa de aquisição PCI-6602 da *National Instruments*. A Figura 7.5 mostra as conexões necessárias para o funcionamento do ensaio. O encoder RCML15 da *RENCO* é responsável pelo envio de sinais elétricos provenientes da variação da posição angular dos movimentos de elevação e arfagem, já o encoder Q9863 da *HP* do movimento de deslocamento. A placa PCI-6602, recebe os sinais e faz o tratamento necessário para fornecer os dados na unidade em graus. A Figura 7.6 mostra os dados dos encoders sendo acessados através do *MATLAB/Simulink.* Nela também é possível verificar a aplicação da quadratura de cada encoder $(4 \times Resolução)$ e a conversão de graus para radianos com o intuito de estimar as velocidades de cada movimento com a técnica de Tracking Loop, sendo K_p e K_i os ganhos para cálculo da saída $\boldsymbol{y}$ formada pelas posições estimadas $\hat{p}_e$ e as segundas velocidades estimadas $\hat{v}_{e2}$.

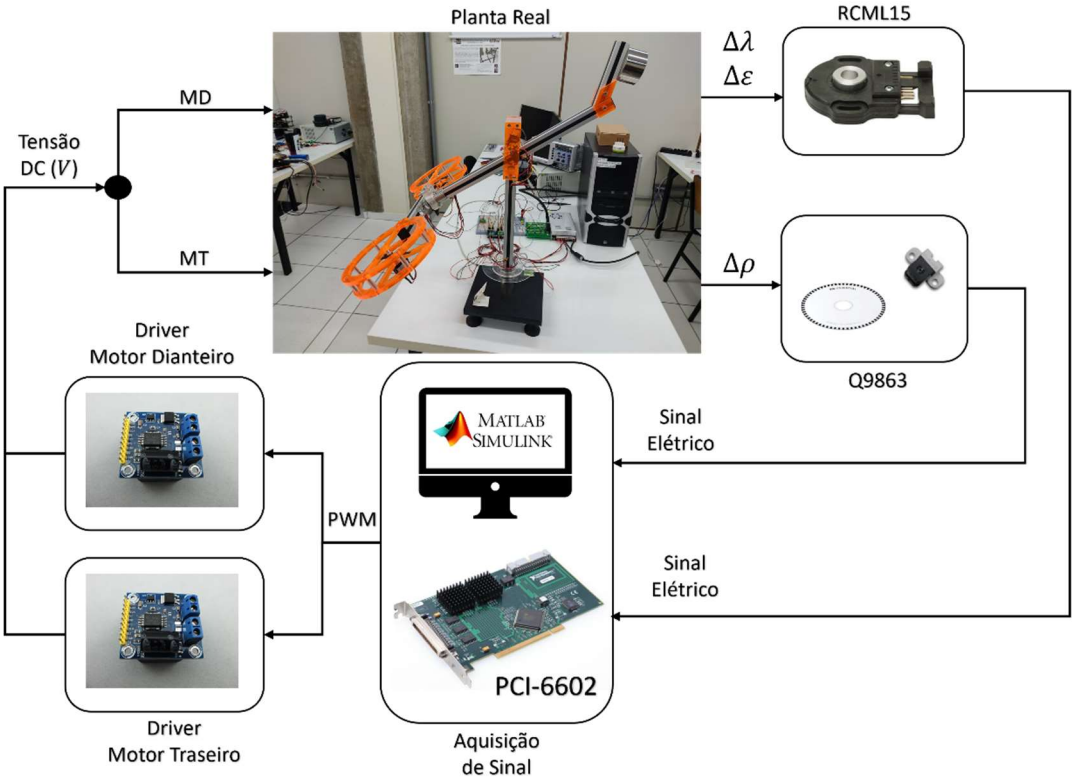


Figura 7.5 – Diagrama para experimento da planta real
Fonte: Autoria própria

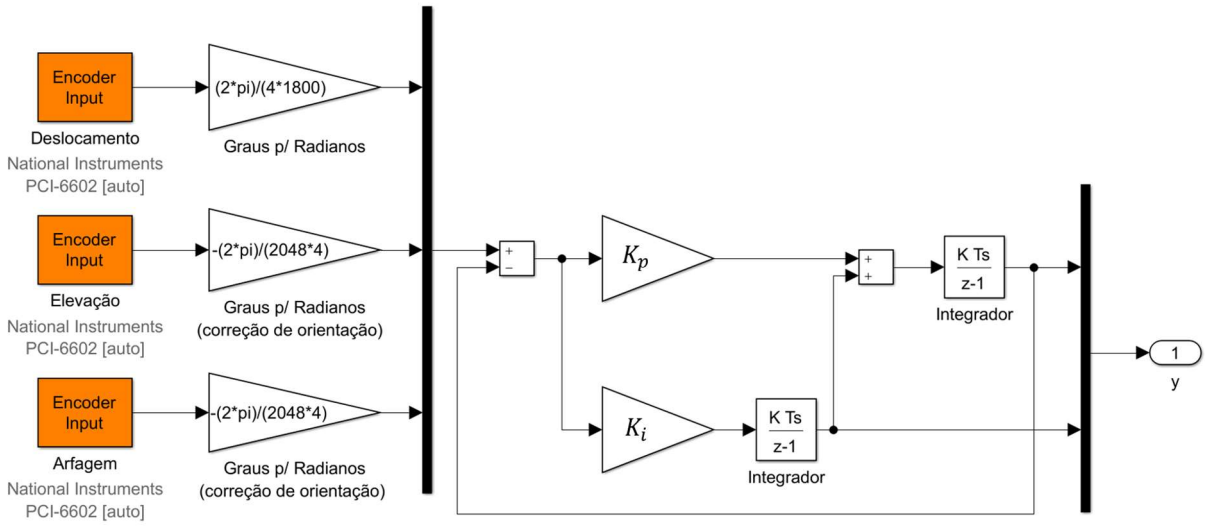


Figura 7.6 – Aquisição dos dados dos encoders e Tracking Loop
Fonte: Autoria própria

A Figura 7.7 mostra o tratamento dos sinais de controle pela plataforma *MATLAB/Simulink*. Incialmente aplica-se um offset em cada sinal, atribuindo as forças de empuxo para equilíbrio na posição inicial de elevação u_{op}. Após isso, com os polinômios de identificação do motor dianteiro y_d e do motor traseiro y_d relaciona-se as forças de empuxo do sistema em N com a porcentagem equivalente de ciclo ativo do sinal PWM. Conforme a Figura 7.5, quem recebe este sinal é o driver de potência SX8847 da *Sonxun*, enviando a tensão correspondente em corrente contínua para cada motor.

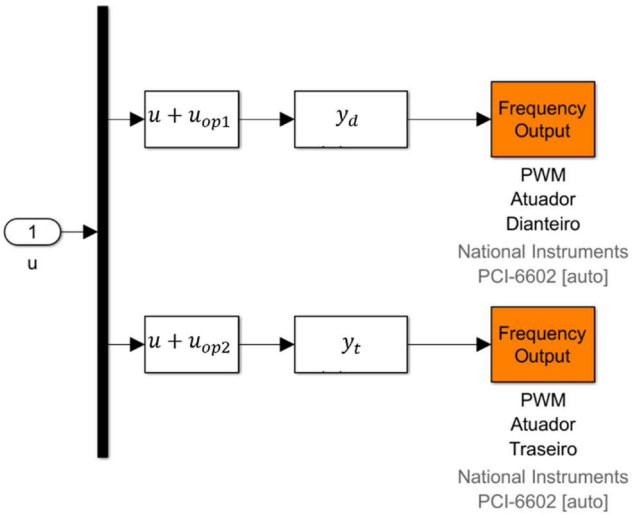


Figura 7.7 – Tratamento dos sinais de controle
Fonte: Autoria própria

A Figura 7.8 demonstra o funcionamento do controle da planta real em 2 momentos diferentes, onde as referências de deslocamento r_λ e elevação r_ε são as mesmas da Figura 7.4. Os ganhos do controle serão discutidos no próximo capítulo.

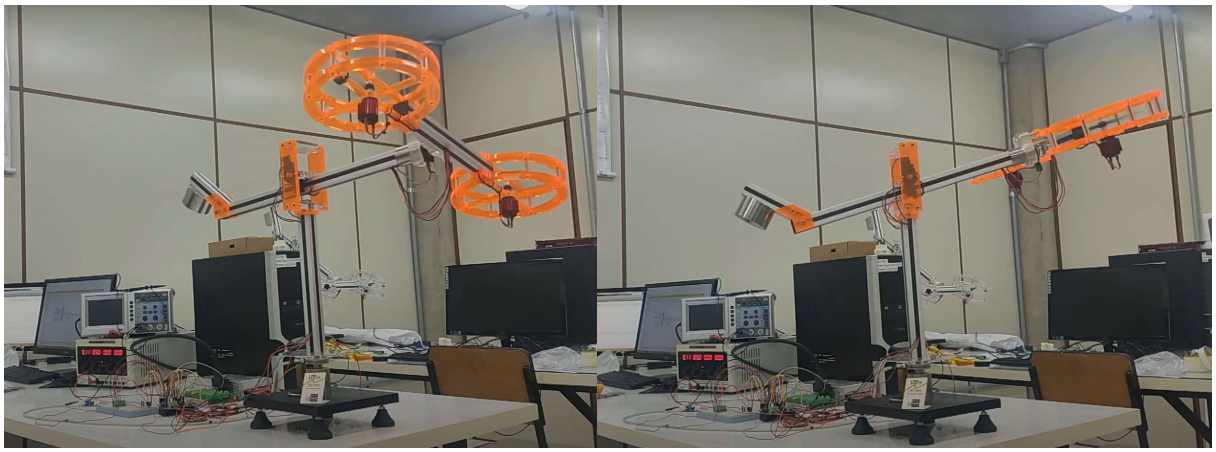


Figura 7.8 – Ensaio Real. t = 31,5 s (esquerda). t = 40,22 s (direita)
Fonte: Autoria própria

7.4 RESULTADOS EXPERIMENTAIS: CONTROLE[3]

Neste capítulo são apresentados resultados experimentais provenientes de quatro experimentos distintos. Nos experimentos, aplicou-se o controle seguidor no Helicóptero de 3 GDL com o método de atribuição de autoestrutura completa. No primeiro experimento tem-se o objetivo de analisar as ações de controle do sistema sobre o movimento de elevação, validando a relação entre modelo virtual e a identificação dos atuadores. Os demais experimentos possuem como objetivo analisar as referências de posição angular, de deslocamento e de elevação com a saída $\boldsymbol{y}$ do sistema, comparando-os entre modelo linear, não linear e planta real. As ações de controle também são comparadas com a mesma metodologia.

Os dados do controle seguidor, aplicando a atribuição por autoestrutura completa são:

$$\sigma(\boldsymbol{A}+\boldsymbol{BK}) = [-1.20+0.12\mathrm{i} \quad -1.20-0.12\mathrm{i} \quad -0.85+0.11\mathrm{i} \quad -0.85-0.11\mathrm{i} \quad \ldots \\ -0.85+0.10\mathrm{i} \quad -0.85-0.10\mathrm{i} \quad -1.50 \quad -0.95]$$

[3] Vídeo do experimento: <https://www.youtube.com/watch?v=8momKDfKi8g>. Acessado em 21 de setembro de 2021.

$$\boldsymbol{K_1} = \begin{bmatrix} -5.8594 & -3.1911 & 1.6444 & 1.2558 & 0.8597 & 1.7959 \\ 5.8594 & 3.1911 & 1.6444 & 1.2558 & -0.8597 & -1.7959 \end{bmatrix}$$

$$\boldsymbol{K_2} = \begin{bmatrix} 0.6877 & 0.4322 \\ -0.6877 & 0.4322 \end{bmatrix}$$

$$\boldsymbol{q} = \begin{bmatrix} -1 & 0 & -1 & 0 & -1 & 0 & -1 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 & -1 \end{bmatrix}$$

7.4.1 Experimento 1

O sinal de referência do experimento 1 é composto por:

- r_λ = Sinal constante em 0°.
- r_ε = Sinal constante 32,14°.

Este experimento tem como objetivo identificar a diferença dos empuxos para cada tipo de abordagem dada uma referência constante na posição angular de elevação. A posição λ = 32,14° é quando os três planos dos três graus de liberdade do sistema são ortogonais.

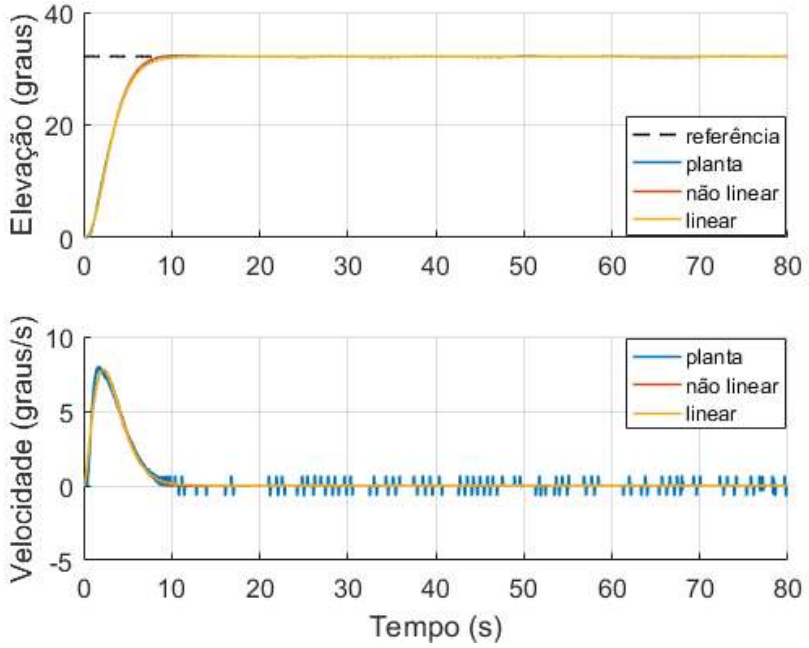


Figura 7.9 – Posição e velocidade angular de elevação do Experimento 1
Fonte: Autoria própria

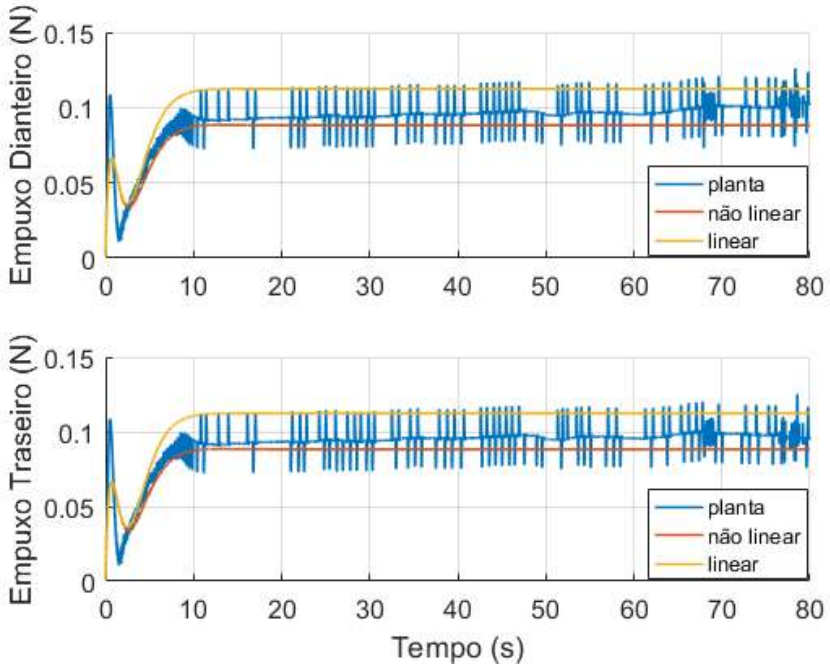


Figura 7.10 – Empuxo Dianteiro e Traseiro do Experimento 1
Fonte: Autoria própria

A Figura 7.9 mostra os resultados experimentais da posição e velocidade angular de elevação, destacando referência, dados da planta, dados não lineares e lineares. A Figura 7.10 mostra as ações de controle, tanto para o empuxo dianteiro, quanto empuxo traseiro. É possível verificar certa diferença para cada tipo de abordagem, algo esperado, já que o modelo linear possui os empuxos de linearização u_{op} para a compensação na posição inicial de deslocamento considerados em seu sistema. A ação de controle no modelo experimental apresenta perda de potência por aquecimento nos motores.

