UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÂO E EDUCAÇÃO PROFISSIONAL
DEPARTAMENTO ACADÊMICO DE ELÉTRICA
ENGENHARIA ELÉTRICA

VINICIUS GOMES DE OLIVEIRA ANDRADE

CONTROLADOR FUZZY DE BAIXO CUSTO APLICADO A UM CONVERSOR BUCK

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2019

VINICIUS GOMES DE OLIVEIRA ANDRADE

CONTROLADOR FUZZY DE BAIXO CUSTO APLICADO A UM CONVERSOR BUCK

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina TCC2, do curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. André Sanches Fonseca Sobrinho

CORNÉLIO PROCÓPIO
2019

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia Elétrica

FOLHA DE APROVAÇÃO

Vinicius Gomes de Oliveira Andrade

Controlador fuzzy de baixo custo aplicado a um conversor buck

Trabalho de conclusão de curso apresentado às 13:50hs do dia 10/06/2019 como requisito parcial para a obtenção do título de Engenheiro Eletricista no programa de Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

_____

Prof(a). Dr(a). André Sanches Fonseca Sobrinho - Presidente (Orientador)

_____

Prof(a). Dr(a). Luis Fernando Caparroz Duarte - (Membro)

_____

Prof(a). Dr(a). Márcio Mendonça - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

Dedico este trabalho à minha mãe, meu pai e meu irmão.

AGRADECIMENTOS

Minha eterna gratidão a todos aqueles que me acompanharam, que de alguma forma me ajudaram e influenciaram para que eu conseguisse chegar até aqui. Sei que nada disso seria possível sem algumas pessoas muito especiais.

Agradeço ao meu orientador Prof. Dr. André Sanches, pela sabedoria, atenção e paciência, nada disso também seria possível, sem a dedicação, com que me guiou nesta trajetória.

Agradeço imensamente à todos os professores e professoras que fizeram parte de minha formação até o presente momento, pois além de serem responsáveis pelo conhecimento que adquiri, são também pessoas pelas quais tenho imensa admiração e orgulho, pelo privilégio de conhecer e ter sido aluno, muitos foram um grande incentivo para não desistir.

Muito obrigado à Universidade Tecnológica Federal do Paraná, por ter me acolhido e guiado meus caminhos. A toda comunidade que compõe essa grande Universidade, todos servidores que brilhantemente fazem da UTFPR um admirável exemplo de instituição.

Agradeço ao governo que facilitou o acesso ao ensino superior, que ampliou a rede de ensino gratuito de qualidade, sem esse incentivo, seria muito mais difícil.

Aos meus pais, Alisson e Maria do Carmo, agradeço a vida, criação, amor e educação. Mãe, muito obrigado por tudo, mas principalmente pelo esforço que sempre fez, para me dar uma boa educação, quando eu nem tinha noção de como isso era importante. Pai, muito obrigado por tudo também, mas principalmente pelo suporte e incentivo, por nunca ter deixado eu desistir. Agradeço imensamente por ter o privilégio de ser filho de vocês, tenho dois excelentes exemplos de seres humanos para me espelhar e sou muito grato por isso. Amo muito vocês!

Ao meu irmão, Augusto, você também é um grande orgulho e sou imensamente grato por tê-lo como irmão, é um grande amigo e parceiro, é em quem sempre irei confiar e contar, te amo!

Aos meus colegas de sala ou de percurso, que foram muitos, gostaria de citar todos os nomes importantes, daqueles que de alguma forma, também contribuíram para que eu chegasse até aqui, porém posso acabar deixando de citar alguém, sendo assim minha gratidão é eterna a todos que fizeram parte dessa trajetória, alguns destes se tornaram amigos importantes, que espero tê-los presentes ao longo da vida. A aqueles, que por consequências da vida, não estão mais presentes, terei sempre um lugar especial em minha memória.

Agradeço também aos meus avós, tios, tias, primos, primas e todas as pessoas que compõe e fazem parte da minha família, direta ou indiretamente, muitos tiveram presentes e contribuíram para que eu chegasse nesse momento.

E por último, mas não menos importante, agradeço as pessoas com quem trabalhei e ou convivi na Fundação Dom Cabral, PUC Minas, Prodel Eventos e CEEP de Assaí, nesses locais tive a oportunidade de conhecer e conviver com pessoas que contribuíram muito para meu desenvolvimento, deixo aqui meu muito obrigado a todos!

“Só há duas maneiras de viver a vida: a primeira é vivê-la como se os milagres não existissem. A segunda é vivê-la como se tudo fosse milagre.” Albert Einstein

RESUMO

ANDRADE, Vinicius Gomes de Oliveira. **Controlador fuzzy de baixo custo aplicado a um conversor buck**. 2019. 77 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

Este trabalho apresenta um estudo que envolve principalmente as áreas de Sistemas Microcontrolados, Sistemas Inteligentes, Sistemas de Controle e Eletrônica de Potência. Tem como objetivo propor uma solução de controle inteligente à um conversor estático CC-CC, com baixo custo de implementação. O controle do chaveamento de um conversor CC-CC abaixador (tipo buck) utilizando a Lógica Fuzzy, é implementado em um firmware para ser executado no microcontrolador PIC18F4550 da Microchip®. O firmware apresenta uma boa alternativa de baixo custo, para o controle inteligente de sistemas de complexa modelagem, os resultados obtidos são satisfatórios quando comparados aos resultados obtidos via simulação no Simulink/MATLAB®.

Palavras-chave: Lógica Fuzzy. Microcontrolador. Conversor Buck. PIC18F4550.

ABSTRACT

ANDRADE, Vinicius Gomes de Oliveira. **Low cost fuzzy controller applied to a buck converter.** 2019. 77 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

This work presents a study that mainly involves the areas of Microcontrolled Systems, Intelligent Systems, Control Systems and Power Electronics. It aims to propose a smart control solution to a static converter CC-CC, with low implementation cost. The switching control of a DC-DC converter (buck type) using the Fuzzy Logic, is implemented in a firmware to run on Microchip® PIC18F4550 microcontroller. The firmware presents a good alternative of low cost, for the intelligent control of systems of complex modeling, the obtained results are satisfactory when compared to the results obtained by simulation in Simulink / MATLAB®.

Keywords: Fuzzy logic. Microcontroller. Buck Converter. PIC18F4550.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos, o uso de controladores digitais tornou-se cada vez mais popular para controlar conversores CC-CC. Esses esquemas de controle incorporam microcontroladores para analisar o sinal de entrada e produzir o sinal de saída apropriado para o controle do chaveamento destes conversores. A popularidade do uso de microcontroladores surge a partir de sua versatilidade, robustez, confiabilidade, compatibilidade com outros sistemas digitais e sua capacidade de incorporar esquemas avançados de controle aliado ao grande número de funções e recursos presentes em um único chip. (KOLAKOWSKI, 2009)

1.1 Delimitação do tema

Conversores estáticos de corrente contínua (CC), ou simplesmente conversores CC-CC, representam um ramo importante da Eletrônica de Potência, ciência que no século 21 está gerando grandes impactos sobre o desenvolvimento da indústria mundial, e especialmente na brasileira, possibilitando avanços na área de conversão de energia, no tratamento de energias renováveis como a eólica, a solar e a de célula de combustível. É importante ressaltar que diversas áreas foram beneficiadas com o avanço da tecnologia em eletrônica de potência, tais como, programas aeroespaciais, aeronavais, a indústria de informática, de automação industrial, acionamentos elétricos e controle de processos industriais. Estas áreas obtiveram a possibilidade de arquitetar fontes de alimentação mais eficientes, com menor peso e volume. (BARBI, 2006)

A lógica *fuzzy* já foi aplicada com sucesso em diversos problemas de difícil modelagem e não lineares da engenharia. Na literatura existem diversas demonstrações que comprovam como os controladores *fuzzy* podem reduzir o custo de desenvolvimento e fornecerem melhor desempenho em situações que as técnicas de controle clássico são de difícil modelagem ou implementação. (SPATTI, 2017)

1.2 Problemas e justificativas

A ideia básica de controle é comandar um sistema para executar o que se deseja, monitorando o desempenho do sistema e ajustando sua entrada de modo a forçar o desempenho desejado. A saída ou estados do sistema são medidos e devolvidos ao controlador. Com base nessas informações, o controlador decide como alterar a entrada do sistema para melhorar o

desempenho dele. Grande parte do controle convencional é baseado em modelos matemáticos ou funções de transferência. No entanto, em alguns casos, esses métodos falham, porque um modelo matemático suficientemente preciso do sistema não é conhecido. Nesses casos, se um conhecimento suficiente sobre como controlar o sistema for disponibilizado por um "especialista", um sistema difuso pode ser projetado para controlar efetivamente o sistema. (LILLY, 2010)

Os conversores CC-CC possuem componentes não lineares e estrutura variável no tempo, o que torna a modelagem complexa, sendo que essa situação se agrava, à medida que a carga se altera. Com os avanços no hardware digital e técnicas de controle digital, se torna viável a implementação de um controle *fuzzy* aos conversores CC-CC. (PERRY *et al*., 2007)

Processadores digitais de sinais ou DSP's (do inglês, *Digital Signal Processor*) são geralmente otimizados para executar sistemas que necessitam de realizar muitos cálculos com alta velocidade de processamento, como é o caso de amplos sistemas de controle *fuzzy*, porém dependendo da situação, são dispendiosos e não possuem funções de controle e comunicação, como funções de conversão analógico-digital (A/D) e saídas com modulação por largura de pulso (PWM), funcionalidades as quais estão geralmente presentes em microcontroladores de baixo custo. Desta maneira, o objetivo de implementar um controle *fuzzy* em um conversor CC-CC executado a partir de um microcontrolador de baixo custo seria interessante para diversas aplicações que necessitam destes conversores. (GUPTA *et al*, 1997)

O microcontrolador utilizado neste trabalho é o PIC18F4550 da *Microchip*®, este possui um desempenho computacional satisfatório, com diversos recursos presentes em um único chip, aliado a um preço econômico e robustez. Contudo o principal motivo da escolha desse microcontrolador é sua disponibilidade nos laboratórios do câmpus de Cornélio Procópio da Universidade Tecnológica Federal do Paraná.

1.3 Objetivo geral

O objetivo deste trabalho é desenvolver um sistema de controle *fuzzy* para um conversor CC-CC abaixador (*buck*) sendo executado por um microcontrolador de baixo custo.

1.3.1 Objetivos específicos

- Desenvolver um *firmware* para o microcontrolador PIC18F4550, com o projeto completo do sistema *fuzzy* de controle;

- Utilizar o conversor analógico para digital do PIC18F4550 para realizar a aquisição das variáveis de entrada do sistema *fuzzy* de controle;
- Gerar um sinal PWM em função do resultado do sistema *fuzzy*.
- Verificar por meio de simulação a atuação do sistema *fuzzy* presente no microcontrolador em um conversor CC-CC abaixador (*buck*).

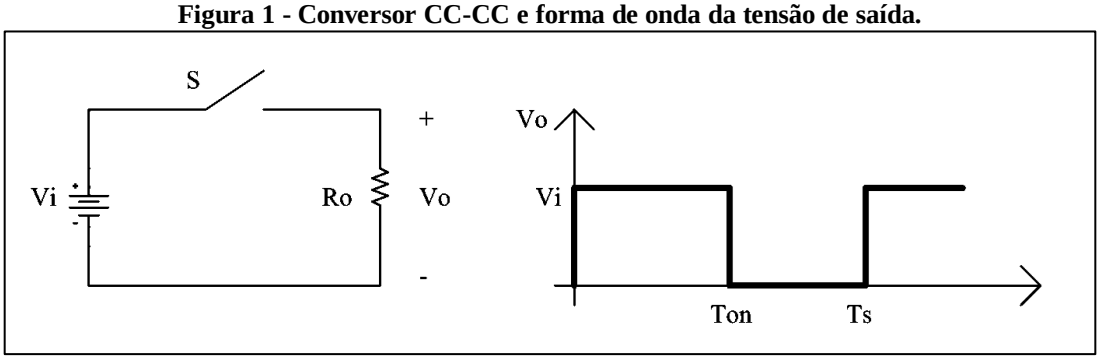
2. FUNDAMENTAÇÃO TEÓRICA

2.1 Conversor chaveado CC-CC abaixador *buck*

Conversores chaveados CC-CC são aparelhos compostos por semicondutores de potência atuando como interruptores e por elementos passivos, como indutores e capacitores, que tem por função controlar o fluxo de potência de uma fonte de entrada para uma fonte de saída. (PETRY, 2001)

Na Figura 1 mostra o princípio de funcionamento de um conversor CC-CC abaixador básico.

Figura 1 - Conversor CC-CC e forma de onda da tensão de saída.



Fonte: Petry (2001, p.3).

O intervalo de comutação é definido como:

$$T_s = \frac{1}{F_s} \tag{1}$$

Sendo que F_s é a frequência de comutação.

A razão entre o intervalo de comutação (Ts) e o intervalo de condução do interruptor S (Ton) é definido por razão cíclica e dada por:

$$D = \frac{T_{on}}{T_s} \tag{2}$$

A tensão média na saída deste conversor é calculada por:

$$V_o = \frac{1}{T_s}\int_0^{T\infty} V_i \, dt = V_i \, \frac{T_{on}}{T_s} \tag{3}$$

Simplificando, a partir da Equação (2), tem-se:

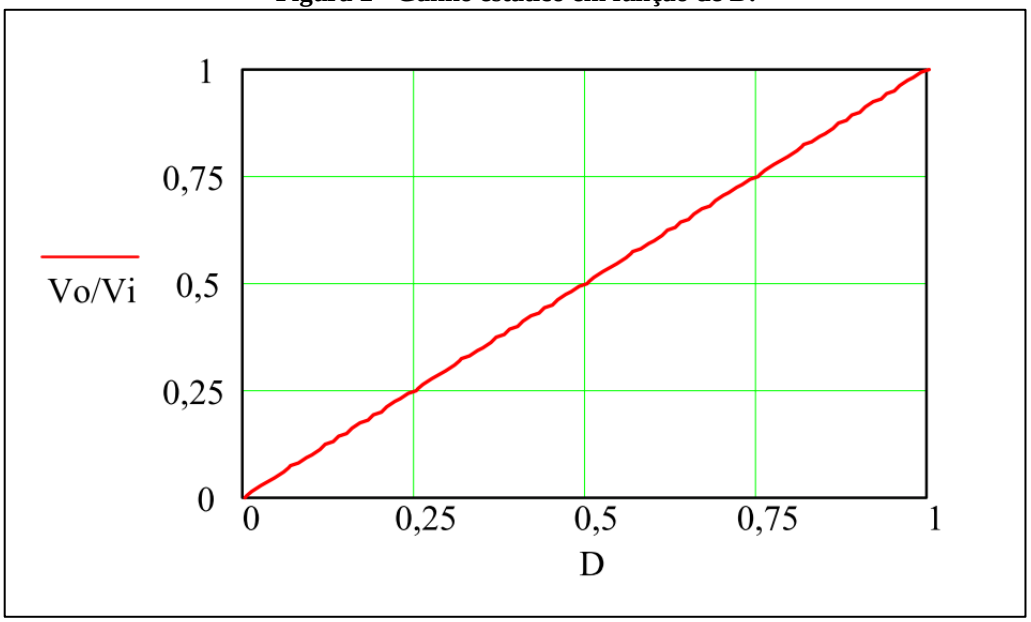
$$V_o = D \,.\, V_i \tag{4}$$

A relação entre a tensão de saída e a tensão de entrada é definida como ganho estático do conversor, dada então por:

$$D = \frac{V_o}{V_i} \tag{5}$$

Pelo gráfico mostrado na Figura 2 pode-se notar que a variação da tensão de saída com a razão cíclica é linear. (PETRY, 2001)

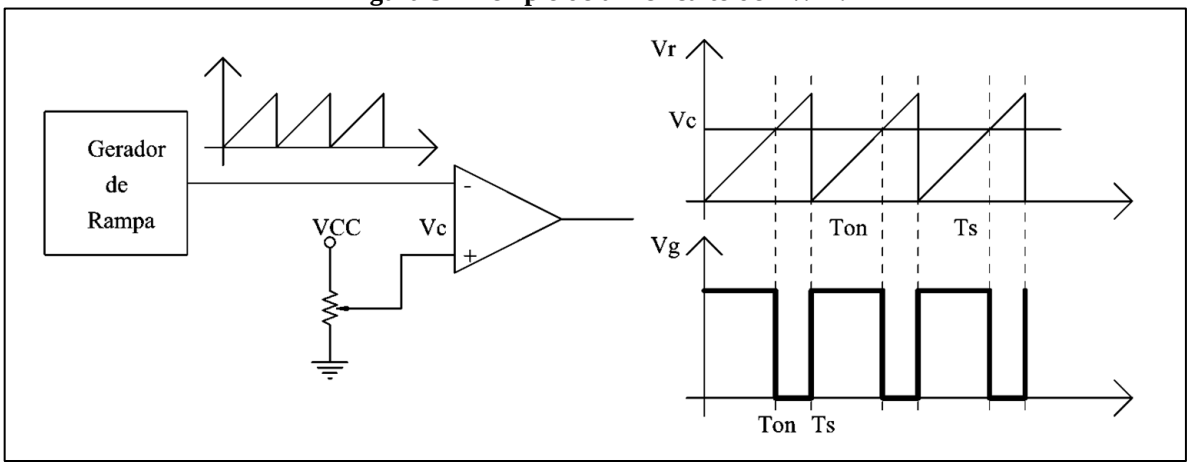
Figura 2 - Ganho estático em função de D.



Fonte: Petry (2001, p.4).

Os sinais de comando da chave podem ser gerados com frequência de comutação fixa ou variável. Uma maneira de gerar os sinais de comando com frequência fixa é através de modulação por largura de pulso ou P*WM* (do inglês, *Pulse Width Modulation*). A Figura 3 representa uma das técnicas para realizar a modulação por largura de pulso. (PETRY, 2001)

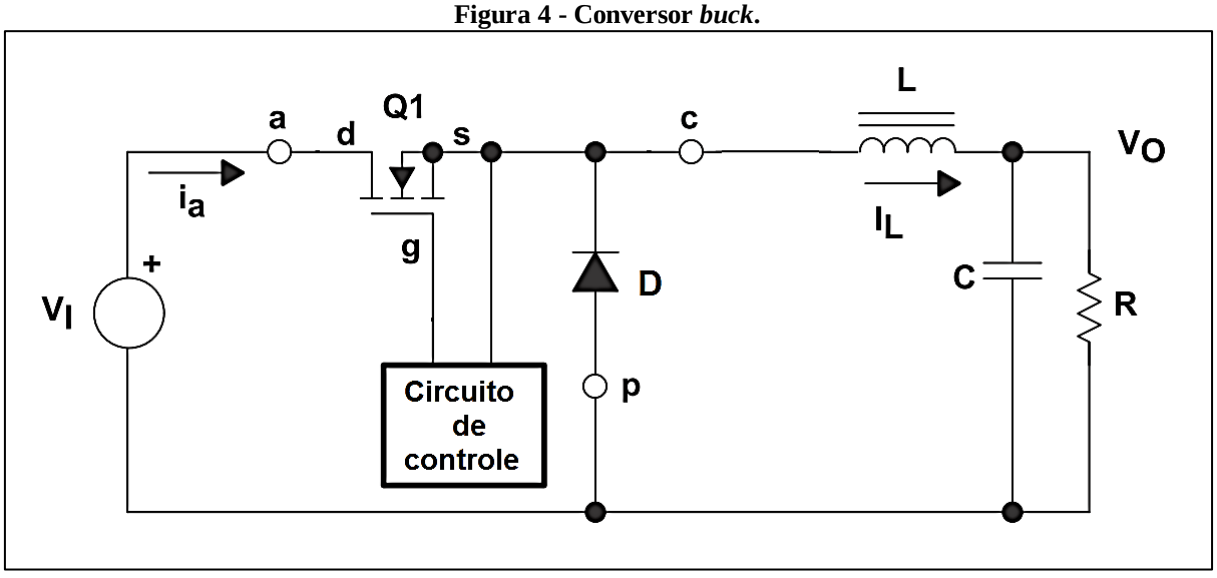
Figura 3 -Exemplo de um circuito de PWM.



Fonte: Petry (2001, p.4).

O conversor *buck* é um conversor abaixador de tensão, caracterizado por produzir um valor médio de tensão na saída menor que o valor médio da tensão de entrada, enquanto a corrente média de saída é maior que a corrente média de entrada. Teoricamente, esse tipo de conversor possibilita uma variação contínua da tensão (Vo) na carga desde zero até o valor de tensão de entrada (Vi). A Figura 4 mostra um esquema simplificado do circuito do conversor *buck*. (BARBI, 2006)

Figura 4 - Conversor *buck*.

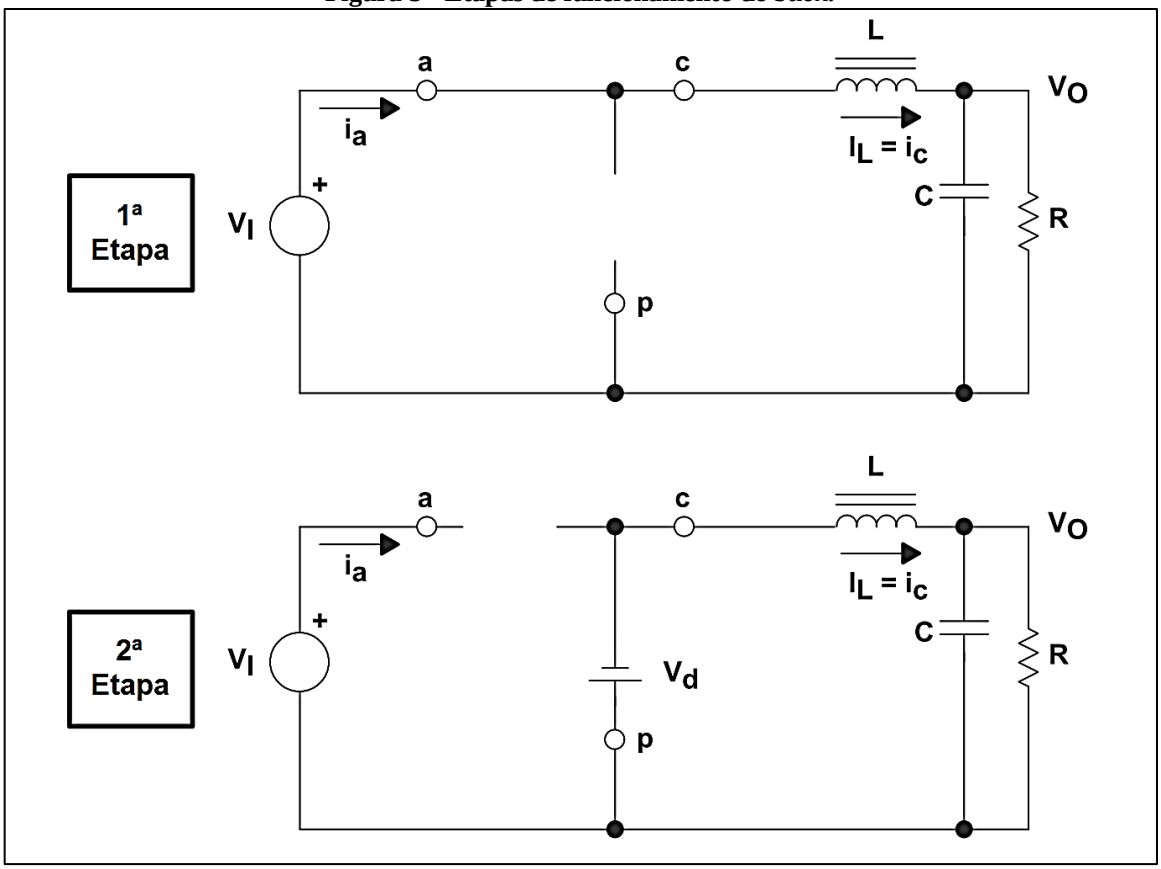


Fonte: Adaptado de Texas Instruments (1999, p.02).

O componente responsável pelo chaveamento do esquemático da Figura 4 é Q1, que geralmente é um MOSFET de potência. O diodo, D, é geralmente chamado diodo de roda livre. O indutor, L, e capacitor, C, compõem o filtro de saída. O resistor, R, representa a carga vista pela saída do conversor. (TEXAS INSTRUMENTS, 1999)

As etapas de funcionamento do conversor *buck* estão descritas na Figura 5 a seguir.

Figura 5 - Etapas de funcionamento do *buck*.



Fonte: Adaptado de Texas Instruments (1999, p.04).

