

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO



Rafael Gruber Bernstein

**Modelagem acoplada de uma Plataforma Stewart
acionada por atuadores elétricos utilizando Grafos de
Ligação**

Projeto de Graduação

Projeto de Graduação apresentado ao Departamento de Engenharia
Mecânica da PUC-Rio
Orientador: Allan Nogueira de Albuquerque

Rio de Janeiro
Dezembro de 2024

AGRADECIMENTOS

Agradeço ao meu orientador.

Agradeço aos professores e a coordenação de engenharia da PUC-Rio pelo apoio contínuo durante meus estudos

RESUMO

Modelagem acoplada de uma Plataforma Stewart acionada por atuadores elétricos utilizando Grafos de Ligação

O trabalho tem como objetivo desenvolver a modelagem acoplada de uma Plataforma Stewart, um manipulador paralelo que possui seis graus de liberdade e é acionado por atuadores elétricos. A Plataforma Stewart é amplamente utilizada em diversas áreas, como simulação de voo e robótica, devido à sua capacidade de realizar movimentos complexos com alta precisão. Para otimizar o controle do sistema, é fundamental criar modelos dinâmicos que possam ser computados em tempo real.

A metodologia proposta inclui a realização de experimentos para a obtenção dos parâmetros do modelo, utilizando Grafos de Ligação (*Bond Graphs*) para estruturar a dinâmica do sistema. Essa abordagem permite uma visualização clara e organizada das equações e restrições envolvidas. Além disso, serão implementados controladores robustos que visam melhorar a precisão do rastreamento de trajetória, minimizando os erros gerados por variáveis desconhecidas, distúrbios externos e dinâmicas não modeladas.

A pesquisa não só busca contribuir para o entendimento da dinâmica da Plataforma Stewart, mas também para o desenvolvimento de técnicas de modelagem e controle que possam ser aplicadas em sistemas similares. Espera-se que os resultados obtidos proporcionem avanços significativos na eficiência e na precisão dos manipuladores paralelos.

Palavras-chave: Plataforma Stewart, manipulador paralelo, modelagem dinâmica, controle robusto, gráficos de ligação.

ABSTRACT

Coupled Modeling of a Stewart Platform Driven by Electric Actuators Using Bond Graphs.

The aim of this work is to develop the coupled modeling of a Stewart Platform, a parallel manipulator with six degrees of freedom, powered by electric actuators. The Stewart Platform is widely used in various fields, such as flight simulation and robotics, due to its ability to perform complex movements with high precision. To optimize the control of the system, it is essential to create dynamic models that can be computed in real time.

The proposed methodology includes conducting experiments to obtain the model parameters, utilizing Bond Graphs to structure the dynamics of the system. This approach allows for a clear and organized visualization of the equations and constraints involved. Additionally, robust controllers will be implemented to improve trajectory tracking accuracy, minimizing errors caused by unknown variables, external disturbances, and unmodeled dynamics.

This research aims not only to contribute to the understanding of the dynamics of the Stewart Platform but also to develop modeling and control techniques that can be applied to similar systems. It is expected that the results obtained will provide significant advances in the efficiency and precision of parallel manipulators.

Keywords: Stewart Platform, parallel manipulator, dynamic modeling, robust control, Bond Graphs.

Sumário

1. Introdução	10
1.1. Motivação	10
1.2. Objetivos	12
1.3. Estrutura do Trabalho	12
2. Fundamentação Teórica	14
2.1. Mecanismos de cadeia fechada	14
2.1.1 Graus de liberdade	15
2.1.2. Cinemática via cadeia vetorial	16
2.2. Dinâmica via fluxo de potência	17
2.3. Manipuladores Paralelos	18
2.3.1. Plataformas Stewart	18
2.4. Atuadores Elétricos	21
2.4.1. Curvas de desempenho de motores DC	22
2.5. Modelagem Acoplada com o uso de Grafos de Ligação	23
2.6. Modelo de Inércia das pernas	28
2.7. Arquitetura de Controle	29
3. Metodologia	30
3.1. Modelagem Matemática	30
3.1.1 Modelo 1 - Atuador elétrico	31
3.1.2 Modelo 2 - Plataforma Stewart	42
3.1.3 Modelo 3 - Modelagem acoplada	43
3.2. Simulações Numéricas	45
3.3. Validação Experimental	49
3.3.1. Experimento 1	49
3.3.2. Experimento 2	52
4. Discussão de Resultados	56
5. Conclusões e Considerações Finais	61
6. Referências Bibliográficas	62

7. Apêndice	64
7.1 Código MATLAB modelo simples e curva de velocidade experimental sem filtro	64
7.2 Código MATLAB sinal filtrado	69
7.3 Código MATLAB ajuste de curva.....	71
7.4 Código MATLAB ajuste de parâmetros.....	74
7.5 Código MATLAB comparação entre modelo e experimento + controle	82
7.6 Código Arduino	88

Lista de figuras

- 1.1.1 Exemplo de aplicação em um simulador de voo
- 2.5.1 Diagrama esquemático da Plataforma Stewart
- 2.5.2 Ângulos e posições de uma junta universal.
- 2.6.1 Esquema do atuador elétrico.
- 2.7.1 Resumo dos elementos de Grafos de Ligação.
- 3.1.1 Modelo do Grafo do atuador.
- 3.1.2 Modelo do Grafo do atuador com causalidades
- 3.1.3 Modelo em Grafos de Ligação do atuador com números identificadores.
- 3.1.4
- 3.1.5 Cinemática inversa da plataforma plana via Grafos de Ligação
- 3.1.6 Montagem acoplada via Grafos de Ligação (apenas 3 atuadores)
- 3.1.7 Montagem acoplada via Grafos de Ligação
- 3.1.8 Estratégia de controle do atuador (trabalhos futuros)
- 3.2.1 Curva de Força x Velocidade fornecida pelo fabricante
- 3.2.2 Modelo no Simulink.
- 3.2.3 Resposta da força do atuador para diferentes tensões e força externa 0
- 3.2.4 Resposta da velocidade do atuador para diferentes tensões e força externa 0
- 3.2.5 Esquema de controle no Simulink
- 3.3.1 Atuador elétrico LA 12.
- 3.3.2 Esquema de ligação do atuador com a fonte
- 3.3.3 Visualização real do experimento
- 3.3.4 Esquema de ligação do atuador com a fonte e Arduino
- 3.3.5 Velocidade do atuador sem filtro

- 3.3.6 Posição e velocidade do atuador após filtro passa-baixos
- 3.3.7 Comparação entre Curva ajustada e velocidade medida
- 4.1 Comparação entre modelo desenvolvido e experimentos para diferentes tensões
- 4.2 Comparação entre modelo calibrado e experimentos para diferentes tensões
- 4.3 Erro relativo para cada tensão
- 4.4 Esquema ajustado de controle no Simulink
- 4.5 Resultado da simulação de malha fechada para controle de posição.
- 4.6 Resultado do sinal da tensão para a simulação de malha fechada

Lista de tabelas

- | | |
|-------|---|
| 3.2.1 | Tabela com o valor dos parâmetros. |
| 3.3.1 | Tabela de medições do 1º experimento. |
| 3.3.2 | Tabela de velocidades e corrente do 1º experimento |
| 3.3.3 | Tabela de parâmetros das curvas medias de cada
tensão |
| 4.1 | Tabela com o valor dos parâmetros do modelo
calibrado. |

1. Introdução

Este estudo delinea a modelagem acoplada de uma plataforma Stewart, um manipulador paralelo de seis graus de liberdade, ativado por atuadores elétricos. Mediante a integração desses componentes, visa-se analisar e compreender o comportamento dinâmico do conjunto, bem como validar os modelos teóricos através de simulações numéricas e experiências práticas. A modelagem conjunta desses elementos viabilizará o desenvolvimento de estratégias de controle eficientes para aplicações em diversas áreas, como robótica, simulação de movimento, análise de desempenho e otimização de trajetórias, entre outras. A utilização de atuadores elétricos proporcionará maior precisão e velocidade nos movimentos da plataforma, resultando em maior eficiência e eficácia em suas aplicações. Compreender o comportamento dinâmico do sistema acoplado permitirá aprimorar o controle do manipulador paralelo, tornando-o mais preciso e confiável. A modelagem acoplada também possibilitará simulações em tempo real, proporcionando uma análise detalhada e precisa do comportamento do sistema em diferentes condições e cenários, o que permitirá a otimização dos parâmetros de controle e melhor desempenho da plataforma Stewart em diversas áreas, como a robótica industrial, medicina e indústria automotiva.

1.1. Motivação

A motivação para este Trabalho de Conclusão de Curso (TCC) surge em um contexto de crescente interesse global por sistemas robóticos avançados, que se destacam pela precisão e eficiência em uma variedade de aplicações, incluindo indústrias, medicina e entretenimento. A demanda por tecnologias que atendam a essas necessidades em constante evolução exige abordagens inovadoras e relevantes. Nesse cenário, a modelagem acoplada

da Plataforma Stewart movido por atuadores elétricos se destaca como uma solução promissora. Essa combinação não apenas representa um avanço significativo em desempenho, mas também oferece uma nova perspectiva sobre a confiabilidade e versatilidade dos sistemas robóticos.

Os atuadores elétricos, quando integrados à Plataforma Stewart, trazem uma série de vantagens que justificam a escolha deste tema para o TCC. Em primeiro lugar, a precisão e o controle oferecidos por esses atuadores são fundamentais para aplicações que requerem movimentos delicados e controlados. Isso é particularmente relevante em áreas como simulações médicas e treinamentos em ambientes virtuais, onde a exatidão dos movimentos pode impactar diretamente a eficácia do aprendizado e da prática. A capacidade de realizar ajustes finos na posição e na velocidade da plataforma permite que os sistemas robóticos se adaptem a diferentes cenários, aumentando sua aplicabilidade.



Figura 1.1.1 Plataforma Stewart do Laboratório de Desenvolvimento de Sistemas Mecatrônicos (LDSM).

Diferentemente dos atuadores hidráulicos, que requerem um sistema complexo de tubulações e reservatórios, os atuadores elétricos são geralmente mais compactos e fáceis de integrar. Essa menor complexidade

mecânica resulta em um sistema mais leve e, conseqüentemente, mais ágil, o que é fundamental em aplicações que demandam alta performance e precisão.

Além disso, a manutenção dos atuadores elétricos tende a ser menos cara e mais simples em comparação aos sistemas hidráulicos. A ausência de fluidos pressurizados reduz o risco de vazamentos e falhas mecânicas, tornando o sistema mais confiável e seguro. Essa confiabilidade é um fator essencial para aplicações críticas, onde a continuidade operacional é a prioridade.

Portanto, este TCC se propõe a estudar a modelagem desse sistema acoplado, que promete contribuir significativamente para o avanço das tecnologias robóticas e abrir novos horizontes para futuras inovações no setor.

1.2. Objetivos

Os objetivos deste trabalho incluem a modelagem matemática integrada da Plataforma Stewart com o manipulador paralelo acionado por atuadores elétricos, a realização de simulações numéricas detalhadas para avaliar o comportamento dinâmico do sistema de forma extremamente precisa, a execução de experimentos rigorosos para validar a precisão dos modelos teóricos, obtenção dos parâmetros do sistema e, finalmente, propor estratégias de controle eficientes e inovadoras para aplicações práticas em diversas áreas, abrangendo desde a indústria robótica até a medicina avançada.

1.3. Estrutura do Trabalho

A estrutura deste trabalho está organizada da seguinte forma: a introdução (Capítulo 1) apresenta a motivação e relevância da modelagem acoplada da Plataforma Stewart acionado por atuadores elétricos, com o objetivo de otimizar a eficiência e precisão nos sistemas robóticos. A

fundamentação teórica (Capítulo 2) discute os conceitos essenciais e teorias fundamentais da Plataforma Stewart, atuadores elétricos, modelagem acoplada e todos os tópicos necessários para a realização do trabalho, explorando as metodologias e princípios de funcionamento desses sistemas. A metodologia (Capítulo 3) detalha minuciosamente os procedimentos de modelagem matemática, simulações numéricas avançadas e validação experimental, utilizando técnicas aprimoradas de análise e métodos de otimização para obter resultados confiáveis e precisos. Os resultados e discussões (Capítulo 4) incluem uma análise abrangente e detalhada da dinâmica do sistema, comparando os resultados obtidos por modelos teóricos e experimentais, destacando principais diferenças e limitações inerentes. Por fim, as conclusões e considerações finais (Capítulo 5) oferecem uma síntese dos principais achados do estudo, abordando as implicações práticas e teóricas dos resultados e sugerindo possíveis direções futuras de pesquisa. As referências bibliográficas (Capítulo 6) são amplas e diversificadas, englobando estudos recentes e clássicos de renomados pesquisadores na área de robótica e controle de sistemas acoplados.

2. Fundamentação Teórica

Este tópico abrangerá uma variedade de subtemas que exploram desde os mecanismos de cadeia fechada até a arquitetura de controle, cada um contribuindo para uma visão abrangente do funcionamento e da modelagem do sistema proposto por esse trabalho.

Iniciaremos com os mecanismos de cadeia fechada, onde discutiremos os graus de liberdade e a cinemática via cadeia vetorial, fundamentais para entender como os componentes interagem dentro de um sistema. A seguir, abordaremos a dinâmica via fluxo de potência, que fornece insights sobre a transferência de energia e a eficiência operacional em sistemas elétricos.

Em continuidade, as curvas de desempenho de motores DC serão analisadas, destacando suas características e aplicações práticas. O estudo da dinâmica dos manipuladores paralelos, das plataformas Stewart e dos atuadores elétricos serão essenciais para a modelagem matemática dos diversos sistemas.

Além disso, a modelagem acoplada será explorada, enfatizando a importância da integração entre diferentes modelos para uma análise precisa na forma de Grafos de Ligação. O modelo de inércia das pernas será apresentado como um aspecto crítico na dinâmica dos manipuladores, seguido pela discussão sobre a arquitetura de controle, que inclui o desenvolvimento de modelos de espaço de estado e o controle da trajetória, essenciais para garantir o funcionamento eficiente e preciso dos sistemas.

2.1. Mecanismos de cadeia fechada

Os mecanismos de cadeia fechada são sistemas mecânicos compostos por múltiplos corpos rígidos interligados, formando um ciclo contínuo que permite movimento relativo entre os componentes. Esses mecanismos são

essenciais em diversas aplicações, como robótica, onde a precisão e a estabilidade do movimento são cruciais. Eles podem ser encontrados em plataformas de simulação, manipuladores industriais e sistemas de posicionamento.

- **Cadeias Cinéticas:** Os mecanismos de cadeia fechada são formados por elos conectados por juntas, criando uma estrutura que pode ser analisada como uma cadeia cinemática.
- **Restrições:** Ao contrário das cadeias abertas, as cadeias fechadas têm restrições em ambas as extremidades, o que limita a liberdade de movimento.
- **Aplicações:** São utilizados em simuladores de voo, braços robóticos e sistemas de controle de movimento.

2.1.1 Graus de liberdade

Os graus de liberdade (do inglês “*Degree of Freedom*”, ou DoF) referem-se ao número de movimentos independentes que um mecanismo pode realizar. O cálculo dos DoF é essencial para entender a complexidade do mecanismo e sua capacidade de movimento.

O critério de Grubler (Albuquerque, A. N., 2017) para calcular os DoF é dada por:

$$F = \Lambda(n - j - 1) + \sum_{i=1}^j f_i - f_p \quad (2.1.1)$$

onde:

- F é o número de graus de liberdade do sistema
- f_i é o número de graus de movimento associados à junta i

- f_p é o número de graus de liberdade passivos do sistema
- j é o número total de juntas do sistema
- j_f é o número de juntas com f graus de liberdade
- n é o número de elos no sistema
- Λ é o número de graus de liberdade do espaço de atuação do sistema (3 para movimentação planar e 6 para espaciais)

Essa formulação é essencial para o entendimento e posterior modelagem do sistema.

2.1.2. Cinemática via cadeia vetorial

A cinemática via cadeia vetorial é uma abordagem gráfica utilizada para descrever o movimento dos mecanismos. Essa técnica permite calcular as posições, velocidades e acelerações dos elos em um mecanismo fechado.

Para um mecanismo com n elos, a posição do i -ésimo elo pode ser expressa como:

$$p_i = p_{i-1} + R_i(\theta_i) \cdot l_i \quad (2.1.2)$$

onde:

- p_i é a posição do elo i
- $R_i(\theta_i)$ é a matriz de rotação associada ao ângulo θ_i
- l_i é o vetor que representa o comprimento do elo.

As velocidades podem ser obtidas através da diferenciação da equação acima em relação ao tempo:

$$\dot{p}_i = \dot{p}_{i-1} + \frac{dR_i(\theta_i)}{dt} + R_i(\theta_i) \cdot \dot{\theta}_i \quad (2.1.3)$$

A matriz Jacobiana J_i , que relaciona as velocidades articulares às velocidades lineares dos elos, pode ser expressa como:

$$\dot{p}_i = J_i \cdot \dot{q} \quad (2.1.4)$$

Onde $\dot{q}$ representa as velocidades articulares e J_i é definida como:

$$J_i = \begin{bmatrix} \frac{\partial p_{1x}}{\partial q_1} & \dots & \frac{\partial p_{1j}}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial p_{1j}}{\partial q_1} & \dots & \frac{\partial p_{ij}}{\partial q_n} \end{bmatrix} \quad (2.1.5)$$

Essa abordagem permite calcular as velocidades angulares e lineares dos elos em função das taxas de variação das variáveis articulares.

2.2. Dinâmica via fluxo de potência

Segundo Bera, T.K. (2011), a dinâmica via fluxo de potência se concentra em como a energia é transferida entre os componentes de um mecanismo durante seu funcionamento. Essa abordagem é fundamental para entender as interações entre forças e movimentos dentro do sistema, permitindo uma análise mais profunda do desempenho e da eficiência do mecanismo.

Neste contexto, a potência é uma medida crucial que relaciona a força aplicada a um objeto e a velocidade com que essa força é aplicada. Em

sistemas mecânicos, a análise do fluxo de potência ajuda a identificar como a energia é distribuída entre diferentes partes do mecanismo, incluindo atuadores e componentes móveis. Isso é especialmente importante em aplicações onde o controle preciso do movimento e da energia é necessário, como em robótica e máquinas industriais.