7.4.2 Experimento 2

O sinal de referência do experimento 2 é composto por:

- r_λ = Sinal com forma de onda quadrada, amplitude de 25°, offset de +25° e período de 40 s.
- r_ε = Sinal com forma de onda quadrada, amplitude de 20°, offset de +30° e período de 40 s.

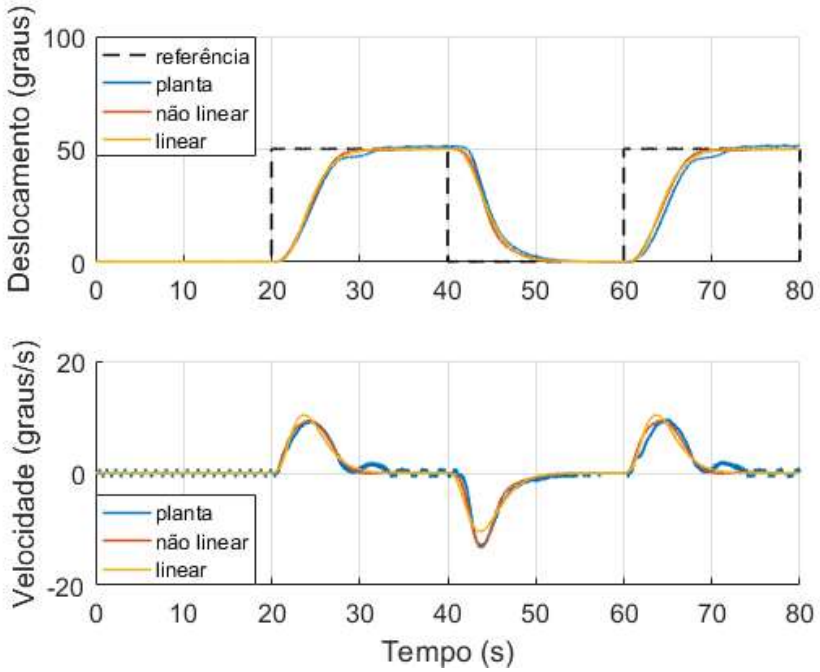


Figura 7.11 – Posição e velocidade angular de deslocamento do Experimento 2
Fonte: Autoria própria

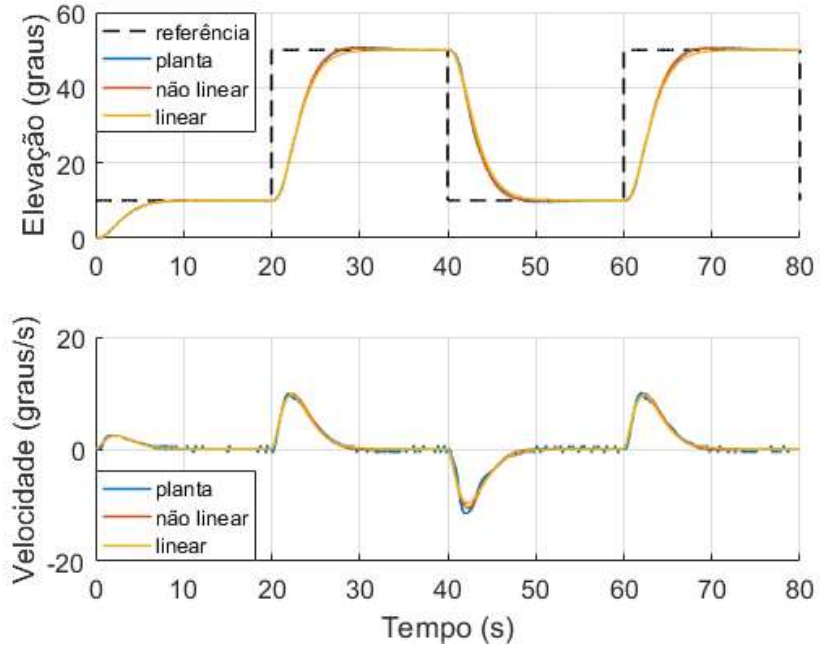


Figura 7.12 – Posição e velocidade angular de elevação do Experimento 2
Fonte: Autoria própria

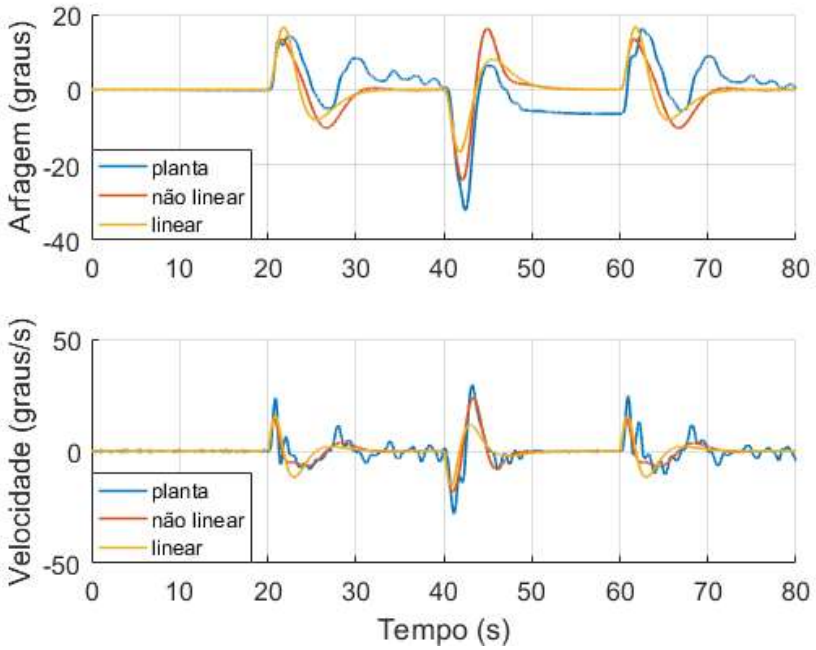


Figura 7.13 – Posição e velocidade angular de arfagem do Experimento 2
Fonte: Autoria própria

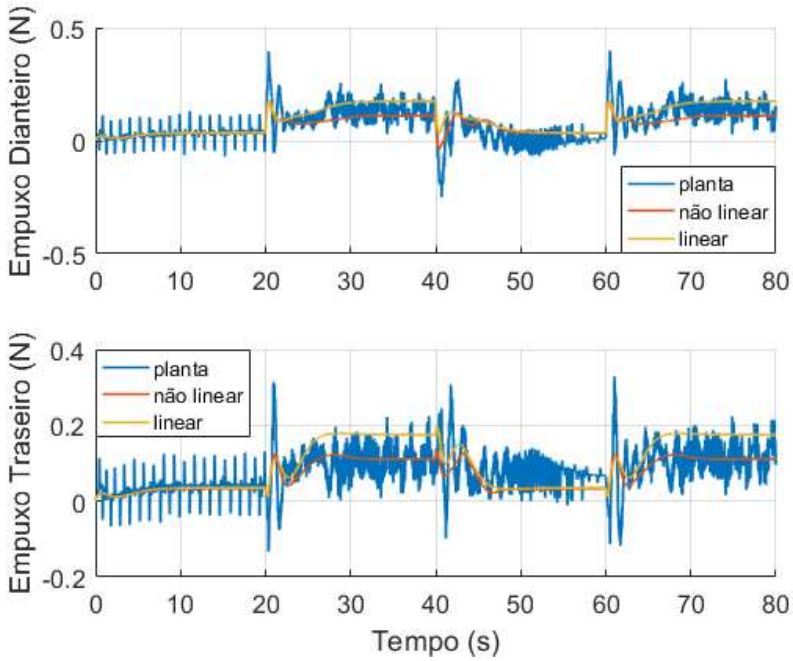


Figura 7.14 – Empuxo Dianteiro e Traseiro do Experimento 2
Fonte: Autoria própria

As Figuras 7.11, 7.12 e 7.13 mostram os resultados experimentais da posição e velocidade angular de deslocamento, elevação e arfagem, respectivamente, destacando as referências quadradas, dados da planta, dados não lineares e lineares.

A Figura 7.14 mostra as ações de controle, tanto para o empuxo dianteiro, quanto empuxo traseiro para este experimento.

Os dois movimentos controlados obtiveram reações parecidas para os 3 tipos de abordagens. Próximo dos 30 s um erro no movimento de deslocamento no ensaio real provocou uma correção no movimento de arfagem, ocasionando um erro de offset posteriormente.

7.4.3 Experimento 3

O sinal de referência do experimento 3 é composto por:

- r_λ = Sinal com forma de onda senoidal, amplitude de 25°, offset de +25° e período de 40 s.
- r_ε = Sinal com forma de onda senoidal, amplitude de 20°, offset de +30° e período de 40 s.

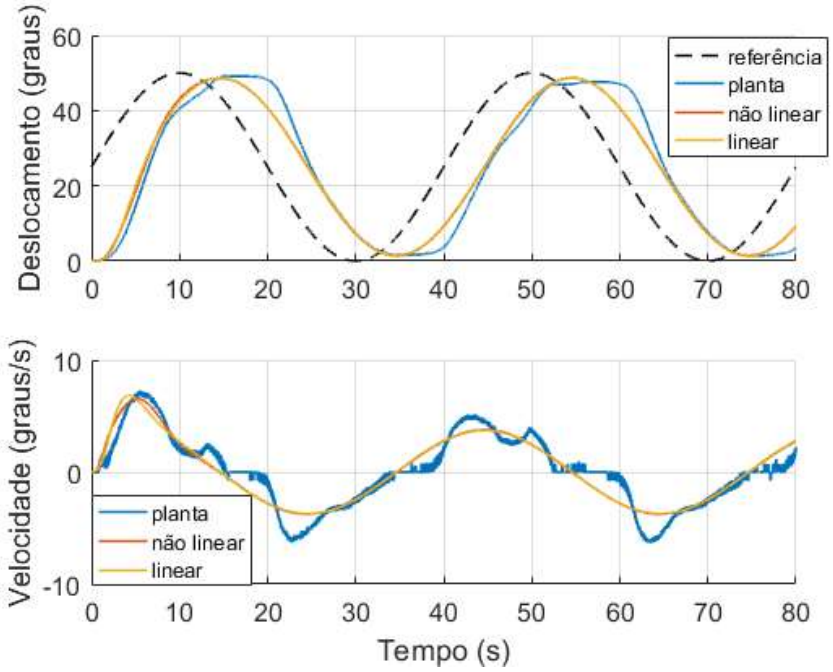


Figura 7.15 – Posição e velocidade angular de deslocamento do Experimento 3
Fonte: Autoria própria

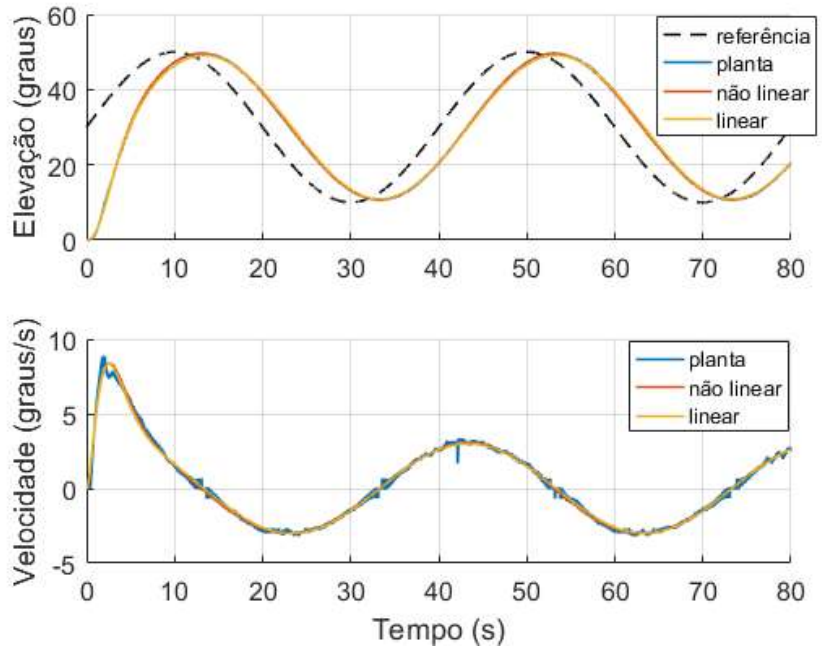


Figura 7.16 – Posição e velocidade angular de elevação do Experimento 3
Fonte: Autoria própria

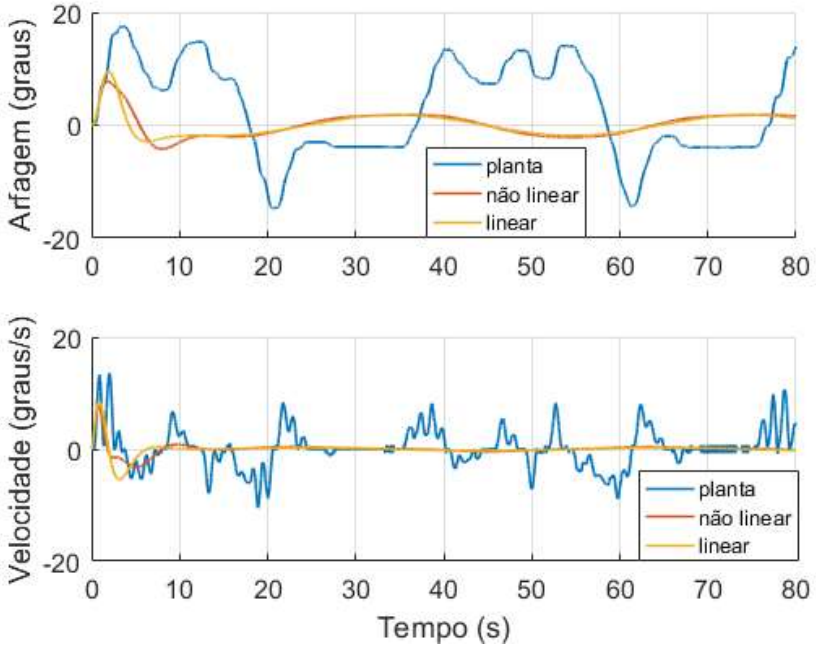


Figura 7.17 – Posição e velocidade angular de arfagem do Experimento 3
Fonte: Autoria própria

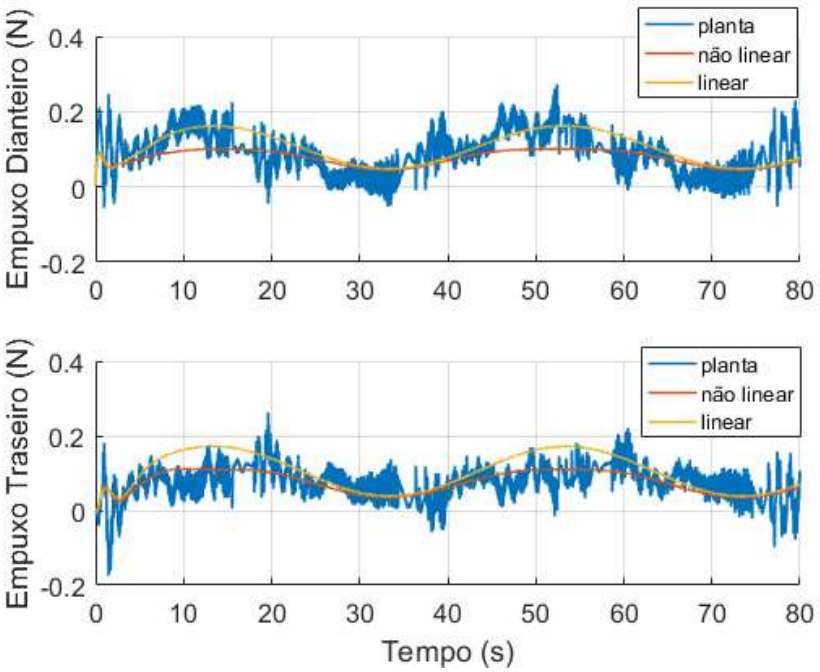


Figura 7.18 – Empuxo Dianteiro e Traseiro do Experimento 3
Fonte: Autoria própria

As Figuras 7.15, 7.16 e 7.17 mostram os resultados experimentais da posição e velocidade angular de deslocamento, elevação e arfagem, respectivamente, destacando as referências senoidais, dados da planta, dados não lineares e lineares. A Figura 7.18 mostra as ações de controle, tanto para o empuxo dianteiro, quanto empuxo traseiro para este experimento.

