Gerenciamento de energia em processadores com o uso de lógica Fuzzy

Augusto José Silva Firmo

Dissertação de Mestrado apresentada à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Antonio Marcus Nogueira Lima
Orientador
Saulo Oliveira Dornellas Luiz
Orientador

Campina Grande, Paraíba, Brasil

Gerenciamento de energia em processadores com o uso de lógica Fuzzy

Augusto José Silva Firmo

Dissertação de Mestrado apresentada em Agosto de 2018

Antonio Marcus Nogueira Lima
Orientador
Saulo Oliveira Dornellas Luiz
Orientador

Angelo Perkusich
Componente da Banca
Jaidilson Jó da Silva
Componente da Banca

Campina Grande, Paraíba, Brasil, Agosto de 2018

F525g Firmo, Augusto José Silva.
Gerenciamento de energia em processadores com o uso de lógica fuzzy / Augusto José Silva Firmo. – Campina Grande, 2018.
111 f. : il. color.

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2018.
"Orientação: Prof. Dr. Antonio Marcus Nogueira Lima, Prof. Dr. Saulo Oliveira Dornellas Luiz ".
Referências.

1. Processador. 2. CPU. 3. Gerenciamento de energia. 4. Fuzzy. 5. Modelagem. I. Lima, Antonio Marcus Nogueira. II. Luiz, Saulo Oliveira Dornellas. III. Título.

CDU 004.31(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MARIA ANTONIA DE SOUSA CRB 15/398

"GERENCIAMENTO DE ENERGIA EM PROCESSADORES COM O USO DE LÓGICA FUZZY"

AUGUSTO JOSÉ SILVA FIRMO

DISSERTAÇÃO APROVADA EM 05/09/2018

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Orientador(a)

SAULO OLIVEIRA DORNELLAS LUIZ, D.Sc., UFCG
Orientador(a)

ANGELO PERKUSICH, D.Sc., UFCG
Examinador(a)

JAIDILSON JO DA SILVA, D.Sc., UFCG
Examinador(a)

CAMPINA GRANDE - PB

Aos meus pais e amigos

Agradecimentos

Agradeço primeiramente aos meus pais, Genilda e Venilton, por todos os sacrifícios realizados para a minha educação, por todo apoio, dedicação e exemplos dados durante toda minha vida.

Ao meu orientador, o Professor Antonio Marcus Nogueira Lima, muito obrigado por aceitar seguir esse caminho comigo e dedicação ao longo deste trabalho.

Ao Professor Saulo Oliveira Dornellas Luiz, por toda dedicação, tempo, conversas e risadas. Obrigado por ser este grande professor e fã de Star Wars.

Aos amigos de laboratório, Breno, Bruna, Andrei, Marcus, Tony, Pedro, Jonatas, Wislayne, Fabiano, Juan, Aislan, Natália e Karla. Agradeço imensamente todas as risadas, conversas, cafés e saídas. Sem vocês, este caminho se tornaria muito mais penitente.

Aos amigos da vida, anões, elfos e magos. Sem suas amizades, a vida não teria nenhuma graça. E em especial a minha amiga Camilla por conseguir me aturar neste último ano de mestrado.

Resumo

Computadores de propósito geral são usados em diversos ambientes, tais como residências, centrais de entretenimento, escritórios, ou em aplicações nas quais não se requer grande poder de processamento. Nestas aplicações, o gerenciamento de potência em seus processadores permite reduzir de forma efetiva o consumo de energia sem alterar o desempenho do processador na execução das tarefas solicitadas. O principal objetivo deste trabalho foi o desenvolvimento de um software que permite processar as medições da carga de trabalho e efetuar o controle de utilização do processador mediante a variação da frequência de operação da unidade central de processamento. Para a utilização da Lógica Fuzzy foram desenvolvidos modelos de utilização de processadores para serem usados na implementação de outros tipos de controladores e uma plataforma experimental que permite alterar as frequências nos núcleos dos processadores e efetuar a aquisição de dados, tais como nível de utilização do processador, a potência e a energia consumidas. Para cargas de trabalho de média e baixa intensidade, o gerenciador proposto neste trabalho foi capaz de promover reduções da ordem de 20% no consumo de energia. A utilização da Lógica Fuzzy demanda pouco consumo de energia e dessa forma tem pouco impacto (0,93% da energia consumida) no gerenciamento de energia. Concluiu-se que o uso de técnicas de aprendizagem e reconhecimento de padrões como Lógica Fuzzy se mostrou eficaz no desenvolvimento de modelos de utilização dos processadores, como também na concepção e implementação da lei de controle.

Abstract

General-purpose computers are used in a variety of environments, such as residences, entertainment centers, offices, or in applications where high processing power is not required. In these applications, power management in its processors allows you to effectively reduce power consumption without changing processor performance in performing the requested tasks. The main objective of this work was the development of a software that allows to process the measurements of the workload and to control the utilization of the processor by varying the frequency of operation of the Central Processing Unit. For the use of Fuzzy logic, we have developed models for the processors ultilization to be used in the implementation of other types of controllers and an experimental platform that allows to change the frequencies in the cores of the processors and data acquisition, such as the level of processor utilization, the power and energy consumed. For medium and low intensity workloads, the manager proposed in this work was able to promote reductions in the order of 20 % in energy consumption. The use of Fuzzy logic requires little energy consumption and thus has little impact (0.93 % of energy consumed) in energy management. It was concluded that the use of learning techniques and recognition of patterns as Fuzzy logic has proved to be effective in the development of models for the use of processors, as well as in the conception and implementation the law of control.

Sumário

Glossário

BIOS	*Basic Input / Output System*
CI	Circuito Integrado
CMOS	*Complementary Metal Oxide Semiconductor*
CPU	Unidade Central de Processamento
DVFS	Escalonamento Dinâmico de Tensão e Frequência
GDE	Gerenciamento Dinâmico de Energia
GEBT	Gerenciamento de Energia Baseado em Tarefas
GEND	Gerenciamento de Energia no Nível do Dispositivo
NTP	*Network Time Protocol*
OS	Sistema Operacional
PWM	*Pulse width Modulation*
USB	*Universal Serial Bus*
VLSI	*Very Large Scale Integration*
S	Conjunto de frequências do processador
s_0	Frequência máxima do processador
$s(k)$	Frequência do processador no instante k
$y(k)$	Utilização do processador no instante k
$\iota(k)$	Ociosidade do processador no instante k
$u(k)$	Quociente entre s_0 e $s(k)$
$D(k)$	*Duty cicle* no instante k
$e(k)$	Sinal de erro no instante k
$\Delta e(k)$	Sinal de mudança de erro no instante k
μ_{ij}	Função de pertencimento i da entrada j
μ_k	*K-ésimo* momento de uma variável aleatória

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O gerenciamento dinâmico de energia (GDE) é uma das mais populares e bem sucedidas técnicas de redução de consumo de energia em circuitos integrados (CI) comerciais, especialmente microprocessadores.[1]

Esta técnica permite a redução do consumo de energia em sistemas microprocessados em tempo de execução, ou seja, durante a utilização do sistema. Isto é feito com mudanças na frequência/tensão de operação ou mesmo o desligamento de componentes ociosos no sistema.

Em computadores de uso geral, o gerenciamento de energia é desejado por várias razões, porém destacam-se como mais importantes:

1. Redução geral do consumo de energia.
2. Prolongação da duração da bateria para sistemas portáteis.
3. Redução dos custos de refrigeração dos componentes.
4. Redução do ruído sonoro causado por ventoinhas e micro motores.

O gerenciamento de energia dirigido pelo sistema operacional tem duas categorias: *(i)* escalonamento dinâmico de tensão e frequência do processador; *(ii)* o gerenciamento dos estados de energia dos periféricos (disco rígido, monitor, interfaces de rede, etc.). Quanto ao gerenciamento de energia de dispositivos, é possível destacar dois tipos: *(i)* o gerenciamento de energia no nível do dispositivo (GEND), uma abordagem centrada em *hardware*, que é baseada na simples observação das requisições feitas ao dispositivo, as quais são consideradas indistinguíveis, por não serem observadas as suas fontes; *(ii)* gerenciamento de energia baseado em tarefas (GEBT), uma abordagem centrada em *software*, que utiliza informações provenientes do sistema operacional para classificar as requisições para um dispositivo de acordo com as tarefas às quais elas são associadas, isto é, os processos do sistema operacional.[2]

No *Kernel* do Linux, a técnica de escalonamento dinâmico de tensão e frequência nos processares é implementada através de políticas denominadas *Governors*. A maioria dessas políticas funcionam definindo a frequência entre o valor máximo e mínimo permitido, não levando em consideração a utilização atual do processador.

Existem cinco tipos de *Governors* no *Kernel* do Linux, sendo estes:

1. *Performance*: Configura a frequência da CPU para o máximo permitido.
2. *Powersave*: Configura a frequência da CPU para o mínimo permitido.
3. *Userspace*: Permite que o usuário, ou qualquer programa rodando com *user ID* como *root* definir uma frequência específica para a CPU.
4. *Ondemand*: Configura a frequência da CPU entre os níveis máximos e mínimos permitidos de acordo com o uso passado do processador.
5. *Conservative*: Muito parecido com o *Ondemand*, diferindo por optar em permanecer mais tempo em frequências altas.

Dentre todos os *Governors* presentes no *Kernel* do Linux, o *Ondemand* e o *Conservative* são os únicos que levam em consideração o uso do processador, esse aspecto tornam estes *Governors* reativos, pois necessitam analisar informações que ocorreram no passado para poder estimar a frequência que deverá ser aplicada sobre o processador, a fim de atender as demandas computacionais exigidas pelos programas em execução no sistema [3]. O *Userspace Governor* é a única política que permite as aplicações definirem as tensões e frequências que deverão ser utilizadas pelo processador. Sendo assim, não há um canal de comunicação efetivo entre as aplicações e o *Governor*, de modo que o *Governor* possa ter o controle das tensões e frequências utilizadas por cada uma das tarefas em execução no sistema em tempo real.

Além destas políticas citadas, o Linux permite a criação de novas políticas ou modificação destas no gerenciamento da frequência adotada pela CPU. Devido à natureza não previsível das cargas de trabalho em computadores de uso geral, o uso de técnicas de controle inteligente se faz necessário.

Muitas técnicas de gerenciamento de energia são formuladas tomando como base os modelos de consumo de energia dos sistemas que serão aplicados em questão, para o caso de processadores, devido a imprevisibilidade da sua carga de trabalho ao longo do tempo, o desenvolvimento de um modelo preciso se torna bastante complicado.

Boa parte dos modelos para o consumo de energia em processadores como apresentado por Barros [4] são abstratos, não utilizando informações sobre a utilização ou carga

de trabalho impostas sobre o processador, tornando-os inviáveis no desenvolvimento de técnicas de gerenciamento de energia eficazes.

Sendo assim, é importante considerar situações em que as técnicas de gerenciamento de energia não são definidas a partir de um modelo acurado do consumo de energia do sistema. Para o caso de processadores, os gerenciadores de energia *Ondemand* do sistema Linux ou o *Processor Power Manager* do sistema Windows são implementados sem incorporar informações sobre os modelos de consumo de energia dos processadores, sendo baseados principalmente na utilização do processador. Isto ocorre principalmente pelo grande números de processadores existentes no mercado, sendo complicado o incorporamento de informações dos modelos do consumo de energia de todos os processadores disponíveis.

Neste contexto a modelagem de um sistema computacional torna-se custosa, pois equações matemáticas que descrevam os comportamentos gerados pelo processador juntamente com o sistema operacional, escalonadores de tarefas, BIOS (*Basic Input / Output System*) dentre outros, são complicadas de serem deduzidas. Dessa forma outras formas de modelagem são utilizadas em processadores, como modelos probabilísticos, apresentados por Bosnacki [5] e Dargie [6] [7]. Assim o uso de Lógica Fuzzy pode ser uma boa ferramenta na modelagem da utilização de um sistema computacional por ser capaz de abstrair a complexidade matemática no desenvolvimento do modelo de um sistema computacional.

1.1 Objetivo geral

O objetivo principal deste trabalho é desenvolver um controlador para o gerenciamento dinâmico de energia em processadores baseado em Lógica Fuzzy.

1.2 Objetivos específicos

Para a conclusão e validação do objetivo principal deste trabalho, os seguintes objetivos específicos devem ser concluídos:

1. Prover um *software* capaz de realizar a comunicação entre as cargas de trabalho e o controlador de frequências do processador do sistema operacional, para que ambos trabalhem em conjunto na redução do consumo de energia no processador.

2. Desenvolvimento de um controlador de frequências para processadores baseado em Lógica Fuzzy levando em consideração a utilização atual do processador.

3. Desenvolvimento e construção de uma plataforma experimental para a medição e aquisição de dados de potência em sistemas computacionais.

4. Demonstrar experimentalmente que a metodologia proposta é capaz de reduzir o consumo de energia do processador, em relação aos gerenciadores já existentes no sistema Linux.

1.3 Organização do texto

No Capítulo 2 é apresentada a revisão bibliográfica realizada para este trabalho, sendo dividida em duas partes. A primeira para o gerenciamento dinâmico de energia e a segunda para controladores Fuzzy.

No Capítulo 3 é apresentado a base conceitual para o restante do trabalho, fazendo a descrição de consumo de energia em processadores, exemplificando como são suas tarefas e o sistema Linux. Também é apresentado o modelo de tarefas periódicas a ser usado como base para os modelos do processador desenvolvidos. São apresentados os principais conceitos sobre lógica Fuzzy usados e os algoritmos de clusterização adotados.

No Capítulo 4 é apresentada a plataforma experimental desenvolvida, descrevendo a aquisição de dados e equipamentos utilizados.

No Capítulo 5 são apresentados os modelos Fuzzy desenvolvidos, ressaltando a metodologia adotada para a identificação, o planejamento experimental para aquisição de dados, a estrutura dos modelos e validação.

No Capítulo 6 são apresentados os controladores desenvolvidos, ressaltando suas funções de pertinência, base de regras e saídas, como também a validação dos mesmos por simulação.

No Capítulo 7 são apresentados os dados experimentais obtidos para o controlador desenvolvido em comparação com o *Governor Ondemand*.

Por fim, o Capítulo 8 apresenta as conclusões retiradas do trabalho, apresentando suas principais contribuições e sugestões para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

A revisão bibliográfica desde trabalho pode ser dividida em três partes. A primeira delas trata do gerenciamento dinâmico de energia em processadores de maneira geral. A segunda parte trata dos controladores Fuzzy para o gerenciamento dinâmico de energia em processadores de computadores de propósito geral. E a terceira parte trada dos modelos Fuzzy usando técnicas de aprendizado como clusterização ou redes neurais. Cada uma destas partes será melhor detalhada adiante.

2.1 Gerenciamento dinâmico de energia

Benini et al. [8] desenvolveram um sistema de gerenciamento de energia usando modelos estocásticos. Além disso, esses autores mostraram que o problema fundamental de se encontrar uma política ótima para o desempenho do sistema pode ser formulado como um problema de otimização estocástica.

Luiz et al. [9] desenvolveram um sistema de gerenciamento de energia para sistemas alimentados por baterias, modelando o sistema por cadeias de Markov no tempo discreto, em associação ao modelo de bateria. Sendo assim, uma formulação matemática rigorosa do problema utilizada, resultando em uma solução que uniu tanto desempenho quanto qualidade melhores que técnicas usadas anteriormente.

Sridharan, Gupta e Mahapatra [10] apresentaram um algoritmo para a redução de energia consumida no sistema, porém preservando ao mesmo tempo a probabilidade do sistema funcionar sem ocorrência de falhas, ou seja, mantendo sua confiabilidade, já que algoritmos deste tipo podem levar a uma redução na confiabilidade por conta do tamanho cada vez menor dos dispositivos. A ideia geral apresentada no trabalho é um algoritmo que use um *controlador proporcional com realimentação* que calcule a relação total de faltas com base na demanda de tarefas. Porém seu desenvolvimento é dependente de um modelo do sistema, não sendo desenvolvido assim para computadores de propósito geral.

Wang et al. [11] apresentaram o controle da utilização de processadores em tempo real, onde a malha principal controla a frequência de cada núcleo do processador enquanto a malha de controle secundário adota uma adaptação de taxas para o controle de todos os processadores que processam determinada tarefa. O controlador que desenvolveram é baseado em um modelo linearizado do processador que foi usado. Sendo assim, é necessário o desenvolvimento de um novo controlador para diferentes tipos de processadores.