A dinâmica dos mecanismos pode ser estudada por meio da identificação das relações de equilíbrio entre forças e torques, considerando fatores como inércia, rigidez e amortecimento. Essa análise permite projetar sistemas que não apenas funcionem de maneira eficiente, mas também respondam corretamente as variações nas condições operacionais.

2.3. Manipuladores Paralelos

Os manipuladores paralelos são sistemas robóticos compostos por múltiplos braços mecânicos conectados em paralelo, oferecendo altíssimo desempenho em termos de velocidade, precisão e rigidez. Eles são amplamente utilizados em uma vasta gama de aplicações que demandam movimentos de alta precisão, como simulação de voo, cirurgias robóticas minimamente invasivas e manufatura de componentes complexos em indústrias de diversos setores. Este trabalho abordará os princípios de funcionamento desses sistemas, explorando sua modelagem matemática.

2.3.1. Plataformas Stewart

As plataformas Stewart, também conhecidas como plataformas paralelas, são dispositivos mecânicos que se baseiam em seis pistões hidráulicos ou atuadores lineares para controlar precisamente a posição e orientação de uma plataforma móvel. Essas plataformas são amplamente utilizadas em várias indústrias para uma variedade de aplicações especializadas.

Uma das áreas onde as plataformas Stewart se destacam é na simulação de movimento. Elas são comumente implementadas como parte de simuladores de voo e simuladores de veículos, proporcionando aos pilotos e condutores uma experiência realista e imersiva. Com a ajuda dos atuadores lineares, essas plataformas são capazes de recriar movimentos complexos, como inclinação, rotação e translação, possibilitando aos usuários uma sensação autêntica de voar ou dirigir.

Além disso, as plataformas Stewart também são amplamente empregadas em sistemas de testes de vibração. Com a capacidade de gerar movimentos altamente precisos, esses dispositivos são usados para simular e reproduzir vibrações em laboratórios, permitindo que os engenheiros realizem testes rigorosos em diferentes produtos e componentes. Isso é particularmente útil em indústrias como automotiva, aeroespacial e eletrônica, onde a resistência e a durabilidade são importantes considerações.

Além das áreas mencionadas acima, as plataformas Stewart também encontram aplicações significativas no campo da realidade virtual. Combinadas com sistemas de visualização e tecnologias de rastreamento de movimento, essas plataformas podem proporcionar aos usuários uma experiência imersiva e envolvente em ambientes virtuais. Ao simular de forma realista a sensação de movimento, as plataformas Stewart contribuem para uma experiência de realidade virtual mais autêntica e emocionante.

Nesse trabalho, foi importante entender a dinâmica desse sistema complexo de 6 graus de liberdade (Damic, V., 2014). Com o objetivo de modelar esse sistema, é imperativo que a sua cinemática inversa seja estabelecida para que seja possível o seu controle. Para chegar no resultado desejado, Albuquerque, A. N. (2012), em sua tese, primeiramente obteve a geometria inversa descrita na Figura 2.5.1 e Figura 2.5.2:

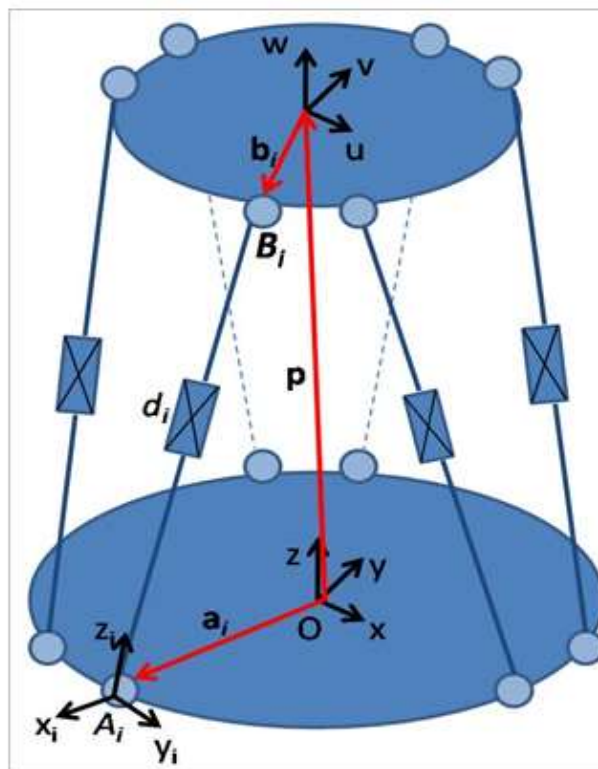


Figura 2.5.1 Diagrama esquemático da Plataforma Stewart.

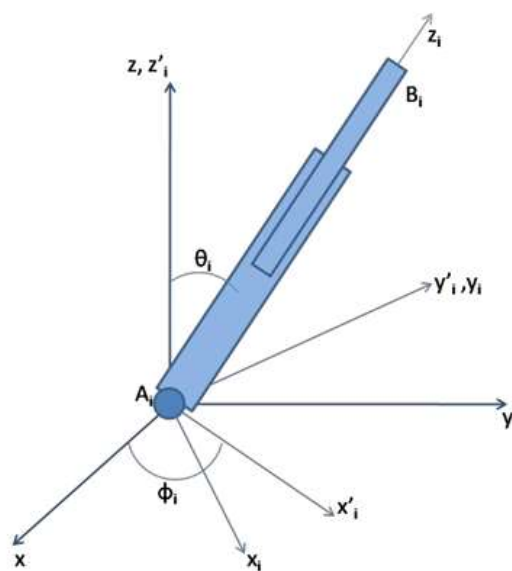


Figura 2.5.2 Ângulos e posições de uma junta universal.

Após a conclusão de seus cálculos, foi possível obter a cinemática inversa via cadeia vetorial (seção 2.1.2.) que descreve a Jacobiana inversa do sistema tal que:

$$J^{-1} = \begin{bmatrix} c\phi_1 s\theta_1 & s\phi_1 s\theta_1 & c\theta_1 & b_{1v}c\theta_1 - b_{1w}s\phi_1 s\theta_1 & b_{1w}c\phi_1 s\theta_1 - b_{1u}c\theta_1 & b_{1u}s\phi_1 s\theta_1 - b_{1v}c\phi_1 s\theta_1 \\ c\phi_2 s\theta_2 & s\phi_2 s\theta_2 & c\theta_2 & b_{2v}c\theta_2 - b_{2w}s\phi_2 s\theta_2 & b_{2w}c\phi_2 s\theta_2 - b_{2u}c\theta_2 & b_{2u}s\phi_2 s\theta_2 - b_{2v}c\phi_2 s\theta_2 \\ c\phi_3 s\theta_3 & s\phi_3 s\theta_3 & c\theta_3 & b_{3v}c\theta_3 - b_{3w}s\phi_3 s\theta_3 & b_{3w}c\phi_3 s\theta_3 - b_{3u}c\theta_3 & b_{3u}s\phi_3 s\theta_3 - b_{3v}c\phi_3 s\theta_3 \\ c\phi_4 s\theta_4 & s\phi_4 s\theta_4 & c\theta_4 & b_{4v}c\theta_4 - b_{4w}s\phi_4 s\theta_4 & b_{4w}c\phi_4 s\theta_4 - b_{4u}c\theta_4 & b_{4u}s\phi_4 s\theta_4 - b_{4v}c\phi_4 s\theta_4 \\ c\phi_5 s\theta_5 & s\phi_5 s\theta_5 & c\theta_5 & b_{5v}c\theta_5 - b_{5w}s\phi_5 s\theta_5 & b_{5w}c\phi_5 s\theta_5 - b_{5u}c\theta_5 & b_{5u}s\phi_5 s\theta_5 - b_{5v}c\phi_5 s\theta_5 \\ c\phi_6 s\theta_6 & s\phi_6 s\theta_6 & c\theta_6 & b_{6v}c\theta_6 - b_{6w}s\phi_6 s\theta_6 & b_{6w}c\phi_6 s\theta_6 - b_{6u}c\theta_6 & b_{6u}s\phi_6 s\theta_6 - b_{6v}c\phi_6 s\theta_6 \end{bmatrix} \quad (2.5.1)$$

Esse resulta é de suma importância para o controle e posterior acoplamento da Plataforma Stewart com os atuadores elétricos.

2.4. Atuadores Elétricos

Os atuadores elétricos são componentes essenciais em sistemas de controle e automação, responsáveis por converter energia elétrica em movimento mecânico. Na tese de Albuquerque, A. N.(2017), os atuadores elétricos, especialmente os lineares, desempenham um papel crucial na dinâmica e controle de mecanismos paralelos.

Os atuadores elétricos funcionam através da conversão de energia elétrica em energia mecânica. Eles podem ser classificados em diferentes tipos, incluindo motores elétricos (como os de corrente contínua e alternada) e atuadores lineares. Os atuadores lineares, em particular, são projetados para converter movimento rotativo em movimento linear, utilizando mecanismos como roscas ou engrenagens. Essa capacidade de gerar movimento controlado é amplamente utilizada em aplicações que requerem precisão, como robótica e sistemas automatizados.

A dinâmica dos atuadores elétricos é descrita por equações que consideram as forças e torques envolvidos no movimento. A relação entre a

força aplicada pelo atuador e sua velocidade é fundamental para entender o desempenho do sistema. Para um sistema com múltiplos atuadores, as relações de equilíbrio são formuladas considerando todos os fluxos de potência, permitindo uma análise detalhada do comportamento do sistema sob diferentes condições operacionais. Essas relações estão representadas pela

Figura 2.6.1., que representa o atuador elétrico escolhido para esse trabalho (LINAK, modelo LA12).

Além disso, a modelagem dinâmica dos atuadores envolve a consideração de fatores como inércia, rigidez e amortecimento. Esses fatores influenciam diretamente o desempenho do atuador e sua capacidade de resposta a comandos de controle. A análise dessas características é crucial para otimizar o funcionamento dos sistemas mecânicos que utilizam atuadores elétricos.

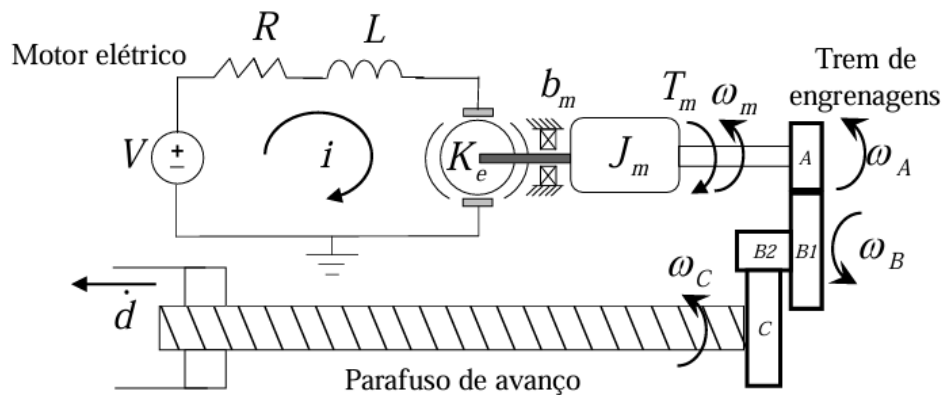


Figura 2.6.1 Esquema do atuador elétrico.

2.4.1. Curvas de desempenho de motores DC

As curvas de desempenho dos motores DC são essenciais para compreender como esses motores operam sob diferentes condições. Essas curvas representam graficamente as relações entre torque, velocidade e

eficiência, fornecendo informações valiosas para a seleção e o dimensionamento adequados dos motores em diversas aplicações.

A curva que relaciona torque e velocidade mostra como o torque gerado pelo motor varia com a velocidade de rotação. Geralmente, à medida que a velocidade aumenta, o torque tende a diminuir devido à força contra eletromotriz que se desenvolve no motor. Essa relação é crítica para aplicações onde o motor deve operar em diferentes faixas de carga.

Além disso, as curvas de eficiência revelam como a eficiência do motor muda em função da potência mecânica produzida. A eficiência é um indicador importante da performance do motor, pois reflete quanta energia elétrica é convertida em trabalho mecânico útil. Compreender essas curvas permite otimizar o uso dos motores em sistemas mecânicos, garantindo que operem dentro das especificações desejadas para maximizar o desempenho e minimizar o consumo energético.

Essas análises são fundamentais para auxiliar na criação de um modelo matemático que consiga representar corretamente todos os parâmetros do sistema.

2.5. Modelagem Acoplada com o uso de Grafos de Ligação

A modelagem acoplada de uma Plataforma Stewart com manipulador paralelo de 6 graus de liberdade acionado por atuadores elétricos envolve a integração cuidadosa e minuciosa dos modelos matemáticos individuais desses componentes em um sistema unificado e coerente. Isso requer uma compreensão profunda e abrangente das interações dinâmicas e cinemáticas entre os diversos subsistemas, bem como a consideração detalhada de parâmetros essenciais, tais como inércia e atrito.

Para realizar esse estudo, é preciso abordar a técnica utilizada para o acoplamento e definição de cada sistema, os Grafos de Ligação:

Os Grafos de Ligação (*Bond Graphs*) são uma abordagem eficaz para modelar sistemas dinâmicos, permitindo a interconexão de diferentes domínios físicos, como mecânico, elétrico e hidráulico. Essa técnica é baseada na representação da energia como uma "moeda comum" entre os diversos domínios, facilitando a análise e o design de sistemas complexos.

Os Grafos de Ligação são compostos por elementos que representam diferentes características dos sistemas. Cada elemento possui uma descrição específica e uma tendência de causalidade, que indica a direção do fluxo de energia. Considerando as variáveis de potência representadas pelo esforço (e) e fluxo (f), temos que:

- **Inertância (I):**

- **Descrição:** Armazena energia na forma de energia cinética. Sendo I, a indutância, sua equação característica tem a forma de:

$$f = \frac{e}{I} \quad (2.7.1)$$

- **Causalidade:** Quando a seta causal está próxima ao elemento, indica que o fluxo é imposto pelo sistema; quando está distante, o momento é imposto pelo elemento.

- **Capacitância (C):**

- **Descrição:** Armazena energia na forma de energia potencial. Sendo C a capacitância, sua equação característica tem a forma de:

$$e = \frac{f}{c} \quad (2.7.2)$$

- **Causalidade:** A seta causal distante do elemento indica que o deslocamento é um estado independente do sistema.
- **Resistência (R):**
 - **Descrição:** Dissipa energia convertendo-a em formas não recuperáveis. Sendo R a resistência, sua equação característica tem a forma de:

$$e = Rf \quad (2.7.3)$$

- **Causalidade:** A colocação da seta causal não afeta a estrutura das equações, mas influencia a dinâmica do sistema.
- **Fonte de Esforço:**
 - **Descrição:** Impõe um esforço ao sistema, como tensão elétrica ou força.
 - **Causalidade:** A seta causal deve estar distante do elemento, indicando que o esforço é imposto ao sistema.
- **Fonte de Fluxo:**
 - **Descrição:** Impõe um fluxo ao sistema, como corrente elétrica ou velocidade.
 - **Causalidade:** A seta causal deve estar próxima ao elemento, indicando que o esforço é imposto pelo sistema.
- **Transformador (TF):**
 - **Descrição:** Relaciona esforços e fluxos entre diferentes domínios sem perda de energia. Sendo N_e a razão de transferência, as relações são definidas como :

$$TF = \begin{cases} f_2 = N_e f_1 \\ e_1 = N_e e_1 \end{cases} \quad (2.7.4)$$

- **Causalidade:** Uma seta causal está no elemento e a outra está fora dele.
- **Girador (GY):**
 - **Descrição:** Relaciona um esforço a um fluxo e vice-versa. Sendo K_e a constante do girador, temos as seguintes relações:

$$GY = \begin{cases} e_1 = K_e f_2 \\ e_2 = K_e f_1 \end{cases} \quad (2.7.4)$$

- **Causalidade:** Ambas as setas podem estar próximas ou distantes do elemento.
- **Junta Comum de Esforço (0-junction):**
 - **Descrição:** Mantém o mesmo esforço em todos os laços conectados; a soma dos fluxos que entram deve ser zero.
 - **Causalidade:** Uma seta causal está na junta; as outras devem estar fora.
- **Junta Comum de Fluxo (1-junction):**
 - **Descrição:** Mantém o mesmo fluxo em todos os laços; a soma dos esforços que entram deve ser zero.
 - **Causalidade:** Uma seta causal deve estar fora da junta; as outras devem estar dentro.

Vale observar que ao montar um grafo de ligação, deve-se respeitar os fluxos e causalidades das junções, o que pode resultar em causalidades forçadas. Essas invertem a relação de esforço e fluxo dos elementos afetados.

A Figura 2.7.1 representa um resumo dos elementos dos Grafos de Ligação (Filho G. L.T., 2002)

Elementos		Relação Constitutiva	Símbolo Convencional	Sistema Elétrico	Sistema Hidráulico
Variáveis de Potência	Esforço	e	e	Tensão, U [V]	Pressão, p [Pa]
	Fluxo	f	f	Corrente elétrica, i [A]	Vazão, Q [m ³ /s]
Variáveis Integrais	Momentum	$P_p = \int e dt$	P_p	Fluxo elétrico, λ [V.s]	Quantidade Movimento de pressão, P_p [Pa.s]
	Deslocamento	$q = \int f dt$	q	Carga elétrica, q [C]	Volume, V [m ³]
Inertância	Indutância de energia ao sistema	$f(t) = \frac{1}{L} \int_0^t e(t).dt$	$\frac{e}{f} \rightarrow L$	Indutância Elétrica [H]	Indutância Fluida [Pa.s ² /m ³]
Capacitância	Capacidade de acumular energia	$e(t) = \frac{1}{C} \int_0^t f(t).dt$	$\frac{e}{f} \rightarrow C$	Capacitância Elétrica [F]	Capacitância fluida [m ³ /Pa]
Resistência	Elemento dissipador de energia	$e(t) = f(t).R$	$\frac{e}{f} \rightarrow R$	Resistência elétrica [Ω]	Resistência fluida [Pa.s/m ³]
Fonte de Esforço	Supre o sistema		$\frac{E(t)}{Se} \rightarrow$	Fonte de tensão	Fonte de pressão
Fonte de Fluxo	Supre o sistema		$Sf \rightarrow (Ft)$	Fonte de corrente	Fonte de vazão
Transformador	Ampliam ou reduzem a amplitude de uma entrada	$e_1 = m e_2$ $m f_1 = f_2$	$\frac{m}{T} \frac{e}{f} \rightarrow$	Transformador de potência	Pistão hidráulico
Girador	Relaciona esforço de entrada e fluxo de saída e vice versa	$e_1 = \Omega f_2$ $\Omega f_1 = e_2$	$\frac{\Omega}{Gy} \rightarrow$	Gerador Motor	Turbina Bomba
Elementos de Junções	Junção " 0 "	$e_1 = e_2 = e_3$ $f_1 - f_2 - f_3 = 0$	$\frac{1}{3} \rightarrow$	Impedância em paralelo	Derivações em sistemas hidráulicos
	Junção " 1 "	$f_1 = f_2 = f_3$ $e_1 - e_2 - e_3 = 0$	$\frac{1}{3} \rightarrow$	Impedância em série	Perda de carga serie

Figura 2.7.1 Resumo dos elementos de Grafos de Ligação.