Os dois movimentos controlados obtiveram reações atrasadas no ensaio real em comparação com as modelagens. A todo momento há erros no movimento de deslocamento no ensaio real, provocando correções na posição de arfagem.

7.4.4 Experimento 4

O sinal de referência do experimento 4 é composto por:

- r_λ = Sinal com forma de serra, amplitude de 25°, offset de +25° e período de 40 s.
- r_ε = Sinal com forma de serra, amplitude de 20°, offset de +30° e período de 40 s.

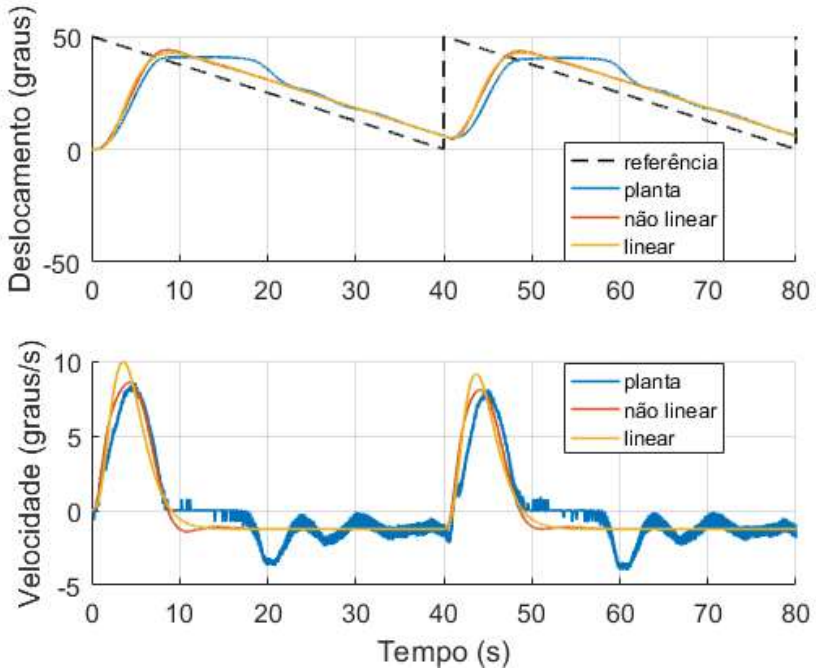


Figura 7.19 – Posição e velocidade angular de deslocamento do Experimento 4
Fonte: Autoria própria

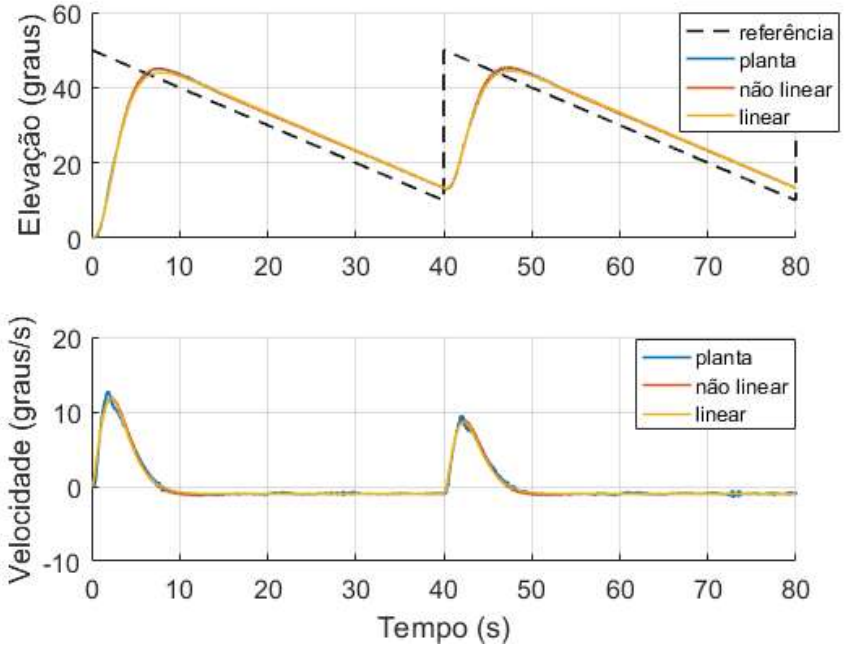


Figura 7.20 – Posição e velocidade angular de elevação do Experimento 4
Fonte: Autoria própria

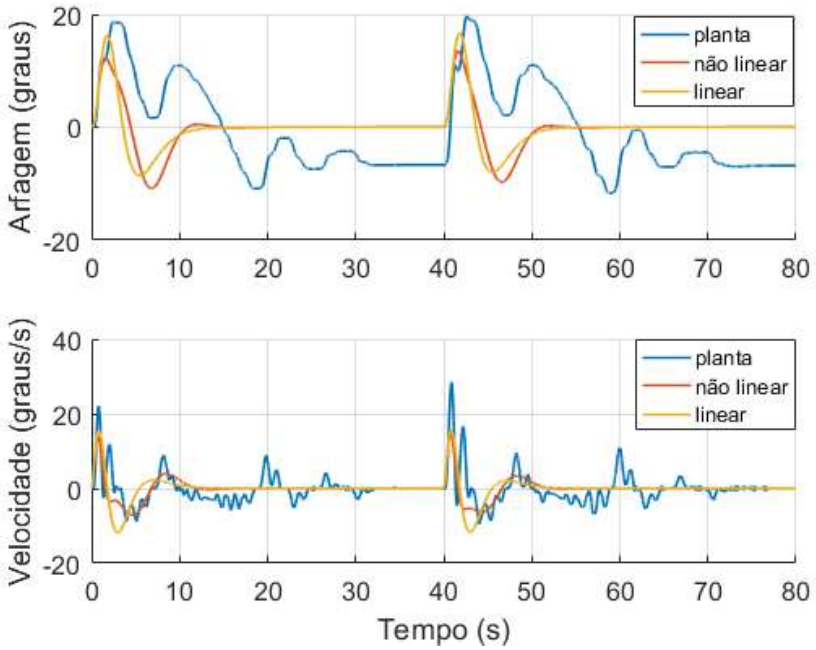


Figura 7.21 – Posição e velocidade angular de arfagem do Experimento 4
Fonte: Autoria própria

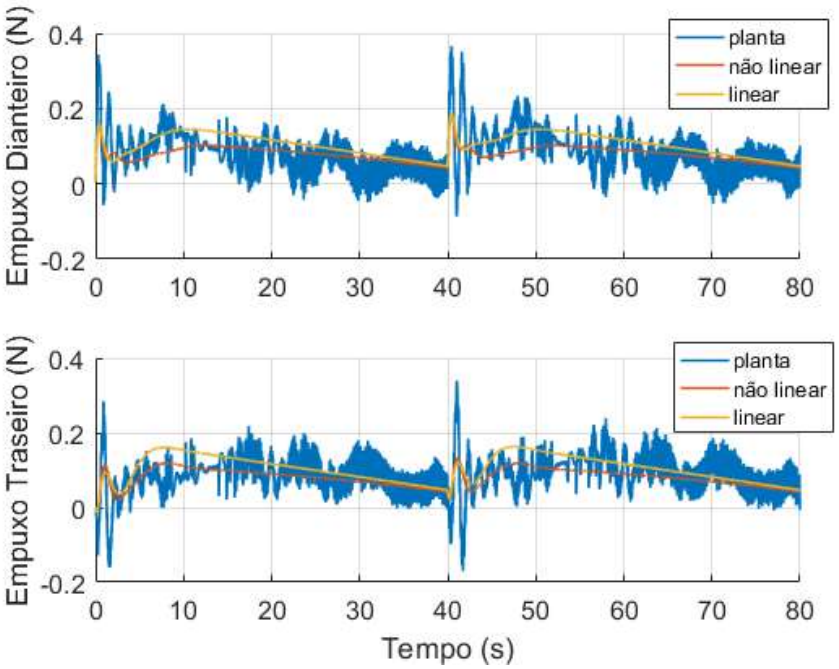


Figura 7.22 – Empuxo Dianteiro e Traseiro do Experimento 4
Fonte: Autoria própria

As Figuras 7.19, 7.20 e 7.21 mostram os resultados experimentais da posição e velocidade angular de deslocamento, elevação e arfagem, respectivamente, destacando as referências em forma de serra, dados da planta, dados não lineares e lineares. A Figura 7.22 mostra as ações de controle, tanto para o empuxo dianteiro,

quanto empuxo traseiro para este experimento. Para este caso, já não há atrasos, em comparação com o experimento com ondas senoidais, porém ainda persiste o erro do deslocamento no ensaio real, concatenando seu erro no movimento de arfagem (parte de atuação do sistema).

A diferença entre o ensaio real e as simulações referente ao movimento de arfagem parece ter dois motivos relacionados. Os valores de empuxo para o movimento de deslocamento são pequenos, precisando de um maior ângulo de arfagem para compensar a força de atrito dos cabos da haste. Durante o ensaio é possível ver correções na arfagem para alcançar o valor de deslocamento referenciado.

Nos resultados de controle as ações de controle dos atuadores na planta real se mostraram mais ruidosas em comparação com as simuladas, tanto linear, quanto não linear. Este ruído é característico em motores DC com imã permanente, mas pode ser diminuído quando sobrescrito por outros motivos. A aplicação do Tracking Loop nos encoders para a estimativa de velocidade tem como objetivo atenuar os pulsos digitais para um sinal filtrado, e como consequência diminuir o ruído na ação de controle, por exemplo. Uma alternativa para atenuação está sobre o processo de identificação dos atuadores, pois os polinômios de identificação são obtidos com uma abrangência de intervalo de força maior do que a necessária para os ensaios de controle da planta, podendo descaracterizar o intervalo de força mais utilizado.

7.5 DISCUSSÃO: CONSIDERAÇÕES DA CONSTRUÇÃO

O desenvolvimento de uma planta real apresenta certas variáveis que devem ser consideradas, mas comumente não são. Por exemplo, na prototipagem virtual do Helicóptero de 3 GDL mantém-se o centro de massa de cada corpo rígido em simetria com o plano de movimento deste, assim não havendo tendências de movimento dada um momento de inércia fora das restrições. Porém uma planta real possui componentes, como o motor, em que a densidades dos materiais que o compõem são diferentes, criando assim certa divergência entre o centro de massa do protótipo virtual com o da planta real. Além de que erros de usinagens e montagem são comuns na criação de um protótipo real, salientando estes problemas.

Essa dissimetria pôde ser verificada no laboratório. No corpo da aeronave, ocorreu um deslocamento do centro de massa em direção ao motor traseiro, criando uma posição angular de arfagem positiva em repouso. Para compensar essa posição, colocou-se uma massa de 10 g próxima ao motor dianteiro, conforme a Figura 7.23. A massa acrescentada foi adicionada a massa total do corpo da aeronave no protótipo virtual.

Tal análise é importante para levantar o questionamento sobre a diferença entre os centros de massa dos outros corpos entre o protótipo virtual e planta real, fato que pode acarretar em discrepâncias no resultado de controle durante os experimentos.

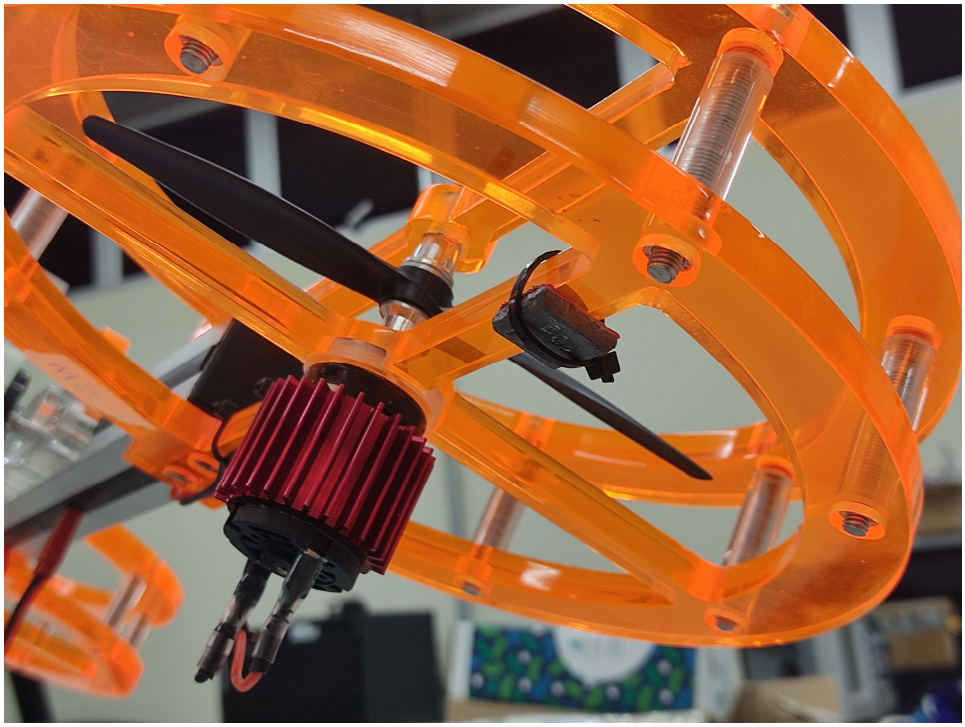


Figura 7.23 – Peso para compensar posição angular de arfagem.
Fonte: Autoria própria

Outro problema, observado durante a construção do protótipo real é o erro propagado nas medições. As dimensões das peças, assim como o peso destas concatenam um erro para o cálculo da densidade, entrada necessária para a prototipagem virtual. A configuração dos corpos rígidos também sofre com a resolução e aferição dos equipamentos de medição. O posicionamento do contrapeso, por exemplo, tem interferência na dinâmica do sistema, porém dada a sua geometria, encontra-se dificuldade em encontrar a tangente para seu posicionamento, devendo

assim utilizar a porca fixadora como referência, que por sua vez possui uma certa folga com o perfil de alumínio, podendo assim, criar um erro de posicionamento. A Figura 7.24 mostra a geometria do contrapeso, dificultando seu posicionamento.

