UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Felipe Scrideli Stefanoni
Stefan Thiago Cury Alves dos Santos

Projeto de controlador PI com base em dados experimentais de resposta em frequência

São Carlos

2017

Felipe Scrideli Stefanoni
Stefan Thiago Cury Alves dos Santos

Projeto de controlador PI com base em dados experimentais de resposta em frequência

Monografia apresentada ao Curso de Curso de Engenharia Elétrica com Ênfase em Sistemas de Energia e Automação, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Manoel Luis de Aguiar

São Carlos
2017

Stefanoni, Felipe Scrideli
S816p Projeto de controlador PI com base em dados experimentais de resposta em frequência / Felipe Scrideli Stefanoni, Stefan Thiago Cury Alves dos Santos; orientador Manoel Luis de Aguiar. – São Carlos, 2017.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2017.

1. Controlador PI. 2. Motor de corrente contínua. 3. Resposta em frequência. 4. Simulação. 5. Estabilidade. I. Santos, Stefan Thiago Cury dos. II. Titulo.

FOLHA DE APROVAÇÃO

Nome: Felipe Scrideli Stefanoni

Título: "Projeto de controlador PI com base em dados experimentais"

Trabalho de Conclusão de Curso defendido e aprovado em 27 / 11 / 2017,

com NOTA 9,5 (nove, cinco), pela Comissão Julgadora:

Prof. Dr. Manoel Luis de Aguiar - Orientador - SEL/EESC/USP

Prof. Titular Marco Henrique Terra - SEL/EESC/USP

Profa. Titular Vilma Alves de Oliveira - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

FOLHA DE APROVAÇÃO

Nome: Stefan Thiago Cury Alves dos Santos

Título: "Projeto de controlador PI com base em dados experimentais"

Trabalho de Conclusão de Curso defendido e aprovado em 27 / 11 / 2017,

com NOTA 9,5 (nove, cinco), **pela Comissão Julgadora:**

Prof. Dr. Manoel Luis de Aguiar - Orientador - SEL/EESC/USP

Prof. Titular Marco Henrique Terra - SEL/EESC/USP

Profa. Titular Vilma Alves de Oliveira - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Dedicamos este trabalho às nossas famílias, uma retribuição ao seu esforço ao longo de nossas vidas e destes nossos anos na universidade.

AGRADECIMENTOS

À fonte de toda força, inspiração, motivação e alegria, Àquele que dá capacidade à humanidade para o desenvolvimento da ciência, a Deus toda a nossa gratidão. Às nossas famílias, base sólida para nossa vida, suporte emocional, espiritual, financeiro, não temos palavras para agradecê-los.

Agradecemos especialmente ao Prof. Dr. Manoel Luís de Aguiar, nosso orientador, cujo auxílio ao longo dos anos na faculdade nos tem sido inestimável, em especial na orientação deste projeto; agradecemos também ao Laboratório de Controle e Eletrônica de Potência - LACEP-SEL/EESC, por fornecer o espaço e os materiais necessários para o desenvolvimento da pesquisa, e aos demais professores responsáveis pelo laboratório. Agradecemos, também, aos professores Shankar P. Bhattacharyya e Ricardo Alzate por fornecerem a inspiração para o projeto ao nos apresentar, em uma de suas visitas na Escola de Engenharia de São Carlos, a metodologia utilizada como base do nosso trabalho.

Não podemos deixar de citar os amigos de laboratório, Paulo, Carlos, William, Thales, Allan, Celton, Edson e Marcelo; além de darem suporte à pesquisa, tornaram os momentos no laboratório ainda mais agradáveis.

Agradecemos ao Grupo desenvolvedor do Pacote USPSC, o qual desenvolveu o modelo para LaTeX utilizado neste relatório.

Por fim, agradecemos a todos que, direta ou indiretamente, colaboraram para o desenvolvimento deste projeto.

"A sabedoria é um adorno na prosperidade
e um refúgio na adversidade."
Aristóteles

RESUMO

STEFANONI, F.; SANTOS, S. **Projeto de controlador PI com base em dados experimentais de resposta em frequência**. 2017. 111p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2017.

Este trabalho realiza uma implementação e teste real do método de acordo com a proposta apresentada por Keel e Bhattacharyya (2008). Foi utilizado um sistema de malha fechada, composto por um motor CC acoplado mecanicamente a um tacogerador, acionado via *chopper* de quatro quadrantes com chaves controladas por geradores de pulsos PWM provenientes de um microcontrolador. Assim, a partir da resposta em frequência experimental obteve-se o conjunto de controladores PI estabilizantes por meio de tratamento de dados. Os dados de simulação e experimentais em malha fechada foram comparados, de maneira a se realizar a análise crítica dos resultados obtidos.

Palavras-chave: Controlador PI. Motor de corrente contínua. Resposta em frequência. Simulação. Estabilidade.

ABSTRACT

STEFANONI, F.; SANTOS, S. **PI controller project based on experimental data**. 2017. 111p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2017.

This project performs and implements a real test of the method *PI controller design based on experimental data*, according to the proposal presented by Keel e Bhattacharyya (2008). By a joint of simulated and experimental data of the frequency response of a closed loop system, composed by a DC motor controlled by Chopper and mechanically grouped with a DC generator, was obtained a bundle of PI controllers through data processing. Both simulated and experimental data in closed loop were compared so that a critical analysis of the results could be done.

Keywords: PI Controller. Direct current motor. Frequency response. Simulation. Stability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARM	*Advanced RISC Machine*
CC	Corrente Contínua
CCS	*CodeComposer*
DC	Duty Cycle
FIR	Resposta ao Impulso Finita
PI	Proporcional-Integral
PWM	*Pulse Width Modulation*
SPCD	Semi-Plano Complexo Direito

LISTA DE SÍMBOLOS

®	Marca registrada
j	Unidade imaginária
ω	Frequência (em rad/s)
r_P	Grau relativo da planta
r_C	Grau relativo do controlador
r_G	Grau relativo do sistema
n	Grau do denominador
m	Grau do numerador
$\rightarrow$	Tende para
∞	Infinito
f_0	Frequência de amostragem
T_0	Tempo de amostragem
K_P	Ganho proporcinal
K_I	Ganho integral
$\mathbb{Z}$	Conjunto dos números inteiros
$\Delta_0^\infty(\phi(\omega))$	Variação líquida de fase
z^+	Número de zeros no SPCD
$\sigma(X)$	Assinatura de $X(j\omega)$
K_t	Constante de torque do motor CC
K_e	Constante de força contra-eletromotriz do motor CC
R_a	Resistência de armadura
L_a	Indutância de armadura
J	Momento de inércia do motor CC
B	Coeficiente de atrito viscoso do motor CC

V_{in}	Tensão de alimentação
f	Frequência de operação
A_{ref}	Amplitude da entrada
A_{out}	Amplitude da saída
v_{ref}	Velocidade da entrada
v_{out}	Velocidade da saída

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho de TCC refere-se do estudo e verificação experimental do método proposto inicialmente por S. Bhattacharyya e L. Keel. Este trabalho se propõe a encontrar um conjunto de parâmetros K_P e K_I que define uma área no plano de soluções K_P x K_I, os quais são obtidos por análise da resposta em frequência do sistema sob estudo.

A resposta em frequência é obtida a partir de um controlador candidato inicial e, em seguida, o sistema é ensaiado em malha fechada com o mesmo. A metodologia proposta por S. Bhattacharyya se baseia na resposta em frequência deste sistema de malha fechada não requisitando nenhuma outra informação sobre a planta do sistema. Nesta metodologia, os dados da resposta em frequência irão fornecer um conjunto de parâmetros do controlador PI que garantem como premissa principal que o sistema permaneça estável para todos os PI's candidatos no conjunto de soluções.

Neste trabalho de TCC usou-se como objeto de estudo um motor CC de imã permanente acionado por um *chopper* de 4-quadrante em ponte H. A presença de incertezas paramétricas, imprecisão de medidas em ensaios e ocorrência de ruídos de medições apresentam-se como desafios de aplicação da técnica e que foi alvo de estudo neste projeto.

A execução deste trabalho foi desenvolvida em 2 frentes, sendo uma dedicada ao estudo e assimilação da proposta do método em estudo e outra na preparação e implementação de uma bancada de ensaios para testes e verificação da metodologia em aplicações reais.

Como resultado da etapa de estudo da técnica, compôs-se inicialmente um algoritmo para realização da simulação da resposta em frequência e a devida extração dos gráficos de Bode e, por fim, seguido de outro algoritmo que processa os dados adquiridos e gera o conjunto de soluções no plano K_P x K_I,.

Após a finalização da montagem da bancada de testes, também foi necessário e realizado um algoritmo contendo filtragens das medidas e geração da versão experimental da resposta em frequência. Em seguida, o mesmo algoritmo de processamento da resposta gera a versão experimental do conjunto de soluções.

Finalmente, verificou-se por simulação e testes reais a pertinência de PI's candidatos dentro e fora da região estabilizante.

1.1 Proposta e objetivos

O estudo de novas técnicas para determinação de controladores é uma tendência nas aplicações de sistemas e controle na sociedade. O projeto de controlador PI com base

em dados experimentais, em toda sua extensão, sugere tal motivação: encontrar maneiras assertivas de modelar sistemas e, em especial, determinar um conjunto controladores aptos partindo somente da resposta em frequência do sistema de estudo sem requisitar sua função de transferência

Primeiramente, os parâmetros da planta são devidamente encontrados através de ensaios e, portanto, possíveis de serem representados em modelos computacionais. Para que assim, em malha fechada, a resposta frequência possa ser simulada e posteriormente testada em bancada.

Com a conformidade entre os dados simulados e reais, o algoritmo de determinação dos controladores pode ser executado e, por fim, a dinâmica do sistema em malha fechada testada para um conjunto de K_Ps e K_Is pertinentemente escolhidos na região encontrada.

1.2 Organização do trabalho

Após esta introdução, o trabalho apresenta, em seu segundo capítulo, uma breve exposição teórica acerca dos temas relevantes ao projeto: teoria de resposta em frequência, teoria de controladores Proporcional-Integral (PI), além do método Projeto de controlador PI com base em dados experimentais, introduzido por Keel e Bhattacharyya (2008). Este capítulo também apresenta os algoritmos para obtenção experimental tanto da resposta em frequência como do conjunto de controladores PI estabilizantes.

No terceiro capítulo, é introduzido o modelo de simulação computacional do sistema; o diagrama de simulação, os parâmetros dos componentes da planta, os algoritmos seguidos e os resultados preliminares obtidos são ali exibidos.

A apresentação detalhada da bancada experimental, contendo os fundamentos teóricos dos elementos do sistema e o procedimento experimental para caracterização dos mesmos, além dos experimentos comprobatórios da caracterização do sistema, são vistos no Capítulo 4.

O Capítulo 5 apresenta os resultados simulados e experimentais obtidos, para a resposta em frequência, conjunto de controladores estabilizantes e respostas ao degrau. São realizadas comparações entre os resultados obtidos em ambiente de simulação e os alcançados em bancada.

Finalmente, o Capítulo 6 apresenta as considerações finais sobre o projeto e o método estudado, conjuntamente com propostas para a evolução desta linha de pesquisa.

2 FUNDAMENTOS TEÓRICOS

Sistemas de controle estão maciçamente presentes no cotidiano humano, abrangendo as mais diversas áreas de conhecimento; aparelhos eletrônicos como celulares e *smartphones*, sistemas de geração, transmissão e distribuição de energia elétrica, eletrodomésticos de linha branca como refrigeradores e máquinas de lavar, equipamentos industriais como motores elétricos e robôs manipuladores, e até mesmo sistemas biológicos como culturas bacterianas, podem ser considerados sistemas de controle, uma vez que possuem em si componentes, ou, em termo mais técnico, processos, que podem ser controlados de forma a operarem de uma maneira específica. De fato, segundo Dorf e Bishop (2008), um sistema de controle é uma interconecção de componentes (processos) que formam uma configuração a qual implica em uma resposta desejada para o sistema.

Se é fato que existem processos importantes para a vida e as atividades humanas, torna-se essencial conhecê-los e estudá-los, de modo que recebam tratamento adequado para que desempenhem suas atividades de forma desejada, tendo em vista o contexto do sistema.

2.1 Técnicas de análise e controle de sistemas

Existem maneiras de se representar modelos de um processo ou sistema, geralmente relacionadas à interação entre a(s) entrada(s) para a construção da(s) saída(s). Formas muito utilizadas, mas que não se encontram no escopo principal deste projeto, são: equações diferenciais; funções de transferência; representação no espaço de estados; diagrama de blocos; diagramas de fluxo de sinal; resposta ao impulso; resposta ao degrau. De acordo com as características do sistema - número de entradas e saídas, linearidade do sistema ou possibilidade de linearização, conhecimento detalhado do processo, entre outras - pode-se utilizar estas representações (DORF; BISHOP, 2008).

No escopo deste projeto, ganha destaque a representação da relação entre entrada e saída de um sistema através da resposta frequencial, descrita por meio do diagrama de Bode. Esta representação será mais profundamente estudada a seguir.

2.1.1 Fundamentos da resposta em frequência

A representação de um sistema no domínio da frequência apresenta vantagens sobre a representação no domínio do tempo. A primeira, e principal, é utilizar simples multiplicações para associar sinais e processos distintos, em vez de convoluções no tempo; também é possível destacar a facilidade de se captar informações como a ordem do sistema, quantidade de polos e zeros, estabilidade, e outras (OPPENHEIM; WILLSKY, 1997).

Matematicamente, um processo que tenha entrada $x(t)$ e saída $y(t)$ pode ser representado como:

$$y(t) = p(t) * x(t) \tag{2.1}$$

sendo $*$ a operação matemática de convolução. Se a equação 2.1 for descrita no domínio da frequência, utilizando-se a transformada de Fourier, ter-se-á:

$$Y(j\omega) = P(j\omega)X(j\omega) \tag{2.2}$$

ou

$$\frac{Y(j\omega)}{X(j\omega)} = P(j\omega) = |P(j\omega)|e^{j\phi(\omega)} \tag{2.3}$$

Desta forma, $P(j\omega)$ é a resposta em frequência do processo estudado $p(t)$. Como se vê, expressa-se o sistema como a razão entre sua saída e sua entrada (considerando uma entrada senoidal, e consequentemente, sendo o sistema linear e invariante no tempo, uma saída senoidal), e o resultado é definido por amplitudes e fases em cada frequência.

O diagrama de Bode representa graficamente este resultado matemático. Em gráficos distintos em escala semilogarítmica, apresenta-se amplitude, em dB, e fase, em graus ou radianos, em função da frequência analisada, em rad/s. A Figura 2.1 mostra um exemplo de diagrama de Bode para um sistema de segunda ordem, descrito em 2.4.

$$P(s) = \frac{(s+30)}{(s+100)(s+500)(s+3000)} \tag{2.4}$$

Diz-se que o sistema exemplificado possui grau relativo 2, uma vez que é o resultado da subtração entre o grau do denominador n e o grau do numerador m - respectivamente, 3 e 1. Matematicamente, pode-se afirmar que o grau relativo do sistema é $r_P = n - m = 2$ (KEEL; BHATTACHARYYA, 2008).

Detalhes importantes para a aplicação deste projeto são relacionados à influência dos polos e zeros sobre as curvas de amplitude e fase. A primeira informação relevante é que cada zero do sistema altera a inclinação da curva de amplitude em $+20dB/década$, enquanto cada polo altera a inclinação desta curva em $-20dB/década$ (OPPENHEIM; WILLSKY, 1997). Portanto, é possível conhecer a ordem do sistema verificando a inclinação da curva de amplitude para $\omega \to \infty$ (KEEL; BHATTACHARYYA, 2008). Também pode-se pré-determinar o comportamento do sistema ao longo do eixo de frequências através do esboço das assíntotas (como desenhado em vermelho na Figura 2.1).

A segunda, referente à curva de fase, é que cada zero incrementa a fase do sistema em $\pi/2rad$ para $\omega \to \infty$, já cada polo incrementa a fase em $-\pi/2rad$ para $\omega \to \infty$

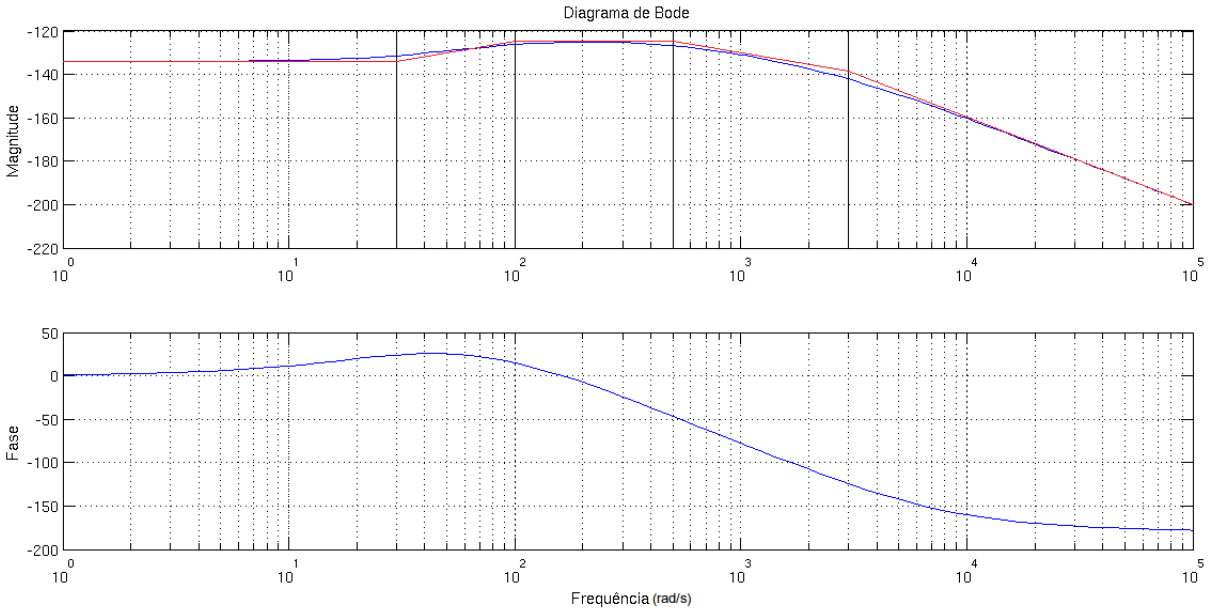

Figura 2.1: Diagrama de bode de um sistema de segunda ordem com três polos e um zero.

(OPPENHEIM; WILLSKY, 1997). Assim, também é possível descobrir a ordem do sistema verificando a fase a qual o sistema tende para $\omega \to \infty$ (KEEL; BHATTACHARYYA, 2008).

Além disso, se há a presença de pares de polos complexos tais que o coeficiente de amortecimento correspondente for menor que $\frac{\sqrt{2}}{2} \approx 0,7071$, haverá um pico na resposta em frequência, correspondente a este coeficiente de amortecimento (OLIVEIRA; AGUIAR; VARGAS, 2016).

2.1.2 Fundamentos de controladores PI

Conhecido o processo, deve-se elaborar uma estratégia de controle para que o sistema tenha resposta desejada. O objetivo, então, torna-se obter um controlador que, em série com o processo estudado e acrescendo-se a realimentação negativa do sistema, resultará na característica de resposta desejada - estabilidade, margens de ganho e fase, tempo de acomodação, sobressinal, entre outros (DORF; BISHOP, 2008). Na Figura 2.2, é possível ver um exemplo genérico de sistema de controle com realimentação.

Reconhece-se, no sistema representado, três componentes: a planta ou processo $P(s)$, o controlador $C(s)$, e o componente de sensoriamento $H(s)$. Além destes, há a representação dos sinais: de referência $R(s)$, de erro $E(s)$, de controle $U(s)$ e de saída $Y(s)$ (ASTROM; HAGGLUND, 2006). Esta notação será utilizada ao longo deste documento.

Dentre os controladores utilizados nas aplicações de engenharia, os controladores do tipo PI ganham destaque. Formados por uma componente proporcional acrescida de um

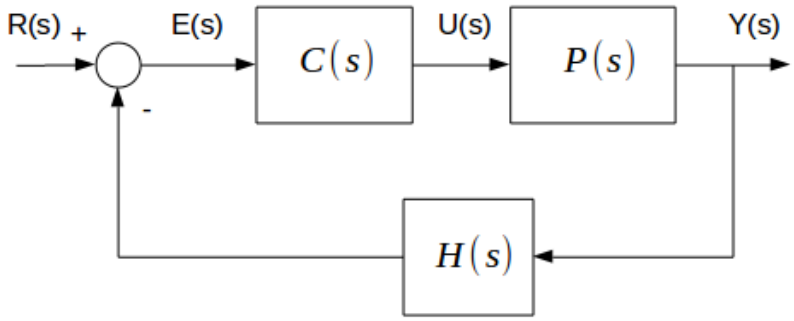

Figura 2.2: Sistema de controle com realimentação negativa.

integrador (polo na origem do plano-s), são compensadores simples e funcionais; podem ser implementados analogicamente, com circuitos eletrônicos, ou digitalmente, por meio de microcontroladores. O equacionamento no domínio da frequência é apresentado na Equação 2.5; a Transformada-z equivalente, obtida por aproximação de integral por um somatório, e para frequência de amostragem $f_0 = 1/T_0$ (OLIVEIRA; AGUIAR; VARGAS, 2013), é apresentado na Equação 2.6, e a lei de controle discreta equivalente pode ser vista na Equação 2.7 (ANTONIOU, 2006).

$$C(s) = K_P + \frac{K_I}{s} = \frac{K_P s + K_I}{s} \tag{2.5}$$

$$C(z) = \frac{K_P z + (T_0 K_I - K_P)}{z - 1} \tag{2.6}$$

$$u[k] = u[k-1] + K_P e[k] + (T_0 K_I - K_P) e[k-1] \tag{2.7}$$

Segundo Nise (2002), o acréscimo de um polo a malha aberta na origem aumenta o tipo do sistema de uma unidade, o que melhora o erro de estado estacionário; por exemplo, caso o sistema seja do Tipo 0 e apresente erro finito, a ação integral fará com que o sistema passe a responder com erro estacionário nulo. Nise (2002) acrescenta que o ajuste do ganho proporcional possibilita obter uma resposta transitória desejada.

Assim, sucintamente, as duas principais características do controlador PI são: zerar (ou reduzir) o erro em estado estacionário; e prover uma dinâmica de resposta transitória desejada ao sistema (KHADRAOUI et al., 2014).

2.2 Obtenção da resposta em frequência da planta

Caso o processo estudado possa ser representado por função de transferência, é possível obter a resposta em frequência matematicamente, através da Equação 2.3. Dispondo-se de recursos computacionais, é possível utilizar a função `bode` dos *softwares* Octave e MATLAB ®. No caso de não se conhecer a descrição matemática do processo, pode-se obter experimentalmente sua resposta em frequência. Os passos a serem seguidos encontram-se a seguir:

1. Estabeleça um ponto de operação para o sistema (tensão, velocidade, corrente, etc.);

2. Acrescente sobre este ponto de operação um sinal senoidal de amplitude e frequência conhecidas;

3. Obtenha a resposta do sistema, que deverá ser senoidal;

4. Meça a amplitude do sinal de saída, calcule a razão entre as amplitudes de saída e de entrada $|P(j\omega)|$;

    a) Para expressar a relação em dB, calcule $|P_{dB}(j\omega)| = 20log(|P(j\omega)|)$;

5. Meça a defasagem entre os sinais de entrada e saída $\phi(P(j\omega))\doteq\phi(\omega)$;

6. Repita os passos anteriores variando a frequência ao longo de um intervalo, o qual deve conter toda a informação de resposta em frequência do processo; em outras palavras, a variação da curva de amplitude fora deste espaço não deve se alterar, deve ser constante em $n * 20dB/década$, $n \in \mathbb{Z}$, bem como a curva de fase deve estar próxima o suficiente de sua assíntota $k * \pi/2$, $k \in \mathbb{Z}$;

7. Represente os dados obtidos em escala semi-log, tal como representado na Figura 2.3.

Destaca-se que os passos acima descritos podem ser seguidos tanto em processos estáveis $P(j\omega)$ a malha aberta como em sistemas a malha fechada $G(j\omega)$. Neste último caso, supondo conhecimento do controlador $C(j\omega)$ e realimentação unitária, é possível obter a resposta em frequência do processo por meio da Equação 2.8 (FNAEICH et al., 2014).

$$P(j\omega) = \frac{G(j\omega)}{C(j\omega)(1 - G(j\omega))} \tag{2.8}$$

2.3 Algoritmo para obtenção de controles estabilizantes

Tendo-se a resposta em frequência da planta $P(j\omega)$, ou do sistema $G(j\omega)$, é possível calcular o conjunto estabilizante de controladores seguindo o procedimento descrito por Keel e Bhattacharyya (2008).

A relevância deste método está no fato de ser independente de modelagens matemáticas do processo. Segundo Ioannou e Sun (1996), projetar controladores que alterem o comportamento e a resposta de uma planta desconhecida pode ser um problema enfadonho e desafiador. Por outro lado, caso se opte pela modelagem matemática da planta, o processo para a obtenção desse modelo, de fato, é apenas um passo intermediário entre a análise e o projeto do controlador (KHADRAOUI et al., 2013). Sendo assim, o procedimento aqui estudado e apresentado, isento das necessidades de modelagem da planta, representa uma alternativa para o projeto de controladores na condição de desconhecimento do modelo do processo.