O acoplamento entre dois Grafos de Ligação ocorre através da interconexão das juntas e elementos. As regras para garantir um acoplamento válido incluem:

- **Conexão Direta:** Elementos como transformadores podem ser conectados diretamente, formando um novo transformador equivalente.
- **Ordem Importante:** A sequência dos elementos importa, especialmente ao conectar giradores e transformadores.
- **Simplificações:** É possível simplificar grafos complexos removendo juntas sem efeito ou combinando elementos diretamente conectados.

Dessa forma, torna-se evidente que após a compreensão da dinâmica do sistema a ser estudada, ele pode ser representado por um grafo de ligação e subsequentemente acoplado em outros sistemas, o que é imperativo para a progressão desse trabalho.

2.6. Modelo de Inércia das pernas

O modelo de inércia das pernas é considerado no modelo da planta, mas não no controlador, com o objetivo de avaliar a eficácia do controlador em condições de parâmetros incertos. A simulação feita por BERA T.L (2011) demonstra um desempenho preciso no rastreamento da trajetória, mesmo sem considerar a inércia das pernas. No entanto, ao incluir a inércia, observa-se um aumento significativo nas forças necessárias para os atuadores, evidenciando que a desconsideração da inércia pode levar a diferenças consideráveis nas forças calculadas, especialmente em altas velocidades. Entretanto, foi observado que para velocidades baixas, os efeitos da inércia são insignificantes. Dessa forma, para o escopo desse trabalho, usamos apenas baixas velocidades como meios de simplificação do modelo matemático e redução de erros quanto a validação experimental.

2.7. Arquitetura de Controle

Para o escopo desse trabalho, foi implementado um controlador PID para o controle de posição do atuador.

O controle PID (Proporcional, Integral e Derivativo) é um método amplamente utilizado em sistemas de controle automático, que combina três ações de controle para ajustar a saída de um sistema e minimizar o erro entre a saída desejada (*setpoint*) e a saída real. Os três componentes principais do controle PID são: o termo proporcional, que gera uma saída proporcional ao erro atual; o termo integral, que acumula o erro ao longo do tempo para corrigir desvios persistentes; e o termo derivativo, que prevê o comportamento futuro do erro com base na sua taxa de variação.

O componente proporcional é responsável por fornecer uma resposta imediata ao erro, mas um ganho muito alto pode causar oscilações indesejadas. Já o termo integral é essencial para eliminar o erro estacionário, embora possa introduzir instabilidade se não for ajustado corretamente. Por sua vez, o componente derivativo ajuda a suavizar a resposta do sistema, proporcionando uma ação corretiva antecipada e reduzindo as oscilações.

O controle PID é amplamente aplicado em diversas áreas, como controle de temperatura, controle de velocidade em motores, sistemas de automação industrial e robótica. Sua simplicidade e facilidade de implementação, aliadas à flexibilidade para se adaptar a diferentes tipos de sistemas, tornam-no uma ferramenta valiosa para engenheiros e técnicos.

3. Metodologia

A abordagem metodológica adotada neste trabalho abrange diversas etapas cruciais. Inicialmente, realizamos uma análise intensiva para compreender profundamente os elementos do sistema. Posteriormente, avançamos para a fase de modelagem matemática, na qual formulamos equações extremamente complexas que representam com precisão o comportamento dinâmico do sistema. É essencial mencionar que essa etapa exige um cuidado meticuloso para considerar todos os fatores relevantes. Após a modelagem matemática, iniciamos as simulações numéricas, usando algoritmos para realizar simulações rigorosas e detalhadas. O objetivo é avaliar o desempenho teórico do modelo desenvolvido, garantindo a eficiência e confiabilidade do sistema em análise. Em seguida, passamos aos experimentos práticos, fundamentais para validar quantitativamente as previsões teóricas. Realizamos experimentos controlados com parâmetros definidos em um ambiente controlado, oferecendo insights valiosos sobre a precisão e aplicabilidade do modelo. Essa abordagem sistemática assegura uma compreensão abrangente do sistema estudado, combinando modelagem matemática, simulações numéricas e experimentos práticos para avaliação crítica dos resultados.

3.1. Modelagem Matemática

A modelagem matemática será fundamentada nas equações de dinâmica de corpos rígidos e nas leis de controle de atuadores elétricos, visando descrever detalhadamente as relações cinemáticas e dinâmicas do sistema acoplado. Utilizaremos técnicas de modelagem por Grafos de Ligação para representar as interações complexas entre os componentes da plataforma Stewart e do manipulador paralelo, considerando todas as restrições e acoplamentos. O objetivo é garantir um modelo matemático

robusto e eficiente, aplicável em diversas áreas desde a engenharia de controle até a robótica avançada. Essa abordagem permitirá uma completa compreensão do comportamento e desempenho do sistema, possibilitando a otimização dos algoritmos de controle e a identificação de melhorias. Assim, almejamos obter resultados mais precisos e confiáveis, promovendo avanços significativos no desenvolvimento de sistemas robóticos complexos e de alto desempenho.

Devido a restrições computacionais e dificuldade de medição de múltiplos parâmetros diante da complexidade do acoplamento de 6 atuadores elétricos, apenas o Modelo 1 (Atuador Elétrico) foi simulado e estudado para esse trabalho.

3.1.1 Modelo 1 - Atuador elétrico

Para maximizar a versatilidade proporcionada pelo método de Grafos de Ligação no acoplamento de diversos subsistemas, iniciamos o processo com a modelagem do atuador elétrico. O diagrama de Grafos de Ligação utilizado na análise deste atuador levou em consideração a influência de dois estímulos externos, representados pelas variáveis U (voltagem) e F (força). Essas variáveis são cruciais para o desenvolvimento da solução proposta, pois simbolizam fatores determinantes que impactam o comportamento do sistema.

Além disso, a identificação dos componentes de inércia foi realizada através da análise de elementos resistivos R_1 , bm_1 , bc_1 e bv_1 provenientes da resistividade elétrica e ao atrito viscoso e dos elementos inerciais L_1 , Jm_1 , Jc_1 , mA_1 originados da indutância do motor, momentos de inercia dos eixos do motor, do sistema de transmissão e da haste do atuador. A compreensão desses elementos é fundamental para a análise dinâmica do sistema, dado que a inércia está diretamente relacionada à resistência do corpo a alterações em seu estado de movimento. Por outro lado, os elementos de resistência foram caracterizados por meio dos resistores e das resistências

ao rolamento, atritos internos e resistividade elétrica, os quais desempenham um papel significativo na dissipação de energia e na eficiência do sistema.

Foi conduzida uma análise das inter-relações entre os diferentes "subsistemas" que compõem o atuador. Em particular, a interação entre as partes elétrica e mecânica do sistema, designada como girador (GY), merece destaque. Esta relação é essencial para garantir o funcionamento integrado do atuador. Adicionalmente, foram examinadas as conexões entre o eixo do motor e o trem de engrenagens, assim como entre o parafuso de avanço e a haste. Essas interconexões são representadas no diagrama como os transformadores (TF), evidenciando a complexidade e a interdependência dos componentes dentro do sistema. O resultado dessa abordagem pode ser observado na Figura 3.1.1.

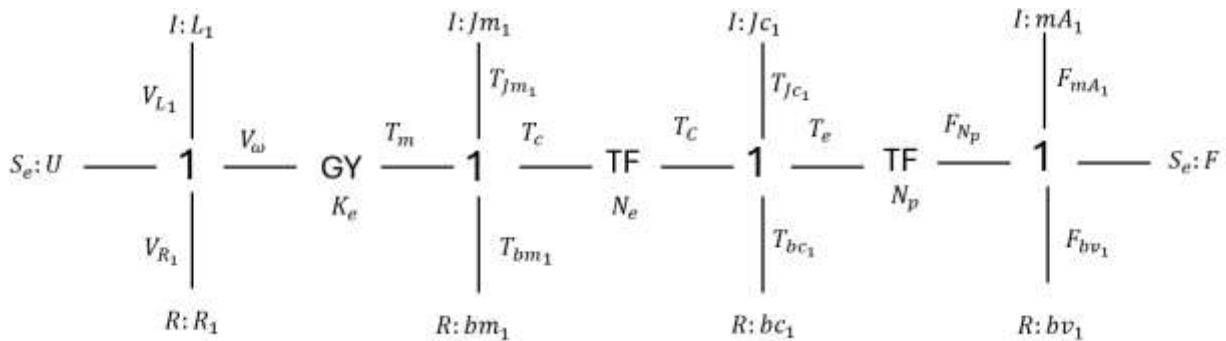


Figura 3.1.1 Modelo do atuador em Grafos de Ligação: sem fluxo de potência e barras causais.

Após a definição dos elementos e das conexões pertinentes, torna-se imprescindível configurar a causalidade do sistema. De acordo com as diretrizes delineadas na seção 2.7.2, é possível identificar duas relações de causalidade forçada, sendo elas: $I: J_{m1}$ e $I: J_{c1}$. Essa configuração é atribuída ao fato de que o sistema é regulado pelas inércias do motor e pela massa do atuador, que atuam como variáveis de estado.

A análise dessas causalidades revela a interdependência entre as variáveis, onde as inércias do motor influenciam diretamente o comportamento dinâmico do sistema, enquanto a massa do atuador desempenha um papel crucial na resposta do sistema às variações de entrada. Essa interação complexa entre as variáveis de estado é fundamental para a compreensão do funcionamento do sistema como um todo.

Além disso, a identificação precisa dessas causalidades permite uma modelagem mais eficaz e a implementação de estratégias de controle mais robustas, assegurando que o sistema opere dentro dos parâmetros desejados. Portanto, a configuração adequada da causalidade não apenas fundamenta a teoria subjacente ao modelo, mas também tem implicações práticas significativas para o desempenho e a estabilidade do sistema em questão. Aplicando a causalidade ao modelo representado na Figura 3.1.2, temos que:

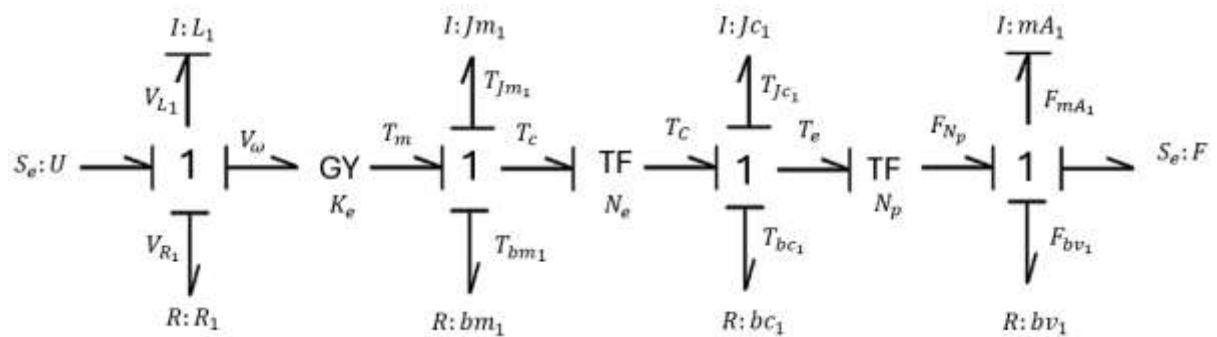


Figura 3.1.2 Modelo em Grafos de Ligação do atuador com causalidades.

Com a presença de todos os elementos necessários devidamente identificados no grafo, é possível avançar com a modelagem utilizando as equações fundamentais dos Grafos de Ligação, representado pela Figura 3.1.3.

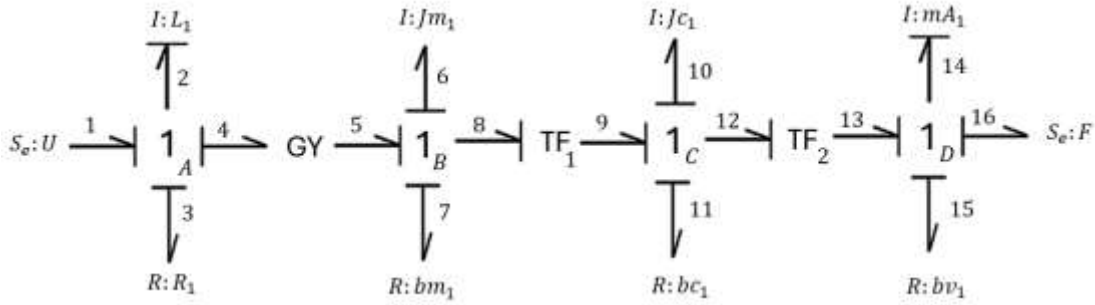


Figura 3.1.3 Modelo em Grafos de Ligação do atuador com números identificadores.

Através da representação descrita acima podemos inferir:

$$I(2): \begin{cases} \dot{p}_2 = e_2 \\ f_2 = \frac{p_2}{L} \end{cases} \quad (3.1.1)$$

$$I(6): \begin{cases} \dot{p}_6 = e_6 \\ p_6 = J_{m_1} f_6 \end{cases} \quad (3.1.2)$$

$$I(10): \begin{cases} \dot{p}_{10} = e_{10} \\ p_{10} = J_{c_1} f_{10} \end{cases} \quad (3.1.3)$$

$$I(14): \begin{cases} \dot{p}_{14} = e_{14} \\ f_{14} = \frac{p_{14}}{mA_1} \end{cases} \quad (3.1.4)$$

$$R(3): e_3 = R_1 f_3 \quad (3.1.5)$$

$$R(7): e_7 = bm_1 f_7 \quad (3.1.6)$$

$$R(11): e_{11} = bc_1f_{11} \quad (3.1.7)$$

$$R(15): e_{15} = bV_1f_{15} \quad (3.1.8)$$

$$GY = \begin{cases} e_4 = K_e f_5 \\ e_5 = K_e f_4 \end{cases} \quad (3.1.9)$$

$$TF_1 = \begin{cases} f_9 = N_e f_8 \\ e_8 = N_e e_9 \end{cases} \quad (3.1.10)$$

$$TF_2 = \begin{cases} f_{13} = N_p f_{12} \\ e_{12} = N_p e_{13} \end{cases} \quad (3.1.11)$$

$$1(A) = \begin{cases} f_1 = f_2 \\ f_3 = f_2 \\ f_4 = f_2 \\ e_2 = e_1 - e_3 - e_4 \end{cases} \quad (3.1.12)$$

$$1(B) = \begin{cases} f_5 = f_8 \\ f_6 = f_8 \\ f_7 = f_8 \\ e_8 = e_5 - e_6 - e_7 \end{cases} \quad (3.1.13)$$

$$1(C) = \begin{cases} f_9 = f_{12} \\ f_{10} = f_{12} \\ f_{11} = f_{12} \\ e_{12} = e_9 - e_{10} - e_{11} \end{cases} \quad (3.1.14)$$

$$1(D) = \begin{cases} f_{13} = f_{14} \\ f_{15} = f_{14} \\ f_{16} = f_{14} \\ e_{14} = e_{13} - e_{15} - e_{16} \end{cases} \quad (3.1.15)$$

$$S_e(1): e_1 = U \quad (3.1.16)$$

$$S_e(16): e_{16} = F \quad (3.1.17)$$

Com as equações montadas, começamos a construir o espaço de estado que descreve a cinemática do atuador, considerando p_2 e p_{14} como as variáveis de estado. Começando por $\dot{p}_2$:

$$\dot{p}_2 = e_2 = e_1 - e_3 - e_4 \quad (3.1.18)$$

O esforço e_4 está definido pela equação 3.1.9, que, ao substituirmos as equações 3.1.13, 3.1.10, 3.1.14, 3.1.11, 3.1.15 e finalmente 3.1.4, obtemos que:

$$e_4 = \frac{K_e}{N_e N_p m A_1} p_{14} \quad (3.1.19)$$

O esforço e_3 está definido pela equação 3.1.5, que, ao substituirmos as equações 3.1.12 e 3.1.1 temos que:

$$e_3 = \frac{R_1}{L} p_2 \quad (3.1.20)$$

Ao substituirmos as equações 3.1.16, 3.1.19 e 3.1.20 na equação 3.1.18, obtemos o valor final de $\dot{p}_2$:

$$e_3 = U - \frac{R_1}{L} p_2 - \frac{K_e}{N_e N_p m A_1} p_{14} \quad (3.1.21)$$