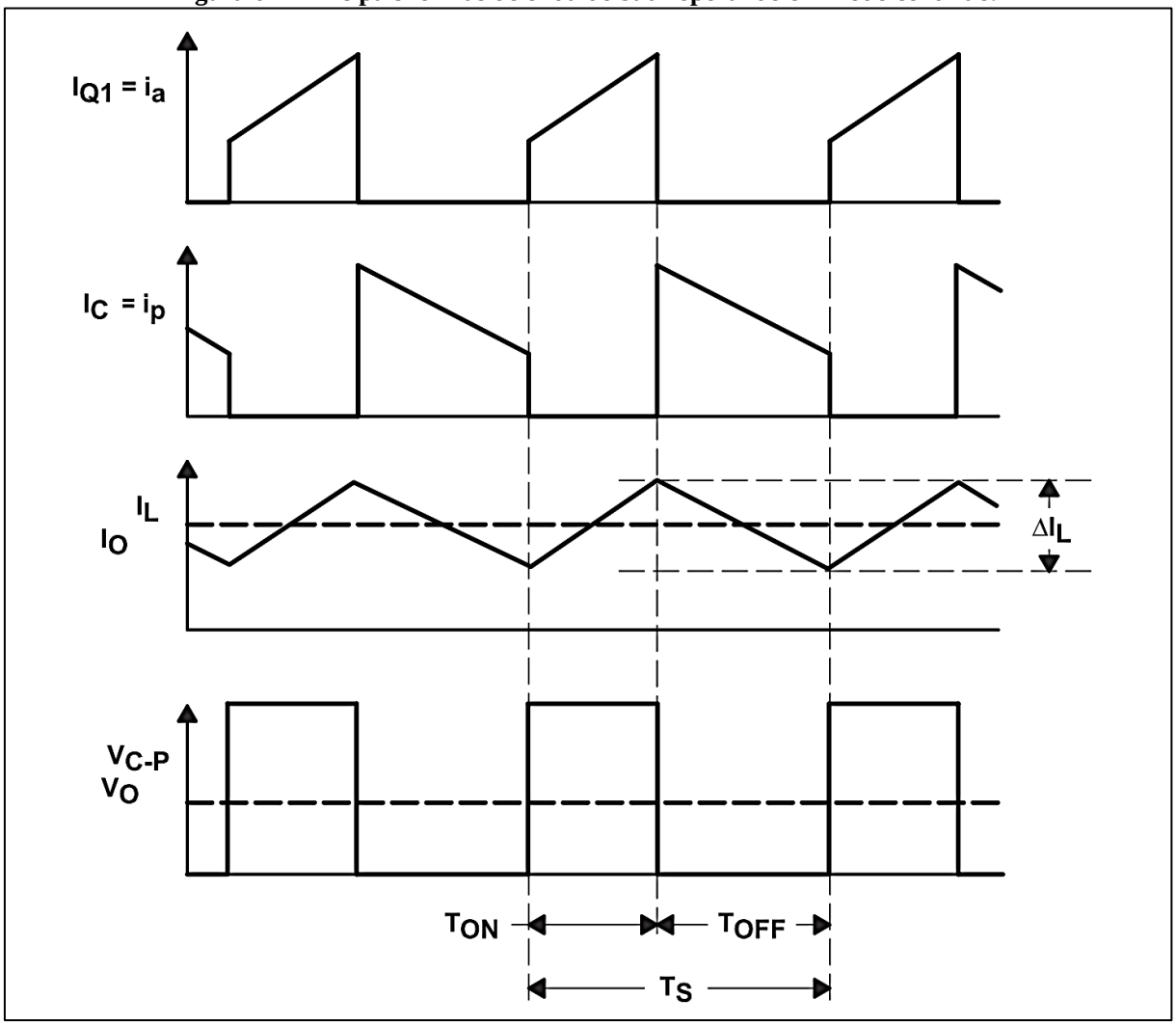
- 1ª Etapa (0, DTs): Q1 está conduzindo. A corrente circula por L e pela saída. Nesta etapa V_I fornece energia para a saída e para a magnetização do indutor L.
- 2ª Etapa (DTs, $(1 - D)$Ts): Q1 está bloqueado. No instante de abertura de S o diodo D entra em condução. A energia do indutor é transferida para a carga, isto é, o indutor é desmagnetizado.

O conversor *buck* pode operar em três modos distintos: (PETRY, 2001)

- Condução Contínua**:** a corrente em L não se anula durante um período de comutação;
- Condução Descontínua: a corrente em L se anula a cada período de comutação;
- Condução Crítica: a corrente em L está no limiar de se anular a cada período de comutação.

As principais formas de onda do conversor *buck* operando em modo contínuo são mostradas na Figura 6.

Figura 6 – Principais formas de onda do *buck* operando em modo contínuo.



Fonte: Adaptado de Texas Instruments (1999, p.05).

Como a tensão média sobre o indutor deve ser nula, então,

$$V_o = V_{cp_{med}} = \frac{1}{T_s}\int_0^{DT_s} V_i \, dt \tag{6}$$

Sendo assim,

$$\frac{V_o}{V_i} = D \tag{7}$$

A partir da análise do circuito, por definição, considerando uma condução contínua, que a variação da corrente no indutor é dada pela Equação (8).

$$\Delta I_L = \frac{(V_i - V_o)\,.\,D}{f_s\,.L_f} \tag{8}$$

Em que Vi é a tensão de entrada, Vo é a tensão na carga, fs é a frequência de chaveamento e L é a indutância do elemento armazenador de energia.

Por definição, considerando o *buck* em condução contínua, que a variação da tensão no capacitor é dada pela seguinte fórmula:

$$\Delta V_C = \frac{\Delta I_L}{8 \cdot f_s \cdot C_f} \tag{9}$$

Em que ΔI_L é a variação da corrente no indutor, fs é a frequência de chaveamento e Cf é o valor da capacitância do elemento armazenador de energia.

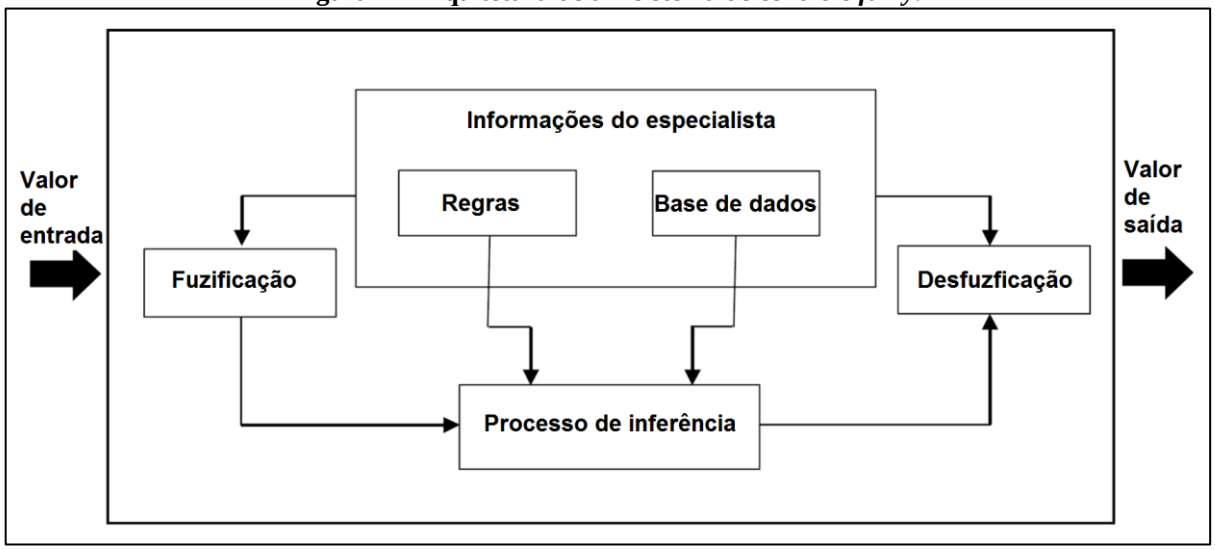
2.2 Lógica *fuzzy*

A proposição de conjuntos *fuzzy*, apresentada por Zadeh em 1965, pode ser utilizada para descrever em termos matemáticos as informações imprecisas de um determinado processo, através de um conjunto de regras linguísticas. (ZADEH, 1965)

O procedimento para o projeto de controladores *fuzzy* é baseado nas informações do especialista, sendo em seguida ajustado de forma a obter um ponto satisfatório de operação do sistema. Consiste no processo de inferência como descrito na Figura 7, sendo dividido em três etapas: fuzzificação da entrada, base de dados/regras e defuzzificação. As etapas são descritas a seguir: (LEE, 1990)

- Fuzzificação: as variáveis de entrada, valores de tensão ou de corrente do sistema são traduzidas em valores *fuzzy* verbais. Esta parte do sistema inclui a definição das funções de pertinência e variáveis linguísticas, bem como a determinação do universo de discurso de cada uma destas variáveis.
- Inferência: utiliza-se regras do tipo se/então, de forma a produzir uma saída *fuzzy* a ser utilizada no cálculo da variável de saída do processo.
- Defuzzificação: Nesta etapa, as variáveis *fuzzy*, obtidas na inferência, são transformadas em variáveis reais a serem enviadas ao processo.
- Base de dados/regras: Etapa de definição das funções de pertinência dos sistemas de fuzzificação e defuzzificação, bem como de definição do conjunto de regras *fuzzy*.

Figura 7 - Arquitetura de um sistema de controle *fuzzy*.



Fonte: Adaptado de Cavallaro (2015, p.12361).

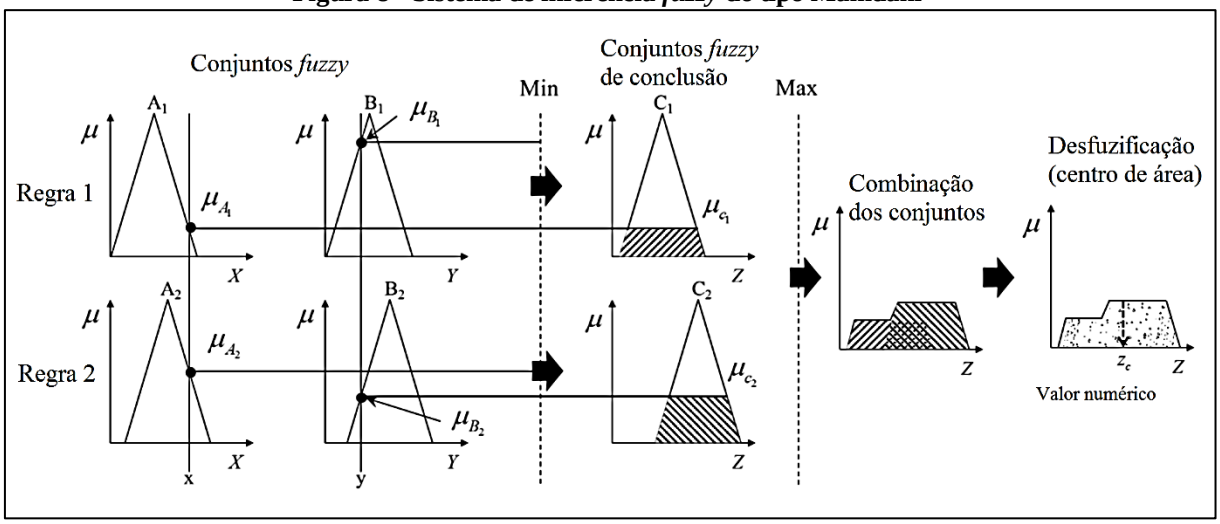
2.2.1 Sistema de inferência *fuzzy*

A inferência *fuzzy* permite o tratamento e manipulação de informações incertas e imprecisas, as quais estão representadas por uma família de conjuntos *fuzzy*. Tais sistemas de inferência oferecem uma forma sistemática para a modelagem de sistemas, cujas informações a respeito dos mesmos são fornecidas de forma qualitativa. Por tanto, a representação do sistema pode ser realizada por meio de variáveis linguísticas, as quais expressam o comportamento do sistema. Variáveis linguísticas são aquelas que permitem a descrição de informações que estão normalmente disponibilizadas de forma qualitativa representadas por conjunto de funções de pertinência. As principais operações entre variáveis linguísticas são realizadas por meio da utilização dos conectivos “E”, “OU” e “NÃO”, empregados para compor os relacionamentos lógicos entre os termos das variáveis linguísticas. Esses conectivos “E” e “OU” são também definidos por meio de operadores de interseção e união. Uma relação matemática indica como estão associados os elementos de um conjunto em relação aos elementos de um outro conjunto. Nas relações *fuzzy*, o nível de associação entre dois conjuntos *fuzzy* é também fornecida por meio de graus de pertinência que possuem valores entre 0 e 1. O processo de inferência *fuzzy*, também conhecido como raciocínio aproximado, permite mapear o conhecimento de um sistema por meio de regras *fuzzy* do tipo “SE-ENTÃO”. A partir da análise de um conjunto finito dessas regras, pode-se então determinar, por meio do processo de inferência, o comportamento das variáveis de saída do sistema. Assim, as regras associadas ao processo de inferência *fuzzy* possuem a seguinte forma: (SPATTI, 2017)

- SE <condição> ENTÃO <ação>.

Dentre os métodos de inferência *fuzzy*, o método dos mínimos e máximos, proposto por Mamdani em 1975, tem sido amplamente utilizado, por ter uma abordagem de raciocínio simples e de fácil interpretação. A Figura 8 mostra um exemplo para melhor compreensão desse sistema de inferência. (CHO *et al*, 2017)

Figura 8 - Sistema de inferência *fuzzy* do tipo Mamdani



Fonte: Adaptado de Cho *et al* (2017, p.05).

No exemplo da Figura 8 existem duas regras:

- Regra 1: SE < "x" é "A_1" E "y" é B_1 > ENTÃO < "z" é C_1 >;
- Regra 2: SE < "x" é "A_2" E "y" é B_2 > ENTÃO < "z" é C_2 >.

Sendo que $\mu_{Cj}\ (z_j)$, é o valor numérico referente ao grau de pertinência da saída em função dos dois graus de pertinência relativos a entrada, $\mu_{Ai}(x_i)$ e $\mu_{Bi}(y_i)$, descritos na Equação (10)

$$\mu_{Cj}\ (z_j) = \bigvee_{n=1}^{R}[\mu_{Ai}(x_i) \wedge \mu_{Bi}(y_i)] \tag{10}$$

Onde, *i* e *j* variam de acordo com a quantidade de conjuntos *fuzzy* das entradas e saídas, *R* é a quantidade de regras e ∨ é o operador de máximo e ∧ o operador de mínimo. Conforme demonstrado na Equação (10), os conjuntos *fuzzy* de saída Cj são derivados tomando um valor menor entre os graus de pertinência dos conjuntos *fuzzy* de entrada Ai e Bi, em cada regra, e o valor máximo dos conjuntos *fuzzy* de saída Cj é a combinação dos conjuntos *fuzzy* de conclusão C, que é o resultado da inferência *fuzzy*. Após este processo é necessário que o conjunto de valores C seja convertido em um valor numérico (Zc) pelo processo de defuzificação, sendo

que um dos métodos que pode ser utilizado é centro de área (centróide), calculado a partir da Equação (11). (CHO *et al*, 2017)

$$z_C = \frac{\sum_{j=1}^{k} \mu_{Cj}(z_i).z_i}{\sum_{j=1}^{k} \mu_{Cj}(z_i)} \quad (11)$$

2.3 Microcontrolador

Microcontroladores são circuitos integrados programáveis que agregam em um único chip diversos dispositivos, mas basicamente possuem um processador, memória, pinos de entrada/saída e periféricos.

O microcontrolador utilizado neste trabalho será o PIC18F4550 da *Microchip*® por ser um microcontrolador de fácil acesso e comum nos laboratórios da UTFPR. As principais características desse microcontrolador são:

- *CPU RISC*;
- Código fonte compatível com os conjuntos de instruções PIC16 e PIC17;
- *USB V2.0* compatível com até 12 Mb/s a velocidade máxima;
- Modos de *clock* externo de até 48 MHz;
- Instruções de 16 bits, caminho de dados de 8 bits;
- Níveis de prioridade para interrupções;
- Multiplicador de Hardware 8 x 8 de Ciclos Únicos;
- Memória do Programa *Flash* de 16 kBytes;
- Memória de dados *RAM* de 2048 Bytes;
- Memória de dados *EEPROM* de 256 Bytes;

Recursos Periféricos:

- Alta corrente de dreno / fonte (25mA / 25mA);
- Três temporizadores / contadores de 16 bits;
- Um temporizador / contador de 8 bits com pré-escala*;*
- Dois módulos de captura / comparação (CCP/PWM);
- Captura de 16 bits, máxima resolução de 6.25ns;
- Comparação 16 bits, máxima resolução de 100ns;
- *Serial Peripheral Interface* de 3 fios (suporta todos os 4 modos SPI);
- Modo *Inter-Integrated Circuit - I²C Master* e *Slave*;

- Módulo *Universal Synchronous Asynchronous Receiver Transmitter – USART*;
- Conversor Analógico para Digital de Treze Canais de 10 bits;

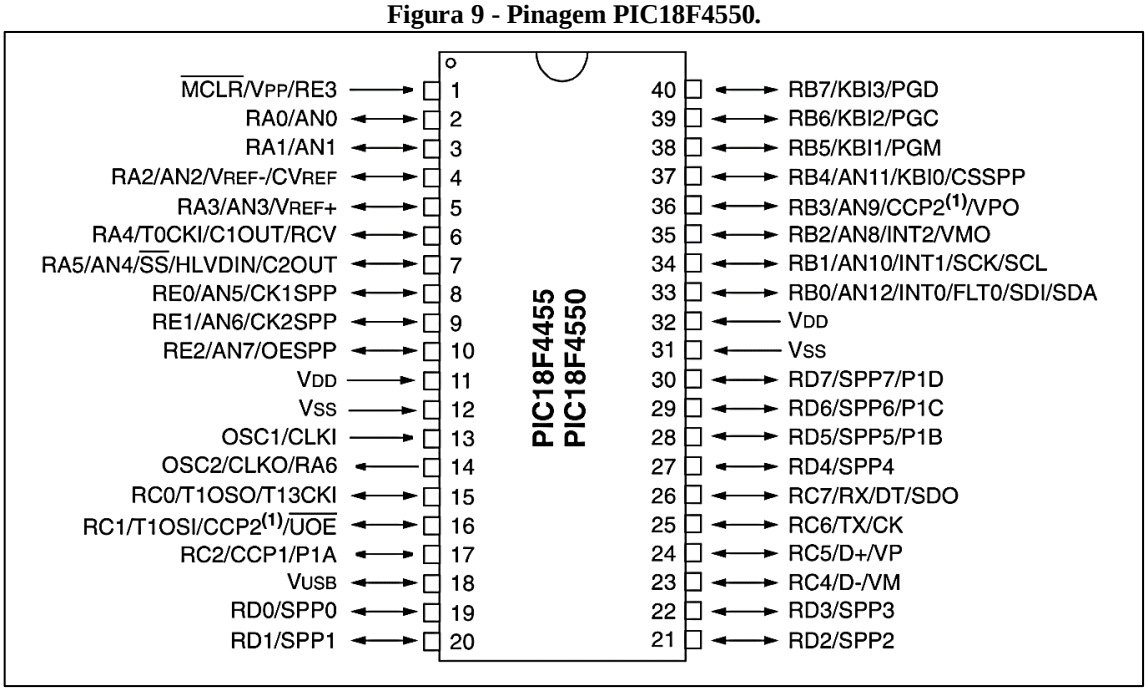
Recursos Especiais:

- *Power-On Reset*;
- Temporizador de ativação (*PWRT)* e Temporizador de inicialização do oscilador (OST);
- 1.000 ciclos de apagamento / gravação *Enhanced Flash Program Memory*;
- 1.000.000 de ciclos de apagamento / gravação *EEPROM Data Memory;*
- *Watchdog Timer* (*WDT*) com oscilador próprio *RC On-Chip*;
- Proteção de código programável;
- Modo de economia de energia no modo *SLEEP*;
- Uso da entrada como oscilador secundário de *clock* de até 32kHz;
- Alimentação única de 5V para a programação serial via dois pinos;
- Depuração *In-Circuit (ICD)*
- Tecnologia *CMOS FLASH* de baixa potência e alta velocidade;
- Design totalmente estático;
- Ampla faixa de tensão operacional de 2.0V a 5.5V.

Pinos de Entradas e saídas:

- 35 pinos de *input-output I / O* com controle de direção individual;
- *DIP (dual in-line package)* de 40 pinos;

Figura 9 - Pinagem PIC18F4550.



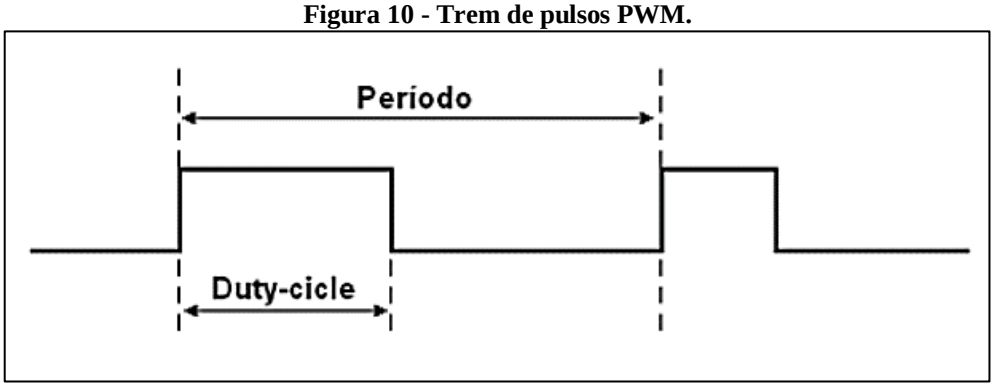
Fonte: Microchip® (2009, p.04)

Para a implementação do sistema de controle *fuzzy* proposto por esse trabalho, o módulo CCP e o módulo de conversão analógico/digital serão de suma importância, pois serão a interface de controle o sistema, no caso o conversor *buck.*

2.3.1 Modo *PWM* do Módulo CCP

Nesse modo o módulo CCP permite utilizar sinais modulados em largura de pulso (*PWM – Pulse Width Modulation*), que consiste em representar um valor pelo *duty-cycle* (isto é, tempo em nível lógico alto) de um trem de pulsos de frequência fixa como demonstrado pela Figura 10. A maior parte das aplicações de *PWM* para microcontroladores se aproveita da propriedade da energia de um sinal retangular ser proporcional ao seu *duty-cycle* (a energia de um sinal está relacionada com a área entre o sinal e o eixo do tempo). (FONSECA SOBRINHO, 2017)

Figura 10 - Trem de pulsos PWM.



Fonte: Fonseca Sobrinho (2017)

O PWM precisa de uma base de tempo que dará a frequência do sinal. O modulo CCP utiliza o *Timer 2* (TMR2) para conseguir essa base. Também é necessário que o pino CCP2 seja configurado como saída. A geração do período do sinal *PWM* ocorre da seguinte maneira: cada vez que TMR2 coincide com PR2, o pino CCP2 é colocado em nível lógico ‘1’ e TMR2 é reiniciado, gerando assim a frequência do sinal. O período do sinal PWM pode ser calculado pela Equação (12). (FONSECA SOBRINHO, 2017)

$$T_{PWM} = 4 * T_{OSC} * T2CKPS[1:0] * (PR2 + 1) \quad (12)$$

Onde, T_{PWM} é o período do PWM, T_{OSC} é o período do oscilador, $T2CKPS[1:0]$ é o fator de pré-escala do TMR2 e PR2 o valor de carga de comparação do TMR2 de 8 bits (0 a 255).

A geração do período do *duty-cycle* do sinal PWM em porcentagem de 0% a 100% do período total do sinal *PWM*, pode ser calculado a partir da Equação (13).

$$T_{DUTY} = T_{OSC} * T2CKPS[1:0] * DC[9:0] \quad (13)$$

Onde,

T_{DUTY} é o período do duty-cycle, T_{OSC} é o período do oscilador, $T2CKPS[1:0]$ é o fator de *pré-escala* do TMR2 e $DC[9:0]$ é o valor de carga para determinar o *duty-cycle* de 10 bits (0 a 1023)

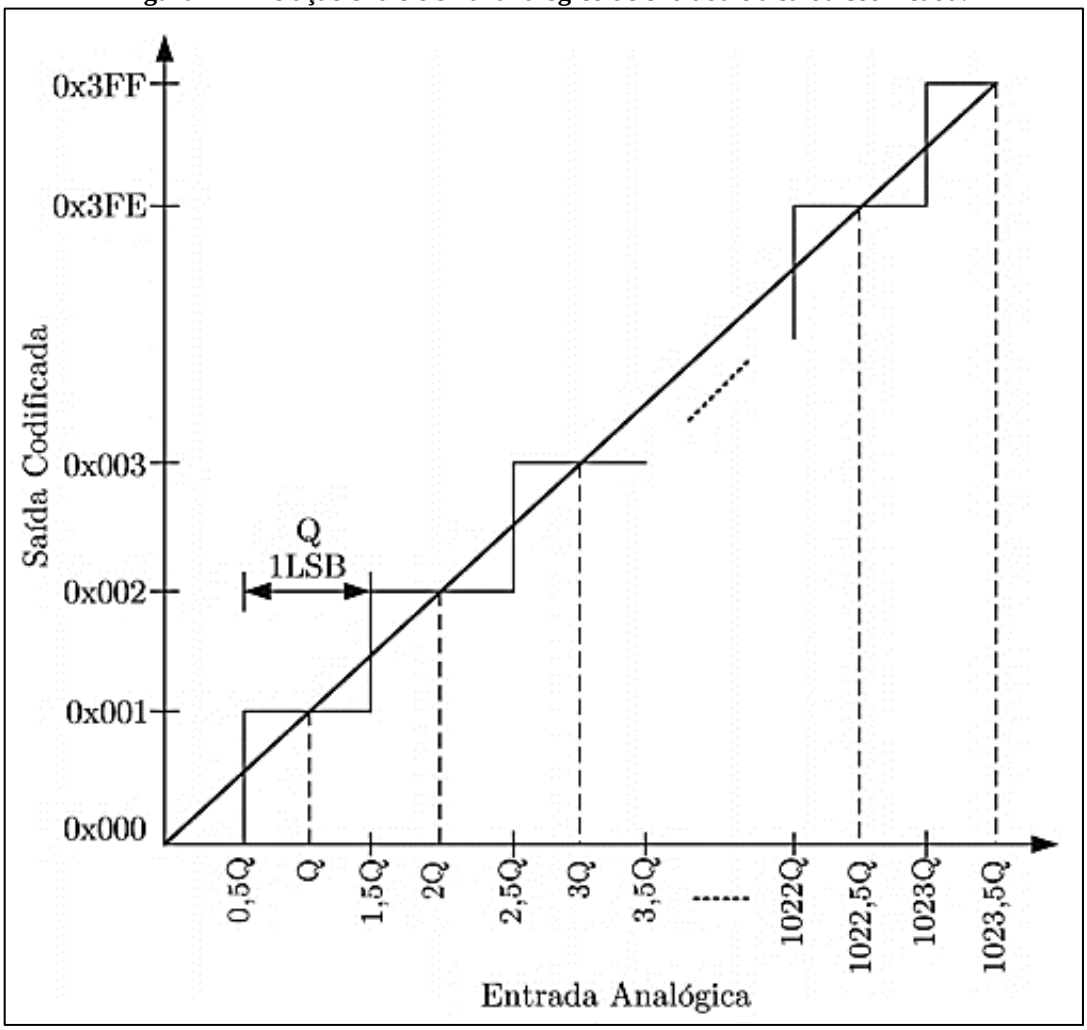
2.3.2 Módulo de conversão analógico digital (A/D) de 10 *bits*

Este módulo permite a conversão de um sinal de entrada analógico em sinal digital com resolução de 10 bits. O módulo possui cinco registradores:

- Registrador ADRESH, armazena o resultado binário mais significativo da conversão A/D.
- Registrador ADRESL, armazena o resultado binário menos significativo da conversão A/D.
- Demais registradores de controle, ADCON0, ADCON1 e ADCON2.