A sequência de passos iniciais para obter o conjunto estabilizante, conhecendo-se a resposta em frequência de um processo estável $P(j\omega)$, é descrita a seguir:

1. Determine o grau relativo da planta $r_P = n - m$ analisando a inclinação em altas frequências do gráfico de magnitude do diagrama de Bode de $P(j\omega)$;

2. Determine a variação líquida de fase do sistema para a faixa de frequências $\omega \in [0, \infty)$, denotada por $\Delta_0^\infty[\phi(\omega)]$;

3. Calcule o número de zeros do sistema no SPCD do plano-s pela igualdade:

$$\Delta_0^\infty[\phi(\omega)] = -[r_P + 2z^+]\frac{\pi}{2} \tag{2.9}$$

Caso a planta seja instável, ou se a resposta em frequência foi obtida para um sistema a malha fechada $G(j\omega)$, os passos iniciais são:

1. Determine a resposta em frequência do controlador $C(s)$;

2. Encontre a resposta em frequência da planta pela equação 2.8;

3. Verifique o grau relativo da planta $r_P = n - m$ analisando a inclinação em altas frequências do gráfico de magnitude do diagrama de Bode de $P(j\omega)$;

4. Determine o número de zeros no SPCD z_C^+ e o grau relativo r_C do controlador $C(s)$;

5. Determine a variação líquida de fase do sistema $G(j\omega)$ para a faixa de frequências $\omega \in [0, \infty)$, denotada por $\Delta_0^\infty[\phi(\omega)]$;

6. Calcule a assinatura de $G(j\omega)$ através da equação abaixo:

$$\sigma(G) = \frac{2}{\pi}\Delta_0^\infty[\phi(\omega)] \tag{2.10}$$

7. Calcule o número de zeros do sistema no SPCD do plano-s pela igualdade:

$$z^+ = \frac{1}{2}[-r_P - r_C - 2z_C^+ - \sigma(G)] \tag{2.11}$$

8. Calcule $g(\omega)$ pela relação:

$$g(\omega) = -\frac{cos(\phi(\omega))}{|P(j\omega)|^2} \tag{2.12}$$

Após o procedimento inicial, deve-se seguir a sequência apresentada:

1. Fixe uma constante proporcional $K_P = K_P^*$, faça $g(\omega) = K_P^*$, com $g(\omega)$ determinado pela equação 2.12; determine o conjunto de frequências de multiplicidade ímpar $\Omega = \{\omega_1, \omega_2, ..., \omega_{l-1}\}$ que seja solução dessa equação, tal que $\omega_1 < \omega_2 < ... < \omega_{l-1}$;

2. Estabeleça $\omega_0 = 0$, $\omega_l = \infty$ e $j = sgn\{\bar{F}_i(\infty^-, K_P^*)\}$, sendo:

$$\bar{F}_i(\omega, K_P) = K_P|P(j\omega)|^2 + |P(j\omega)|cos(\phi(\omega)) \tag{2.13}$$

3. Determine todos os conjuntos de inteiros $i_t \in \{+1, -1\}$ que solucionem:

   a) Para r_P par:

$$j(-1)^{l-1}[i_0 - 2i_1 + 2i_2 + ... + 2(-1)^{l-1}i_{l-1} + (-1)^l i_l] = r_P + 2z_+ + 2 \tag{2.14}$$

   b) Para r_P ímpar:

$$j(-1)^{l-1}[i_0 - 2i_1 + 2i_2 + ... + 2(-1)^{l-1}i_{l-1}] = r_P + 2z_+ + 2 \tag{2.15}$$

4. Para a constante proporcional fixada $K_P = K_P^*$, encontre os valores de estabilização para a constante integrativa K_I pela inequação:

$$\left[K_I + \frac{\omega_t sen[\phi(\omega_t)]}{|P(j\omega)|^2}\right] i_t > 0 \tag{2.16}$$

   para $t = 0, 1, ..., l$.

5. Repita os itens anteriores atualizando K_P dentro do intervalo prescrito (conforme abordado a seguir).

Uma condição necessária para a estabilização do sistema a malha fechada é que a função $g(\omega) = K_P^*$ possua pelo menos k raízes de multiplicidade ímpar, sendo

$k \geq \frac{r_P + 2z^+ + 2}{2} - 1$, se r_P for par;

$k \geq \frac{r_P + 2z^+ + 3}{2} - 1$, se r_P for ímpar;

Desta forma, a faixa de frequências utilizada $\bar{\omega} \in (\omega_{min}, \omega_{max})$ deve ser tal que contenha todas as k raízes. Por fim, a constante proporcional deve pertencer ao intervalo $K_P = (K_P^{min}, K_P^{max})$ tal que $K_P^{min} = min[g(\omega)]$ e $K_P^{max} = max[g(\omega)]$ para $\omega \in \bar{\omega}$.

2.3.1 Apresentação do código

O algoritmo apresentado na Seção 2.3 conduziu à construção de um código, desenvolvido utilizando o *software* MATLAB ®. Este encontra-se no Apêndice A. Sua estrutura é apresentada a seguir, na Figura 2.3, por meio de fluxograma. No próximo capítulo, será apresentado o modelo de simulação do sistema, para o qual foram aplicados os algoritmos supracitados.

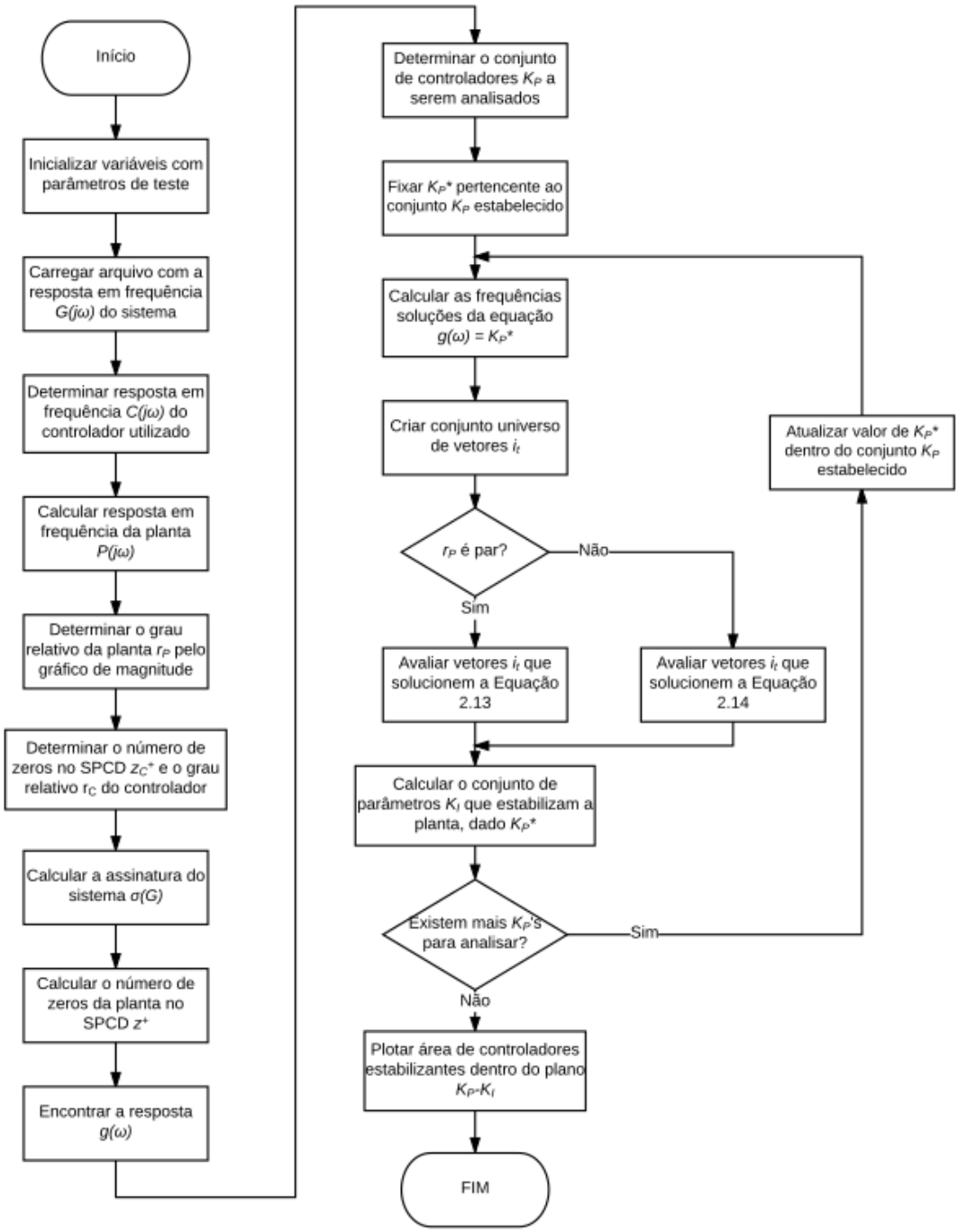

Figura 2.3: Fluxograma referente ao algoritmo para obtenção dos controladores estabilizantes.

3 SIMULAÇÕES EM AMBIENTE COMPUTACIONAL

O primeiro passo tomado para análise do sistema foi a simulação em ambiente computacional, utilizando-se o *software* MATLAB® e o ambiente Simulink ®. Para esta etapa, faz-se necessário conhecer os parâmetros do sistema, que serão apresentados a seguir.

3.1 Apresentação inicial da bancada

A bancada utilizada neste projeto é composta, sucintamente, por: um banco de duas baterias em série, com tensão de saída de 24V; uma ponte H para acionamento; um conjunto formado por um motor CC acoplado mecanicamente a um tacogerador - cujas características elétricas e mecânicas são idênticas às do motor CC; um sensor Hall; e um microcontrolador ARM®.

Na simulação inicial, foram utilizados os dados de placa para a descrição das características elétricas do motor CC no Simulink, estes dados podem ser vistos na Figura 3.1; enquanto isso, os parâmetros mecânicos foram determinados experimentalmente, com o procedimento descrito no Capítulo 4. Assim, os parâmetros utilizados nas simulações são apresentados na Tabela abaixo. Vale destacar que uma descrição mais aprofundada destes elementos, assim como o procedimento para a determinação experimental dos seus parâmetros elétricos e mecânicos, será realizada no Capítulo 4.

Tabela 3.1: Dados e parâmetros do sistema simulado.

Microcontrolador	
Frequência f_0	3 kHz
Motor CC (Dados de	**placa)**
K_t	0,0833 Nm/A
K_e	8,6 mV/rpm
R_a	1,04 Ω
L_a	3,3 mH
Motor CC (Dados	**experimentais)**
J	904,22 g/m^2
B	16,637 Ns
Alimentação	
V_{in}	24 V

3.2 Apresentação do diagrama Simulink® e *script* do MATLAB®

Na Figura 3.2, é possível ver o diagrama de blocos montado para simulação do sistema no ambiente Simulink®.

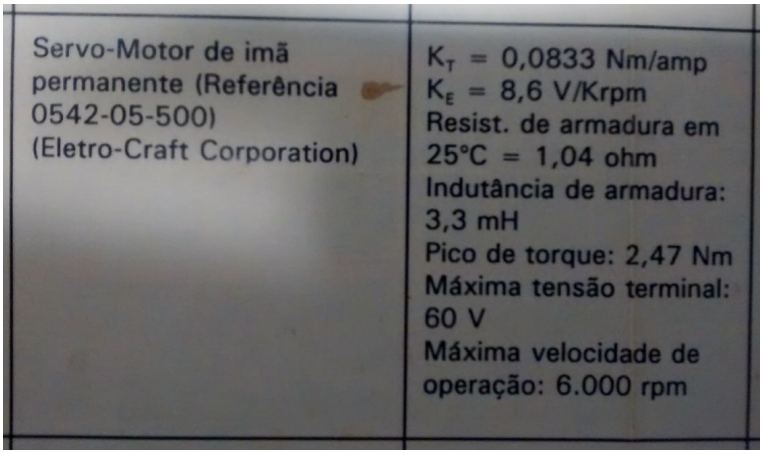

Figura 3.1: Dados elétricos do motor CC utilizado, fornecidos pelo fabricante.

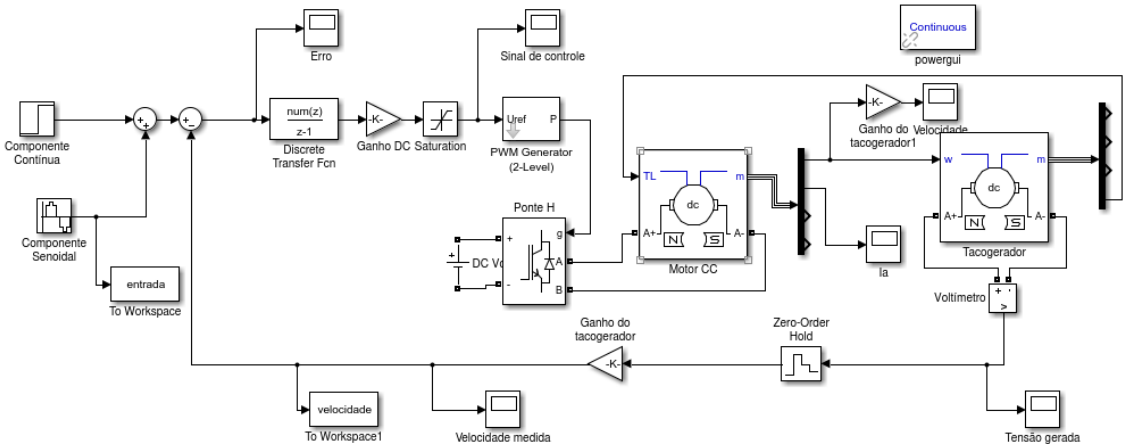

Figura 3.2: Diagrama de blocos para simulação do sistema.

O sistema foi simulado no ponto de operação de 1200rpm, com oscilação de amplitude 100rpm. O patamar de referência foi definido tendo-se como base a velocidade máxima que o motor poderia assumir, em torno de 2500rpm para 24V; assim, tomou-se um valor intermediário, que, a princípio, contemplaria satisfatoriamente as variações em amplitude numa excitação senoidal. Já a amplitude foi definida levando-se em conta que o sinal de saída seria atenuado para altas frequências, porém com possibilidade de amplificação para frequências intermediárias, uma vez que o sistema analisado é de segunda ordem.

Além disso, como se vê, a obtenção da resposta em frequência do sistema é em malha fechada, com controlador PI mostrado na Equação 3.1 (versão analógica) e na Equação 3.2 (versão digital).

$$C(s) = \frac{K_P s + K_I}{s} = \frac{0.01s + 2}{s} \tag{3.1}$$

$$C(z) = \frac{K_P + (\frac{K_I}{f_0} - K_P)z^{-1}}{1 - z^{-1}} = \frac{0.01 - 0,009333z^{-1}}{1 - z^{-1}} \tag{3.2}$$

Em seguida, construiu-se um código de MATLAB® para automatizar o processo de obtenção da resposta em frequência, de acordo com o algoritmo apresentado na Seção 2.2. O código é apresentado no Apêndice B. Abaixo, vê-se o fluxograma referente ao código.

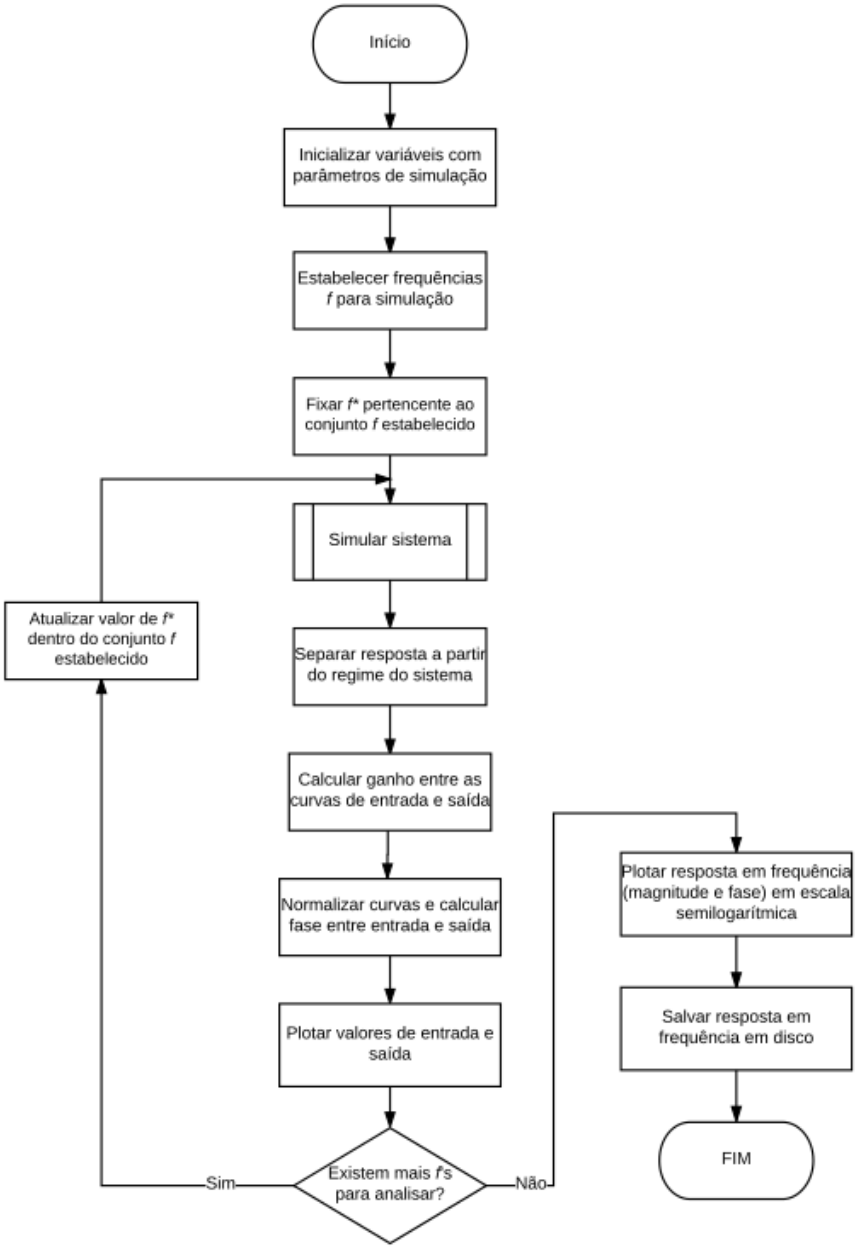

Figura 3.3: Fluxograma referente ao algoritmo para obtenção dda resposta em frequência em ambiente de simulação.

O conjunto de frequências adotadas para este projeto, tanto em simulação computacional como em experimentação em bancada, abrange o intervalo entre $0,8Hz$ e $50Hz$.

Optou-se por usar três conjuntos diferentes de frequências, de forma a otimizar o tempo de execução de cada simulação; por exemplo, para o primeiro conjunto, de frequências mais baixas ($0,8Hz$), o tempo de simulação foi de $4/f$, sendo f a frequência considerada (linha 39); para o segundo conjunto, com frequências intermediárias ($1Hz$ a $29Hz$), o tempo de simulação foi de $28/f$ (linha 62); e para o terceiro conjunto ($30Hz$ a $50Hz$), o tempo de simulação foi de $52/f$ (linha 85).

Esta estratégia foi adotada de forma a garantir a análise de um número mínimo de períodos em regime senoidal, após o transitório referente ao degrau de 1200rpm. Este número mínimo foi de três períodos para o primeiro vetor, quatro períodos para o segundo vetor, e oito períodos para o terceiro vetor. Após a simulação e a obtenção das senoides, o *script* calcula o ganho e a fase, respectivamente, por

$$||G_{dB}(j\omega)|| = 20 * \log\left(\frac{\max(v_{out}) - \min(v_{out})}{\max(v_{ref}) - \min(v_{ref})}\right) \tag{3.3}$$

e

$$\phi(G(j\omega)) = ((t - t_0) * 360 * \frac{\omega}{2\pi}) = ((t - t_0) * 360 * f) \tag{3.4}$$

v_{out} representa a velocidade do sistema e v_{ref} representa a referência de velocidade; t indica o instante no qual o sinal v cruza o zero, enquanto t_0 mostra o instante no qual v_{ref} cruza o zero; e $f = 1/T_0 = \omega/2\pi$ é a frequência de operação do sistema, em Hz. A Figura 3.4 exemplifica os parâmetros anteriormente descritos.

Primeiramente, retira-se o valor médio de velocidade, ou seja, o *setpoint* de 1200rpm. A seguir, calcula-se o ganho do sistema para a determinada frequência através da razão entre as amplitudes de entrada A_{ref} e saída A_{out}; porém, como trata-se de senoides, a amplitude é igual à metade do valor pico a pico, que é dado por $max(v) - min(v)$. Dessa forma, é possível reescrever a equação de ganho como:

$$||G(j\omega)|| = \frac{A_{out}}{A_{ref}} = \frac{(\frac{max(v_{out}) - min(v_{out})}{2})}{(\frac{max(v_{ref}) - min(v_{ref})}{2})} = \frac{max(v_{out}) - min(v_{out})}{max(v_{ref}) - min(v_{ref})} \tag{3.5}$$

que é igual à Equação 3.3.

Para calcular a fase, utiliza-se a relação pertinente aos sinais periódicos, de que um período T_0 equivale a 360°: $\phi = \Delta t \frac{360°}{T_0}$. Uma vez que a defasagem avaliada refere-se aos sinais de entrada e saída, utiliza-se o intervalo de tempo entre o momento em que o sinal de referência passa pelo zero e o instante em que o sinal de saída passa pelo zero. Assim, chega-se à Equação 3.4.

Mais detalhes sobre as frequências utilizadas, as estratégias para processamento dos sinais, e os resultados deste ensaio para obtenção da resposta frequencial, serão apresentados na Seção 5.1.

3.3 Resultados de simulação

Ao executar o *script* apresentado na Seção 3.2, foram gerados dois arquivos de saída: um arquivo de extensão '.mat', contendo a variável `diagramabode`, a qual armazena, respectivamente em cada uma de suas três colunas, os valores de frequência em Hz, de amplitude em dB e fase em graus, e que será utilizado para obtenção dos controles estabilizantes; e uma figura, com a representação clássica do Diagrama de Bode, tal como apresentado na Figura 2.1. Tais resultados serão comentados a seguir.

3.3.1 Resposta em frequência obtida em simulação

Foi realizada a simulação do sistema nas condições apresentadas anteriormente, ou seja, utilizando-se os dados de placa do motor CC. Dessa forma, foram calculadas magnitude e fase do sistema para as diversas frequências consideradas. No Capítulo 5, são apresentados os resultados de simulação com os parâmetros aferidos no sistema, conforme procedimento e dados do Capítulo 4. A Figura 3.4 mostra um exemplo de resposta, para a frequência de $14Hz$. A seguir, obteve-se a resposta em frequência, mostrada na Figura 3.5.

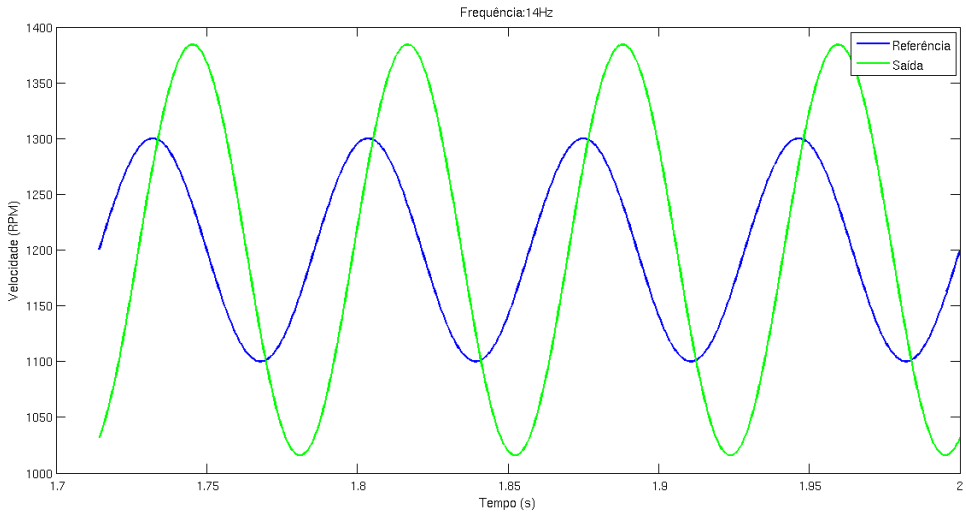

Figura 3.4: Resposta obtida pela simulação do sistema com excitação senoidal de $14Hz$.

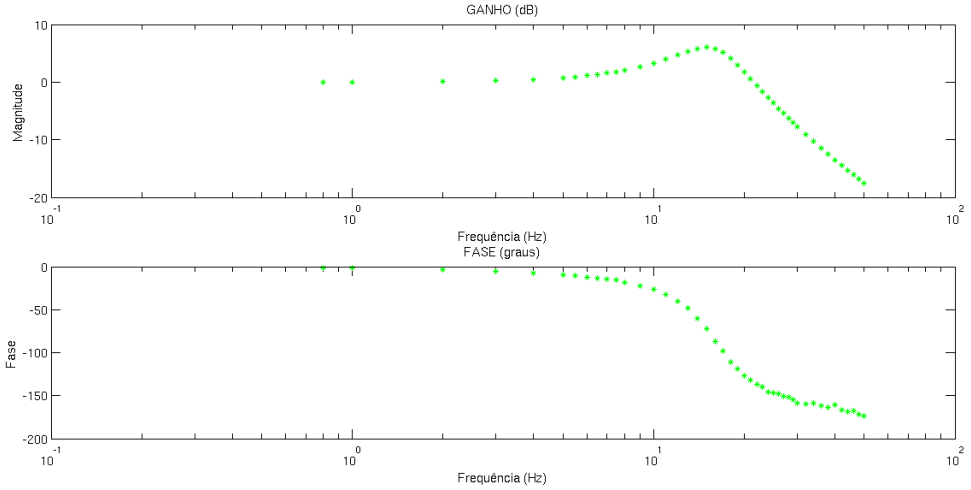

Figura 3.5: Diagrama de Bode obtido por simulação do sistema a malha fechada.