Para o cálculo de $\dot{p}_{14}$, analogamente ao que foi feito para $\dot{p}_2$, temos que:

$$\dot{p}_{14} = e_{14} = e_{13} - e_{16} - e_{15} \quad (3.1.22)$$

O esforço e_{15} está definido pela equação 3.1.8, que, ao substituirmos as equações 3.1.15 e 3.1.4 temos que:

$$e_{15} = \frac{bV_1}{mA_1} p_{14} \quad (3.1.23)$$

O esforço e_{13} está definido pela equação 3.1.11, que, ao substituirmos as equações 3.1.14 temos que:

$$e_{13} = \frac{1}{N_p} (e_9 - e_{10} - e_{11}) \quad (3.1.24)$$

O esforço e_{11} está definido pela equação 3.1.7, que, ao substituirmos as equações 3.1.14, 3.1.11 e 3.1.4 temos que:

$$e_{11} = \frac{bc_1}{N_p mA_1} p_{14} \quad (3.1.25)$$

O esforço e_{10} representa uma causalidade forçada, dessa forma definimos:

$$e_{10} = \dot{p}_{10} = Jc_1 \dot{f}_{10} \quad (3.1.26)$$

Ao substituírmos as equações 3.1.14 e 3.1.4 na equação 3.1.26, obtemos:

$$e_{10} = \frac{Jc_1}{N_p mA_1} \dot{p}_{14} \quad (3.1.27)$$

Analogamente ao esforço e_{13} , o esforço e_9 definido pela equação 3.1.10, que, ao substituírmos as equações 3.1.13 temos que:

$$e_{13} = \frac{1}{N_e} (e_5 - e_6 - e_7) \quad (3.1.28)$$

O esforço e_7 está definido pela equação 3.1.6, que, ao substituírmos as equações 3.1.13, 3.1.9 e 3.1.19 temos que:

$$e_7 = \frac{bm_1}{N_p N_p m A_1} p_{14} \quad (3.1.29)$$

O esforço e_6 representa uma causalidade forçada, dessa forma definimos:

$$e_6 = \dot{p}_6 = J c_1 \dot{f}_6 \quad (3.1.30)$$

Ao substituírmos as equações 3.1.13 , 3.1.9 e 3.1.19 na equação 3.1.30, obtemos:

$$e_6 = \frac{J m_1}{N_p m A_1} \dot{p}_{14} \quad (3.1.31)$$

O esforço e_5 está definido pela equação 3.1.9, que, ao substituírmos as equações 3.1.12e 3.1.1 temos que:

$$e_5 = \frac{K_e}{L} p_2 \quad (3.1.32)$$

Por fim, ao substituir as equações 3.1.23 - 3.1.32 e 3.1.17 na equação 3.1.22, obtemos:

$$\begin{aligned} \dot{p}_{14} = \frac{1}{N_p} \left(\frac{1}{N_e} \left(\frac{K_e}{L} p_2 - \frac{Jm_1}{N_p m A_1} \dot{p}_{14} - \frac{bm_1}{N_p N_p m A_1} p_{14} \right) \right. \\ \left. - \frac{Jc_1}{N_p m A_1} \dot{p}_{14} - \frac{bc_1}{N_p m A_1} p_{14} - F \right. \\ \left. - \frac{bV_1}{m A_1} p_{14} \right) \end{aligned} \quad (3.1.33)$$

Reorganizando em formato de uma variável de estado chegamos em:

$$\dot{p}_{14} = \frac{\frac{K_e}{N_e N_p L} p_2 - \left(\frac{bm_1}{N_e^2 N_p^2 m A_1} + \frac{bc_1}{N_p^2 m A_1} + \frac{bV_1}{m A_1} \right) p_{14} - F}{1 + \frac{Jm_1}{N_e^2 N_p^2 m A_1} + \frac{Jc_1}{N_p^2 m A_1}} \quad (3.1.34)$$

Considerando a definição de espaço de estado e as variáveis de estado:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \quad (3.1.35)$$

$$\dot{x} = \begin{bmatrix} \dot{p}_2 \\ \dot{p}_{14} \end{bmatrix} \quad (3.1.36)$$

$$u = \begin{bmatrix} U \\ F \end{bmatrix} \quad (3.1.36)$$

Temos que $A, B \in \mathbb{R}^2$ devido as 2 variáveis de estado entradas:

$$A = \begin{bmatrix} -\frac{R_1}{L} & -\frac{K_e}{N_e N_p m A_1} \\ \frac{K_e}{N_e N_p L} & \frac{b m_1}{N_e^2 N_p^2 m A_1} + \frac{b c_1}{N_p^2 m A_1} + \frac{b V_1}{m A_1} \\ \frac{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}}{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}} & \frac{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}}{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}} \end{bmatrix} \quad (3.1.37)$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -\frac{1}{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}} \end{bmatrix} \quad (3.1.38)$$

Para que validar nosso modelo, torna-se imprescindível a escolha de saídas que possam ser medidas experimentalmente. Dessa forma, foram escolhidas como saídas a força exercida pelo atuador ($\dot{p}_{14}$), a velocidade do atuador (f_{14}) e a corrente do motor (f_2). Ao utilizarmos as equações 3.1.37, 3.1.4 e 3.1.1, obtemos:

$$C = \begin{bmatrix} \frac{K_e}{N_e N_p L} & \frac{b m_1}{N_e^2 N_p^2 m A_1} + \frac{b c_1}{N_p^2 m A_1} + \frac{b V_1}{m A_1} \\ \frac{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}}{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}} & \frac{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}}{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}} \\ 0 & \frac{1}{m A_1} \\ \frac{1}{L} & 0 \end{bmatrix} \quad (3.1.39)$$

$$D = \begin{bmatrix} 0 & 1 \\ 0 & \frac{1}{1 + \frac{J m_1}{N_e^2 N_p^2 m A_1} + \frac{J c_1}{N_p^2 m A_1}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (3.1.40)$$

Para o controle de posição do atuador elétrico, foi construído um sistema de malha fechada composto por um controlador do tipo PID, o atuador e uma função $G(s)$ que transforma o sinal de saída de velocidade do atuador em posição Figura 3.1.4.

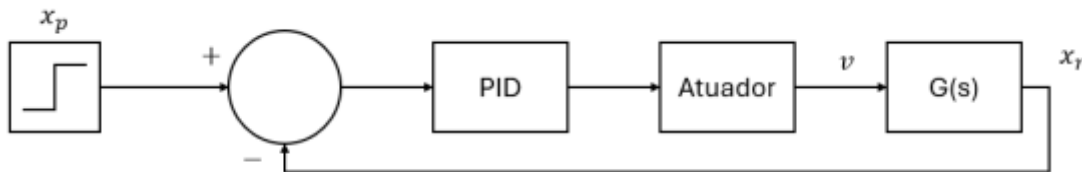


Figura 3.1.4 Diagrama de Blocos de controle do atuador elétrico.

3.1.2 Modelo 2 - Plataforma Stewart

Assim como abordado na seção 2.5, a cinemática inversa da plataforma Stewart pode ser descrita em forma de Grafos de Ligação, assim como apresentado parcialmente na Figura 3.1.5.

Neste modelo, algumas hipóteses simplificadoras foram assumidas:

- A contribuição do momento de inércia dos atuadores foi desprezada considerando o modelo de inércia das pernas descrita na seção 2.8.
- O modelo completo da plataforma Stewart tem 3 velocidades a mais, contabilizando os 6 graus de liberdade. Dessa forma $J^{-1} \in \mathbb{R}^6$, tendo 8 MTF em cada velocidade.

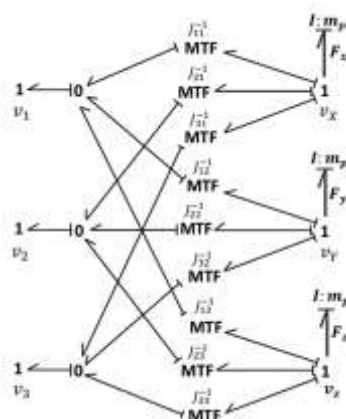


Figura 3.1.5- Cinemática inversa da plataforma plana via Grafos de Ligação.

3.1.3 Modelo 3 - Modelagem acoplada

Considerando as vantagens apresentadas na seção 2.7 sobre a utilização de grafos na montagem acoplada, foi possível desenvolver a Figura 3.1.5 seguindo as mesmas hipóteses apresentadas na seção 3.1.2. Os modelos do atuador e da plataforma foram acoplados pelas junções 1 responsáveis pela velocidade saída dos atuadores.

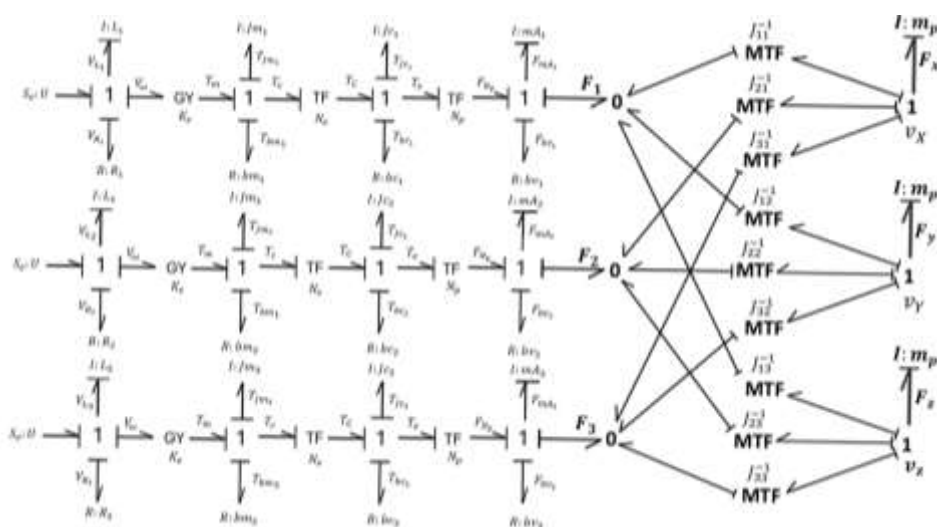


Figura 3.1.6 Montagem acoplada via Grafos de Ligação (parcial, mostrando apenas três atuadores).

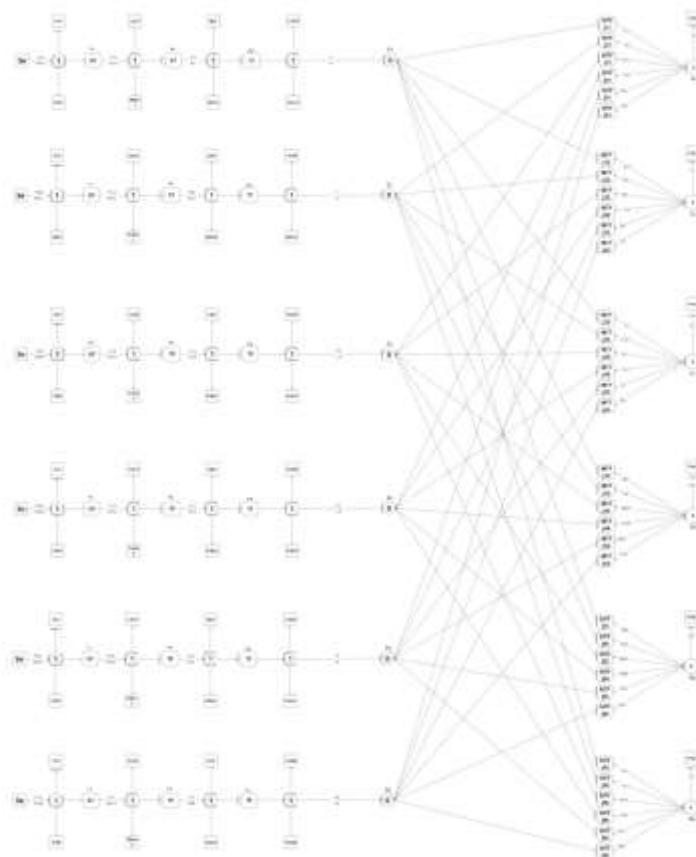


Figura 3.1.7 Montagem acoplada via Grafos de Ligação

Para futuros experimentos, é possível considerar a estrutura de controle sugerida por Albuquerque (2017).

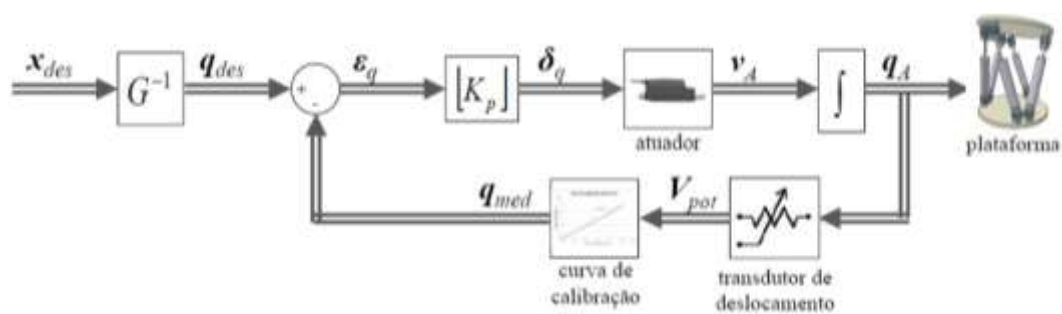


Figura 3.1.8 Estratégia de controle do atuador proposta.

3.2. Simulações Numéricas

As simulações numéricas foram cruciais para analisar o comportamento dinâmico do atuador elétrico. Utilizando modelos criados no MATLAB® e Simulink®, fomos capazes de configurar parâmetros não fornecidos pelo datasheet do atuador como as resistências internas ao aproximar nossos resultados com curvas fornecidas como a Figura 3.2.1.

Após a configuração do sistema, foi possível obter de forma precisa o comportamento do sistema diante de diferentes entradas de força e potencial elétrico utilizando o modelo desenvolvido na seção 3.1.1. Os parâmetros finais encontrados encontram-se na Tabela 3.2.1.

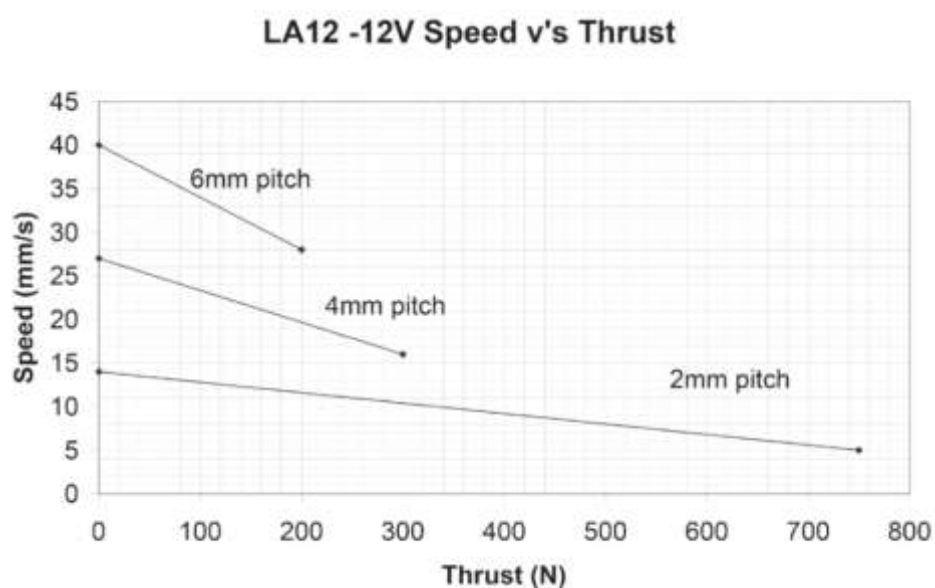


Figura 3.2.1 Curva de Força x Velocidade fornecida pelo fabricante

Substituindo esses valores nas matrizes obtidas na seção 3.1.1, obtivemos as seguintes matrizes:

$$A = \begin{bmatrix} -9.52 & -1.0574 \\ 0.0104 & -0.0031 \end{bmatrix} \quad (3.2.1)$$

$$B = \begin{bmatrix} 0.001 & -1.0574 \\ 0 & -0.003 \end{bmatrix} \quad (3.2.2)$$

$$C = \begin{bmatrix} 0.0104 & -0.0015 \\ 0 & 0.0057 \\ 1 & 0 \end{bmatrix} \quad (3.2.3)$$

$$D = \begin{bmatrix} 0 & -0.5629 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (3.2.4)$$

Parâmetro	Valor	Unidades
L	0.001	H
Jm_1	$1.00 * 10^{-6}$	kg/m^2
Jc_1	$5.62 * 10^{-5}$	kg/m^2
mA_1	$1.75 * 10^{-1}$	kg
R_1	9.52	Ω
bm_1	$8.80 * 10^{-5}$	$\frac{N.m.s}{rad}$
bc_1	$8.80 * 10^{-5}$	$\frac{N.m.s}{rad}$
bV_1	$8.80 * 10^{-5}$	$\frac{N.m.s}{rad}$
K_e	$2.51 * 10^{-2}$	$\frac{V.s}{rad}$
N_e	1	-
N_p	$1.3564 * 10^{-4}$	-

Tabela 3.2.1 Tabela com o valor dos parâmetros.

A partir dessas matrizes, construímos o sistema no Simulink Figura 3.2.2, onde simulamos as respostas para diferentes combinações de entradas (U e F) na forma de um degrau. A Figura 3.2.2 apresenta o diagrama de blocos deste modelo e as Figuras 3.2.3 e 3.2.4 apresentam as respostas no tempo obtidas.

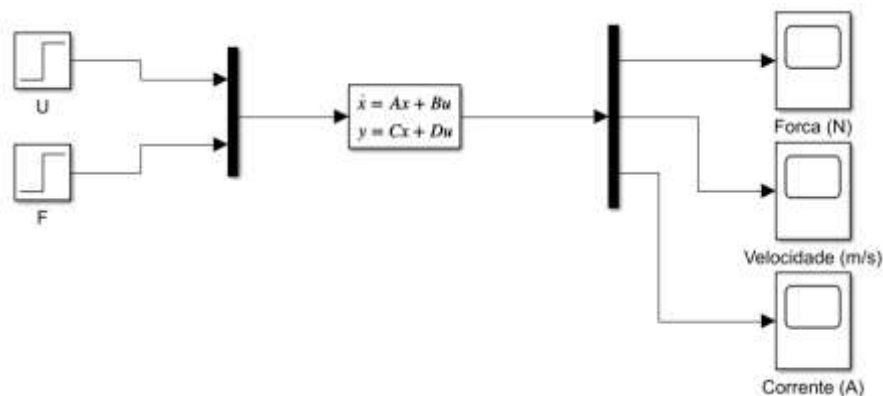


Figura 3.2.2 Modelo no Simulink.