Chen et al. [12] desenvolveram um método para o escalonamento dinâmico de tensão e frequência para processadores multinúcleo onde os recursos compartilhados por todos os núcleos formam um único domínio de tensão e frequência. O controlador adotado é do tipo PID devido à sua simplicidade. Estes autores não consideraram sistemas em tempo real no desenvolvimento do seu controlador.

Anghel et al. [13] desenvolveram duas técnicas para o gerenciamento de energia em processadores, pelo escalonamento dinâmico da frequência dos mesmos. Um primeiro algoritmo baseado no sistema imunológico humano e um segundo baseado em Lógica Fuzzy. O primeiro usa o antígeno humano (partícula capaz de deflagrar a produção de anticorpos) como modelo para representar o consumo de energia e o estado de desempenho do processador. E a segunda técnica usa Lógica Fuzzy para adaptar os estados de desempenho do processador para as cargas de trabalho subsequentes.

Enomoto e Kobayashi [14] desenvolveram um algoritmo para o gerenciamento dinâmico de energia usando escalonamento dinâmico de tensão e frequência onde é usado estimação de dados para deduzir a tensão e frequência ótimas para o ponto de operação solicitados pela carga de trabalho atual.

Barros [4] apresentou uma revisão bibliográfica dos principais modelos desenvolvidos para a potência de processadores desde 1992 até 2013. É possível notar a mudança nestes modelos levando em consideração a evolução da tecnologia em circuitos integrados. Porém na maioria destes modelos, considera-se o princípio do consumo de energia devido ao chaveamento dos processadores e o consumo por corrente de fuga presentes nos transistores quando não há transições de nível lógico.

2.2 Controladores Fuzzy

He et al. [15] desenvolveram um sistema de escalonamento de tarefas e um controlador PID baseados em Lógica Fuzzy para sistemas em tempo real. O escalonador de tarefas faz uso da Lógica Fuzzy para decidir a próxima tarefa que será executada de acordo com o tempo restante para seu *deadline*. A utilização da CPU é controlado pelo controlador PID Fuzzy no lugar de um controlador PID convencional.

Rattanatamrong e Fortes [16] propuseram um recurso de escalonamento usando Ló-

gica Fuzzy, no qual um controlador Fuzzy, um adaptador de utilização de tarefas e um escalonador de tarefas em tempo real trabalham em conjunto. Considerando as incertezas que podem estar presentes nos sistemas e nos ambientes de implementação, o controlador Fuzzy determina o total de utilização da CPU. O adaptador de utilização aloca cada uma das tarefas para que não se ultrapasse o total de utilização disponível. E o escalonador de tarefas, escalona cada tarefa para ser executada em múltiplos processadores.

Zerkaoui e Bosche [17] apresentaram um sistema de gerenciamento de energia para sistemas híbridos desenvolvido por meio de Lógica Fuzzy. Esses autores afirmaram que o projeto do controlador por meio de Controle Clássico teria dificuldades devido às incertezas paramétricas e não linearidades existentes no sistema, e que o projeto do controlador por meio de Lógica Fuzzy ofereceria flexibilidade para lidar com essas dificuldades.

Basaran, Suzer e Kang [18] apresentaram um controlador baseado em Lógica Fuzzy para a utilização da CPU, porém enfatizaram a análise de estabilidade para o sistema Fuzzy em malha fechada, provando-a matematicamente. Em uma aplicação em tempo real comparam o controlador Fuzzy com um controlador PI e um controlador baseado no modelo preditivo de utilização para um conjunto de cargas de trabalho periódicas. O controlador Fuzzy conseguiu os melhores resultados em termos de consumo de energia.

Jacomet e Walti [19] apresentaram uma arquitetura de um processador VLSI (*Very Large Scale Integration*) Fuzzy, fabricado com tecnologia CMOS. Devido à arquitetura paralela do bloco de regras Fuzzy, ou seja, mais de uma regra pode ser ativada por um valor de entrada, um desempenho mais alto é alcançado em relação a estes processadores sem o uso de técnicas Fuzzy.

2.3 Modelos Fuzzy

Yang et al. [20] apresentam um novo algoritmo de clusterização Fuzzy para a análise de grandes quantidades de dados. Os dados de maior dimensão devem ser divididos em dados de menores dimensões, e em seguida estes dados com menores dimensões devem ser agrupados de acordo com seu comportamento. O algoritmo desenvolvido combina clusterização Fuzzy com algoritmos genéticos, tornando a convergência dos grupos mais rápida.

O'Neal e Brisk [21] apresentam uma pesquisa sobre os tipos de modelos de processadores mais empregados, sendo estes modelos estatísticos baseados em modelos probabilísticos para determinação de utilização dos mesmos. Modelos de dinâmicas térmicas para modelagem do consumo de potência e temperatura dos processadores. E modelos de plataforma cruzada, onde dados aquisitados do processador são cruzados em dois modelos de regressão, dessa forma aumentando a precisão do modelo.

Gu [22] apresenta uma nova abordagem para a clusterização subtrativa, agrupando-a com outras técnicas de clusterização, dessa forma cada tipo de clusterização é adotado onde possui melhor desempenho. Este novo método aumenta a velocidade de tomada de decisão dos pontos iniciais de cada grupo, como também aumenta a velocidade de convergência destes.

Ahmed et al. [23] apresentam um método de identificação de movimentos humanos baseada em sistemas neuro-fuzzy. Dividindo-se em três partes principais. A primeira é a aquisição de dados com câmeras com sensores de movimento. A segunda fase é de treinamento, na qual a clusterização subtrativa é usada como uma ferramenta de otimização para determinar o melhor número de funções de pertencimento fuzzy. Ajudando na redução do tempo de treinamento. A terceira fase é a de reconhecimento dos movimentos, onde cada gesto é classificado como um dos gestos treinados em tempo real. O trabalho emprega somente o reconhecimento do movimento dos membros superiores, porém dá margem para reconhecimento de movimentos de todo o corpo.

2.4 Conclusões

Neste capítulo foram apresentados alguns trabalhos que usem como base a lógica Fuzzy para o controle de utilização em processadores, como também controladores de outros tipos e outras técnicas para o controle desta utilização, dessa forma, sendo visto diversas maneiras de realizar o controle de utilização e o gerenciamento dinâmico de energia nos processadores. Também foi apresentado usos de lógica Fuzzy unida a técnicas de clusterização para modelagem de sistemas e tipos de modelos de processadores encontrados na literatura. Dessa forma será apresentado nos próximos capítulos conceitos fundamentais sobre lógica Fuzzy, modelagem e controle com a mesma técnica.

Capítulo 3

Fundamentação teórica

Neste capítulo serão apresentados diversos conceitos utilizados durante a elaboração do trabalho. Os conceitos iniciais abordam o consumo de energia em processadores e cargas de trabalho em sistemas operacionais de propósito geral. O restante irá abordar conceitos utilizados nos desenvolvimento dos modelos e controladores do processador.

3.1 Consumo de energia em processadores

Considerando que para a realização de uma determinada tarefa, existe um gasto de energia que possivelmente pode ser reduzido, então o sistema pode tornar-se energeticamente mais eficiente. Portanto, nesse sentido, a meta para os processadores é executar a mesma carga de trabalho com menos energia ou, então, fazer mais com a mesma energia[4].

O consumo de energia em processadores e outros tipos de circuitos integrados se dá pela tecnologia empregada na sua construção, sendo usado tecnologia CMOS (*"complementary metal-oxide-semiconductor"*) para os variados tipos de processadores [24]. As principais fontes de consumo de energia em componentes CMOS são sua tensão de alimentação, chaveamento e corrente de fuga:

1. Tensão de alimentação: o consumo de energia em componentes CMOS é proporcional ao quadrado da sua tensão de alimentação. Reduzindo este nível de tensão ao menor possível que proveja o desempenho requerido, pode-se reduzir o consumo de energia consideravelmente.

2. Chaveamento: a maior parte do consumo de energia em componentes CMOS está no chaveamento dos seus valores de saída. Sendo assim, é possível se reduzir a energia gasta nestes componentes reduzindo a velocidade em que operam para o mínimo necessário para o desempenho requerido e ainda evitando trocas de estados desnecessárias em suas saídas.

3. Corrente de fuga: mesmo quando um componente CMOS não está ativo, uma quantidade de corrente flui do circuito para o substrato. A única maneira de eliminá-la é removendo o suprimento de energia do componente. Porém isso faria com que o tempo para reconectar este componente ao sistema e reiniciar seus estados iniciais aumentasse. Assim o desligamento do componente é realizado eventualmente, por exemplo por meio de *power gating*.

Existem dois tipos de gerenciamento de tensão que podem ser usados em processadores: gerenciamento de energia estático e gerenciamento de energia dinâmico. O primeiro não depende do estado ou atividades realizadas pelo processador. O segundo executa ações de controle baseado nos estados ou atividades do processador para a redução do consumo de energia.

De maneira geral, pode-se dividir a potência dissipada em processadores e outros tipos de circuitos integrados em dois grupos: a Potência Dinâmica (P_{din}) e a Potência Estática (P_{est}), sendo a Potência Total consumida nos processadores a soma destas. A potência dinâmica leva em consideração os dois primeiros itens citados nesta seção, ou seja, leva em consideração a potência dissipada quando o componente está ativo, que será tanto maior, quanto maior for a frequência de operação do processador. A potência estática leva em consideração o terceiro item citado, ou seja, é a potência dissipada por cada transistor quando este não estão operando, devido à sua corrente de fuga.

Barros [4] apresenta as expressões mais comuns encontradas na literatura para representar matematicamente as potências dissipadas pelos processadores. Estas expressões são apresentadas na Tabela 3.1.

Tabela 3.1: Variantes das expressões da potência elétrica nos processadores.

Referência correspondente	Potência total
(CHANDRAKASAN; SHENG; BRODERSEN, 1992)[25]	$P = A(C_l V_{dd} f_{clk}) + I_{sc} V_{dd} + I_{leak} V_{dd}$
(BURD; BRODERSEN, 1995)[26]	$P \approx V_{dd}^2 f_{clk} C_l$
(MUDGE, 2001) [27]	$P = ACV_{dd}^2 f_{clk} + AV_{dd} I_{short} f_{clk} + V_{dd} I_{leak}$
(WEGLARZ; SALUJA; LIPASTI, 2002)[28]	$P = ACV_{dd}^2 f_{clk}$
(KIM et al., 2003) [29]	$P = ACV_{dd}^2 f_{clk} + V_{dd} I_{leak}$
(VENKATACHALAM; FRANZ, 2005) [30]	$P \approx ACV_{dd}^2 f_{clk}$
(CHANG et al., 2010)[31]	$P = AC_l V_{dd}^2 + I_{leak} V_{dd}$
(KORTHIKANTI; AGHA, 2010) [32]	$P = C_L V_{dd}^2 f_{clk} + I_{leak} V_{dd}$
(USMAN; KHAN; KHAN, 2013) [33]	$P = V_{dd}^2 f_{clk} C_l$

onde I_{short} é a corrente de curto-circuito, I_{leak} é a corrente de fuga, I_{sc} é a corrente

de chaveamento, V_{dd} é a tensão de alimentação, C_l é a capacitância de comutação, f_{clk} é a frequência de chaveamento e A é o fator de atividade, uma constante que se refere a quantas transições de estado binárias ocorrem no *chip*. São apresentados na Figura 3.1 os parâmetros citados em um inversor CMOS.

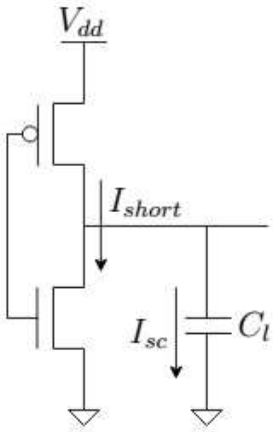

Figura 3.1: Correntes e tensões presentes em um inversor CMOS.
Fonte: Autor.

3.2 Escalonamento dinâmico de tensão e frequência

A idéia principal da técnica de escalonamento dinâmico de tensão e frequência (DVFS) é que a tensão de alimentação e a frequência do *clock* do processador podem ser dinamicamente controladas até o nível mais baixo possível enquanto respeite as constantes de tempo das tarefas em realização. Mesmo que esta técnica possua alguns custos (como dificuldade de implementação), seu uso é válido, pois pode oferecer economia de energia [34].

A expressão em (3.1) define uma aproximação do consumo de energia dinâmica do processador, apresentada por Shin, Lee e Kim [35].

$$E \propto C_l f_{clk} V_{dd}^2 \tag{3.1}$$

na qual C_l é a capacitância de comutação, f_{clk} é a frequência de chaveamento e V_{dd} é a tensão fornecida. Como pode-se notar, o consumo de energia é proporcional então à tensão e frequência usadas no processador. Essas características tornam técnicas de DVFS bastante eficazes no gerenciamento de energia em um processador.

Atualmente, a técnica DVFS só é possível devido à grande maioria dos processadores utilizarem as tecnologias CMOS e MOSFET (*Metal Oxide Semiconductor Field Effect Transistor*), pois elas permitem controlar via *software* as tensões e frequências que são

utilizadas pelo processador, permitindo que sejam feitas trocas constantes de frequência à medida que o sistema necessite de mais ou menos capacidade [36].

Dessa forma o surgimento desta técnica permitiu que fosse possível o desenvolvimento de políticas de gerenciamento dinâmico das tensões e frequências utilizadas no processador para uso em sistemas operacionais.

Em sistemas reais, existem dois tipos principais de técnicas DVFS:

1. DVFS intra-tarefa: as tensões e frequências são definidas pelas tarefas de tempo real, levando em consideração apenas os dados locais da aplicação. Uma das formas de se obter essas tensões e frequências é por meio das análises estáticas do grafo de fluxo de controle.

   Estes grafos são abstrações estatísticas do fluxo de execução de uma aplicação, extraídos a partir do seu código fonte [37].

2. DVFS inter-tarefa: As tensões e frequências são definidas tarefa por tarefas a cada instante de atualização do escalonador do sistema, levando em consideração as informações de todas as tarefas em execução no sistema.

3.3 Modelo de tarefas periódicas

Num sistema de computação de propósito geral, deseja-se que a utilização do processador seja menor que 100%, pois, se surgem novas requisições da carga de trabalho quando a utilização do processador é igual a 100%, essas requisições não serão servidas imediatamente [2].

Inicialmente é necessário analisar a relação entre a variável controlada $y(t)$ (a utilização do processador) e o sinal de controle $u(t)$, que afeta a frequência de operação do processador. O tempo de execução δ de uma tarefa pode ser subdividido em dois subintervalos: [38].

1. *On-chip* (δ^{on}), quando o processador executa instruções de tarefas. Este subintervalo é inversamente proporcional à frequência de operação do processador durante a execução da tarefa.

2. *Off-chip* (δ^{off}), quando o processador aguarda o final da execução de transações com a memória externa. Este subintervalo não depende da frequência de operação atual do processador.

Considerando a frequência máxima do processador s_0 em um conjunto S de frequências do processador, definido em 3.2, com k frequências disponíveis. O subintervalo *on-chip* é

$\delta^{on}(s_0)$ para uma tarefa que é executada durante o período de amostragem h. Então, se a frequência fosse $s < s_0$, o subintervalo *on-chip* seria $\delta^{on}(s) = \frac{s_0}{s}\delta^{on}(s_0)$. Logo, o tempo de execução δ de uma tarefa é $\frac{s_0}{s}\delta^{on}(s_0) + \delta^{off}$ [2]. Na Figura 3.2, é apresentado o efeito da redução da frequência no tempo de execução de uma tarefa.

$$S = \{s_{k-1}, s_{k-2}, s_{k-3}, ..., s_0\} \tag{3.2}$$

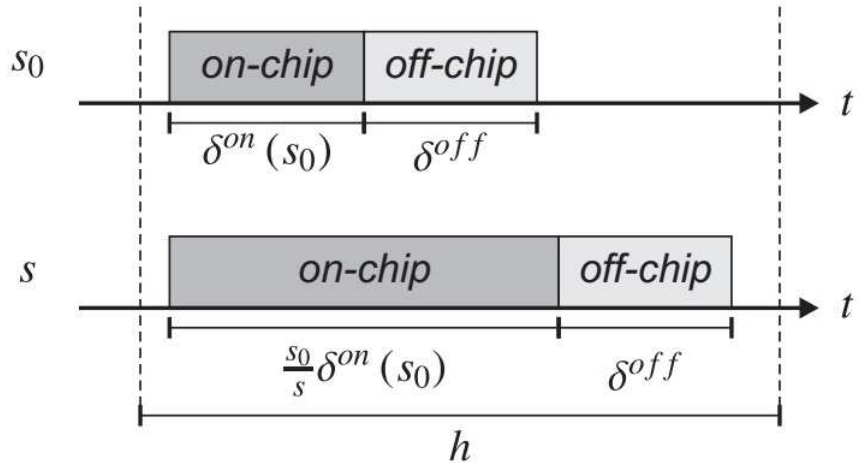

Figura 3.2: Efeito da redução da frequência no tempo de execução de uma tarefa.
Fonte: Adaptado de Luiz [2].