O processo de conversão realiza a discretização de uma quantidade finita de um sinal analógico e o quantiza por um determinado número de bits, no caso do PIC18F4550, 10 *bits*, isso quer dizer que a entrada analógica pode ser configurada, de modo que o intervalo, por exemplo de 0 a 5,0V máximo, sejam representados por 1024 níveis quantizados Q. A Figura 11 representa a relação entre a entrada analógica e saída codificada. Onde o erro dessa conversão, denominado erro de quantização, pode assumir valores de 0 a 1 LSB/2. (FONSECA SOBRINHO, 2017)

Figura 11 - Relação entre o sinal analógico de entrada e a saída codificada.



Fonte: Fonseca Sobrinho (2017)

3. METODOLOGIA

A ideia deste trabalho surgiu a partir da elaboração do projeto final apresentado à disciplina de Sistemas Inteligentes Aplicados a Engenharia da Universidade Tecnológica Federal do Paraná, câmpus Cornélio Procópio, cursada no segundo semestre de 2017. O projeto, disponível no Anexo A, consistiu no estudo comparativo de um controlador *fuzzy* em relação a um controlador clássico PI, aplicado no controle do chaveamento do conversor *buck*. O conversor estático CC-CC abaixador do tipo *buck*, como explicado na Seção 2.1 deste trabalho, possui uma característica linear entre a tensão aplicada em sua entrada versus a tensão obtida em sua saída. Contudo esse sistema não é linear, devido a presença de elementos semicondutores, como o elemento responsável pelo chaveamento e o diodo. A partir destas premissas, o controle inteligente realizado a partir da lógica *fuzzy*, como justificado na Seção 2.2, é interessante, principalmente quando existem variações na tensão de entrada e saída do conversor.

3.1 Características do conversor *buck*

Considerando uma possível implementação, estabeleceu-se valores corrente e tensões normalmente utilizados na prática. O dimensionamento do capacitor e indutor, utilizados na simulação, foram calculados considerando como base a fundamentação teórica da Seção 2.1, as equações (9) e (10), anteriormente apresentadas e o *buck* operando em condução contínua, essas características estão descritas na Tabela 1.

Tabela 1 - Características do conversor *buck*.

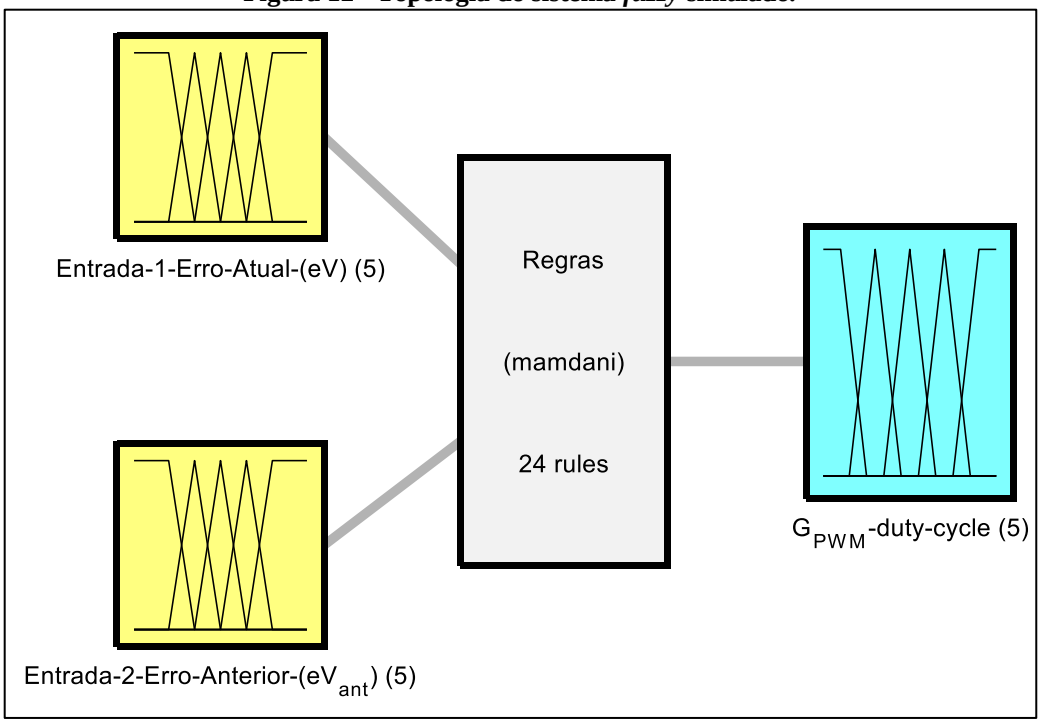
Tensão de entrada	Vi = 12,0 V
Tensão de saída	Vo = 5,0 V
Corrente na carga	Io = 2,0 A
Frequência de chaveamento	fs = 3,0 kHz
Ondulação de corrente no indutor	ΔIL = 1,0 %
Ondulação de tensão na carga	ΔVc = 1,0 %
Valor do indutor	Lf = 70,0 mH
Valor do capacitor	Cf = 16,7 μF

Fonte: Autoria própria.

3.2 Topologia do sistema *fuzzy*

Inicialmente, no surgimento da ideia deste trabalho, o sistema *fuzzy* desenvolvido, consistia em duas variáveis de entrada e uma variável de saída, as entradas eram o valor da corrente na saída do conversor e o erro absoluto da tensão na saída do conversor, a saída era o valor do *duty-cycle* do *PWM*. O sistema *fuzzy* de controle foi elaborado com o auxílio da *Fuzzy Logic Toolbox*™ disponível no software *MATLAB*® como os resultados obtidos via simulações no *Simulink*, que também é uma ferramenta do *MATLAB*®. Os ajustes dos conjuntos de pertinência e as regras do sistema *fuzzy* foram elaboradas intuitivamente com tentativas e correções nos erros, o que gerou bons resultados. No entanto, com o aprofundamento dos estudos nas bibliografias consultadas, foi possível observar que já existem técnicas e modelos bem consolidados no emprego de sistemas *fuzzy* no controle de dispositivos análogos ao conversor *buck*. Assim o sistema *fuzzy* foi modificado e aprimorado, tendo como principais bases de referência, o exemplo 4.1.1 de Lilly (p.46, 2010) e a publicação de Gupta (1997), portanto, não mais utilizou-se o valor de corrente como uma das entradas, e sim o valor do erro absoluto da tensão de uma amostra anterior, a topologia do sistema é descrito pela Figura 12, como ressalva, o algarismo (5), presente nas entradas e saída, indica a quantidade de funções de pertinência contida nas mesmas.

Figura 12 - Topologia do sistema *fuzzy* simulado.



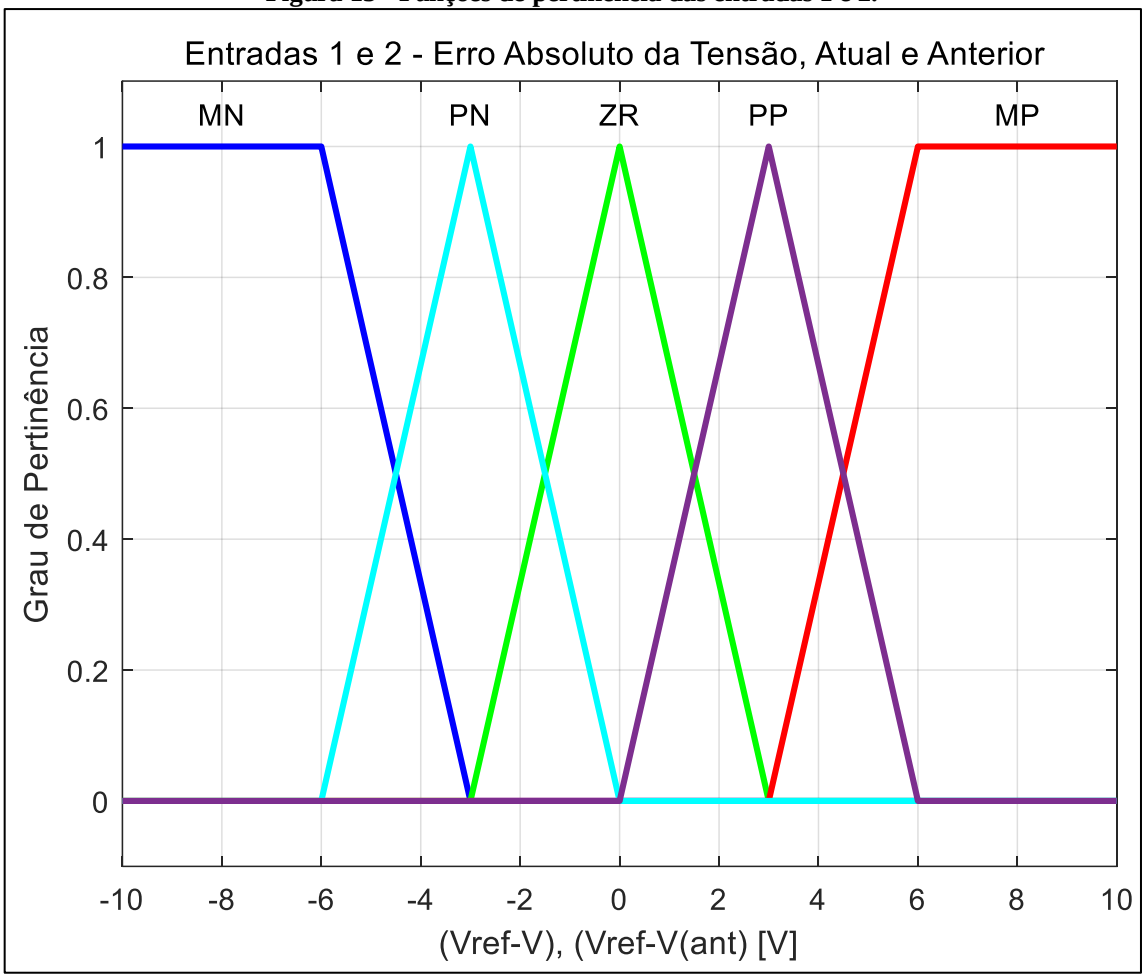
Fonte: Autoria própria.

O erro absoluto é calculado pela diferença entre o valor de referência, nesse caso é 5,0V, e o valor medido na saída do *buck* e pode ser descrito pela a Equação (14).

$$Erro_V(eV) = V_{Ref.} - V_{Medido} \tag{14}$$

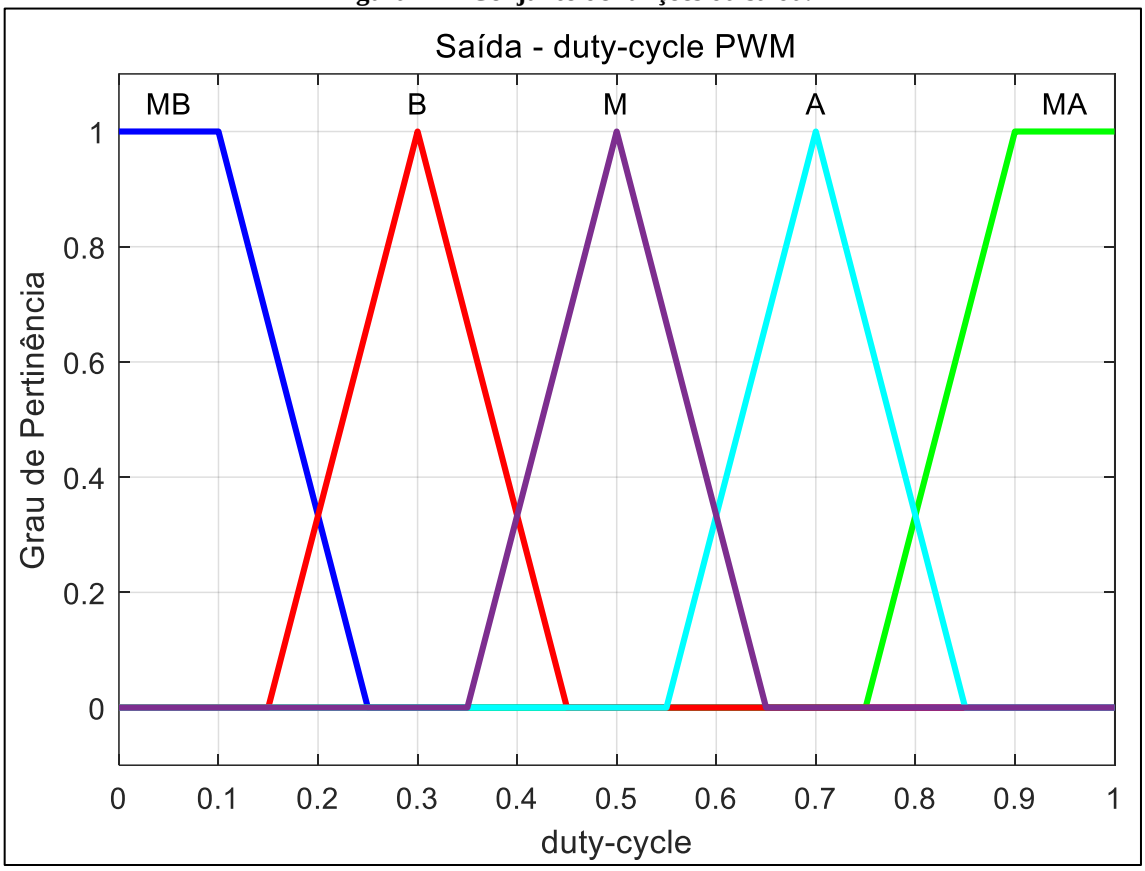
As entradas foram definidas com as funções de pertinências considerando os termos linguísticos, MN (Muito Negativo), PN (Pouco Negativo), ZR (Zero), PP (Pouco Positivo), MP (Muito Positivo), que são relativas ao erro absoluto da tensão, como pode ser observado na Figura 13. A saída foi definida com as funções e respectivos termos, MB (Muito Baixo), B (Baixo), M (Médio), A (Alto) e MA (Muito Alto), que são referentes ao duty-cycle do PWM, as funções representadas pela Figura 14.

Figura 13 - Funções de pertinência das entradas 1 e 2.



Fonte: Autoria própria.

Figura 14 – Conjunto de funções da saída.



Fonte: Autoria própria.

Após a definição das funções das entradas e saída, utilizando o método de inferência *Mamdani*, baseando-se no exemplo 4.1.1 de Lilly (p.46, 2010), determinou-se vinte e quatro regras, que são:

1. **SE** (*Erro Atual* **É** MN) **E** (*Erro Anterior* **É** MN) **ENTÃO** (*duty-cycle* **É** MB);
2. **SE** (*Erro Atual* **É** MN) **E** (*Erro Anterior* **É** PN) **ENTÃO** (*duty-cycle* **É** MB);
3. **SE** (*Erro Atual* **É** MN) **E** (*Erro Anterior* **É** ZR) **ENTÃO** (*duty-cycle* **É** MB);
4. **SE** (*Erro Atual* **É** MN) **E** (*Erro Anterior* **É** PP) **ENTÃO** (*duty-cycle* **É** B);
5. **SE** (*Erro Atual* **É** MN) **E** (*Erro Anterior* **É** MP) **ENTÃO** (*duty-cycle* **É** M);
6. **SE** (*Erro Atual* **É** PN) **E** (*Erro Anterior* **É** MN) **ENTÃO** (*duty-cycle* **É** MB);
7. **SE** (*Erro Atual* **É** PN) **E** (*Erro Anterior* **É** PN) **ENTÃO** (*duty-cycle* **É** MB);
8. **SE** (*Erro Atual* **É** PN) **E** (*Erro Anterior* **É** ZR) **ENTÃO** (*duty-cycle* **É** B);
9. **SE** (*Erro Atual* **É** PN) **E** (*Erro Anterior* **É** PP) **ENTÃO** (*duty-cycle* **É** M);
10. **SE** (*Erro Atual* **É** PN) **E** (*Erro Anterior* **É** MP) **ENTÃO** (*duty-cycle* **É** A);
11. **SE** (*Erro Atual* **É** ZR) **E** (*Erro Anterior* **É** MN) **ENTÃO** (*duty-cycle* **É** MB);
12. **SE** (*Erro Atual* **É** ZR) **E** (*Erro Anterior* **É** PN) **ENTÃO** (*duty-cycle* **É** B);
13. **SE** (*Erro Atual* **É** ZR) **E** (*Erro Anterior* **É** PP) **ENTÃO** (*duty-cycle* **É** A);

14. SE (*Erro Atual* **É** ZR) **E** (*Erro Anterior* **É** MP) **ENTÃO** (*duty-cycle* **É** MA);
15. SE (*Erro Atual* **É** PP) **E** (*Erro Anterior* **É** MN) **ENTÃO** (*duty-cycle* **É** B);
16. SE (*Erro Atual* **É** PP) **E** (*Erro Anterior* **É** PN) **ENTÃO** (*duty-cycle* **É** M);
17. SE (*Erro Atual* **É** PP) **E** (*Erro Anterior* **É** ZR) **ENTÃO** (*duty-cycle* **É** A);
18. SE (*Erro Atual* **É** PP) **E** (*Erro Anterior* **É** PP) **ENTÃO** (*duty-cycle* **É** MA);
19. SE (*Erro Atual* **É** PP) **E** (*Erro Anterior* **É** MP) **ENTÃO** (*duty-cycle* **É** MA);
20. SE (*Erro Atual* **É** MP) **E** (*Erro Anterior* **É** MN) **ENTÃO** (*duty-cycle* **É** M);
21. SE (*Erro Atual* **É** MP) **E** (*Erro Anterior* **É** PN) **ENTÃO** (*duty-cycle* **É** A);
22. SE (*Erro Atual* **É** MP) **E** (*Erro Anterior* **É** ZR) **ENTÃO** (*duty-cycle* **É** MA);
23. SE (*Erro Atual* **É** MP) **E** (*Erro Anterior* **É** PP) **ENTÃO** (*duty-cycle* **É** MA);
24. SE (*Erro Atual* **É** MP) **E** (*Erro Anterior* **É** MP) **ENTÃO** (*duty-cycle* **É** MA);

As regras podem ser resumidas pela Tabela 2, e como pode ser observado, pelas cores, o *duty-cicle* vai variando do azul MB (Muito Baixo) para o vermelho (Muito Alto), de acordo com a variação do erro absoluto da tensão. Quando o erro atual e anterior são ZR (Zero), significa que valor de tensão na saído do *buck* alcançou o valor de referência desejado.

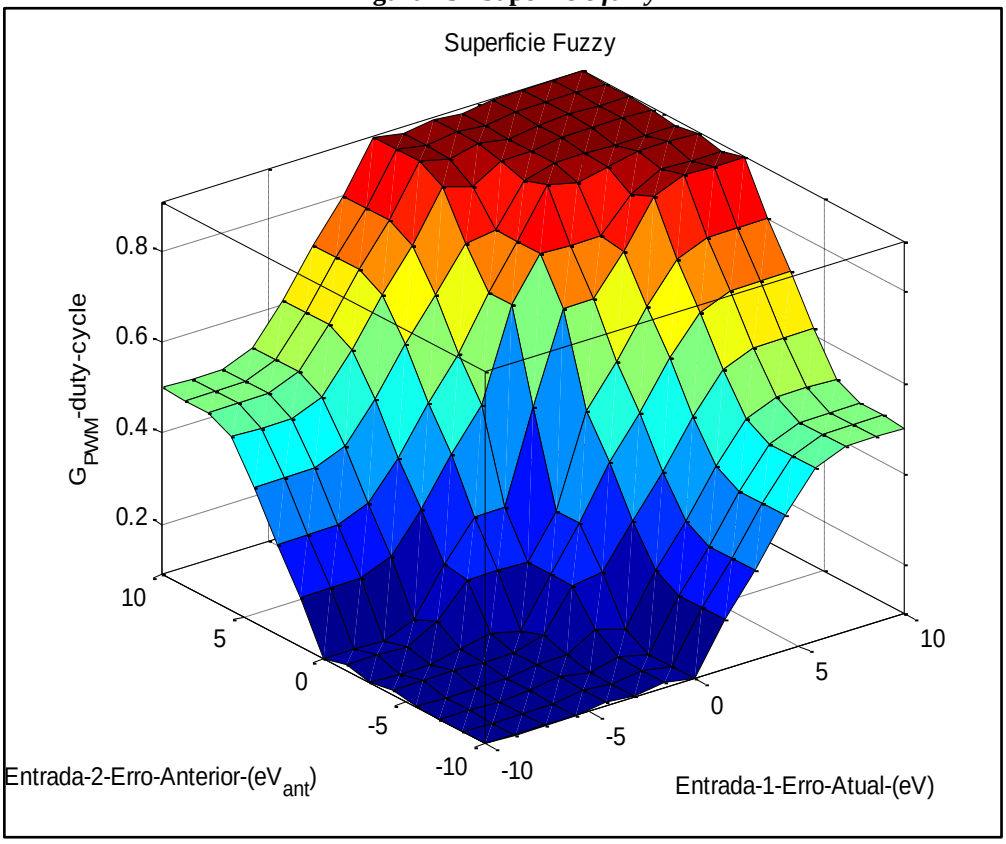
Tabela 2 - Conjunto de regras.

		Erro Atual				
		MN	**PN**	**ZR**	**PP**	**MP**
Erro Anterior	**MN**	MB	MB	MB	B	M
	PN	MB	MB	B	M	A
	ZR	MB	B	---	A	MA
	PP	B	M	A	MA	MA
	MP	M	A	MA	MA	MA

Fonte: Autoria própria.

A representação gráfica do sistema de controle, é descrita na Figura 15, denominada de superfície *fuzzy*, é gerada pelo MATLAB® e possibilita uma melhor compreensão da arquitetura do sistema *fuzzy*.

Figura 15– Superfície *fuzzy*



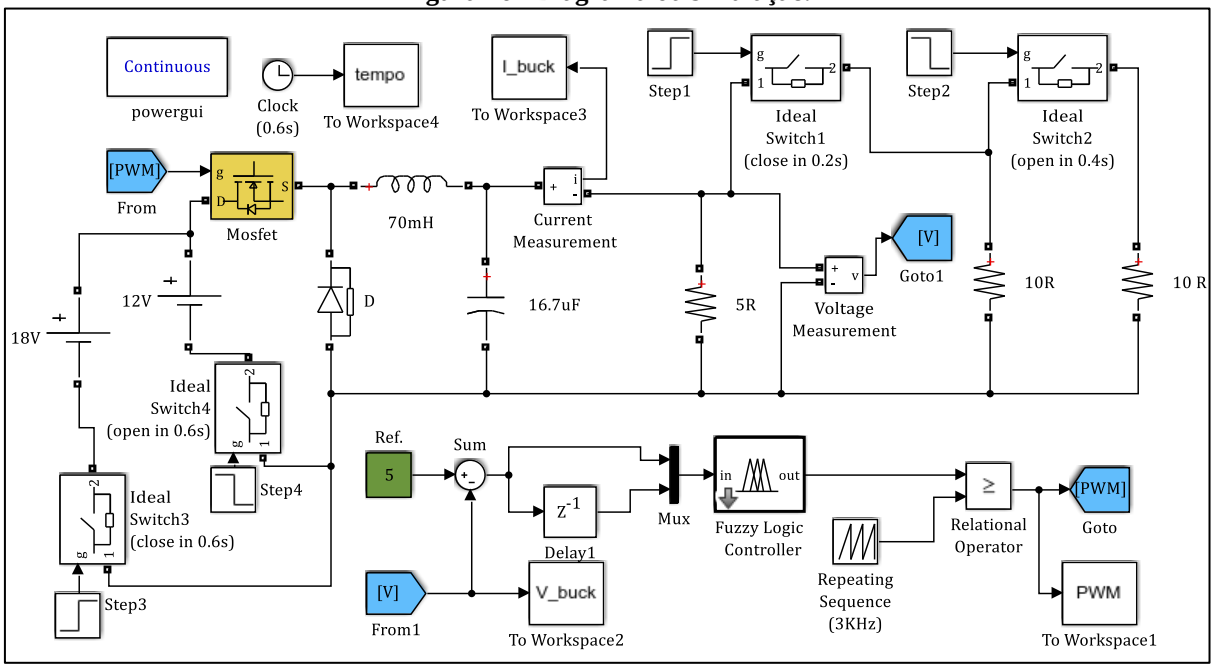
Fonte: Autoria própria.