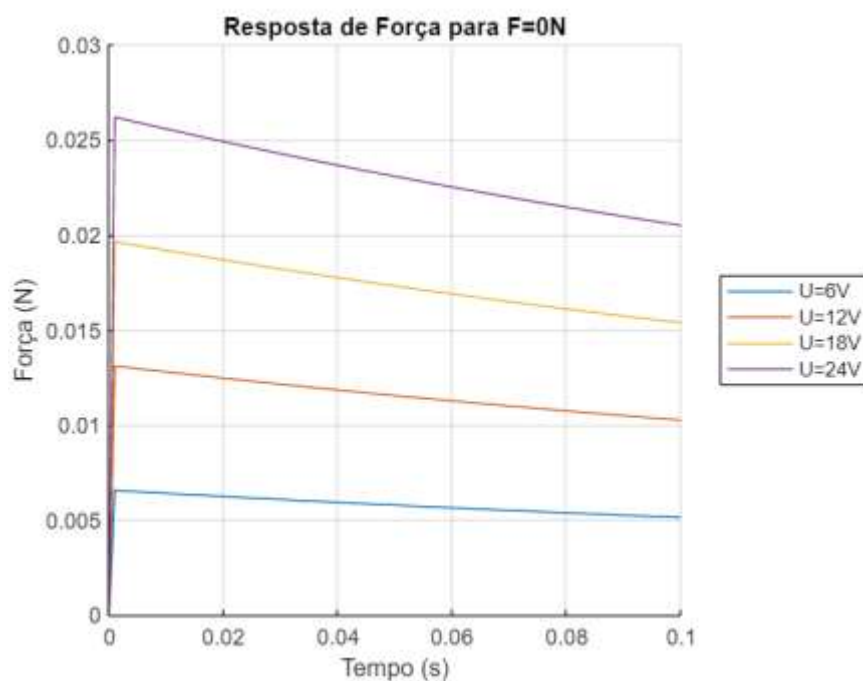


Figura 3.2.3 Resposta da força do atuador para diferentes tensões elétricas e força externa nula.

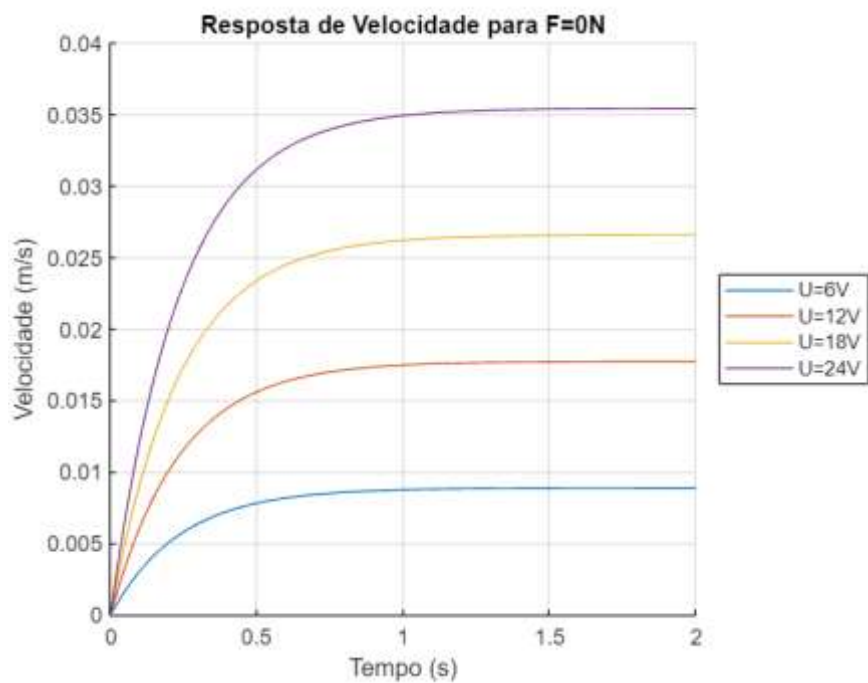


Figura 3.2.4 Resposta da velocidade do atuador para diferentes tensões elétricas e força externa nulas.

Assim como definido na seção 3.1.3, o controle do atuador foi simulado no Simulink como demonstra a Figura 3.2.5:

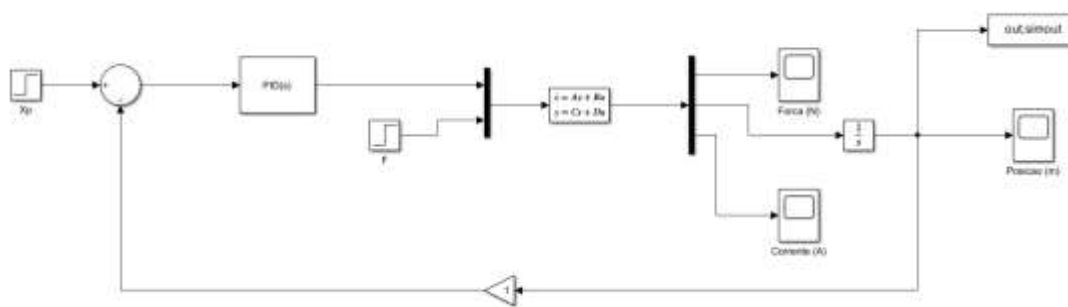


Figura 3.2.5 Esquema de controle no Simulink.

Para fins de simulação, a função $G(s)$ foi estimada como $\frac{1}{s}$ somente e os valores das constantes do controlador PID foram estimados através de testes.

3.3. Validação Experimental

A validação experimental foi conduzida através de testes físicos no atuador elétrico. Foram registradas as velocidades e correntes exercidos pelos atuadores para diferentes forças, permitindo a comparação entre os resultados experimentais e as previsões da modelagem matemática e das simulações numéricas. Isso verificou a precisão do modelo teórico e avaliou a eficácia do sistema proposto. Todos os experimentos contaram com um atuador elétrico LA 12 da LINAK (Figura 3.3.1).



Figura 3.3.1 Atuador elétrico LA 12.

3.3.1. Experimento 1

O primeiro experimento realizado teve como objetivo medir as velocidades do eixo do atuador e a corrente elétrica que passava no mesmo para diferentes tensões e força externa igual a zero. Para isso, foram medidos

3 vezes os tempos de extensão e retração do eixo do atuador. Ao mesmo tempo, foi também salvo o valor da corrente registrada pela fonte (Figuras 3.3.2 e 3.3.3). Os resultados obtidos encontram-se na Tabela 3.1.1.

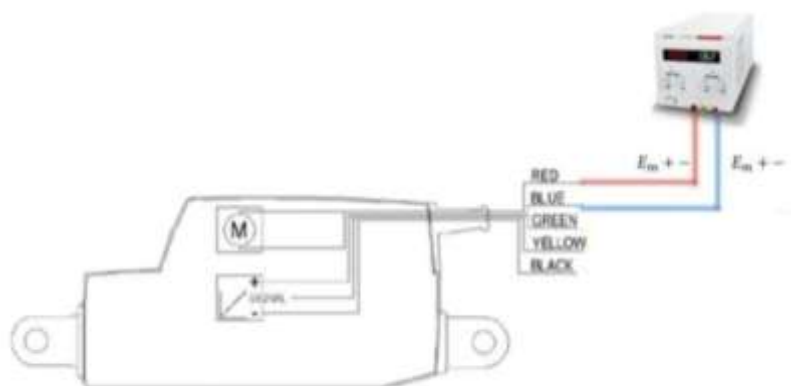


Figura 3.3.2 Esquema de ligação do atuador com a fonte.



Figura 3.3.3 - Visualização real do experimento.

	1ª Medição (s)	2ª Medição (s)	3ª Medição (s)	Corrente (A)
6V Extensão	10.5	10.6	10.7	0.19
6V Retração	11.3	11.5	12.4	0.23
12V Extensão	4.7	5	4.9	0.24
12V Retração	5.1	5	5	0.28
18V Extensão	3.1	3	3.2	0.26
18V Retração	3	3.18	3.19	0.32
24V Extensão	2.21	2.27	2.35	0.3
24V Retração	2.38	2.35	2.26	0.37

Tabela 3.3.1 - Tabela de medições do 1º experimento.

Para obter a velocidade, foi realizado a divisão entre o comprimento do eixo do atuador (100 mm) pela média aritmética das medições. Assim, foram obtidos os resultados apresentados na Tabela 3.3.2 para comparação com o modelo teórico.

	Velocidade média (mm/s)	Corrente (A)
6V Extensão	9.43	0.19
6V Retração	8.53	0.23
12V Extensão	20.54	0.24
12V Retração	19.87	0.28
18V Extensão	32.26	0.26
18V Retração	32.02	0.32
24V Extensão	43.92	0.3
24V Retração	42.92	0.37

Tabela 3.3.2 - Tabela de medições do 1º experimento.

A corrente e as velocidades no movimento de extensão e retração diferem devido à configuração interna do atuador, no qual ele apresenta maior resistência interna à movimento quando retrai.

3.3.2. Experimento 2

Para o segundo experimento, de forma análoga ao primeiro experimento, foi medido a velocidade de extensão do eixo do atuador. Entretanto, a medição da velocidade foi realizada através da leitura do potenciômetro do atuador por um Arduino Uno (Figura 3.3.4).

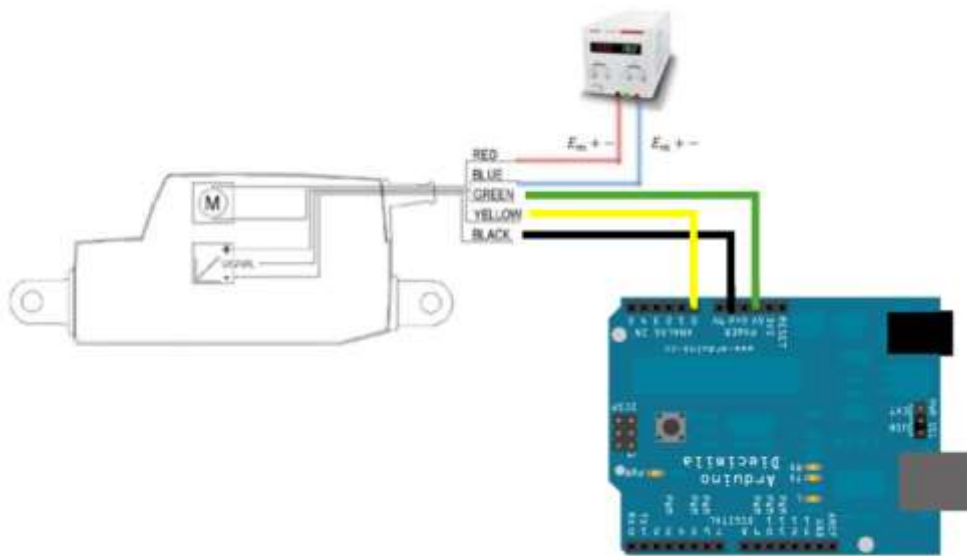


Figura 3.3.4 Esquema de ligação do atuador com a fonte e Arduino

Para diferentes tensões, o modulo Arduino registrou a posição do eixo do atuador através da leitura do potenciômetro. Após experimentos para 6V, 12V, 18V e 24V, a medição da posição ao longo do tempo foi salva. Devido a grande quantidade de ruído presente nas medições (Figura 3.3.5.), foi implementado um filtro passa-baixas de ordem $N = 9$ e frequência de corte

normalizada $\omega_n = 0.001$ como mostra a Figura 3.3.5. Posteriormente foi obtida a velocidade, derivando a posição e com ela foi feito uma aproximação do seu comportamento pela equação 3.3.1., que representa a resposta de um sistema criticamente amortecido para cada medição. As curvas ajustadas obtidas (Figura 3.3.6) e seus respectivos parâmetros para cada tensão estão representados na Tabela 3.3.3.

$$y(t) = K_p M \left(1 - \left(1 + \frac{t}{\tau_s} \right) e^{-\frac{t}{\tau_s}} \right) \quad (3.3.1)$$

Onde M representa a tensão em Volts e $K_p, \tau_s \in \mathbb{Z}$.

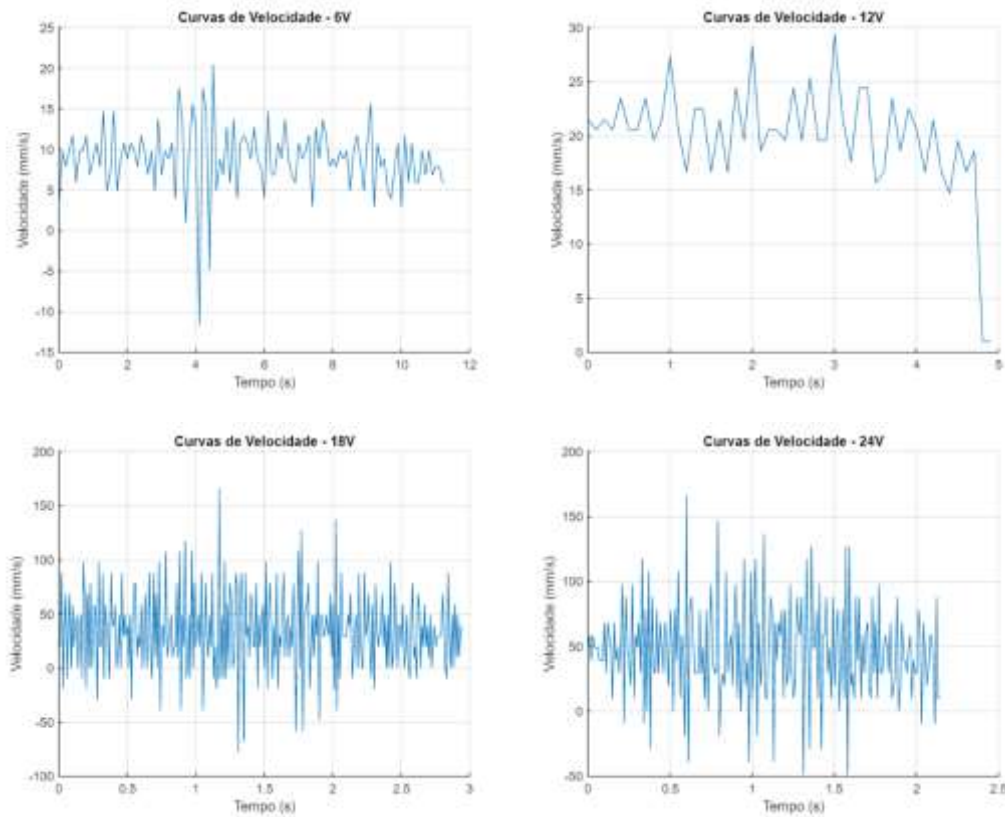


Figura 3.3.5 Velocidade do atuador sem filtro.

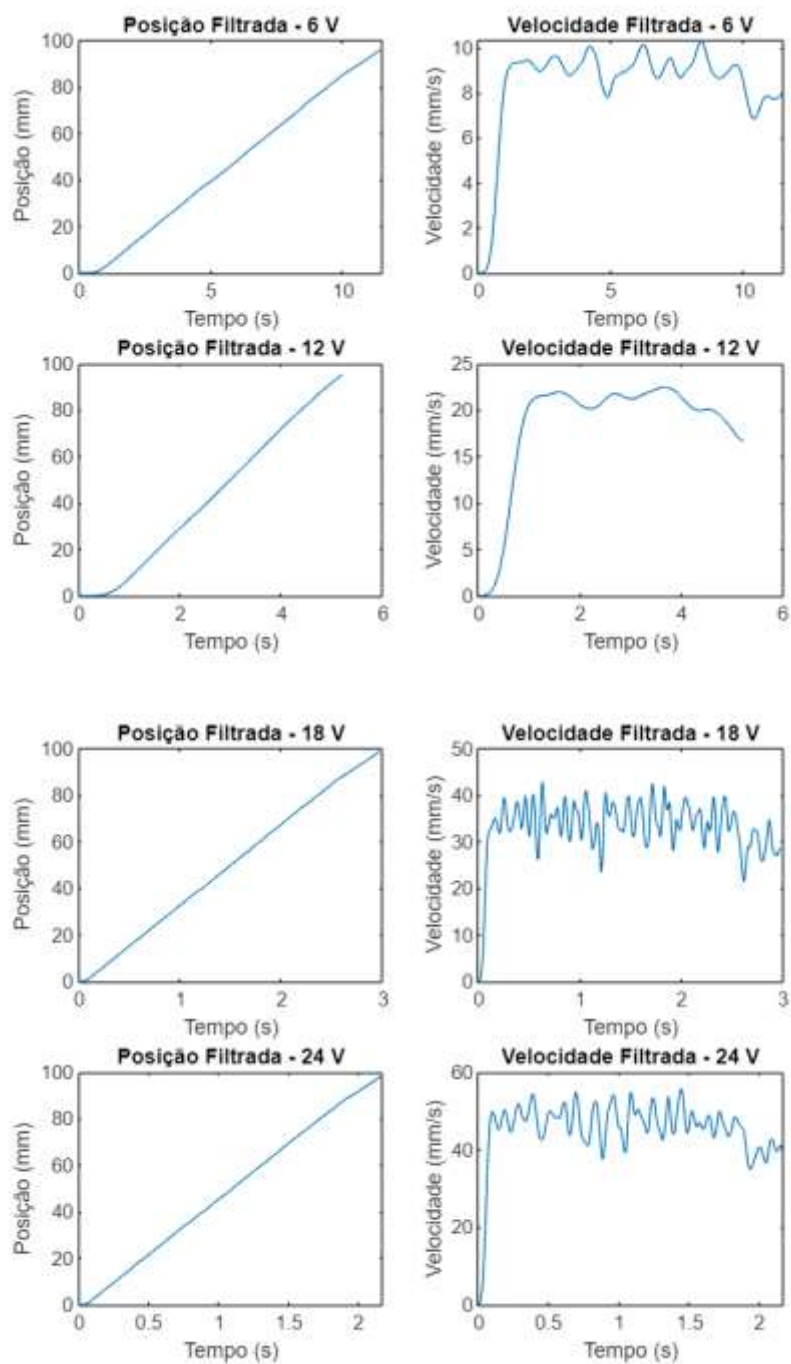


Figura 3.3.6 Posição e velocidade do atuador após filtro passa-baixos.