Pode-se ver que a característica do sistema é de segunda ordem: a inclinação em altas frequências da curva de magnitude aproxima-se de $40dB/década$, e a curva de fase tende a 180° para $\omega \to \infty$.

3.3.2 Conjunto de controladores obtidos por meio do algoritmo a partir dos dados de simulação

Após o carregamento do arquivo de extensão '.mat', que contém o conjunto de dados representante da resposta em frequência do sistema simulado, bastou executar o *script* para geração do conjunto de controladores estabilizantes, apresentado no no Apêndice A e introduzido na Seção 2.3.1. O conjunto estabilizante obtido, representado de forma gráfica, é apresentado na Figura 3.6.

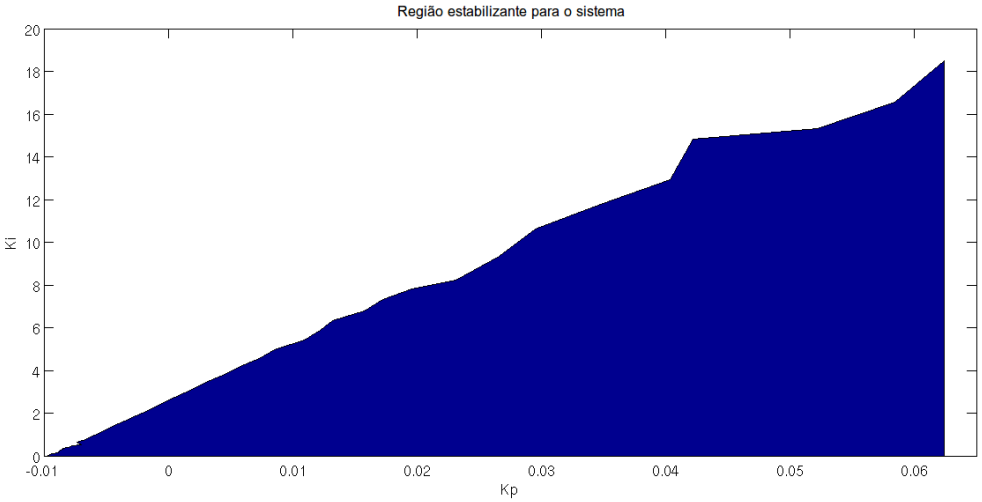

Figura 3.6: Conjunto de controladores estabilizantes obtido por simulação.

Vê-se que há uma área no plano $K_P - K_I$ na qual se garante a estabilidade do sistema a malha fechada, considerando-se as frequências analisadas. Essa área é delimitada pelos intervalos $-0,00970 \leq K_P \leq 0,0620$ e $0 < K_I \leq 18,485$. É importante lembrar que uma das condições iniciais para a sintonia de um controlador PI é que o ganho integral deve ser positivo, o que é respeitado no conjunto estabilizante obtido.

Dando prosseguimento ao projeto, o próximo capítulo descreve em detalhes a planta analisada neste projeto, dando ênfase na sua implementação em bancada e resultados obtidos experimentalmente.

4 PROJETO E IMPLEMENTAÇÃO DA BANCADA DE ENSAIOS

No escopo do projeto, além do estudo teórico descrito no capítulo 2 é imprescindível uma parcela experimental para efeito de comprovação. Neste capítulo serão descritos os procedimentos e equipamentos utilizados na elaboração da bancada contendo o motor CC, a instrumentação, sensores, o *chopper* de acionamento e a plataforma digital de execução dos ensaios.

4.1 Apresentação detalhada da bancada

O projeto requisitou uma bancada com equipamentos e elementos, entre eles: motor de corrente contínua, tacogerador, circuito de potência, sensores, fonte e circuitos condicionadores de sinal, proteção e de medição de grandezas. Todos os equipamentos da bancada e os instrumentos de medição utilizados foram projetados ou adquiridos pelo LACEP.

4.1.1 Motor CC e tacogerador

O motor de corrente contínua representa o elemento a ser controlado. É composto por seu corpo rotativo e seus terminais de alimentação. Na Tabela 4.1 os seus dados nomais.

Tabela 4.1: Dados nominais do motor CC e do tacogerador

K_t	0,0833 Nm/A
K_e	8,6 V/Krpm
R_a	1,04 ohms
L_a	3,3 mH
V_{max}	60 V
ω_{max}	6000 rpm

A escolha de um motor de ímã permanente deu-se por sua simplicidade, isto é, por possuir modelo linear. Seus dois terminais são devidamente conectados no circuito de potência, a interface entre a fonte de alimentação contínua e o controlador. Suas constantes elétricas, mecânicas e forma construtiva oferecem a flexibilidade requisitada para cada estado de operação.

O tacogerador, com as mesmas características do motor CC, foi a estratégia adotada para o sensoriamento de velocidade que é calculada com base na tensão gerada em seus terminais. Um *Encoder* para a contagem das rotações poderia ser utilizado, entretanto, optou-se por tratar a tensão gerada através de um conversor analógico-digital proveniente

do microcontrolador utilizado, conforme será apresentado posteriormente. O acoplamento e fixação do conjunto moto-gerador, como apresentado na Figura 4.1, foram projetados de maneira que as perturbações na rotação do conjunto fossem mínimas.

O funcionamento do motor de corrente contínua é baseado no torque resultante da defasagem entre dois vetores de campos eletromagnéticos. Um, constante, dado pelo imã permanente e o outro, variável, pela imposição de uma corrente contínua nos enrolamentos da armadura. Para que o torque médio durante a rotação do eixo seja mantido na mesma orientação é usado um comutador, cuja função é inverter a polaridade da corrente circulante nas bobinas da armadura e portanto a orientação do campo magnético do eixo (LARAMORE, 1990).

No tacogerador, da mesma forma que no motor CC, também possui campo magnético constante transversal ao eixo rotativo do motor gerador por um ímã permanente. Assim, ao receber torque e rotação externos, as bobinas contidas no eixo, imersas a este campo, são sujeitas à Lei de Lenz. Portanto, resultando em tensão e corrente nos seus terminais da armadura, com sentido de anular a variação de fluxo aplicada nas bobinas. Para que a tensão e correntes gerados mantenham-se na mesma polaridade durante a rotação do eixo é necessário, também, o comutador.

Figura 4.1: Motor CC e tacogerador acoplados.

Na Tabela 4.1 foram listados os parâmetros nominais do motor. Todavia, houve a necessidade de serem novamente calculados através de ensaios para que as simulações fossem o mais próximo da operação real. Para obtenção dos novos parâmetros o motor CC

e o tacogerador foram submetidos a ensaios em vazio Em cada ponto de operação foram medidos: velocidade (ω), corrente (I_a) e a tensão gerada no tacogerador (E_g), apresentados na Tabela 4.3. Assim, através das equações 4.1 e 4.2 de regime permanente da máquina de corrente contínua, os parâmetros K_e e B foram calculados para cada ponto e a média entre eles utilizada.

$$K_e = \frac{60E_g}{2\pi\omega} \tag{4.1}$$

$$B = E_g I_a (\frac{60}{2\pi\omega})^2 \tag{4.2}$$

Para o J é necessário analisar comportamento de decaimento da velocidade. Para isso, retirou-se alimentação do motor CC e analisou-se o comportamento da tensão gerada nos terminais do tacogerador através do osciloscópio. Observa-se, na Figura 4.2 o comportamento da tensão gerada que de 40 Volts em regime cai para $36,8$ % do seu valor em $0,4$ segundos, aproximadamente, como apresentado na tabela 4.2. Assim, como as máquinas possuem parâmetros e constantes equivalentes, através da equação 4.3 o parâmetro pode ser calculado, (FITZGERALD, 1975). Na Tabela 4.4 são apresentados os parâmetros médios reais do motor CC e do tacogerador.

$$J = B\ t_{decaimento} \tag{4.3}$$

Tabela 4.2: Teste de Decaimento

$t_{decaimento}$
0,4

Tabela 4.3: Ensaio em Vazio

E_g	I_a	ω(rpm)	B	K_e
21,72	0,5	2363	0,00017736	0,08777432
25,06	0,525	2738	0,00016004	0,08740152
31,78	0,63	3360	0,00016172	0,09032043
15	0,42	1635	0,00021491	0,08760823

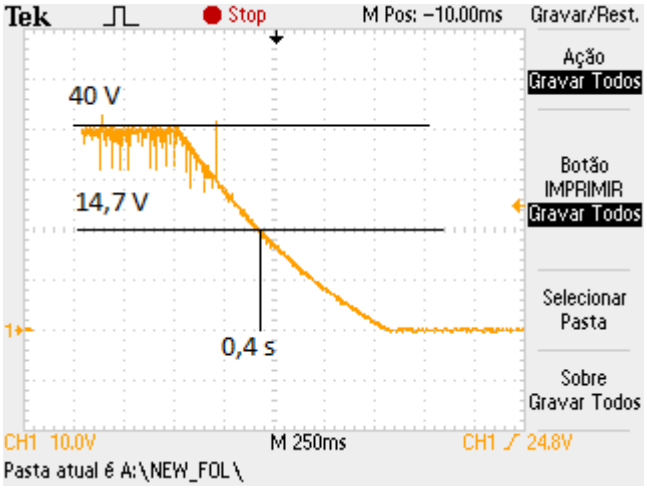

Figura 4.2: Teste experimental de decaimento de velocidade

Tabela 4.4: Parâmetros experimentais do motor CC

K_t	0,0887 Nm/A
K_e	8,57 V/Krpm
R_a	1,13 ohms
L_a	3,3 mH
B	$1,785.10^{-5}$ Ns
J	$7,4101.10^{-5}$ g/m^2

4.1.2 *Chopper* do acionamento do motor CC

O motor de corrente contínua de ímã permanente é controlado pela tensão em seus terminais. Dessa maneira, como interface entre os sinais de controle e a fonte CC há o circuito de potência. Este, desenvolvido no LACEP, é composto de um *chopper* de quatro quadrantes, uma ponte H de transistores e diodos roda livre e seus respectivos *drivers* em uma única placa impressa, como apresentado na Figura 4.4. O circuito tem como entradas uma alimentação para seus elementos ativos, os terminais da fonte CC conectadas na ponte H e os sinais de saída do controlador. A configuração de quatro quadrantes proporciona o acionamento em 4 quadrantes necessário ao experimento. Na Figura 4.3 um esquemático do *chopper* de 4 quadrantes.

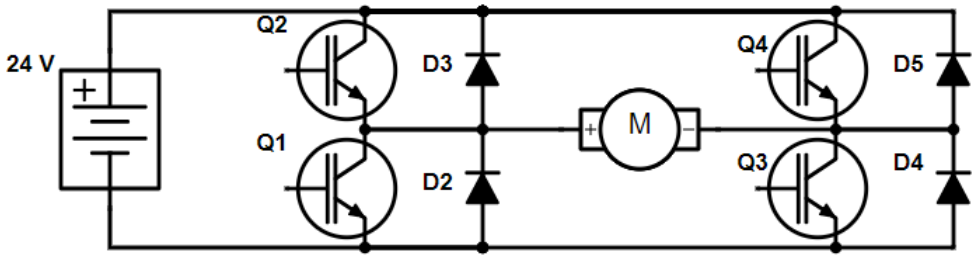

Figura 4.3: Esquemático *chopper* de 4 quadrantes com transistores

O *chopper*, como um conversor CC/CC, é capaz de variar a amplitude da tensão e corrente entregue a carga por uma fonte CC. Através de sinais de comando às suas chaves (transistores) as amplitudes podem ser controladas tão quanto o quadrante de operação do conversor (SEN, 1996). A seguir, os quatro casos:

1. **Tensão e corrente de armadura positivos**: Acionamento do motor. Neste modo, a velocidade e o torque no motor possuem mesmo sentido.

2. **Tensão positiva e corrente de armadura negativa**: Frenagem do motor. Neste modo, há um torque contrário ao sentido de rotação do motor, freando o mesmo. Desta forma, seu funcionamento é equivalente a um gerador.

3. **Tensão e corrente de armadura negativos**: acionamento do motor (sentido contrário) Equivalente ao primeiro quadrante, a velocidade e torque possuem mesmo sentido e portanto opera como acionamento do motor, porém com referenciais opostos ao do primeiro quadrante.

4. **Tensão negativa e corrente de armadura positva**: Frenagem do motor (gerador). Equivalente ao segundo quadrante, porém, com referencial de rotação e torque opostos.

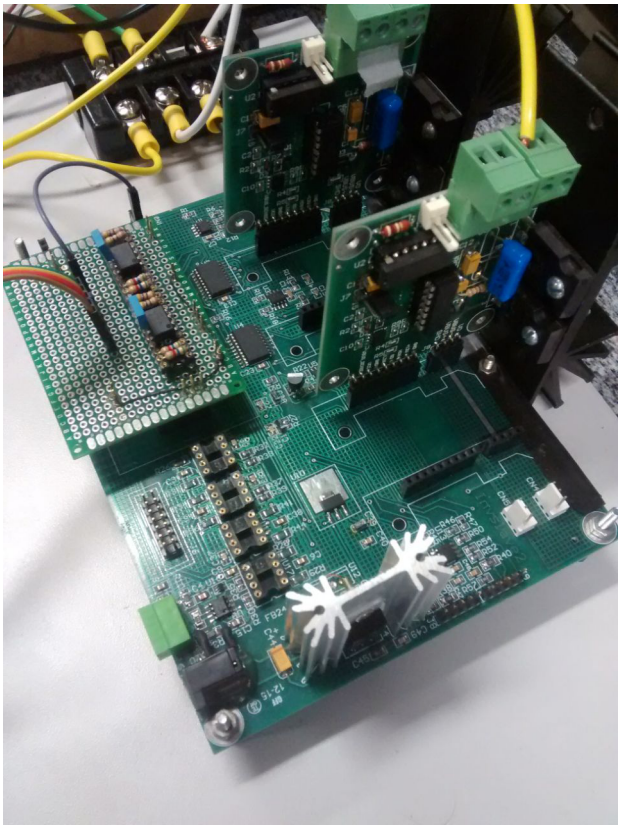

Figura 4.4: Circuito impresso de acionamento

Os sinais de comando aos transistores serão dados pelo microcontrolador através de sinais de modulação por largura de pulso (*PWM*). O transistor utilizado no projeto é do tipo IGBT para aplicações de alta frequência de modelo **IRG4PC50F** da **International Rectfier**. O IGBT (*Insulated Port Bipolar Transistor*) é um transistor de fácil acionamento e de alta velocidade de comutação, como os respectivos MOSFET's e BJT's (BARBI, 2006). Seus valores máximos de corrente e tensão encontram-se na tabela 4.5:

Tabela 4.5: Limites e valores máximos do IGBT **IRG4PC50F**

I_c à 25°C	70 A
I_c à 100°C	39 A
V_{CEO}	600 V

4.1.3 Microcontrolador ARM ®

O microcontrolador utilizado no projeto foi o da categoria ARM ® Cortex M3-Series da Texas Instruments. O módulo ARM ® é parte da *launchpad* LM3S8962, uma plataforma de desenvolvimento com dispositivos auxiliares e periféricos de extrema relevância no projeto, conforme apresentado na figura 4.5. Sua alimentação e comunicação com o compilador é mediante uma conexão USB com o computador de trabalho. A seguir, os principais componentes da plataforma:

- Display Gráfico
- Comunicação Ethernet
- Módulos Geradores de PWM
- 42 portas I/O
- Conversor AD de 10 *bits*
- Botões de comando

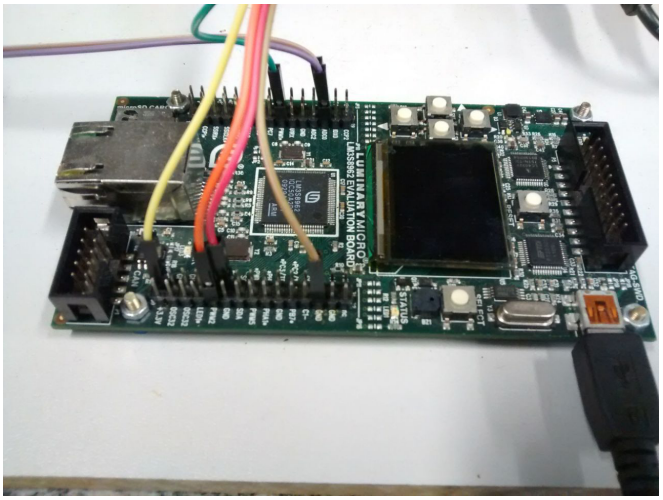

Figura 4.5: Placa de desenvolvimento e microcontrolador ARM ®

Sua programação é realizada na linguagem C/C++, mediante ao *software* Code-Composer (CCS), também da Texas Instruments, disponibilizado pelo kit de instalação da plataforma de desenvolvimento. Além do mais, disponibiliza-se de uma biblioteca de funções já implementadas Stellaris ® Peripheral Driver Library que facilita a configuração de cada periférico requisitado.

O microcontrolador é responsável pelos sinais de controle e acionamento do motor. Nele são tratados os dados de entradas de sensores e do tacogerador adquiridos pelo conversor analógico digital e os respectivos de saída, enviados para o circuito de potência, através dos geradores de PWM. Na Figura 4.6 o fluxograma do código do anexo D de acionamento.

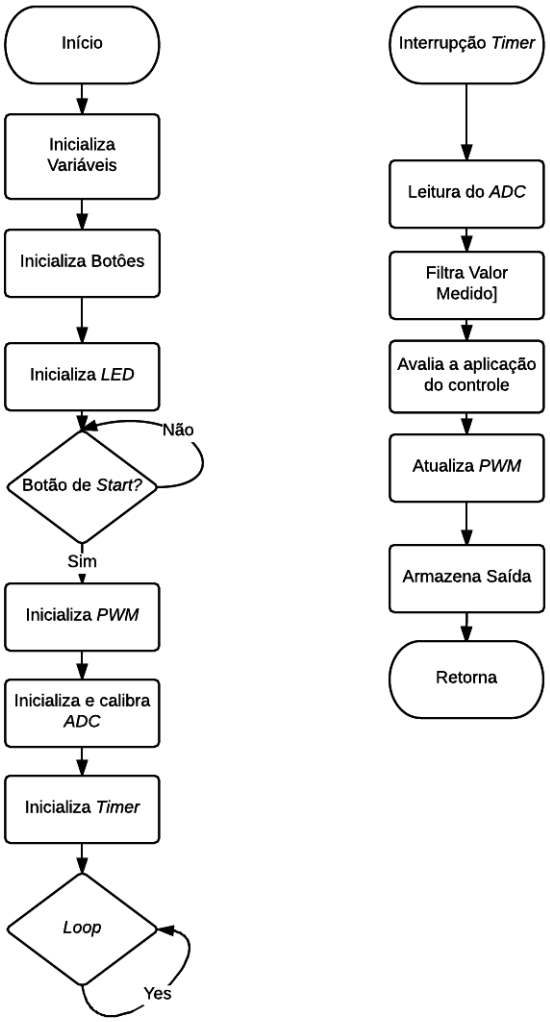

Figura 4.6: Fluxograma do Microcontrolador ARM

Os geradores PWM têm como parâmetros a identificação das saídas, o período da onda e o *duty cycle*. Este que representa a fração da onda efetivamente em T_{ON}, isto é, o período em que o transistor encontra-se acionado. Na figura 4.7 uma representação de modulação.

Para evitar que duas chaves do mesmo par (esquerdo ou direito) sejam acionadas ao mesmo tempo, isto é, um curto-circuito no *chopper*, é utilizado o conceito de *Dead-Band*. Ou seja, é adicionado um atraso de tempo nos PWMs de cada uma das chaves de cada par por via de *software*, garantindo que só um transistor esteja acionado em cada mudança de pulso.

A frequência de 3 kHz uma vez definida é mantida constante durante toda a malha de controle. O *duty cycle*, por sua vez, é atualizado seguindo as equações de controle, conforme apresentado na Figura 4.6. Uma vez que a saída da equação de controle é dada em Volts foi requisitada uma normalização com base na tensão de operação, 24 Volts. Nas

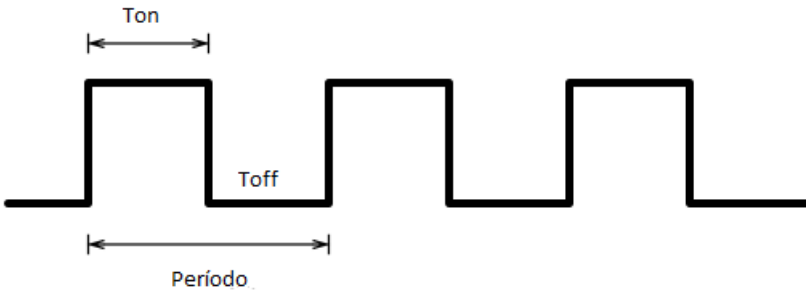

Figura 4.7: Modulação por largura de pulso

figuras 4.8 e 4.9 a configuração de acionamento dos transistores para 1º e 2º quadrantes respectivamente (DEWAN, 1975).

Uma ação de controle negativa, isto é, um *Duty Cycle* negativo não possui sentido físico, mas sim indica que é exigida a operação em outro quadrante. Na Figura 4.9, segundo quadrante de operação, observa-se que os pares de chaves da direita e da esquerda têm acionamento invertido ao apresentado na Figura 4.8. Além do mais, para o respectivo da esquerda, inverte-se o sinal do *duty cycle* multiplicando por -1.

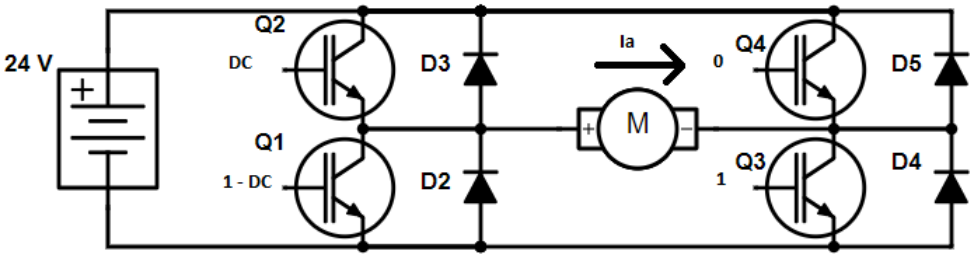

Figura 4.8: Esquemático *chopper* para operação no 1º quadrante.

O conversor analógico-digital (ADC) disponível pela plataforma tem resolução de 10 *bits* e faixa de operação de 0 a 3 Volts. Através da equação 4.4 calcula-se a resolução do ADC:

$$Vadc = \frac{V_{max}}{2^{Nbits} - 1} = 0,00293255V \tag{4.4}$$

Antes de realizar qualquer medição pelo ADC foi executado um algoritmo de calibração para que possíveis incertezas na leitura fossem reduzidas. Sem qualquer tensão nos pinos de entrada, foram lidos 100 valores pelo ADC e por fim a média deles calculada, de maneira que os valores medidos correspondessem somente às incertezas intrínsecas da plataforma. Assim, para qualquer leitura processada no microcontrolador é subtraída a

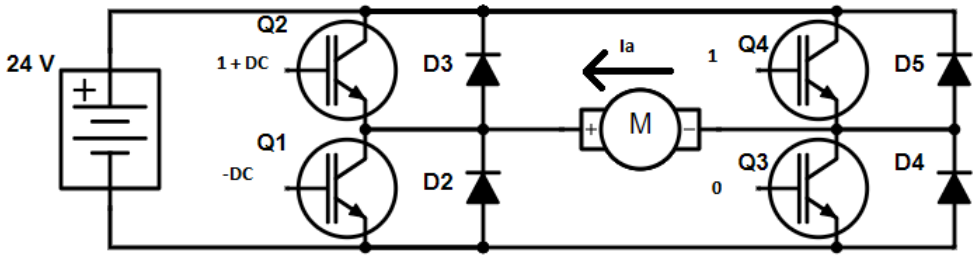

Figura 4.9: Esquemático *chopper* para operação no 2º quadrante.

média obtida, garantindo maior exatidão das medições e consequentemente nos cálculos em que as utilizam (BOUWENS, 1987).

4.1.4 Sensores e condicionadores de sinais

Foram requisitados circuitos auxiliares que condicionassem os sinais tanto de entrada quanto saída, a nível de módulo e polaridade de tensão e corrente. Para entrada, um divisor resistivo e um diodo, uma vez que a tensão gerada pelo tacogerador é maior que 3 Volts , limite do conversor analógico-digital utilizado conforme já apresentado. Dessa forma, para qualquer faixa de tensão gerada o microcontrolador fará a leitura seguindo a conversão apresentada pela equação 4.4.

O diodo, por sua vez, é responsável por proteger o microcontrolador de uma possível tensão negativa gerada nos terminais do tacogerador. Na Figura 4.10 o esquemático para condicionador de sinal para entrada no ADC.

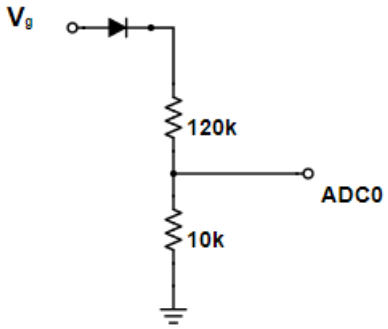

Figura 4.10: Esquemático do circuito condicionador.