Se N múltiplas tarefas $i = 0, 1, ..., N-1$, são executadas durante o período de amostragem h, cada uma durante um intervalo δ_i, então o período de tempo Δ no qual o processador está ativo pode ser representado por (3.3).

$$\Delta = \sum_{i=0}^{N-1} \delta_i = \frac{s_0}{s} \sum_{i=0}^{N-1} \delta_i^{on}(s_0) + \sum_{i=0}^{N-1} \delta_i^{off} \tag{3.3}$$

Desta forma o utilização do processador y será o período de tempo Δ dividido pelo tempo de amostragem h, como apresentado em (3.4).

$$y(s) = \frac{\Delta}{h} = \frac{s_0}{s} \underbrace{\frac{1}{h} \sum_{i=0}^{N-1} \delta_i^{on}(s_0)}_{u^{on}} + \underbrace{\frac{1}{h} \sum_{i=0}^{N-1} \delta_i^{off}}_{u^{off}} = \frac{s_0}{s} y^{on} + y^{off} \tag{3.4}$$

Em tempo de execução, os valores de u^{on} e u^{off} são desconhecidos, necessitando de técnicas de estimação como mínimos quadrados recursivos, o que pode resultar em uma sobrecarga no gerenciador de energia, isto pode ser evitado pelo uso de outras técnicas, tal como Lógica Fuzzy.

3.4 Lógica Fuzzy

Os princípios de lógica Fuzzy foram desenvolvidos primeiramente por Jan Lukasiewicz (1878-1956), que em 1920 desenvolveu e introduziu conjuntos com grau de pertinência, e os combinou aos conceitos da lógica clássica, desenvolvida por Aristóteles. Tal trabalho ofereceu os fundamentos para que na década de 60, Lofti Asker Zadeh, professor de Ciências da Computação da Universidade da Califórnia, chegasse a ser o primeiro autor de uma publicação sobre lógica Fuzzy [39].

Tradicionalmente, no projeto dos sistemas de controle a modelagem é feita matematicamente, usando-se equações algébricas, diferenciais e outros métodos de cálculo. A maioria das formas intuitivas de se modelar um processo analiticamente usa os princípios fundamentais da física. Tal modelagem, entretanto, requer um excelente domínio do aspecto físico e matemático do processo, o que pode não ser tão simples de se obter. Outro modo é realizar a identificação do sistema usando dados reais da planta para produzir um modelo caixa-preta (ou "black-box"). A estrutura dos modelos tipo caixa-preta normalmente não possuem conexão com a estrutura do sistema real e os parâmetros do modelo não têm significado físico. Outro problema é que muitos algoritmos foram desenvolvidos para sistemas lineares. Entretanto, a maioria dos processos reais é não-linear (como é o caso de motores Diesel) e podem ser aproximados apenas localmente por modelos. Em outros casos, são feitas suposições simplificadas, distantes da realidade do processo. Por outro lado, existem casos em que a informação existente é imprecisa ou de qualidade insuficiente e não pode ser usada de modo efetivo pelas aproximações tradicionais de modelagem. Isto implica que muitos processos ou não podem ser descritos matematicamente de forma adequada, ou suas descrições são muito complexas para ser de qualquer utilidade prática. Isto estimula o uso de técnicas de controle que não requeiram obrigatoriamente modelos matemáticos da planta, como é o caso dos controladores nebulosos ou das redes neurais [40].

Para ilustrar como a lógica Fuzzy funciona, um exemplo simples será apresentado. As regras desta lógica permitem expressar relações entre dados de entrada e saída, como a utilização gerada a partir de uma frequência e uma carga de trabalho aplicada a um processador. Seja a regra[41]:

Se Frequência É baixa E Carga de trabalho É Alta *ENTÃO Utilização É Muito Alta*

Os componentes dessa regra são denotados como:

1. “Frequência, carga de trabalho e utilização” são variáveis linguísticas,
2. “Baixa, alta e muito alta” são termos linguísticos,

3. "E" é um operador lógico,

4. "Frequência É baixa, Carga de trabalho É alta" são confirmações linguísticas,

5. "Frequência É baixa E Carga de trabalho É alta" é a premissa,

6. "Utilização É Muito Alta" é o consequente.

Frequentemente as variáveis linguísticas são chamadas de entradas quando pertencem à premissa e saídas quando pertencem ao consequente. O número total de regras irá depender da resolução dos conjuntos Fuzzy escolhidos e da precisão necessária para a saída. Os componentes das regras Fuzzy são melhores explicados a seguir:

3.4.1 Definição

Um universo de discurso é um conjunto clássico que contém todos os objetos de um domínio de interesse representado por $X = [x_1, \ldots, x_n]$.

Seja $U \subseteq \Re^n$ um universo de discurso. Um subconjunto Fuzzy A em U é caracterizado pela sua função de pertencimento $\mu_A : U \rightarrow [0,1]$ com $\mu_A(X)$ representando o grau de pertencimento de $X \in U$ em A.

3.4.2 Funções de pertencimento

Nas regras Fuzzy, as variáveis linguísticas são expressas na forma de conjuntos Fuzzy. No exemplo anterior, as variáveis linguísticas frequência e carga de trabalho são classificadas pelos termos linguísticos baixo, médio e alto. Esses termos linguísticos são definidos por uma associação a funções de pertencimento. Estas funções de pertencimento definem um grau de pertencimento das variáveis linguísticas aos conjuntos Fuzzy. Os graus de pertencimento são definidos como: $\mu_i(F)$ para a frequência F, na qual i representa baixo, médio e alto; e $\mu_i(W)$, para a carga de trabalho W na qual i representa baixo, médio e alto. Por exemplo um valor de frequência de 0,85 GHz é considerado baixo com $\mu_{baixo}(F) = 0{,}37$, médio com $\mu_{medio}(F) = 0{,}13$ e alto com $\mu_{alto}(F) = 0$, e uma carga de trabalho com intensidade de 0,66 é considerado médio com $\mu_{medio}(W) = 0{,}2$ e alto com $\mu_{alto}(W) = 0{,}3$ como apresenta a Figura 3.3.

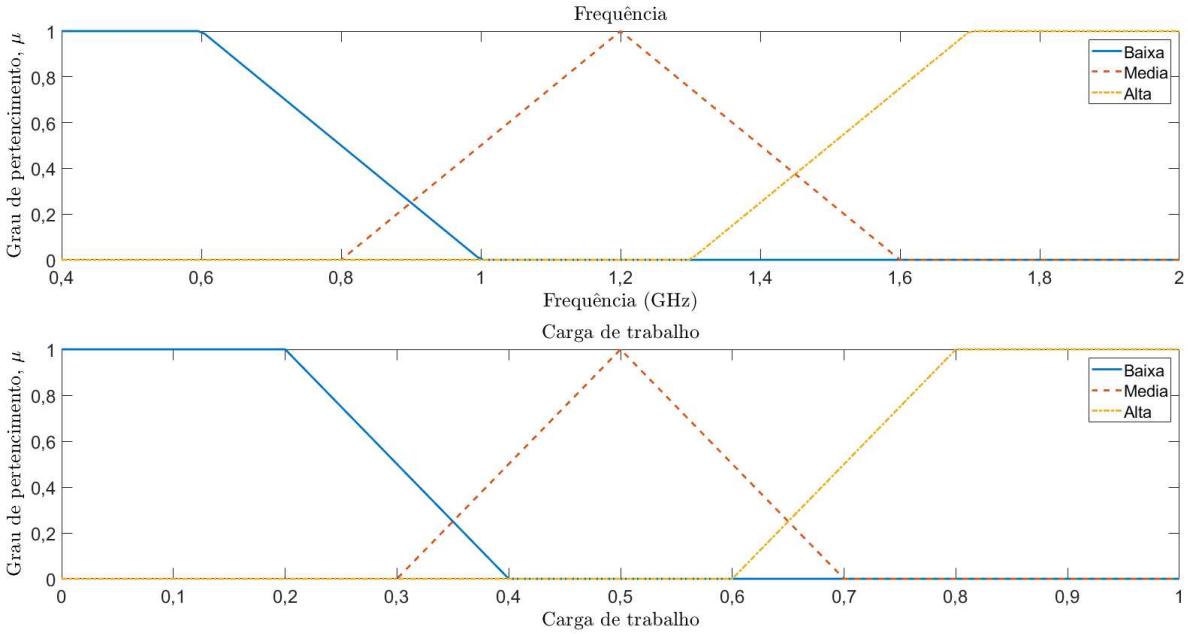

Figura 3.3: Funções de pertencimento para as variáveis linguísticas frequência F e carga de trabalho W.

Fonte: Autor

Este processo de calcular graus de pertencimento através de um valor de uma variável linguística é chamado de *fuzzyficação*. Funções de pertencimento são funções de uma variável, dessa forma os sistemas Fuzzy lidam com cada entrada separadamente, e estas entradas são combinadas nas regras através de operadores lógicos como E ou OU.

3.4.3 Operadores lógicos

Operadores lógicos Fuzzy são uma extensão dos operadores lógicos *booleanos*. São iguais aos operadores *booleanos* para o caso dos graus de pertencimento serem 0 ou 1. Para o caso de duas variáveis linguísticas, existem diversas funções aplicáveis aos operadores lógicos E ou OU, chamadas de t-normas e s-normas.

Uma função $\bigtriangledown : [0,1]^2 \rightarrow [0,1]$ é dita ser comutativa, associativa e monotônica se $\bigtriangledown$ satisfaz as seguintes propriedades, respectivamente, para $\forall a.b \in [0,1]$[42]:

1. Comutatividade: $\bigtriangledown(a,b) = \bigtriangledown(b,a)$
2. Associatividade: $\bigtriangledown(a, \bigtriangledown(b,c)) = \bigtriangledown(\bigtriangledown(a,b),c)$
3. Monotonicidade $\bigtriangledown(a,b) \leq \bigtriangledown(c,d)$ se $a \leq c$ e $b \leq d$

Um operador $\top : [0,1]^2 \rightarrow [0,1]$ é denominado de t-norma se $\top$ é comutativo, associativo e monotônico e verifica a seguinte propriedade, para $\forall a \in [0,1]$:

Elemento neutro = 1: $\top(a,1) = a$

Um operador $\perp : [0,1]^2 \rightarrow [0,1]$ é denominado de s-norma se $\perp$ é comutativo, associativo e monotônico e verifica a seguinte propriedade, para $\forall a \in [0,1]$:

Elemento neutro = 1: $\perp(a,0) = a$

Uma t-norma e uma s-norma são duais em relação a uma operação de negação $\neg : [0,1] \rightarrow [0,1]$ se para $\forall a, b \in [0,1]$ satisfazem as leis de Morgan:

$$\neg(\top(a,b)) = \perp(\neg a, \neg b)$$

$$\neg(\perp(a,b)) = \top(\neg a, \neg b)$$

onde o operador de negação é dado por:

$$\neg a = 1 - a$$

São apresentadas na Tabela 3.2 as principais t-normas e s-normas duais mais utilizadas na literatura. A norma Zadeh representa a função miníma e máxima para t-norma e s-norma respectivamente. A norma Probabilista representa a função produto e soma algébrica para t-norma e s-norma respectivamente. A norma Lukasiewicz representa a função diferença limitada e soma limitada para t-norma e s-norma respectivamente [41].

Tabela 3.2: Principais t-normas e s-normas duais.

Nome	t-norma	s-norma
Zadeh	$min(a,b)$	$max(a,b)$
Probabilista	$a.b$	$a + b - ab$
Lukasiewicz	$max(a + b - 1, a)$	$min(a + b,1)$

Para sistemas de classificação as funções máximas e mínimas são mais populares. Para aproximação e identificação, as funções de produto e soma algébrica são mais usadas devido a sua suavidade e diferenciabilidade e para alguns sistemas de aprendizado como neuro-fuzzy, as funções diferença e soma limitada podem oferecer vantagens.

3.4.4 Regras Fuzzy

Com as funções dos operadores lógicos, é possível combinar os graus de pertencimento de todas as variáveis linguísticas na premissa da regra. Usando o exemplo citado anteriormente, temos os graus de pertencimento para um valor de frequência de 0,85 GHz em baixo com $\mu_{baixo}(F) = 0{,}37$, médio com $\mu_{medio}(F) = 0{,}13$ e alto com $\mu_{alto}(F) = 0$, e uma carga de trabalho com intensidade de 0,66 em baixa com $\mu_{baixo}(W) = 0$ em média com $\mu_{medio}(W) = 0{,}2$ e alta com $\mu_{alta}(W) = 0{,}3$.

A combinação de todos os graus de pertencimento de todas as variáveis linguísticas chama-se grau de cumprimento das regras e expressa o quão bem as premissas de uma regra combinam com os valores de entrada específicos, neste exemplo, frequência de 0,85 GHz e carga de trabalho com intensidade de 0,66.

Desta forma, somente regras com grau de cumprimento acima de zero são relevantes para os valores de entrada recebidos. Todas as outras regras ficam inativas. No exemplo, regras que sejam ativadas com $\mu_{alto}(F)$ ou $\mu_{baixo}(W)$ não são ativadas para os valores de entrada recebidos. É necessário se tomar cuidado para que todo o espaço de entradas seja coberto pelas funções de pertencimento para evitar casos em que todas as regras Fuzzy são inativas.

3.4.5 Agregação

Após o grau de cumprimento de cada regra ser calculado para todas as regras Fuzzy, os consequentes devem ser calculados e acumulados para se gerar uma saída para o sistema Fuzzy. Este processo é chamado de *defuzzificação* e é responsável por obter um valor numérico para a saída. É importante notar que esta *defuzzificação* não é necessária para os casos onde a saída é apresentada diretamente para um humano. O processo para obter um valor numérico de saída depende dos tipos de sistema Fuzzy, os quais são descritos na próxima seção.

3.4.6 Sistemas de inferência

Esta seção irá apresentar os tipos de sistemas Fuzzy usados neste trabalho: Mamdani, *singleton* e Takagi-Sugeno. Apresentando seus conceitos básicos, como tipos de entradas, definições de regras, e tipos de saídas.

Sistemas Fuzzy tipo Mamdani

O sistema Fuzzy tipo Mamdani possui regras na forma:

$$R_1 : Se\ u_1 = A_{i1}\ E\ u_2 = A_{i2}\ E\ ...u_p = A_{ip}\ ENTÃO\ y = B_i$$

onde $u_1, \ldots,\ u_p$ são as p entradas do sistema Fuzzy, y é a saída, o indexador $i = 1, ...\ M$, percorre todas as M regras Fuzzy, A_{ij} condiz ao conjunto Fuzzy usado pela entrada u_j na regra i, e B_i é o conjunto Fuzzy usado pela saída na regra i.

Neste tipo de sistema Fuzzy, tanto as entradas quanto as saídas são descritas por variáveis linguísticas. Contudo a análise deste sistema exige um maior processamento computacional para sua validação. Desta forma, este tipo de sistema Fuzzy segue os passos:

Fuzzificação → Agregação → Ativação → Acumulação → Defuzzificação

Na fase de fuzzificação os graus de pertencimento de todas as variáveis linguísticas são calculados e denotados por $\mu_{ij}(u_j)$, $i = 1,...,M$ e $j = 1,...,p$.

Na fase de agregação estes graus de pertencimento são combinados de acordo com os operadores Fuzzy determinados nas regras do sistema. Para a t-norma do produto, o grau de cumprimento da regra, para a regra i será:

$$\mu_i(\underline{u}) = \mu_{i1}(u_1)\mu_{i2}(u_2)\ldots\mu_{ip}(u_p) \tag{3.5}$$

Na fase de ativação os graus de cumprimento das regras são usados para calcular a ativação das saídas de cada regra. Na fase de acumulação, as saídas de ativação de todas as regras são colocadas juntas. A acumulação produz um conjunto Fuzzy, que é o conjunto de saída do sistema Fuzzy. Se nenhum valor numérico for necessário para a saída, o mecanismo de inferência para aqui.