3.3 Informações da simulação no *Simulink*

Utilizou-se como parâmetro, na simulação, um tempo de 0.8 segundos de duração total, uma variação de carga a cada 0.2 segundos e por último um aumento de 50% na tensão de entrada do *buck*. A carga inicialmente de 5 Ω, demanda uma corrente de 1 A, a demanda de corrente dobra após 0.2 segundos e em 0.4 segundos a carga muda para 3.33 Ω, consumindo uma corrente de 1.5 A. O diagrama esquemático da simulação realizada está ilustrado com detalhes na Figura 16. A dinâmica da simulação, resume-se na sequência a seguir:

0. Tensão de entrada de 12V, comutador (*Ideal Switch)* 2 e 4 fechados e 1 e 3 abertos;
1. O conversor *buck* e o sistema *fuzzy* de controle entra em operação com carga de 5 Ω;
2. *Ideal Switch1* é fechado em 0.2 segundos e a carga equivalente é de 2.5 Ω;
3. *Ideal Switch2* é aberto em 0.4 segundos e a carga equivalente é de 3.33 Ω;
4. *Ideal Switch4* abre e *Ideal Switch3* fecha, em 0.6 segundos, tensão de entrada é de 18V;
5. Fim da simulação em 0.8 segundos.

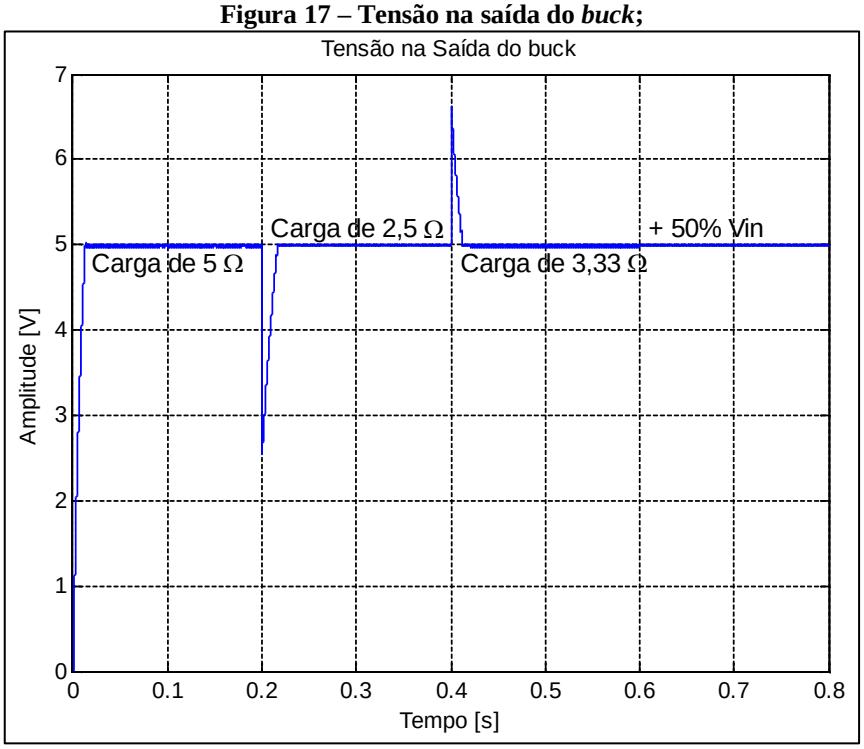
Figura 16 - Diagrama da simulação.



Fonte: Autoria própria.

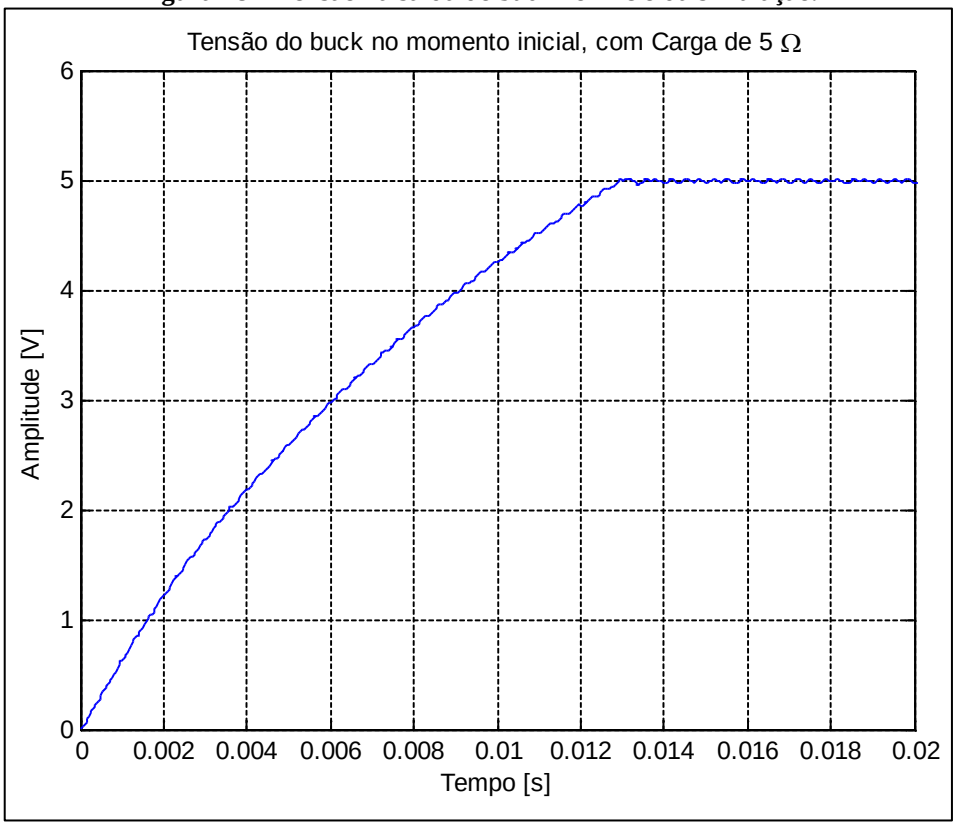
O diagrama representado na Figura 16 foi simulado e os dados exportados para o ambiente de trabalho do *MATLAB*® através dos blocos *"To Workspace"* com as respectivas variáveis de interesse, de tal modo que, a partir do *script* disponível no Apêndice A, foram geradas as Figuras de 17 a 23, que demostram um desempenho satisfatório do controlador *fuzzy* aplicado ao *buck.*

Figura 17 – Tensão na saída do *buck*;



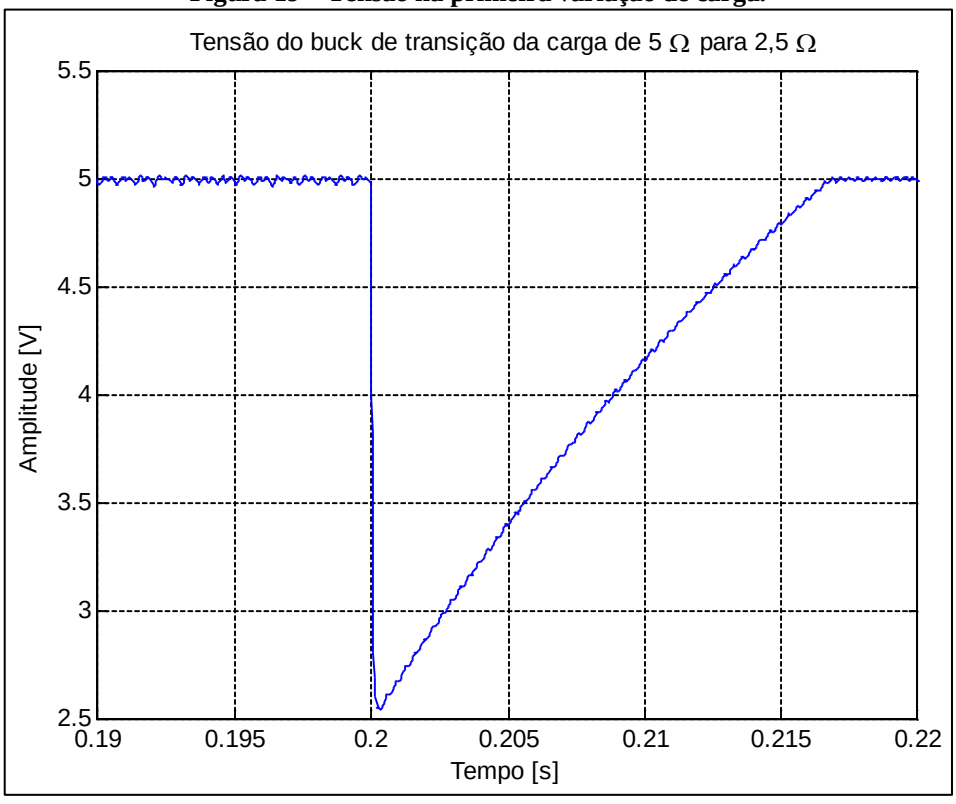
Fonte: Autoria própria.

Figura 18 – Tensão na saída do *buck* no início da simulação.



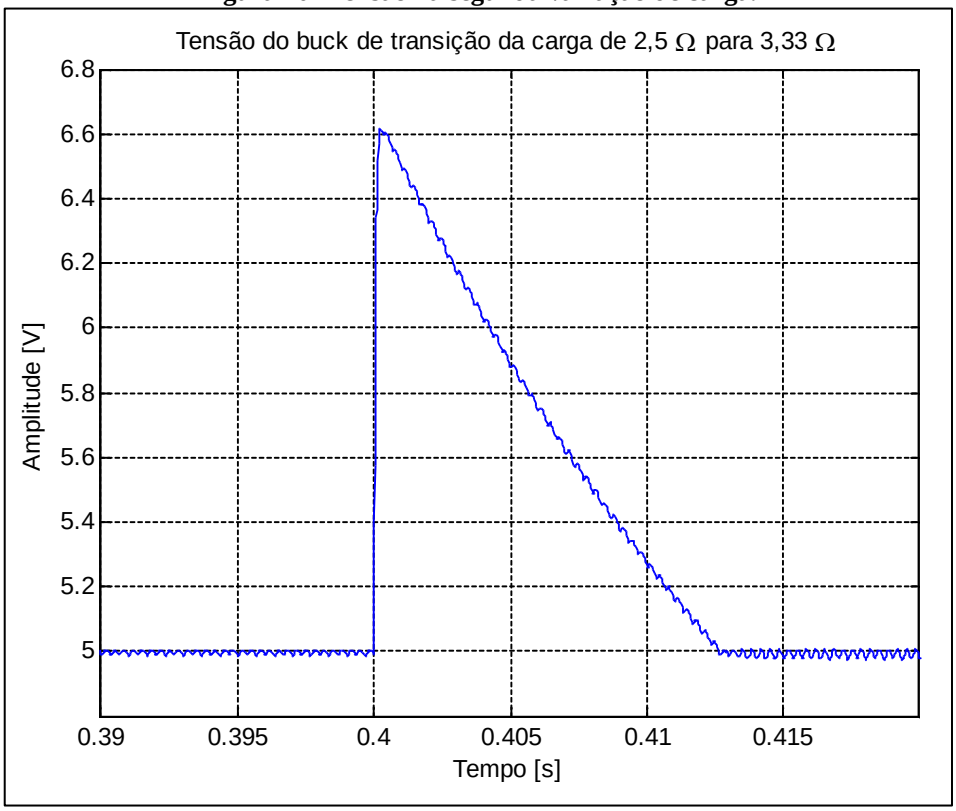
Fonte: Autoria própria.

Figura 19 – Tensão na primeira variação de carga.



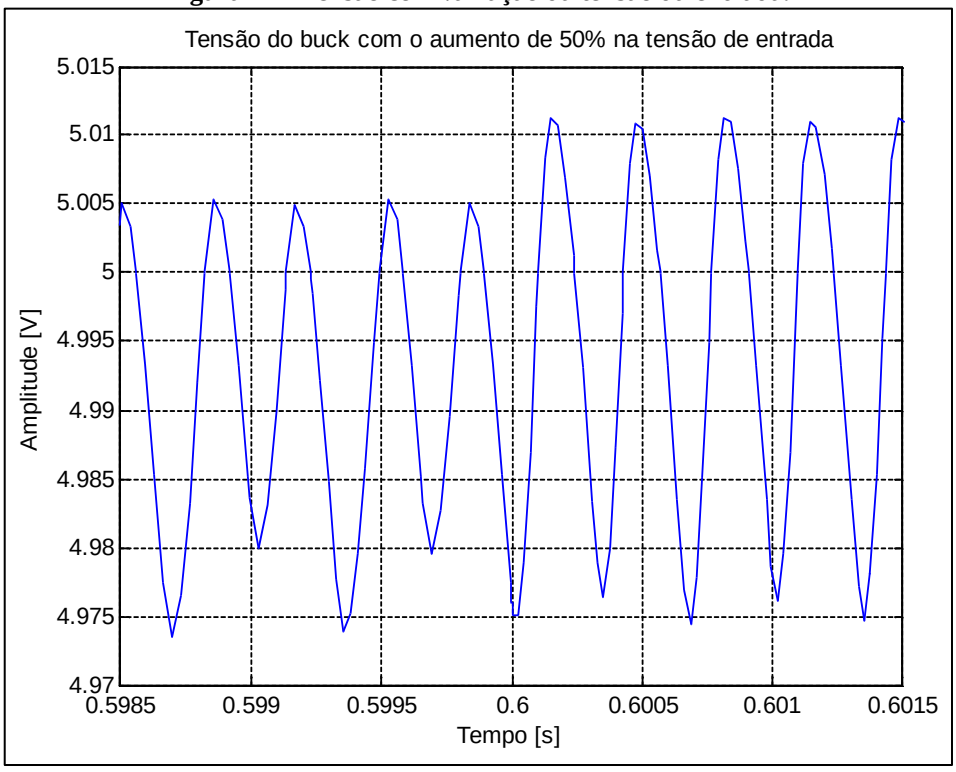
Fonte: Autoria própria.

Figura 20 -Tensão na segunda variação de carga.



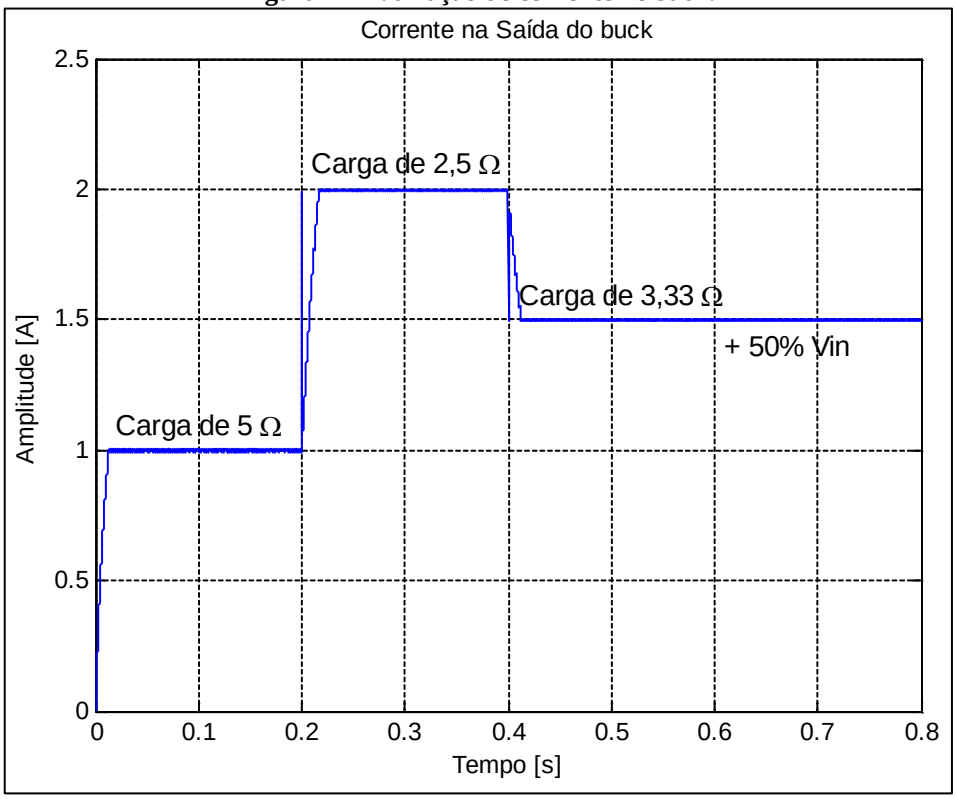
Fonte: Autoria própria.

Figura 21 - Tensão com variação da tensão da entrada.



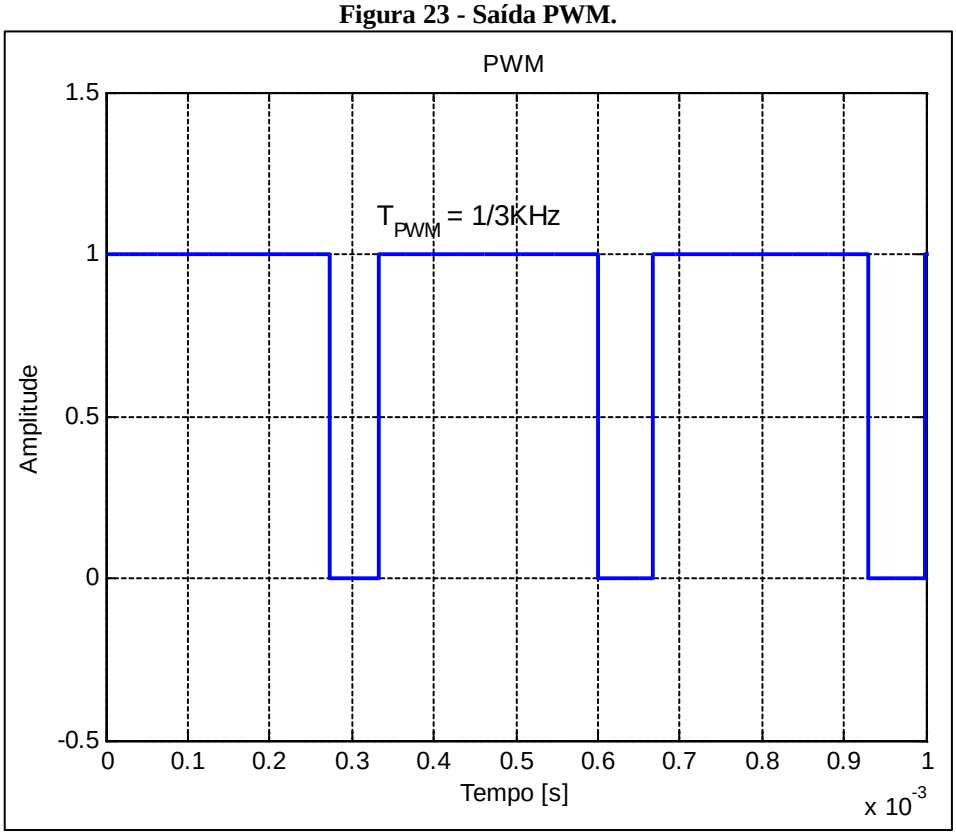
Fonte: Autoria própria.

Figura 22 - Variação de corrente no *buck*.



Fonte: MATLAB/Simulink®.

Figura 23 - Saída PWM.



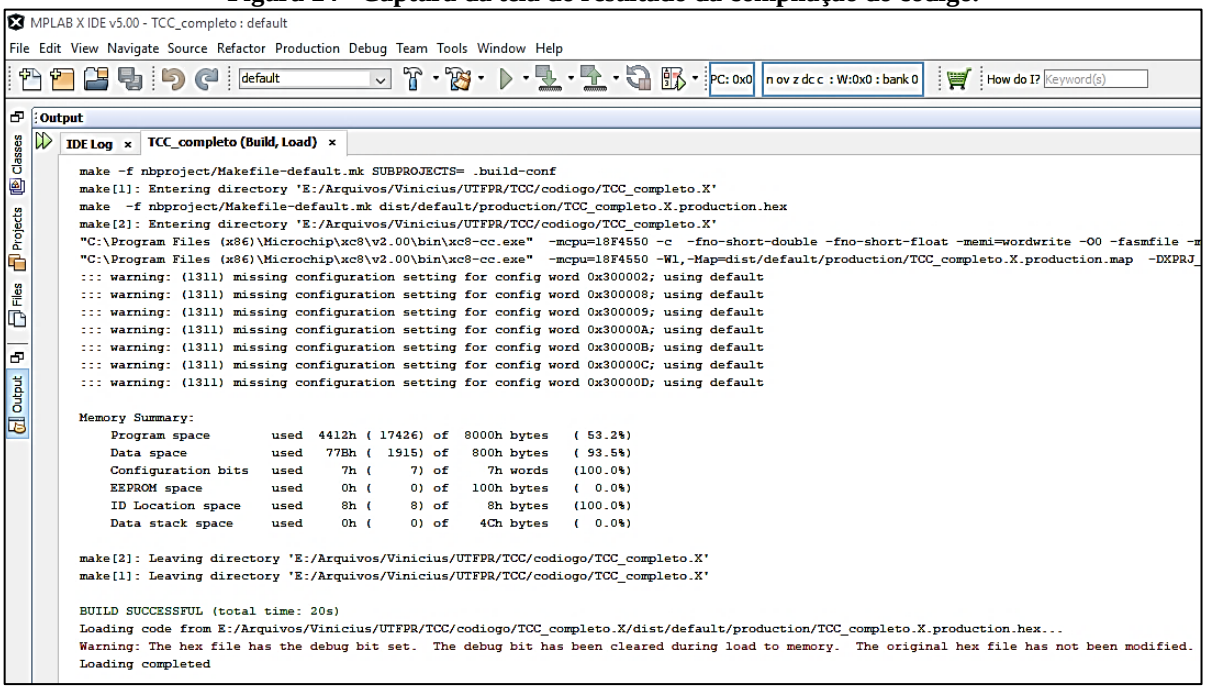
Fonte: MATLAB/Simulink®.

3.4 Desenvolvimento do firmware para o PIC18F4550

A partir do sistema implementado no S*imulink/MATLAB*® como descrito na Seção 3.2, baseando-se no código fonte e trabalho de Fonseca Sobrinho (2017), foi desenvolvido um código em linguagem C no ambiente de desenvolvimento do software MPLAB® X (Microchip, 2018), disponível no Apêndice B. O código foi baseado na arquitetura do PIC18F4550 (Microchip, 2006). A Figura 24 mostra a imagem capturada da tela de resultado da compilação do código.

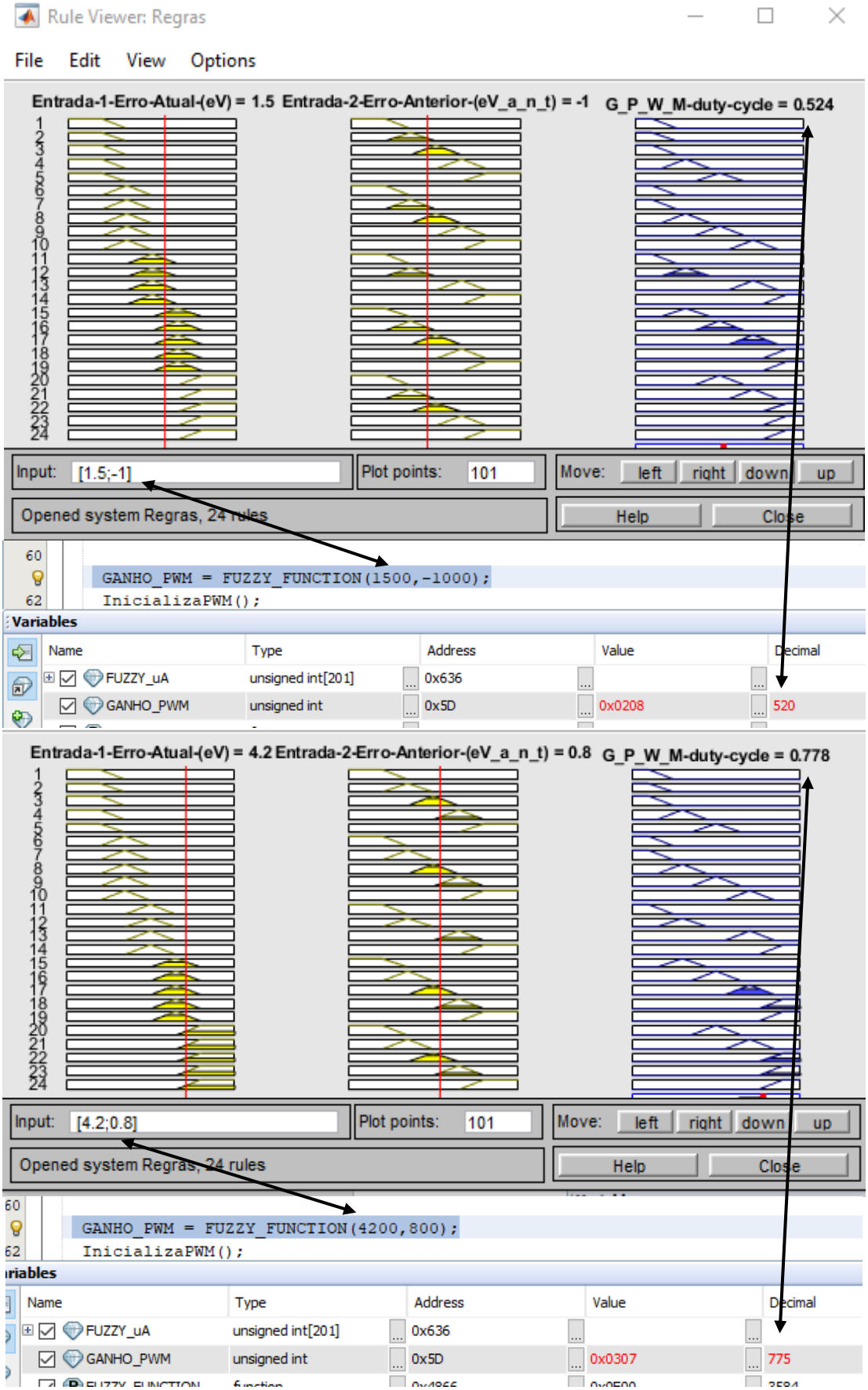
O MPLAB® X permite a simulação do código, de modo que é possível acompanhar o valor das variáveis, a cada linha executada. Assim foi possível validar os resultados obtidos do código implementado, com os resultados da *Fuzzy Logic Toolbox*™ do *MATLAB*®, como pode ser observado pelas capturas de tela disponíveis na Figura 25. É importante mencionar, os valores de tensão de entrada da função *fuzzy* implementada são em mV (mili Volts).