Figura 7.24 – Geometria do contrapeso complicando posicionamento
Fonte: Autoria própria

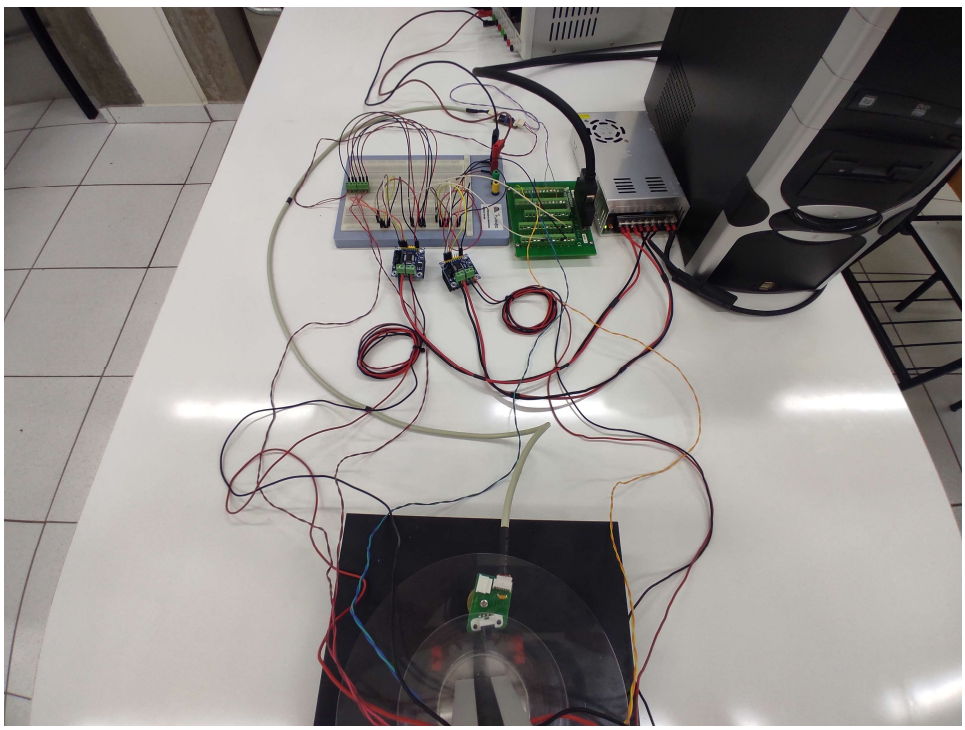


Figura 7.25 – Cabos da Haste.
Fonte: Autoria própria

Forças de atrito são difíceis de identificar em um sistema real, por tal motivo foram desconsideradas sobre os eixos de movimento do protótipo virtual. Além disso, a presença de cabos na planta real cria uma oposição aos movimentos do sistema. A Figura 7.25 mostra uma grande quantidade de cabos que influenciam o movimento de deslocamento da planta.

8 CONCLUSÕES

Este trabalho descreve a construção de um protótipo experimental baseado em helicóptero Tandem com 3 graus de liberdade. Além da construção detalhada, o texto apresenta modelos que visam representar a dinâmica do helicóptero. Uma ferramenta de apoio adotada foi a modelagem de sistemas multicorpos, através da prototipagem virtual.

As decisões tomadas no software CAE, ADAMS, foram para diminuir a complexidade do modelo, evitando restrições, tais como aquelas relacionadas a forças de contato ou de atrito. Por um lado, essa decisão facilita a reprodução do trabalho, pois o software facilita a portabilidade. Porém, não são geradas equações de um sistema dinâmico mais complexo, como aquelas onde há assimetria nos momentos de inércia de um corpo em movimento.

A atribuição por autoestrutura completa determina a resposta temporal do sistema em malha fechada. A dificuldade encontrada nessa estratégica é saber o intervalo de autovalores e autovetores associados que geram uma resposta em malha fechada satisfatória, visto que o processo do helicóptero Tandem é não linear.

Uma dificuldade observada durante os experimentos foi as divergências entre a planta real e o protótipo virtual, problemas de identificação, talvez resultante de forças não consideradas ou limitações inerentes dos modelos.

O trabalho apresentado detalhou todos os procedimentos em nível de reprodução, gerando assim uma contribuição para o desenvolvimento de plantas de controle de interesse educacional, possibilitando a estudantes e pesquisadores o contato com a prototipagem virtual, identificação de sistemas, aquisição de dados, instrumentação, mecatrônica e sistemas de controle.

8.1 TRABALHOS FUTUROS

Recomenda-se como trabalhos futuros:

- Aplicar forças de contato e atrito no protótipo virtual e comparar os resultados;

- Desenvolver um protótipo de um Helicóptero de 3 graus de liberdade com dissimetria vertical entre os dois atuadores, aproximando mais de um helicóptero Tandem real;
- Desenvolver um sistema embarcado livre de fios para a comunicação dos sensores de posição, retirando a interferência dos cabos nos movimentos do helicóptero;
- Realizar a identificação do modelo a partir de equações diferenciais não lineares;
- Aplicar novas técnicas de controle linear e não linear.

REFERÊNCIAS

ALFA INSTRUMENTS. **Célula de carga - S**. 2019. Disponível em: <http://www.alfainstrumentos.com.br/produto/s/>. Acesso em: 16 de jun. de 2021.

BOUABDALLAH, S. **Design and control of quadrotors with application to autonomous flying**. PhD Thesis. École Polytechnique Fédérale de Lausanne. Lausanne 115p. 2007.

BRAGA, M. A; SANTANA, P. H. de R. Q. e A. **Concepção de um veículo aéreo não-tripulado do tipo quadrirrotor.** Trabalho de Graduação (Engenharia Mecatrônica) Universidade de Brasília. 164p – DF 2008.

CHEN, C. **Linear System Theory and Design**. 3 Ed. New York; Oxford University Press, 1999.

COSTA, E. B. **Algoritmos de controle aplicados a estabilização de voo de um quadrotor**. Dissertação (Engenharia Elétrica – Sistemas de Energia). Universidade Federal de Juiz de Fora. 133p. Juiz de Fora – MG 2012.

DASSAULT SYSTÉMES. **SOLIDWORKS Simulation**. 2019. Disponível em: <https://www.solidworks.com/product/solidworks-simulation>. Acesso em: 16 de jun. de 2021.

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.

DZUL, A; HAMEL, T; LOZANO, R. **Nonlinear Control for a Tandem Rotor Helicopter**. 15th Triennial World Congress, Barcelona, Spain, p. 229-234, 2002.

DZUL, A; HAMEL, T; LOZANO, R. **Nonlinear Control for a Tandem Rotor Helicopter**. Systems, Man and Cybernetics. IEEE International Conference, v.6, p.6-9 , 2002.

HIBBELER, R. **Estática: mecânica para engenharia**. 10. ed. São Paulo: Pearson Prentice Hall, 2005.

HOMA, J. **Aerodinâmica e teoria de voo**. 28. Ed. São Paulo: ASA, 2003.

HOUPIS, C. H.; SHELDON. **Linear Control System Analysis and Design with MATLAB**. 6. Ed. Taylor and Francis Group, 2014.

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.

KARAMAN, K; BEKAROGLU, Y.T; SOYLEMEZ, M.T; UÇAK, K; OKE GUNEL, G**. Controlling 3-DOF Helicopter via Fuzzy PID Controller.** 9th International Conference on Electrical and Eletronics Engineering (ELECO), p. 869-873, 2015. DOI 10.1109/ELECO.2015.7394494.

KOGAGIL, B.M; OZCAN, S; ARICAN, A.C; GUZEY, U.M; COPUR, E.H; SALAMEI, M.U. **Adaptive Control of a 3 DoF Helicopter with Linear and Nonlinear Reference Models**. 6th International Conference on Control Engineering & Information Technology, 2018. DOI 10.1109/CEIT.2018.8751799.

LEE, S.-H.; SONG, J.-B. Ac**celeration 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>. Acesso em: 16 de jun. de 2021.

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.

MALAQUIAS, W. **Prototipagem virtual, simulação e controle de um quadrirrotor utilizando controle seguidor com atribuição de autoestrutura completa**. 93 f. Dissertação (Mestrado em Engenharia Mecânica) — Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2017.

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

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

MATHWORKS. **Object-Oriented Programming in MATLAB**. 2019. Disponível em: <https://www.mathworks.com/discovery/object-oriented-programming.html>. Acesso em: 16 de jun. de 2021.

MATOS, N. M. R. de. **Análise do funcionamento de um servomotor de corrente alternada com ímanes permanentes**. 86 f. Dissertação (Mestrado em Eletrotécnica e Computadores) — Universidade do Porto e Universidade Regional de Blumenau, Blumenau, 2012.

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>. Acesso em: 16 de jun. de 2021.

MOLLON, Matheus Fernando. **Projeto, Identificação e Controle de Uma Plataforma de Quadrirotor**. 2020. 200 f. Dissertação de mestrado – Programa de Pós-graduação em Engenharia Mecânica, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2020.

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.

MONTEZUMA, M. A. F. **Metodologia para identificação e controle de um protótipo de uma plataforma de movimento com 2 G.D.L**. 167 f. Tese (Doutorado em Engenharia Mecânica) — Universidade de São Paulo, São Carlos, 2010.

MSC SOFTWARE. **ADAMS: the multibody dynamics simulation solution**. 2019. Disponível em: <https://www.mscsoftware.com/product/adams>. Acesso em: 13 de jun. de 2021.

MSC SOFTWARE. **Multibody dynamics**. 2019. Disponível em: <https://www.mscsoftware.com/application/multibody-dynamics>. Acesso em: 13 de jun. de 2021.

NATIONAL INSTRUMENTS. **NI 660X specifications**. 2009. Disponível em: <http://www.ni.com/pdf/manuals/372141b.pdf>. Acesso em: 16 de jun. de 2021.

NATIONAL INSTRUMENTS. **NI 6251**. 2016. Disponível em: <http://www.ni.com/pdf/manuals/375213c.pdf>. Acesso em: 16 de jun. de 2021.

NATIONAL INSTRUMENTS. **O que é aquisição de dados?** 2019. Disponível em: <https://www.ni.com/data-acquisition/what-is/pt/>. Acesso em: 16 de jun. de 2021.

NATIONAL INSTRUMENTS. **PCI-6602**. 2019. Disponível em: < https://www.ni.com/pt-br/support/model.pci-6602.html>. Acesso em: 16 de jun. de 2021.

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

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

PASQUINI J., J. (1998). **Modelagem e análise da dinâmica lateral de veículo automotivos**. 75p. Dissertação (Mestrado em Engenharia Mecânica) – Escola de Engenharia de São Carlos, Universidade de São Carlos, Universidade de São Paulo, São Carlos.

Paula, J. C. de. **Desenvolvimento de um VANT do tipo quadrirrotor para obtenção de imagens aéreas em alta definição**. Dissertation. Universidade Federal do Paraná – UFPR. 110p Curitiba – PR 2012.

POSTLETHWAITE, I; PREMPAIN, E; TURKOGLU, E. TURNER, M.C. **Design and flight testing of various H-infinite controllers for the Bell 205 helicopter**. Control Engineering Practive, 2005. DOI 10.1016/J.conengprac.2003.12.008.

RENCO. **RCML 15 selection guide**. 2019. Disponível em: <https://www.renco.com/en/products/rcml15.html>. 16 de jun. de 2021.

ROSARIO, J. M. **Princípios de mecatrônica**. São Paulo: Pearson Prentice Hall, 200

QUANSER. **3 DOF Helicopter: simulink courseware**. 2017. Disponível em: Acesso em: 13 de mai. de 2021. Disponível em: <https://www.quanser.com/products/3-dof-helicopter/>. Acesso em: 01 de jun. de 2021.

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 Paraná (UTFPR), Cornélio Procópio - PR, 2015.

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

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

SUNADA, M. M. **Acionamento de um posicionador linear de ultraprecisão empregando uma redução harmonic drive com controle de velocidade**. Dissertação de mestrado (Engenharia Mecânica) Universidade Federal de Santa Catarita – UFSC. 162p Florianópolis – SC 2007.

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.

WATKINSON, J. **The Art of the Helicopter**. 1. Ed. Elsevier Butterworth-Heinemann, 2004.

YU, YAO; LU, GENG; SUN, CHANGYIN; LIU, HAO. **Robust backstepping decentralized tracking control for a 3-DOF helicopter**. Nonlinear Dynamics, p.947-960, 2015. DOI 10.1007/s11071-015-2209-8.