Um valor numérico de saída pode ser calculado pelo passo final, chamado defuzzificação. Existem diversos métodos para o cálculo deste valor, sendo os mais comuns o método de centro-da-área (C-o-A), centro-do-máximo (C-o-M) e a Média-do-máximo (M-o-M).

O método de defuzzificação C-o-A e C-o-M são contínuos e utilizam o máximo das funções de pertinência e o método M-o-M é descontínuo. O método C-o-A é mais rápido e muito usado em *softwares* e processadores de lógica Fuzzy, porém ignora a superposições de funções de pertinência.

O método C-o-A, calcula o centróide da área composta que representa o termo de saída Fuzzy (μ_{OUT}), este termo de saída é formado pela união de todas as contribuições de regras. O centróide é um ponto que divide a área de μ_{OUT} em duas partes de mesma área. É descrito na expressão (3.6) o cálculo do centróide da área.

$$y^{CoA} = \frac{\sum_{i=1}^{N} u_i \mu_{OUT}(u_i)}{\sum_{i=1}^{N} \mu_{OUT}(u_i)} \tag{3.6}$$

onde $\mu_{OUT}(u_i)$ é a área de uma função de pertinência modificado pelo resultado da inferência nebulosa e u_i é a área do centróide da função de pertinência individual.

O método C-o-M determina o valor mais comum de cada termo e então processa o valor que indica melhor veracidade com o número real. As pertinências deste resultado são consideradas "pesos" de valores típicos de termos rotulados. Logo, o valor que melhor se adapta é o "ponto de equilíbrio" dos pesos por meio de média ponderada. É apresentado na expressão (3.7) como este método é calculado, no qual $\mu_{0,k}(u_i)$, $k = 1, 2, ..., n$ são os pontos que ocorrem os máximos das funções de pertinência da saída.

$$y^{CoM} = \frac{\sum_{i=1}^{N} u_i \sum_{k=1}^{n} \mu_{0,k}(u_i)}{\sum_{i=1}^{N} \sum_{k=1}^{n} \mu_{0,k}(u_i)} \tag{3.7}$$

O método M-o-M seleciona o valor mais plausível entre os obtidos e despreza os demais, desconsiderando o formato das funções de pertinência da saída. Realizando a média de todos os máximos, como apresentado em (3.8).

$$y^{MoM} = \sum_{m=1}^{M} \frac{u_m}{M} \tag{3.8}$$

Onde u_m é o $m-$ésimo elemento do universo de discurso, onde a função $\mu_{OUT}(u_i)$ tenha um máximo e M é o número total destes elementos.

Para aplicações em malha fechada, que possuem propriedade de continuidade, na qual a saída de controle Fuzzy influencia a variável de um processo, indica-se a defuzzificação C-o-M. Para reconhecimento de padrões, indica-se o método M-o-M e para suporte à decisão o método de defuzzificação depende do contexto, podendo-se utilizar M-o-M ou C-o-M [43].

Sistemas Fuzzy tipo *Singleton*

A complexidade da fase de deffuzificação no sistema tipo Mamdani pode ser simplificada consideravelmente pelo uso de funções de pertencimento para as saídas do tipo *singleton*. As regras para o sistema tipo *singleton* são da forma:

$$R_i : Se\ u_1 = A_{i1}\ E\ u_2 = A_{i2}\ E\ ...u_p = A_{ip}\ ENT\tilde{A}O\ y = s_i$$

onde s_i é um valor real, chamado de *singleton* da regra i. Neste tipo de sistema, os processos de fuzzificação, agregação e ativação são idênticos aos do sistema Mamdani, porém a saída do sistema pode ser calculada como apresentado em (3.9), onde M denota o número de regras e $\mu_i(\underline{u})$ é o grau de cumprimento da regra i.

$$y = \frac{\sum_{i=1}^{M} s_i \mu_i(\underline{u})}{\sum_{i=1}^{M} \mu_i(\underline{u})} \tag{3.9}$$

O grau de cumprimento da regra i depende do tipo de t-norma usada. Para a t-norma produto, pode-se definir o grau de cumprimento para a regra i como apresentado em (3.10).

$$\mu_i(\underline{u}) = \mu_{i1}(u_1) \cdot \mu_{i2}(u_2) \cdot ... \cdot \mu_{ip}(u_p) \tag{3.10}$$

Sistemas Fuzzy tipo Takagi-Sugeno

Em 1985 Takagi-Sugeno [44] propuseram um novo tipo de sistema Fuzzy com regras do tipo:

$$R_i : Se\ u_1 = A_{i1}\ E\ u_2 = A_{i2}\ E\ ...u_p = A_{ip}\ ENTÃO\ y = f_i(u_1, u_2, ..., u_p)$$

Tal tipo de regra pode ser visto como uma extensão dos sistema Fuzzy do tipo *singleton*. Enquanto o tipo *singleton* ainda pode ser visto como um tipo especial de função de pertencimento, as funções $f_i(u_1, u_2, ..., u_p)$ definitivamente não são conjuntos Fuzzy. Somente a premissa do tipo Takagi-Sugeno pode ser linguisticamente interpretada.

Se as funções $f_i(u_1, u_2, ..., u_p)$ forem escolhidas como constantes, teremos então um modelo do tipo *singleton*, e neste caso também amplamente chamado de sistema Fuzzy Takagi-Sugeno de ordem zero. Comumentemente, as aplicações para o sistema Takagi-Sugeno se dão com f_i de primeira ordem. Para sistemas com mais de uma entrada, a saída y é obtida pela ponderação das várias saídas obtidas por cada regra, como apresentado em (3.11).

$$y = \frac{\sum_{i=1}^{M} w_i \mu_i(\underline{u})}{\sum_{i=1}^{M} w_i} \tag{3.11}$$

onde M é o número de regras do controlador,$\mu_i(\underline{u})$ é o grau de cumprimento da regra i como exemplificado em (3.10), y_i é a saída referente à regra i, w_i é o peso atribuído à regra i. Este peso é determinado como o menor grau de pertencimento entre os pertencentes à regra i, como apresentado na Figura 3.4. O peso w_1 é determinado como o menor grau de pertencimento das entradas u_1 e u_2 aos grupos A_{11} e A_{12}; e o peso w_2 é determinado como o menor grau de pertencimento das mesmas entradas aos grupos A_{21} e A_{22}.

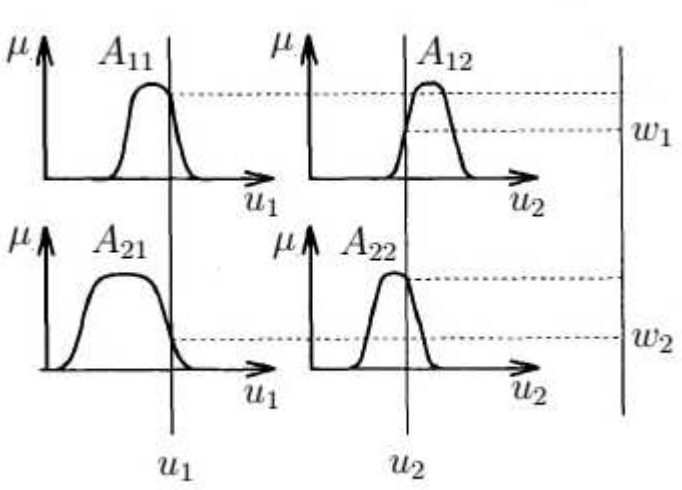

Figura 3.4: Definição dos pesos para o controlador Takagi-Sugeno.
Fonte: Adaptado de Jang [45].

3.4.7 Classificação dos Sistemas Fuzzy

Os tipos de sistemas Fuzzy apresentados na Seção 3.4.6 podem ser classificados como sistemas estáticos, sistemas adaptativos ou sistemas recorrentes.

Sistemas Fuzzy estáticos

Os sistemas Fuzzy estáticos são definidos como um modelo de processamento direto, ou seja, sem nenhum tipo de realimentação. Suas regras, funções de pertencimento e funções de saídas são determinadas geralmente por conhecimento especialista e não usam nenhum tipo de algoritmo de aprendizagem nas suas determinações.

Sistemas Fuzzy adaptativos

Os sistemas Fuzzy adaptativos são definidos como sistemas Fuzzy em conjunto com algum algoritmo de aprendizagem, seja para a determinação das regras, ou das suas funções de pertencimento, ou ambos. Seja $X \in U$, onde $U \subseteq \Re^n$ é o universo de discurso para X, e A é um subconjunto Fuzzy em U. O sistema Fuzzy adaptativo possui como características:

1. Base de regras: Sejam M regras do tipo SE-ENTÂO com funções de pertinência gaussianas:

$$\mu_{ij}(u_j) = exp\left[-\left(\frac{u_j - \bar{u_{ji}}}{\sigma_{ij}}\right)^2\right] \tag{3.12}$$

   onde $\bar{u_{ji}}$ e $\sigma_{ij} > 0$ são parâmetros da função gaussiana, ambos ajustáveis.

2. Fuzzificador: O mecanismo de fuzzificação executa um mapeamento da entrada $u \in U$ para as imagens $\mu_{ij}(u_j)$ das funções de pertencimento adotadas em (3.12).

3. Deffuzificador: A deffuzificação é feita como apresentada em (3.11). Dessa forma, a partir da t-norma do produto, o sistema Fuzzy possui a saída apresentara em (3.13), aplicando (3.12) em (3.11).

$$y = \frac{\sum_{i=1}^{M} w_i \left[\prod_{i=1}^{p} exp\left(-\left(\frac{u_j - \overline{u}_{ji}}{\sigma_{ij}}\right)^2\right)\right]}{\sum_{i=1}^{M} w_i} \tag{3.13}$$

   A saída apresentada em (3.13) é utilizada como iniciação para o modelo, pois usando técnicas de agrupamento (*clustering*) é possível definir durante o treinamento o número de regras M, como também os parâmetros das funções de pertencimento e funções de saída.

3.5 Sistema de Inferência Neuro-Fuzzy Adaptativo

O Sistema de Inferência Neuro-Fuzzy Adaptativo (*Adaptive Neuro-Fuzzy Inference System* - ANFIS) é uma rede neural proposta por Jang (1993), cuja ideia básica é de implementar um sistema de inferência Fuzzy por meio de uma arquitetura paralela distribuída, neste caso, a de uma Rede Neural Artificial (RNA). Dessa forma que os algoritmos de aprendizado possam ser usados para ajustar este sistema de inferência Fuzzy [46].

As entradas associados com as funções de pertencimento são ajustados via um algoritmo de aprendizado. O ajuste destes parâmetros é efetuado utilizando o algoritmo *back propagation* ou uma combinação deste com um algoritmo do tipo mínimos quadrados. Esta estrutura programa sistemas do tipo Takagi-Sugeno, com funções lineares ou constantes nos consequentes das regras que formam o sistema, tendo estas regras pesos unitários [47].

3.5.1 Arquitetura ANFIS

A arquitetura de um sistema ANFIS consiste em uma rede adaptativa que usa algoritmos de aprendizagem, possuindo uma forma similar ao sistema Fuzzy do tipo Takagi-Sugeno para tratar das partes de premissa e consequente. É apresentado na Figura 3.5 a arquitetura geral do ANFIS, consistindo em cinco camadas. A primeira e quarta camadas contem os nós adaptativos, enquanto as demais contém nós fixos. Cada camada do ANFIS é melhor explicada a seguir:

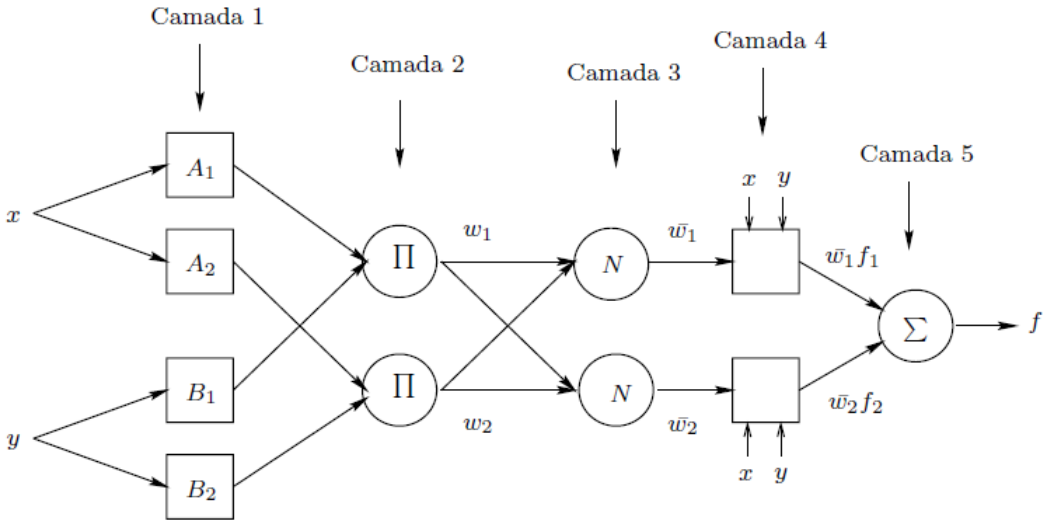

Figura 3.5: Arquitetura do sistema ANFIS.
Fonte: Adaptado de Huamani [47].

1. Camada 1: Cada nó nesta camada adapta uma função de pertencimento de algum parâmetro de entrada. A saída de cada nó é o valor de grau de pertencimento atribuído à entrada correspondente à função de pertencimento adaptada. Por exemplo, a função de pertencimento pode ser do tipo gaussiana, ou do tipo sino, apresentadas em (3.14) e (3.15) respectivamente.

$$\mu_{Ai}(x) = exp\left[-\left(\frac{x - c_i}{2a_i}\right)^2\right] \tag{3.14}$$

$$\mu_{Ai}(x) = \frac{1}{1 + \left|\frac{x-c_i}{a_i}\right|^{2b_i}} \tag{3.15}$$

   As saídas da camada são os graus de pertencimento $\mu_{Ai}(x)$ para $i = 1{,}2$ ou $\mu_{Bi}(y)$ para $i = 3{,}4$ como apresentado em (3.16) e (3.17) respectivamente.

$$O_{1,i} = \mu_{Ai}(x) \tag{3.16}$$

$$O_{1,i} = \mu_{Bi}(x) \tag{3.17}$$

   onde os graus $\mu_{Ai}(x)$ e $\mu_{Bi}(y)$ são referentes aos conjuntos Fuzzy A_i e B_i respectivamente e a_i, b_i, c_i são os parâmetros que podem mudar o formato das funções de pertencimento.

2. Camada 2: Cada nó desta camada é fixo e não-adaptativo. A saída deste nó é o resultado de uma função do operador E, ou seja, uma função min, uma função produto ou uma função diferença limitada, dos sinais de entrada como apresentado em (3.18) para $i = 1{,}2$. Onde $\cdot$ representa a função do operador E, sendo o produto a mais comum de ser usada. Cada saída deste nó representa o peso de cada regra, representado por w_i para a regra i.

$$O_{2,i} = w_i = \mu_{Ai}(x) \cdot \mu_{Bi}(y) \tag{3.18}$$

3. Camada 3: Cada nó desta camada é fixo e não adaptativo. Cada nó representa uma relação entre o peso da regra i com a soma do peso de todas as regras existentes, como apresentado em (3.19)

$$O_{3,i} = \overline{w}_i = \frac{w_i}{\sum_i w_i} \tag{3.19}$$

4. Camada 4: Cada nó desta camada é adaptativo para uma certa saída. A saída de cada nó é apresentado em (3.20).

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i(p_i x + q_i y + r_i) \tag{3.20}$$

onde $\overline{w}_i$ é a saída da camada anterior (terceira saída) para a regra i e $(p_i x + q_i y + r_i)$ são os parâmetros do nó, chamados de parâmetros conseguentes.

5. Camada 5: O nó único nesta camada é fixo e não-adaptativo. Este computa a saída global do sistema como uma soma de todos os sinais de entrada do nó anterior, como apresentado em (3.21).

$$O_{5,i} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{3.21}$$

3.5.2 Algoritmo de aprendizado híbrido

Na arquitetura ANFIS, a primeira e quarta camada contêm os parâmetros que podem ser modificados com o tempo. A primeira camada contêm os parâmetros de premissa enquanto a quarta camada contêm os parâmetros consequentes. Para a estimação destes parâmetros é necessário um método de aprendizagem que possa treinar premissa e conseguente, desta forma, um algoritmo híbrido foi proposto por [48].