Figura 24 - Captura da tela do resultado da compilação do código.



Fonte: Autoria própria.

Figura 255 - Capturas de tela no processo de validação do código.



Fonte: Autoria própria.

4. RESULTADOS

4.1 Resultados comparativos

Para efeito comparativo das simulações, os resultados do código implementado e o MATLAB®, foi criado uma matriz com sessenta e quatro combinações possíveis das entradas para o respectivo valor de *duty-cycle*, para ambas situações, estão descritos pela Tabela 3.

Tabela 3 - Resultados comparativos entre o MPLAB® X e MATLAB®

			Erro Anterior Absoluto da Tensão (eV_anterior)							
MPLAB® X	índice y	0	1	2	3	4	5	6	7
	índice x	VALOR(mV)	**-10000**	**-7143**	**-4286**	**-1429**	**1428**	**4285**	**7142**	**9999**
Erro Absoluto Atual (eV_atual)	0	**-10000**	90	90	105	105	200	385	500	500
	1	**-7143**	90	90	105	105	200	385	500	500
	2	**-4286**	100	100	100	200	310	500	610	610
	3	**-1429**	105	105	205	205	500	685	795	795
	4	**1428**	200	200	310	500	790	790	890	890
	5	**4285**	385	385	500	685	790	895	895	895
	6	**7142**	500	500	610	795	890	895	905	905
	7	**9999**	500	500	610	795	890	895	905	905
			Erro Anterior Absoluto da Tensão (eV_anterior)							
MATLAB®	índice y	0	1	2	3	4	5	6	7
	índice x	VALOR(V)	**-10,000**	**-7,143**	**-4,286**	**-1,429**	**1,428**	**4,285**	**7,142**	**9,999**
Erro Absoluto Atual (eV_atual)	0	**-10,000**	0,090	0,090	0,102	0,104	0,200	0,390	0,500	0,500
	1	**-7,143**	0,090	0,090	0,102	0,104	0,200	0,390	0,500	0,500
	2	**-4,286**	0,102	0,102	0,102	0,209	0,312	0,500	0,610	0,610
	3	**-1,429**	0,104	0,104	0,209	0,205	0,500	0,688	0,800	0,800
	4	**1,428**	0,200	0,200	0,312	0,500	0,795	0,791	0,896	0,896
	5	**4,285**	0,390	0,390	0,500	0,688	0,791	0,898	0,898	0,898
	6	**7,142**	0,500	0,500	0,610	0,800	0,896	0,898	0,910	0,910
	7	**9,999**	0,500	0,500	0,610	0,800	0,896	0,898	0,910	0,910

Fonte: Autoria própria.

Os valores são acessados pelo seu respectivo índice *duty-cycle* (x,y), por exemplo, para a situação que *eV_atual* é -7143 (índice x = 1) e o *eV* é 1428 (ídice y = 4), o valor de *duty-cycle* retornado será 200 (*duty-cycle*(1,4)).

Como observado na Tabela 3 os resultados são bem próximos, porém a diferença é que no *firmware* os valores são números inteiros, os valores dos erros de tensão são dados em

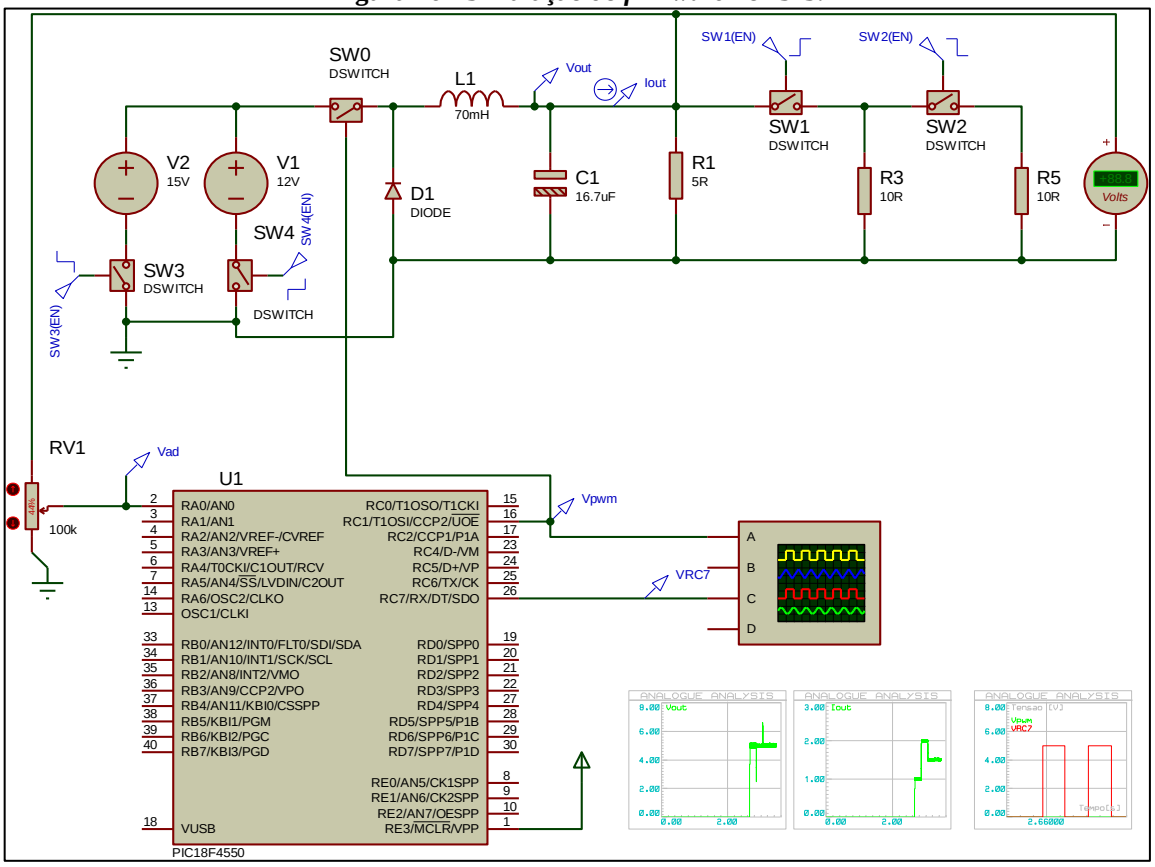
mV e os valores de ganho do *duty-cycle* de 0 a 1000. Contudo, os valores obtidos são bem próximos, apesar dos arredondamentos gerados.

4.2 Simulação do *firmware* no *ISIS/Proteus*

Para observar o comportamento dinâmico do sistema proposto, foi utilizado o *ISIS/Proteus*® (Labcenter Electronics, 2010), que é um ambiente de simulação que mais se aproxima do ambiente real, com a possibilidade de simulação de *firmwares* de diversos microcontroladores e circuitos integrados. Sendo assim, a ideia foi replicar a topologia e as características do circuito simulado no *Simulink*, como pode ser observado na Figura 26. Assim foram realizadas as mesmas variações nas cargas e na tensão de entrada do *buck*.

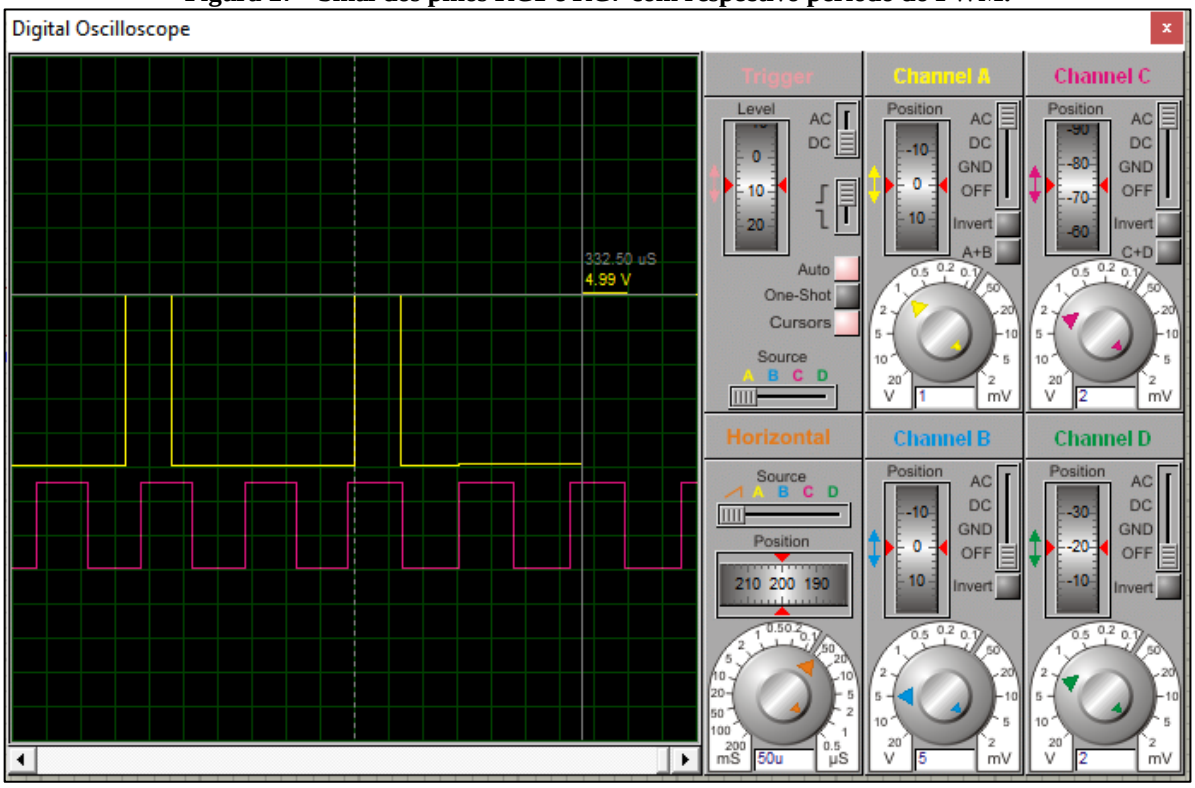
É importante ressaltar que o sistema *fuzzy* consome um tempo considerável de processamento do PIC, que no início dos testes não convergiam bons resultados, pois é necessário que o sistema *fuzzy* gere uma resposta em um tempo inferior ao período do sinal do PWM gerado, que é de aproximadamente 333,33 µs (1/3KHz), porém cada execução da rotina do sistema *fuzzy*, demorava cerca de 200 ms. Assim foi necessária uma adaptação no código, para que o sistema *fuzzy* carregasse, em um primeiro momento, uma matriz com algumas possíveis combinações de erro atual e erro anterior da tensão e o respectivo valor de *duty-cycle*. Devido a limitação de memória do PIC a matriz gerada (Tabela 3) tem dimensões 8x8, ou seja, 64 combinações igualmente espaçadas, geradas a partir do universo de discurso das duas entradas e saída do sistema *fuzzy*. Portanto o PIC necessita de um tempo de 2,66 s de inicialização, para o carregamento dos valores, posteriormente ele executa a rotina de repetição de controle, que realiza a leitura da tensão na saída do *buck*, calcula o erro atual e juntamente com o erro anterior, realiza a busca do valor do *duty-cycle* para a combinação mais próxima correspondente. Para medir o intervalo de tempo na simulação, utilizou-se o pino RC7, que muda seu estado lógico a cada ciclo de execução da rotina de repetição. Esse intervalo de tempo na simulação e o período do sinal do *PWM* gerado, pode ser observado na captura de tela do osciloscópio virtual da Figura 27.

Figura 26 - Simulação do *firmware* no ISIS.



Fonte: Autoria própria.

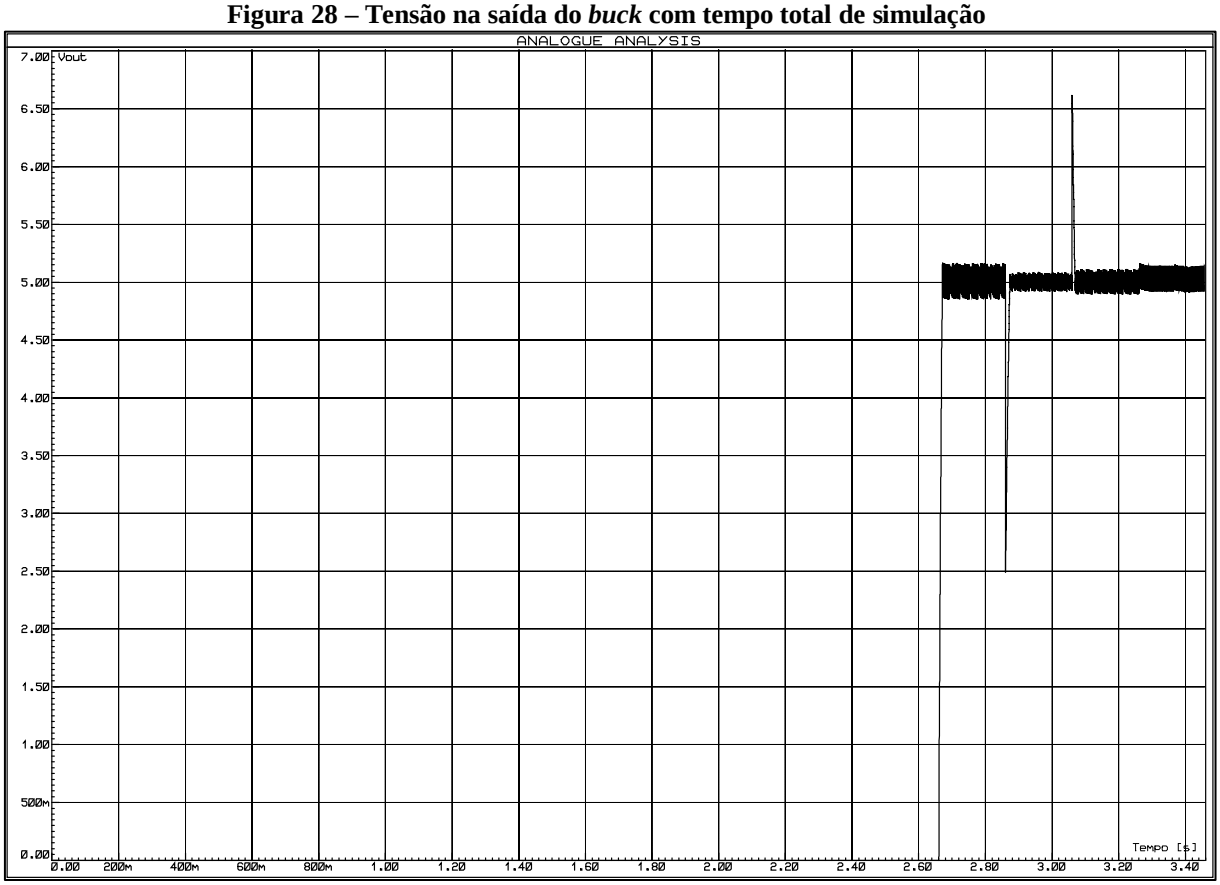
Figura 27 - Sinal dos pinos RC1 e RC7 com respectivo período do PWM.



Fonte: Autoria própria.

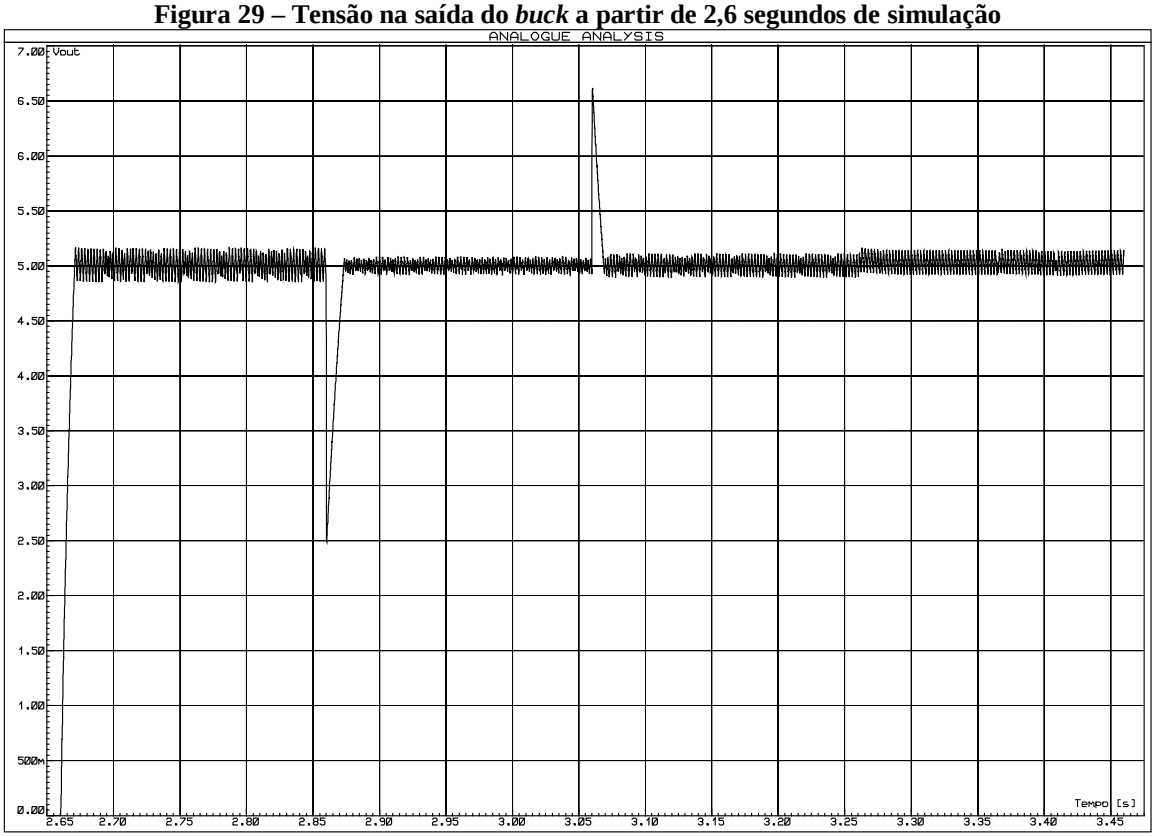
Como pode ser observado nas Figuras 28 a 29, os gráficos se aproximam dos resultados da simulação do S*imulink/MATLAB*®, disponíveis nas Figuras 17 e 23 da Seção 3.3, porém com uma maior variação nos valores de tensão e corrente, que se justifica devido ao tamanho limitado da matriz utilizada para relacionar os valores.

Figura 28 – Tensão na saída do *buck* com tempo total de simulação



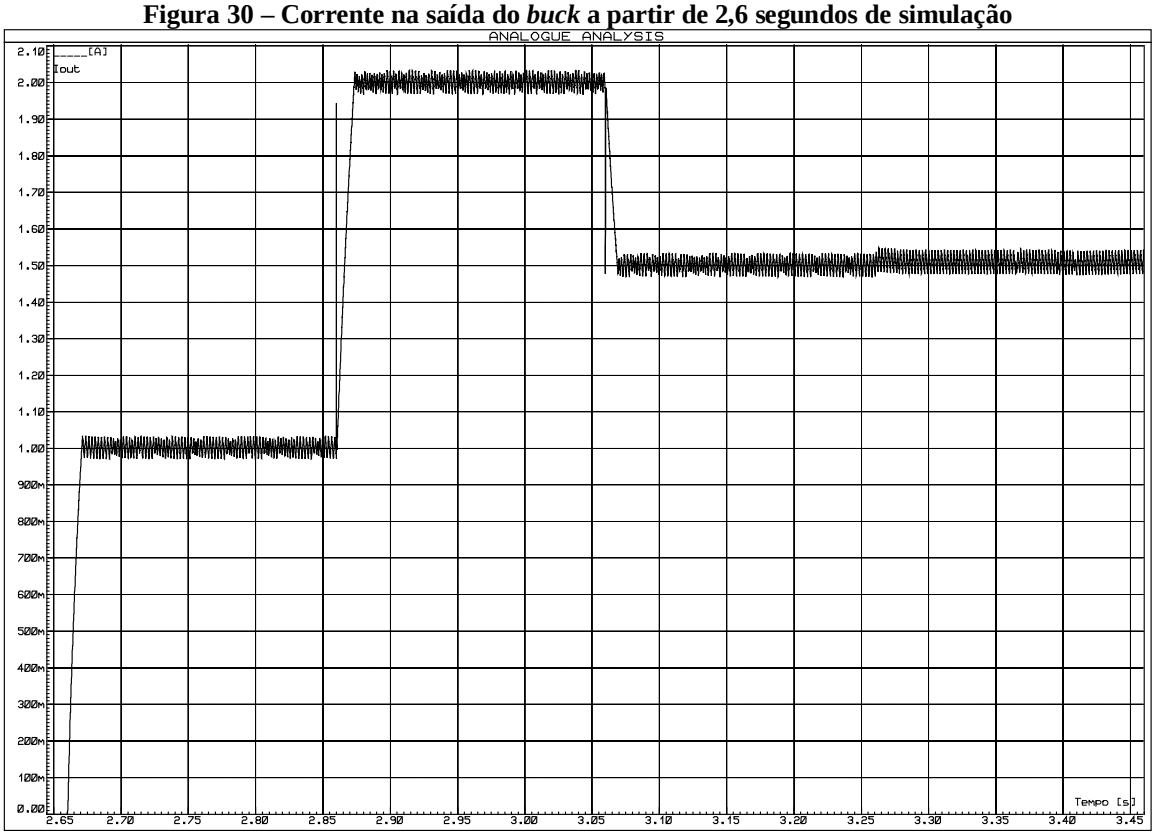
Fonte: Autoria própria.

Figura 29 – Tensão na saída do *buck* a partir de 2,6 segundos de simulação



Fonte: Autoria própria.

Figura 30 – Corrente na saída do *buck* a partir de 2,6 segundos de simulação



Fonte: Autoria própria.

5. CONCLUSÃO

Considerando que o objetivo deste trabalho foi propor uma solução de controle inteligente aliado ao baixo custo, os resultados foram satisfatórios, e dependendo da aplicação, esse controlador inteligente pode ser uma solução. Contudo, a maior importância desse estudo foi a integração de diversas áreas de conhecimentos obtidos em todos os períodos do curso de Engenharia Elétrica, que se consolidaram, com maior intensidade, no momento de desenvolvimento deste trabalho de conclusão de curso.

É importante ressaltar que o sistema *fuzzy* implementado não é exatamente o sistema simulado no *Simulink*, mas sim uma aproximação, para ser executado a partir do PIC18F4550. Devido ao tempo de resposta, se fez necessário a adaptação no código, como mencionado no Seção 4.2, que durante a inicialização do microcontrolador, uma matriz com os valores é gerada, o que demandou um tempo de aproximadamente 2,6 segundos para que o controle entrassem em operação, como pode ser observado na Figura 28. Sendo assim essa matriz pode ser gerada previamente, com a execução do algoritmo do sistema *fuzzy* de outra maneira, exterior ao microcontrolador e posteriormente armazenada no mesmo, de modo a evitar esse tempo de inicialização. Contudo para uma aplicação que não necessite de um tempo de resposta tão alto, esse sistema pode ser utilizado.

O tipo do conversor CC-CC *buck* utilizado nesse trabalho não necessariamente necessita de técnicas de controle inteligente, como *fuzzy*, pois sua modelagem não é complicada. Porém, os estudos realizados neste trabalho podem ser aprimorados e aprofundados de modo que possibilite a implementação em sistemas de modelagem mais complexa.

REFERÊNCIAS

BARBI, Ivo; MARTINS, Denizar Cruz. **Eletrônica de potência:** conversores CC-CC básicos não isolados. 2.ed. Florianopolis, SC: Ed. do Autor, 2006. 380p. ISBN 8590520323.

CAVALLARO, F. A. **Takagi-Sugeno Fuzzy Inference System for Developing a Sustainability Index of Biomass**. Sustainability 2015, 7, 12359-12371. Disponível em: < http://www.mdpi.com/2071-1050/7/9/12359>. Acesso em: 12 mai. 2018.

CHO, H.-C.; Lee, D.H.; Ju, H.; Park, H.-C.; Kim, H.-Y.; Kim, K.S. **Fire Damage Assessment of Reinforced Concrete Structures Using Fuzzy Theory.** Appl. Sci. 2017, 7, 518. Disponível em: <http://www.mdpi.com/2076-3417/7/5/518>. Acesso em: 12 mai. 2018.

FONSECA SOBRINHO, A. S. **Sistemas microcontrolados**. Notas de aula: Sistemas Microcontrolados, UTFPR. Cornélio Procópio, 2017.

FONSECA SOBRINHO, A. S. **Utilização de lógica fuzzy em um sistema embarcado para classificação da resposta de sensores de fluxo de ar.** UTFPR. Cornélio Procópio, 2017.

GUPTA T., R. R. Boudreaux, R. M. Nelms and J. Y. Hung, "**Implementation of a fuzzy controller for DC-DC converters using an inexpensive 8-b microcontroller,**" in *IEEE Transactions on Industrial Electronics*, vol. 44, no. 5, pp. 661-669, Oct 1997.

ISIS PROTEUS. Versão 7.8 SP2. 1989-2010. Labcenter Eletronics, Inc.

KOLAKOWSKI, Terry, **"Fuzzy Logic Control of a Switched-Inductor PWM DC-DC Buck Converter in CCM"**. Browse all Theses and Dissertations. 2009, Paper 300. Disponível em: < https://corescholar.libraries.wright.edu/cgi/viewcontent.cgi?article=1439&context=etd_all> Acesso em: 21 mai. 2018.