APÊNDICE A – CAPÍTULO 2

```
function varargout = eeaa(varargin)
%EEAA    Entire eigenstructure assignment algorithm
%    [K1,K2] = EEAA(SYSC,E,SIGMA,Qf) computes the gain matrices K1 and K2,
%    respectively for the state feedback and integral of the error of the
%    tracking system (control law u = K1*x + K2*z)
%        SYSC is the continuous state-space representation of the system
%        E is the controlled outputs matrix
%        SIGMA is the closed-loop eigenvalue spectrum for the state
%        feedback
%        Qf is the desired Q matrix obtained through the linear combination
%        of the vectors that span the null space
%
%    [K1,K2,EV,V,Q,EEA] = EEAA(SYSC,E,SIGMA,Qf) also returns the closed
%    loop eigenvalues, the matrices V and Q, and the structure EEA with the
%    calculations of every step in the entire eigenstructure assignment
%    algorithm
%
%    K = EEAA(SYSC,SIGMA,Qf) computes the gain matrix K for the state
%    feedback (control law u = K*x)
%
%    [K,EV,V,Q,EEA] = EEAA(SYSC,SIGMA,Qf) also returns the closed loop
%    eigenvalues, the matrices V and Q, and the structure EEA with the
%    calculations of every step in the entire eigenstructure assignment
%    algorithm

    %% Verify the function arguments
    if nargin ~= 3 && nargin ~=4
        error('Invalid number of arguments.');
    else
        % Get argument SYSC
        SYSC = varargin{1,1};
        % Verify if SYSC is a continuous state-space
        if ~isa(SYSC,'ss') || SYSC.Ts ~= 0
            error(strcat('Enter the continuous state-space represen', ...
                'tation of the system for SYSC.'));
        end

        % Get argument E
        if nargin == 4
            E = varargin{1,2};
        end

        % Get argument SIGMA
        SIGMA = varargin{1,end-1};

        % Get argument Qf
        Qf = varargin{1,end};
    end

    %% Extract info from the inputs to run the algorithm
    n = length(SYSC.A); % Order of the state matrix A
    m = size(SYSC.B,2); % Number of control inputs
    if nargin == 4 % Tracking system
        p = size(E,1); % Number of controlled states
    else % State feedback only
        p = 0; % Number of controlled states
    end
    S_order = (n+p)+m; % Order of the matrix S

    %% Determine the matrices A and B used for the entire eigenstructure
    % assignment and verify the controllability condition
    if nargin == 3 % State feedback only
        A = SYSC.A;
        B = SYSC.B;

        % Verify if the matrix pair (A,B) is controllable, the number of
        % linearly independent rows or columns of the controllability
        % matrix must be equal to the nth state order
        if rank(ctrb(A,B)) ~= n
            error('The matrix pair (A,B) is not controllable!');
        end
    else % Tracking system
```

```
    A = [SYSC.A zeros(n,p);
         -E     zeros(p)];
    B = [SYSC.B;
         zeros(p,m)];

    % Verify if the matrix pair (A,B) is controllable, the number of
    % linearly independent rows or columns of the matrix must be equal
    % to the nth state order plus the p number of controlled states
    if rank([SYSC.B SYSC.A;
             zeros(p,m) -E]) ~= n+p
        error('The matrix pair (Abar,Bbar) is not controllable!');
    end
end

%% Verify if SIGMA and A sizes are matching
if n+p ~= length(SIGMA) % remember p = 0 for state feedback only
    error(strcat('The length of the closed-loop eigenvalue spectr', ...
        'um does not match the n-th order of the system.'));
end

%% Verify if SIGMA is in the left-half of the complex plane
if any(real(SIGMA) > 0)
    error(strcat('The closed loop eigenvalue spectrum is not in t', ...
        'he left-half of the complex plane.'));
end

%% Verify if the eigenvalues have their complex conjugate pairs (needs to be tested)
SIGMA_map = zeros(1,n+p);
for counter = 1:n+p
    if isreal(SIGMA(counter))
        SIGMA_map(counter) = counter;
    else
        if counter > n
            candidates = n+1:n+p;
        else
            candidates = 1:n;
        end
        candidates(counter) = [];
        for search = candidates
            if conj(SIGMA(counter)) == SIGMA(search)
                SIGMA_map(counter) = search;
                break;
            end
            if search == candidates(end)
                error(strcat('There are eigenvalues without a', ...
                    ' complex conjugate pair.'));
            end
        end
    end
end

%% Verify if the size of the matrix Qf is correct (m x n+p)
if any(size(Qf) ~= [m n+p])
    error('Matrix Qf size does not match (m x n+p)');
end

%% Preallocate the structure for the entire eigenstructure assignment
EEA = struct('eigenvalue', num2cell(SIGMA));

%% Build the matrix S for each of the eigenvalues of SIGMA and
% determine the eigenvectors that span the null space

% Variable that indicates if there was a row interchange
interchange = false;

for counter = 1:n+p
    % Form the matrix S = [A-lambda*I B]
    EEA(counter).So = [A-EEA(counter).eigenvalue*eye(n+p) B];

    % Put the matrix into hermite normal form
    % Augment with sufficient rows of zeros to form a square matrix
    EEA(counter).S = [rref(EEA(counter).So);
                      zeros(m,S_order)];

    % Interchange rows, so that the leading ones in each row are on the
    % principal diagonal
    position = find(real(diag(EEA(counter).S(1:n+p,:))) ~= 1);
```

```
        if ~isempty(position)
            interchange = true;
            for diagonal = position
                column = find(real(EEA(counter).S(diagonal,:)) == 1,1, ...
                    'first');
                if isempty(column)
                    error('Error finding the leading 1 on HNF!');
                end
                EEA(counter).S(diagonal:S_order,:) = ...
                    circshift(EEA(counter).S(diagonal:S_order,:), ...
                    [column-diagonal 0]);
            end
        end

        % Replace all 0's on the principal diagonal with -1's
        positions = find(real(diag(EEA(counter).S)) == 0);
        positions = sub2ind(size(EEA(counter).S), positions, positions);
        EEA(counter).S(positions) = -1;

        % The columns with -1 are the vectors that span the null space
        EEA(counter).span_kerS = ...
            EEA(counter).S(:,real(diag(EEA(counter).S)) == -1);

        % Operation to verify that the correct ker S has been achieved
        operation = [A-EEA(counter).eigenvalue*eye(n+p) B]* ...
            EEA(counter).span_kerS;
        % The operation must result in a zero matrix
        if any(any(abs(real(operation)) > 1e-10))
            error('The wrong ker S was achieved!');
        end
    end

    %% Select the eigenvectors and form the matrix K
    % Initialize matrices V and Q
    V = zeros(n+p,n+p);
    Q = zeros(m,n+p);

    % If there was a row interchange the eigenvector selection might not be
    % correct
    if interchange
        warning(strcat('There was a row interchange in the algorith', ...
            'm execution, the resultant Q matrix may differ.'));
    end

    % Select the eigenvectors
    Qf = -1*Qf;
    SIGMA_pos = [];
    for counter = 1:n+p
        if counter <= SIGMA_map(counter)
            SIGMA_pos = [SIGMA_pos counter];
        end
    end
    for counter = SIGMA_pos
        if isreal(SIGMA(counter)) % Real eigenvalues
            aux = 0;
            for count_m = 1:m
                aux = aux + EEA(counter).span_kerS(:,count_m)* ...
                    Qf(count_m,counter);
            end
            V(:,counter) = real(aux(1:n+p));
            Q(:,counter) = real(aux(n+p+1:end));
        else % Complex eigenvalues
            aux = 0;
            for count_m = 1:m
                aux = aux + EEA(counter).span_kerS(:,count_m)* ...
                    (complex(Qf(count_m,counter),Qf(count_m,SIGMA_map(counter))));
            end
            V(:,counter) = real(aux(1:n+p));
            Q(:,counter) = real(aux(n+p+1:end));
            V(:,SIGMA_map(counter)) = imag(aux(1:n+p));
            Q(:,SIGMA_map(counter)) = imag(aux(n+p+1:end));
        end
    end

    % Determine the matrix K
    K = real(Q*inv(V));
```

```
    % Operation to verify that the eigenvalue spectrum has been correctly
    % assigned
    Acl = A+B*K;
    sigma_Acl = eig(Acl);
    operation = sort(SIGMA,'ascend')-sort(sigma_Acl,'ascend')';
    % The operation must result in a zero matrix
    if any(abs(real(operation)) > 1e-5)
        error('The eigenvalue spectrum was incorrectly assigned!');
    end

    %% Form the function output
    if nargin == 4 % Tracking system
        % Separate matrix K
        K1 = K(:,1:n);
        K2 = K(:,n+1:end);

        varargout = {K1,K2,sigma_Acl,V,Q,EEA};
    else % State feedback only
        varargout = {K,sigma_Acl,V,Q,EEA};
    end
end
```

```
function varargout = deeaa(varargin)
%DEEAA    Entire eigenstructure assignment discrete algorithm
%    [K1,K2] = DEEAA(SYSC,E,SIGMA,Qf,st) computes discrete the gain
%    matrices K1 and K2, respectively for the state feedback and integral
%    of the error of the tracking system (control law u = K1*x + K2*z)
%        SYSC is the continuous state-space representation of the system
%        E is the controlled outputs matrix
%        SIGMA is the closed-loop eigenvalue spectrum for the state
%        feedback
%        Qf is the desired Q matrix obtained through the linear combination
%        of the vectors that span the null space
%        st is the discretization sample time
%
%    [K1,K2,EV,V,Q,EEA] = DEEAA(SYSC,E,SIGMA,Qf,st) also returns the closed
%    loop eigenvalues, the matrices V and Q, and the structure EEA with the
%    calculations of every step in the entire eigenstructure assignment
%    algorithm
%
%    K = DEEAA(SYSC,SIGMA,Qf,st) computes the gain matrix K for the state
%    feedback (control law u = K*x)
%
%    [K,EV,V,Q,EEA] = DEEAA(SYSC,SIGMA,Qf,st) also returns the closed loop
%    eigenvalues, the matrices V and Q, and the structure EEA with the
%    calculations of every step in the entire eigenstructure assignment
%    algorithm

    %% Verify the function arguments
    if nargin ~= 4 && nargin ~=5
        error('Invalid number of arguments.');
    else
        % Get argument SYSC
        SYSC = varargin{1,1};
        % Verify if SYSC is a continuous state-space
        if ~isa(SYSC,'ss') || SYSC.Ts ~= 0
            error(strcat('Enter the continuous state-space represen', ...
                'tation of the system for SYSC.'));
        end

        % Get argument E
        if nargin == 5
            E = varargin{1,2};
        end

        % Get argument SIGMA
        SIGMA = varargin{1,end-2};

        % Get argument Qf
        Qf = varargin{1,end-1};

        % Get argument st
        st = varargin{1,end};
    end

    %% Extract info from the inputs to run the algorithm
```

```
n = length(SYSC.A); % Order of the state matrix A
m = size(SYSC.B,2); % Number of control inputs
if nargin == 5 % Tracking system
    p = size(E,1); % Number of controlled states
else % State feedback only
    p = 0; % Number of controlled states
end
S_order = (n+p)+m; % Order of the matrix S

%% Determine the matrices A and B used for the entire eigenstructure
% assignment and verify the controllability condition
if nargin == 4 % State feedback only
    A = SYSC.A;
    B = SYSC.B;

    % Verify if the matrix pair (A,B) is controllable, the number of
    % linearly independent rows or columns of the controllability
    % matrix must be equal to the nth state order
    if rank(ctrb(A,B)) ~= n
        error('The matrix pair (A,B) is not controllable!');
    end
else % Tracking system
    A = [SYSC.A zeros(n,p);
         -E     zeros(p)];
    B = [SYSC.B;
         zeros(p,m)];

    % Verify if the matrix pair (A,B) is controllable, the number of
    % linearly independent rows or columns of the matrix must be equal
    % to the nth state order plus the p number of controlled states
    if rank([SYSC.B SYSC.A;
             zeros(p,m) -E]) ~= n+p
        error('The matrix pair (Abar,Bbar) is not controllable!');
    end
end
% Discretization of the matrices A and B
[A,B] = c2d(A,B,st);

%% Verify if SIGMA and A sizes are matching
if n+p ~= length(SIGMA) % remember p = 0 for state feedback only
    error(strcat('The length of the closed-loop eigenvalue spectr', ...
        'um does not match the n-th order of the system.'));
end

%% Verify if SIGMA is in the left-half of the complex plane
if any(real(SIGMA) > 0)
    error(strcat('The closed loop eigenvalue spectrum is not in t', ...
        'he left-half of the complex plane.'));
end

%% Verify if the eigenvalues have their complex conjugate pairs
% (needs to be tested)
SIGMA_map = zeros(1,n+p);
for counter = 1:n+p
    if isreal(SIGMA(counter))
        SIGMA_map(counter) = counter;
    else
        if counter > n
            candidates = n+1:n+p;
        else
            candidates = 1:n;
        end
        candidates(counter) = [];
        for search = candidates
            if conj(SIGMA(counter)) == SIGMA(search)
                SIGMA_map(counter) = search;
                break;
            end
            if search == candidates(end)
                error(strcat('There are eigenvalues without a', ...
                    ' complex conjugate pair.'));
            end
        end
    end
end

%% Verify if the size of the matrix Qf is correct (m x n+p)
```

```
if any(size(Qf) ~= [m n+p])
    error('Matrix Qf size does not match (m x n+p)');
end

%% Representation of the eigenvalue spectrum on the z-plane
SIGMA = exp(SIGMA*st);

%% Preallocate the structure for the entire eigenstructure assignment
EEA = struct('eigenvalue', num2cell(SIGMA));

%% Build the matrix S for each of the eigenvalues of SIGMA and
% determine the eigenvectors that span the null space

% Variable that indicates if there was a row interchange
interchange = false;

for counter = 1:n+p
    % Form the matrix S = [A-lambda*I B]
    EEA(counter).So = [A-EEA(counter).eigenvalue*eye(n+p) B];

    % Put the matrix into hermite normal form
    % Augment with sufficient rows of zeros to form a square matrix
    EEA(counter).S = [rref(EEA(counter).So);
                      zeros(m,S_order)];

    % Interchange rows, so that the leading ones in each row are on the
    % principal diagonal
    position = find(real(diag(EEA(counter).S(1:n+p,:))) ~= 1);
    if ~isempty(position)
        interchange = true;
        for diagonal = position
            column = find(real(EEA(counter).S(diagonal,:)) == 1,1, ...
                'first');
            if isempty(column)
                error('Error finding the leading 1 on HNF!');
            end
            EEA(counter).S(diagonal:S_order,:) = ...
                circshift(EEA(counter).S(diagonal:S_order,:), ...
                [column-diagonal 0]);
        end
    end

    % Replace all 0's on the principal diagonal with -1's
    positions = find(real(diag(EEA(counter).S)) == 0);
    positions = sub2ind(size(EEA(counter).S), positions, positions);
    EEA(counter).S(positions) = -1;

    % The columns with -1 are the vectors that span the null space
    EEA(counter).span_kerS = ...
        EEA(counter).S(:,real(diag(EEA(counter).S)) == -1);

    % Operation to verify that the correct ker S has been achieved
    operation = [A-EEA(counter).eigenvalue*eye(n+p) B]* ...
        EEA(counter).span_kerS;
    % The operation must result in a zero matrix
    if any(any(abs(real(operation)) > 1e-10))
        error('The wrong ker S was achieved!');
    end
end

%% Select the eigenvectors and form the matrix K
% Initialize matrices V and Q
V = zeros(n+p,n+p);
Q = zeros(m,n+p);

% If there was a row interchange the eigenvector selection might not be
% correct
if interchange
    warning(strcat('There was a row interchange in the algorith', ...
        'm execution, the resultant Q matrix may differ.'));
end

% Select the eigenvectors
Qf = -1*Qf;
SIGMA_pos = [];
for counter = 1:n+p
    if counter <= SIGMA_map(counter)
```

```
            SIGMA_pos = [SIGMA_pos counter];
        end
    end
    for counter = SIGMA_pos
        if isreal(SIGMA(counter)) % Real eigenvalues
            aux = 0;
            for count_m = 1:m
                aux = aux + EEA(counter).span_kerS(:,count_m)* ...
                    Qf(count_m,counter);
            end
            V(:,counter) = real(aux(1:n+p));
            Q(:,counter) = real(aux(n+p+1:end));
        else % Complex eigenvalues
            aux = 0;
            for count_m = 1:m
                aux = aux + EEA(counter).span_kerS(:,count_m)* ...
                    (complex(Qf(count_m,counter),Qf(count_m, ...
                    SIGMA_map(counter))));
            end
            V(:,counter) = real(aux(1:n+p));
            Q(:,counter) = real(aux(n+p+1:end));
            V(:,SIGMA_map(counter)) = imag(aux(1:n+p));
            Q(:,SIGMA_map(counter)) = imag(aux(n+p+1:end));
        end
    end

    % Determine the matrix K
    K = real(Q*inv(V));

    % Operation to verify that the eigenvalue spectrum has been correctly
    % assigned
    Acl = A+B*K;
    sigma_Acl = eig(Acl);
    operation = sort(SIGMA,'ascend')-sort(sigma_Acl,'ascend')';
    % The operation must result in a zero matrix
    if any(abs(real(operation)) > 1e-5)
        error('The eigenvalue spectrum was incorrectly assigned!');
    end

    %% Form the function output
    if nargin == 5 % Tracking system
        % Separate matrix K
        K1 = K(:,1:n);
        K2 = K(:,n+1:end);

        varargout = {K1,K2,sigma_Acl,V,Q,EEA};
    else % State feedback only
        varargout = {K,sigma_Acl,V,Q,EEA};
    end
end
```