Existem duas partes do algoritmo, chamadas de caminho direto e caminho contrário. Na parte de caminho direto os parâmetros da premissa na primeira camada devem está em regime permanente, dessa forma, uma estimação recursiva por mínimos quadrados (*Recursive least square estimator*, RLSE) é aplicada no consequente da quarta camada e por serem parâmetros lineares, este método de estimação pode ser aplicado para acelerar a convergência dos mesmos. Após os parâmetros do consequente serem obtidos, os dados de entrada são passados novamente para a rede neural adaptativa e a saída gerada é comparada com o saída atual.

Enquanto a parte de caminho contrário está sendo executada, os parâmetros consequentes devem está em regime permanente. Assim, os parâmetros da premissa na primeira camada são estimados usando o método de aprendizagem do gradiente descendente ou retropropagação.

Método de aprendizagem retropropagação

O algoritmo de retropropagação consiste em basicamente duas etapas [48]:

1. Etapa *foward*: As entradas são apresentadas e propagadas camada por camada mantendo os pesos fixos; a saída é comparada com a saída desejada, dessa forma, é obtido o sinal de erro.

2. Etapa *backward*: O erro é retropropagado, sendo os pesos atualizados via o método do gradiente descendente.

Suponha que uma dada rede adaptativa possui L camadas e a $k-$ésima camada possui $\Omega(k)$ nós. Podemos assumir o nó na posição i da camada k por (k,i) e a saída desse nó por O_i^k. Como a saída deste nó depende dos sinais de entrada e seus parâmetros, temos:

$$O_i^k = O_i^k(O_1^{k-1}, \ldots O_{\Omega(k-1)}^{k-1}, a, b, c, \ldots) \tag{3.22}$$

onde $a, b, c, \ldots$ são os parâmetros deste nó, vindos das equações (3.14) e (3.15). Também é usada a mesma notação para a saída do nó e sua função.

Seja um conjunto de dados de treinamento com P entradas, o objetivo e diminuir a função objetivo que pode ser definida como a medida de erro para a entrada p onde $(1 \leq p \leq P)$ dos dados de treinamento como a soma dos erros quadráticos.

$$E_p = \sum_{m=1}^{\Omega(L)} (T_{m,p} - O_{m,p}^L)^2 \tag{3.23}$$

onde $T_{m,p}$ é o $m-$ésimo componente da saída para a entrada p, e $O_{m,p}^L$ é o $m-$ésimo da saída atual.

Método de aprendizagem RLSE

Para a obtenção dos parâmetros consequentes, pode-se realizar uma combinação linear dos mesmos [45]:

$$\begin{aligned} f &= \bar{w}_1 f_1 + \bar{w}_2 f_2 \\ &= \bar{w}_1(p_1 x + q_1 y + r_1) + \bar{w}_2(p_2 x + q_2 y + r_2) \\ &= (\bar{w}_1 x)p_1 + (\bar{w}_1 y)q_1 + (\bar{w}_1)r_1 + (\bar{w}_2 x)p_2 + (\bar{w}_2 y)q_2 + (\bar{w}_2)r_2 \end{aligned} \tag{3.24}$$

Com N dados de treinamento fornecidos por (3.24), podemos reescrevê-la como:

$$\begin{aligned} f_1 &= (\bar{w}_1 x)_1 p_1 + (\bar{w}_1 y)_1 q_1 + (\bar{w}_1)_1 r_1 + (\bar{w}_2 x)_2 p_2 + (\bar{w}_2 y)_2 q_2 + (\bar{w}_2)_2 r_2 \\ &\vdots \\ f_n &= (\bar{w}_1 x)_n p_1 + (\bar{w}_1 y)_n q_1 + (\bar{w}_1)_n r_1 + (\bar{w}_2 x)_n p_2 + (\bar{w}_2 y)_n q_2 + (\bar{w}_2)_n r_2 \end{aligned} \tag{3.25}$$

Para simplificar a equação (3.25), pode-se reescrevê-la na forma matricial como apresentado em (3.26).

$$A\theta = y \tag{3.26}$$

onde θ é um vetor $M \times 1$. M se refere ao número de elementos que são do conjunto de parâmetros do consequente. A é uma matriz $P \times M$, onde P é o número de N dados de treinamento provindos da rede adaptativa. A saída y é um vetor $P \times 1$ cujo elementos são N dados de saída providos de uma rede adaptativa.

Normalmente o número de dados de treinamento é maior que o número de parâmetros consequentes a serem estimados, dessa forma, uma solução para θ pode ser obtida pela minimização do erro quadrado $||A\theta = y^2||$. Por um estimador de mínimos quadrados, θ pode ser definido como (3.27):

$$\theta^* = (A^T A)^{-1} A^T y \tag{3.27}$$

Usando estimação dos mínimos quadrados recursivos, a equação (3.27) se torna:

$$\begin{aligned} \theta_{i+1} &= \theta_i + P_{i+1} a_{i+1} (y_{i+1}^T - a_{i+1}^T \theta_i) \\ P_{i+1} &= P_i - \frac{P_i + a_{i+1} a_{i+1}^T P_i}{1 + a_{i+1}^T P_i a_{i+1}}, i = 1, 2, \ldots, P-1 \end{aligned} \tag{3.28}$$

onde a_i^T é um vetor linha de cada linha de A, y_i é o $i-$ésimo elemento de y. P_i é chamado de matriz de covariância e é definida por (3.29).

$$P_i = (A^T A)^- 1 \tag{3.29}$$

3.6 Método de clusterização Fuzzy C-Means (FCM)

O algoritmo FCM tem como objetivo encontrar grupos Fuzzy para um conjunto de dados. Para alcançar este objetivo, o algoritmo precisa minimizar uma função de custo associada às distâncias entre os dados e os centros dos grupos aos quais tais dados pertencem com algum grau de pertinência [49]. A função de custo é apresentada em (3.30), onde n é o número de dados a serem agrupados, c é o número de *clusters* adotados, m é um número real positivo, com $m > 1$, x_i é o $i-$ésimo dado a ser agrupado de um vetor de dados $\mathbf{X} = [x_1, x_2, \cdots, x_n]$, c_j é o $j-$ésimo *cluster* de um vetor de *clusters* $\mathbf{C} = [c_1, c_2, \cdots, c_c]$ a serem determinados e μ_{ij} é o grau de pertencimento do dado x_i ao *cluster* c_j.

$$J_{FCM} = \sum_{i=1}^{c} \sum_{j=1}^{n} (\mu_{ij})^m ||x_i - c_j||^2 \tag{3.30}$$

Os valores iniciais dos graus de pertencimento μ_{ij} são definidos de maneira aleatória e usados para atualizar os valores dos *clusters* c_j como apresentado em (3.31).

$$c_j = \frac{\sum_{i=1}^{n}(\mu_{ij})^m x_i}{\sum_{i=1}^{n}(\mu_{ij})^m} \tag{3.31}$$

Quando os valores dos *clusters* são atualizados, os valores dos graus de pertencimento de cada dado também são atualizados, como apresentado em (3.32).

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c}(\frac{||x_i - c_j||}{||x_i - c_k||})^{\frac{2}{m-1}}} \tag{3.32}$$

Dessa forma, a função de custo (3.30) é calculada novamente e esse processo se repete até J_{FCM} ser menor que o erro mínimo adotado ou até o número máximo de interações ser alcançado.

Os graus de pertencimento μ_{ij} podem ser agrupados em uma matriz, chamada de matriz de pertencimento Fuzzy, definida por (3.33).

$$\mu_{\mathbf{FCM}} = \begin{bmatrix} \mu_{11} & \mu_{12} & \dots & \mu_{1n} \\ \mu_{21} & \mu_{22} & \dots & \mu_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{c1} & \mu_{c2} & \dots & \mu_{cn} \end{bmatrix} \tag{3.33}$$

O algoritmo FCM é apresentado a seguir:

1. Determine o valor de m;
2. Determine a quantidade de *clusters* c;
3. Determine o valor do erro máximo adotado no processo e o número máximo de iterações;
4. Inicie a matriz de pertencimento Fuzzy aleatoriamente;
5. Calcule os *clusters* usando (3.31);
6. Atualize os valores dos graus de pertencimento usando (3.32);
7. Calcule a função de custo (3.30);
8. Repita os passos 5, 6 e 7 até que J_{FMC} seja menor que o valor determinado para o erro máximo ou até que se alcance o número máximo de iterações determinadas.

3.7 Método de clusterização subtrativa

O método *subtrative clustering* é uma extensão do método *mountain clustering* proposto por R. Yager [50]. Foi desenvolvido por Stephen L. Chui [51] e tem por objetivo estimar os centros dos *clusters* e quantidade de *clusters* de um conjunto de dados.

Este método assume que cada ponto é um centro de *cluster* em potencial e calcula para cada ponto do conjunto de dados um potencial que representa a probabilidade daquele ponto ser um centro do *cluster*. Este cálculo é baseado no comportamento da densidade dos n pontos que constituem o conjunto de dados. O potencial é calculado por meio de (3.34):

$$P_i = \sum_{j=1}^{n} e^{-\alpha||x_i - x_j||^2} \tag{3.34}$$

onde $\alpha = \frac{4}{r_a^2}$ e r_a é uma constante positiva.

O potencial é calculado em função da distância entre o possível centro de *cluster* e os outros pontos. O ponto que tiver uma maior quantidade de vizinhos terá um maior potencial e será o primeiro centro de *cluster*. A constante r_a representa o raio do *cluster* e desta forma limita a vizinhança.

O algoritmo de clusterização subtrativa realiza basicamente os seguintes passos:

1. Depois de calculado o potencial de cada um dos pontos que constituem o conjunto de dados que representa o problema, escolhe-se o primeiro centro de *cluster* (x_1^*, P_1^*) e este será o que tiver maior potencial.

2. Escolhido o primeiro centro de *cluster* é realizada uma correção dos potenciais dos dados não escolhidos como centro de *clusters* usando (3.35).

$$P_i = P_i - P_1^* e^{-\beta||x_i - x_1^*||^2} \tag{3.35}$$

   onde $\beta = \frac{4}{r_b^2}$ e r_b é uma constante positiva maior que r_a para escolher um novo centro de *cluster* longe da vizinhança do centro de *cluster* já escolhido. Normalmente admite-se $r_b = 1{,}5r_a$.

3. Escolhido o segundo centro de *cluster*, repete-se o passo anterior de forma a escolher todos os centros de *cluster* que representam o conjunto de dados.

4. Realizar o número de iterações necessário até que todos os centros de *clusters* sejam encontrados.

O algoritmo utiliza como critério de parada um coeficiente de aceitação e de rejeição de um dado em se tornar um centro de *cluster*, ou seja, ele limita a região de dados que podem ser centros de *clusters* [51]. O processo de clusterização permite informar a faixa de influência do centro de *cluster* para cada dimensão dos dados, ou seja, se tivermos um conjunto de dados com duas entradas podemos informar para cada entrada uma faixa de influência. Considera-se cada centro de *cluster* como uma regra Fuzzy que descreve o comportamento do sistema, ou seja, o número de regras será igual a quantidade de *clusters* do sistema. O processo de clusterização irá determinar o número de regras e funções de pertinência dos antecedentes e depois utilizará o método de estimação dos mínimos quadrados para gerar os consequentes de cada regra[52].

3.8 Conclusão

Neste capítulo foram apresentados os conceitos básicos que serão usados no desenvolvimento do trabalho. Foi apresentado como ocorre o consumo de energia em processadores, especificando cada parcela da potência consumida. Uma introdução ao escalonamento de tensão e frequência para processadores. Foi apresentado o modelo de tarefas a ser usado, importante para se determinar a utilização e frequência a ser adotada pelo processador e os conceitos básicos da Lógica Fuzzy, técnica usada no desenvolvimento do controlador deste trabalho. Dando base aos próximos capítulos deste trabalho sobre o desenvolvimento do controlador Fuzzy e da plataforma experimental que será usada nos ensaios.

Capítulo 4

Plataforma Experimental

Será apresentado a seguir a plataforma de medição e aquisição de dados que foi desenvolvida no trabalho com o intuito da realização de experimentos e aquisição de dados do processador e consumo de energia, usando-se o controlador desenvolvido e os *Governors* já presentes no sistema Linux, para que seja realizada a comparação entre todos os métodos. Também será apresentado o sistema em que se deseja realizar a redução do consumo de energia.

4.1 Aquisição de dados

A aquisição de dados pode ser dividida em duas partes principais. A primeira consiste na aquisição de dados do processador, obtidas diretamente por medições no sistema operacional e a segunda parte na aquisição de dados de consumo de potência, obtidas por medições externas ao computador.

4.1.1 Aquisição de dados do processador

A plataforma experimental usada é um computador pessoal composto por um processador *Intel i5-6500* com um conjunto de frequências {800 MHz, 1 GHz,1,1 GHz, 1,3 GHz, 1,5 GHz, 1,7 GHz, 1,8 GHz, 2 GHz, 2,2 GHz, 2,3 GHz, 2,5 GHz, 2,7 GHz, 2,9 GHz, 3 GHz, 3,2 GHz}, 8 GB de memória e 1 TB de disco rígido. O sistema operacional adotado foi o *Linux Mint 18.3 Sylvia Cinnamon* 64-bit.

O arquivo /proc/stat[1] foi usado para realizar a medição da utilização e ociosidade do processador. Tal arquivo possui uma variedade de diferentes dados do sistema desde que esse é iniciado pela última vez. Um exemplo do seu conteúdo em respeito ao processador e seus núcleos é apresentado a seguir:

[1] http://www.linuxhowtos.org/manpages/5/proc.htm

```
>> cat /proc/stat

cpu 79242 0 74306 842486413 756859 6140 67701
cpu0 49663 0 40234 104757317 542691 4420 39572
cpu1 2724 0 2118 105420424 767 1719 6084
cpu2 18578 0 18430 105191522 204592 0 714
cpu3 513 0 979 105428698 739 0 2907
cpu4 1623 0 2105 105426291 444 0 3373
cpu5 3491 0 5326 105414798 7134 0 3087
cpu6 1636 0 3081 105420689 201 0 8229
cpu7 1011 0 2029 105426670 288 0 3731
```

Os intervalos de amostragem dos núcleos dos processadores são em unidades de tempo USER_HZ or *Jiffies* (usualmente milésimos de segundos, e podem mudar de acordo com a arquitetura ou distribuição usada). Os números do arquivo /proc/stat são valores de contadores que são incrementados a cada intervalo de amostragem em que o processador ou determinado núcleo realiza uma determinada atividade. Como a utilização e ociosidade são calculadas em valores absolutos, não é necessário nenhum tipo de conversão para estes contatores. Percebeu-se que intervalos de amostragem abaixo de $50ms$ podem resultar em más medições devido a não atualização dos valores numéricos no arquivo /proc/stat.

A primeira linha "cpu" do arquivo /proc/stat contêm dados de todos os núcleos do processador. As próximas linhas "cpui", onde $i = 0, 1, 2, 3, ..., 7$, contêm dados de cada núcleo no processador. Cada coluna destas linhas se refere a um tipo de atividade do núcleo em questão.

- 1º (*User*): Processos executando no modo de usuário;
- 2º (*Nice*): Processos de alta prioridade executando em modo de usuário;
- 3º (*System*): Processos executando no modo *Kernel*;
- 4º (*Idle*): Tempo de espera;
- 5º (*IOwait*): Espera por processos de entrada e saída;
- 6º (*Irq*): Interrupção de serviços;
- 7º (*Softirq*): Processos de manutenção.

Baseado nestes dados, é possível avaliar a utilização de cada núcleo do processador i. A soma de todas estas colunas é chamada de t_{total} e a soma das colunas *Idle* e *IOwait* é chamado de t_{idle}. Medições destas variáveis são feitas a cada intervalo de amostragem. O intervalo decorrido Δt_{total} e o intervalo de ociosidade Δt_{idle} são mostrados em (4.1) e (4.2) respectivamente[53].

$$\Delta t_{total}(t) = t_{total}(t) - t_{total}(t-1) \tag{4.1}$$

$$\Delta t_{idle}(t) = t_{idle}(t) - t_{idle}(t-1) \tag{4.2}$$

A utilização y e a ociosidade ι são determinadas como apresentado em (4.3) e (4.4) respectivamente.

$$y(t) = \frac{\Delta t_{total}(t) - \Delta t_{idle}(t)}{\Delta t_{total}(t)} \tag{4.3}$$

$$\iota(t) = \frac{\Delta t_{idle}(t)}{\Delta t_{total}(t)} = 1 - y(t) \tag{4.4}$$

4.1.2 Aquisição do consumo de potência

Para a aquisição do consumo de potência, foi usado o medidor *WattsUp*, capaz de realizar medições de diversas grandezas como potência, tensão, corrente, frequência e fator de potência.