Para a saída foi implementado um circuito optoacoplador. Isolando, portanto, os

pinos de saída dos geradores PWM das entradas do circuito de potência. Tal ferramenta é necessária para proteger o controlador e sua plataforma, mantendo suas referências de terra desconexas. Nas Figuras 4.11 e 4.12 um esquemático do circuito optoacoplador e o circuito implementado respectivamente .

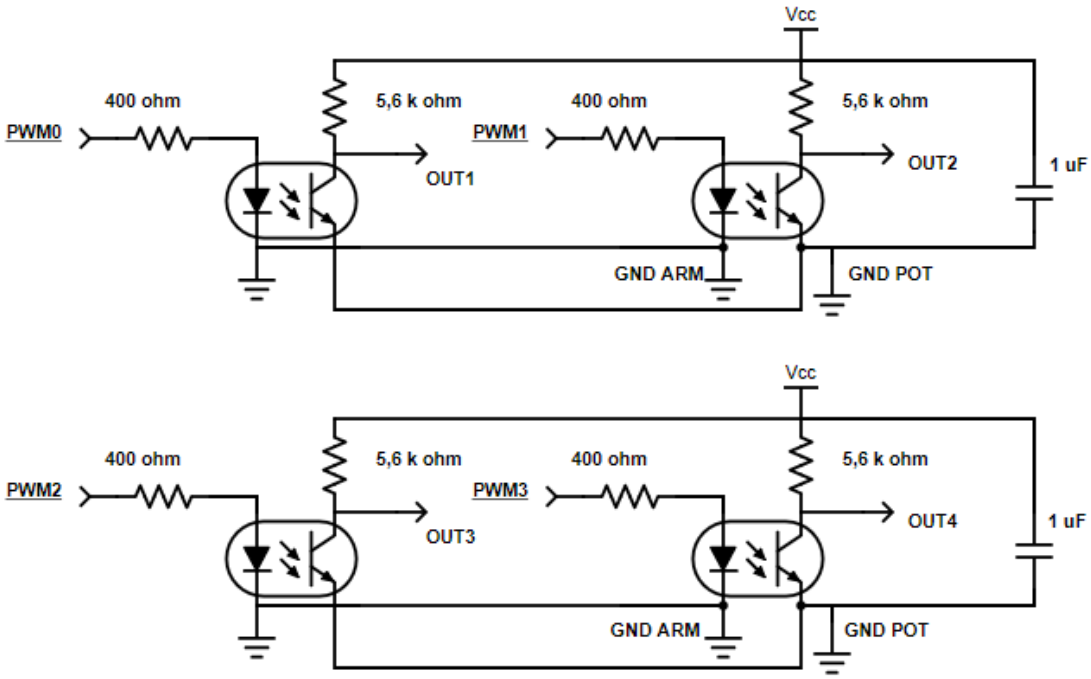

Figura 4.11: Esquemático circuito optoacoplador.

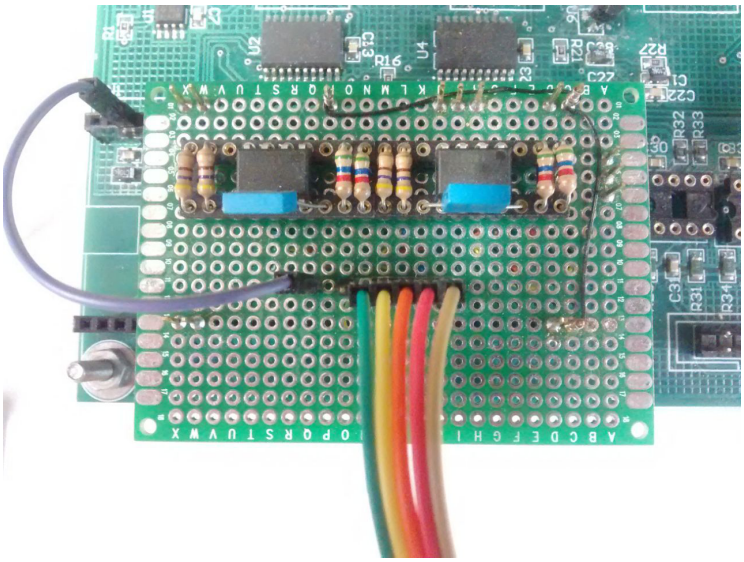

Figura 4.12: Circuito condicionador de sinal de saída do microcontrolador.

Além do mais, a bancada possui um módulo de sensores *Hall* possibilitando, assim,

o sensoriamento da corrente no motor necessário para um controle de corrente e torque em projetos futuros.

4.1.5 Fontes de alimentação

Inicialmente, foi utilizada uma fonte de corrente contínua para alimentação na bancada, a MPL-3303 M da Minipa. Foi sujeita à 20 Volts em cada par de terminais, totalizando 40 Volts, tensão de operação escolhida para os experimentos. Suas saídas são ligadas diretamente no circuito de potência, cada terminal conectado no seu respectivo braço da ponte H.

Entretanto, devido ao seu limite de corrente máxima a fonte não era ideal para os pontos de operação do sistema, que exigiam na maioria dos casos valor acima dos 3 Ampéres máximos para realização do controle, como apresentado na Figura 4.13. Portanto, optou-se por utilizar um conjunto de duas baterias de 12 Volts cada, totalizando 24 Volts. Observa-se que tal nível de tensão é inferior à operação nominal conforme apresentado na Tabela 4.1 porém atende as exigências de corrente. A tensão inferior ao nominal obrigou a escolha de velocidades menores nos testes realizados.

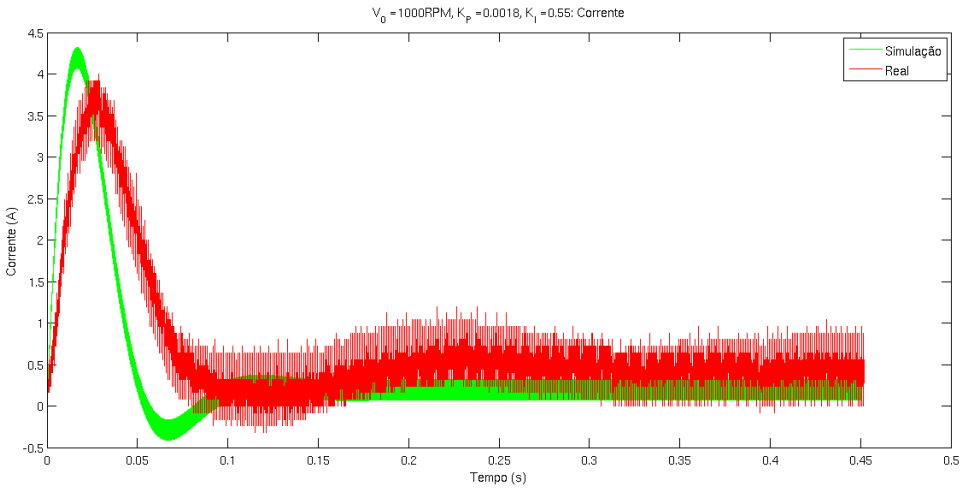

Figura 4.13: Resposta ao degrau (corrente) com corrente acima de 3 Ampéres.

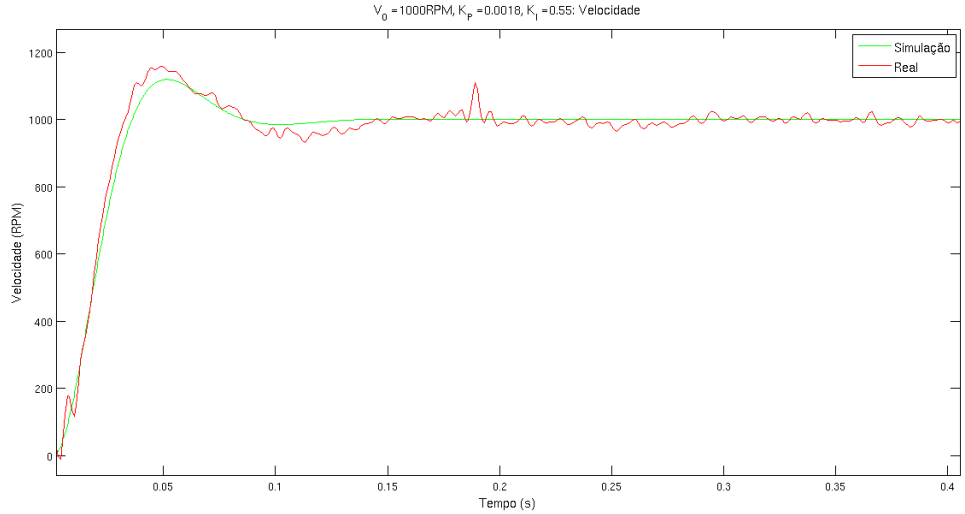

Figura 4.14: Resposta ao degrau (velocidade) com corrente acima de 3 Ampéres.

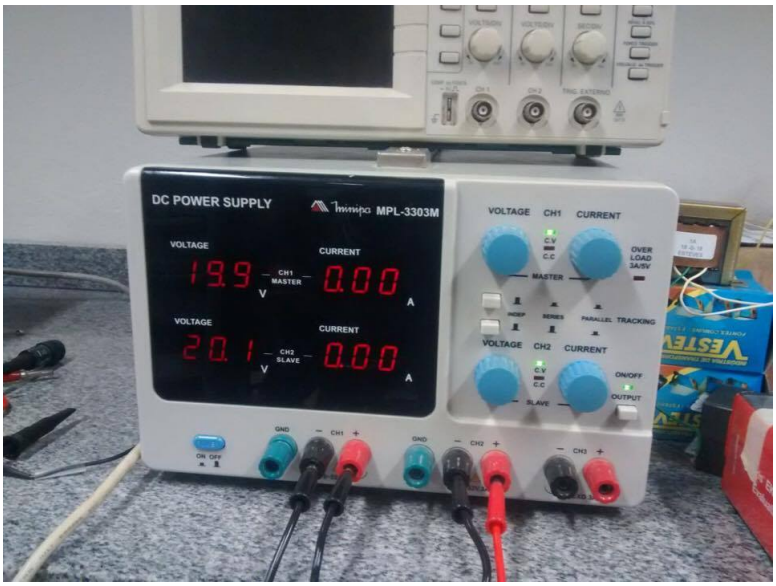

Figura 4.15: Fonte de corrente contínua.

4.1.6 Instrumentação utilizada

A medição e comprovação das grandezas como corrente, tensão e velocidade são necessárias para a confirmação dos resultados esperados. Assim, usou-se de equipamentos de instrumentação como tacômetro, osciloscópio e suas pontas para medição de tensão e corrente, além de dispositivos para armazenagem dos respectivos dados medidos.

O tacômetro em sua configuração de leitura óptica foi utilizado para constatar a veracidade da velocidade aplicada no motor com a referência de controle. O osciloscópio ficou designado à leitura e verificação da tensão gerada pelo tacogerador e da corrente aplicada no motor. Foi configurado de maneira que o transitório de cada ponto de operação

fosse verificado e, por fim, armazenados na memória portátil CompactFlash ® e tratados no Matlab ® .

Figura 4.16: Osciloscópio para aferição da corrente e tensão gerada no tacogerador.

4.2 Dados experimentais

Com a bancada devidamente montada e testada os experimentos pertinentes ao projeto podem ser realizados. Nesta sessão os procedimentos de obtenção da resposta em frequência e resposta em malha aberta tão quanto seus resultados serão apresentados. Na Figura 4.17, a bancada e cada um de seus componentes indicados.

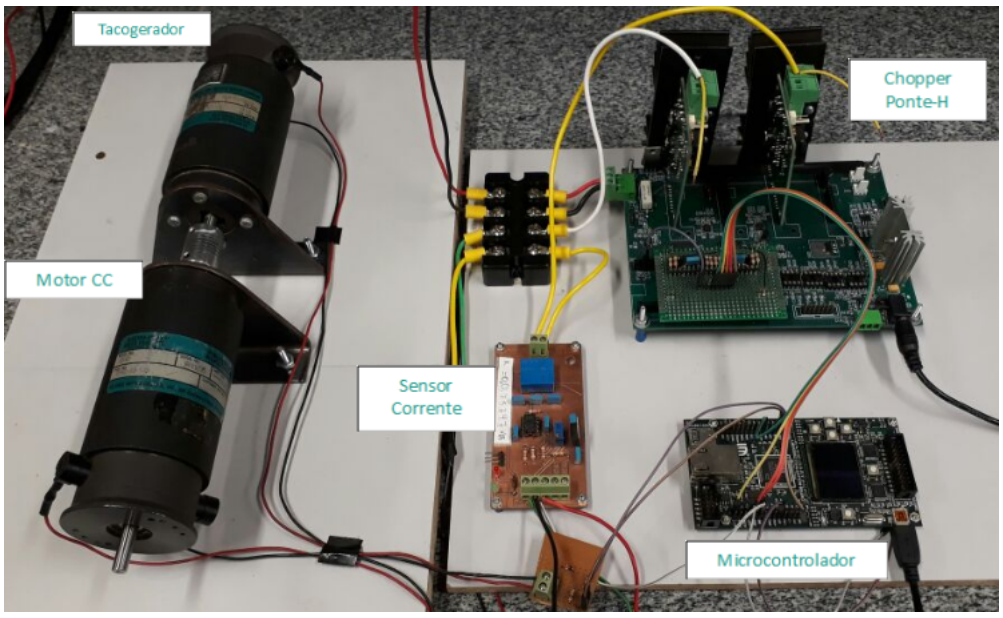

Figura 4.17: Vista completa da bancada.

4.2.1 Procedimento experimental utilizado

A primeira etapa do procedimento experimental é a obtenção da resposta em frequência. A seguir, o procedimento utilizado detalhado:

1. **Definir ponto de operação do sistema:** Fixa-se a frequência de chaveamento e velocidade de referência a partir do *duty cycle* e varia-se a amplitude e a frequência da excitação senoidal escolhidos para os testes.

2. **Ajustar as variáveis devidas no código do controlador:** Com o ponto de operação definido, deve-se ajustar no *script* do microcontrolador as respectivas variáveis.

3. **Energizar a bancada e executar o programa:** Conecta-se os terminais da bateria com os do *chopper*, energiza-se o circuito de potência e a placa de desenvolvimento. Inicia-se o algoritmo pressionando o botão de *Start*.

4. **Armazenar as entradas e saídas do ponto de operação:** Salva-se os dados de entrada e saída do microcontrolador em arquivos de texto. As variáveis apresentadas no osciloscópio são armazenadas na memória CompactFlash ®.

5. **Refazer para as demais frequências:** Interrompe-se o programa com o botão de *Reset* da plataforma e retorna para o primeiro passo.

O código utilizado no microcontrolador para obtenção das respostas é apresentado no apêndice D seguindo o fluxograma na Figura 4.18.

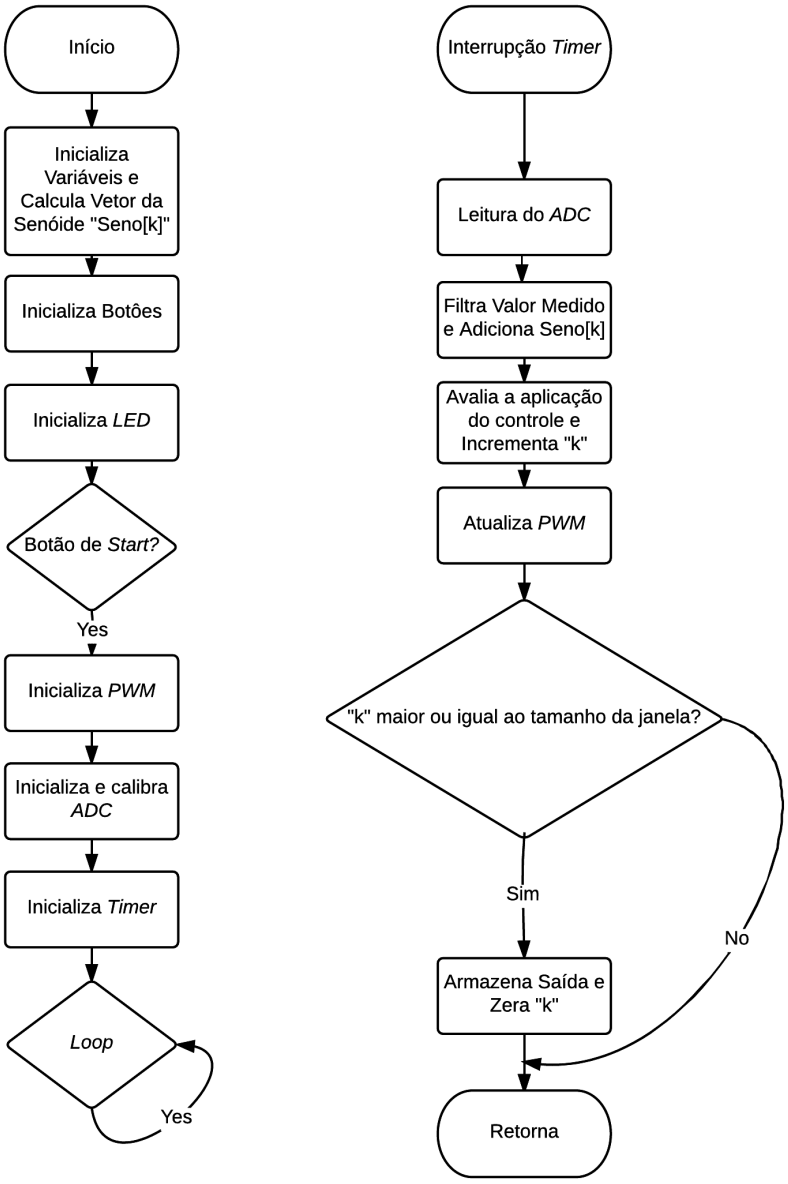

Figura 4.18: Fluxograma de obtenção da resposta em frequência.

Na Figura 4.19 é visualizada um exemplo de resposta em frequência da velocidade com sua componente senoidal a uma frequência de 7 Hz.

Além da resposta em frequência analisou-se a resposta ao degrau em malha aberta do sistema. Para que assim, a dinâmica esperada de primeira ordem do sistema e as simulações possam ser comparadas. A seguir, exemplos de resposta ao degrau em malha aberta para *duty cycles* diferentes.

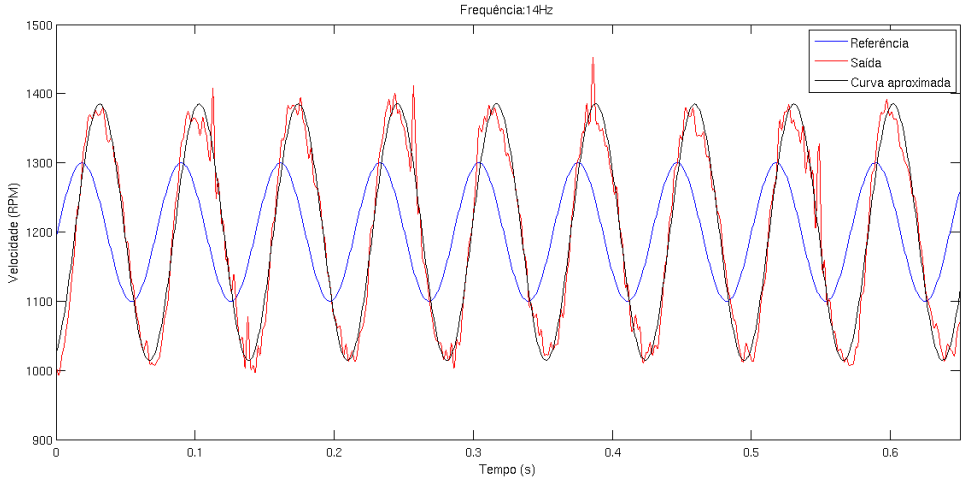

Figura 4.19: Velocidade senoidal para 7 Hz.

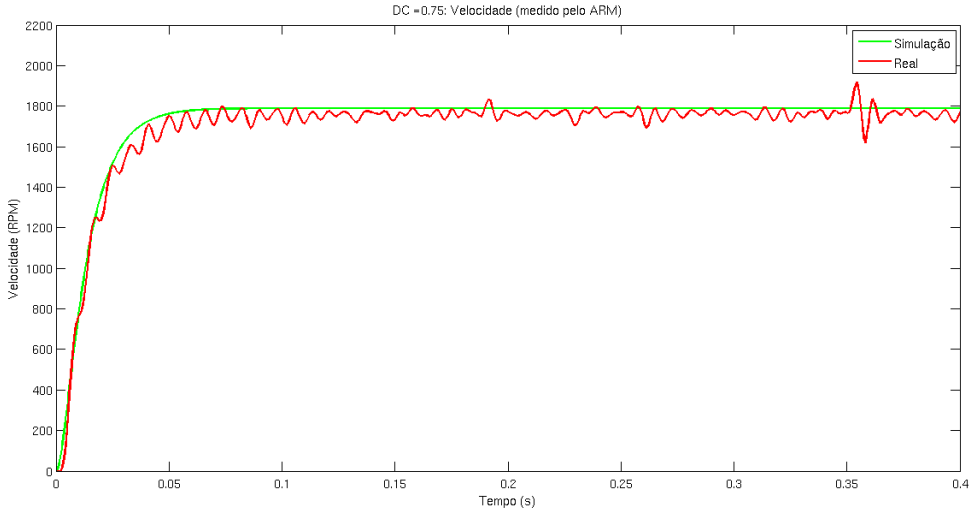

Figura 4.20: Velocidade em malha aperta para *duty cycle* de 0,75.

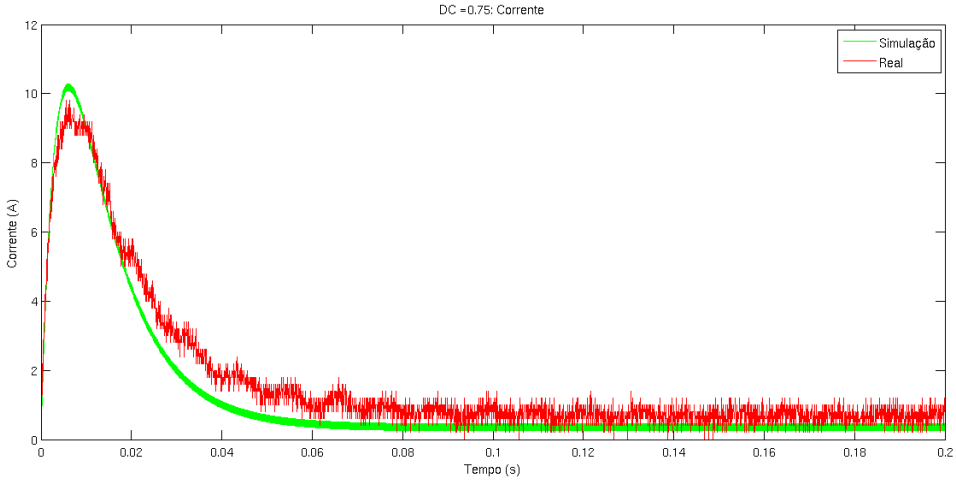

Figura 4.21: Corrente em malha aperta para *duty cycle* de 0,75.

Observa-se, portanto, que existe similaridade entre os dados experimentais e os simulados, confirmando os modelos e parâmetros apresentados durante todo o capítulo. No capítulo seguinte, a análise e tratamento dos dados experimentais serão discutidos e interpretados.

5 RESULTADOS E DISCUSSÃO

Conhecido o sistema, os parâmetros de simulação e os procedimentos experimentais, o passo seguinte é a obtenção dos resultados. Deve-se ressaltar que, para os próximos passos, os parâmetros do Simulink para o diagrama mostrado na Figura 3.2 estão ajustados de acordo com os parâmetros obtidos de forma experimental, conforme descrito no Capítulo 4.

Logo após, realizar-se-á a análise comparativa dos dados, primeiramente das respostas em frequência, dos conjuntos de controladores estabilizantes, e por fim, das respostas ao degrau utilizando-se os controladores informados.

5.1 Respostas em frequência

O algoritmo apresentado na Seção 2.2 foi usado para gerar as respostas frequenciais do sistema, em simulação e em bancada. O conjunto de frequências adotadas para esta etapa abrange o intervalo entre $0,8Hz$ e $50Hz$, com passo entre frequências de 1Hz - à excessão do intervalo entre $5Hz$ e $8Hz$ (passo de $0,5Hz$) e do intervalo entre $30Hz$ e $50Hz$ (passo de $2Hz$).

Em simulação, não foi necessário adotar nenhuma outra estratégia de processamento dos dados além das apresentadas na seção 3.2. A seguir, os resultados de excitação senoidal para as frequências de $0,8Hz$ (Figura 5.1), $7Hz$ (Figura 5.2), $14Hz$ (Figura 5.3) e $36Hz$ (Figura 5.4). Na Figura 5.11, vê-se a resposta em frequência obtida através dos dados experimentais.