APÊNDICE B – CAPÍTULO 4

```
function [alfa_graus,beta_graus,gama_graus] = inversoZXZEuler(R)
% Solução para o cosseno de beta como valor positivo

    % Determina seno de beta positivo
    sen_beta = sqrt(R(1,3)^2 + R(2,3)^2);
    % Determina o cosseno de beta
    cos_beta = R(3,3);
    % Determina o ângulo beta em graus
    beta_graus = atan2d(sen_beta,cos_beta);

    % Verificando os possíveis casos do ângulo beta
    switch beta_graus
        % Caso seja igual a 0 graus
        case 0
            % Adota o ângulo alfa como 0 graus
            alfa_graus = 0;
            % Determina seno de gama
            sen_gama = R(2,1);
            % Determina cosseno de gama
            cos_gama = R(2,2);
            % Determina o ângulo gama em graus
            gama_graus = atan2d(sen_gama,cos_gama);

        % Caso seja igual a 180 graus
        case {180, -180}
            % Adota o ângulo alfa como 0 graus
            alfa_graus = 0;
            % Determina seno de gama
            sen_gama = -R(2,1);
            % Determina cosseno de gama
            cos_gama = -R(2,2);
            % Determina o ângulo gama em graus
            gama_graus = atan2d(sen_gama,cos_gama);

        % Para todos os casos em que cosseno de beta é diferente de zero
        otherwise
            % Determina o seno de alfa
            sen_alfa = R(1,3)/sen_beta;
            % Determina o cosseno de alfa
            cos_alfa = -R(2,3)/sen_beta;
            % Determina o ângulo alfa em graus
            alfa_graus = atan2d(sen_alfa,cos_alfa);
            % Determina o seno de gama
            sen_gama = R(3,1)/sen_beta;
            % Determina o cosseno de gama
            cos_gama = R(3,2)/sen_beta;
            % Determina o ângulo gama em graus
            gama_graus = atan2d(sen_gama,cos_gama);
    end
end
```

```
%% Cálculo dos Ângulos Z-X-Z de Euler
clear all; close all; clc;

%% Ângulos Z-X-Z de Euler
syms ca sa cb sb cy sy;
Ra = [ca -sa 0;
      sa  ca 0;
      0   0  1];
Rb = [1  0   0;
      0  cb -sb;
      0  sb  cb];
Ry = [cy -sy 0;
      sy  cy 0;
      0   0  1];
RE = Ra*Rb*Ry;

%% Orientação da Base
R1 = [ 1.00,  0.01,  0.00;
       0.00,  0.00, -1.00;
      -0.01,  1.00,  0.00]'
```

```
[alfa,beta,gama] = inversoZXZEuler(R1);
O1 = [alfa beta gama]

%% Orientação da Haste Vertical
R2 = [ 0.00,  1.00,  0.00;
      -1.00,  0.00,  0.00;
       0.00,  0.00,  1.00]'
[alfa,beta,gama] = inversoZXZEuler(R2);
O2 = [alfa beta gama]

%% Orientação do Braço de Sustentação
R3 = [ 0.00, -0.58,  0.82;
       0.00, -0.82, -0.58;
       1.00,  0.00,  0.00]'
[alfa,beta,gama] = inversoZXZEuler(R3);
O3 = [alfa beta gama]

%% Orientação do Helicóptero
R4 = [ 1.00,  0.00,  0.00;
       0.00,  0.54, -0.84;
       0.00,  0.84,  0.54]'
[alfa,beta,gama] = inversoZXZEuler(R4);
O4 = [alfa beta gama]

%% Orientação do Contrapeso
R5 = [ 0.00,  0.97, -0.22;
      -1.00,  0.00,  0.00;
       0.00,  0.22,  0.97]'
[alfa,beta,gama] = inversoZXZEuler(R5);
O5 = [alfa beta gama]
```

```
%% Modelo matemático no equilíbrio estático
clear all; close all; clc; format short;

%% Espaço de estados linear contínuo com análise de equilíbrio estático

%Este código carrega as matrizes lineares geradas pelo ADAMS, e as
%configura no formato de espaço de estados. O modelo do sistema é salvo em
%"SYSC_ee.mat"

SYSC = ss(load('linear_ee_a'), ...
          load('linear_ee_b'), ...
          load('linear_ee_c'), ...
          load('linear_ee_d'))
save('SYSC_ee.mat','SYSC');

% sys = ss(A,B,C,D) Cria um modelo de espaços de estaodos contínuo da
% forma:
%                   dx/dt = Ax + Bu
%                       y = Cx + Du
```

```
%% Script do controle do HELICÓPTERO DE 3GDL
clear all; close all; clc; format short;

%% Configuração geral
% Parâmetros da simulação

st = 1/200; % Tempo de amostragem (s)
%É o mesmo tempo utilizado no PWM para a identificação dos motores

t = 40; % Tempo total da simulação (s)

% Condição inicial da elevação
y2_0 = -30.9067;
% Condição dada pela análise estática do ADAMS

%% Modelo Adams não linear

%Importação do modelo não linear:
%B_nao_linear_;
```

```
%"B_nao_linear_" comentado    = Simulação com o modelo linear
%"B_nao_linear_" descomentado = Simulação com o modelo Não Linear

%% Projeto do controlade seguidor pelo método de atribuição de autoestrutura completa:

% Carrega o modelo linear contínuo
load('_modelos/SYSC_ee.mat');

% Converte para o modelo discreto
SYSD = c2d(SYSC,st);
invC = inv(SYSC.C); %%Y=CX => X=inv(C)*Y

% Seleção das variáveis de saída controladas
S = [1 0 0 0 0 0;
     0 1 0 0 0 0];



% Atribuição de autoestruta completa
E = SYSC.C([1 2],:); %seleciona as variáveis controláveis (elevacao/deslocamento)

SIGMA = [-0.90+0.12i -0.90-0.12i -0.85+0.11i -0.85-0.11i -0.80+0.10i -0.80-0.10i -1.00 -0.95];
%espectro de auto-valores de malha fechada
%SIGMA = [desloc_v desloc_p elev_v elev_p arfg_v arfg_a integ_desloc integ_elev]

Qf = [-1  0 -1  0 -1  0 -1 -1;
       1  0 -1  0  1  0  1 -1];
[K1,K2,EV,V,Q] = eeaa(SYSC,E,SIGMA,Qf)
% Salvar dados nos arquivos
dados_linear = 'linear';
dados_naolinear = 'naolinear';
```

```
%% Script do plot dos dados simout
close all;

%% Configuração geral
linewidth = 2.5;
fs_axis_label = 20;
fs_legend = 14;
fs_axis_tick = 14;

%% Variáveis de controle
figure;
% Força de empuxo dianteiro
subplot(2,1,1);
plot(simout.time,simout.signals.values(:,3),'LineWidth',linewidth);
grid on;
ylabel({'Empuxo';'Dianteiro (N)'},'FontSize',fs_axis_label);
ax = gca;
ax.FontSize = fs_axis_tick;
% Força de empuxo traseiro
subplot(2,1,2);
plot(simout.time,simout.signals.values(:,4),'LineWidth',linewidth);
grid on;
ylabel({'Empuxo';'Traseiro (N)'},'FontSize',fs_axis_label);
xlabel('Tempo (s)','FontSize',fs_axis_label);
ax = gca;
ax.FontSize = fs_axis_tick;

%% Variáveis de saída controladas e sinal de referência
figure;
% Deslocamento
subplot(2,1,1);
plot(simout.time,rad2deg(simout.signals.values(:,1)),'--k','LineWidth',linewidth);
hold on;
plot(simout.time,rad2deg(simout.signals.values(:,5)),'LineWidth',linewidth);
grid on;
ylabel({'Posição Angular de';'Deslocamento (graus)'},'FontSize',fs_axis_label);
l = legend('referência','saída');
set(l,'FontSize',fs_legend);
ax = gca;
ax.FontSize = fs_axis_tick;
% Elevação
subplot(2,1,2);
```

```
plot(simout.time,rad2deg(simout.signals.values(:,2)),'--k','LineWidth',linewidth);
hold on;
plot(simout.time,rad2deg(simout.signals.values(:,6)),'LineWidth',linewidth);
grid on;
ylabel({'Posição Angular';'de Elevação (graus)'},'FontSize',fs_axis_label);
xlabel('Tempo (s)','FontSize',fs_axis_label);
l = legend('referência','saída');
set(l,'FontSize',fs_legend);
ax = gca;
ax.FontSize = fs_axis_tick;

%% Variáveis de saída
figure;
% Posição angular de deslocamento
subplot(3,2,1);
plot(simout.time,rad2deg(simout.signals.values(:,1)),'--k','LineWidth',linewidth);
hold on;
plot(simout.time,rad2deg(simout.signals.values(:,5)),'LineWidth',linewidth);
grid on;
ylabel({'Posição Angular de';'Deslocamento (graus)'},'FontSize',fs_axis_label);
l = legend('referência','saída');
set(l,'FontSize',fs_legend);
ax = gca;
ax.FontSize = fs_axis_tick;
% Velocidade angular de deslocamento
subplot(3,2,2);
plot(simout.time,rad2deg(simout.signals.values(:,8)),'r','LineWidth',linewidth);
grid on;
ylabel({'Velocidade Angular de';'Deslocamento (graus/s)'},'FontSize',fs_axis_label);
set(l,'FontSize',fs_legend);
ax = gca;
ax.FontSize = fs_axis_tick;

% Posição angular de elevação
subplot(3,2,3);
plot(simout.time,rad2deg(simout.signals.values(:,2)),'--k','LineWidth',linewidth);
hold on;
plot(simout.time,rad2deg(simout.signals.values(:,6)),'LineWidth',linewidth);
grid on;
ylabel({'Posição Angular';'de Elevação (graus)'},'FontSize',fs_axis_label);
l = legend('referência','saída');
set(l,'FontSize',fs_legend);
ax = gca;
ax.FontSize = fs_axis_tick;
% Velocidade angular de elevação
subplot(3,2,4);
plot(simout.time,rad2deg(simout.signals.values(:,9)),'r','LineWidth',linewidth);
grid on;
ylabel({'Velocidade Angular';'de Elevação (graus/s)'},'FontSize',fs_axis_label);
set(l,'FontSize',fs_legend);
ax = gca;
ax.FontSize = fs_axis_tick;

% Posição angular de arfagem
subplot(3,2,5);
plot(simout.time,rad2deg(simout.signals.values(:,7)),'LineWidth',linewidth);
grid on;
ylabel({'Posição Angular';'de Arfagem (graus)'},'FontSize',fs_axis_label);
xlabel('Tempo (s)','FontSize',fs_axis_label);
set(l,'FontSize',fs_legend);
ax = gca;
ax.FontSize = fs_axis_tick;
% Velocidade angular de arfagem
subplot(3,2,6);
plot(simout.time,rad2deg(simout.signals.values(:,10)),'r','LineWidth',linewidth);
grid on;
ylabel({'Velocidade Angular';'de Arfagem (graus/s)'},'FontSize',fs_axis_label);
xlabel('Tempo (s)','FontSize',fs_axis_label);
set(l,'FontSize',fs_legend);
ax = gca;
ax.FontSize = fs_axis_tick;
```

ROTEIRO ADAMS

Etapa 1 - Configurar o protótipo virtual do Adams - View:

Criar um novo modelo:

- Nomear como: _Helicoptero_3_GDL
- Gravity em: Earth Normal (-Global Y)
- Units como: MMKS - mm,kg,N,s,deg
- Escolher o diretório
- Em Settings > Units, alterar Mass para Gram

Etapa 2 - Configurar o corpo da Base:

Importar 1_Base.x_t:

- Nomear como: Base
- (Part Name)
- Adicionar na Base o Marker do centro de massa (Add to Part) e nomear como: CM
- Posicao do CM (mm): 0.89, 49.63, 0.00
- Orientacao do CM (Angulos Z-X-Z de Euler): -179.4271, 90.0000, 180.0000

- Massa (g): 9126.68
- Px = 39001054.84
- Py = 39420624.80
- Pz = 75153755.80
- Indicar o Center of Mass Marker como: Base.CM
- Indicar o Inertia Reference Marker como: Base.CM

Restringir os GDL da Base com uma Fixed Joint:

- Nomear como: JF_Ground
- Nomear o Marker fixo como: JF_Ground
- Nomear o Marker movel como: JF_Ground
- Posicao dos Markers: 0.0, 0.0, 0.0
- Orientacao dos Markers: 180.0, 90.0, 180.0

O Tools > Model Verify deve indicar 0 GD**L**

Etapa 3 - Configurar o corpo da Haste Vertical

Importar 2_Haste_Vertical.x_t:

- Nomear como: Haste_Vertical
- Adicionar na Haste_Vertical o Marker do centro de massa e nomear como: CM
- Posicao do CM (mm): 0.00, 308.06, -0.53
- Orientacao do CM (Angulos Z-X-Z de Euler): 0.00, 0.00, 90.00

- Massa (g): 989.52
- Px = 826242.16
- Py = 30378637.05
- Pz = 30810039.63
- Indicar o Center of Mass Marker como: Haste_Vertical.CM
- Indicar o Inertia Reference Marker como: Haste_Vertical.CM

Restringir a Haste_Vertical com uma Revolute Joint (Movimento de deslocamento):

- Nomear como: JR_Deslocamento
- Nomear o Marker fixo como: JR_Deslocamento_Fixo
- Nomear o Marker movel como: JR_Deslocamento_Movel
- Posicao dos Markers: 0.00, 524.7, 0.00
- Orientacao dos Markers: 180.0, 90.0, 90.0

O Tools > Model Verify deve indicar 1 GDL

Etapa 4 - Configurar o corpo do Braço de Sustentação

Importar 3_Braco_Sustentacao.x_t:

- Nomear como: Braco_Sustentacao
- Adicionar no Braco_Sustentacao o Marker do centro de massa e nomear como: CM
- Posicao do CM (mm): 0.00, 480.52, 43.64
- Orientacao do CM (Angulos Z-X-Z de Euler): 90.0000, 90.0000, 125.2724

- Massa (g): 1282.51
- Px = 812353.63
- Py = 59004364.90
- Pz = 59477260.07
- Indicar o Center of Mass Marker como: Braco_Sustentacao.CM
- Indicar o Inertia Reference Marker como: Braco_Sustentacao.CM

Restringir o Braco_Sustentacao com uma Revolute Joint (Movimento de elevacao):

- Nomear como: JR_Elevacao
- Nomear o Marker fixo como: JR_Elevacao_Fixo
- Nomear o Marker movel como: JR_Elevacao_Movel
- Posicao dos Markers: 0.00, 524.7, 0.00
- Orientacao dos Markers: 270.0, 90.0, (270+57.86)

O Tools > Model Verify deve indicar 2 GDL

Etapa 5 - Configurar o corpo do Helicóptero

Importar 4_Helicoptero.x_t (com propriedades de massa dos Cabos_Helicoptero):

- Nomear como: Helicoptero
- Adicionar no Helicoptero o Marker do centro de massa e nomear como: CM
- Posicao do CM (mm): 0.00, 325.23, 265.04
- Orientacao do CM (Angulos Z-X-Z de Euler): 180.00, 57.2648, 180.00

- Massa (g): 1453.37
- Px = 4419642.11
- Py = 66679650.27
- Pz = 69780236.70
- Indicar o Center of Mass Marker como: Helicoptero.CM
- Indicar o Inertia Reference Marker como: Helicoptero.CM

Restringir o Helicoptero com uma Revolute Joint (Movimento de arfagem):

- Nomear como: JR_Arfagem
- Nomear o Marker fixo como: JR_Arfagem_Fixo
- Nomear o Marker movel como: JR_Arfagem_Movel
- Posicao dos Markers: 0.00, 329.01, 269.14
- Orientacao dos Markers: 180.0, (90+57.86), 270.0

O Tools > Model Verify deve indicar 3 GDL

Etapa 6 - Configurar a Forca de Empuxo Dianteiro:

Criar a variavel de entrada da Forca de Empuxo Dianteiro (u1):

- Nomear como: u1_Forca_Empuxo_Dianteiro

Adicionar uma Force (Single-Component) no Helicóptero:

- Configurar Run-Time Direction como: Body Moving
- Configurar Characteristic como: Custom
- Nomear como: Empuxo_Dianteiro
- Nomear o primeiro Marker como: Empuxo_Dianteiro_1
- Nomear o segundo Marker como: Empuxo_Dianteiro_2
- Posicao dos Markers: -242.00, 327.82, 268.39
- Orientacao dos Markers: 180.0, 57.86, 180.0

- Alterar a Function:
  - Runtime function: VARVAL(u1_Forca_Empuxo_Dianteiro)
  - Getting Object Data: Results Data

Etapa 7 - Configurar a Forca de Empuxo Traseiro:

Criar a variavel de entrada da Forca de Empuxo Traseiro (u2):

- Nomear como: u2_Forca_Empuxo_Traseiro

Adicionar uma Force (Single-Component) no Helicoptero:

- Configurar Run-Time Direction como: Body Moving
- Configurar Characteristic como: Custom
- Nomear como: Empuxo_Traseiro
- Nomear o primeiro Marker como: Empuxo_Traseiro_1
- Nomear o segundo Marker como: Empuxo_Traseiro_2
- Posicao dos Markers: 242.00, 327.82, 268.39
- Orientacao dos Markers: 180.0, 57.86, 180.0

- Alterar a Function:
  - Runtime function: VARVAL(u2_Forca_Empuxo_Traseiro)
  - Getting Object Data: Results Data

Etapa 8 - Configurar as variaveis de saida:

Criar a variavel de saida da Posicao Angular de Deslocamento (y1):

- Nomear como: y1_Posicao_Angular_Deslocamento
- Criar a Measure: y1_Posicao_Angular_Deslocamento_VALUE_1
- Configurar F(time, ...)=:
  - Runtime Function: y1_Posicao_Angular_Deslocamento_VALUE_1
  - Getting Object Data: Results Data
- Configurar a Measure y1_Posicao_Angular_Deslocamento_VALUE_1:
  - Function Measure: AZ(JR_Deslocamento_Movel, JR_Deslocamento_Fixo)
  - Units: angle
  - Getting Object Data: Results Data

Criar a variavel de saida da Posicao Angular de Elevacao (y2):

- Nomear como: y2_Posicao_Angular_Elevacao
- Criar a Measure: y2_Posicao_Angular_Elevacao_VALUE_1
- Configurar F(time, ...)=:
  - Runtime Function: y2_Posicao_Angular_Elevacao_VALUE_1
  - Getting Object Data: Results Data
- Configurar a Measure y2_Posicao_Angular_Elevacao_VALUE_1:

- Function Measure: AZ(JR_Elevacao_Movel, JR_Elevacao_Fixo)
- Units: angle
- Getting Object Data: Results Data

Criar a variavel de saida da Posicao Angular de Arfagem (y3):

- Nomear como: y3_Posicao_Angular_Arfagem
- Criar a Measure: y3_Posicao_Angular_Arfagem_VALUE_1
- Configurar F(time, ...)=:
  - Runtime Function: y3_Posicao_Angular_Arfagem_VALUE_1
  - Getting Object Data: Results Data
- Configurar a Measure y3_Posicao_Angular_Arfagem_VALUE_1:
  - Function Measure: AZ(JR_Arfagem_Movel, JR_Arfagem_Fixo)
  - Units: angle
  - Getting Object Data: Results Data

Criar a variavel de saida da Velocidade Angular de Deslocamento (y4):

- Nomear como: y4_Velocidade_Angular_Deslocamento
- Criar a Measure: y4_Velocidade_Angular_Deslocamento_VALUE_1
- Configurar F(time, ...)=:
  - Runtime Function: y4_Velocidade_Angular_Deslocamento_VALUE_1
  - Getting Object Data: Results Data
- Configurar a Measure y4_Velocidade_Angular_Deslocamento_VALUE_1:
  - Function Measure: WZ(JR_Deslocamento_Movel, JR_Deslocamento_Fixo, JR_Deslocamento_Fixo)
  - Units: angular_velocity
  - Getting Object Data: Results Data

Criar a variavel de saida da Velocidade Angular de Elevacao (y5):

- Nomear como: y5_Velocidade_Angular_Elevacao
- Criar a Measure: y5_Velocidade_Angular_Elevacao_VALUE_1
- Configurar F(time, ...)=:
  - Runtime Function: y5_Velocidade_Angular_Elevacao_VALUE_1
  - Getting Object Data: Results Data
- Configurar a Measure y5_Velocidade_Angular_Elevacao_VALUE_1:
  - Function Measure: WZ(JR_Elevacao_Movel, JR_Elevacao_Fixo, JR_Elevacao_Fixo)
  - Units: angular_velocity
  - Getting Object Data: Results Data

Criar a variavel de saida da Posicao Angular de Arfagem (y6):

- Nomear como: y6_Velocidade_Angular_Arfagem
- Criar a Measure: y6_Velocidade_Angular_Arfagem_VALUE_1
- Configurar F(time, ...)=:
  - Runtime Function: y6_Velocidade_Angular_Arfagem_VALUE_1
  - Getting Object Data: Results Data
- Configurar a Measure y6_Velocidade_Angular_Arfagem_VALUE_1:
  - Function Measure: WZ(JR_Arfagem_Movel, JR_Arfagem_Fixo, JR_Arfagem_Fixo)
  - Units: angular_velocity
  - Getting Object Data: Results Data

Etapa 9 - Configurar o corpo do Contrapeso:

Importar 5_Contrapeso.x_t:

- Nomear como: Contrapeso
- Adicionar no Contrapeso o Marker do centro de massa e nomear como: CM
- Posicao do CM (mm): 0.00, 730.41, -267.52
- Orientacao do CM (Angulos Z-X-Z de Euler): 180.0000, 12.7787, -90.0000

- Massa (g): 1618.73
- Px = 859362.16
- Py = 859408.12
- Pz = 1163964.18
- Indicar o Center of Mass Marker como: Contrapeso.CM
- Indicar o Inertia Reference Marker como: Contrapeso.CM

Restringir o Contrapeso com uma Fixed Joint:

- Nomear como: JF_Contrapeso
- Nomear o Marker fixo como: JF_Contrapeso
- Nomear o Marker movel como: JF_Contrapeso
- Posicao dos Markers: 0.00, 735.38, -245.73
- Orientacao dos Markers: 0.0, 167.14, 180.0

O Tools > Model Verify deve indicar 3 GDL

APÊNDICE C – CAPÍTULO 5

```
clear all; close all; clc; format long;

% Parâmentros da Aquisição

f = 800;    % Frequência de amostragem (Hz) - [FrequÃªncia do PWM deve ser a mesma]
st = 1/f;   % Tempo de amostragem (s)
t = 10 - st; % Tempo total da aquisição (s)
```

```
%% Script para salvar dados da aquisição

% Este Script deve ser inicilizado logo após a aquisição em [E_model_daq_lcc]

% Massas(kg)utilizadas:
m = [0.0;               % 0.0 kg  = an = 1
     0.1;               % 0.1 kg  = an = 2
     0.2;               % 0.2 kg  = an = 3
     0.3;               % 0.3 kg  = an = 4
     0.4;               % 0.4 kg  = an = 5
     0.5;               % 0.5 kg  = an = 6
     0.6;               % 0.6 kg  = an = 7
     0.7;               % 0.7 kg  = an = 8
     0.8;               % 0.8 kg  = an = 9
     0.9;               % 0.9 kg  = an = 10
     1.0                % 1.0 kg  = an = 11
];

% Número da aquisição
an = 11;

% Frequência 800Hz
hz = 800;

% Estrutura contendo os dados:
data.sample_time = st;   %Tempo de amostragem (s)
data.total_time = t;     %Tempo total da aquisição (s)
data.mass = m(an);       %Matriz com as massas utilizadas (kg)
data.signal = simout;    %Matriz com os sinais de tensão aquisitados

% Salva os dados na pasta "data"
save(sprintf('data/Hz_%d_an_%d',hz,an),'data');  %Hz_(frequência)_an_(massa)
```

```
%% Script para aproximação polinomial

%Este Script deve ser utilziado após ter todas as aquisições de todos os
%pesos em avaliação

clear all; close all; clc; format long;

%% Parâmetros
hz = 800;                % Frequência de aquisição
dan = 1:11;              % Vetor com número de aquisições = Número de massas
po = 1;                  % Grau do polinômio de aproximação
plot_lc_signal = false;   % Plotar os sinais de cada aquisição (true=sim, false=não)


%% Processamento dos Sinais aquisitados

%loop "for" acessando cada posição do vetor "dan" por iteração
for an = dan
    % Carrega os dados para a massa (an) correspondente
    load(sprintf('data/Hz_%d_an_%d',hz,an));

    if an == 1
        % Vetor tempo
        lcs.t = (0:data.sample_time:data.total_time)';
    end

    % Armazena a força aplicada em cada aquisição (converte de kg para N)
    lcs.w(an,1) = data.mass * 9.81;

    % Armazena o sinal de aquisição (uma coluna para cada massa)
```

```
    lcs.y(:,an) = data.signal;

    % Armazena a média do sinal, subtraindo o offset inicial da célula de carga
    lcs.y_mean(an,1) = mean(lcs.y(:,an)) - mean(lcs.y(:,1));

    % Plot dos sinais da célula de carga (original e filtrado)
    if plot_lc_signal
        figure;
        plot(lcs.t,lcs.y(:,an));
        title(sprintf('Aquisição: %d. Força Aplicada: %dN. Frequência:
%dHz.',an,lcs.w(an,1),hz));
        xlabel('Tempo (s)'); ylabel('Tensão (V)');
        legend('sinal original');
        grid on;
    end
end

%%
figure;
plot(lcs.t,lcs.y(:,11));
title('Aquisição: 11. Força Aplicada: 9.8100N. Frequência: 800Hz.');
xlabel('Tempo (s)'); ylabel('Tensão (V)');
xlim([0 5]);
legend('sinal original');
grid on;

%%
% Ajute polinomial
P_lc = polyfit(lcs.y_mean,lcs.w,po);


%% Plot da avaliação do ajuste polinomial
V = lcs.y_mean;

figure;
plot(V,polyval(P_lc,V),'LineWidth',1);
hold on;
plot(V,lcs.w,'x','LineWidth',1);
grid on;
title('Avaliação do Ajuste Polinomial.');
xlabel('Tensão (V)','FontSize',13); ylabel('Força (N)','FontSize',13);
xlim([0 V(end)]); ylim([0 lcs.w(end)]);
legend('ajuste polinomial de 1^o grau','pontos de calibração','Location','NorthWest');
ax = gca;
ax.FontSize = 12;

%% Salva o polinômio da célula de carga na pasta "polynomial"
save(sprintf('polynomial/P_lc_%dHz',hz),'P_lc')

%% Limpa variáveis auxiliares
% clearvars -except lcs P_lc;

%% Referências

%https://www.mathworks.com/help/matlab/ref/polyfit.html
%https://www.mathworks.com/help/matlab/ref/polyval.html
```

```
clear all; close all; clc; format long;

% Parâmetros
f =  800;          % Frequência de amostragem da carga estática (Hz)
st = 1/f;          % Tempo de amostragem
t_bias = 10 - st; % Tempo total da aquisição para determinar o offset do sistema

%variável bias será jogada ao workspace
```

```
%% Script para iniciar a aquisição de identificação

% Calcula média dos valores de offset com os dados em bias
offset = mean(bias(:,1));
hz=800;  %frequencia da calibragem da celula de carga

% Carrega o polinômio da célula de carga
load(sprintf('polynomial/P_lc_%dHz',hz));
```

```
% Parâmetros:
f = 200;           % Frequência de amostragem (Hz)
st = 1/f;          % Tempo de amostragem (s)

% Parâmetros para Degrau de PWM:
num_intervalos = 8;                  % Número de degraus do PWM
t_intervalo = 5;                      % Tempo de aquisição em cada degrau (s)
amostras_intervalo = t_intervalo/st;  % Número de amostras em cada intervalo
amostras_ensaio_total = amostras_intervalo*num_intervalos*2; % Total de amostras (subida e
descida)
ts = amostras_ensaio_total*st;               % Tempo total (s)
sinal_pwm = zeros(1,amostras_ensaio_total);  % Vetor de 0s com o número de amostras totais
passo = 10;                                  % Incremento do PWM
pwm_porcentagem = passo;                     % Valor inicial do PWM
time = [1:amostras_ensaio_total]*st;         % Vetor tempo (avanço amostral)

for i=1:amostras_ensaio_total %laço percorrendo todo Vetor Tempo
   if(i<=(amostras_ensaio_total/2))                                          % subida
      if(mod(i,amostras_intervalo)==0 && i<(amostras_ensaio_total/2))  % Se já percorrido um
degrau E ainda na subida
         pwm_porcentagem = pwm_porcentagem+passo;                           % aumenta-se o PWM
      end
   else                                                                      %descida
       if(mod(i,amostras_intervalo)==0 && i<(amostras_ensaio_total))    % Se já percorrido um
degrau E na descida
         pwm_porcentagem = pwm_porcentagem-passo;                           % diminui-se o PWM
       end
   end
   sinal_pwm(i) = pwm_porcentagem/100;                                       %sinal pwm em relação
a amostra
end

entrada = [time' sinal_pwm'];                                                %entrada para o
simulink
```

```
%Variáveis vindo em Dados_Motor_n :

% Coluna 1    Coluna 2     Coluna 3
% Raw Data   Force Data    Time Data

amostras_consideradas = 2/st;                                                %retirar n
segundos de cada amostra
faixas_empuxo_crescente = zeros(amostras_intervalo,num_intervalos);          %matriz     linhas
= numero de amostras por degrau; colunas = numero de intervalos de pwm
faixas_empuxo_decrescente = zeros(amostras_intervalo,num_intervalos);        %matriz     linhas
= numero de amostras por degrau; colunas = numero de intervalos de pwm

%faixas_empuxo_crescente:
%Coluna 1   %Coluna 2  %Coluna 3   %Coluna 4  %Coluna 5  ......
%10% PWM     20% PWM    30% PWM     40% PWM    50% PWM

%faixas_empuxo_decrescente:
%Coluna 1   %Coluna 2  %Coluna 3   %Coluna 4  %Coluna 5  ......
%80% PWM     70% PWM    60% PWM     50% PWM    60% PWM

%O loop for a seguir, separa os dados  da segunda coluna (Force Data) e os
%coloca nas colunas de faixas_empuxo_crescente e faixas_empuxo_decrescente
%de acordo com o PWM aplicado.

k=1;    %seleciona a coluna de intensidade de PWM
aux=1;  %auxiluar para guardar linha de (Force Data)

for i=1:num_intervalos*2         %varrendo todos os intervalos [crescente/decrescente]
 if(i<=num_intervalos)           %crescente
 faixas_empuxo_crescente(:,k) = Dados_Motor_1(aux:((amostras_intervalo*i)),2);   %Coloca na
coluna de PWM os dados da coluna (Force Data), indo da linha aux até a linha daquele intervalo
 else                            %decrescente
 faixas_empuxo_decrescente(:,k) = Dados_Motor_1(aux:((amostras_intervalo*i)),2); %Coloca na
coluna de PWM os dados da coluba (Force Data), indo da linha aux até a linha daquele intervalo
 end
 aux =(amostras_intervalo*i)+1;  %Linha Inicial de um degrau de PWM em (Force Data)
 if(i==num_intervalos)           %degrau topo
   k=1;                          %reinicia auxiliar de coluna para processo decrescente
 else
   k=k+1;                        %incrementa auxiliar de coluna
 end
```

```
end


fec_faixa_estavel = faixas_empuxo_crescente(amostras_consideradas:amostras_intervalo,:);
%retira os 2 primeiros segundos de cada degrau de PWM
fed_faixa_estavel = faixas_empuxo_decrescente(amostras_consideradas:amostras_intervalo,:);
%retira os 2 primeiros segundos de cada degrau de PWM


valor_medio_empuxo = zeros(1,num_intervalos);   %vetor para guardar as médias de forças para
cada PWM
k=num_intervalos+1;                             %auxiliar para ultima coluna do vetor
descrescente

for i=1:num_intervalos  %andando por todo vetor de médias
valor_medio_empuxo(:,i) = mean([fec_faixa_estavel(:,i);fed_faixa_estavel(:,k-i)]); %calculo de
força referente a cada pwm (crescente e decrescente)
end
%%

save(sprintf('data/Cenario_M2_20'),'valor_medio_empuxo','fed_faixa_estavel','fec_faixa_estavel
','num_intervalos');

%% Plot da Curva

valor_medio_empuxo = [ 0 valor_medio_empuxo]; %adicionando 0 a primeira posição do vetor
PWM = [0:1:num_intervalos]*10;                % PWM de 0 até 100

po = 3;
P_M = polyfit(PWM,valor_medio_empuxo,po);

figure;
plot(PWM,valor_medio_empuxo,'ro')
grid on
xlabel('PWM(%)');
ylabel('Empuxo(N)');
hold on;
plot(PWM,polyval(P_M,PWM),'LineWidth',1);

%%
save(sprintf('data/P_M2_20'),'P_M');
```

```
%%PLOT DA CURVA

%M1 = MOTOR TRASEIRO
%M2 = MOTOR DIANTEIRO

clear; clc; format long;

%carrega cenário de aquisição
load(sprintf('data/Cenario_M2_2'),'valor_medio_empuxo','fed_faixa_estavel','fec_faixa_estavel'
,'num_intervalos');

%valor médio => colunas em linhas
for i=1:num_intervalos
    valor_medio(i,1) = valor_medio_empuxo(:,i);
end
valor_medio = [0; valor_medio];

%criando variável PWM e a carregando com as variáveis
for i=1:num_intervalos
    PWM(i,1) = (i*10);
end
PWM = [0;PWM];

po = 3;  %nível do ajuste polinimial
P = polyfit(PWM,valor_medio,po); %coeficientes do polinômio de ajuste


figure;
plot(PWM,valor_medio,'ro')
grid on
title('Motor Dianteiro');
xlabel('PWM(%)','FontSize',13);
ylabel('Empuxo(N)','FontSize',13);
hold on;
```

```
plot(PWM,polyval(P,PWM),'LineWidth',1);  %eixo x = PWM   eixo y = Valores do polinômio P nos
pontos PWM
legend('média amotral','ajuste polinomial de 3^o grau','Location','NorthWest');
ax = gca;
ax.FontSize = 12;


po2 = 3;
P_M = polyfit(valor_medio,PWM,po2);

figure;
plot(valor_medio,PWM,'ro')
grid on
title('Motor Dianteiro');
xlabel('Empuxo(N)','FontSize',13);
ylabel('PWM(%)','FontSize',13);
hold on;
plot(valor_medio,polyval(P_M,valor_medio),'LineWidth',1);
legend('média amotral','ajuste polinomial de 3^o grau','Location','NorthWest');
ax = gca;
ax.FontSize = 12;

save(sprintf('polynomial/P_MD_2'),'P_M');
```

```
%Resultado medio

%M1 = MOTOR TRASEIRO
%M2 = MOTOR DIANTEIRO
clear; clc; format long;

n_experimentos = 20; %numero de experimentos em cada motor
grau_polinomio = 3; %grau do polinomio
polinomios_d = zeros(20,grau_polinomio+1); %matriz para armazenar os polinomios do motor
dianteiro
polinomios_t = zeros(20,grau_polinomio+1); %matriz para armazenar os polinomios do motor
traseiro
force = 0:0.1:0.8; %eixo x, contendo a força em N

for i=1:n_experimentos
    load(sprintf('polynomial/P_MD_%d',i));
    polinomios_d(i,:) = P_M;
end

for k=1:n_experimentos
    load(sprintf('polynomial/P_MT_%d',k));
    polinomios_t(k,:) = P_M;
end

figure;
for j=1:n_experimentos
    plot(force,polyval(polinomios_d(j,:),force),'LineWidth',1)
    hold on;
end

grid on
title('Motor Dianteiro');
xlabel('Empuxo(N)','FontSize',13);
ylabel('PWM(%)','FontSize',13);
legend('Testes realizados','Location','NorthWest');
ax = gca;
ax.FontSize = 12;

figure;
for l=1:n_experimentos
    plot(force,polyval(polinomios_t(l,:),force),'LineWidth',1)
    hold on;
end

grid on
title('Motor Traseiro');
xlabel('Empuxo(N)','FontSize',13);
ylabel('PWM(%)','FontSize',13);
legend('Testes realizados','Location','NorthWest');
ax = gca;
ax.FontSize = 12;
```

```
%%

Rep=20;  % quantidade de experimentos

for p=1:Rep
    heapnorm{p} = polyval(polinomios_d(p,:),force); %polinomios_d / polinomios_t
    p
end

[vecMean_norm,vecStd_norm] = findMeanStd(heapnorm);

vecAmostras= 0:0.1:0.8;

figure;
plot(vecAmostras, vecMean_norm,'k-', vecAmostras,vecMean_norm+vecStd_norm, 'b-.',...
    vecAmostras,vecMean_norm-vecStd_norm, 'b-.');
title('Desvio Padrão - Motor Dianteiro');
grid on;
xlabel('Empuxo(N)','FontSize',13);
ylabel('PWM(%)','FontSize',13);
legend('curva de Ajuste');
ax = gca;
ax.FontSize = 12;
```