LEE, C. C., **"Fuzzy logic in control systems: fuzzy logic controller. I,"** in IEEE Transactions on Systems, Man, and Cybernetics, vol. 20, no. 2, pp. 404-418, Mar/Apr 1990.

LILLY, J. H. **Fuzzy control and identification.** John Wiley, 2010. 231p. Online ISBN: 9780470874240. Disponível em: < https://onlinelibrary.wiley.com/doi/book/10.1002/9780470874240>. Acesso em: 23 fev. 2019.

MAMDANI, E.H. S. **Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies.** Volume 7, Issue 1, 1975, Pages 1-13, ISSN 0020-7373. Disponível em: < http://www.sciencedirect.com/science/article/pii/S0020737375800022>. Acesso em: 12 mai. 2018.

MATLAB. Versão R2012b. 1994-2012, The MathWorks, Inc.

MICROCHIP. **PIC18F2455/2550/4455/4550 Data Sheet.** Microchip Technology Inc, 2006. Disponível em: < http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf> Acesso em: 01 mai. 2018.

MPLAB X IDE. Versão 5.00. 2010-2018, Microchip Technology, Inc.

OLIVEIRA NETTO, A. A. de. **Metodologia da pesquisa científica:** guia prático para a apresentação de trabalhos acadêmicos. 3. ed. rev. e atual. Florianópolis: Visual Books, 2008.

PERRY A. G., G. Feng, Y. F. Liu and P. C. Sen, **"A Design Method for PI-like Fuzzy Logic Controllers for DC–DC Converter,"** in *IEEE Transactions on Industrial Electronics*, vol. 54, no. 5, pp. 2688-2696, Oct. 2007.

PERTY, Clóvis Antônio. **Introdução aos Conversores CC-CC.** INEP/EEL, UFSC. Florianópolis Ed. Do Autor, agosto de 2001 17p. Disponível em: <http://www.professorpetry.com.br/Bases_Dados/Apostilas_Tutoriais/Introducao_Conversores_CC_CC.pdf>. Acesso em: 01 mai. 2018.

SIMULINK. Versão R2012b. 1994-2012, The MathWorks, Inc.

SPATTI, D. H. **Inferência fuzzy**. Notas de aula: Sistemas Inteligentes Aplicados a Engenharia, UTFPR. Cornélio Procópio, 2017.

TEXAS INSTRUMENTS. **Application Report: Understanding Buck Power Stages in Switchmode Power Supplies.** Texas Instruments Incorporated, Copyright Mar. 1999. 36 p. Disponível em: < http://www.ti.com/lit/an/slva057/slva057.pdf >. Acesso: em 05 mai. 2018.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ. Sistema de Bibliotecas. **Normas para elaboração de trabalhos acadêmicos.** Curitiba: UTFPR, 2009.

ZADEH, L.A. **Fuzzy sets.** Information and Control, Volume 8, Issue 3, 1965, Pages 338-353, ISSN 0019-9958. Acesso em: 06 mai. 2018. Disponível em: <http://www.sciencedirect.com/science/article/pii/S001999586590241X>.

APÊNDICE

APÊNDICE A – Script do MATLAB utilizado para geração das figuras

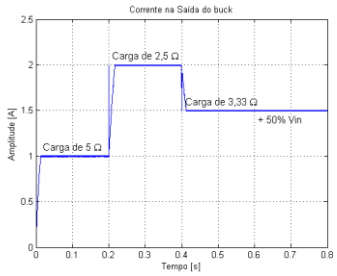
```
%Script para a geração dos gráficos
clc; close all;

%Corrente
figure
plot(tempo,I_buck,'b')
title('Corrente na Saída do buck')
text(0.02,1.1,'Carga de 5 \Omega','FontSize',11)
hold on
text(0.21,2.1,'Carga de 2,5 \Omega','FontSize',11)
text(0.41,1.6,'Carga de 3,33 \Omega','FontSize',11)
text(0.61,1.4,'+ 50% Vin','FontSize',11)
xlabel('Tempo [s]')
ylabel('Amplitude [A]')
grid on

%Tensão
figure
plot(tempo,V_buck,'b')
title('Tensão na Saída do buck')
text(0.02,4.8,'Carga de 5 \Omega','FontSize',11)
hold on
text(0.21,5.2,'Carga de 2,5 \Omega','FontSize',11)
text(0.41,4.8,'Carga de 3,33 \Omega','FontSize',11)
text(0.61,5.2,'+ 50% Vin','FontSize',11)
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
grid on

%Momentos de Variação da Tensão
figure
%de 0.0 a 0.002s
plot(tempo,V_buck,'b')
title('Tensão do buck no momento inicial, com Carga de 5
\Omega')
xlim([0.0 0.02])
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
grid on
%de 0.19s a 0.22s
figure
plot(tempo,V_buck,'b')
title('Tensão do buck de transição da carga de 5 \Omega para
2,5 \Omega')
xlim([0.19 0.22])
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
grid on
```

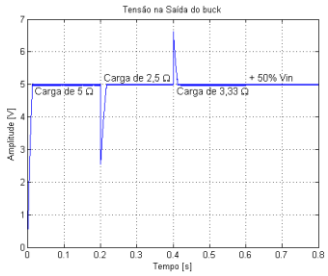
```
%de 0.39s a 0.42s
figure
plot(tempo,V_buck,'b')
title('Tensão do buck de transição da carga de 2,5 \Omega para
3,33 \Omega')
xlim([0.39 0.42])
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
grid on
%de 0.5985s a 0.6015s
figure
plot(tempo,V_buck,'b')
title('Tensão do buck com o aumento de 50% na tensão de
entrada')
xlim([0.5985 0.6015])
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
grid on

%saída PWM
figure
plot(tempo,PWM,'b','LineWidth',1.5)
title('PWM')
text(0.00033,1.1,'T_P_W_M = 1/3KHz','FontSize',11)
xlabel('Tempo [s]')
ylabel('Amplitude')
xlim([0 0.001])
ylim([-0.5 1.5])
grid on


%Figuras referentes ao sistema Fuzzy

%Carregando os dados gerados a partir do Fuzzy Toolbox
rules_fuzzy = readfis('regras.fis');

figure %Topologia do sistema Fuzzy
plotfis(rules_fuzzy)

figure %Funções de pertinência das entrada 1 e 2
plotmf(rules_fuzzy,'input',1)
title('Entradas 1 e 2 - Erro Absoluto da Tensão, Atual e
Anterior')
xlabel('(Vref-V), (Vref-V(ant) [V]')
ylabel('Grau de Pertinência')
grid on

figure %Funções de pertinência da saída
plotmf(rules_fuzzy,'output',1)
title('Saída - duty-cycle PWM')
xlabel('duty-cycle [%]')
```

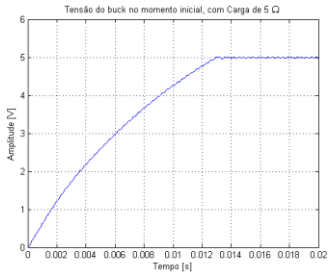
```
ylabel('Grau de Pertinência')
grid on

figure %Superfície Fuzzy
gensurf(rules_fuzzy)
title('Superficie Fuzzy')
box on

showrule(rules_fuzzy) %Regras
```

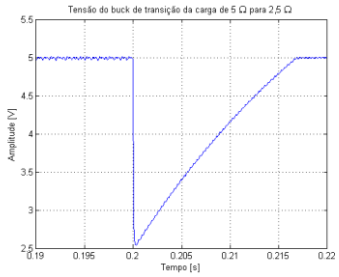
```
ans =
1. If (Entrada-1-Erro-Atual-(eV) is MN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MN) then (G_P_W_M-duty-cycle is MB) (1)
2. If (Entrada-1-Erro-Atual-(eV) is MN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PN) then (G_P_W_M-duty-cycle is MB) (1)
3. If (Entrada-1-Erro-Atual-(eV) is MN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is ZR) then (G_P_W_M-duty-cycle is MB) (1)
4. If (Entrada-1-Erro-Atual-(eV) is MN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PP) then (G_P_W_M-duty-cycle is B) (1)
5. If (Entrada-1-Erro-Atual-(eV) is MN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MP) then (G_P_W_M-duty-cycle is M) (1)
6. If (Entrada-1-Erro-Atual-(eV) is PN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MN) then (G_P_W_M-duty-cycle is MB) (1)
7. If (Entrada-1-Erro-Atual-(eV) is PN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PN) then (G_P_W_M-duty-cycle is MB) (1)
8. If (Entrada-1-Erro-Atual-(eV) is PN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is ZR) then (G_P_W_M-duty-cycle is B) (1)
9. If (Entrada-1-Erro-Atual-(eV) is PN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PP) then (G_P_W_M-duty-cycle is M) (1)
10. If (Entrada-1-Erro-Atual-(eV) is PN) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MP) then (G_P_W_M-duty-cycle is A) (1)
11. If (Entrada-1-Erro-Atual-(eV) is ZR) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MN) then (G_P_W_M-duty-cycle is MB) (1)
12. If (Entrada-1-Erro-Atual-(eV) is ZR) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PN) then (G_P_W_M-duty-cycle is B) (1)
13. If (Entrada-1-Erro-Atual-(eV) is ZR) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PP) then (G_P_W_M-duty-cycle is A) (1)
14. If (Entrada-1-Erro-Atual-(eV) is ZR) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MP) then (G_P_W_M-duty-cycle is MA) (1)
15. If (Entrada-1-Erro-Atual-(eV) is PP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MN) then (G_P_W_M-duty-cycle is B) (1)
16. If (Entrada-1-Erro-Atual-(eV) is PP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PN) then (G_P_W_M-duty-cycle is M) (1)
17. If (Entrada-1-Erro-Atual-(eV) is PP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is ZR) then (G_P_W_M-duty-cycle is A) (1)
18. If (Entrada-1-Erro-Atual-(eV) is PP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PP) then (G_P_W_M-duty-cycle is MA) (1)
19. If (Entrada-1-Erro-Atual-(eV) is PP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MP) then (G_P_W_M-duty-cycle is MA) (1)
20. If (Entrada-1-Erro-Atual-(eV) is MP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MN) then (G_P_W_M-duty-cycle is M) (1)
```

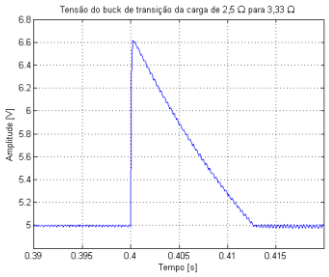
```
21. If (Entrada-1-Erro-Atual-(eV) is MP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PN) then (G_P_W_M-duty-cycle is A) (1)
22. If (Entrada-1-Erro-Atual-(eV) is MP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is ZR) then (G_P_W_M-duty-cycle is MA) (1)
23. If (Entrada-1-Erro-Atual-(eV) is MP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is PP) then (G_P_W_M-duty-cycle is MA) (1)
24. If (Entrada-1-Erro-Atual-(eV) is MP) and (Entrada-2-Erro-
Anterior-(eV_a_n_t) is MP) then (G_P_W_M-duty-cycle is MA) (1)
```

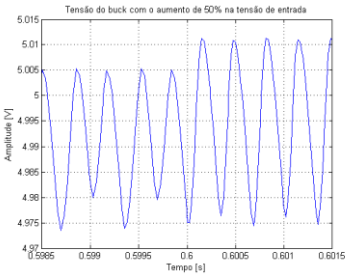


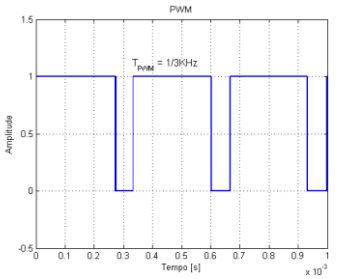


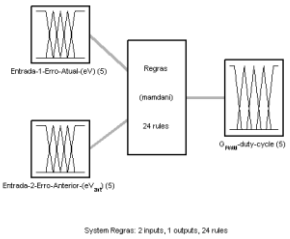


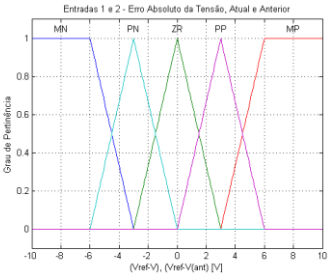


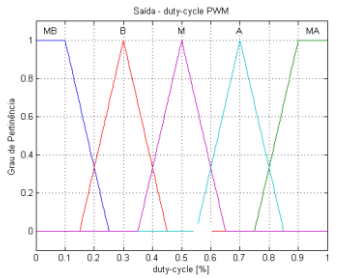


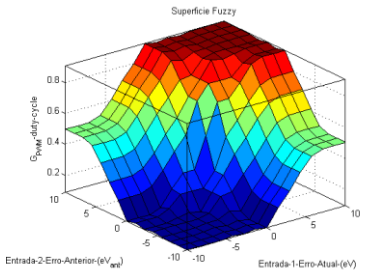












APÊNDICE B – Código do MPLAB® X em linguagem C

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>

/************** COMENTADO PARA TESTE PRÁTICO ***********************/

// Configurções
#pragma config PLLDIV = 5 // PLL para 20MHz
#pragma config CPUDIV = OSC1_PLL2 // Oscilador principal a 48MHz
#pragma config FOSC = HSPLL_HS // Oscilador HS para CPU e USB
#pragma config WDT = OFF // Watchdog desativado
#pragma config PBADEN = OFF // Pinos do PORTB começam como digitais
#pragma config LVP = OFF // Desabilita gravação em baixa tensão
#pragma config DEBUG = ON // habilita debug

// Defines fuzzy
#define FUZZY_UNI              1000   // Universo de discurso (0-1000) [1000]
#define FUZZY_PER              1000   // Grau de pertinência (0-1000) [1000]
#define FUZZY_MEP              0      // Menor ponto de discretização [0]
#define FUZZY_MAP              200    // Maior ponto de discretização [200]
#define NP_MAT         8      // Número de pontos armazenados pela matriz ganho NxN
#define K              1//11.73020528 // Constante de transformação  da Conversão A/D   -> K =(tensão máxima no circuito em mV)12000/1023(Valor máximo conversor AD em decimal)

//Variáveis Globais
unsigned int FUZZY_uGANHO[3][201];
unsigned int FUZZY_uA[201];
int MATRIZ_GANHO[NP_MAT][NP_MAT];
int VETOR_ERRO[NP_MAT];
int LER_AD=0;
unsigned int GANHO_PWM=100;
int ERRO_V_ANT=0;
int ERRO_V=0;
int SET=5000/K; //SET = Valor desejado na saída do conversor buck, em mV/K
int xk = 0;
int yk = 0;

//Funções
unsigned int FUZZY_FUNCTION(long FUZZY_ERRO_V, long FUZZY_ERRO_V_ANT);
unsigned int FUZZY_TRIAN(long a, long m, signed long b, long x);
unsigned int FUZZY_TRAP1(long a, long m, signed long b, long x);
unsigned int FUZZY_TRAP2(long a, long m, signed long b, long x);
unsigned int FUZZY_MIN(unsigned int a, unsigned int b);
unsigned int FUZZY_MAX(unsigned int a, unsigned int b);

```

```
45 void InicializaPWM(void);
46 void InicializaConvAD(void);
47 void SetaFreqPWM(void);
48 void PWMDutyCycle(void);
49 void GeraDados(void);
50
51 //********** Função Main *************
52 void main (void){
53   TRISCbits.TRISC1 = 0;  // Pino RC1 é saída PWM
54   PORTCbits.RC1 = 0;
55   TRISCbits.TRISC7 = 0;  // Pino RC7 é saída
56   PORTCbits.RC7 = 0;
57   GeraDados(); // Procedimento para criar a base de dados do sistema Fuzzy
58   //GANHO_PWM = FUZZY_FUNCTION(4200,800);
59   InicializaPWM();
60   SetaFreqPWM();
61   InicializaConvAD();
62
63   for(;;){
64     PORTCbits.RC7 = ~PORTCbits.RC7;// Utilização do pino RB7 para medir
velocidade de processamento do código
65     ADCON0bits.GO_DONE = 1; // Inicia a conversão A/D
66     LER_AD = (256*ADRESH + ADRESL);
67     //LER_AD = 346;
68     ERRO_V = (SET-LER_AD);
69       for(int i=0;i<NP_MAT;i++){ // Compara os valores de erros com a base de dados
do sistema fuzzy
70         if ((ERRO_V_ANT >= VETOR_ERRO[i])&(ERRO_V_ANT <= VETOR_ERRO[i+1])){
71           xk = i;
72         }
73         if ((ERRO_V >= VETOR_ERRO[i])&(ERRO_V <= VETOR_ERRO[i+1])){
74           yk = i;
75         }
76     }
77     //GANHO_PWM = LER_AD;
78     GANHO_PWM = MATRIZ_GANHO[xk][yk];
79     PWMDutyCycle();
80     ERRO_V_ANT = ERRO_V;
81   }
82 }
83
84 //*********************************************************************
85 //***************************** Função Fuzzy ***************************
86 //*********************************************************************
87 unsigned int FUZZY_FUNCTION(long FUZZY_ERRO_V, long FUZZY_ERRO_V_ANT){
88
89   //*** Variáveis locais
90   unsigned int FUZZY_erroV[5];   //Erro absoluto ATUAL da tensão
91   unsigned int FUZZY_erroV_ant[5]; //Erro absoluto ANTERIOR da tensão
```

```
92
93  long FUZZY_CDANUM;
94  long FUZZY_CDADEN;
95  long FUZZY_CDA;
96
97  long FUZZY_CONT1=0;
98  long FUZZY_CONT2=0;
99 //  long FUZZY_CONT3=0;
100
101  //*** Cálculo da variável de entrada FUZZY_erroV (Erro Absoluto ATUAL da Tensão)
102  FUZZY_erroV[0]=0; FUZZY_erroV[1]=0; FUZZY_erroV[2]=0; FUZZY_erroV[3]=0; FUZZY_erroV[4]=0;
103
104  FUZZY_erroV[0]=FUZZY_TRAP1(-10000/K,-6000/K,-3000/K,FUZZY_ERRO_V);//MN(Muito Negativo)
105  FUZZY_erroV[1]=FUZZY_TRIAN(-6000/K,-3000/K,0,FUZZY_ERRO_V); //PN(Pouco Negatovo)
106  FUZZY_erroV[2]=FUZZY_TRIAN(-3000/K,0,3000/K,FUZZY_ERRO_V); //ZR(Zero))
107  FUZZY_erroV[3]=FUZZY_TRIAN(0,3000/K,6000/K,FUZZY_ERRO_V); //PP(Pouco Positivo)
108  FUZZY_erroV[4]=FUZZY_TRAP2(3000/K,6000/K,10000/K,FUZZY_ERRO_V); //MP(Muito Positivo)
109
110  //*** Cálculo da variável de entrada FUZZY_erroV_ant (Erro Absoluto ANTERIOR da Tensão)
111  FUZZY_erroV_ant[0]=0; FUZZY_erroV_ant[1]=0; FUZZY_erroV_ant[2]=0; FUZZY_erroV_ant[3]=0; FUZZY_erroV_ant[4]=0;
112
113  FUZZY_erroV_ant[0]=FUZZY_TRAP1(-10000/K,-6000/K,-3000/K,FUZZY_ERRO_V_ANT);//MN(Muito Negativo)
114  FUZZY_erroV_ant[1]=FUZZY_TRIAN(-6000/K,-3000/K,0,FUZZY_ERRO_V_ANT); //PN(Pouco Negatovo)
115  FUZZY_erroV_ant[2]=FUZZY_TRIAN(-3000/K,0,3000/K,FUZZY_ERRO_V_ANT); //ZR(Zero))
116  FUZZY_erroV_ant[3]=FUZZY_TRIAN(0,3000/K,6000/K,FUZZY_ERRO_V_ANT); //PP(Pouco Positivo)
117
FUZZY_erroV_ant[4]=FUZZY_TRAP2(3000/K,6000/K,10000/K,FUZZY_ERRO_V_ANT); //MP(Muito Positivo)
118
119  //*** Armazenamento da função de pertinência para as regras que estão ativadas (até 4 regras) com conectivo AND e Implicação(Mamdani)
120  FUZZY_CONT1=0;
121  for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
122   FUZZY_uGANHO[0][FUZZY_CONT2]=0;
123   FUZZY_uGANHO[1][FUZZY_CONT2]=0;
124   FUZZY_uGANHO[2][FUZZY_CONT2]=0;
```

```
125    FUZZY_uGANHO[3][FUZZY_CONT2]=0;
126  }
127
128  //** Regras para Ganho MB(Muito Baixo) - (0, 0.100, 0.250) - (6 regras)
129    if(FUZZY_erroV[0]>0 && FUZZY_erroV_ant[0]>0){ //1. SE (Erro Atual É MN) E
(Erro Anterior É MN) ENTÃO (duty-cycle é MB)
130
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
131
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP1(0,100,250,(FUZZY_CO
NT2*(FUZZY_UNI/FUZZY_MAP)));
132
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[0],FUZZY
_erroV_ant[0]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[0],FUZZY_e
rroV_ant[0]);
133       }
134       FUZZY_CONT1++;
135    }
136
137    if(FUZZY_erroV[0]>0 && FUZZY_erroV_ant[1]>0){ //2. SE (Erro Atual É MN) E
(Erro Anterior É PN) ENTÃO (duty-cycle é MB)
138
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
139
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP1(0,100,250,(FUZZY_CO
NT2*(FUZZY_UNI/FUZZY_MAP)));
140
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[0],FUZZY
_erroV_ant[1]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV_ant[0],FUZ
ZY_erroV[1]);
141       }
142       FUZZY_CONT1++;
143    }
144    if(FUZZY_erroV[0]>0 && FUZZY_erroV_ant[2]>0){ //3. SE (Erro Atual É MN) E
(Erro Anterior É ZR) ENTÃO (duty-cycle é MB)
145
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
146
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP1(0,100,250,(FUZZY_CO
NT2*(FUZZY_UNI/FUZZY_MAP)));
147
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[0],FUZZY
_erroV_ant[2]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[0],FUZZY_e
rroV_ant[2]);
148       }
149       FUZZY_CONT1++;
```

```
150    }
151
152    if(FUZZY_erroV[1]>0 && FUZZY_erroV_ant[0]>0){ //6. SE (Erro Atual É PN) E
(Erro Anterior É MN) ENTÃO (duty-cycle é MB)
153
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
154
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP1(0,100,250,(FUZZY_CO
NT2*(FUZZY_UNI/FUZZY_MAP)));
155
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[1],FUZZY
_erroV_ant[0]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[1],FUZZY_e
rroV_ant[0]);
156       }
157       FUZZY_CONT1++;
158    }
159
160    if(FUZZY_erroV[1]>0 && FUZZY_erroV_ant[1]>0){ //7. SE (Erro Atual É PN) E
(Erro Anterior É PN) ENTÃO (duty-cycle é MB)
161
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
162
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP1(0,100,250,(FUZZY_CO
NT2*(FUZZY_UNI/FUZZY_MAP)));
163
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[1],FUZZY
_erroV_ant[1]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[1],FUZZY_e
rroV[1]);
164       }
165       FUZZY_CONT1++;
166    }
167
168    if(FUZZY_erroV[2]>0 && FUZZY_erroV_ant[0]>0){ //11. SE (Erro Atual É ZR) E
(Erro Anterior É MN) ENTÃO (duty-cycle É MB)
169
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
170
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP1(0,100,250,(FUZZY_CO
NT2*(FUZZY_UNI/FUZZY_MAP)));
171
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[2],FUZZY
_erroV_ant[0]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[2],FUZZY_e
rroV_ant[0]);
172       }
173       FUZZY_CONT1++;
174    }
```

```
175
176  //** Regras para Ganho B(Baixo) - (0.150, 0.300, 0.450) - (4 regras)
177    if(FUZZY_erroV[0]>0 && FUZZY_erroV_ant[3]>0){ // 4. SE (Erro Atual É MN) E
(Erro Anterior É PP) ENTÃO (duty-cycle é B)
178
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
179
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(150,300,450,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
180
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[0],FUZZY
_erroV_ant[3]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[0],FUZZY_e
rroV_ant[3]);
181       }
182       FUZZY_CONT1++;
183    }
184
185    if(FUZZY_erroV[1]>0 && FUZZY_erroV_ant[2]>0){ //8.  SE (Erro Atual É PN) E
(Erro Anterior É ZR) ENTÃO (duty-cycle É B);
186
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
187
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(150,300,450,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
188
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[1],FUZZY
_erroV_ant[2]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[1],FUZZY_e
rroV_ant[2]);
189       }
190       FUZZY_CONT1++;
191    }
192
193    if(FUZZY_erroV[2]>0 && FUZZY_erroV_ant[1]>0){ // 12. SE (Erro Atual É ZR) E
(Erro Anterior É PN) ENTÃO (duty-cycle É B)
194
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
195
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(150,300,450,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
196
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[2],FUZZY
_erroV_ant[1]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[2],FUZZY_e
rroV_ant[1]);
197       }
198       FUZZY_CONT1++;
199    }
```

```
200
201    if(FUZZY_erroV[3]>0 && FUZZY_erroV_ant[0]>0){ // 15. SE (Erro Atual É PP) E
(Erro Anterior É MN) ENTÃO (duty-cycle É B)
202
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
203
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(150,300,450,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
204
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[3],FUZZY
_erroV_ant[0]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[3],FUZZY_e
rroV_ant[0]);
205        }
206        FUZZY_CONT1++;
207    }
208
209  //** Regras para Ganho M(Medio) - (0.350, 0.500, 0.650) - (4 regras)
210    if(FUZZY_erroV[0]>0 && FUZZY_erroV_ant[4]>0){ // 5. SE (Erro Atual É MN) E
(Erro Anterior É MP) ENTÃO (duty-cycle é M);
211
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
212
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(350,500,650,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
213
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[0],FUZZY
_erroV_ant[4]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[0],FUZZY_e
rroV_ant[4]);
214        }
215        FUZZY_CONT1++;
216    }
217
218    if(FUZZY_erroV[1]>0 && FUZZY_erroV_ant[3]>0){ // 9. SE (Erro Atual É PN) E
(Erro Anterior É PP) ENTÃO (duty-cycle É M);
219
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
220
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(350,500,650,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
221
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[1],FUZZY
_erroV_ant[3]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[1],FUZZY_e
rroV_ant[3]);
222        }
223        FUZZY_CONT1++;
224    }
```