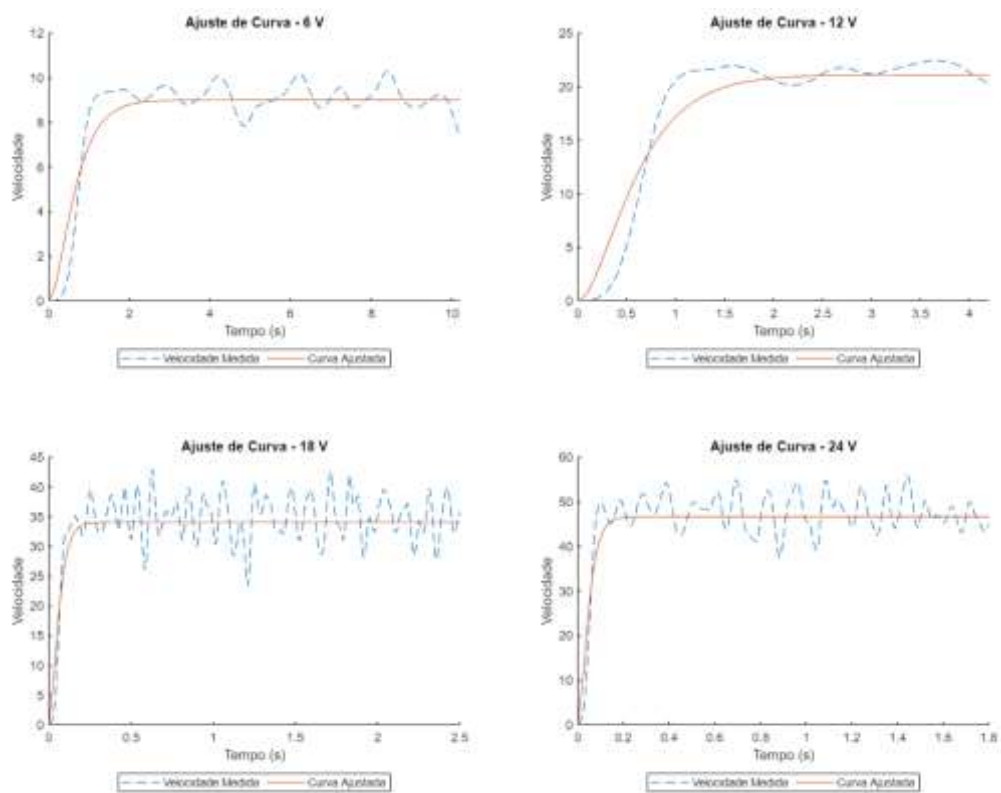


Figura 3.3.7 Comparação entre Curva ajustada e velocidade medida

Tensão\Parâmetro	K_p	τ_s
6V	1.4103	0.3528
12V	1.7049	0.3229
18V	1.8599	0.0323
24V	1.9007	0.0253

Tabela 3.3.3 - Tabela de parâmetros das curvas médias de cada tensão

4. Discussão de Resultados

Com os resultados obtidos no experimento 2, foi possível ser feita a comparação com o modelo desenvolvido na seção 3.2. (Figuras 4.1).

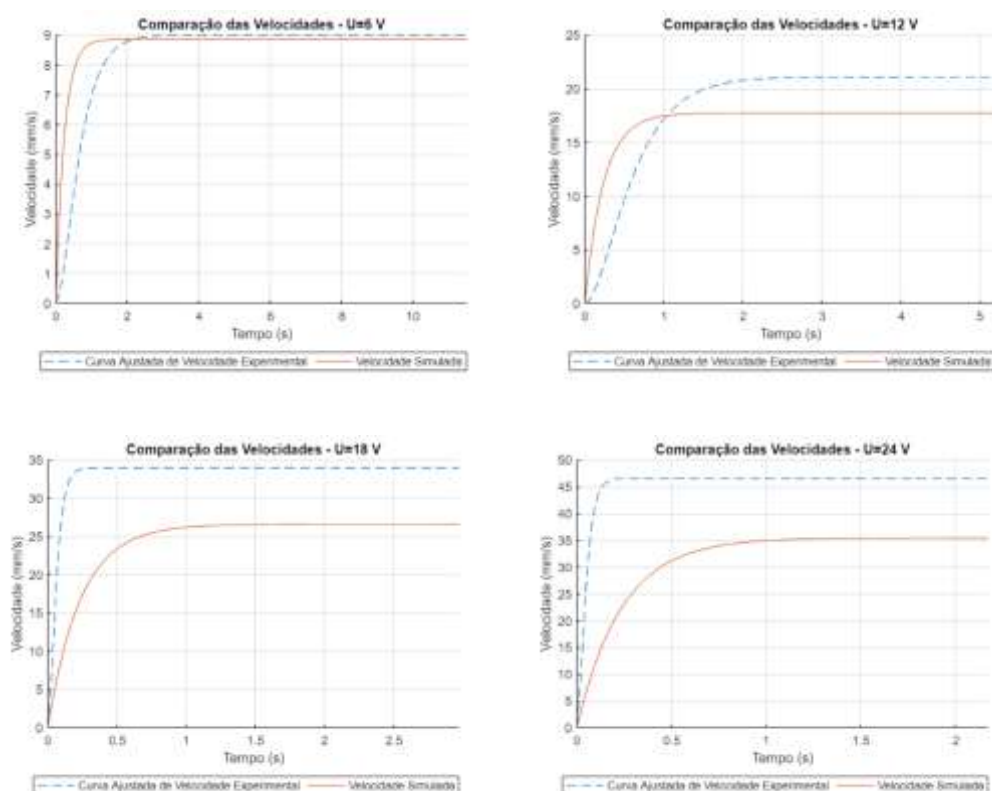


Figura 4.1 Comparação entre modelo desenvolvido e experimentos para diferentes tensões

Através dos gráficos apresentados, foi possível identificar a limitação do modelo matemático desenvolvido. O modelo prova-se fiel à realidade para a tensão de 6V (Figura 4.1), porém, gradativamente perde a precisão com o aumento da velocidade/tensão no atuador. Esse fenômeno pode ser explicado, principalmente, pelo ajuste dos parâmetros do sistema (Tabela 3.2.1).

Dessa forma, foram realizados ajustes utilizando a técnica de otimização por exame de partículas, presente no toolbox do MATLAB® para calibrar os principais parâmetros que poderiam sofrer alterações quanto seu valor, nominalmente os atritos, momentos de inercia e resistência elétrica do motor do atuador. Dessa forma, o novo modelo obteve uma precisão maior como demonstram as Figuras 4.2 e 4.3, com seus parâmetros representados na tabela 4.1.

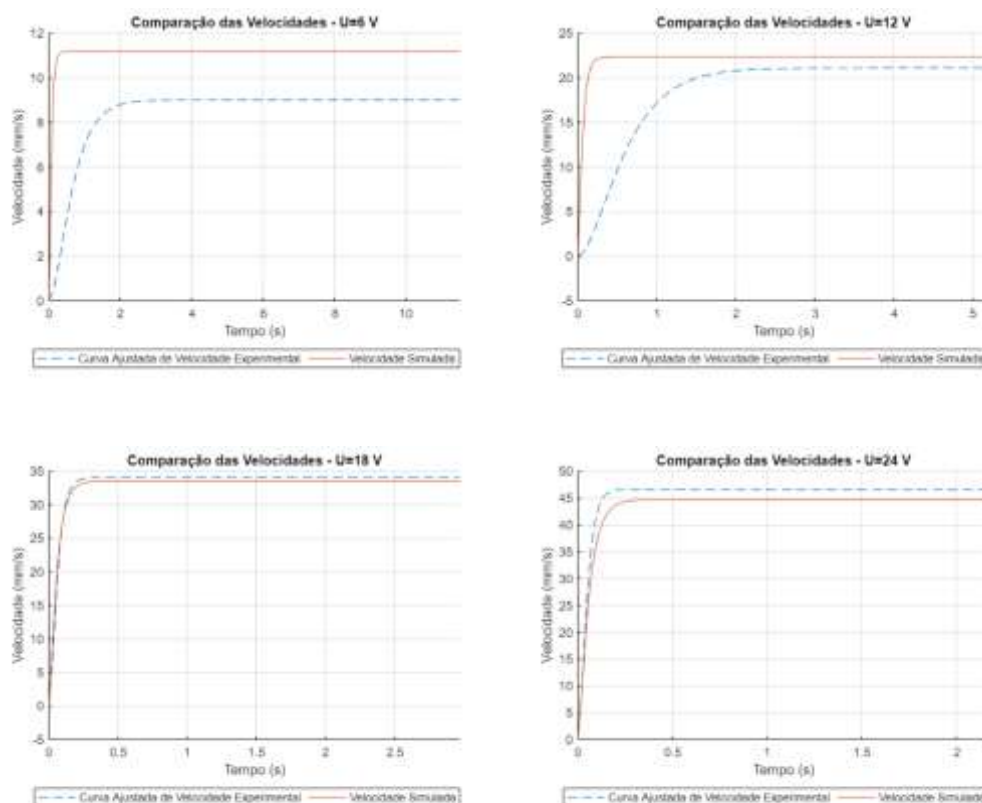


Figura 4.2 Comparação entre modelo calibrado e experimentos para diferentes tensões

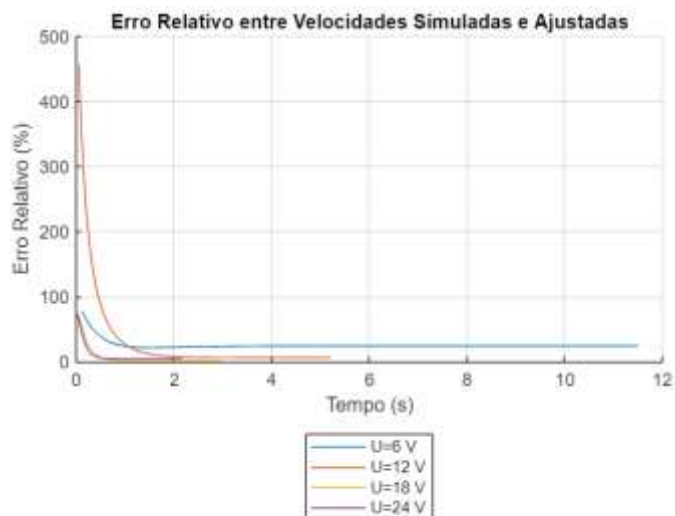


Figura 4.3 Erro relativo para cada tensão

Parâmetro	Valor antigo	Valor calibrado	Unidades
L	0.001	0.001	H
Jm_1	$1.00 \cdot 10^{-6}$	0.0001	kg/m^2
Jc_1	$5.62 \cdot 10^{-5}$	0.00094193	kg/m^2
mA_1	$1.75 \cdot 10^{-1}$	$1.75 \cdot 10^{-1}$	kg
R_1	9.52	0.0952	Ω
bm_1	$8.80 \cdot 10^{-5}$	0.0088	$\frac{N.m.s}{rad}$
bc_1	$8.80 \cdot 10^{-5}$	0.0037992	$\frac{N.m.s}{rad}$
bV_1	$8.80 \cdot 10^{-5}$	0.0059783	$\frac{N.m.s}{rad}$
K_e	$2.51 \cdot 10^{-2}$	$2.51 \cdot 10^{-2}$	$\frac{V.s}{rad}$
N_e	1	1	-
N_p	$1.3564 \cdot 10^{-4}$	$1.3564 \cdot 10^{-4}$	-

Tabela 4.1 Tabela com o valor dos parâmetros do modelo calibrado.

Pode-se observar que o modelo calibrado oferece boa precisão para o regime permanente, mas não tem o mesmo sucesso para o regime transiente. Isso se deve, principalmente, pela dificuldade de leitura e especificidades do regime transiente que são difíceis de serem estimadas como a transição dos atritos estáticos para o dinâmico e outras resistências estáticas.

Com o novo modelo, foi possível testar o controle do atuador (Figura 4.4) para estabilizar a posição do eixo do atuador em 50 mm (entrada X_p), foi adquirida a curva da resposta do sistema (Posição (m)), apresentada na Figura 4.5. Nesse modelo, a saída do controlador (Figura 4.6) é a tensão que alimenta o modelo. Os parâmetros utilizados para o controlador foram $k_p = 420$, $k_i = 0$, $k_d = 0$ e $N = 1000$.

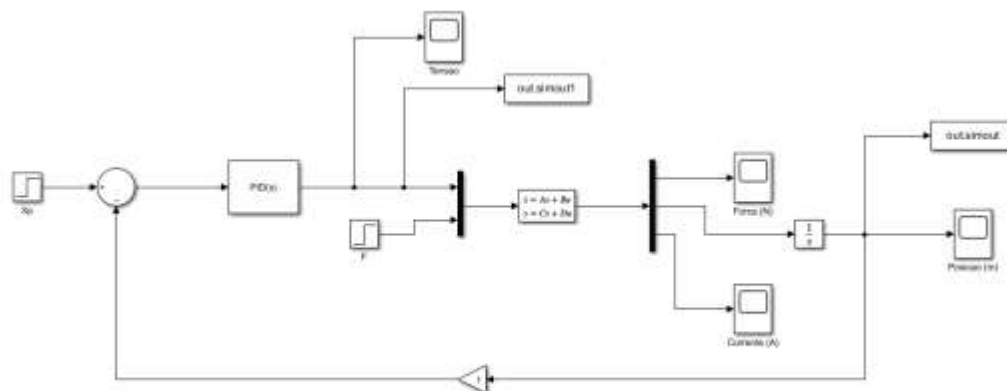


Figura 4.4 Esquema ajustado de controle no Simulink.



Figura 4.5 Resultado da simulação de malha fechada para controle de posição.

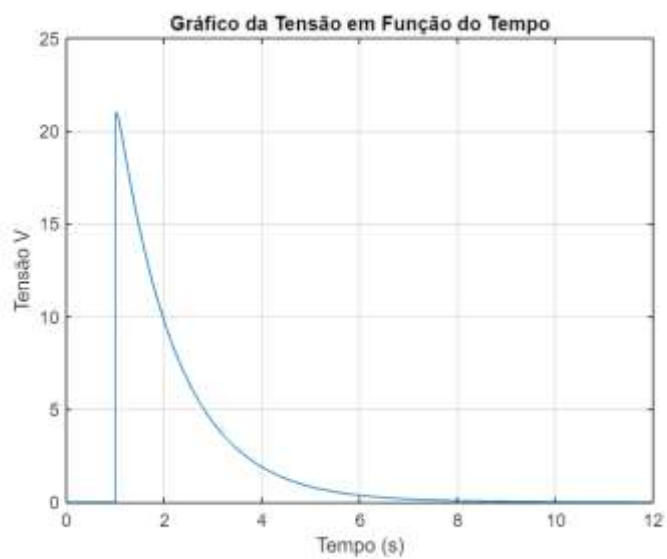


Figura 4.6 Resultado do sinal da tensão para a simulação de malha fechada.

5. Conclusões e Considerações Finais

Este trabalho teve como objetivo desenvolver a base da modelagem acoplada de uma Plataforma Stewart, um manipulador paralelo que possui seis graus de liberdade e é acionado por atuadores elétricos. A Plataforma Stewart é amplamente utilizada em diversas áreas, como simulação de voo e robótica, devido à sua capacidade de realizar movimentos complexos com alta precisão.

A metodologia proposta incluiu a realização de experimentos para a obtenção dos parâmetros do modelo, utilizando Grafos de Ligação (*Bond Graphs*) para estruturar a dinâmica do sistema. Essa abordagem permitiu uma visualização clara e organizada das equações e restrições envolvidas.

Esta pesquisa buscou contribuir para o entendimento da dinâmica da Plataforma Stewart, mas também para o desenvolvimento de técnicas de modelagem e controle que possam ser aplicadas em sistemas similares. Os resultados obtidos serviram para confirmar a verossimilhança do modelo físico com o sistema real.

Próximos trabalhos neste tema incluem a montagem dos atuadores elétricos na Plataforma Stewart existente no Laboratório de Desenvolvimento de Sistemas Mecatrônicos e a implementação dos diagramas de controle propostos no sistema real.

6. Referências Bibliográficas

DAMIC V., COHODAR M., **Dynamic Analysis of Stewart Platform by Bond Graphs**, DAAAM, 2014

YILDIZ I., OMURLU V.E., SAGIRLI A., **A Novel Visualization Technique in Bond Graph Method for Modeling of a Generalized Stewart Platform**, International Conference on Robotics and Biomimetics, Tailândia, 2009

BERA T.K., SAMANTARAY A.K., **Bond Graph model-based inversion of planar parallel manipulator systems**, International Journal of Modelling and Simulation, EUA, 2011

YILDIZ I., SAGIRLI A., **Dynamic Modeling of a Generalized Stewart Platform by Bond Graph Method Utilizing a Novel Spatial Visualization Technique**, International Review of Aerospace Engineering (IREASE), EUA, 2016

FILHO G. L.T., **O uso da técnica dos Grafos de Ligação na simulação de sistemas hidráulicos em regime transitório**, Mecânica Computacional Vol. XXI, Argentina, 2002

ALBUQUERQUE A. N., **Dinâmica e Controle de Mecanismos Paralelos: Integração Modelo Analítico Fechado, Transdutores Inerciais e Atuadores Elétricos Lineares**, Tese de Doutorado PUC-Rio, Brasil, 2017

ALBUQUERQUE A. N., **Modelagem e simulação de uma plataforma de Stewart controlada usando sensores inerciais**, Dissertação de Mestrado PUC-Rio, Brasil, 2012

7. Apêndice

7.1 Código MATLAB modelo simples e curva de velocidade experimental sem filtro

```

clc;
clear all;

% Definir variáveis simbólicas
syms U F P2 L dP6 P6 Jm1 dP10 P10 Jc1 P14 mA1 R1 bm1 bc1 bV1 Ke Ne Np

%Matriz A
A_11 = - R1/L;
A_12 = - Ke/(Ne*Np*mA1);
A_21 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * Ke/(Ne*Np*L);
A_22 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * ( -
1)*(bc1/(Np^2*mA1)+ bm1/(Ne^2*Np^2*mA1) + bV1/mA1);

A = [A_11 A_12; A_21 A_22];

%Matriz B
B_11 = 1;
B_12 = 0;
B_21 = 0;
B_22 = -1*(1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1);

B = [B_11 B_12; B_21 B_22];

% Y_1 = forca
% Y_2 = velocidade
% Y_3 = corrente

%Matriz C
C_11 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * Ke/(Ne*Np*L);
C_12 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * ( -
bc1/(Np^2*mA1) + bV1/mA1);
C_21 = 0;
C_22 = 1/mA1;

```



```

C_31 = 1/L;
C_32 = 0;

C = [C_11 C_12; C_21 C_22; C_31 C_32];

%Matriz D
D_11 = 0;
D_12 = -1*(1 + Jm1/((Ne*Np)^(2*mA1) + Jc1/(Np^2*mA1))^( -1));
D_21 = 0;
D_22 = 0;
D_31 = 0;
D_32 = 0;

D = [D_11 D_12; D_21 D_22; D_31 D_32];

% Substituição por valores reais
L_value = 1.00*10^-3;
Jm1_value = 1.00*10^-6;
Jc1_value = 5.62*10^-5;
mA1_value = 1.75*10^-1; %real
R1_value = 9.52;
bm1_value = 8.80*10^-5;
bc1_value = 8.80*10^-5;
bV1_value = 8.80*10^-5;
Ke_value = 2.51*10^-2;
Ne_value = 1;
Np_value = 1.3564*10^-4;

% Substituir parâmetros nas matrizes
params = [L, Jm1, Jc1, mA1, R1, bm1, bc1, bV1, Ke, Ne, Np];
values = [L_value, Jm1_value, Jc1_value, mA1_value, R1_value,
bm1_value, bc1_value, bV1_value, Ke_value, Ne_value, Np_value];

A = double(subs(A, params, values))
B = double(subs(B, params, values))
C = double(subs(C, params, values))
D = double(subs(D, params, values))