Este medidor foi desenvolvido pela Vernier[2] e é apresentado na Figura 4.1. Este medidor pode ser dividido em duas partes principais, uma contendo sua unidade digital composta por um Microchip PIC18F45J10[3] e quatro memórias 64K 24C512N EEPROMs[4], responsáveis pelo gerenciamento da placa e painel LCD e para o armazenamento dos dados respectivamente. Sua segunda parte é responsável pelas medições oferecidas pelo equipamento, consistindo em um chip de medição de potência dedicada ADE7763[5], um medidor de energia ativo de fase única, capaz de lidar com todas as medições essenciais.

Este modelo de medidor possui como principais características:

- Tensão: 100-277 V;
- Frequência: 60/50 Hz;
- Corrente máxima: 15 A;

[2]https://www.vernier.com/
[3]http://ww1.microchip.com/downloads/en/DeviceDoc/39682E.pdf
[4]https://pdf.datasheetcatalog.com/datasheets/270/160591_DS.pdf
[5]http://www.analog.com/media/en/technical-documentation/data-sheets/ADE7763.pdf

Figura 4.1: Medidor WattsUP.
Fonte: Autor.

- Tempo de amostragem mínimo: 1 segundo;
- Precisão: ± 3% para cargas acima de 10 W e ± 5% para cargas abaixo de 10 W;
- Dimensões: 18x10x5 cm;
- Peso: 0,7 Kg;
- Comunicação: Interface USB.

4.1.3 Plataforma

O diagrama da plataforma experimental desenvolvida é apresentada na Figura 4.2. A rede de alimentação é conectada ao *WattsUp* e este alimenta o computador usado. Os dados de potência gerados pelo *WattsUp* são requisitados por USB para um *notebook* com relógio sincronizado com o computador pelo protocolo NTP[6] (*Network Time Protocol*). Os dados do processador e potência são arquivados em arquivos em forma de matrizes.

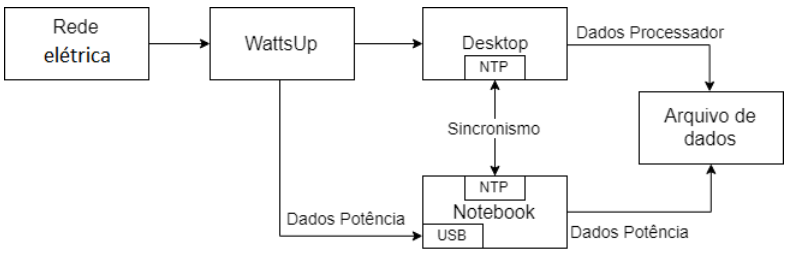

Figura 4.2: Diagrama da plataforma desenvolvida.
Fonte: Autor.

[6]https://ntp.br/guia-linux-comum.php

4.1.4 Algoritmo de aquisição do processador

O algoritmo em C desenvolvido para a aquisição usada no treinamento do modelo consiste nas etapas apresentadas no fluxograma da Figura 4.3(a). Inicialmente é feito a alocação da memória para as matrizes utilizadas, em seguida todos os núcleos são definidos com o *governor userspace* com o comando:

```
>> echo userspace > /sys/devices/system/cpu/cpui/cpufreq/scaling_governor
```

Onde i em "cpui" equivale ao núcleo do processador onde será realizada a alteração. É determinado o tempo de amostragem h, acima de $50ms$ para garantia da atualização do arquivo /proc/stat. É determinado o tempo de amostragem em cada frequência do processador T_f. Inicia-se a carga de trabalho desejada e em seguida a aquisição na frequência determinada. Para a determinação da frequência desejada usa-se o comando:

```
>> cpufreq-set -c i -f fa
```

Onde i equivale ao núcleo do processador onde será realizada a alteração e fa corresponde a frequência desejada em Hz. Quando o tempo T_f é atingido, acontece a mudança para a nova frequência desejada e repete-se até que todas as frequências sejam executadas, então, encera-se a carga de trabalho e cria-se os arquivos de dados.

Nos dados de treinamento foi usado os tempos de amostragem h de $100ms$ e T_f de $60s$. Além de serem executadas todas as frequências disponíveis para o processador.

A aquisição dos dados do processador com o controlador foi realizado praticamente da mesma forma, tendo a ausência do tempo T_f já que a frequência era determinada pelo próprio controlador e tempo de amostragem h também de $100ms$, porém o tempo de escrita no arquivo de dados era de $1s$ devido ao tempo mínimo de amostragem no *WattsUp* ser $1s$.

4.1.5 Algoritmo de aquisição da potência

A aquisição dos dados de potência ocorrem através da comunicação entre o *WattsUp* e um computador externo. Dessa forma também foi criado um *software* em C para esta aquisição, consistindo nas etapas apresentadas no fluxograma da Figura 4.3(b). Primeiramente ocorre a alocação da memória para as matrizes utilizadas, em seguida ocorre um teste de comunicação entre o *WattsUp* e o computador externo, caso esta falhe, é enviada uma mensagem de erro para que medidas de correção sejam tomadas. Caso não ocorram falhas, é determinado o tempo de amostragem k, com no mínimo $1s$ e ocorre a aquisição da potência e do horário do sistema e assim a escrita dos arquivos de dados de potência.

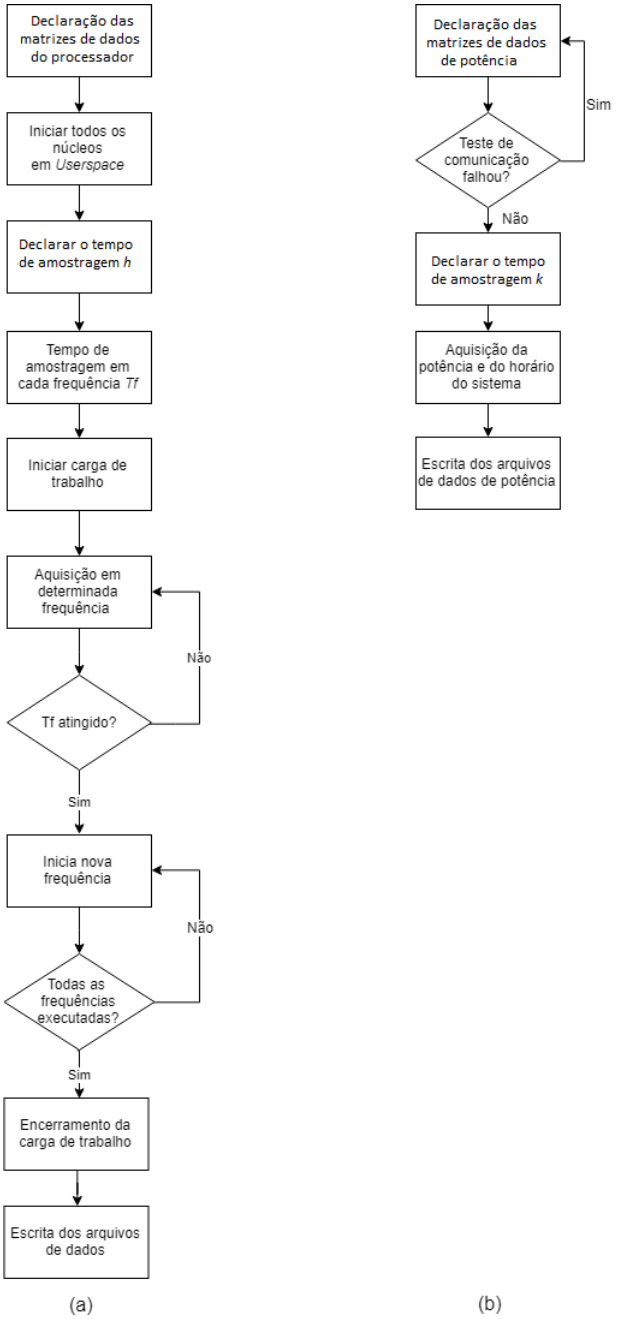

Figura 4.3: Fluxograma do *software* de aquisição de dados do processador (a) e da aquisição dos dados de potência(b).

Fonte: Autor.

4.2 Conclusão

Este capítulo apresentou a plataforma experimental que será usada neste trabalho, evidenciando o seu uso para a aquisição de dados do processador e dados de potência com o uso do *WattsUp* em forma de fluxogramas, explicando os *softwares* desenvolvidos para tais tarefas. Foi apresentada a metodologia usada para os cálculos dos dados do processador, evidenciando os arquivos utilizados, descrevendo-os. Para a potência foi apresentado

as etapas de funcionamento do *WattsUp*, evidenciando de maneira sucinta seus aspectos construtivos. Também foi apresentado o computador que será usado no trabalho, evidenciando seu processador.

Capítulo 5

Modelo Fuzzy do processador

Neste Capitulo será apresentado a metodologia e os modelos Fuzzy desenvolvidos para o processador. A primeira parte irá abordar metodologias para a identificação de modelos do tipo Fuzzy, apresentando as estruturas e os passos seguidos para a identificação computacional. As partes seguintes irão apresentar o desenvolvimento dos modelos com os dados aquisitados, como também sua validação.

5.1 Metodologia de identificação

Seja um sistema representado por M, um mapeamento do espaço de entrada para o espaço da saída. Seja C a família de modelos de sistemas não lineares tal que $S \in C$. O problema de identificação consiste em determinar uma coleção de modelos $\hat{C} \subset C$ e um mapeamento $\hat{M} \in \hat{C} \subset C$ tal que $\hat{M}$ aproxime M[54].

Dessa forma o problema de aproximação consiste em determinar um sistema aproximado $\hat{S}$, tal que:

$$||\hat{M} - M|| \leq \epsilon \tag{5.1}$$

no qual ϵ é um erro máximo permitido.

No caso de modelos Fuzzy, o problema de identificação consiste em selecionar um modelo $\hat{M}$ e ajustar suas funções de pertencimento e regras baseado no erro entre a saída do modelo e a saída real do sistema, de forma que a estrutura Fuzzy desenvolvida possua um comportamento idêntico ou mais próximo possível ao comportamento real do sistema modelado.

Existem duas estruturas para o processo de identificação Fuzzy que podem ser utilizadas:

1. Estrutura paralela: é apresentado na Figura 5.1 o modelo paralelo para um sistema não-linear com atrasos iguais a 1 em ambas realimentações. Nesta estrutura o sis-

tema real é realimentado por sua saída e o sistema modelado também é realimentado por sua saída.

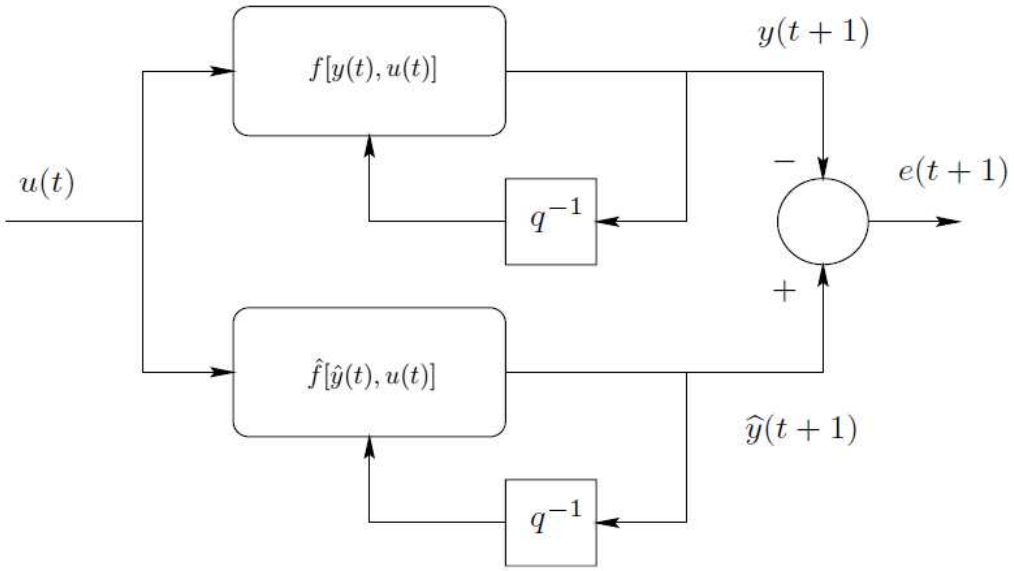

Figura 5.1: Estrutura paralela para identificação de sistemas.
Fonte: Autor.

O modelo pode ser descrito por (5.2).

$$\hat{y}(t+1) = \hat{f}[\hat{y}(t), u(t)] \tag{5.2}$$

2. Estrutura série-paralela: Nesta estrutura, tanto o sistema real quanto o modelo deste é realimentado pela saída real do sistema, como apresentado na Figura 5.2.

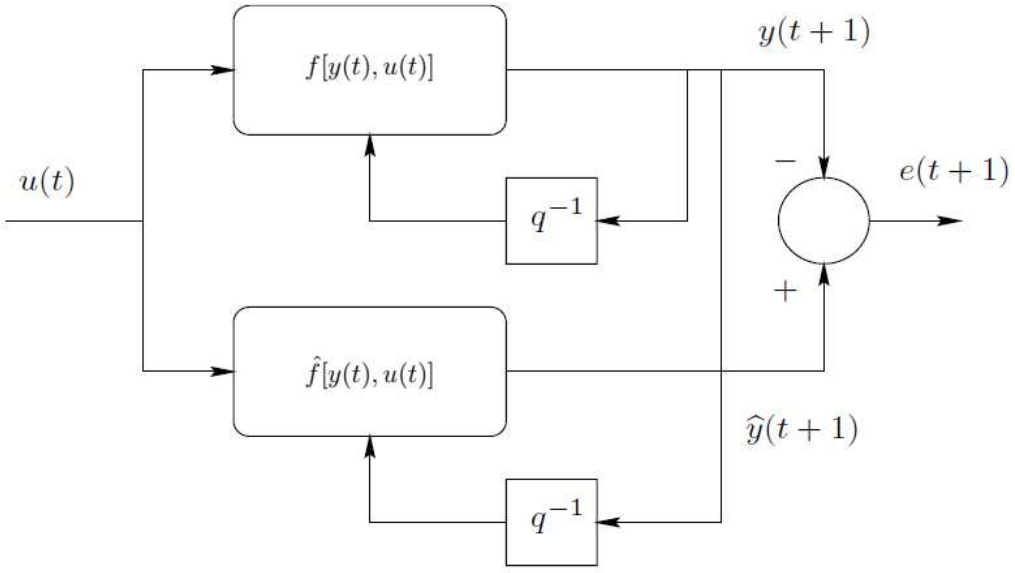

Figura 5.2: Estrutura série-paralela para identificação de sistemas.
Fonte: Autor.

O modelo pode ser descrito por (5.3)

$$\hat{y}(t+1) = \hat{f}[y(t), u(t)] \tag{5.3}$$

As principais vantagens em usar a estrutura série-paralela são:

(a) Dado que o sistema assumido é BIBO estável, todos os sinais de entrada utilizados durante o processo de identificação no modelo Fuzzy ou, qualquer outra abordagem, serão limitados.

(b) Devido à não existência do laço de realimentação no modelo de identificação, é possível utilizar o método de retropropagação para o ajuste dos parâmetros diminuindo assim, o custo computacional.

O processo de identificação de sistemas não-lineares pode ser desenvolvido de forma analítica ou computacional, podendo-se empregar as duas formas juntas ou separadas. A identificação analítica envolve a análise dinâmica do sistema e é desenvolvido um modelo matemático para o mesmo. A identificação computacional envolve a aquisição estatística das características de entrada e saída do sistema e seu uso no desenvolvimento de um modelo que aproxime o comportamento observado. Este tipo de identificação pode ser resumido nos seguintes passos:

1. Planejamento experimental: Determinação de como os dados serão aquisitados. Nesta etapa ocorre a escolha dos equipamentos ou métodos de medição que serão

adotados, taxas de amostragem, níveis adotados para entrada além da seleção das entradas e saídas do modelo.

2. Seleção da estrutura do modelo: Seleção da estrutura do modelo adotado (paralela ou série-paralela neste trabalho) e determinação dos parâmetros a serem ajustados (em modelos do tipo Fuzzy são número e tipo de funções de pertencimento para entradas e saídas, número de regras e demais características).