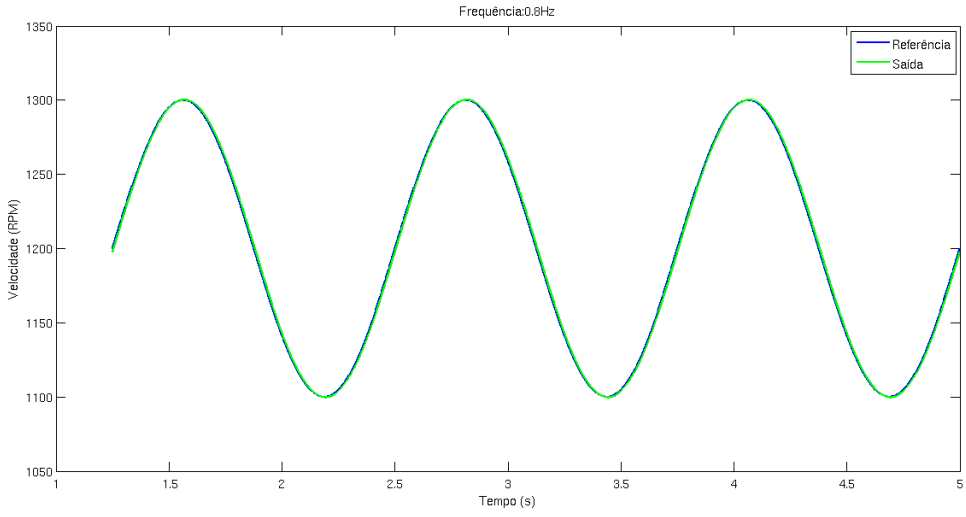

Figura 5.1: Resposta de excitação senoidal (Simulação): $f = 0,8Hz$.

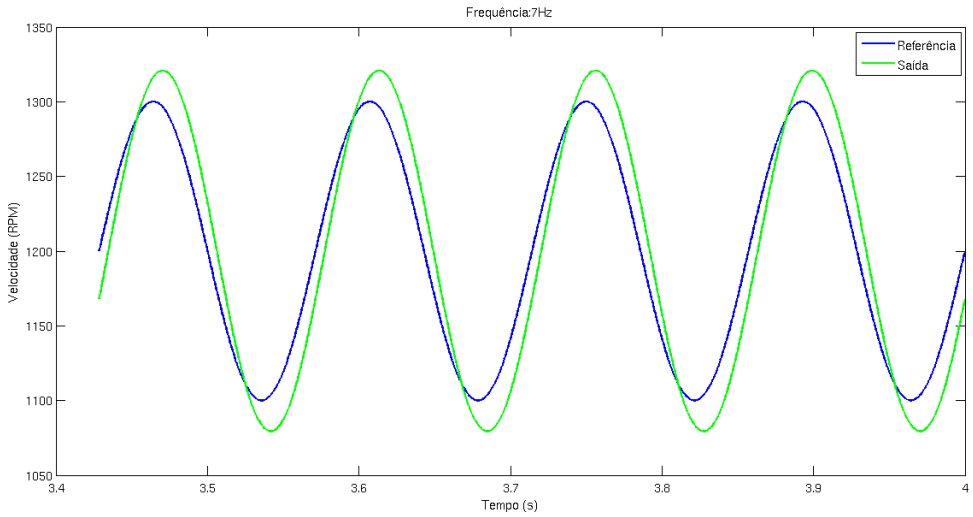

Figura 5.2: Resposta de excitação senoidal (Simulação): $f = 7Hz$.

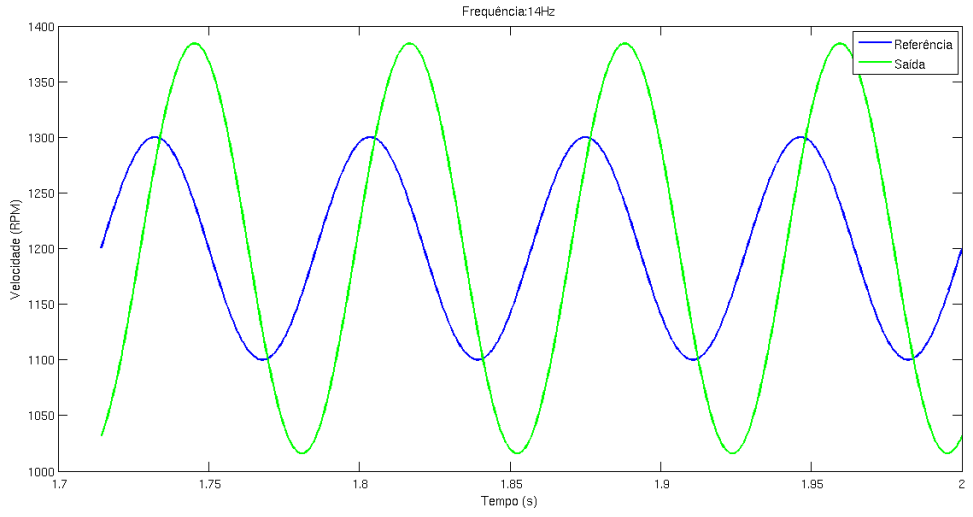

Figura 5.3: Resposta de excitação senoidal (Simulação): $f = 14Hz$.

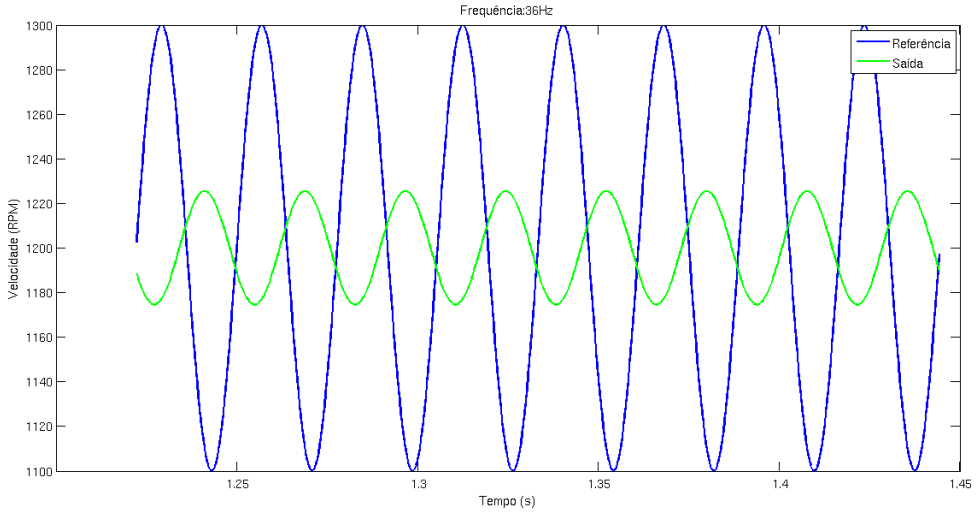

Figura 5.4: Resposta de excitação senoidal (Simulação): $f = 36Hz$.

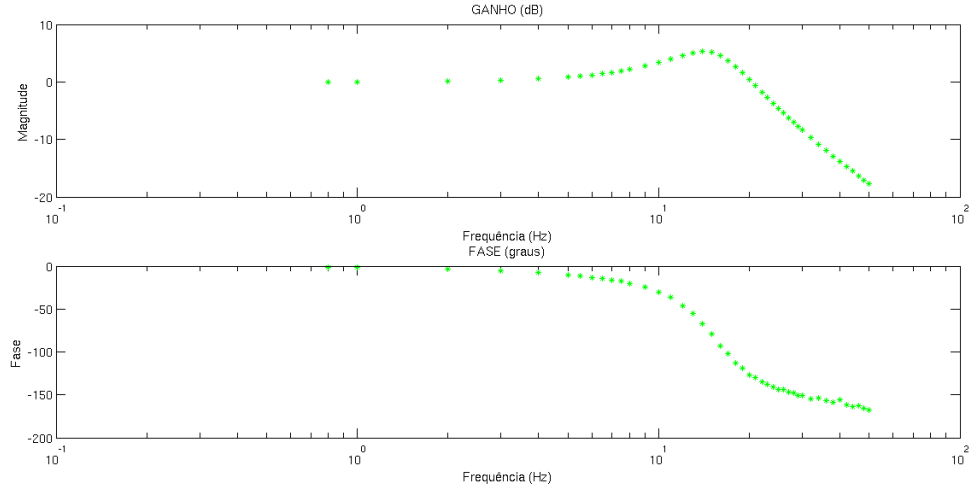

Figura 5.5: Resposta em frequência do sistema a malha fechada (Simulação).

Já para os resultados experimentais, foi preciso filtrar os sinais antes de analisá-los. Adotou-se, para isso, um filtro passa-baixas de resposta ao impulso finita (FIR), ou filtro não-recursivo (ANTONIOU, 2000), de ordem 150 e frequência de corte 600Hz. O comando de MATLAB® utilizado para criar este filtro foi o `designfilt`, e a sua resposta em frequência é apresentada na Figura 5.6. A seguir, o filtro foi aplicado aos sinais analisados por meio do uso do comando `filtfilt`.

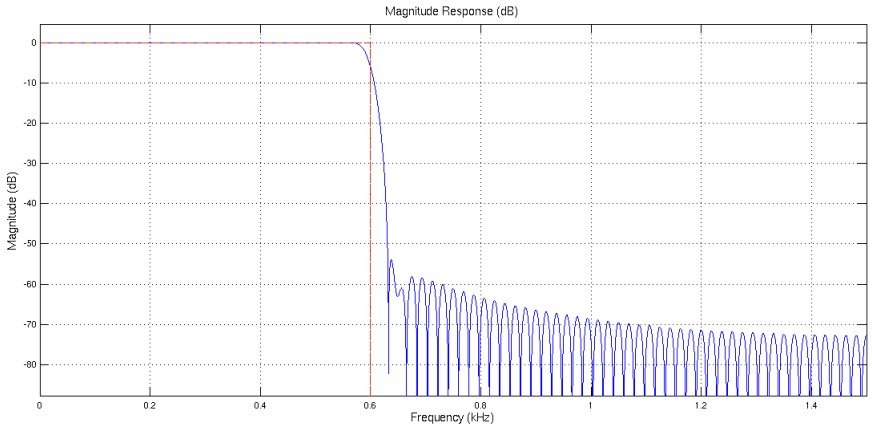

Figura 5.6: Resposta em frequência do filtro FIR aplicado às respostas senoidais (experimental).

Após a filtragem, foi utilizado o comando `fit` do MATLAB® para identificar amplitude e defasagem dos sinais de entrada e saída. Com estas informações, pode-se calcular magnitude e fase entre entrada e saída, de forma a se obter a resposta em frequência.

Como exemplo dos resultados de excitação senoidal, obtidos em bancada, tem-se as Figuras 5.7, 5.8, 5.9 e 5.10, referentes, respectivamente, às frequências de $0,8Hz$, $7Hz$, $14Hz$ e $36Hz$. Na Figura 5.11, vê-se a resposta em frequência obtida através dos dados experimentais.

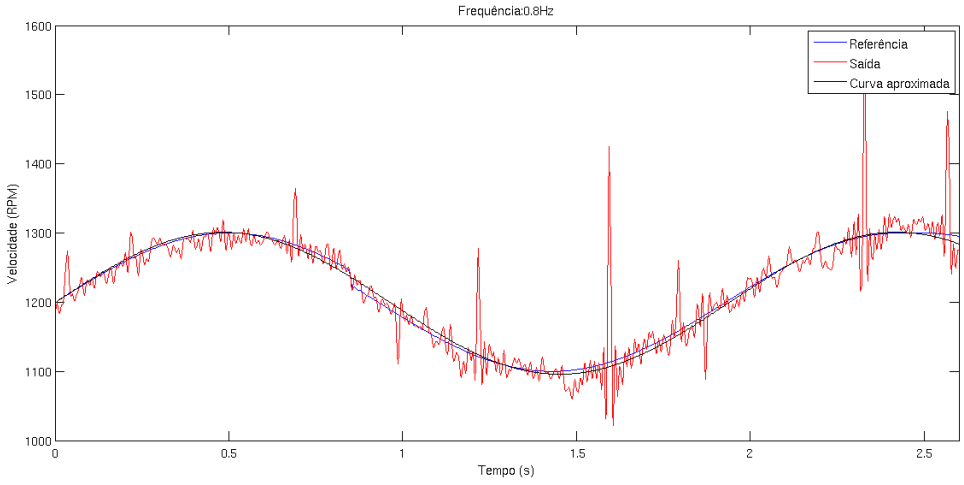

Figura 5.7: Resposta de excitação senoidal (Experimental): $f = 0,8Hz$.

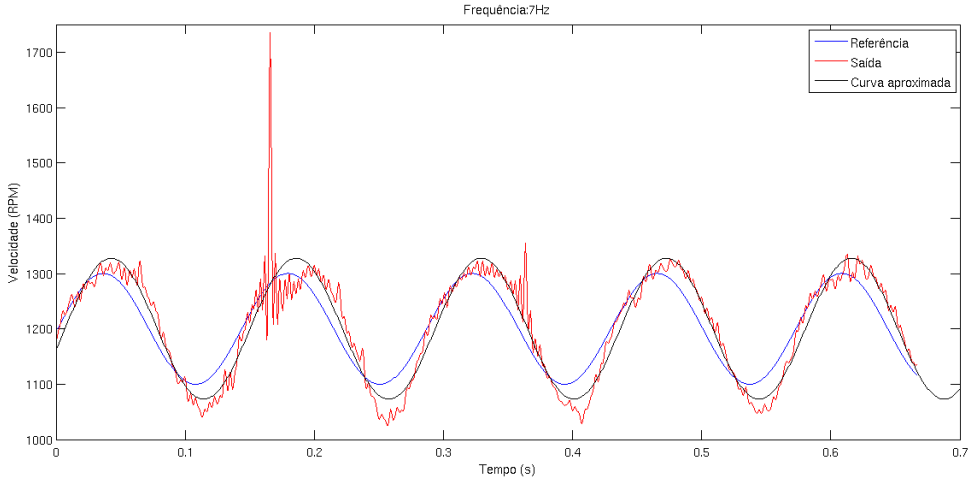

Figura 5.8: Resposta de excitação senoidal (Experimental): $f = 7Hz$.

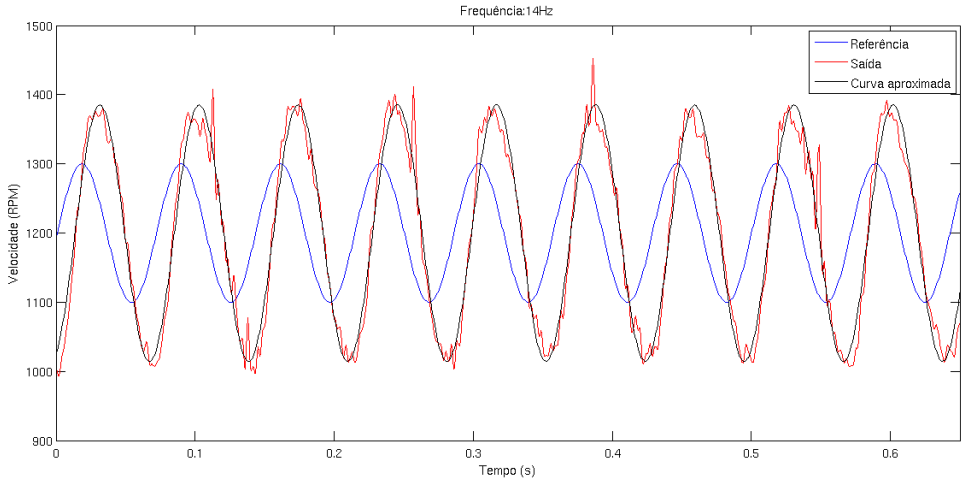

Figura 5.9: Resposta de excitação senoidal (Experimental): $f = 14Hz$.

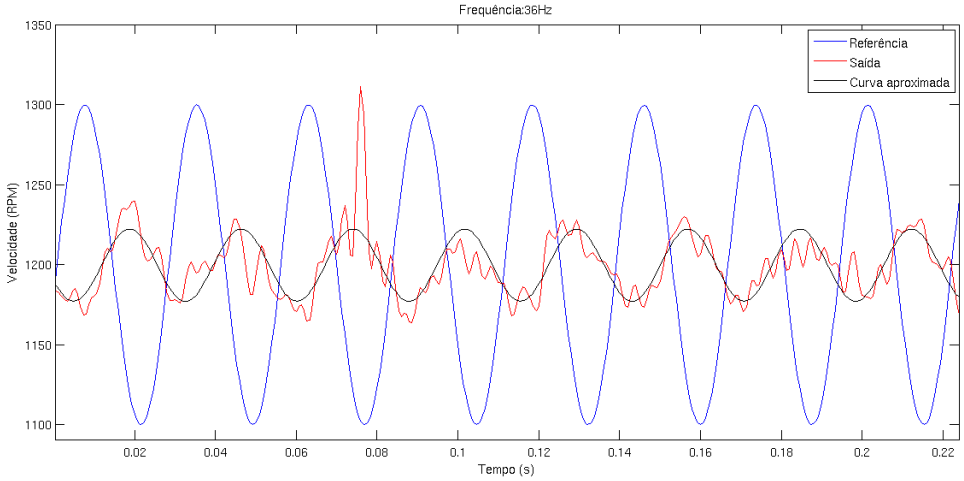

Figura 5.10: Resposta de excitação senoidal (Experimental): $f = 36Hz$.

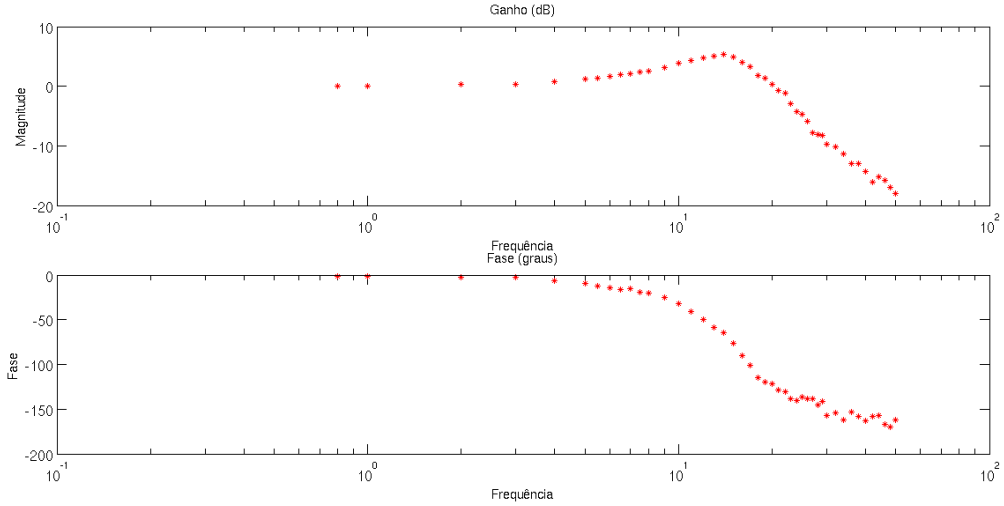

Figura 5.11: Resposta em frequência do sistema a malha fechada (Experimental).

Por fim, na Figura 5.12 pode-se ver as duas respostas em frequência obtidas.

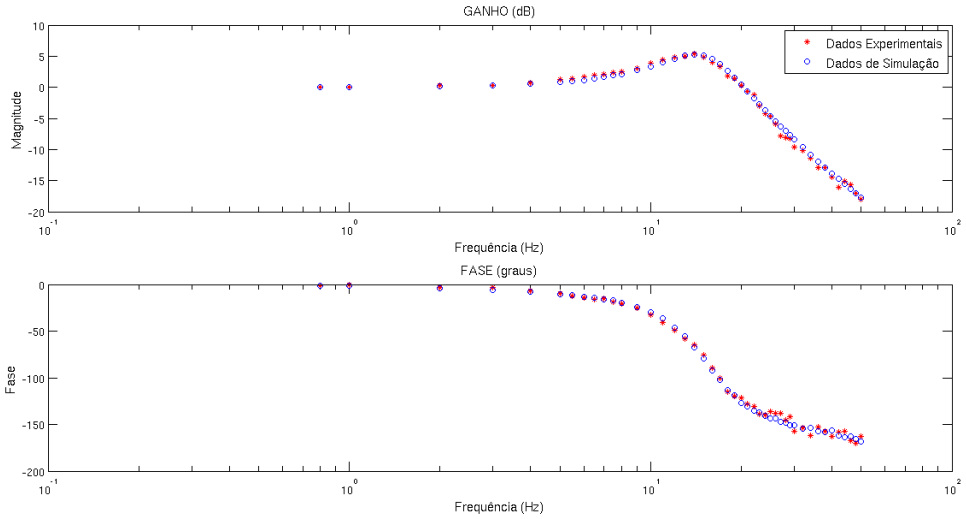

Figura 5.12: Respostas em frequência do sistema a malha fechada: dados simulados e experimentais.

Como se pode ver, a característica de resposta em ambos os casos é semelhante: o sistema é de segunda ordem, com polos complexos tais que o coeficiente de amortecimento é menor que 0,7071. Além disso, vê-se que a banda passante e o valor de pico no gráfico de magnitude e a frequência natural no gráfico de fase apresentam valores próximos, comparando-se resultados simulados e experimentais. Portanto, pode-se validar o sistema de simulação para a resposta em frequência.

Ainda assim, devido às dificuldades de aferição da resposta frequencial à medida que a frequência aumenta, os dados experimentais apresentam certo ruído. Estas diferenças entre as respostas em frequência devem levar a resultados distintos no conjunto de controladores obtidos para as duas situações; esta análise será realizada a seguir.

5.2 Famílias de controladores obtidos

Na Figura 5.13 vê-se, em forma gráfica, os conjuntos de controladores estabilizantes obtidos, para os dados simulados e experimentais.

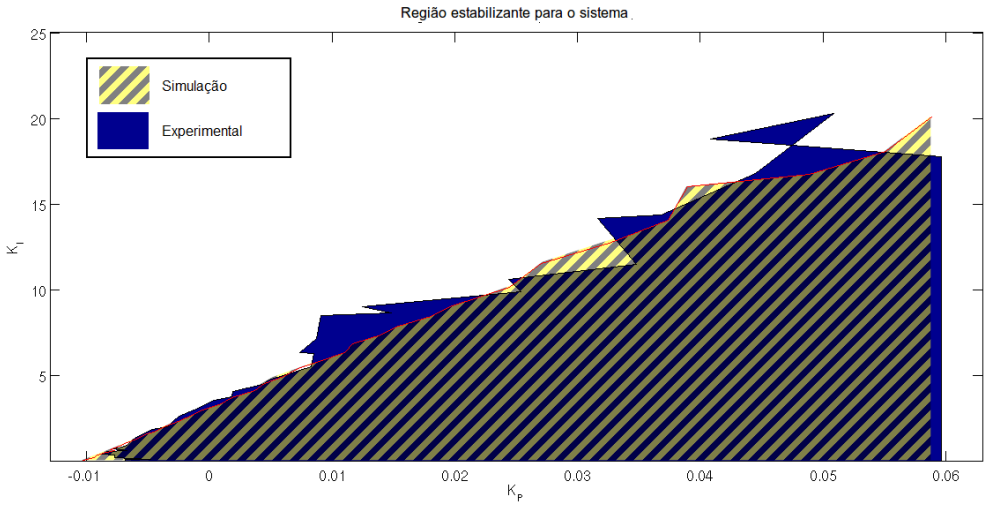

Figura 5.13: Conjunto de controladores estabilizantes: dados simulados e experimentais.

Primeiramente, destaca-se que houve uma pequena diferença entre os intervalos dos ganhos K_P e K_I, comparando-se dados simulados e experimentais. Neste último, os valores de K_P abrangeram de $-0,008806$ a $0,05964$, e K_I apresentou valores de $0,07622$ a $20,30076$. Já em simulação computacional, K_P variou entre $-0,01033$ e $0,05888$, enquanto K_I, entre $0,05888$ e $20,05910$.

A diferença entre os intervalos de valores para K_P e K_I se deve ao fato de o intervalo utilizado para o ganho K_P ser oriundo do intervalo obtido para a função $g(\omega)$, que por sua vez, é relacionada aos dados experimentais de resposta em frequência da planta.

Pode-se ver que, ainda que tenham havido divergências em detalhes das respostas em frequência, existe um grande conjunto de intersecção entre as áreas demarcadas. Além disso, os limites que demarcam a região de controladores estabilizantes são bem próximos, a despeito das irregularidades inerentes aos dados experimentais.

5.3 Comparação de resposta em degrau

Com a obtenção do conjunto de controladores estabilizantes para o sistema, deseja-se averiguar a validade deste método. Para isto, realizaram-se testes de resposta a um degrau de velocidade 1000 rpm, tanto em ambiente de simulação como experimentalmente.

A Figura 5.14 mostra todos os pares de controladores $K_P - K_I$ testados para o sistema. Alguns destes pares encontram-se dentro da região de estabilidade, outros encontram-se fora da tal região, e por fim, também foram testados controladores próximos da fronteira de estabilidade, por vezes dentro de recortes oriundos dos dados experimentais. O *script* para análise dos resultados destes testes encontra-se no Apêndice C.

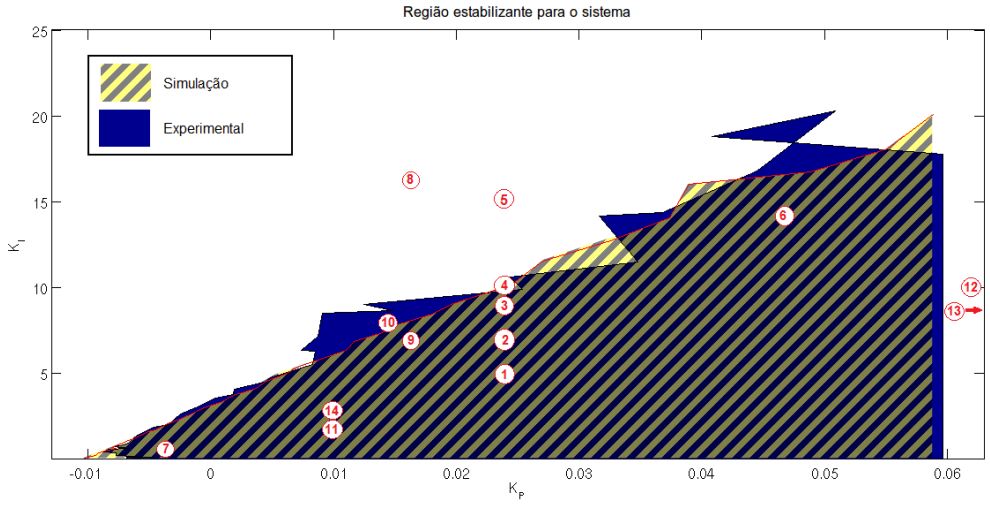

Figura 5.14: Conjunto de controladores estabilizantes: Pontos adotados para testes.