```
225
226    if(FUZZY_erroV[3]>0 && FUZZY_erroV_ant[1]>0){ // 16. SE (Erro Atual É PP) E
(Erro Anterior É PN) ENTÃO (duty-cycle É M);
227
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
228
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(350,500,650,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
229
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[3],FUZZY
_erroV_ant[1]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[3],FUZZY_e
rroV_ant[1]);
230        }
231        FUZZY_CONT1++;
232    }
233
234    if(FUZZY_erroV[4]>0 && FUZZY_erroV_ant[0]>0){ // 20. SE (Erro Atual É MP) E
(Erro Anterior É MN) ENTÃO (duty-cycle É M)
235
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
236
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(350,500,650,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
237
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[4],FUZZY
_erroV_ant[0]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[4],FUZZY_e
rroV_ant[0]);
238        }
239        FUZZY_CONT1++;
240    }
241
242  //** Regras para Ganho A(Alto) - (0.550, 0.700, 0.850) - (4 regras)
243    if(FUZZY_erroV[1]>0 && FUZZY_erroV_ant[4]>0){ // 10. SE (Erro Atual É PN) E
(Erro Anterior É MP) ENTÃO (duty-cycle É A)
244
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
245
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(550,700,850,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
246
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[1],FUZZY
_erroV_ant[4]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[1],FUZZY_e
rroV_ant[4]);
247        }
248        FUZZY_CONT1++;
249    }
```

```
250
251    if(FUZZY_erroV[2]>0 && FUZZY_erroV_ant[3]>0){ // 13. SE (Erro Atual É ZR) E
(Erro Anterior É PP) ENTÃO (duty-cycle É A)
252
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
253
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(550,700,850,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
254
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[2],FUZZY
_erroV_ant[3]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[2],FUZZY_e
rroV_ant[3]);
255        }
256        FUZZY_CONT1++;
257    }
258
259    if(FUZZY_erroV[3]>0 && FUZZY_erroV_ant[2]>0){ // 17. SE (Erro Atual É PP) E
(Erro Anterior É ZR) ENTÃO (duty-cycle É A)
260
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
261
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(550,700,850,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
262
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[3],FUZZY
_erroV_ant[2]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[3],FUZZY_e
rroV_ant[2]);
263        }
264        FUZZY_CONT1++;
265    }
266
267    if(FUZZY_erroV[4]>0 && FUZZY_erroV_ant[1]>0){ // 21. SE (Erro Atual É MP) E
(Erro Anterior É PN) ENTÃO (duty-cycle É A);
268
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
269
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRIAN(550,700,850,(FUZZY_C
ONT2*(FUZZY_UNI/FUZZY_MAP)));
270
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[4],FUZZY
_erroV_ant[1]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[4],FUZZY_e
rroV_ant[1]);
271        }
272        FUZZY_CONT1++;
273    }
274
```

```
275  //** Regras para Ganho MA(Muito Alto) - (0.750, 0.900, 1.000) - (6 regras)
276    if(FUZZY_erroV[2]>0 && FUZZY_erroV_ant[4]>0){ // 14. SE (Erro Atual É ZR) E
(Erro Anterior É MP) ENTÃO (duty-cycle É MA)
277
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
278
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP2(750,900,1000,(FUZZY_
CONT2*(FUZZY_UNI/FUZZY_MAP)));
279
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[2],FUZZY
_erroV_ant[4]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[2],FUZZY_e
rroV_ant[4]);
280       }
281       FUZZY_CONT1++;
282    }
283
284    if(FUZZY_erroV[3]>0 && FUZZY_erroV_ant[4]>0){ // 19. SE (Erro Atual É PP) E
(Erro Anterior É MP) ENTÃO (duty-cycle É MA)
285
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
286
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP2(750,900,1000,(FUZZY_
CONT2*(FUZZY_UNI/FUZZY_MAP)));
287
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[3],FUZZY
_erroV_ant[4]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[3],FUZZY_e
rroV_ant[4]);
288       }
289       FUZZY_CONT1++;
290    }
291
292    if(FUZZY_erroV[4]>0 && FUZZY_erroV_ant[4]>0){ // 24. SE (Erro Atual É MP) E
(Erro Anterior É MP) ENTÃO (duty-cycle É MA)
293
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
294
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP2(750,900,1000,(FUZZY_
CONT2*(FUZZY_UNI/FUZZY_MAP)));
295
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[4],FUZZY
_erroV_ant[4]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[4],FUZZY_e
rroV_ant[4]);
296       }
297       FUZZY_CONT1++;
298    }
299
```

```
300    if(FUZZY_erroV[3]>0 && FUZZY_erroV_ant[3]>0){ // 18. SE (Erro Atual É PP) E (Erro Anterior É PP) ENTÃO (duty-cycle É MA)
301
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
302
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP2(750,900,1000,(FUZZY_CONT2*(FUZZY_UNI/FUZZY_MAP)));
303
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[3],FUZZY_erroV_ant[3]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[3],FUZZY_erroV_ant[3]);
304       }
305       FUZZY_CONT1++;
306    }
307
308    if(FUZZY_erroV[4]>0 && FUZZY_erroV_ant[3]>0){ // 23.       SE (Erro Atual É MP) E (Erro Anterior É PP) ENTÃO (duty-cycle É MA)
309
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
310
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP2(750,900,1000,(FUZZY_CONT2*(FUZZY_UNI/FUZZY_MAP)));
311
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[4],FUZZY_erroV_ant[3]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[4],FUZZY_erroV_ant[3]);
312       }
313       FUZZY_CONT1++;
314    }
315
316    if(FUZZY_erroV[4]>0 && FUZZY_erroV_ant[2]>0){ // 22.       SE (Erro Atual É MP) E (Erro Anterior É ZR) ENTÃO (duty-cycle É MA)
317
for(FUZZY_CONT2=FUZZY_MEP;FUZZY_CONT2<=FUZZY_MAP;FUZZY_CONT2++){
318
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_TRAP2(750,900,1000,(FUZZY_CONT2*(FUZZY_UNI/FUZZY_MAP)));
319
if(FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]>FUZZY_MIN(FUZZY_erroV[4],FUZZY_erroV_ant[2]))
FUZZY_uGANHO[FUZZY_CONT1][FUZZY_CONT2]=FUZZY_MIN(FUZZY_erroV[4],FUZZY_erroV_ant[2]);
320       }
321       FUZZY_CONT1++;
322    }
323   //*** Agregação das regras ativadas e cáculo do CDA
324   FUZZY_CDANUM=0;
```

```
325  FUZZY_CDADEN=0;
326  for(FUZZY_CONT1=FUZZY_MEP;FUZZY_CONT1<=FUZZY_MAP;FUZZY_CONT1++){
327
FUZZY_uA[FUZZY_CONT1]=FUZZY_MAX(FUZZY_MAX(FUZZY_MAX(FUZZY_uGANHO[0][
FUZZY_CONT1],FUZZY_uGANHO[1][FUZZY_CONT1]),FUZZY_uGANHO[2][FUZZY_CONT1
]),FUZZY_uGANHO[3][FUZZY_CONT1]);
328     FUZZY_CDANUM=FUZZY_CDANUM+(FUZZY_uA[FUZZY_CONT1]*FUZZY_CONT1);
329   FUZZY_CDADEN=FUZZY_CDADEN+FUZZY_uA[FUZZY_CONT1];
330  }
331  FUZZY_CDA=((FUZZY_CDANUM/FUZZY_CDADEN)*(FUZZY_UNI/FUZZY_MAP));
332  return (FUZZY_CDA);
333 }
334
335 //*********************************************************************
336 //************************** Função Fuzzy Min **************************
337 //*********************************************************************
338 unsigned int FUZZY_MIN(unsigned int a, unsigned int b){
339
340  if(a<=b) return(a);
341  else return(b);
342 }
343
344 //*********************************************************************
345 //************************** Função Fuzzy Max **************************
346 //*********************************************************************
347 unsigned int FUZZY_MAX(unsigned int a, unsigned int b){
348
349  if(a>=b) return(a);
350  else return(b);
351 }
352
353 //*********************************************************************
354 //************************ Função Fuzzy Triangular ***********************
355 //*********************************************************************
356 unsigned int FUZZY_TRIAN(long a, long m, long b, long x){
357
358  //* Variável local
359  unsigned int u;
360
361  if (x<=a) u=0;
362  else if (x>a && x<m) u=(((x-a)*FUZZY_PER)/(m-a));
363  else if (x==m) u=FUZZY_PER;
364  else if (x>m && x<b) u=(((b-x)*FUZZY_PER)/(b-m));
365  if (x>=b) u=0;
366  return(u);
367 }
368
369 //*********************************************************************
370 //************************ Função Fuzzy Trapezoidal **********************
```

```
371 //*********************************************************************
372 unsigned int FUZZY_TRAP1(long a, long m, long b, long x){
373
374  //* Variável local
375  unsigned int u;
376
377  if(x>=a && x<=m) u=FUZZY_PER;
378  else if(x>m && x<b) u=(((b-x)*FUZZY_PER)/(b-m));
379  else if(x>=b) u=0;
380
381  return(u);
382 }
383
384 unsigned int FUZZY_TRAP2(long a, long m, long b, long x){
385
386  //* Variável local
387  unsigned int u;
388
389  if (x<=a) u=0;
390  else if(x>a && x<m) u=(((x-a)*FUZZY_PER)/(m-a));
391  else if(x>=m && x<=b) u=FUZZY_PER;
392
393  return(u);
394 }
395 //*********************************************************************
396 //****************** Funções para configurar o PWM ********************
397 //*********************************************************************
398
399 void InicializaPWM(void){
400
401  // Configuração TIMER2
402  T2CON = 0b01111111; //Postscale -> 1:16 e Prescaler ->16
403  // Bit 7 = 0 -> Não implementado
404  // Bit 6-3 = 1111 -> Postscale 1:16
405  // Bit 2 = 1 -> Timer2 ligado
406  // Bit 1-0 = 1x -> Prescaler 16.
407  CCP2CON = 0b00001111; //configura CCP2 como um PWM
408  // Bit 7-6 = 0 -> Não implementado
409  // Bit 5-4 = 00 -> DC2B1:DC2B0: PWM Duty Cycle Bit 1 and Bit 0 for CCP2 Module
410  // Bit 3-0 = 11xx -> PWM mode
411 }
412
413 void SetaFreqPWM(void){
414
415 // Tpwm = 4 * Tosc * T2CKPS[1:0] *(PR2-1)
416 // Onde:
417 // Tpwm : período do PWM
418 // Tosc: período do oscilador
419 // T2CKPS[1:0]: fator de pré-escala do timer 2
```

```
420 // PR2: valor de carga de comparação do timer 2 (8 bits - 0 a 255)
421
422 // Frequência do PWM -> 3KHz (333.33us)
423 // Prescaler do TMR2 = 16
424 // Tosc = 1/Fosc = 1/48MHz = 20.83ns
425 // PR2 = [Tpwm/(4*Tosc*T2CKPS)]-1
426 // PR2 = [333.33us / (4 * 20.83ns * 16)] - 1 = 249.037 ~ 249
427 PR2 = 249;
428 }
429
430  void PWMDutyCycle(void){
431  unsigned int DC;
432  DC = GANHO_PWM;
433  // Tduty = Tosc*T2CKPS[1:0]*DC[9:0]
434  // Onde:
435  // Tduty : período do duty-cycle
436  // Tosc: período do oscilador
437  // T2CKPS[1:0]: fator de pré-escala do timer 2
438  // DC[9:0]: valor de carga para determinar o duty-cycle (10 bits - 0 a 1023)
439
440  // Tduty = 333.33us para 100% duty cycle
441  // Tosc = 20.83ns
442  // T2CKPS = 16
443  // DC[9:0] = Tduty/Tosc*T2CKPS[1:0]
444  // DC[9:0] = 333.33us /(20.83ns*16) = 1000.15
445
446  // A equação acima é para o cáculo do Duty Cycle utilizando o tempo em
447  // segundos. DC[9:0] = 1000 é referente a aproximadamente 100% de duty cycle,
assim DC[9:0] = 500 é 50% de duty cycle e assim por diante.
448  // Os registradores são configurados a partir do valor de DC[9:0] da seguinte
maneira:
449  //CCP2CONbits.DC2B0 = DC[0]; -> CCP2CONbits.DC2B0 = DC[9:0]
450  //CCP2CONbits.DC2B1 = DC[1]; -> CCP2CONbits.DC2B1 = DC[9:0] >> 1
451  //CCPR2L = DC[9:2]; ->CCPR2L = DC[9:0] >>2
452  CCP2CONbits.DC2B0 = DC;
453  CCP2CONbits.DC2B1 = DC >> 1;
454  CCPR2L = DC >> 2;
455 }
456
457 //*********************************************************************
458 //********* Função para configurar o Conversor A/D ********************
459 //*********************************************************************
460
461  void InicializaConvAD(void){
462  ADCON0 = 0b00000001; // Seleciona AN0 e liga módulo A/D
463  ADCON1 = 0b00001110; // Habilita AN0, Vref+=Vcc e Vref-=Vss
464  ADCON2 = 0b10000111; // Resultado alinhado à direita, 0TAD e FRC
465 }
466
```