3. Estimação dos parâmetros: Ajuste dos parâmetros determinados no item anterior, utilizando os dados aquisitados sobre o sistema e os algorítimos de estimação.

4. Validação: Validação do modelo para os dados de teste. Existem muitas ferramentas estatísticas para validação de modelos, mas a principal ferramenta para a maioria das aplicações de modelagem de processos é a análise residual. Neste trabalho, estas duas técnicas serão unidas para a validação.

A abordagem para identificação utilizada neste trabalho pode ser ilustrada pela Figura 5.3. O modelo Fuzzy é conectado em paralelo com o sistema não linear e o sinal de erro entre as saídas do sistema e do modelo Fuzzy é utilizado para a atualização dos parâmetros ajustáveis do modelo.

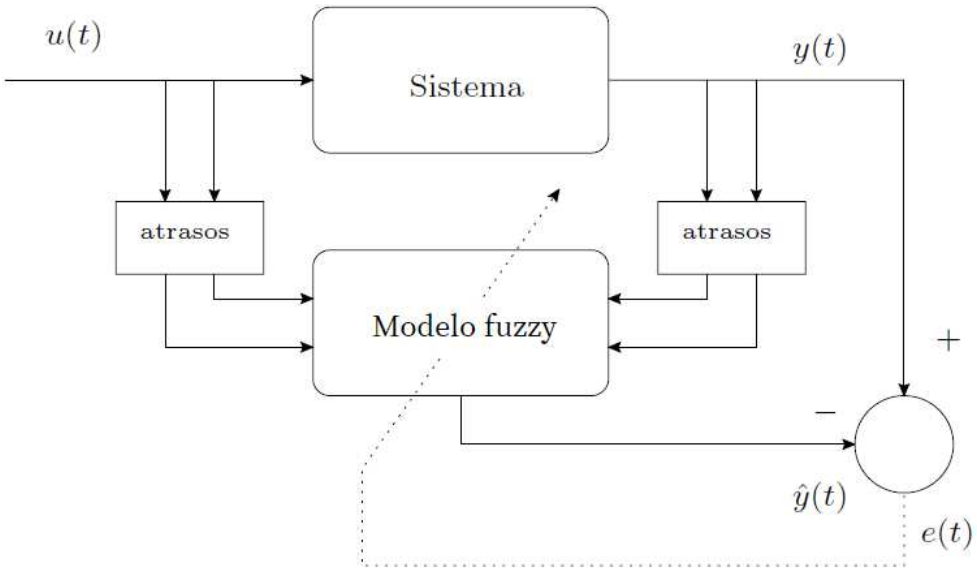

Figura 5.3: Identificação usando lógica Fuzzy.
Fonte: Autor.

5.2 Planejamento experimental

Luiz [2] descreveu algumas cargas de trabalho usuais em computadores de uso geral, mostrando a relação existente entre frequência e utilização. É apresentada nas Figuras (5.4) e (5.5) a utilização média em função da frequência, para uma carga de trabalho leve e outra pesada respectivamente. É possível observar a redução de utilização no processador com o aumento da frequência do mesmo, mostrando como estas duas variáveis se relacionam.

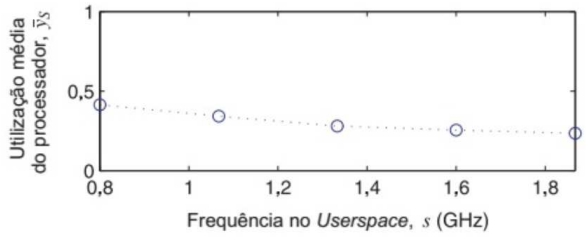

Figura 5.4: Utilização em função da frequência para carga de trabalho leve.
Fonte: Adaptado de Luiz[2].

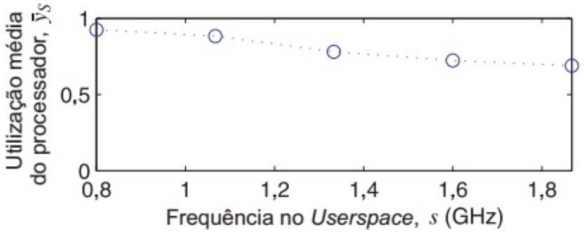

Figura 5.5: Utilização em função da frequência para carga de trabalho pesada.
Fonte: Adaptado de Luiz[2].

Os dados aquisitados do processador foram as frequências escolhidas, a utilização do processador y e a ociosidade ι. A escolha do uso da utilização e frequência devido à relação existente entre estas grandezas como apresentado por Luiz [2]. A frequência normalizada é apresentada em (5.4), onde s_0 é a frequência máxima do processador e $s(t)$ é a frequência atual do processador. A divisão ocorre devido a utilização ser uma função monotonicamente crescente de $\frac{s_0}{s}$ como apresentado em (3.4).

$$u(t) = \frac{s_0}{s(t)} \tag{5.4}$$

Diferentes valores de $y^{\mathbf{on}}$ e $y^{\mathbf{off}}$ precisam ser gerados para cada tipo de carga de trabalho em (3.4). A estimação destes parâmetros em tempo de execução requer o uso de técnicas de estimação tais como mínimos quadrados recursivos, o que pode causar uma sobrecarga

no gerenciador de energia. Isto pode ser evitado pelo uso de $\iota(t-1)$, que fornece uma indicação indireta do tipo de carga de trabalho em execução, e assim, quando adicionada ao modelo, o torna mais genérico.

Foram selecionadas cargas de trabalho para gerar demanda no processador, sendo estas definidas de acordo com a utilização gerada. Extrema[1] para utilizações entre 55% e 95%, pesada[2] para utilizações entre 35% e 85%, média[2] para utilizações entre 15% e 55%, leve[2] para utilizações entre 10% e 35% e mínima[3] para utilizações entre 3% e 14%. Tais cargas são apresentadas na Tabela 5.1.

Tabela 5.1: Descrição das cargas de trabalho usadas.

Carga	Descrição
Extrema	Vídeo com resolução 4K (3840x2160) 60 fps
Pesada	Vídeo com resolução 4K (3840x2160) 30 fps
Média	Vídeo com resolução 1080p (1920x1080) 60 fps
Leve	Vídeo com resolução 1080p (1920x1080) 30 fps
Mínima	Música em formato mp3 com taxa de 320 kbps

Os vídeos são codificados em H.264 por meio da biblioteca x264 [4]. O H.264 é um método e formato para compressão de vídeo, o processo de conversão de vídeo digital em um formato que ocupa menos capacidade quando é armazenado ou transmitido. Compressão ou codificação de vídeo é uma tecnologia essencial para aplicações como televisão digital, TV móvel, videoconferência e transmissão de vídeo pela internet [55].

Para conseguir esta compressão, o codificador H.264 atua removendo redundâncias nos sinais de vídeo, sendo as principais:

1. Redundância espacial: em um *frame* de um vídeo, *pixels* próximos geralmente são correlacionados, dessa forma, o codificador reescreve as informações do *frame* aproveitando informações de *pixels* anteriores, utilizando menos informação para representar cada *pixel*.

2. Redundância temporal: em uma sequência de *frames*, muitas vezes um *pixel* não muda muito sua posição ou não muda. Dessa forma, o codificador baseia-se nos valores dos *pixels* dos *frames* anteriores para utilizar menos informação na escrita dos *pixels* atuais.

[1] http://bbb3d.renderfarming.net/download.html
[2] https://durian.blender.org/download/
[3] http://freemusicarchive.org/
[4] http://www.videolan.org/developers/x264.html

3. Redundância Psico-Visual: a capacidade de visualização do olho humano possui limites para a distinção de detalhes. Dessa forma, o codificador retira informações que não poderiam ser percebidas pelo olho humano, reduzindo a quantidade de informações no vídeo.

Sendo assim, o processamento exigido para os vídeos se torna menor quando existe uma sequências de *frames* parecidos entre si e com imagens de fundo estáticas, resultando em um consumo menor de potência nestas sequências e exige um processamento maior para vídeos com muitas transições de *frames* o que resulta em um maior consumo de potência.

Foi desenvolvido um *software* em C para executar em espaço de usuário do Linux para aplicar frequências desejadas em todos os núcleos do processador com o uso do utilitário *indicator-cpufreq*[5] e realizar medições da utilização y e da ociosidade ι.

Os testes executados para treinar os modelos consistiram em executar as cargas de trabalho descritas na Tabela 5.1 em diferentes frequências, e medir a utilização y e ociosidade ι do processador durante o teste (primeira linha do arquivo /proc/stat). Cada carga de trabalho foi executada por 16 minutos, com a frequência distribuída como apresentado:

1. Para $t \in [0, 60]s$: frequência em 3,2GHz.
2. Para $t \in (60, 120]s$: frequência em 800MHz.
3. Para $t \in (120, 180]s$: frequência em 1GHz.
4. Para $t \in (180, 240]s$: frequência em 1,1GHz.
5. Para $t \in (240, 300]s$: frequência em 1,3GHz.
6. Para $t \in (300, 360]s$: frequência em 1,5GHz.
7. Para $t \in (360, 420]s$: frequência em 1,7GHz.
8. Para $t \in (420, 480]s$: frequência em 1,8GHz.
9. Para $t \in (480, 540]s$: frequência em 2,0GHz.
10. Para $t \in (540, 600]s$: frequência em 2,2GHz.
11. Para $t \in (600, 660]s$: frequência em 2,3GHz.
12. Para $t \in (660, 720]s$: frequência em 2,5GHz.
13. Para $t \in (720, 780]s$: frequência em 2,7GHz.

[5] https://launchpad.net/indicator-cpufreq

14. Para $t \in (780, 840]s$: frequência em 2,9GHz.

15. Para $t \in (840, 900]s$: frequência em 3,0GHz.

16. Para $t \in (900, 960]s$: frequência em 3,2GHz.

Para que o modelo perceba o máximo e o mínimo de utilização que acontece em cada carga de trabalho escolhida, o teste é iniciado com a máxima frequência possível em cada núcleo do processador, obtendo-se a mínima utilização para aquela carga. Após 60 segundos, a frequência é mudada para o mínimo possível em cada núcleo do processador, obtendo-se assim a máxima utilização para aquela carga. Em seguida as frequências são incrementadas a cada 60 segundos para que seja obtida a utilização para cada uma destas. Estas mudanças de frequências causam o efeito inverso na ociosidade do processador, ou seja, durante os primeiros 60 segundos será obtido a ociosidade máxima, usando a máxima frequência e nos próximos 60 segundos a ociosidade mínima usando-se a mínima frequência.

Este teste também se estende para cada uma das cinco cargas de trabalho citadas na Tabela 5.1. Dessa forma, o teste total tem duração de 80 minutos, divididos em:

1. Para $t \in [0, 16]min$: execução da carga extrema.

2. Para $t \in (16, 32]min$: execução da carga pesada.

3. Para $t \in (32, 48]min$: execução da carga média.

4. Para $t \in (48, 64]min$: execução da carga leve.

5. Para $t \in (64, 80]min$: execução da carga mínima.

5.3 Estrutura do modelo

O conjunto de dados usados para o modelo Fuzzy é composto pelo quociente $u(t)$ entre a máxima frequência e frequência atual, a ociosidade atrasada $\iota(t-1)$ e a utilização $y(t)$, como definido em (5.5).

$$Z^N = \{(u(t),\iota(t-1),y(t))\},\ t = 1,2\ldots,N \tag{5.5}$$

onde N é o número de dados aquisitados.

Os dados usados para o treinamento dos modelos são apresentados na Figura 5.6. Estes dados foram obtidos por meio dos testes descritos na Seção 5.2. Como pode ser observado, conforme a frequência aplicada ao processador aumenta, a utilização do mesmo diminui, enquanto que a ociosidade aumenta, evidenciando o comportamento esperado

para este tipo de sistema. Também é possível notar a diferença existente na utilização e ociosidade para as cargas de trabalho atuando na mesma frequência, a utilização mais baixa e ociosidade mais alta encontrada é para a carga mínima, como esperado, em seguida temos este comportamento para a carga leve, seguido das cargas média, pesada e extrema. Com base nestes dados, a estrutura usada para o treinamento dos modelos Fuzzy é apresentada na Figura 5.7.

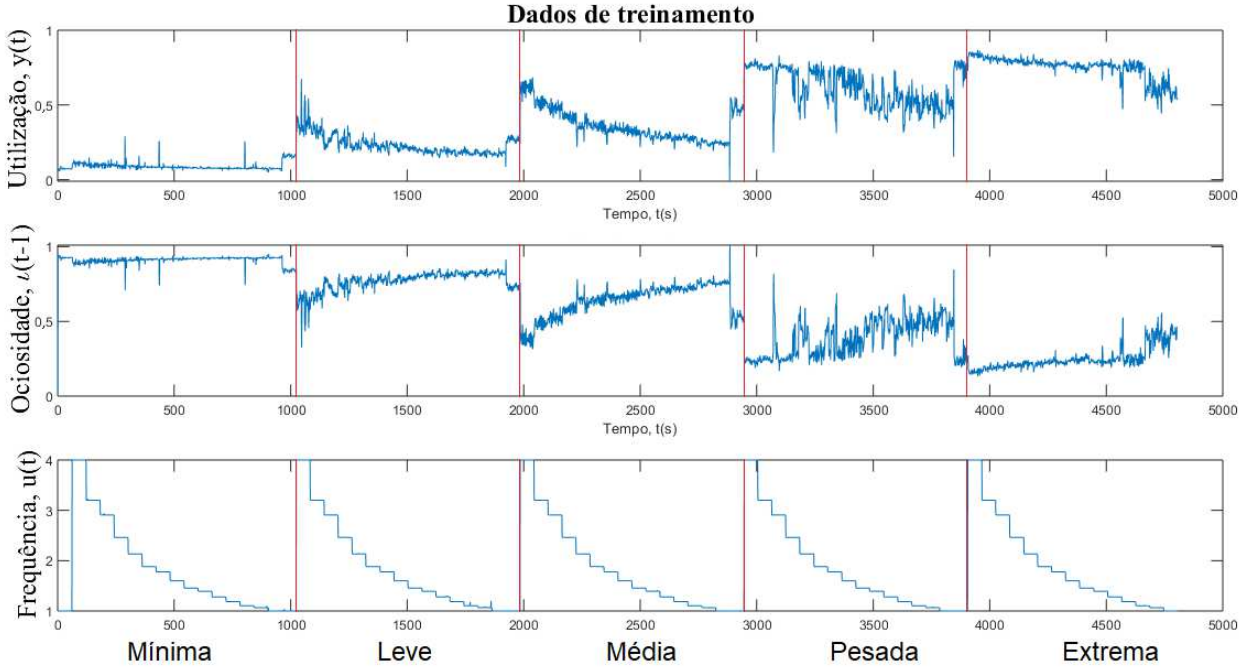

Figura 5.6: Dados utilizados no treinamento do modelos Fuzzy.

Fonte: Autor.

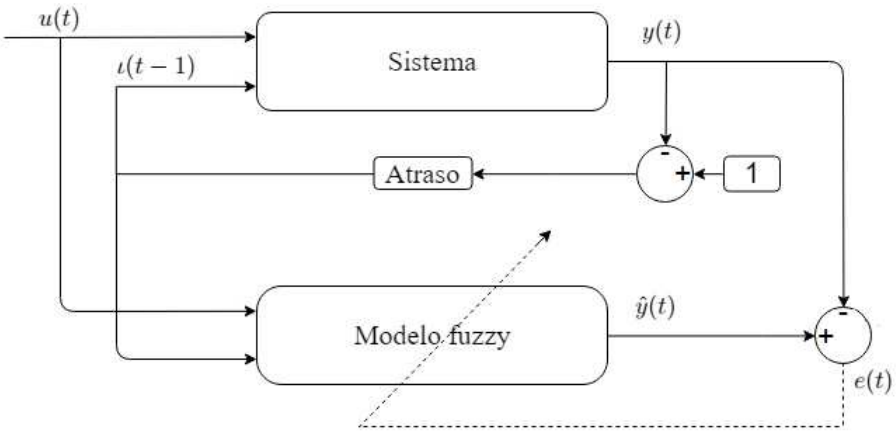

Figura 5.7: Estrutura usada para o treinamento.

Fonte: Autor.