A Tabela abaixo apresenta os valores de controladores apontados na Figura 5.14.

Tabela 5.1: Controladores utilizados para averiguação da resposta ao degrau.

Identificação PI	K_P	K_I
1	0,024	5
2	0,024	7
3	0,024	9
4	0,024	10
5	0,024	15
6	0,048	14
7	-0,004	0,4
8	0,016	16
9	0,016	7
10	0,014	8
11	0,01	2
12	0,062	10
13	0,072	10
14	0,01	2,5

Dos pontos citados, são reproduzidas a seguir as respostas referentes aos controladores 1, 2, 4, 6, 7, 11, 12 e 14. Será apresentada a resposta de velocidade para cada configuração de controlador; porém, ressalta-se que, em todos os testes, foram monitoradas corrente e ação de controle, para verificar seu comportamento. A primeira e a última variáveis foram aquisitadas experimentalmente através do ARM ®, enquanto a segunda foi obtida pelo osciloscópio, com o uso da ponta de prova apropriada.

Os dados obtidos pelo microcontrolador foram tratados com um filtro passa-baixas FIR, de ordem 50 e frequência de corte 600Hz. Como no caso anterior, da Seção 5.1, o comando utilizado para criar este filtro foi o `designfilt`, e a sua resposta em frequência

é apresentada na Figura 5.15. A seguir, o filtro foi aplicado aos sinais analisados por meio do uso do comando `filtfilt`.

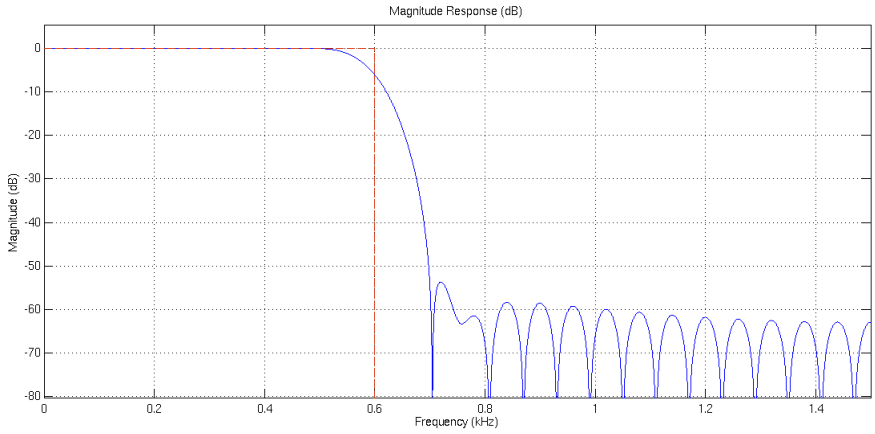

Figura 5.15: Resposta em frequência do filtro FIR aplicado às respostas ao degrau (experimental).

A primeira análise aqui exibida refere-se ao controlador 4, que encontra-se na fronteira da região estável; sua resposta ao degrau encontra-se na Figura 5.16.

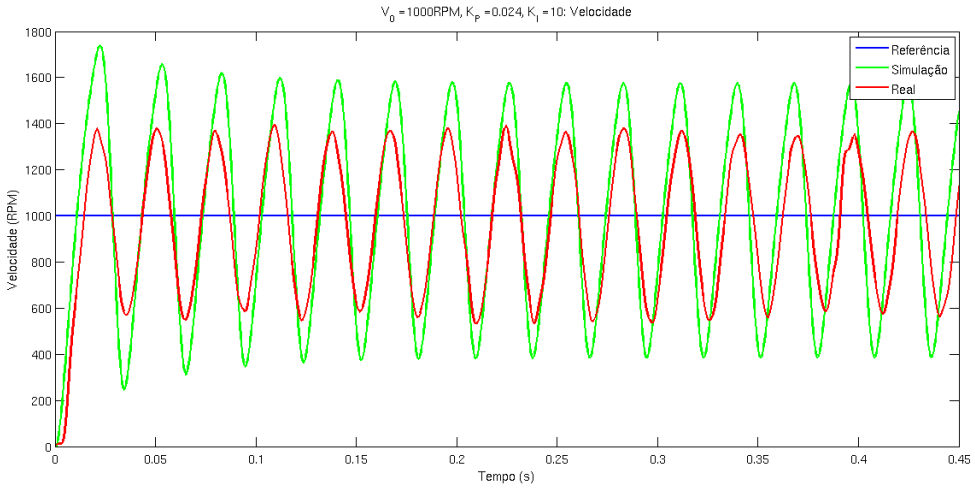

Figura 5.16: Comportamento da velocidade para o controlador 4: $K_P = 0,024$, $K_I = 10$.

Apesar de estar na fronteira, e dentro da região "estável" para os dados simulados, o sistema com o controlador acima apresentou resultado oscilatório após ser excitado por uma referência degrau.

A seguir, testou-se um controlador na extensão da área estabilizadora, para $K'_P s$ maiores. O par de parâmetros referente a esta estratégia é o 12. Sua resposta é verificada na Figura 5.17.

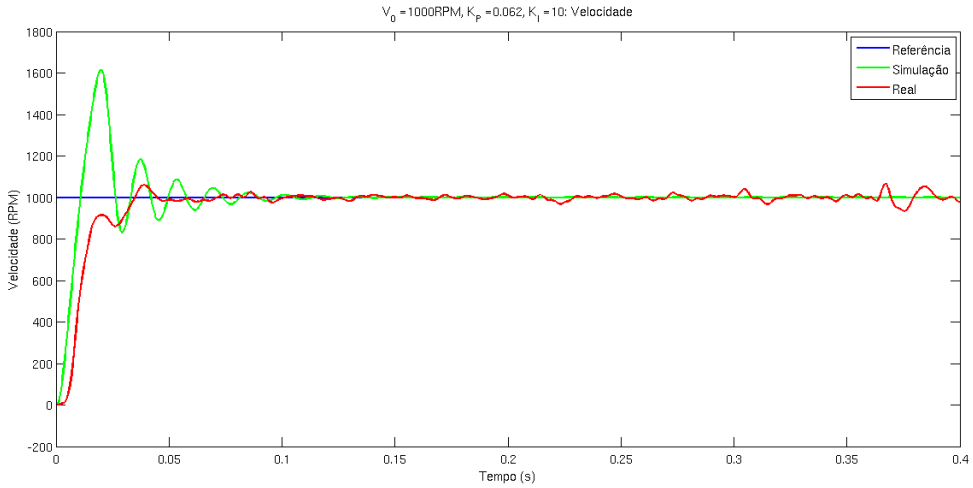

Figura 5.17: Comportamento da velocidade para o controlador 12: $K_P = 0,062$, $K_I = 10$.

A resposta obtida é estável, porém há diferença significativa entre a resposta real e a simulada. Isto leva a concluir que extender o conjunto de controladores estabilizantes ainda pode levar a controladores que estabilizem o sistema, mas cuja resposta a malha fechada poderá não corresponder às previsões.

Por fim, dentro do conjunto esperado de controladores estáveis, verificam-se os pares 1, 2, 6, 7, 11 e 14. A seguir, pode-se ver o comportamento simulado e experimental da velocidade do sistema a malha fechada para cada um destes casos, respectivamente nas Figuras 5.18, 5.19, 5.20, 5.21, 5.22 e 5.23.

O resultado de maior destaque é o referente ao controlador 6, o qual apresentou resposta oscilatória. Este é um resultado importante pois ressalta o cuidado que se deve ter ao escolher controladores dentro da região estabilizante, dependendo da aplicação desejada e do comportamento que se deseja para o sistema a malha fechada. Destacam-se também os controladores 1 e 2, que apresentaram resposta mais rápida nos experimentos físicos (em termos do tempo de acomodação). Em relação aos demais, as respostas são estáveis, com proximidade entre resultados simulados e experimentais.

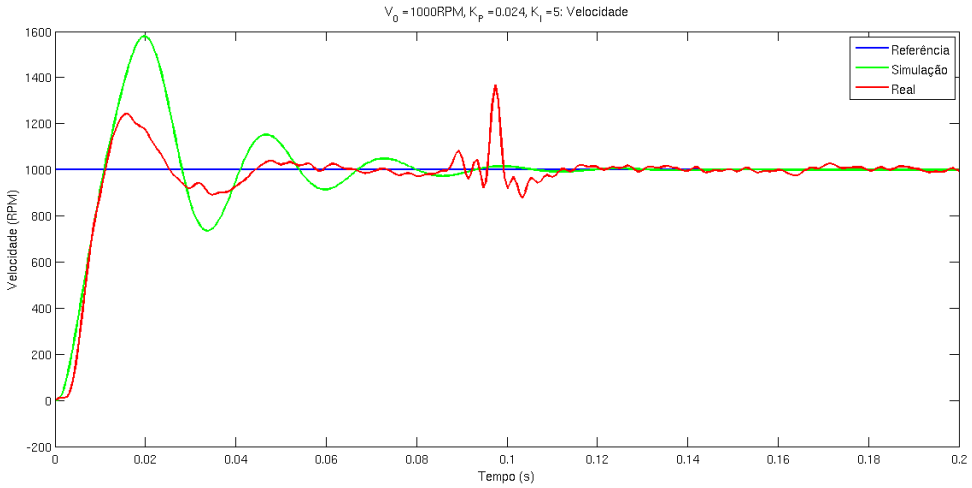

Figura 5.18: Comportamento da velocidade para o controlador 1: $K_P = 0,024$, $K_I = 5$.

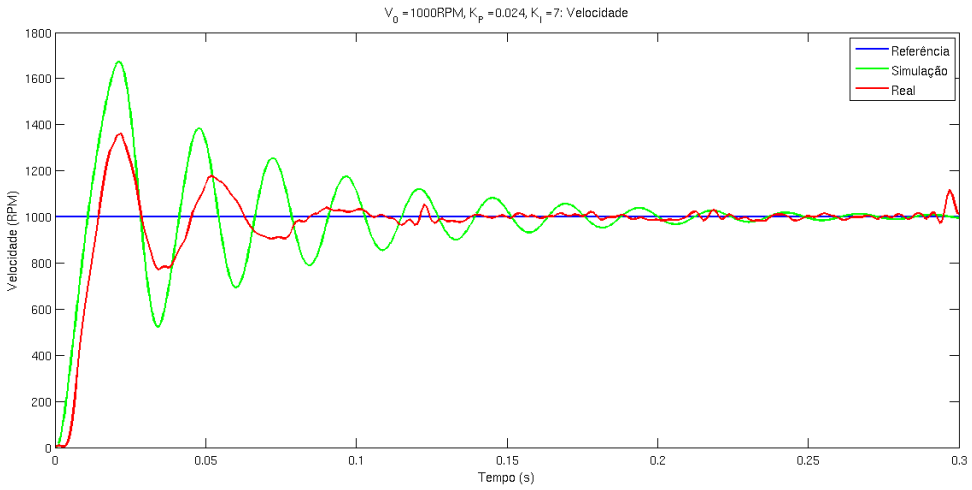

Figura 5.19: Comportamento da velocidade para o controlador 2: $K_P = 0,024$, $K_I = 7$.

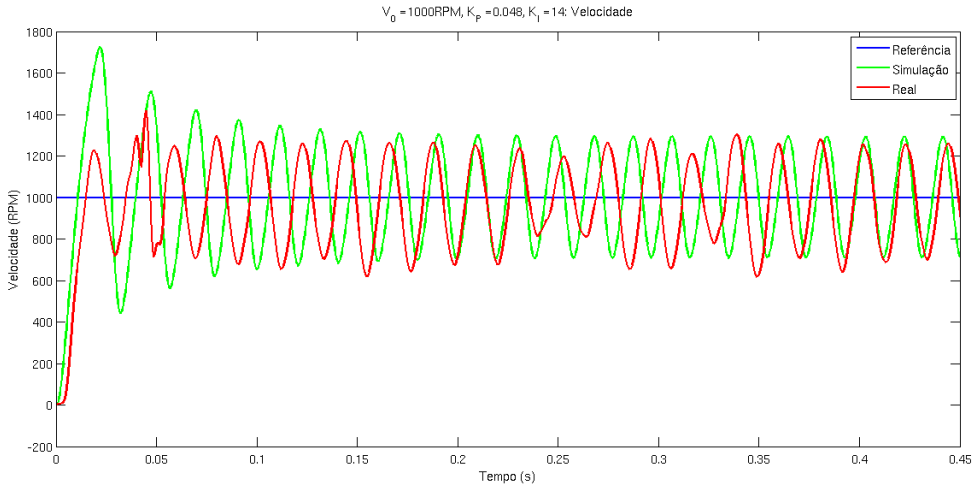

Figura 5.20: Comportamento da velocidade para o controlador 6: $K_P = 0,048$, $K_I = 14$.

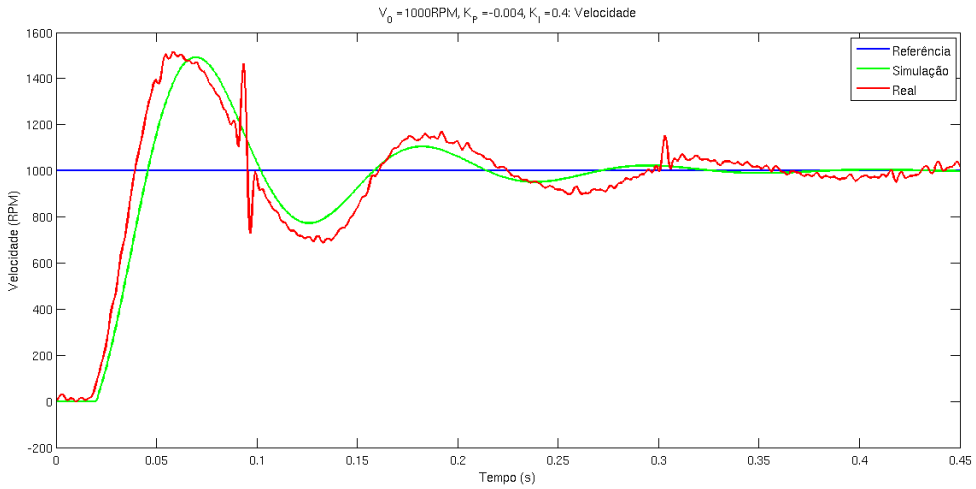

Figura 5.21: Comportamento da velocidade para o controlador 7: $K_P = -0,004$, $K_I = 0,4$.

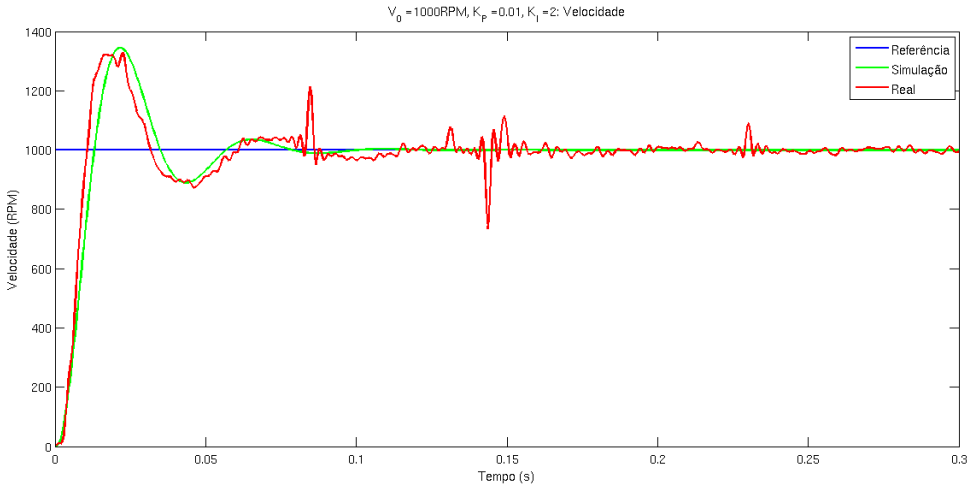

Figura 5.22: Comportamento da velocidade para o controlador 11: $K_P = 0,01$, $K_I = 2$.

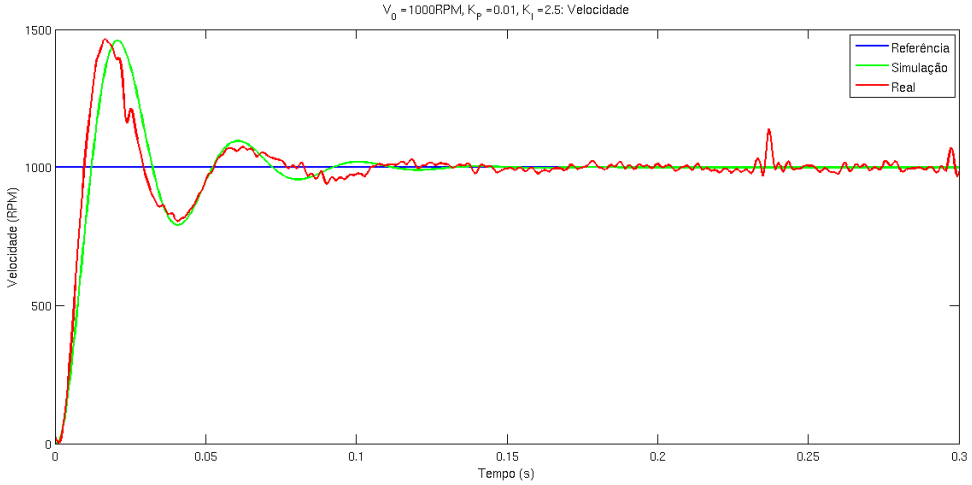

Figura 5.23: Comportamento da velocidade para o controlador 14: $K_P = 0,01$, $K_I = 2,5$.

6 CONCLUSÃO

O procedimento adotado para obtenção do conjunto de controladores estabilizantes mostrou-se eficaz, isto é, para o sistema analisado, os resultados experimentais são condizentes com as simulações realizadas - seja para a resposta em frequência como para a resposta ao degrau. Ressalta-se também a robustez do procedimento em relação às variações dos dados experimentais intrínsecas ao sistema de medição.

A bancada utilizada é composta por sistemas lineares muito difundidos na literatura, para os quais há facilidade de atribuição de modelos, destacando as máquinas de corrente contínua; a despeito disso, como houve forte correlação entre os dados simulados e experimentais, verificou-se a validade do algoritmo neste contexto. É possível, assim, expandir a aplicação dos conceitos utilizados neste projeto para sistemas mais complexos e que sejam de difícil modelagem, tais como sistemas não-lineares, sistemas de ordem elevada, entre outros.

Além da conformidade entre resultados simulados e experimentais, houve também correlação entre o conjunto de controladores estabilizantes obtido e as respostas ao degrau; apenas um dos controladores, cuja resposta esperada era estável, apresentou instabilidade frente a uma excitação degrau.

Sugere-se, como evolução desta linha de pesquisa, a averiguação de parâmetros de desempenho, tais como tempo de subida, sobressinal, etc, dentro das áreas de controladores estabilizantes. Também se propõe a utilização deste algoritmo em sistemas desconhecidos, ou de difícil descrição por modelos.

Espera-se que este projeto seja um ponto de partida para o aprofundamento da técnica aqui apresentada, que, apesar de ainda não muito difundida e estudada, mostrou-se concisa e eficiente.

REFERÊNCIAS

ANTONIOU, A. **Digital Filters: Analysis, Design, and Applications**. 2a. ed. Upper Saddle River, NJ, USA: McGraw-Hill Education, 2000.

_____. **Digital Signal Processing Signals, Systems & Filters**. 1a. ed. Upper Saddle River, NJ, USA: Tata McGraw Hill, 2006.

ASTROM, K.; HAGGLUND, T. **Advanced PID Control**. Research Triangle Park, NC, USA: The Instrumentation, Systems, and Automation Society - ISA, 2006.

BARBI, I. **Eletrônica de Potência**. 6a. ed. Florianópolis: Edição do Autor, 2006.

BOUWENS, A. J. **Digital Instrumentation**. 2a. ed. Upper Saddle River, NJ, USA: McGraw-Hill, 1987.

DEWAN, S. B. **Power Semiconductor Circuits**. 1a. ed. Ann Arbor, MI, USA: John Wiley & Sons, Inc, 1975.

DORF, R. C.; BISHOP, R. H. **Modern Control Systems**. 11a. ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2008.

FITZGERALD, A. E. **Máquinas Elétricas**. 1a. ed. São Paulo: McGraw-Hill's do Brasil, 1975.

FNAEICH, M. A. et al. A measurement-based approach for speed control of induction machines. **IEEE Journal of Emerging and Selected Topics in Power Electronics**, Vol. 2, n. n. 2, p. pp. 308–318, June 2014.

IOANNOU, P.; SUN, J. **Robust Adaptive Control**. 1a. ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

KEEL, L. H.; BHATTACHARYYA, S. P. Control synthesis free of analytical models: Three term controllers. **IEEE Transactions on Automatic Control**, v. 53, n. 6, July 2008.

KHADRAOUI, S. et al. Design of reduced-order controllers using a set of measurements: Application to a dc servomotor. In: AMERICAN CONTROL CONFERENCE (ACC), 2013. Washington, DC, USA, 2013. p. 4331–4336.

_____. Electrical machines, drives and power systems. In: AMERICAN CONTROL CONFERENCE (ACC), 2013. **A model-free design of reduced-order controllers and application to a DC servomotor**. Washington, DC, USA: Automatica 50, 2014. p. 2142–2149.

LARAMORE, R. D. **Electrical Machines and Transformers**. 2a. ed. New York, NY, USA: John Wiley & Sons, 1990.

NISE, N. S. **Engenharia de Sistemas de Controle**. 3a. ed. Rio de Janeiro: LTC Editora, 2002.

OLIVEIRA, V. A.; AGUIAR, M. L.; VARGAS, J. B. **Sistemas de controle: Aulas de laboratório**. 2a. ed. São Carlos: EESC/USP, 2013.

_____. **Engenharia de controle: Fundamentos e aulas de laboratório**. 1a. ed. Rio de Janeiro: Elsevier, 2016.

OPPENHEIM, A. V.; WILLSKY, A. S. **Signals and Systems**. 2a. ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.

SEN, P. C. **Principles of Electrical Machines and Power Electronics**. 2a. ed. New York, NY, USA: John Wiley & Sons, Inc, 1996.