```

```

% Experimento realizado com  $F = 0$ 
% Tabela_Experimento Linhas (Extensao e Retracao):
% 1-2 6V
% 3-4 12V
% 5-6 18V
% 7-8 24V

% Tabela_Experimento Colunas:
% 1 teste (segundos)
% 2 teste (segundos)
% 3 teste (segundos)
% 4 corrente (A)

L_atuador = 100; % mm
% Tabela_Experimento = [10.5 10.6 10.7 0.19; 11.3 11.5 12.4 0.23; 4.7 5
4.9 0.24; 5.1 5 5 0.28; 3.1 3 3.2 0.26; ...
%      3 3.18 3.19 0.32; 2.21 2.27 2.35 0.3; 2.38 2.35 2.26 0.37]

% Definindo a matriz original
Tabela_Experimento = [10.5 10.6 10.7 0.19;
                      11.3 11.5 12.4 0.23;
                      4.7 5 4.9 0.24;
                      5.1 5 5 0.28;
                      3.1 3 3.2 0.26;
                      3 3.18 3.19 0.32;
                      2.21 2.27 2.35 0.3;
                      2.38 2.35 2.26 0.37];

% Calculando a média aritmética das três primeiras colunas
media_tempo = mean(Tabela_Experimento(:, 1:3), 2);

% Calculando a nova matriz
nova_matriz = [L_atuador ./ media_tempo, Tabela_Experimento(:, 4)];

% Definindo as legendas para as linhas
legendas = {'6V extensão (mm/s)', '6V retração (mm/s)', '12V extensão
(mm/s)', '12V retração (mm/s)', ...
            '18V extensão (mm/s)', '18V retração (mm/s)', '24V extensão
(mm/s)', '24V retração (mm/s)'};

```

```
% Exibindo os resultados
disp('Resultados:');
for i = 1:size(nova_matriz, 1)
    fprintf('%s: %.4f, %.2f\n', legendas{i}, nova_matriz(i, 1),
nova_matriz(i, 2));
end
```

```
% Importar o arquivo Excel
filename = 'experimento 2 Tcc.xlsx'; % Nome do arquivo
data = readtable(filename); % Ler os dados a partir da planilha

data = data{3:end, :};

% Normalizar os dados para o intervalo de 0 mm a 100 mm
data_normalized = (data - min(data(:))) / (max(data(:)) - min(data(:)))
* 100;

% Calcular o tempo em segundos
time_100ms = (0:size(data_normalized, 1)-1) * 0.1; % Tempo para as 4
primeiras colunas (100 ms)
time_10ms = (0:size(data_normalized, 1)-1) * 0.01; % Tempo para as 4
últimas colunas (10 ms)

velocities_first_four = diff(data_normalized(:, 1:4)) ./
diff(time_100ms'); % Velocidade nas primeiras 4 colunas

velocities_last_four = diff(data_normalized(:, end-3:end)) ./
diff(time_10ms'); % Velocidade nas últimas 4 colunas
```

```

% Plotar velocidades das colunas 1 e 2 (6V)
figure;
hold on;
plot(time_100ms(1:end-1), velocities_first_four(:, 2), 'DisplayName',
'Velocidade 2');
title('Curvas de Velocidade - 6V');
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');
grid on;

% Plotar velocidades das colunas 3 e 4 (12V)
figure;
hold on;
plot(time_100ms(1:end-1), velocities_first_four(:, 4), 'DisplayName',
'Velocidade 4');
title('Curvas de Velocidade - 12V');
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');
grid on;

% Plotar velocidades das colunas 5 e 6 (18V)
figure;
hold on;
plot(time_10ms(1:end-1), velocities_last_four(:, 2), 'DisplayName',
'Velocidade 6');
title('Curvas de Velocidade - 18V');
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');

grid on;

% Plotar velocidades das colunas 7 e 8 (24V)
figure;
hold on;
plot(time_10ms(1:end-1), velocities_last_four(:, 4), 'DisplayName',
'Velocidade 8');
title('Curvas de Velocidade - 24V');
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');
grid on;

```

7.2 Código MATLAB sinal filtrado

```

clc;
clear all;

% Importar o arquivo Excel
filename = 'experimento 2 Tcc.xlsx';
data = readtable(filename);
data = data{1:end, :}; % Remover as primeiras duas linhas (se necessário)

% Separar os dados em matrizes para cada tensão
matrix_6V = data(:, 1:2); % Colunas 1 e 2 para 6V
matrix_12V = data(:, 3:4); % Colunas 3 e 4 para 12V
matrix_18V = data(:, 5:6); % Colunas 5 e 6 para 18V
matrix_24V = data(:, 7:8); % Colunas 7 e 8 para 24V

% Remover valores NaN de cada matriz
matrix_6V = rmmissing(matrix_6V);
matrix_12V = rmmissing(matrix_12V);
matrix_18V = rmmissing(matrix_18V);
matrix_24V = rmmissing(matrix_24V);

% Normalizar cada matriz individualmente
normalizeMatrix = @(x) (x - min(x(:))) / (max(x(:)) - min(x(:))) * 100;
matrix_6V_normalized = normalizeMatrix(matrix_6V);
matrix_12V_normalized = normalizeMatrix(matrix_12V);
matrix_18V_normalized = normalizeMatrix(matrix_18V);
matrix_24V_normalized = normalizeMatrix(matrix_24V);

% Definir o tempo para cada matriz
time_6V = (0:0.1:(size(matrix_6V_normalized, 1)-1)*0.1)';
time_12V = (0:0.1:(size(matrix_12V_normalized, 1)-1)*0.1)';
time_18V = (0:0.01:(size(matrix_18V_normalized, 1)-1)*0.01)';
time_24V = (0:0.01:(size(matrix_24V_normalized, 1)-1)*0.01)';

% Definir parâmetros do filtro FIR
N = 9; % Ordem do filtro
Wn = 0.001; % Frequência de corte normalizada (0 < Wn < 1)

```

```

% Projetar o filtro FIR passa-baixa
b_filtro = fir1(N, Wn);

% Aplicar o filtro nos dados de posição normalizados
filtered_position_6V = filter(b_filtro, 1, matrix_6V_normalized(:,2));
filtered_position_12V = filter(b_filtro, 1,
matrix_12V_normalized(:,2));
filtered_position_18V = filter(b_filtro, 1,
matrix_18V_normalized(:,2));
filtered_position_24V = filter(b_filtro, 1,
matrix_24V_normalized(:,2));

% Calcular a velocidade usando gradient nos dados filtrados
velocity_filtered_6V = gradient(filtered_position_6V, time_6V);
velocity_filtered_12V = gradient(filtered_position_12V, time_12V);
velocity_filtered_18V = gradient(filtered_position_18V, time_18V);
velocity_filtered_24V = gradient(filtered_position_24V, time_24V);

% Plotar os resultados em subplots separados para cada tensão
figure;

% Subplot para 6 V
subplot(2,2,1);
plot(time_6V, filtered_position_6V);
title('Posição Filtrada - 6 V');
xlabel('Tempo (s)');
ylabel('Posição (mm)');

subplot(2,2,2);
plot(time_6V, velocity_filtered_6V);
title('Velocidade Filtrada - 6 V');
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');

% Subplot para 12 V
subplot(2,2,3);
plot(time_12V, filtered_position_12V);
title('Posição Filtrada - 12 V');
xlabel('Tempo (s)');
ylabel('Posição (mm)');

subplot(2,2,4);

```

```

plot(time_12V, velocity_filtered_12V);
title('Velocidade Filtrada - 12 V');
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');

% Criar uma nova figura para os outros dois conjuntos de dados
figure;

% Subplot para 18 V
subplot(2,2,1);
plot(time_18V, filtered_position_18V);
title('Posição Filtrada - 18 V');
xlabel('Tempo (s)');
ylabel('Posição (mm)');

subplot(2,2,2);
plot(time_18V, velocity_filtered_18V);
title('Velocidade Filtrada - 18 V');
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');

% Subplot para 24 V
subplot(2,2,3);
plot(time_24V, filtered_position_24V);
title('Posição Filtrada - 24 V');
xlabel('Tempo (s)');
ylabel('Posição (mm)');

subplot(2,2,4);
plot(time_24V, velocity_filtered_24V);
title('Velocidade Filtrada - 24 V');
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');

```

7.3 Código MATLAB ajuste de curva

```

clc;
clear all;

% Importar o arquivo Excel
filename = 'experimento 2 Tcc.xlsx';
data = readtable(filename);

```

```

data = data{1:end, :}; % Remover as primeiras duas linhas (se
necessário)

% Separar os dados em matrizes para cada tensão
matrix_6V = data(:, 1:2); % Colunas 1 e 2 para 6V
matrix_12V = data(:, 3:4); % Colunas 3 e 4 para 12V
matrix_18V = data(:, 5:6); % Colunas 5 e 6 para 18V
matrix_24V = data(:, 7:8); % Colunas 7 e 8 para 24V

% Remover valores NaN de cada matriz
matrix_6V = rmmissing(matrix_6V);
matrix_12V = rmmissing(matrix_12V);
matrix_18V = rmmissing(matrix_18V);
matrix_24V = rmmissing(matrix_24V);

% Normalizar cada matriz individualmente
normalizeMatrix = @(x) (x - min(x(:))) / (max(x(:)) - min(x(:))) * 100;
matrix_6V_normalized = normalizeMatrix(matrix_6V);
matrix_12V_normalized = normalizeMatrix(matrix_12V);
matrix_18V_normalized = normalizeMatrix(matrix_18V);
matrix_24V_normalized = normalizeMatrix(matrix_24V);

% Definir o tempo para cada matriz
time_6V = (0:0.1:(size(matrix_6V_normalized, 1)-1)*0.1)';
time_12V = (0:0.1:(size(matrix_12V_normalized, 1)-1)*0.1)';
time_18V = (0:0.01:(size(matrix_18V_normalized, 1)-1)*0.01)';
time_24V = (0:0.01:(size(matrix_24V_normalized, 1)-1)*0.01)';

% Definir parâmetros do filtro FIR
N = 9; % Ordem do filtro
Wn = 0.001; % Frequência de corte normalizada

% Projetar o filtro FIR passa-baixa
b_filtro = fir1(N, Wn);

% Aplicar o filtro nos dados de posição normalizados
filtered_position_6V = filter(b_filtro, 1, matrix_6V_normalized(:,2));
filtered_position_12V = filter(b_filtro, 1,
matrix_12V_normalized(:,2));
filtered_position_18V = filter(b_filtro, 1,
matrix_18V_normalized(:,2));

```



```

filtered_position_24V = filter(b_filtro, 1,
matrix_24V_normalized(:,2));

% Calcular a velocidade usando gradient nos dados filtrados
velocity_filtered_6V = gradient(filtered_position_6V, time_6V);
velocity_filtered_12V = gradient(filtered_position_12V, time_12V);
velocity_filtered_18V = gradient(filtered_position_18V, time_18V);
velocity_filtered_24V = gradient(filtered_position_24V, time_24V);

% Função do modelo para ajuste de curva
modelFun = @(params, t) params(1) * params(2) * (1 - (1 + t /
params(3)) .* exp(-t / params(3)));

% Parâmetros iniciais [Kp, M, tau]
initialParams = [1, NaN, 1]; % Kp=1 e tau=1; M será definido no loop

% Valores de tensão e velocidades filtradas correspondentes
M_values = [6, 12, 18, 24];
filtered_velocities = {velocity_filtered_6V, velocity_filtered_12V,
velocity_filtered_18V, velocity_filtered_24V};
times_for_fit = {time_6V, time_12V, time_18V, time_24V};
limite_x = [10.2, 4.2, 2.5, 1.8];

% Criar uma figura para cada tensão
for i = 1:length(M_values)
    M_value = M_values(i);
    filtered_velocity = filtered_velocities{i};
    time_for_fit = times_for_fit{i};

    % Ajuste de curva usando lsqcurvefit
    initialParams(2) = M_value; % Defina M no vetor de parâmetros
    iniciais
    [params_fit, resnorm] = lsqcurvefit(@(p,t) modelFun([p(1), p(2),
p(3)], t), initialParams, time_for_fit, filtered_velocity);

    % Criar a figura para plotagem
    figure;
    hold on;

    % Plotar os pontos da velocidade filtrada
    plot(time_for_fit, filtered_velocity, '--', 'DisplayName',
'Velocidade Medida');

```

```

% Gerar valores ajustados para plotagem da curva ajustada
t_fit = linspace(min(time_for_fit), max(time_for_fit), 100);
y_fit = modelFun(params_fit, t_fit);
record_params(i,:) = params_fit;

% Plotar a curva ajustada com os parâmetros ajustados
plot(t_fit, y_fit, '-', 'DisplayName', sprintf('Curva Ajustada'));

hold off;

% Adicionar título e legendas
title(['Ajuste de Curva - ', num2str(M_value), ' V']);
xlabel('Tempo (s)');
ylabel('Velocidade');
legend('show', 'Location', 'southoutside', 'Orientation',
'horizontal');
xlim([0 limite_x(i)]);
end

```

7.4 Código MATLAB ajuste de parâmetros

```

clc;
clear all;

% Definir variáveis simbólicas
syms U F P2 L dP6 P6 Jm1 dP10 P10 Jc1 P14 mA1 R1 bm1 bc1 bV1 Ke Ne Np

% Matrizes A
A_11 = -R1/L;
A_12 = -Ke/(Ne*Np*mA1);
A_21 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * Ke/(Ne*Np*L);
A_22 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * ( -
1)*(bc1/(Np^2*mA1)+ bm1/(Ne^2*Np^2*mA1) + bV1/mA1);
A_sym = [A_11 A_12; A_21 A_22];

```

```

% Matrizes B
B_11 = 1;
B_12 = 0;
B_21 = 0;
B_22 = -1*(1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^(-1);
B_sym = [B_11 B_12; B_21 B_22];

% Matrizes C e D
C_sym = [0, 1/mA1];
D_sym = [0,0];

% Substituição por valores reais
L_value = 1.00*10^-3;
Jm1_value = 1.00*10^-6;
Jc1_value = 5.62*10^-5;
mA1_value = 1.75*10^-1;
R1_value = 9.52;
bm1_value = 8.80*10^-5;
bc1_value = 8.80*10^-5;
bV1_value = 8.80*10^-5;
Ke_value = 2.51*10^-2;
Ne_value = 1;
Np_value = 1.3564*10^-4;

% Substituir parâmetros nas matrizes
params = [L, Jm1, Jc1, mA1, R1, bm1, bc1, bV1, Ke, Ne, Np];
values = [L_value, Jm1_value, Jc1_value, mA1_value, R1_value,
bm1_value, bc1_value, bV1_value, Ke_value, Ne_value, Np_value];

A = double(subs(A_sym, params, values));
B = double(subs(B_sym, params, values));
C = double(subs(C_sym, params, values));
D = double(subs(D_sym, params, values));

% Importar o arquivo Excel
filename = 'experimento 2 Tcc.xlsx';
data = readtable(filename);
data = data{1:end, :}; % Remover as primeiras duas linhas (se
necessário)

% Separar os dados em matrizes para cada tensão
matrix_6V = data(:, 1:2); % Colunas 1 e 2 para 6V
matrix_12V = data(:, 3:4); % Colunas 3 e 4 para 12V

```

```

matrix_18V = data(:, 5:6); % Colunas 5 e 6 para 18V
matrix_24V = data(:, 7:8); % Colunas 7 e 8 para 24V

% Remover valores NaN de cada matriz
matrix_6V = rmmissing(matrix_6V);
matrix_12V = rmmissing(matrix_12V);
matrix_18V = rmmissing(matrix_18V);
matrix_24V = rmmissing(matrix_24V);

% Normalizar cada matriz individualmente
normalizeMatrix = @(x) (x - min(x(:))) / (max(x(:)) - min(x(:))) * 100;
matrix_6V_normalized = normalizeMatrix(matrix_6V);
matrix_12V_normalized = normalizeMatrix(matrix_12V);
matrix_18V_normalized = normalizeMatrix(matrix_18V);
matrix_24V_normalized = normalizeMatrix(matrix_24V);

% Definir o tempo para cada matriz
time_6V = (0:0.1:(size(matrix_6V_normalized, 1)-1)*0.1)';
time_12V = (0:0.1:(size(matrix_12V_normalized, 1)-1)*0.1)';
time_18V = (0:0.01:(size(matrix_18V_normalized, 1)-1)*0.01)';
time_24V = (0:0.01:(size(matrix_24V_normalized, 1)-1)*0.01)';

% Definir parâmetros do filtro FIR
N = 9; % Ordem do filtro
Wn = 0.001; % Frequência de corte normalizada

% Projetar o filtro FIR passa-baixa
b_filtro = fir1(N, Wn);

% Aplicar o filtro nos dados de posição normalizados
filtered_position_6V = filter(b_filtro, 1, matrix_6V_normalized(:,2));
filtered_position_12V = filter(b_filtro, 1,
matrix_12V_normalized(:,2));
filtered_position_18V = filter(b_filtro, 1,
matrix_18V_normalized(:,2));
filtered_position_24V = filter(b_filtro, 1,
matrix_24V_normalized(:,2));

% Calcular a velocidade usando gradient nos dados filtrados
velocity_filtered_6V = gradient(filtered_position_6V, time_6V);
velocity_filtered_12V = gradient(filtered_position_12V, time_12V);
velocity_filtered_18V = gradient(filtered_position_18V, time_18V);