APÊNDICE D – CAPÍTULO 6

```
%% Script do controle seguidor com realimentação de estados
clear all; close all; clc; format short;

%% Configuração geral
% Polinômio dos motores
load('_polinomios/P_MD.mat');
P1 = P_M;
load('_polinomios/P_MT.mat');
P2 = P_M;
clear P_M;

% Parâmetros da estimação da posição e velocidade angular pelo tracking loop
TL_Kp = 60;
TL_Ki = 1200;

% Parâmetros da simulação
st = 1/200; % tempo de amostragem (s)
t = 10; % tempo de simulação (s)

%% Projeto do controlade seguidor como realimentação de estados
% Carrega o modelo linear contínuo
load('_modelos/SYSC.mat');
load('_modelos/uop.mat');

% Converte para o modelo discreto
SYSD = c2d(SYSC,st);
invC = inv(SYSC.C);

% Atribuição de autoestruta completa
E = SYSC.C([1 2],:);
% SIGMA = [-0.85+0.11i -0.85-0.11i -0.80+0.10i -0.80-0.10i -0.90+0.12i -0.90-0.12i -1.00 -
0.95];
SIGMA = [-1.20+0.12i -1.20-0.12i -0.85+0.11i -0.85-0.11i -0.85+0.10i -0.85-0.10i -1.50 -0.95];
Qf = [-1  0 -1  0 -1  0 -1 -1;
      -1  0  1  0  1  0  1 -1];
[K1,K2] = eeaa(SYSC,E,SIGMA,Qf)

% Salvar dados nos arquivos
dados_sensores = 'dados_sensores';
```

```
%% Script de plot dos resultados
clear all; close all; clc;

%% Script de plot dos resultados
clear all; close all; clc;

%% Carrega os dados

%load('_dados/dados_sensores_arfagem_1.mat');
load('_dados/dados_sensores_deslocamento_1.mat');

%% Configuração geral
linewidth = 1.5;
fs_axis_label = 11;
fs_legend = 14;
fs_axis_tick = 11;

% Velocidade Angular
plot(dados_sensores.time,dados_sensores.signals.values(:,5),'b','LineWidth',linewidth);
title('Experimento 12 (Deslocamento)', 'FontSize', 10);
hold on;
plot(dados_sensores.time,dados_sensores.signals.values(:,3),'r','LineWidth',linewidth);
grid on;
plot(dados_sensores.time,dados_sensores.signals.values(:,4),'g','LineWidth',linewidth);
grid on;
xlabel('Tempo (s)','FontSize',fs_axis_label); ylabel('Velocidade Angular
(rad/s)','FontSize',fs_axis_label);
l = legend('$\dot{p}_{e}$','$\hat{v}_{e}$','$\hat{v}_{e2}$');
set(l,'Interpreter','Latex');
set(l,'FontSize',fs_legend);
ylim([-0.5 0.5]);
```

```
xlim([6.5 7.5]);
ax = gca;
ax.FontSize = fs_axis_tick;

% Salva em arquivo
print('_resultados/deslocamento','-djpeg');
```

APÊNDICE E – CAPÍTULO 8

```
%% Script do controle seguidor com realimentação de estado projetado pelo
%  método de atribuição de autoestrutura
clear all; close all; clc; format short;

%% Configuração geral

% Carrega o polinômio de Empuxo/PWM dos atuadores
load('_polinomios/P_MD.mat');
P1 = P_M;
load('_polinomios/P_MT.mat');
P2 = P_M;
clear P_M;

% Parâmetros da estimação da posição e velocidade angular pelo tracking loop
TL_Kp = 40;
TL_Ki = 900;

% Parâmetros da simulação
st = 1/200; % tempo de amostragem (s)
t = 80; % tempo de simulação (s)

%% Carrega o modelo não linear do Adams
B_nao_linear_;

%% Projeto do controle seguidor com realimentação de estado

% Carrega o modelo linear contínuo
load('_modelos/SYSC.mat');
load('_modelos/uop.mat');

% Discretização do modelo linear
SYSD = c2d(SYSC,st);
invC = inv(SYSC.C);
S = [1 0 0 0 0 0;
     0 1 0 0 0 0];

% Atribuição de autoestruta completa
E = SYSC.C([1 2],:);
SIGMA = [-1.20+0.12i -1.20-0.12i -0.85+0.11i -0.85-0.11i -0.85+0.10i -0.85-0.10i -1.50 -0.95];
%SIGMA = [desloc_v desloc_p elev_v elev_p arfg_v arfg_a integ_desloc integ_elev]


Qf = [-1  0  -1  0 -1  0 -1 -1;
       1  0  -1  0  1  0  1 -1];

% Ganhos discretos
[K1,K2] = deeaa(SYSC,E,SIGMA,Qf,st)
% Nome dos arquivos para salvar os dados
dados_linear = 'linear_d';
dados_naolinear = 'naolinear_d';
dados_planta = 'planta_d';
```

```
%% Plot da comparação dos modelos
close all; clc;

%% Carrega os dados do controle com modelo linear
load('_dados/linear_d.mat');
%% Carrega os dados do controle com modelo não linear
load('_dados/naolinear_d.mat');
%% Carrega os dados reais
load('_dados/planta_d.mat');

%linear.signals.values(:,1)  = Referência de deslocamento
%linear.signals.values(:,2)  = Referência de elevação
%linear.signals.values(:,3)  = Empuxo dianteiro
%linear.signals.values(:,4)  = Empuxo traseiro
%linear.signals.values(:,5)  = Posição de deslocamento
%linear.signals.values(:,6)  = Posição de elevação
%linear.signals.values(:,7)  = Posição de arfagem
%linear.signals.values(:,8)  = Velocidade de deslocamento
%linear.signals.values(:,9)  = Velocidade de elevação
%linear.signals.values(:,10) = Velocidade de arfagem
```

```
% Posição angular deslocamento-------------------------------------------------
figure;
subplot(2,1,1);
hold on;
plot(linear.time,rad2deg(linear.signals.values(:,1)),'--k','LineWidth',1.2);
plot(planta.time,rad2deg(planta.signals.values(:,5)),'LineWidth',1.2);
plot(naolinear.time,rad2deg(naolinear.signals.values(:,5)),'LineWidth',1.2);
plot(linear.time,rad2deg(linear.signals.values(:,5)),'LineWidth',1.2);
hold off;
grid on;
ylabel('Deslocamento (graus)','FontSize',12);
l = legend('referência','planta','não linear','linear','Location','SouthEast');
set(l,'FontSize',10);
ax = gca;
ax.FontSize = 12;
% ylim([-45 45]);

% Velocidade angular deslocamento
subplot(2,1,2);
hold on;
plot(planta.time,rad2deg(planta.signals.values(:,8)),'LineWidth',1.2);
plot(naolinear.time,rad2deg(naolinear.signals.values(:,8)),'LineWidth',1.2);
plot(linear.time,rad2deg(linear.signals.values(:,8)),'LineWidth',1.2);
hold off;
grid on;
xlabel('Tempo (s)')
ylabel('Velocidade (graus/s)','FontSize',12);
l = legend('planta','não linear','linear','Location','SouthEast');
set(l,'FontSize',10);
ax = gca;
ax.FontSize = 12;
% ylim([-45 45]);
%-------------------------------------------------------------------------------


% Posição angular elevação-----------------------------------------------------
figure;
subplot(2,1,1);
hold on;
plot(linear.time,rad2deg(linear.signals.values(:,2)),'--k','LineWidth',1.2);
plot(planta.time,rad2deg(planta.signals.values(:,6)),'LineWidth',1.2);
plot(naolinear.time,rad2deg(naolinear.signals.values(:,6)),'LineWidth',1.2);
plot(linear.time,rad2deg(linear.signals.values(:,6)),'LineWidth',1.2);
hold off;
grid on;
ylabel('Elevação (graus)','FontSize',12);
l = legend('referência','planta','não linear','linear','Location','SouthEast');
set(l,'FontSize',10);
ax = gca;
ax.FontSize = 12;
% ylim([0 50]);

% Velocidade angular de elevação
subplot(2,1,2);
hold on;
plot(planta.time,rad2deg(planta.signals.values(:,9)),'LineWidth',1.2);
plot(naolinear.time,rad2deg(naolinear.signals.values(:,9)),'LineWidth',1.2);
plot(linear.time,rad2deg(linear.signals.values(:,9)),'LineWidth',1.2);
hold off;
grid on;
xlabel('Tempo (s)')
ylabel('Velocidade (graus/s)','FontSize',12);
l = legend('planta','não linear','linear','Location','SouthEast');
set(l,'FontSize',10);
ax = gca;
ax.FontSize = 12;
% ylim([0 50]);
%-------------------------------------------------------------------------------


% Posição angular arfagem------------------------------------------------------
figure;
subplot(2,1,1);
hold on;
plot(planta.time,rad2deg(planta.signals.values(:,7)),'LineWidth',1.2);
plot(naolinear.time,rad2deg(naolinear.signals.values(:,7)),'LineWidth',1.2);
```

```
plot(linear.time,rad2deg(linear.signals.values(:,7)),'LineWidth',1.2);
hold off;
grid on;
ylabel('Arfagem (graus)','FontSize',12);
l = legend('planta','não linear','linear');
set(l,'FontSize',10);
ax = gca;
ax.FontSize = 12;

% Velocidade angular de arfagem
subplot(2,1,2);
hold on;
plot(planta.time,rad2deg(planta.signals.values(:,10)),'LineWidth',1.2);
plot(naolinear.time,rad2deg(naolinear.signals.values(:,10)),'LineWidth',1.2);
plot(linear.time,rad2deg(linear.signals.values(:,10)),'LineWidth',1.2);
hold off;
grid on;
xlabel('Tempo (s)','FontSize',12);
ylabel('Velocidade (graus/s)','FontSize',12);
l = legend('planta','não linear','linear');
set(l,'FontSize',10);
ax = gca;
ax.FontSize = 12;
%-------------------------------------------------------------------------------


%% Plot das ações de controle-----------------------------------------------------
figure;
% Empuxo dianteiro
subplot(2,1,1);
hold on;
plot(planta.time,planta.signals.values(:,3),'LineWidth',1.2);
plot(naolinear.time,naolinear.signals.values(:,3),'LineWidth',1.2);
plot(linear.time,linear.signals.values(:,3),'LineWidth',1.2);
hold off;
grid on;
ylabel('Empuxo Dianteiro (N)');
l = legend('planta','não linear','linear','Location','SouthEast');
set(l,'FontSize',10);
ax = gca;
ax.FontSize = 12;

% Empuxo traseiro
subplot(2,1,2);
hold on;
plot(planta.time,planta.signals.values(:,4),'LineWidth',1.2);
plot(naolinear.time,naolinear.signals.values(:,4),'LineWidth',1.2);
plot(linear.time,linear.signals.values(:,4),'LineWidth',1.2);
hold off;
grid on;
xlabel('Tempo (s)');
ylabel('Empuxo Traseiro (N)');
l = legend('planta','não linear','linear','Location','SouthEast');
set(l,'FontSize',10);
ax = gca;
ax.FontSize = 12;
%-------------------------------------------------------------------------------
```