```
467 //*********************************************************************
468 //******** Função para gerar dados a partir da função fuzzy************
469 //*********************************************************************
470  void GeraDados(void){
471     // variáveis locais
472     int erro_anterior = -10000/K;
473     int erro_atual = -10000/K;
474     int passo = ((2*10000)/K)/(NP_MAT-1);
475  for(int i=0; i<NP_MAT; i++){
476        for(int j=0; j<NP_MAT; j++){
477            MATRIZ_GANHO[i][j] =
FUZZY_FUNCTION(erro_anterior+i*(passo),erro_atual+j*(passo));// abrange os valores
de erros possíveis, de acordo com os pontos a serem armazenados
478        }
479      VETOR_ERRO[i] = erro_anterior+i*(passo); // abrange os valores de erros
possíveis, entre -10000/K a 10000/K, espaçados de acordo com o numero de pontos a
serem armazenados na memória
480   }
481  }
```

ANEXO A – Estudo Comparativo PI x Controlador *fuzzy* em um Conversor CC-CC *buck* sob Efeitos de Variações de Carga

Comparative Study PI x Fuzzy Controller in a DC-DC Buck Converter Under Load Variations Effects

Paulo Santos; Paulo Yamashita; Vinicius Andrade.
Disciplina de Sistemas Inteligentes Aplicados à Engenharia, ministrada pelo Dr. Danilo Hernane Spatti.
Departamento de Engenharia Elétrica.
UTFPR, Universidade Tecnológica Federal do Paraná.

***Abstract*—This paper presents an application of DC-DC buck converter and push-pull inverter. The proposed fuzzy logic controller is applied in continuous circuit voltage control for the purpose of comparison with the classical, proportional-integrative controller (PI). The proposed system is designed for a maximum load of 100W DC-power.**

Keywords— DC-DC buck; fuzzy logic controller; comparation.

INTRODUÇÃO

Visando diversas aplicações, este trabalho propõe um sistema que permite a comparação entre um controlador clássico proporcional-integral (PI) e um sistema inteligente baseado em lógica *fuzzy*, aplicado ao controle de tensão de conversor CC-CC Buck. Esta estrutura eletrônica pode ser usada em sistemas de telecomunicações, casas, sistemas de armazenamento de energia e outras aplicações, que o controle eficiente, exija a máxima transferência de potência.

O controle *fuzzy* possibilita a supervisão inteligente, baseada apenas em regras linguísticas sobre a operação da máquina elétrica, sendo assim uma estratégia que pode facilitar o controle e melhorar o desempenho em relação a sistemas de controle convencionais. A principal vantagem do controlador *fuzzy* é a não utilização de modelos matemáticos, o que torna o projeto deste controlador mais simples.

O objetivo deste trabalho é a comparação entre o controlador PI com o um controlador *fuzzy* aplicado ao controle de tensão de um Conversor CC-CC Buck. A Análise, projeto e experimentação foram realizados para uma carga máxima de 100 W. Os resultados de simulação são apresentados como forma de comparação dos controladores.

I. CONVERSOR BUCK

O conversor CC-CC abaixador de tensão (conversor *Buck*) produz um valor médio de tensão na saída inferior ao valor médio da tensão de entrada, entretanto a corrente média de saída é maior que a corrente média de entrada. Na teoria, esse tipo de conversor é concebido de forma a possibilitar uma variação contínua da tensão média na carga desde zero até o valor da tensão de alimentação. Esta topologia está representada na Figura 1.

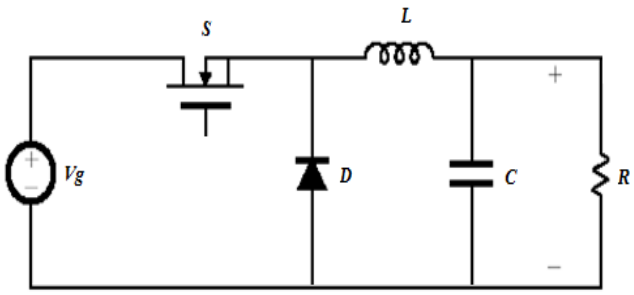


Fig. 1 – Estrutura de potência do Buck convencional.

O seu funcionamento pode ser dividido em duas etapas. A primeira ocorre quando a chave *S* está fechada e a corrente de carga circula pela fonte *Vg* (etapa de transferência de energia da fonte *Vg* para a carga). Na segunda etapa a chave *S* encontra-se aberta e a corrente da carga circula pelo diodo *D* (etapa de roda-livre).

A tensão de saída (*Vo*) é o valor da tensão de entrada (*Vg*) multiplicado pelo valor da razão cíclica (*D*) conforme a equação 1.

$$V_o = D.V_g \tag{1}$$

A razão cíclica (*D*) é a razão entre o tempo em que a chave *S* permanece fechada (T_1) e o tempo total (somatória do tempo (T_1) e o tempo em que a mesma permanece aberta (T_2)) conforme a equação 2.

$$\mathrm{D} = \frac{\mathrm{T_1}}{\mathrm{T_2}} \tag{2}$$

Observa-se pela equação 2 que (*D*) é sempre menor que 1, portanto a tensão de saída (V_o) é sempre menor que a tensão de entrada (V_s).

II. PROJETO DO CIRCUITO

As características de trabalho requeridas para este conversor abaixador estão de acordo com a Tabela 1.

TABELA 1 – Características de projeto do conversor *Buck*.

Tensão de entrada	Vg = 100 V
Tensão de saída	V = 50 V
Potência da carga	Po = 100 W
Frequência de chaveamento	fs = 20 kHz
Ondulação de corrente no indutor	ΔIL = 10 %
Ondulação de tensão na carga	ΔVc = 1 %

De acordo com as especificações apresentadas, calculam-se os parâmetros e valores dos componentes do conversor *Buck*. Tem-se o *duty cycle* de acordo com a equação 3.

$$D = \frac{50}{100} = 0{,}5 \tag{3}$$

Sabendo-se a tensão e potência da carga, calcula-se a corrente que circula pelo indutor e, consequentemente, pela carga, conforme a equação 4.

$$I_L = \frac{100}{50} = 2A \tag{4}$$

Desejando-se obter uma ondulação da corrente no indutor igual a 10% do seu valor médio da corrente de carga (equação 5), calculou-se a indutância por meio da equação 6.

$$\Delta I_L = 0{,}1.I_L = 0{,}1.2 = 0{,}2A \tag{5}$$

$$L = \frac{V_o(1-D)}{\Delta I_L.f_s} = \frac{50(1-0{,}5)}{0{,}2.20.10^3} = 6{,}5\ mH \tag{6}$$

Para obter uma ondulação da tensão igual a 0,1% do valor médio na carga (equação 7), calculou-se o valor do capacitor por meio da equação 8.

$$\Delta V_c = 0{,}001\,.V_0 = 0{,}001\,.50 = 0{,}05V \tag{7}$$

$$C = \frac{\Delta I_L}{8\,.f_s.\Delta V_c} = \frac{0{,}2}{8\,.20.10^3.0{,}05} = 2{,}5\ \mu F \tag{8}$$

III. PROJETO DO CONTROLADOR PI

O conversor Buck, conforme a Figura 2, é controlado por meio de um compensador proporcional-integral (PI), que atua de forma a minimizar o erro gerado entre a tensão de saída e a tensão de referência.

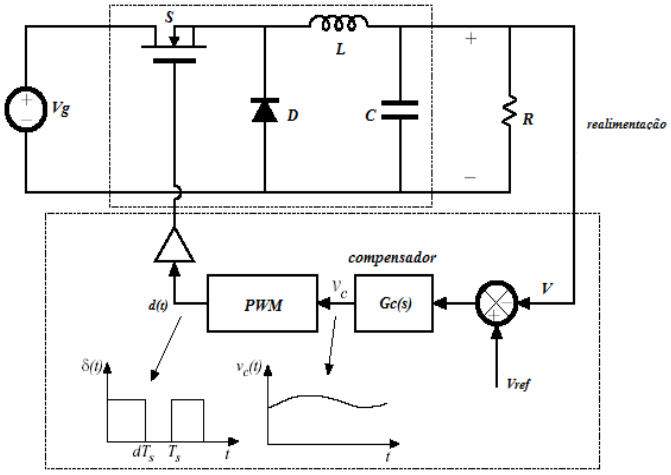


Fig. 2 – Controle PI aplicado ao Buck.

Desta forma, através do diagrama em blocos do controle do conversor apresentado na Figura 3, nota-se que o conversor *Buck* é controlado no modo frequência constante, pois a variável de referência estimada é o PWM que controla conversor.

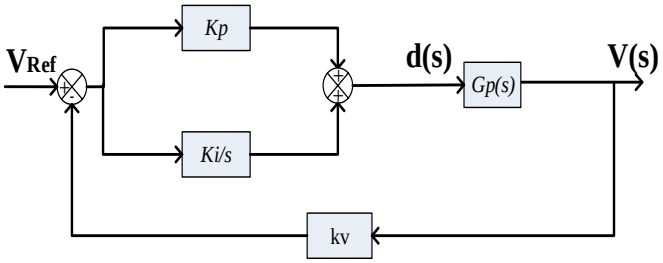


Fig. 3 – Diagrama em blocos do controle.

Os ganhos do controlador (*Kp* e *Ki*) são projetos a fim que o sistema em malha aberta apresente uma margem de fase de 60º na frequência de cruzamento em 0 dB de 1 kHz. Estas especificações permitem que o conversor opere de forma satisfatória em pontos relativamente distantes ao de máxima transferência de potência.

A função de transferência do compensador *Gc(s)* é dada por:

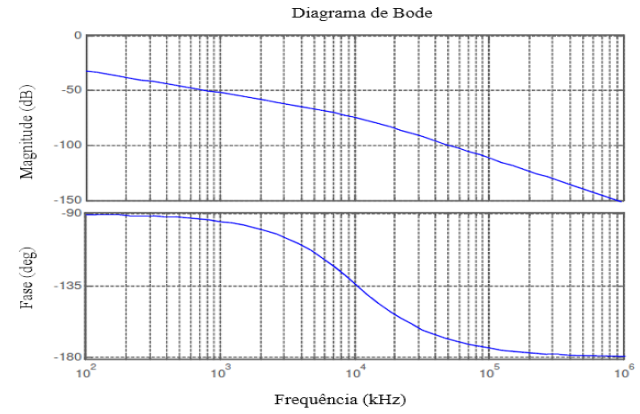
$$G_c(s) = K_p + \frac{K_i}{s} \tag{9}$$

Através do modelo de pequenos sinais, utilizando a técnica da média das equações, a função de transferência do conversor é expressa pela equação 10:

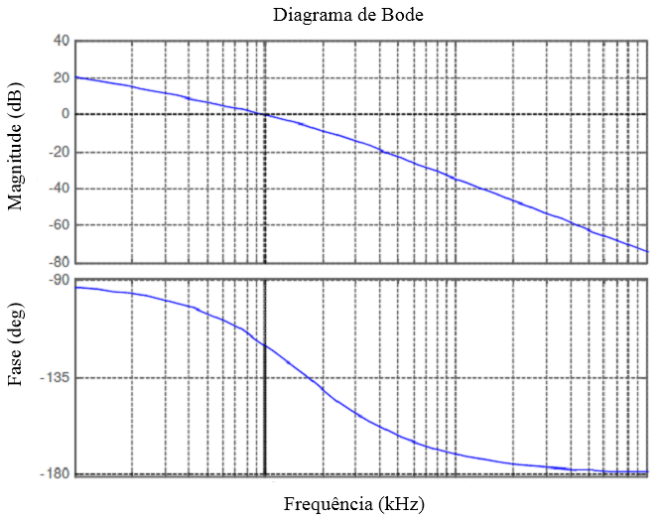
$$G_p = G_{PWM}\,.\,\frac{V}{D}\,.\frac{1}{L.Cs^2+\left(\frac{L}{R}\right)s+1} \tag{10}$$

Através do diagrama de Bode da função de transferência do conversor, ilustrado na Figura 4 (a), os valores obtidos para os ganhos K_i e K_p são iguais a 2,8079.10³ e 0,4675, respectivamente.

Após a implementação do compensador PI utilizando os ganhos encontrados, o diagrama de Bode em malha fechada apresenta uma margem de fase de 60º na frequência de cruzamento em 1 kHz, conforme Figura 4 (b).



(a)



(b)

Fig. 4 - (a) Diagrama de Bode sem o controlador.
(b) Diagrama de Bode com o controlador

IV. PROJETO DO CONTROLADOR FUZZY

A teoria de conjuntos *fuzzy*, apresentada por Zadeh em 1965 (Zadeh, 1965), pode ser utilizada para descrever em

termos matemáticos as informações imprecisas de um determinado processo, através de um conjunto de regras linguísticas. O projeto de um controlador *fuzzy* é inicialmente baseado nas informações do especialista, sendo posteriormente ajustado de forma a obter um ponto satisfatório de operação deste sistema. O controlador *fuzzy* é constituído por estágios de fuzzificação, inferência, defuzzificação e base de dados/regras (Lee, 1990), sendo:

1. Sistema de fuzzificação: Nesta etapa, as variáveis de entrada do sistema são traduzidas em valores *fuzzy* verbais. Esta parte do sistema inclui a definição das funções de pertinência e variáveis linguísticas, bem como a determinação do universo de discurso de cada uma destas variáveis.
2. Inferência: Etapa onde se utiliza regras do tipo se/então, de forma a produzir uma saída *fuzzy* a ser utilizada no cálculo da variável de saída do processo.
3. Sistema de defuzzificação: Nesta etapa, as variáveis *fuzzy*, obtidas na inferência, são transformadas em variáveis reais a serem enviadas ao processo.
4. Base de dados/regras: Etapa de definição das funções de pertinência dos sistemas de fuzzificação e defuzzificação, bem como de definição do conjunto de regras *fuzzy*.

O diagrama representado na Figura 5 descreve a topologia da lógica utilizada na implementação do controlador Fuzzy, substituindo o controlador clássico PI aplicado ao controle de tensão do *buck*.

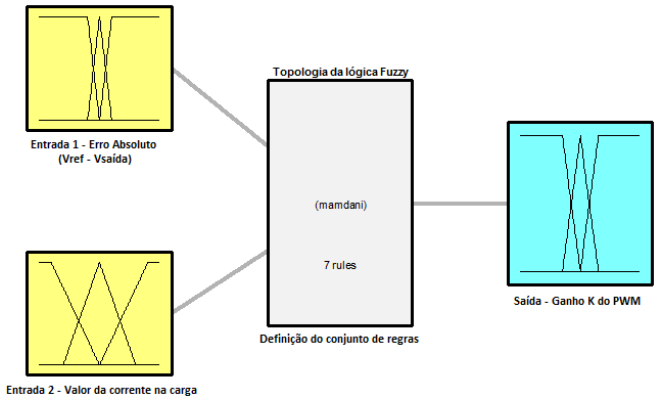


Fig. 5 – Topologia do sistema *Fuzzy*.

Para as entradas foram definidas as funções de pertinências descritas pelas Figuras 6 e 7.

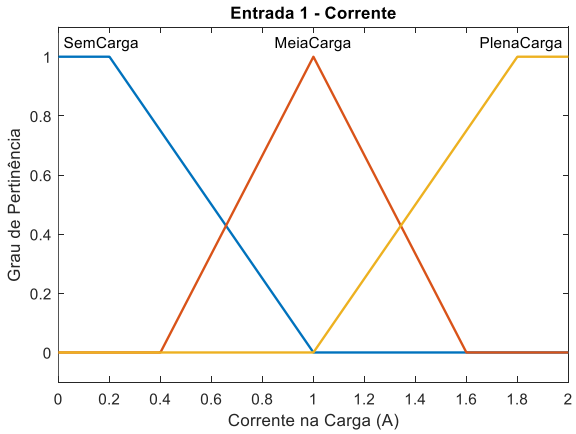


Fig. 6 – Entrada 1.

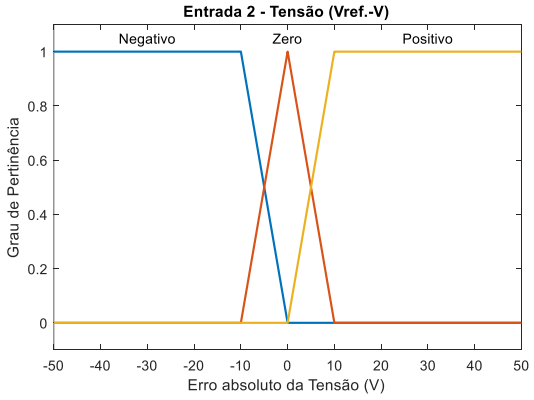


Fig. 7 – Entrada 2.

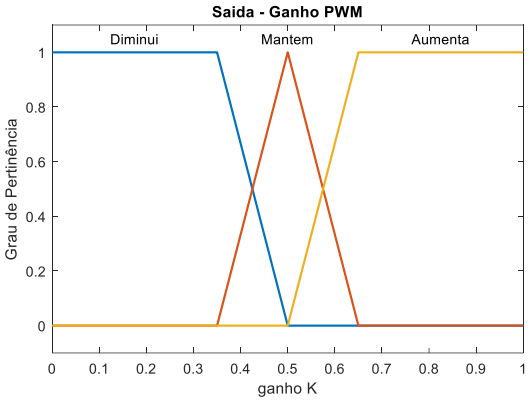


Fig. 8 – Saída para o PWM.

Após a definição das funções de pertinência, determinou-se 7 regras, que estão descritas na Tabela 2.

TABELA 2 – Conjunto de Regras *Fuzzy*.

	Erro Abs. Tensão	Corrente	Ganho
1	negativo	sem carga	diminui
2	negativo	plena carga	diminui
3	zero	sem carga	mantem
4	zero	meia carga	mantem
5	zero	meia carga	mantem
6	positivo	sem carga	aumenta
7	positivo	plena carga	Aumenta

A superfície de controle obtida na Figura 9 é resultado das 7 regras da Tabela 2. A figura gerada a partir do método de inferência do centro de área, além disso, utilizou–se o método de inferência baseado na teoria de Mamdani.

A partira da definição das regras e das funções de pertinência, foi implementado o modelo funcional de ambos os controladores, utilizando a ferramenta computacional MATLAB/Simulink, representado na Figura 10.

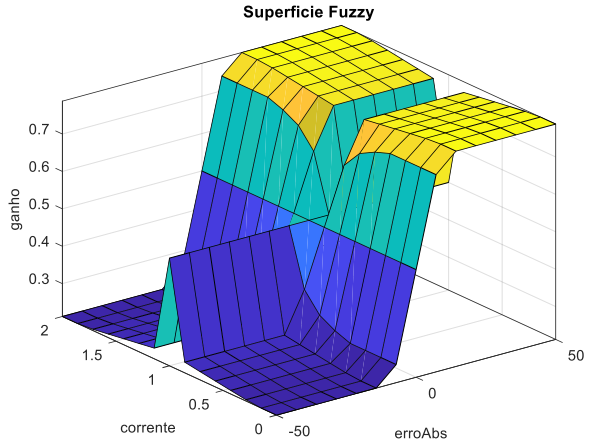


Fig. 9 – Superfície de controle.

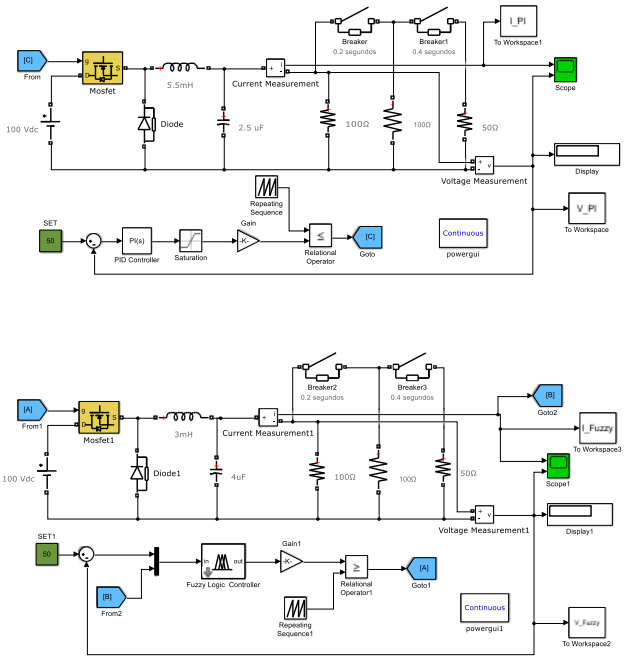
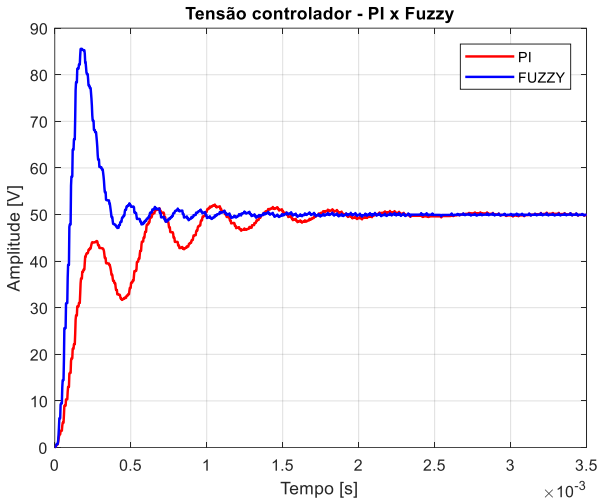


Fig. 10 – Diagrama da Simulação.

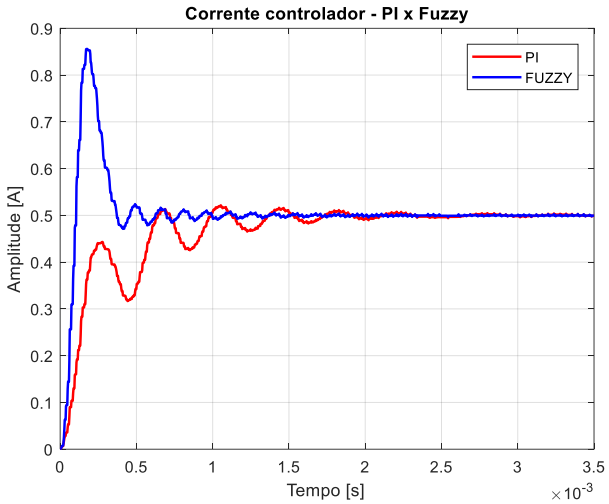
É importante ressaltar que, para a simulação simultânea do controlador PI com o controlador *Fuzzy*, foi necessário que ambos os diagramas estivessem no mesmo arquivo do Simulink, para que os dados exportados para o MATLAB, necessários para a criação dos gráficos, coincidissem a base de tempo, necessária para a comparação.

V. RESULTADOS

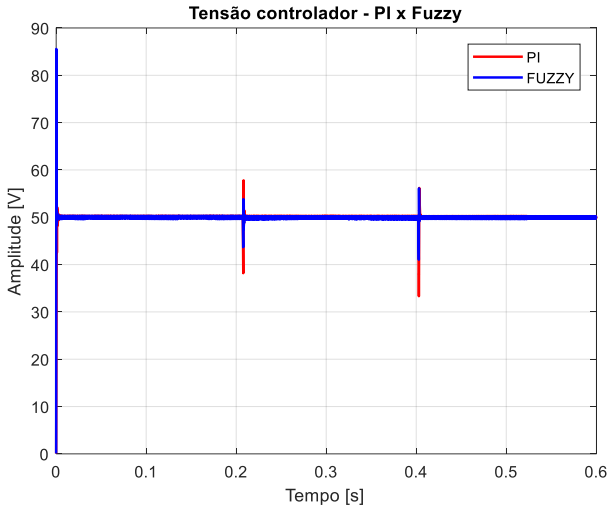
Para a comparação dos controladores, utilizou-se como parâmetro, na simulação, um tempo de 0.6 segundos de duração total, e uma variação de carga a cada 0.2 segundos. A carga inicialmente de 100Ω, demanda uma corrente de 0.5A. Com a demanda de corrente dobrando a cada 0.2 segundos, para que fosse observado o comportamento de cada controlador com a variação da carga. Os gráficos referentes aos resultados estão descritos na Figura 11.



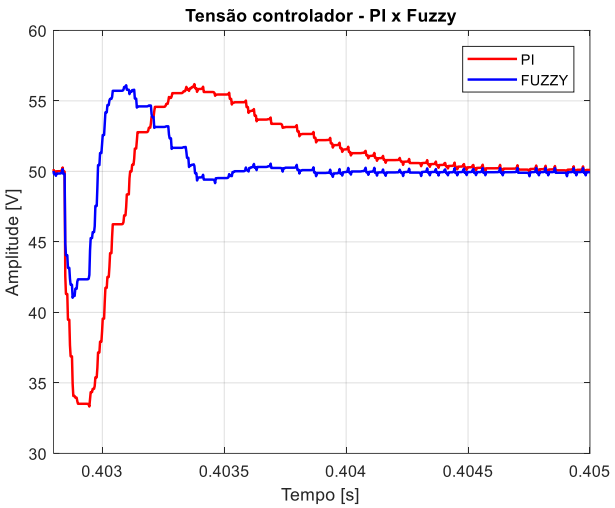
(a)



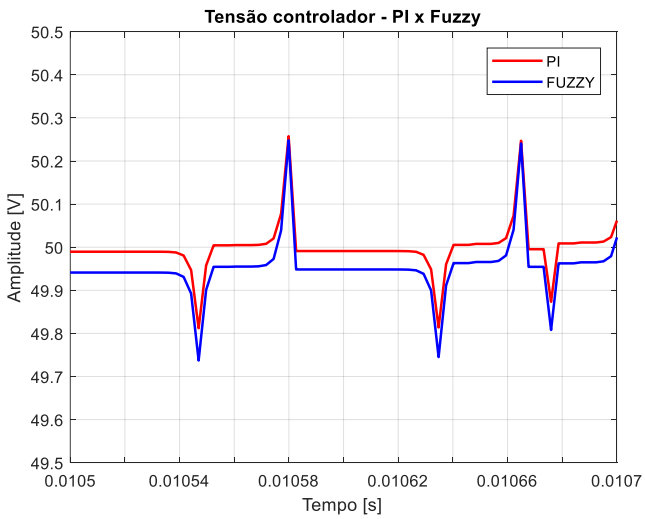
(b)



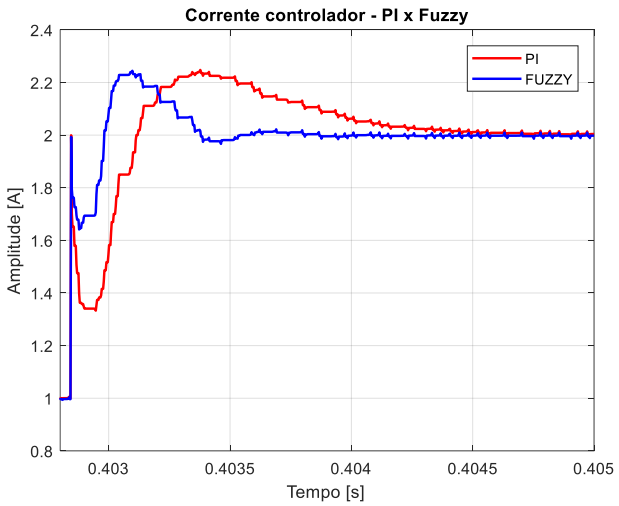
(c)



(d)



(e)



(f)

Fig. 11 – Resultados comparativos. (a) Transitório inicial da tensão. (b) Transitório inicial da corrente. (c) Controle da tensão em 50V. (d) Transitório de tensão de carga de 50 Ω para 25 Ω. (e) Variação da tensão. (f) Transitório de corrente de carga de 50 Ω para 25 Ω.

Tem-se na Figura 12 a corrente do indutor e a tensão sobre a carga constante em 50 V. Observa-se no resultado no intervalo de 0 a 0,2s inicialmente com a metade da carga do sistema. Já no intervalo de 0,2 a 0,4s tem-se um aumento de 50% na carga do sistema. Neste intervalo, observa-se um decaimento da tensão ocorrido em função do aumento da carga.

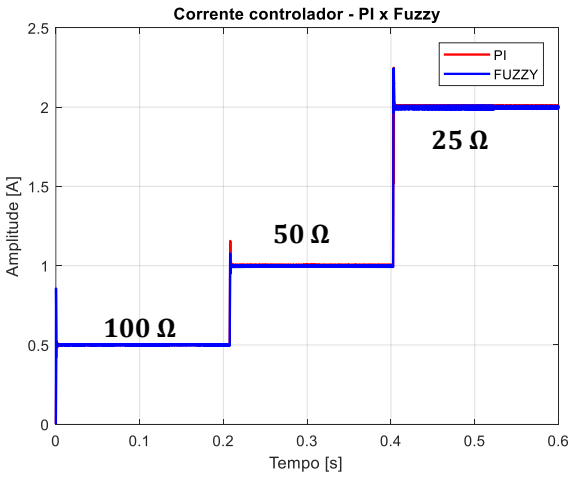


Fig. 12 – Transitórios de corrente.

Na Figura 13, tem-se a ondulação na corrente do indutor *(*ΔIL) e na tensão de saída (ΔVc), ambos os valores obtidos atendem os requisitos de projeto.

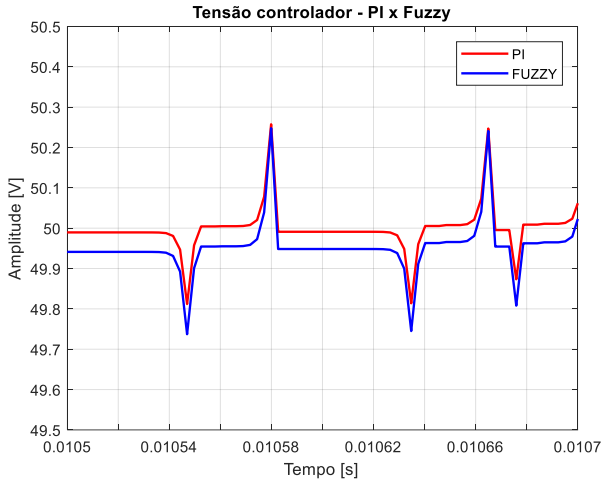


Fig. 13 – Ondulação na corrente do indutor.

CONSIDERAÇÕES FINAIS

Nesse trabalho um controle inteligente foi desenvolvido através da técnica de Lógica *Fuzzy*. Como principal ferramenta, utilizou-se em conjunto o SIMULINK e Fuzzy Logic Toolbox, ambos disponíveis no software MATLAB versão 2012b. Vale destacar que o presente trabalho busca analisar e descobrir vantagens adicionais provenientes da aplicação de sistemas inteligentes em sistemas de difícil modelagem. Ambos os controladores desenvolvidos foram responsáveis por controlar a tensão saída do conversor buck.

O controle desenvolvido é posto como forma de remodelar o controle proporcional clássico e como alternativa eficiente para aplicação em sistemas que exigem controles clássicos tipo Proporcional (P) e/ou Proporcional-Integral (PI), uma vez que tais controles não possuem bom desempenho em sistemas que contenham forte não linearidade, além de terem parâmetros de ganho de difícil sintonia, sendo que muitas das vezes o método de tentativa e erro o mais utilizado para determinar o melhor ajuste do controlador. Ficou claro através da análise da Figura 11 que mostram, respectivamente, as respostas do controle clássico e do controle Fuzzy, que diferentemente do controle clássico, o controle fuzzy teve respostas mais rápidas.

Com isso, pode-se afirmar que a aplicação da lógica fuzzy ao problema em questão pode ser uma solução promissora, adequada para supervisão de sistemas autônomos, uma vez que o uso do conhecimento especialista elimina a necessidade de complexas modelagens matemáticas e permite maior adequação e flexibilidade do sistema de controle.

REFERÊNCIAS

V. B. Malvezzi1, S. A. Silva, L. B. G. Campanhol, B. A. Angélico Lógica Fuzzy Aplicada Nas Malhas De Controle De Corrente E Tensão De Um Filtro Ativo de Potência Paralelo.

F. C. B. Sena, Fuzzy Pitch Controller Applied to PMSG Wind Turbine. In: XXI Congresso da Associação Chilena de Controle Automático ACCA 2014, 21. 2014.

D. H. Spatti, Notas de Aula. UTFPR, 2017.

I. S. Shaw; M. G Simões,. Controle e modelagem Fuzzy. São Paulo: E. Blücher, 1999. 165 p.

I. Barbi, Eletrônica de potência: conversores CC-CC básicos não isolados. 3.ed. Florianópolis Ed. do Autor, 2008. 380 p.

R. H. Eckstein, Sistema para conexão de pequenos aerogeradores com a rede elétrica: análise, projeto e experimentação.

A. Stabile , A. J. Marques Cardoso and C. Boccaletti "Efficiency analysis of power converters for urban wind turbine applications", Conf. Rec. 2nd Sustain. Energy Technol., pp.1 -6 2010.

Z.Y. Zhao, M. Tomizuka, S. Isaka, "Fuzzy Gain Schedulling of PID Controllers," IEEE Transactions on Systems, Man, and Cybern. vol. 23, no 5, pp.1392-1398. 1993.

J. P. A. Vieira, M. V. A. Nunes, Member, U. H. Bezerra and W. Barra Jr. Novas Estratégias de Controle Fuzzy Aplicadas ao Conversor do DFIG para Melhoria da Estabilidade Transitória em Sistemas Eólicos

ANEXO 1: Códigos utilizados.

```
% Script plotagem dos gráficos
clc; clear all; close all;
                                    % normalizando o vetor de dados obtidos
tempo = linspace(0,0.6,218680); % para o tempo da simulação do Simulink

rules_VI = readfis('H:\Arquivos\Vinicius\UTFPR\Sistemas
Inteligentes\Projeto\Arquivos_Principais\rules_VeI.fis');
load('H:\Arquivos\Vinicius\UTFPR\Sistemas
Inteligentes\Projeto\Arquivos_Principais\V_Fuzzy.mat');
load('H:\Arquivos\Vinicius\UTFPR\Sistemas
Inteligentes\Projeto\Arquivos_Principais\I_Fuzzy.mat');
load('H:\Arquivos\Vinicius\UTFPR\Sistemas
Inteligentes\Projeto\Arquivos_Principais\V_PI.mat');
load('H:\Arquivos\Vinicius\UTFPR\Sistemas
Inteligentes\Projeto\Arquivos_Principais\I_PI.mat');

%Corrente
figure
plot(tempo,I_PI,'r','LineWidth',1.5)
title('Corrente controlador - PI x Fuzzy')
xlabel('Tempo [s]')
ylabel('Amplitude [A]')
hold on
grid on
plot(tempo,I_Fuzzy,'b','LineWidth',1.5)
legend('PI','FUZZY')

%Tensão
figure
plot(tempo,V_PI,'r','LineWidth',1.5)
title('Tensão controlador - PI x Fuzzy')
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
hold on
grid on
plot(tempo,V_Fuzzy,'b','LineWidth',1.5)
legend('PI','FUZZY')

%Corrente com zoom - xlim([0.0105 0.0107]
figure
plot(tempo,I_PI,'r','LineWidth',1.5)
title('Corrente controlador - PI x Fuzzy')
xlabel('Tempo [s]')
ylabel('Amplitude [A]')
hold on
grid on
plot(tempo,I_Fuzzy,'b','LineWidth',1.5)
legend('PI','FUZZY')
xlim([0.0105 0.0107])
ylim([0.49 0.51])

%Tensão com zoom xlim([0.0105 0.0107]
figure
plot(tempo,V_PI,'r','LineWidth',1.5)
title('Tensão controlador - PI x Fuzzy')
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
hold on
grid on
plot(tempo,V_Fuzzy,'b','LineWidth',1.5)
legend('PI','FUZZY','Referencia')
xlim([0.0105 0.0107])
```

```
ylim([49.5 50.5])

%Corrente com zoom xlim([0 0.0035])
figure
plot(tempo,I_PI,'r','LineWidth',1.5)
title('Corrente controlador - PI x Fuzzy')
xlabel('Tempo [s]')
ylabel('Amplitude [A]')
hold on
grid on
plot(tempo,I_Fuzzy,'b','LineWidth',1.5)
legend('PI','FUZZY')
xlim([0 0.0035])
%ylim([0.49 0.51])

%Tensão com zoom xlim([0 0.0035])
figure
plot(tempo,V_PI,'r','LineWidth',1.5)
title('Tensão controlador - PI x Fuzzy')
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
hold on
grid on
plot(tempo,V_Fuzzy,'b','LineWidth',1.5)
legend('PI','FUZZY','Referencia')
xlim([0 0.0035])
%ylim([20 70])

%Corrente com zoom xlim([0 0.0035])
figure
plot(tempo,I_PI,'r','LineWidth',1.5)
title('Corrente controlador - PI x Fuzzy')
xlabel('Tempo [s]')
ylabel('Amplitude [A]')
hold on
grid on
plot(tempo,I_Fuzzy,'b','LineWidth',1.5)
legend('PI','FUZZY')
xlim([0.4028 0.405])
%ylim([0.49 0.51])

%Tensão com zoom xlim([0 0.0035])
figure
plot(tempo,V_PI,'r','LineWidth',1.5)
title('Tensão controlador - PI x Fuzzy')
xlabel('Tempo [s]')
ylabel('Amplitude [V]')
hold on
grid on
plot(tempo,V_Fuzzy,'b','LineWidth',1.5)
legend('PI','FUZZY','Referencia')
xlim([0.4028 0.405])
%ylim([20 70])

% figuras referente ao sistema Fuzzy
figure
plotfis(rules_VI)
title('Controle Fuzzy de tensão e corrente em um Buck')

figure
plotmf(rules_VI,'input',2)
title('Entrada 1 - Corrente')
xlabel('Corrente na Carga (A)')
```

```
ylabel('Grau de Pertinência')

figure
plotmf(rules_VI,'input',1)
title('Entrada 2 - Tensão (Vref.-V)')
xlabel('Erro absoluto da Tensão (V)')
ylabel('Grau de Pertinência')

figure
plotmf(rules_VI,'output',1)
title('Saida - Ganho PWM')
xlabel('ganho K')
ylabel('Grau de Pertinência')

figure
gensurf(rules_VI)
title('Superficie Fuzzy')
box on

showrule(rules_VI)
```