Apêndices

APÊNDICE A – *SCRIPT* PARA O CÁLCULO DOS CONTROLADORES ESTABILIZANTES

A seguir encontra-se o *script* desenvolvido para uso no *software* MATLAB®com o intuito de calcular o conjunto de controladores PI estabilizantes para um processo cuja resposta em frequência a malha fechada é conhecida.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Codigo para determinar o conjunto de controladores PI que %
% estabilizam uma planta com resposta em frequencia contida %
% no vetor "diagramabode". Baseado nos passos sugeridos no  %
% artigo "Controller Synthesis Free of Analytical Models:   %
% Three Term Controllers", de Keel e Bhattacharyya.         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all

% Controlador utilizado
kp1 = 1/100;
ki1 = 2;
kd1 = 0;


tensaodc = 1200;
ampl = 100;

% PARA SIMULACAO
nomearq = strcat('simulacao_dadoplaca_',num2str(tensaodc),'_',num2str(ampl),'_',num2str(kp1),...
'_',num2str(ki1));
var1 = 3;

% PARA DADOS EXPERIMENTAIS
% nomearq = strcat('dadosexp_',num2str(tensaodc),'_',num2str(ampl),'_',num2str(kp1),...
%     '_',num2str(ki1));
% var1 = 6;

gc = tf([kp1 ki1], [1 0]);
[zerosc, polosc, constc] = zpkdata(gc,'v');

% Carregar os dados
load(strcat(nomearq,'.mat'));
% Tipo do vetor:
% Linha 1 -> frequencias (em Hz)
% Linha 2 -> ganho em dB
% Linha 3 -> fase em graus
tam_vetor = length(diagramabode(1,:));

% Passar frequencias para rad/s
diagramabode(1,:) = diagramabode(1,:)*2*pi;

% Passar ganho para valor absoluto
diagramabode(2,:) = 10.^(diagramabode(2,:)/20);
```

```
47
48 % Calcular P(jw) pela eq (17)
49 [magcont, fasecont, freqcont] = bode(gc,diagramabode(1,:));
50 g_jomega = diagramabode(2,:).*exp(1i*diagramabode(3,:)*pi/180);
51 c_jomega = magcont.*exp(1i*fasecont*pi/180); % magcont ja esta em valor absoluto
52 c_jomega = permute(c_jomega,[1,3,2]);
53 p_jomega = g_jomega./(c_jomega.*(ones(1,length(g_jomega))-g_jomega));
54 p_jomega_abs = abs(p_jomega);
55 p_jomega_fase = phase(p_jomega);
56
57 % Calcular grau da planta pela inclinacao do grafico em altas frequencias
58 eixox = diagramabode(1,tam_vetor-var1:tam_vetor);
59 data_log = 20*log10(p_jomega_abs(tam_vetor-var1 : tam_vetor));
60
61 rp = round(-(1/20)*diff(data_log)/diff(log10(eixox))+0.2);
62
63 % Determinar o no de zeros no SPCD e o grau do controlador
64 zcmais = sum(zerosc(:) > 0);
65 [magcont, fasecont, freqcont] = bode(gc);
66 tam_vetor_cont = length(magcont(1,1,:));
67 eixox = freqcont(tam_vetor_cont-8 : tam_vetor_cont);
68 data_log=20*log10(magcont(1,tam_vetor_cont-8 : tam_vetor_cont));
69 rc = round(-(1/20)*diff(data_log)/diff(log10(eixox))');
70
71 % Determinar a assinatura de G(jw)
72 delta_fase = pi/2*(round(min(diagramabode(3,:))/90) - round(max(diagramabode(3,:))/90));
73 sigma_g = (2/pi)*delta_fase;
74
75 % Calcular o numero de zeros no SPCD da planta
76 zmais = (-rp - rc - 2*zcmais - sigma_g)/2;
77
78 % Calcular a resposta g(w) (O QUE EH ISSO???)
79 % PARA T = 0 (constante de atraso para controlador diferencial)
80 g_omega = - cos(p_jomega_fase) ./ p_jomega_abs;
81
82
83 %%%%% PARTE 2 %%%%%
84
85 %% Calculo das frequencias que sao solucoes da Eq (27)
86 indice = 1;
87 for kp1 = g_omega(3:end-2)
88
89         tolerancia = abs(min(diff(g_omega))/10000);
90         wl_indices = find(abs(g_omega-kp1) < tolerancia);
91         wl = diagramabode(1,wl_indices);
92         diff_wl = diff(wl_indices);
93
94         %fase de P(jw) nas posicoes wl_indices
95         fase_final = p_jomega_fase(wl_indices);
96
97         %modulo de P(jw) nas posicoes wl_indices
98         p_jomega_abs_final = p_jomega_abs(wl_indices);
99
100
101         for pos = 1:length(wl_indices)-1
102                 if diff_wl(pos) == 1
103                         wl_menor = (g_omega(wl_indices(pos)));
104                         wl_maior = (g_omega(wl_indices(pos+1)));
105                         razao = (wl_maior - kp1)/(wl_maior - wl_menor);
106                         wl(pos) = wl(pos)*razao + wl(pos+1)*(1-razao);
```

```
107                     wl = [wl(1:pos) wl(pos+2:length(wl))];
108                     fase_final(pos) = fase_final(pos)*razao + fase_final(pos+1)*(1-
                            razao);
109                     fase_final = [fase_final(1:pos) fase_final(pos+2:length(
                            fase_final))];
110                     p_jomega_abs_final(pos) = p_jomega_abs_final(pos)*razao +
                            p_jomega_abs_final(pos+1)*(1-razao);
111                     p_jomega_abs_final = [p_jomega_abs_final(1:pos)
                            p_jomega_abs_final(pos+2:length(p_jomega_abs_final))];
112             end
113     end
114
115     % Calculo do conjunto de inteiros it
116     wl = [0 wl inf];
117     fase_final = [0 fase_final -rp*pi/2];
118     p_jomega_abs_final = [p_jomega_abs(1) p_jomega_abs_final 0];
119     elle = length(wl)-1; % elle = 2 pois temos w0, w1 e w2
120     it = ones(1,elle+1);
121     jota = sign(kp1*p_jomega_abs(tam_vetor)^2 + p_jomega_abs(tam_vetor)*cos(
            p_jomega_fase(tam_vetor)));
122     % tomando que sgn(Fi) p/ w->inf eh igual a sgn(Fi) p/ maior w usado
123     resultado = rp + 2*zmais + 2;
124     testeok = 0;
125     numerosolucao = 0;
126     vetorsinais = 1;
127     for pos = 1 : elle-1
128             vetorsinais = [vetorsinais 2*((-1)^(pos))]; % termina na posicao (elle-1)
129     end
130
131     if mod(rp,2) == 0
132             vetorsinais = [vetorsinais (-1)^(elle)];
133     end
134
135     matrizit = [];
136     % Calculo da matriz it
137     for tam = 1 : length(vetorsinais)
138             col1 = [];
139             for tamcol = 1 : 2^(tam-1)
140                     col1 = [col1 [-ones(2^length(vetorsinais)/(2^(tam)),1)' ones(2^
                            length(vetorsinais)/(2^(tam)),1)']];
141             end
142             matrizit = [matrizit col1'];
143     end
144     % Solucao do sistema
145     solucao = [];
146     for ind = 1 : length(matrizit)
147             if sum(matrizit(ind,:).*vetorsinais)*(-1)^(elle-1)*jot  a == resultado
148                     numerosolucao = numerosolucao + 1;
149                     solucao(numerosolucao,:) = matrizit(ind,:);
150             end
151     end
152
153     %% Calculo do conjunto estabilizante Ki
154     limites(1) = -inf;
155     limites(2) = inf;
156     for te = 1 : elle
157             switch wl(te)
158
159             case {Inf} % calcula apenas para valores plausiveis (diferentes de inf)
160
```

```
161                 otherwise
162                         ki_limite = -wl(te)*sin(fase_final(te))/((p_jomega_abs_final(te))
                                ^2);
163
164                         for pos = 1 : numerosolucao
165                                 switch solucao(pos,te) % estuda o sinal do it
166                                 case 1 % inequacao normal, ki > ki_limite
167                                         limites(1) = max([ki_limite limites(1)]);
168                                 case -1 % inverter sinal da desigualdade
169                                         limites(2) = min([ki_limite limites(2)]);
170                                 end
171                         end
172                 end
173         end
174         limitefinal(indice,:) = [limites kp1];
175         indice = indice+1;
176 end
177
178 % Plotar grafico
179 figure
180 area(limitefinal(1:end,3), limitefinal(1:end,2))
181 title('Regiao estabilizadora para o sistema','fontsize',12)
182 xlabel('K_P','fontsize',12)
183 ylabel('K_I','fontsize',12)
184 limitefinal
185 save(strcat('area_',nomearq,'.mat'),'limitefinal');
```

APÊNDICE B – *SCRIPT* PARA AUTOMATIZAÇÃO DE OBTENÇÃO DE DADOS DE SIMULAÇÃO

Neste apêndice encontra-se o código desenvolvido para MATLAB® para obtenção dos dados de resposta em frequência em ambiente de simulação.

```
1
2  %% Informacoes do teste
3
4  clear all
5
6  % DADOS DO MOTOR
7  kt = 0.0887; % Nm/A
8  ke = 0.00857; % V/(rpm)
9  ra = 1.13;
10 la = 3.3e-3;
11 b = 0.000133;
12 j = 1.963508585377e-04;
13
14 % Frequencia de amostragem: 3kHz
15 f_amost = 3000;
16
17 % ARQUIVO DE SIMULACAO
18 simarq = 'simulacao_controlador_certo.slx';
19
20 Eref = 24;
21
22 % TENSAO DE ENTRADA
23 tensaodc = 1200;
24 ampl = 100;
25
26 % GANHOS DO CONTROLADOR PI
27 kp1 = 0.01; % Como Eref = 40V, kp1<1/40
28 ki1 = 2;
29
30 freq1 = 0.8;
31 freq2 = [1:5 5.5:0.5:8 9:29];
32 freq3 = [30:2:50];
33 freq = [freq1 freq2 freq3];
34
35 %% Calculo da resposta em frequencia
36
37 for l = 1:length(freq1)
38         f = freq1(l)
39         texec = 4/f;
40         sim(simarq);
41         janela = length(entrada.Time);
42         inicio = find(entrada.Time>=1/f,1,'first');
43         tempo1 = entrada.Time(inicio:janela);
44         in1 = entrada.Data(inicio:janela);
45         out1 = velocidade.Data(inicio:janela);
46         ganho(l) = 20*log10((max(out1)-min(out1))/(max(in1)-min(in1)));
47         in1norm = 2*(in1 - media(in1,tempo1))/(max(in1)-min(in1));
48         out1norm = 2*(out1 - media(out1,tempo1))/(max(out1)-min(out1));
```

```
49          [lixo,fasek] = min(abs(out1norm));
50          fase(l) = 180*acos(2*mean(in1norm.*out1norm))/pi;
51          fase1(l) = -mod((velocidade.Time(fasek+inicio) - velocidade.Time(inicio))*360*f,
                180);
52          figure
53          plot(tempo1,out1,tempo1,in1+tensaodc)
54          title(strcat('Frequencia: ', num2str(f),'Hz'),'fontsize',12)
55          xlabel('Tempo (s)','fontsize',12)
56          ylabel('Velocidade (RPM)','fontsize',12)
57          legend({'Referencia','Saida'},'fontsize',12)
58  end
59
60  for l = length(freq1)+1 : length(freq1)+length(freq2)
61          f = freq(l)
62          texec = 28/f;
63          sim(simarq);
64          janela = length(entrada.Time);
65          inicio = find(entrada.Time>=24/f,1,'first');
66          tempo1 = entrada.Time(inicio:janela);
67          in1 = entrada.Data(inicio:janela);
68          out1 = velocidade.Data(inicio:janela);
69          ganho(l) = 20*log10((max(out1)-min(out1))/(max(in1)-min(in1)));
70          in1norm = 2*(in1 - media(in1,tempo1))/(max(in1)-min(in1));
71          out1norm = 2*(out1 - media(out1,tempo1))/(max(out1)-min(out1));
72          [lixo,fasek] = min(abs(out1norm));
73          fase(l) = 180*acos(2*mean(in1norm.*out1norm))/pi;
74          fase1(l) = -mod((velocidade.Time(fasek+inicio) - velocidade.Time(inicio))*360*f,
                180);
75          figure
76          plot(tempo1,out1,tempo1,in1+tensaodc)
77          title(strcat('Frequencia: ', num2str(f),'Hz'),'fontsize',12)
78          xlabel('Tempo (s)','fontsize',12)
79          ylabel('Velocidade (RPM)','fontsize',12)
80          legend({'Referencia','Saida'},'fontsize',12)
81  end
82
83  for l = length(freq1)+length(freq2)+1:length(freq1)+length(freq2)+length(freq3)
84          f = freq(l)
85          texec = 52/f;
86          sim(simarq);
87          janela = length(entrada.Time);
88          inicio = find(entrada.Time>=44/f,1,'first');
89          tempo1 = entrada.Time(inicio:janela);
90          in1 = entrada.Data(inicio:janela);
91          out1 = velocidade.Data(inicio:janela);
92          ganho(l) = 20*log10((max(out1)-min(out1))/(max(in1)-min(in1)));
93          in1norm = 2*(in1 - media(in1,tempo1))/(max(in1)-min(in1));
94          out1norm = 2*(out1 - media(out1,tempo1))/(max(out1)-min(out1));
95          [lixo,fasek] = min(abs(out1norm));
96          fase(l) = 180*acos(2*mean(in1norm.*out1norm))/pi;
97          fase1(l) = -mod((velocidade.Time(fasek+inicio) - velocidade.Time(inicio))*360*f,
                180);
98          figure
99          plot(tempo1,out1,tempo1,in1+tensaodc)
100         title(strcat('Frequencia: ', num2str(f),'Hz'),'fontsize',12)
101         xlabel('Tempo (s)','fontsize',12)
102         ylabel('Velocidade (RPM)','fontsize',12)
103         legend({'Referencia','Saida'},'fontsize',12)
104 end
105
```

```
106
107 figure
108
109 subplot(2,1,1)
110 semilogx(freq,ganho, '*k')
111 title('GANHO (dB)')
112 xlabel('Frequencia (Hz)')
113 ylabel('Magnitude')
114
115 subplot(2,1,2)
116 semilogx(freq,fase1, '*k')
117 title('FASE (graus)')
118 xlabel('Frequencia (Hz)')
119 ylabel('Fase')
120
121 % SALVAR DADOS
122
123 % Saida:
124 % 1- frequencia em Hz
125 % 2- ganho em dB
126 % 3- fase em graus
127
128 diagramabode = [freq; ganho; fase1];
129
130 nomearq = strcat('simulacao_',num2str(tensaodc),'_',num2str(ampl),'_',num2str(kp1),...
131 '_',num2str(ki1));
132 save(strcat(nomearq,'.mat'),'diagramabode');
133
134 % SALVAR DIAGRAMA DE BODE
135 savefig(strcat(nomearq,'.fig'));
```

APÊNDICE C – *SCRIPT* PARA OBTENÇÃO E COMPARAÇÃO DE RESPOSTAS AO DEGRAU

A seguir, o *script* desenvolvido para análise de dados experimentais de resposta ao degrau, bem como para simulação do sistema nas mesmas condições de controlador.

```
1
2  %% Informacoes do teste
3
4  clear all
5  % DADOS DO MOTOR
6  kt = 0.0887; % Nm/A
7  ke = 0.00857; % V/(rpm)
8  ra = 1.13;
9  la = 3.3e-3;
10 b = 0.000133;
11 j = 1.963508585377e-04;
12
13
14 % Frequencia de amostragem: 3kHz
15 f_amost = 3000;
16
17 %filtro = '';
18 filtro = '_filtro';
19
20 % ARQUIVO DE SIMULACAO
21 simarq = strcat('simulacao_controlador_certo',filtro,'.slx');
22
23 % FILTRO
24 d = designfilt('lowpassfir','FilterOrder',50,'CutoffFrequency',600, 'SampleRate',3000);
25
26 % TENSAO DE ENTRADA
27 tensaodc = 1000;
28 Ereffonte = 24;
29 Eref = 24;
30
31
32 %% Calculo da resposta em frequencia
33
34 contrs = [0.0018, 0.8];
35 % contrs = [0.01,2; 0.01,2.5; -0.004,0.4; 0.024,5; 0.024,7; 0.024,10;...
36 %     0.048,14; 0.062,10];
37
38 for vels = [1000]
39         for k = 1:length(contrs)
40                 kp1 = contrs(k,1);
41                 ki1 = contrs(k,2);
42                 tensaodc = vels;
43
44                 arquivo = strcat('degrau_',num2str(kp1),'_',num2str(ki1),'_',num2str(
                       tensaodc),filtro,'.dat');
45                 dados1 = load(arquivo);
46                 dados = [dados1(1:2:end)/1000 , dados1(2:2:end)/10];
47                 aquisicao = 3;
```

```
48
49                  contr = filtfilt(d,dados(:,1));
50                  velo = filtfilt(d,dados(:,2));
51                  tempo1 = linspace(0, length(contr)*aquisicao/3000, length(contr));
52
53                  %texec = 0.25;
54                  texec = 0.45;
55                  sim(simarq);
56
57                  figure
58                  plot(controle.Time,controle.Data);
59                  title(strcat('V_0 = ',num2str(tensaodc), 'RPM, K_P = ', num2str(kp1), ',
                        K_I = ', num2str(ki1), ': Acao de controle'),'fontsize',11)
60                  xlabel('Tempo (s)','fontsize',11)
61                  ylabel('Controle','fontsize',11)
62                  hold on
63                  plot(tempo1,contr,'r')
64                  legend({'Simulacao','Real'},'fontsize',11)
65
66                  figure
67                  plot(controle.Time,velocidade.Data);
68                  title(strcat('V_0 = ',num2str(tensaodc), 'RPM, K_P = ', num2str(kp1), ',
                        K_I = ', num2str(ki1), ': Velocidade'),'fontsize',11)
69                  xlabel('Tempo (s)','fontsize',11)
70                  ylabel('Velocidade (RPM)','fontsize',11)
71                  hold on
72                  plot(tempo1,velo,'r')
73                  legend({'Simulacao','Real'},'fontsize',11)
74
75                  figure
76                  plot(corrente.Time,corrente.Data);
77                  title(strcat('V_0 = ',num2str(tensaodc), 'RPM, K_P = ', num2str(kp1), ',
                        K_I = ', num2str(ki1), ': Corrente'),'fontsize',11)
78                  xlabel('Tempo (s)','fontsize',11)
79                  ylabel('Corrente (A)','fontsize',11)
80                  hold on
81                  a = load(strcat('cor_',num2str(kp1),'_',num2str(ki1),'_',num2str(tensaodc
                        ),'.csv'));
82                  plot(a(:,1),a(:,2),'r')
83                  legend({'Simulacao','Real'},'fontsize',11)
84          end
85  end
```

APÊNDICE D – CÓDIGO UTILIZADO NO MICROCONTROLADOR

Abaixo, o código utilizado para obtenção dos dados experimentais e testes de controle em malha fechada.

```
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/interrupt.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/pwm.h"
#include "driverlib/timer.h"
#include "driverlib/sysctl.h"
#include "drivers/rit128x96x4.h"
#include "driverlib/adc.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
****************** MALHA FECHADA DC  ******************

// Lista dos pinos de cada pwm
// PWM 0/ Pf0 - User LED
// PWM 1/ PG1 - Sound
// PWN 4/ PE0 - UP SWITCH
// PWM 5/ PE1 - DOWN SWITCH
// PWM 2 E 3 Tem pinos proprios ( PB0 E PB1)
// Controle PID
// Verificar qual o ganho ADC
//const float ref=353;

// Dados de experimentacao
const float fbode = 6;
float vo = 1000;

// Parametros do controlador
//const float kp = 5e-5;
//const float ki = 0.0015;
//const float kp = 0.0018;
//const float ki = 0.55;
//const float kp = 2.4e-2;
//const float ki = 9;
const float kp = 0.024;
const float ki = 7;



// Dados fisicos
short int vref = 24;
const int freq = 3000;


// Variaveis que mudam com a frequencia
int aquisicao = 1;
short int teste[1001][2];
```

```
51
52 // Variaveis auxiliares
53 float baixo = 0.002;
54 float alto = 0.998;
55 float razaofreq;
56 float alfa1 = 0;
57 float alfa2 = 0;
58 float contr[2] = {0,0};
59 float erro[2] = {0,0};
60 float vmedido = 0;
61 float vmedido0 = 0;
62 float vfiltrado = 0;
63 float vfiltrado0 = 0;
64 float vsenoidal = 0;
65 float vsennormal = 0;
66 float vsennormal0 = 0;
67 float vsenfilt = 0;
68 float vsenfilt0 = 0;
69 float periodo;
70 float base_tempo;
71 int timerint;
72 float dc;
73 float dcref;
74 int start = 0;
75 int amostra = 0;
76 float vsaida = 0;
77 unsigned long ADCvalue;
78 unsigned long vetor_amostra;
79 float ref;
80 volatile unsigned int cont;
81 int auxiliar = 0;
82 int auxrampa = 0;
83 int ver = 0;
84 int i;
85 int en = 0;
86 int contador = 0;
87 float constanteh = 0.04019347; // = (3/1024)*(1/Kdr)*hv = (cte do ADC)^(-1)*(cte do
       divisor resistivo)^(-1)*(cte de realimentacao)
88 char strdc[17] = "Valor DC:    ,  V";
89 char strfreq[17] = "Freq:    ,   Hz";
90 char aqui = 0;
91
92 #ifdef DEBUG
93 void
94 __error__(char *pcFilename, unsigned long ulLine)
95 {
96 }
97 #endif
98
99 // rotina de conversao de decimal para porcentagem
100 void converte() {
101 int aux;
102
103 // VALOR DC
104 aux = vo * 10;
105
106 char parte_dezena = aux / 100;
107 char parte_inteira = aux % 100 / 10;
108 char digito_unidade = (aux % 100) % 10;
109 strdc[10] = parte_dezena + 0x30;
```

```
110 strdc[11] = parte_inteira + 0x30;
111 strdc[13] = digito_unidade + 0x30;
112 }
113
114 void Display() {
115
116 // Clear the screen and tell the user what is happening.
117 //
118 RIT128x96x4Init(1000000);
119 RIT128x96x4StringDraw("  Ref. Senoidal", 18, 24, 15);
120 converte();
121 RIT128x96x4StringDraw(strdc, 18, 40, 15);
122
123 }
124
125 void Inicializa() {
126 IntMasterEnable();
127 SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM);
128 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
129 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
130 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);
131 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
132 SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
133 SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
134 SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);
135 SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
136 //
137 // Set the clocking to run directly from the crystal.
138 //
139 SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
140 SYSCTL_XTAL_8MHZ);
141 SysCtlPWMClockSet(SYSCTL_PWMDIV_1);
142
143 //
144 // Initialize the OLED display.
145 //
146 RIT128x96x4Init(1000000);
147 }
148
149 void InicializaTimer1(void) {
150
151 TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
152 TimerLoadSet(TIMER1_BASE, TIMER_A, base_tempo / 10);
153 //TimerLoadSet(TIMER0_BASE, TIMER_A, amostragem);
154 IntEnable(INT_TIMER1A);
155 //IntEnable(INT_TIMER0B);
156 TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
157 //TimerIntEnable(TIMER0_BASE, TIMER_TIMB_TIMEOUT);
158 }
159
160 void InicializaTimer2(void) {
161
162 TimerConfigure(TIMER2_BASE, TIMER_CFG_PERIODIC);
163
164 TimerLoadSet(TIMER2_BASE, TIMER_A, periodo);
165 IntEnable(INT_TIMER2A);
166 TimerIntEnable(TIMER2_BASE, TIMER_TIMA_TIMEOUT);
167
168 }
169
```

```
170 void InicializaADC(void) {
171
172 ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
173 ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_IE | ADC_CTL_END);
174 ADCSequenceEnable(ADC0_BASE, 3);
175 ADCIntEnable(ADC0_BASE, 3);
176 IntEnable(INT_ADC0SS3);
177
178 }
179
180 void InicializaLeft(void) {
181 // Define a porta PF1 como Entrada, isto e, "Botao Select" .
182 GPIOPinTypeGPIOInput(GPIO_PORTE_BASE, GPIO_PIN_2);
183 GPIOPadConfigSet(GPIO_PORTE_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA,
184 GPIO_PIN_TYPE_STD_WPU);
185 // Declara que a interrupcao ocorra por borda de descida
186
187 GPIOIntTypeSet(GPIO_PORTE_BASE, GPIO_PIN_2, GPIO_FALLING_EDGE);
188 GPIOPinIntEnable(GPIO_PORTE_BASE, GPIO_PIN_2);
189 IntEnable(INT_GPIOE);
190 }
191
192 void InicializaRigth(void) {
193 // Define a porta PF1 como Entrada, isto e, "Botao Select" .
194 GPIOPinTypeGPIOInput(GPIO_PORTE_BASE, GPIO_PIN_3);
195 GPIOPadConfigSet(GPIO_PORTE_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
196 GPIO_PIN_TYPE_STD_WPU);
197 // Declara que a interrupcao ocorra por borda de descida
198
199 GPIOIntTypeSet(GPIO_PORTE_BASE, GPIO_PIN_3, GPIO_FALLING_EDGE);
200 GPIOPinIntEnable(GPIO_PORTE_BASE, GPIO_PIN_3);
201 IntEnable(INT_GPIOE);
202 }
203
204 void InicializaSelect(void) {
205 // Define a porta PF1 como Entrada, isto e, "Botao Select" .
206 GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_1);
207 GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
208 GPIO_PIN_TYPE_STD_WPU);
209 // Declara que a interrupcao ocorra por borda de descida
210
211 GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_FALLING_EDGE);
212 //GPIOPinIntEnable(GPIO_PORTF_BASE, GPIO_PIN_1);
213 //IntEnable(INT_GPIOF);
214 }
215
216 void InicializaLED(void) {
217 // Define a porta PF2 como Saida, isto e, "LED1" .
218 GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);
219 }
220
221 void DutyCPWM() {
222
223 float tempo = periodo*dc;
224 float tfull = periodo*alto;
225 PWMPulseWidthSet(PWM_BASE, PWM_OUT_1, tfull);
226 PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, tempo);
227 PWMPulseWidthSet(PWM_BASE, PWM_OUT_3, tempo);
228 PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, tfull);
229 }
```

```
230
231 void DutyCPWMinv() {
232
233 float tempo = periodo*dc;
234 float tfull = periodo*baixo;
235 PWMPulseWidthSet(PWM_BASE, PWM_OUT_1, tfull);
236 PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, tempo);
237 PWMPulseWidthSet(PWM_BASE, PWM_OUT_3, tempo);
238 PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, tfull);
239 }
240
241 void DeadBand() {
242 // Dead-band de 0.1 us
243 // Deadband para os pwm's 2 e 3
244 PWMDeadBandEnable(PWM_BASE, PWM_GEN_1, SysCtlClockGet() * 100e-9,
245 SysCtlClockGet() * 100e-9);
246 // Deadband para os pwm's 0 e 1
247 PWMDeadBandEnable(PWM_BASE, PWM_GEN_0, SysCtlClockGet() * 100e-9,
248 SysCtlClockGet() * 100e-9);
249 }
250
251 void botaoIntHandler(void){
252
253 if(GPIOPinIntStatus(GPIO_PORTE_BASE, false) == 4) // ESQUERDA
254 {
255 en = 2; // SALVA VALORES NO VETOR
256 GPIOPinIntClear(GPIO_PORTE_BASE, GPIO_PIN_2);
257 }
258 else
259 {
260 en = 0; // PARA DE SALVAR
261 GPIOPinIntClear(GPIO_PORTE_BASE, GPIO_PIN_3);
262 i = 0;
263 }
264
265 }
266
267 void Timer1IntHandler(void) {
268 // Clear the timer interrupt.
269
270 TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
271
272 }
273
274
275 void Controle(void) {
276
277
278 vsaida = (vetor_amostra - ref) * constanteh * 112.53;
279 erro[1] = vsenoidal - vsaida;
280 contr[1] = contr[0] + alfa1 * erro[1] + alfa2 * erro[0];
281
282 if (contr[1] < -24) {
283 contr[1] = -24;
284 }
285 if (contr[1] > 24) {
286 contr[1] = 24;
287 }
288 dc = contr[1]/vref;
289 if (dc < -0.998) {
```