```

```

velocity_filtered_24V = gradient(filtered_position_24V, time_24V);

% Função do modelo para ajuste de curva
modelFun = @(params, t) params(1) * params(2) * (1 - (1 + t /
params(3)) .* exp(-t / params(3)));

% Parâmetros iniciais [Kp, M, tau]
iniParams = [1, NaN, 1]; % Kp=1 e tau=1; M será definido no loop

% Valores de tensão e velocidades filtradas correspondentes
velocity_filtered = {velocity_filtered_6V, velocity_filtered_12V,
velocity_filtered_18V, velocity_filtered_24V};
time_vectors = {time_6V, time_12V, time_18V, time_24V};

% Definir parâmetros de simulação
U_values=[6;12;18;24]; % Entradas de tensão

for i= 1:length(U_values)

    M_value = U_values(i);
    filtered_velocity = velocity_filtered{i};
    time_for_fit = time_vectors{i};

    % Ajuste de curva usando lsqcurvefit
    iniParams(2) = M_value; % Defina M no vetor de parâmetros iniciais
    [params_fit, resnorm] = lsqcurvefit(@(p,t) modelFun([p(1), p(2),
p(3)], t), iniParams, time_for_fit, filtered_velocity);

    % Gerar valores ajustados para plotagem da curva ajustada
    t_fit{i}= linspace(min(time_for_fit), max(time_for_fit), 100);
    y_fit{i} = modelFun(params_fit, t_fit{i});

end

% Parâmetros iniciais para otimização
initialParams=[bm1_value; bc1_value; bV1_value; Jm1_value; Jc1_value;
R1_value];

```

```

% Definir limites para os parâmetros
lb = [bm1_value/1000; bc1_value/1000; bV1_value/1000; Jm1_value/1000;
Jc1_value/1000;R1_value/10]; % Limite inferior
ub = [bm1_value*1000; bc1_value*1000; bV1_value*1000; Jm1_value*1000;
Jc1_value*1000;R1_value*10]; % Limite superior

% Função de custo para otimização
costFunction = @(params) calculateError(params, U_values, y_fit,
time_vectors);

% Executar a otimização por Algoritmos Diferenciais
options =
optimoptions('particleswarm','SwarmSize',100,'MaxIterations',300);
[best_params] = particleswarm(costFunction, length(initialParams), lb,
ub);

% Atualizar os parâmetros com os melhores encontrados.
bm_values=best_params(1);
bc_values=best_params(2);
bV_values=best_params(3);
Jm_values=best_params(4);
Jc_values=best_params(5);
R_values=best_params(6);

new_values = [L_value, Jm_values, Jc_values, mA1_value, R_values,
bm_values, bc_values, bV_values, Ke_value, Ne_value, Np_value];
params_update=[L,Jm1,Jc1,mA1,R1,bm1,bc1,bV1,Ke,Ne,Np];

% Calcular matrizes A_new,B_new,C_new,D_new com novos parâmetros
A_new=double(subs(A_sym ,params_update ,new_values));
B_new=double(subs(B_sym ,params_update ,new_values));
C_new=double(subs(C_sym ,params_update ,new_values));
D_new=double(subs(D_sym ,params_update ,new_values));

% Criar o sistema de espaço de estados
sys_plot=ss(A_new,B_new,C_new,D_new);

% Simulação e plotagem dos resultados
for i = 1:length(U_values)
    U_step = U_values(i);
    t = t_fit{i};

```

```

% Criar a matriz de entrada u com duas colunas
u = [U_step * ones(length(t), 1), zeros(length(t), 1)];

% Simular a resposta do sistema
[y_simulated, t_sim] = lsim(sys_plot, u', t);
y_simulated = y_simulated * 10^3; % Ajustar unidades para mm/s

figure;
hold on;

% Plotar a curva ajustada com os parâmetros ajustados
plot(t_fit{i}, y_fit{i}, '--', 'DisplayName', 'Curva Ajustada de
Velocidade Experimental');

% Plotar velocidade simulada do sistema de espaço de estados
plot(t_sim, y_simulated, '-', 'DisplayName', 'Velocidade
Simulada');

title(sprintf('Comparação das Velocidades - U=%d V', U_step));
xlabel('Tempo (s)');
ylabel('Velocidade (mm/s)');

legend('Location', 'southoutside', 'Orientation', 'horizontal');

xlim([0 max(t_fit{i})]);
grid on;
hold off;
end

% Criar tabela comparativa dos parâmetros originais e novos
param_names = {'L', 'Jm', 'Jc', 'mA', 'R', 'bm', 'bc', 'bV', 'Ke',
'Ne', 'Np'};
original_values = [L_value, Jm1_value, Jc1_value, mA1_value, R1_value,
bm1_value, bc1_value, bV1_value, Ke_value, Ne_value, Np_value];
new_values = [L_value, Jm_values, Jc_values, mA1_value, R_values,
bm_values, bc_values, bV_values, Ke_value, Ne_value, Np_value];

comparison_table = table(param_names', original_values', new_values',
...
                        'VariableNames', {'Parâmetro', 'Valor
Original', 'Valor Novo'});

% Exibir a tabela comparativa
disp(comparison_table);

```

```
%% Função para calcular o erro total entre as curvas ajustadas e as
velocidades filtradas
```

```
function error_total =
calculateError(params_ini,U_values,y_fit,time_vectors)

syms U F P2 L dP6 P6 Jm1 dP10 P10 Jc1 P14 mA1 R1 bm1 bc1 bV1 Ke Ne
Np

% Matrices A
A_11 = -R1/L;
A_12 = -Ke/(Ne*Np*mA1);
A_21 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) *
Ke/(Ne*Np*L);
A_22 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * ( -
1)*(bc1/(Np^2*mA1)+ bm1/(Ne^2*Np^2*mA1) + bV1/mA1);
A_sym = [A_11 A_12; A_21 A_22];

% Matrices B
B_11 = 1;
B_12 = 0;
B_21 = 0;
B_22 = -1*(1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1);
B_sym = [B_11 B_12; B_21 B_22];

% Matrices C e D
C_sym = [0, 1/mA1];
D_sym = [0,0];

L_value = 1.00 * 10^-3;
mA1_value = 1.75 * 10^-1;

Ke_value = 2.51 * 10^-2;
Ne_value = 1;
Np_value = 1.3564 * 10^-4;
```



```

% Atualizar os parâmetros com os novos valores
bm_new = params_ini(1); % bm
bc_new = params_ini(2); % bc
bV_new = params_ini(3); % bV
Jm_new = params_ini(4); % Jm
Jc_new = params_ini(5); % Jc
R_new = params_ini(6); % R

% Valores atualizados
new_values = [L_value, Jm_new, Jc_new, mA1_value, R_new, bm_new,
bc_new, bV_new, Ke_value, Ne_value, Np_value];
params = [L, Jm1, Jc1, mA1, R1, bm1, bc1, bV1, Ke, Ne, Np];

% Calcular matrizes A, B, C e D com os novos parâmetros
A_new = double(subs(A_sym ,params ,new_values));
B_new = double(subs(B_sym ,params ,new_values));
C_new = double(subs(C_sym ,params ,new_values));
D_new = double(subs(D_sym ,params ,new_values));

% Criar o sistema de espaço de estados
sys_plot = ss(A_new,B_new,C_new,D_new);

total_error = 0;

for i = 1:length(U_values)
    U_step = U_values(i);
    t_fit_i = linspace(min(time_vectors{i}), max(time_vectors{i}),
100);
    u = [U_step * ones(length(t_fit_i), 1), zeros(length(t_fit_i),
1)];

    % Simular a resposta do sistema com os parâmetros atualizados.
    [y_simulated, ~] = lsim(sys_plot, u', t_fit_i);
    y_simulated = y_simulated * 10^3; % Ajustar unidades para mm/s

    filtered_velocity = y_fit{i};

    error_i = immse(y_simulated', filtered_velocity); % Erro
quadrático médio.
    total_error = total_error + error_i;
end

error_total = total_error; % Retornar o erro total
end

```

7.5 Código MATLAB comparação entre modelo e experimento + controle

```

clc;
clear all;

% Definir variáveis simbólicas
syms U F P2 L dP6 P6 Jm1 dP10 P10 Jc1 P14 mA1 R1 bm1 bc1 bV1 Ke Ne Np

%Matriz A
A_11 = - R1/L;
A_12 = - Ke/(Ne*Np*mA1);
A_21 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * Ke/(Ne*Np*L);
A_22 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * ( -
1)*(bc1/(Np^2*mA1)+ bm1/(Ne^2*Np^2*mA1) + bV1/mA1);

A = [A_11 A_12; A_21 A_22];

%Matriz B
B_11 = 1;
B_12 = 0;
B_21 = 0;
B_22 = -1*(1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1);

B = [B_11 B_12; B_21 B_22];

% Y_1 = forca
% Y_2 = velocidade
% Y_3 = corrente

%Matriz C
C_11 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * Ke/(Ne*Np*L);
C_12 = (1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^( -1) * ( -
bc1/(Np^2*mA1) + bV1/mA1);
C_21 = 0;
C_22 = 1/mA1;
C_31 = 1/L;
C_32 = 0;

```

```

C = [C_11 C_12; C_21 C_22; C_31 C_32];

%Matriz D
D_11 = 0;
D_12 = -1*(1 + Jm1/((Ne*Np)^2*mA1) + Jc1/(Np^2*mA1))^(-1);
D_21 = 0;
D_22 = 0;
D_31 = 0;
D_32 = 0;

D = [D_11 D_12; D_21 D_22; D_31 D_32];

% Substituição por valores reais
L_value = 1.00*10^-3;
Jm1_value = 0.0001;
Jc1_value = 0.00094193;
mA1_value = 1.75*10^-1; %real
R1_value = 0.0952;
bm1_value = 0.0088;
bc1_value = 0.0037992;
bV1_value = 0.0059783;
Ke_value = 2.51*10^-2;
Ne_value = 1;
Np_value = 1.3564*10^-4;

% Substituir parâmetros nas matrizes
params = [L, Jm1, Jc1, mA1, R1, bm1, bc1, bV1, Ke, Ne, Np];
values = [L_value, Jm1_value, Jc1_value, mA1_value, R1_value,
bm1_value, bc1_value, bV1_value, Ke_value, Ne_value, Np_value];

A = double(subs(A, params, values))
B = double(subs(B, params, values))
C = double(subs(C, params, values))
D = double(subs(D, params, values))

% Importar o arquivo Excel
filename = 'experimento 2 Tcc.xlsx';
data = readtable(filename);

```

```

data = data{1:end, :}; % Remover as primeiras duas linhas (se
necessário)

% Separar os dados em matrizes para cada tensão
matrix_6V = data(:, 1:2); % Colunas 1 e 2 para 6V
matrix_12V = data(:, 3:4); % Colunas 3 e 4 para 12V
matrix_18V = data(:, 5:6); % Colunas 5 e 6 para 18V
matrix_24V = data(:, 7:8); % Colunas 7 e 8 para 24V

% Remover valores NaN de cada matriz
matrix_6V = rmmissing(matrix_6V);
matrix_12V = rmmissing(matrix_12V);
matrix_18V = rmmissing(matrix_18V);
matrix_24V = rmmissing(matrix_24V);

% Normalizar cada matriz individualmente
normalizeMatrix = @(x) (x - min(x(:))) / (max(x(:)) - min(x(:))) * 100;
matrix_6V_normalized = normalizeMatrix(matrix_6V);
matrix_12V_normalized = normalizeMatrix(matrix_12V);
matrix_18V_normalized = normalizeMatrix(matrix_18V);
matrix_24V_normalized = normalizeMatrix(matrix_24V);

% Definir o tempo para cada matriz
time_6V = (0:0.1:(size(matrix_6V_normalized, 1)-1)*0.1)';
time_12V = (0:0.1:(size(matrix_12V_normalized, 1)-1)*0.1)';
time_18V = (0:0.01:(size(matrix_18V_normalized, 1)-1)*0.01)';
time_24V = (0:0.01:(size(matrix_24V_normalized, 1)-1)*0.01)';

% Definir parâmetros do filtro FIR
N = 9; % Ordem do filtro
Wn = 0.001; % Frequência de corte normalizada

% Projetar o filtro FIR passa-baixa
b_filtro = fir1(N, Wn);

% Aplicar o filtro nos dados de posição normalizados
filtered_position_6V = filter(b_filtro, 1, matrix_6V_normalized(:,2));
filtered_position_12V = filter(b_filtro, 1,
matrix_12V_normalized(:,2));
filtered_position_18V = filter(b_filtro, 1,
matrix_18V_normalized(:,2));

```

```

filtered_position_24V = filter(b_filtro, 1,
matrix_24V_normalized(:,2));

% Calcular a velocidade usando gradient nos dados filtrados
velocity_filtered_6V = gradient(filtered_position_6V, time_6V);
velocity_filtered_12V = gradient(filtered_position_12V, time_12V);
velocity_filtered_18V = gradient(filtered_position_18V, time_18V);
velocity_filtered_24V = gradient(filtered_position_24V, time_24V);

% Função do modelo para ajuste de curva
modelFun = @(params, t) params(1) * params(2) * (1 - (1 + t /
params(3)) .* exp(-t / params(3)));

% Parâmetros iniciais [Kp, M, tau]
initialParams = [1, NaN, 1]; % Kp=1 e tau=1; M será definido no loop

% Valores de tensão e velocidades filtradas correspondentes
M_values = [6, 12, 18, 24];
filtered_velocities = {velocity_filtered_6V, velocity_filtered_12V,
velocity_filtered_18V, velocity_filtered_24V};
times_for_fit = {time_6V, time_12V, time_18V, time_24V};

% Criar o sistema de espaço de estados
sys=ss(A,B,C,D);

% Definir parâmetros de simulação
U_values=[6;12;18;24]; % Entradas de tensão

t=0:0.01:14; % Tempo de simulação de 14 segundos

% Inicializar vetor para armazenar erros relativos
num_timesteps = 100; % Defina o número de passos de tempo que você
espera
erro_relativo = zeros(length(U_values), num_timesteps); % Para
armazenar os erros relativos

```

```

% figure; % Criar uma nova figura para comparação das velocidades

for i = 1:length(U_values)
    U_step = U_values(i);
    u = [U_step * ones(size(t)); zeros(size(t))]; % U é constante e F é
zero

    % Simular a resposta do sistema
    [y, t_sim] = lsim(sys, u', t);
    y = y * 10^3; % Converte para mm/s

    % Plotar as velocidades experimentais e simuladas
    figure;
    hold on;

    M_value = M_values(i);
    filtered_velocity = filtered_velocities{i};
    time_for_fit = times_for_fit{i};

    % Ajuste de curva usando lsqcurvefit
    initialParams(2) = M_value; % Defina M no vetor de parâmetros
iniciais
    [params_fit, resnorm] = lsqcurvefit(@(p,t) modelFun([p(1), p(2),
p(3)], t), initialParams, time_for_fit, filtered_velocity);

    % Gerar valores ajustados para plotagem da curva ajustada
    t_fit = linspace(min(time_for_fit), max(time_for_fit),
num_timesteps);
    y_fit = modelFun(params_fit, t_fit);

    % Plotar a curva ajustada com os parâmetros ajustados
    plot(t_fit, y_fit, '--', 'DisplayName', sprintf('Curva Ajustada -
U=%d V', U_step));

    % Plotar velocidade simulada do sistema de espaço de estados
    plot(t_sim, y(:,2), '-', 'DisplayName', sprintf('Velocidade
Simulada - U=%d V', U_step));

    title(sprintf('Comparação das Velocidades - U=%d V', U_step));
    xlabel('Tempo (s)');

```

```

ylabel('Velocidade (mm/s)');

xlim([0 max(t_fit)]);
grid on;

hold off

% Calcular o erro relativo
min_length = min(length(y(:,2)), length(y_fit)); % Determine o
menor tamanho
erro_relativo(i, 1:min_length) = abs(y(1:min_length, 2) -
y_fit(1:min_length)) ./ abs(y_fit(1:min_length)) * 100;
end

% Plotar todas as curvas em uma única figura
figure;
hold on;
for i = 1:length(U_values)
    plot(linspace(0, max(times_for_fit{i}), num_timesteps),
erro_relativo(i,:), 'DisplayName', sprintf('Erro Relativo - U=%d V',
U_values(i)));
end

title('Erro Relativo entre Velocidades Simuladas e Ajustadas');
xlabel('Tempo (s)');
ylabel('Erro Relativo (%)');
legend('Location', 'southoutside', 'Orientation', 'vertical'); %
Alterado para vertical
grid on;
hold off;

% Criar uma legenda
legend_entries = arrayfun(@(u) sprintf('U=%d V', u), U_values,
'UniformOutput', false);
legend(legend_entries, 'Location', 'southoutside', 'Orientation',
'vertical');

```

```
ts = out.simout; % Supondo que 'out.simout' seja a timeseries
```

```

tempo = ts.Time; % Vetor de tempo
posicao = ts.Data; % Valores da posição

% Criando o gráfico
figure;
plot(tempo, posicao);
xlabel('Tempo (s)');
ylabel('Posição (m)');
title('Gráfico da Posição em Função do Tempo com x_desejado = 50 mm');
grid on;

ts1 = out.simout1; % Supondo que 'out.simout1' seja a timeseries

tempo1 = ts1.Time; % Vetor de tempo
posicao1 = ts1.Data; % Valores da tensão

% Criando o gráfico
figure;
plot(tempo, posicao1);
xlabel('Tempo (s)');
ylabel('Tensão V');
title('Gráfico da Tensão em Função do Tempo');
grid on;

```

7.6 Código Arduino

Obs: O código a seguir sofreu alterações quando registrados as tensões 6V e 12V, no qual o delay usado foi de 100 ms.

```
const int sensorPin = A0; // Pino analógico A0
```



```
void setup() {  
    Serial.begin(9600); // Inicializa a comunicação serial  
    while (!Serial) {  
        ; // Espera pela conexão serial (apenas para placas com USB)  
    }  
}  
  
void loop() {  
    int sensorValue = analogRead(sensorPin); // Lê o valor do pino A0  
  
    // Inicia a gravação quando A0 for 9  
    if (sensorValue == 9) {  
        Serial.println("Iniciando gravação...");  
  
        while (true) {  
            sensorValue = analogRead(sensorPin); // Lê o valor novamente  
            Serial.println(sensorValue); // Envia o valor para o computador  
  
            // Para a gravação quando atingir 1023  
            if (sensorValue == 1023) {  
                Serial.println("Gravação concluída.");  
                break; // Sai do loop de gravação  
            }  
        }  
    }  
}
```

```
}  
  
    delay(10); // Atraso para evitar leituras muito rápidas em 10^-2 seg  
  
}  
  
// Para o código após a gravação  
  
while (true); // Mantém o Arduino em um loop infinito  
  
}  
  
}
```