```
290 dc = -0.998;
291 }
292 if (dc > 0.998) {
293 dc = 0.998;
294 }
295
296 //contr[1] = dc * 12;
297
298 if (dc < 0) {
299 dc = 1+dc;
300 DutyCPWMinv();
301 }
302 else
303 {
304 DutyCPWM();
305 }
306 erro[0] = erro[1];
307 contr[0] = contr[1];
308 }
309
310 }
311
312 void Armazenamento(void) {
313 if (i < 1001 ) {
314 teste[i][0] = contr[1]*1000/vref;
315 teste[i][1] = vsaida*10;
316 i++;
317 }
318 }
319
320 void Leitura(void)
321 {
322 ADCProcessorTrigger(ADC0_BASE, 3);
323 vmedido = vetor_amostra;
324 vfiltrado = 0.2846*vfiltrado0 + 0.7154*vmedido0;
325 vetor_amostra = vfiltrado;
326 vfiltrado0 = vfiltrado;
327 vmedido0 = vmedido;
328 }
329
330 void Timer2IntHandler(void) {
331
332 // Clear the timer interrupt.
333 TimerIntClear(TIMER2_BASE, TIMER_TIMA_TIMEOUT);
334
335 //Calcula referencia senoidal EM VOLTS
336 vsennormal0 = vsennormal;
337 vsennormal = vo;
338 vsenfilt0 = vsenfilt;
339 vsenfilt = 0.2846*vsenfilt0 + 0.7154*vsennormal0;
340 vsenoidal = vsennormal;
341
342 Leitura();
343
344 Controle();
345
346
347 // PARA SALVAR, USAR VALOR FILTRADO
348 vsenoidal = vsenfilt;
349
```

```
350 en = 1;
351
352 if (en == 1)
353 {
354 contador++;
355 if (contador > aquisicao)
356 {
357 contador = 0;
358 Armazenamento();
359 }
360 }
361 }
362
363 void EstabeleceRef() {
364 ref = ref + vetor_amostra;
365 amostra++;
366 if (amostra == 100) {
367 PWMGenEnable(PWM_BASE, PWM_GEN_0);
368 PWMGenEnable(PWM_BASE, PWM_GEN_1);
369 ref = ref / amostra;
370 }
371 }
372
373 void ADC0Seq3IntHandler(void) {
374
375 ADCIntClear(ADC0_BASE, 3);
376
377 ADCSequenceDataGet(ADC0_BASE, 3, &ADCvalue);
378 vetor_amostra = ADCvalue;
379 }
380
381 void InicializaPWM() {
382 //
383 // Enable the peripherals used by this example.
384 // Cada pwmgen tem 2 sinais de saida
385 //SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
386
387 //
388 // Set GPIO F0 and G1 as PWM pins.  They are used to output the PWM0 and
389 // PWM1 signals.
390 //
391 GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_0);
392 GPIOPinTypePWM(GPIO_PORTG_BASE, GPIO_PIN_1);
393 GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_0);
394 GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_1);
395 GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_0);
396 GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_1);
397 PWMGenConfigure(PWM_BASE, PWM_GEN_0,
398 PWM_GEN_MODE_DOWN | PWM_GEN_MODE_GEN_SYNC_GLOBAL);
399 PWMGenConfigure(PWM_BASE, PWM_GEN_1,
400 PWM_GEN_MODE_DOWN | PWM_GEN_MODE_GEN_SYNC_GLOBAL);
401 PWMGenConfigure(PWM_BASE, PWM_GEN_2,
402 PWM_GEN_MODE_DOWN | PWM_GEN_MODE_SYNC);
403 PWMGenPeriodSet(PWM_BASE, PWM_GEN_0, periodo);
404 PWMGenPeriodSet(PWM_BASE, PWM_GEN_1, periodo);
405 PWMGenPeriodSet(PWM_BASE, PWM_GEN_2, periodo);
406 //Algumas saidas respondem diferente a dc 0 ou 1. Podem ser invertidos
407 // Evitar chavear as portas 4 e 5 do pwm para que nao bugue os botoes
408 // O essencial e utilizar as portas 0,1,2 e 3 para que a funcao deadband possa ser
      utilizada nos 2 pares e facilite a inversao
```

```
409 DutyCPWM();
410
411 PWMOutputState(PWM_BASE,
412 PWM_OUT_0_BIT | PWM_OUT_1_BIT | PWM_OUT_2_BIT | PWM_OUT_3_BIT,
413 true);
414
415 DeadBand();
416 }
417 void Acionamento(void) {
418
419 // Calcular pelo ganho do sensor e do ADC
420 RIT128x96x4Init(1000000);
421 RIT128x96x4StringDraw("Acionando...", 18, 24, 15);
422
423 //Definir tempo de espera para o acionamento
424 while (auxrampa < 10)
425 {
426 if (TimerIntStatus(TIMER1_BASE, false))
427 {
428 aqui++;
429 if (amostra < 100)
430 {
431 Leitura();
432 EstabeleceRef();
433 }
434 else
435 {
436 if (ver < 10)
437 ver++;
438 else
439 {
440 ver = 0;
441 if (auxrampa < 11)
442 {
443 auxrampa++;
444 dc = (float) auxrampa / 100;
445 }
446 else
447 {
448 auxrampa = 11;
449 dc = 0.51;
450 }
451 }
452 }
453 }
454 }
455
456 // ESCREVE LED
457 GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0xF4);
458 }
459
460 int main(void) {
461
462 for (i = 0; i < 1001; i++) {
463 teste[i][0] = 0;
464 teste[i][1] = 0;
465 }
466
467 // Passando de velocidade para tensao
468
```

```
469 alfa1 = kp;
470 alfa2 = (ki / freq - kp);
471
472 baixo = 0.002;
473 alto = 0.998;
474
475 razaofreq = freq/fbode;
476
477 contador = aquisicao; // Para ler o primeiro valor (seno comecando em 0)
478
479 i = 0;
480 dcref = (vo / vref / 112.53);
481 dc = 0.01;
482 auxrampa = 1;
483 base_tempo = SysCtlClockGet() * 0.05;
484
485 periodo = SysCtlClockGet() / freq;
486
487 Inicializa();
488 InicializaSelect();
489 InicializaLED();
490
491 InicializaPWM();
492 InicializaADC();
493 InicializaTimer1();
494 InicializaTimer2();
495
496 TimerEnable(TIMER1_BASE, TIMER_A);
497 Acionamento();
498
499
500 Display();
501 InicializaRigth();
502 InicializaLeft();
503
504 while (GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_1) == 2) {}
505
506 TimerEnable(TIMER2_BASE, TIMER_A);
507 TimerDisable(TIMER1_BASE, TIMER_A);
508
509 while (1) {
510 IntEnable(INT_GPIOE);
511 while (GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_1) == 2) {}
512 GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0xF4);
513 {
514 dc = 0.002;
515 TimerDisable(TIMER2_BASE, TIMER_A);
516 DutyCPWM();
517 }
518 else
519 TimerEnable(TIMER2_BASE, TIMER_A);*/
520 IntDisable(INT_GPIOE);
521 }
522 }
```

APÊNDICE E – CÓDIGO UTILIZADO NO MICROCONTROLADOR - SENOIDAL

Abaixo, o código utilizado para obtenção dos dados experimentais para as respostas em frequência

```
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/interrupt.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/pwm.h"
#include "driverlib/timer.h"
#include "driverlib/sysctl.h"
#include "drivers/rit128x96x4.h"
#include "driverlib/adc.h"
#include <math.h>
#include <stdio.h>
#include <string.h>
// ******************  DC SENOIDAL ********************

// Lista dos pinos de cada pwm
// PWM 0/ Pf0 - User LED
// PWM 1/ PG1 - Sound
// PWN 4/ PE0 - UP SWITCH
// PWM 5/ PE1 - DOWN SWITCH
// PWM 2 E 3 Tem pinos proprios ( PB0 E PB1)
// Controle PID
// Verificar qual o ganho ADC
//const float ref=353;

// Dados de experimentacao
float vo = 1200;
int ampl = 100;

// Parametros do controlador
//const float kp = 5e-5;
//const float ki = 0.0015;
//const float kp = 0.0018;
//const float ki = 0.55;
//const float kp = 2.4e-2;
//const float ki = 9;
const float kp = 1e-2;
const float ki = 2;


// Dados fisicos
short int vref = 24;
const int freq = 3000;


// Variaveis que mudam com a frequencia
const float fbode = 50;
```

```
49  float seno[60];
50  int tamanho = 60;
51  float cte = (2 * 3.14159265359) / 3000;
52  int aquisicao = 1;
53  short int teste[1001][2];
54
55  // Variaveis auxiliares
56  loat baixo = 0.002;
57  float alto = 0.998;
58  float razaofreq;
59  float alfa1 = 0;
60  float alfa2 = 0;
61  float contr[2] = {0,0};
62  float erro[2] = {0,0};
63  float vmedido = 0;
64  float vmedido0 = 0;
65  float vfiltrado = 0;
66  float vfiltrado0 = 0;
67  float vsenoidal = 0;
68  float vsennormal = 0;
69  float vsennormal0 = 0;
70  float vsenfilt = 0;
71  float vsenfilt0 = 0;
72  float periodo;
73  float base_tempo;
74  int timerint;
75  float dc;
76  float dcref;
77  int start = 0;
78  int amostra = 0;
79  float vsaida = 0;
80  unsigned long ADCvalue;
81  unsigned long vetor_amostra;
82  float ref;
83  volatile unsigned int cont;
84  int auxiliar = 0;
85  int auxrampa = 0;
86  int ver = 0;
87  int i;
88  int k = 0;
89  int en = 0;
90  int contador = 0;
91  float constanteh = 0.04019347; // = (3/1024)*(1/Kdr)*hv = (cte do ADC)^(-1)*(cte do
        divisor resistivo)^(-1)*(cte de realimentacao)
92  char strdc[17] = "Valor DC:    ,  V";
93  char strfreq[17] = "Freq:    ,   Hz";
94  char aqui = 0;
95
96  #ifdef DEBUG
97  void
98  __error__(char *pcFilename, unsigned long ulLine)
99  {
100 }
101 #endif
102
103 // rotina de conversao de decimal para porcentagem
104 void converte() {
105 int aux;
106
107 // VALOR DC
```

```
108 aux = vo * 10;
109
110 char parte_dezena = aux / 100;
111 char parte_inteira = aux % 100 / 10;
112 char digito_unidade = (aux % 100) % 10;
113 strdc[10] = parte_dezena + 0x30;
114 strdc[11] = parte_inteira + 0x30;
115 strdc[13] = digito_unidade + 0x30;
116 }
117
118 void Display() {
119
120 // Clear the screen and tell the user what is happening.
121 //
122 RIT128x96x4Init(1000000);
123 RIT128x96x4StringDraw("  Ref. Senoidal", 18, 24, 15);
124 converte();
125 RIT128x96x4StringDraw(strdc, 18, 40, 15);
126
127 }
128
129 void Inicializa() {
130 IntMasterEnable();
131 SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM);
132 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
133 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
134 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);
135 SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
136 SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
137 SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
138 SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);
139 SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
140 //
141 // Set the clocking to run directly from the crystal.
142 //
143 SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
144 SYSCTL_XTAL_8MHZ);
145 SysCtlPWMClockSet(SYSCTL_PWMDIV_1);
146
147 //
148 // Initialize the OLED display.
149 //
150 RIT128x96x4Init(1000000);
151 }
152
153 void InicializaTimer1(void) {
154
155 TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
156 TimerLoadSet(TIMER1_BASE, TIMER_A, base_tempo / 10);
157 IntEnable(INT_TIMER1A);
158 TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
159 }
160
161 void InicializaTimer2(void) {
162
163 TimerConfigure(TIMER2_BASE, TIMER_CFG_PERIODIC);
164
165 TimerLoadSet(TIMER2_BASE, TIMER_A, periodo);
166 IntEnable(INT_TIMER2A);
167 TimerIntEnable(TIMER2_BASE, TIMER_TIMA_TIMEOUT);
```

```
168
169 }
170
171 void InicializaADC(void) {
172
173 ADCSequenceConfigure(ADC0_BASE, 3, ADC_TRIGGER_PROCESSOR, 0);
174 ADCSequenceStepConfigure(ADC0_BASE, 3, 0, ADC_CTL_IE | ADC_CTL_END);
175 ADCSequenceEnable(ADC0_BASE, 3);
176 ADCIntEnable(ADC0_BASE, 3);
177 IntEnable(INT_ADC0SS3);
178
179 }
180
181 void InicializaLeft(void) {
182 // Define a porta PF1 como Entrada, isto e, "Botao Select" .
183 GPIOPinTypeGPIOInput(GPIO_PORTE_BASE, GPIO_PIN_2);
184 GPIOPadConfigSet(GPIO_PORTE_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA,
185 GPIO_PIN_TYPE_STD_WPU);
186 // Declara que a interrupcao ocorra por borda de descida
187
188 GPIOIntTypeSet(GPIO_PORTE_BASE, GPIO_PIN_2, GPIO_FALLING_EDGE);
189 GPIOPinIntEnable(GPIO_PORTE_BASE, GPIO_PIN_2);
190 IntEnable(INT_GPIOE);
191 }
192
193 void InicializaRigth(void) {
194 // Define a porta PF1 como Entrada, isto e, "Botao Select" .
195 GPIOPinTypeGPIOInput(GPIO_PORTE_BASE, GPIO_PIN_3);
196 GPIOPadConfigSet(GPIO_PORTE_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
197 GPIO_PIN_TYPE_STD_WPU);
198 // Declara que a interrupcao ocorra por borda de descida
199
200 GPIOIntTypeSet(GPIO_PORTE_BASE, GPIO_PIN_3, GPIO_FALLING_EDGE);
201 GPIOPinIntEnable(GPIO_PORTE_BASE, GPIO_PIN_3);
202 IntEnable(INT_GPIOE);
203 }
204
205 void InicializaSelect(void) {
206 // Define a porta PF1 como Entrada, isto e, "Botao Select" .
207 GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_1);
208 GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
209 GPIO_PIN_TYPE_STD_WPU);
210 // Declara que a interrupcao ocorra por borda de descida
211
212 GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_FALLING_EDGE);
213 }
214
215 void InicializaLED(void) {
216 // Define a porta PF2 como Saida, isto e, "LED1" .
217 GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);
218 // Declara que a interrupcao ocorra por borda de descida
219 }
220
221 void DutyCPWM() {
222
223 float tempo = periodo*dc;
224 float tfull = periodo*alto;
225 PWMPulseWidthSet(PWM_BASE, PWM_OUT_1, tfull);
226 PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, tempo);
227 PWMPulseWidthSet(PWM_BASE, PWM_OUT_3, tempo);
```

```
228 PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, tfull);
229 }
230
231 void DutyCPWMinv() {
232
233 float tempo = periodo*dc;
234 float tfull = periodo*baixo;
235 PWMPulseWidthSet(PWM_BASE, PWM_OUT_1, tfull);
236 PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, tempo);
237 PWMPulseWidthSet(PWM_BASE, PWM_OUT_3, tempo);
238 PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, tfull);
239 }
240
241 void DeadBand() {
242 // Dead-band de 0.1 us
243 // Deadband para os pwm's 2 e 3
244 PWMDeadBandEnable(PWM_BASE, PWM_GEN_1, SysCtlClockGet() * 100e-9,
245 SysCtlClockGet() * 100e-9);
246 // Deadband para os pwm's 0 e 1
247 PWMDeadBandEnable(PWM_BASE, PWM_GEN_0, SysCtlClockGet() * 100e-9,
248 SysCtlClockGet() * 100e-9);
249 }
250
251 void botaoIntHandler(void){
252
253 if(GPIOPinIntStatus(GPIO_PORTE_BASE, false) == 4) // ESQUERDA
254 {
255 en = 2; // SALVA VALORES NO VETOR
256 GPIOPinIntClear(GPIO_PORTE_BASE, GPIO_PIN_2);
257 }
258 else
259 {
260 en = 0; // PARA DE SALVAR
261 GPIOPinIntClear(GPIO_PORTE_BASE, GPIO_PIN_3);
262 i = 0;
263 }
264
265 }
266
267 void Timer1IntHandler(void) {
268 // Clear the timer interrupt.
269
270 TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
271
272 }
273
274
275 void Controle(void) {
276
277 vsaida = (vetor_amostra - ref) * constanteh * 112.53;
278 erro[1] = vsenoidal - vsaida;
279 contr[1] = contr[0] + alfa1 * erro[1] + alfa2 * erro[0];
280 if (contr[1] < -24) {
281 contr[1] = -24;
282 }
283 if (contr[1] > 24) {
284 contr[1] = 24;
285 }
286 dc = contr[1]/vref;
287 if (dc < -0.998) {
```

```
288 dc = -0.998;
289 }
290 if (dc > 0.998) {
291 dc = 0.998;
292 }
293
294
295 if (dc < 0) {
296 dc = 1+dc;
297 DutyCPWMinv();
298 }
299 else
300 {
301 DutyCPWM();
302 }
303 erro[0] = erro[1];
304 contr[0] = contr[1];
305 }
306
307 void ControleMA(void) {
308
309 dc =  vsenoidal / vref / 112.53;
310 //dc = 0.5;
311 if (dc < -1) {
312 dc = -1;
313 }
314 if (dc > 1) {
315 dc = 1;
316 }
317 //dc = (dc + 1) * 0.5;
318 DutyCPWM();
319 //DutyCPWMinv();
320 }
321
322 void Armazenamento(void) {
323 if (i < 1001 ) {
324 teste[i][0] = vsenoidal*10;
325 teste[i][1] = vsaida*10;
326 i++;
327 }
328 }
329
330 void Leitura(void)
331 {
332 ADCProcessorTrigger(ADC0_BASE, 3);
333 vmedido = vetor_amostra;
334 vfiltrado = 0.2846*vfiltrado0 + 0.7154*vmedido0;
335 vetor_amostra = vfiltrado;
336 vfiltrado0 = vfiltrado;
337 vmedido0 = vmedido;
338 }
339
340 void Timer2IntHandler(void) {
341
342 // Clear the timer interrupt.
343 TimerIntClear(TIMER2_BASE, TIMER_TIMA_TIMEOUT);
344 vsennormal0 = vsennormal;
345 vsennormal = vo + seno[k];
346 vsenfilt0 = vsenfilt;
347
```

```
348 vsenfilt = 0.2846*vsenfilt0 + 0.7154*vsennormal0;
349 // PARA CALCULOS, USAR VALOR SEM SER FILTRADO
350 vsenoidal = vsennormal;
351
352 Leitura();
353
354 Controle();
355
356 // PARA SALVAR, USAR VALOR FILTRADO
357 vsenoidal = vsenfilt;
358
359 k++;
360 if (k >= tamanho) {
361 k = 0;
362 if (en == 2){
363 en = 1;
364 }
365 }
366
367 if (en == 1)
368 {
369 contador++;
370 if (contador > aquisicao)
371 {
372 contador = 0;
373 Armazenamento();
374 }
375 }
376 }
377
378 void EstabeleceRef() {
379 ref = ref + vetor_amostra;
380 amostra++;
381 if (amostra == 100) {
382 PWMGenEnable(PWM_BASE, PWM_GEN_0);
383 PWMGenEnable(PWM_BASE, PWM_GEN_1);
384 ref = ref / amostra;
385 }
386 }
387
388 void ADC0Seq3IntHandler(void) {
389
390 ADCIntClear(ADC0_BASE, 3);
391
392 ADCSequenceDataGet(ADC0_BASE, 3, &ADCvalue);
393 vetor_amostra = ADCvalue;
394 }
395
396 void InicializaPWM() {
397 //
398 // Enable the peripherals used by this example.
399 // Cada pwmgen tem 2 sinais de saida
400 //SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
401
402 //
403 // Set GPIO F0 and G1 as PWM pins. They are used to output the PWM0 and
404 // PWM1 signals.
405 //
406 GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_0);
407 GPIOPinTypePWM(GPIO_PORTG_BASE, GPIO_PIN_1);
```

```
408 GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_0);
409 GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_1);
410 GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_0);
411 GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_1);
412 PWMGenConfigure(PWM_BASE, PWM_GEN_0,
413 PWM_GEN_MODE_DOWN | PWM_GEN_MODE_GEN_SYNC_GLOBAL);
414 PWMGenConfigure(PWM_BASE, PWM_GEN_1,
415 PWM_GEN_MODE_DOWN | PWM_GEN_MODE_GEN_SYNC_GLOBAL);
416 PWMGenConfigure(PWM_BASE, PWM_GEN_2,
417 PWM_GEN_MODE_DOWN | PWM_GEN_MODE_SYNC);
418 PWMGenPeriodSet(PWM_BASE, PWM_GEN_0, periodo);
419 PWMGenPeriodSet(PWM_BASE, PWM_GEN_1, periodo);
420 PWMGenPeriodSet(PWM_BASE, PWM_GEN_2, periodo);
421 //Algumas saidas respondem diferente a dc 0 ou 1. Podem ser invertidos
422 // Evitar chavear as portas 4 e 5 do pwm para que nao bugue os botoes
423 // O essencial e utilizar as portas 0,1,2 e 3 para que a funcao deadband possa ser
      utilizada nos 2 pares e facilite a inversao
424 /*
425 PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, periodo * dc);
426 PWMPulseWidthSet(PWM_BASE, PWM_OUT_1, periodo * dc);
427 PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, periodo * dc);
428 PWMPulseWidthSet(PWM_BASE, PWM_OUT_3, periodo * dc);
429 */
430 DutyCPWM();
431
432 PWMOutputState(PWM_BASE,
433 PWM_OUT_0_BIT | PWM_OUT_1_BIT | PWM_OUT_2_BIT | PWM_OUT_3_BIT,
434 true);
435
436
437 DeadBand();
438 }
439 void Acionamento(void) {
440
441 // Calcular pelo ganho do sensor e do ADC
442 RIT128x96x4Init(1000000);
443 RIT128x96x4StringDraw("Acionando...", 18, 24, 15);
444
445 //Definir tempo de espera para o acionamento
446 //timerint=TimerIntStatus(TIMER1_BASE,false);
447 while (auxrampa < 10)
448 {
449 if (TimerIntStatus(TIMER1_BASE, false))
450 {
451 aqui++;
452 if (amostra < 100)
453 {
454 Leitura();
455 EstabeleceRef();
456 }
457 else
458 {
459 if (ver < 10)
460 ver++;
461 else
462 {
463 ver = 0;
464 if (auxrampa < 11)
465 {
466 auxrampa++;
```

```
467 dc = (float) auxrampa / 100;
468 }
469 else
470 {
471 auxrampa = 11;
472 dc = 0.51;
473 }
474 }
475 }
476 }
477 }
478
479 // ESCREVE LED
480 GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0xF4);
481 }
482
483 void Desliga(void)
484 {
485 TimerEnable(TIMER1_BASE, TIMER_A);
486 RIT128x96x4Init(1000000);
487 RIT128x96x4StringDraw("Desligando...", 18, 24, 15);
488
489 //Definir tempo de espera para o acionamento
490 timerint = TimerIntStatus(TIMER1_BASE, false);
491 if (dc > 0.5) {
492 while (dc > 0.5) {
493 if (TimerIntStatus(TIMER1_BASE, false)) {
494 if (dc > 0.51) {
495 dc = dc - 0.01;
496 } else
497 dc = 0.5;
498
499 DutyCPWM();
500 }
501 }
502 } else {
503 while (dc < 0.5) {
504 if (TimerIntStatus(TIMER1_BASE, false)) {
505 if (dc < 0.49)
506 dc = dc + 0.01;
507 } else
508 dc = 0.5;
509
510 DutyCPWM();
511 }
512 }
513
514 TimerDisable(TIMER1_BASE, TIMER_A);
515 }
516
517 int main(void) {
518
519 for (i = 0; i < 1001; i++) {
520 teste[i][0] = 0;
521 teste[i][1] = 0;
522 }
523
524 for (k = 0; k < tamanho; k++)
525 {
526 seno[k] = ampl*sin(fbode * cte * k);
```

```
527 }
528
529 k = 0;
530
531 alfa1 = kp;
532 alfa2 = (ki / freq - kp);
533
534 baixo = 0.002;
535 alto = 0.998;
536
537 razaofreq = freq/fbode;
538
539 contador = aquisicao; // Para ler o primeiro valor (seno comecando em 0)
540
541 i = 0;
542 //dcref = ((vo / vref / 112.53) + 1) / 2;
543 dcref = (vo / vref / 112.53);
544 //str="DC = 0.    ";
545 //start=0;
546 dc = 0.01;
547 auxrampa = 1;
548 base_tempo = SysCtlClockGet() * 0.05;
549
550 periodo = SysCtlClockGet() / freq;
551
552 Inicializa();
553 InicializaSelect();
554 InicializaLED();
555
556 InicializaPWM();
557 InicializaADC();
558 InicializaTimer1();
559 InicializaTimer2();
560
561 TimerEnable(TIMER1_BASE, TIMER_A);
562 Acionamento();
563
564 dc = 0.01;
565
566 Display();
567 InicializaRigth();
568 InicializaLeft();
569
570 TimerEnable(TIMER2_BASE, TIMER_A);
571 TimerDisable(TIMER1_BASE, TIMER_A);
572
573 while (1) {
574 IntEnable(INT_GPIOE);
575 while (GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_1) == 2) {}
576 GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0xF4);
577 /*if (dc >= 0.1)
578 {
579 dc = 0.002;
580 TimerDisable(TIMER2_BASE, TIMER_A);
581 DutyCPWM();
582 }
583 else
584 TimerEnable(TIMER2_BASE, TIMER_A);*/
585 IntDisable(INT_GPIOE);
586 }
```

```
587 }
```