5.4 Estimação de parâmetros

Os principais parâmetros a serem estimados para os modelos Fuzzy são as funções de pertencimento das entradas (quantidade e tipo), como as funções de saída e regras. Para o algoritmo ANFIS apresentado na Seção 3.5.1, podem ser usadas funções de pertencimento do tipo gaussiana ou sino, reapresentadas em (5.6) e (5.7) respectivamente por comodidade.

$$\mu_{ij}(x) = exp\left[-\left(\frac{x - c_j}{2a_i}\right)^2\right] \tag{5.6}$$

$$\mu_{ij}(x) = \frac{1}{1 + \left|\frac{x-c_j}{a_i}\right|^{2b}} \tag{5.7}$$

onde a_i, b_i e c_i são ajustados por meio do método de aprendizagem retropropagação.

Para o algoritmo de clusterização FCM apresentado na Seção 3.6, as funções de pertencimento são do tipo gaussiana e definidas durante a atualização do grau de pertencimento de cada dado ao determinado *cluster*, sendo reapresentada em (5.8) por comodidade.

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c}(\frac{||x_i - c_j||}{||x_i - c_k||})^{\frac{2}{m-1}}} \tag{5.8}$$

Desta forma são usados os próprios *clusters* como centros das funções gaussianas e largura σ é definida como apresentado em (5.9).

$$\sigma = \frac{\sum_{i=1}^{n}\sqrt{\frac{-(x_i - c_j)^2}{2log(\mu_{ij})}}}{n} \tag{5.9}$$

onde x_i é o dado de uma entrada, c_j é o *cluster* usado como centro da função de pertencimento gaussiana e μ_{ij} são os graus de pertencimento do dado x_i ao *cluster* c_j como apresentado em (5.8).

Para o algoritmo de clusterização subtrativa apresentado na Seção 3.7, as funções de pertencimento são do tipo gaussiana e definidas como apresentado em (5.10).

$$\mu_{ij} = e^{-\alpha||x_i - c_j||} \tag{5.10}$$

onde α é uma constante positiva usada para determinar os pesos de cada *cluster*.

Para a determinação do número de funções de pertencimento para cada modelo, foram usadas as próprias métricas dos métodos ANFIS e FCM. Apenas o modelo criado por clusterização subtrativa não teve a quantidade de funções de pertencimento determinadas dessa forma, pois esta quantidade é determinada pelo próprio algoritmo, que adota como *clusters* os dados que possuem maior peso como apresentado em (3.34). Assim foi usado

como critério de avaliação para os modelos ANFIS o erro médio quadrático (RMSE) determinado na criação das funções de pertencimento, apresentado em (3.23) e o número de regras criados por estes modelos. Para os modelos FCM, o critério foi a minimização da função de custo apresentada em (3.30) e o número de regras criado por estes modelos.

É apresentado na Tabela 5.2 o erro médio quadrático e a quantidade de regras geradas para os modelos desenvolvidos por meio de ANFIS e funções de pertencimento do tipo sino. É apresentado na Figura 5.8 a relação entre erro e número de funções para melhor visualização. Os erros diminuem a cada nova função de pertencimento, porém isto aumenta muito o número de regras do modelo, já que estas aumentam com o quadrado da quantidade de funções, o tornando bem mais complexo. Porém é possível observar que, até cinco funções de pertencimento o erro diminui com uma taxa maior que acima de cinco funções. Dessa forma foi adotada esta quantidade de funções, já que com mais funções, a complexidade do modelo em relação ao erro o torna inviável.

Tabela 5.2: Número de funções de pertencimento, erro médio quadrático e número de regras para o modelo ANFIS usando funções do tipo sino.

Modelo ANFIS - Função tipo sino		
Número de funções	RMSE	Número de regras
2	0,037940	4
3	0,037475	9
4	0,036646	16
5	0,035985	25
6	0,035863	36
7	0,035810	49
8	0,035752	64
9	0,035531	81
10	0,035321	100
11	0,035042	121
12	0,034874	144
13	0,034623	169
14	0,034265	196
15	0,033985	225

Os erros médios quadráticos e a quantidade de regras geradas para os modelos desenvolvidos com ANFIS e funções de pertencimento do tipo gaussianas são apresentados na Tabela 5.3. É apresentado na Figura 5.9 a relação entre erro e número de funções para melhor visualização.

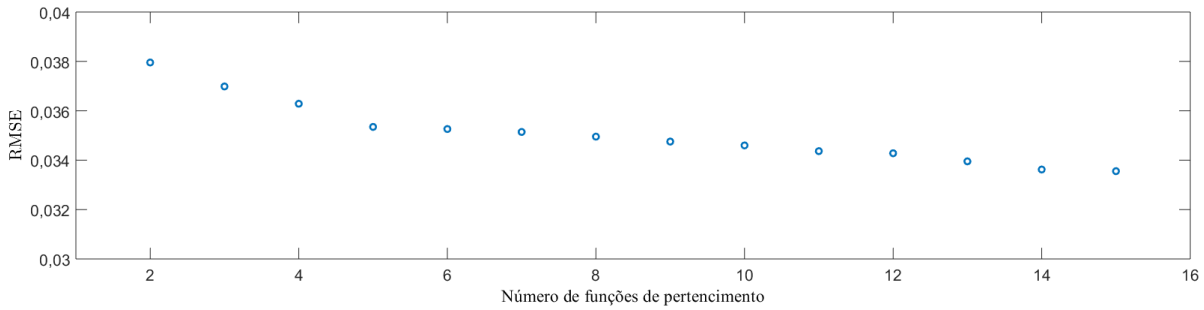

Figura 5.8: RMSE e quantidade de funções para modelos ANFIS com funções tipo sino. Fonte: Autor.

Tabela 5.3: Número de funções de pertencimento, erro médio quadrático e número de regras para o modelo ANFIS usando funções do tipo gaussiana.

Modelo ANFIS - Função tipo gaussiana		
Número de funções	RMSE	Número de regras
2	0,037956	4
3	0,036985	9
4	0,036286	16
5	0,035347	25
6	0,035262	36
7	0,035143	49
8	0,034954	64
9	0,034753	81
10	0,034598	100
11	0,034366	121
12	0,034281	144
13	0,033953	169
14	0,033622	196
15	0,033556	225

O mesmo fenômeno se repete para os modelos ANFIS com funções do tipo gaussiana, desta forma, também foram adotadas cinco funções de pertencimento.

É apresentado na Tabela 5.4 a função de custo J_{FCM} e a quantidade de regras geradas para os modelos desenvolvidos com FCM do tipo Sugeno. É apresentada na Figura 5.10 a relação entre erro e número de funções e em detalhe para melhor visualização.

A função de custo J_{FCM} para os modelos tipo Sugeno é reduzida a cada nova função de pertencimento. É possível observar que, até seis funções de pertencimento, a função é reduzida com uma taxa maior que acima de seis funções. Dessa forma, foi escolhido esta

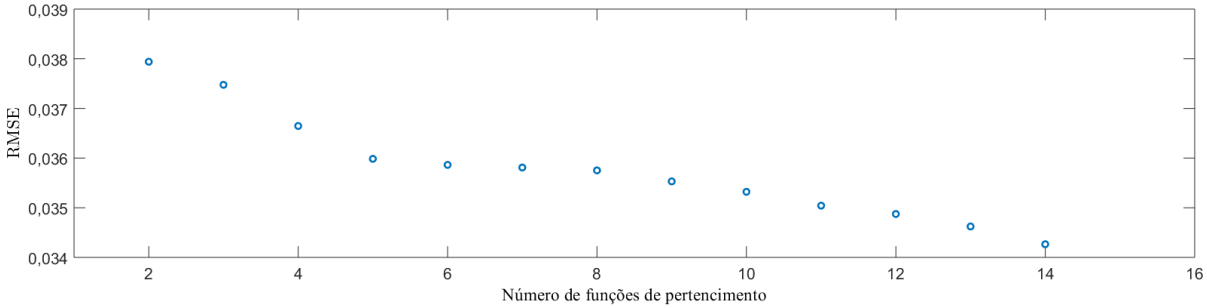

Figura 5.9: RMSE e quantidade de funções para modelos ANFIS com funções tipo gaussiana.

Fonte: Autor.

Tabela 5.4: Número de funções de pertencimento, função de custo e número de regras para o modelo FCM do tipo Sugeno.

Modelo FCM - Tipo Sugeno		
Número de funções	J_{FCM}	Número de regras
2	28,294558	2
3	1,5941530	3
4	0,211992	4
5	0,044504	5
6	0,012511	6
7	0,004251	7
8	0,001679	8
9	0,000765	9
10	0,000413	10
11	0,000211	11
12	0,000116	12
13	0,000066	13
14	0,000039	14
15	0,000024	15

quantidade de funções de pertencimento para este tipo de modelo.

É apresentado na Tabela 5.5 a função de custo e a quantidade de regras geradas para os modelos desenvolvidos com FCM do tipo Mamdani. É apresentado na Figura 5.11 a relação entre erro e número de funções e em detalhe para melhor visualização.

A função de custo J_{FCM} para os modelos tipo Mamdani é reduzida a cada nova função de pertencimento. Novamente até seis funções de pertencimento a função é reduzida com uma taxa maior que acima de seis funções, como exposto no tipo Sugeno. A quantidade

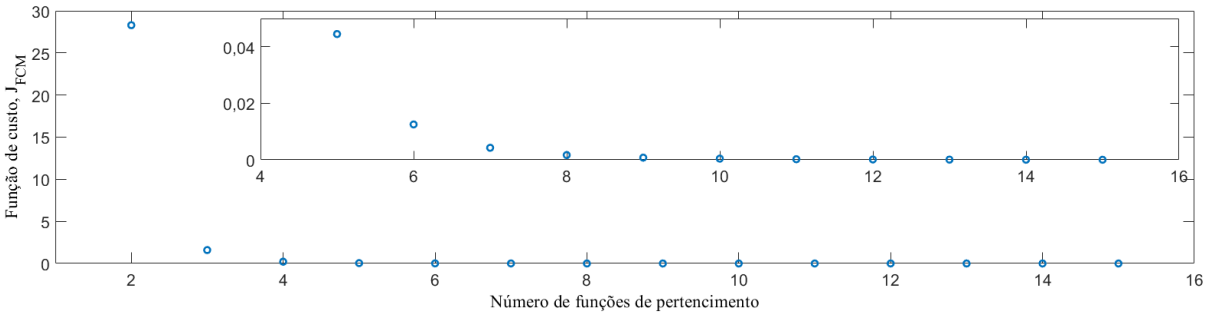

Figura 5.10: Função de custo e quantidade de funções para modelos FCM do tipo Sugeno. Fonte: Autor.

Tabela 5.5: Número de funções de pertencimento, função de custo e número de regras para o modelo FCM do tipo Mamdani.

Modelo FCM - Tipo Mamdani		
Número de funções	J_{FCM}	Número de regras
2	28,294555	2
3	1,594152	3
4	0,212298	4
5	0,044458	5
6	0,012312	6
7	0,004237	7
8	0,001987	8
9	0,000861	9
10	0,000410	10
11	0,000211	11
12	0,000114	12
13	0,000066	13
14	0,000039	14
15	0,000024	15

de funções escolhidas foi seis.

O modelo desenvolvido com clusterização subtrativa possui cinco funções de pertencimento para cada entrada, com seus pesos P_i apresentados na Tabela 5.6.

5.5 Modelos desenvolvidos

As funções de pertencimento desenvolvidas são apresentadas nas Figuras 5.12, 5.13, 5.14, 5.15 e 5.16 para os modelos ANFIS com funções tipo sino, ANFIS com funções tipo

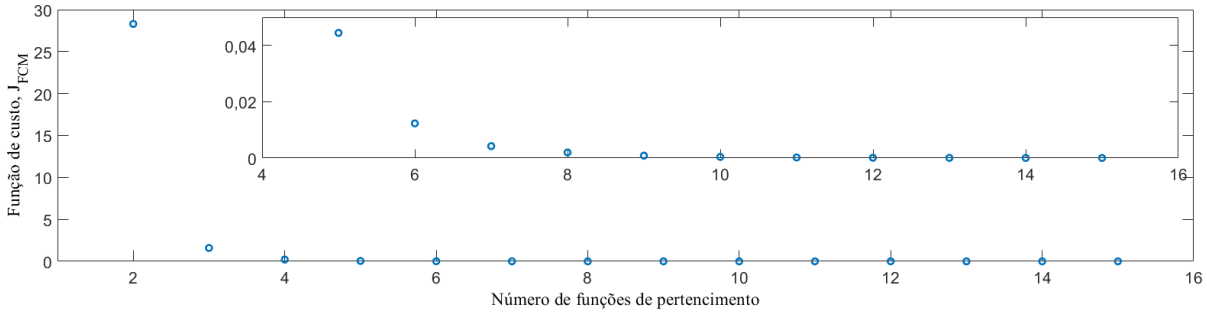

Figura 5.11: Função de custo e quantidade de funções para modelos FCM do tipo Mamdani.

Fonte: Autor.

Tabela 5.6: Pesos de cada *cluster* para o modelo com clusterização subtrativa.

Modelo com clusterização subtrativa	
Cluster	Peso P_i
1	1
2	0,556808
3	0,221431
4	0,218826
5	0,191289

gaussiana, FCM Sugeno, FCM Mamdani e clusterização subtrativa respectivamente. É possível notar que algumas funções de pertencimento nos modelos FCM estão sobrepostas em ambas as entradas: na Figura 5.14 as funções u_4 e u_6, ι_1 e ι_5 para o modelo FCM Sugeno e na Figura 5.15 as funções u_3 e u_4, ι_5 e ι_6 para o modelo FCM tipo Mamdani. Dessa forma, optou-se por reduzir o número de funções destes modelos para cinco, resultando nas funções de pertencimento apresentadas nas Figuras 5.17 e 5.18 para os modelos FCM Sugeno e Mamdani respectivamente, resultando em funções de pertencimento sem sobreposições.

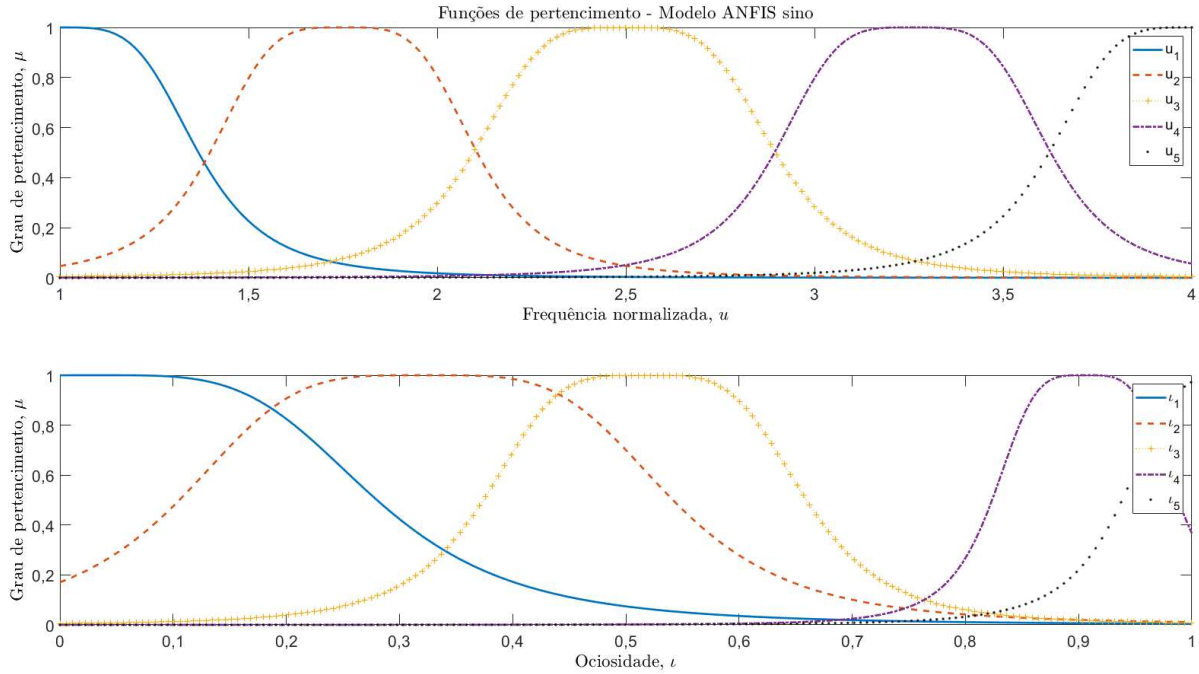

Figura 5.12: Funções de pertencimento para o modelo ANFIS com funções de pertencimento tipo sino.

Fonte: Autor.

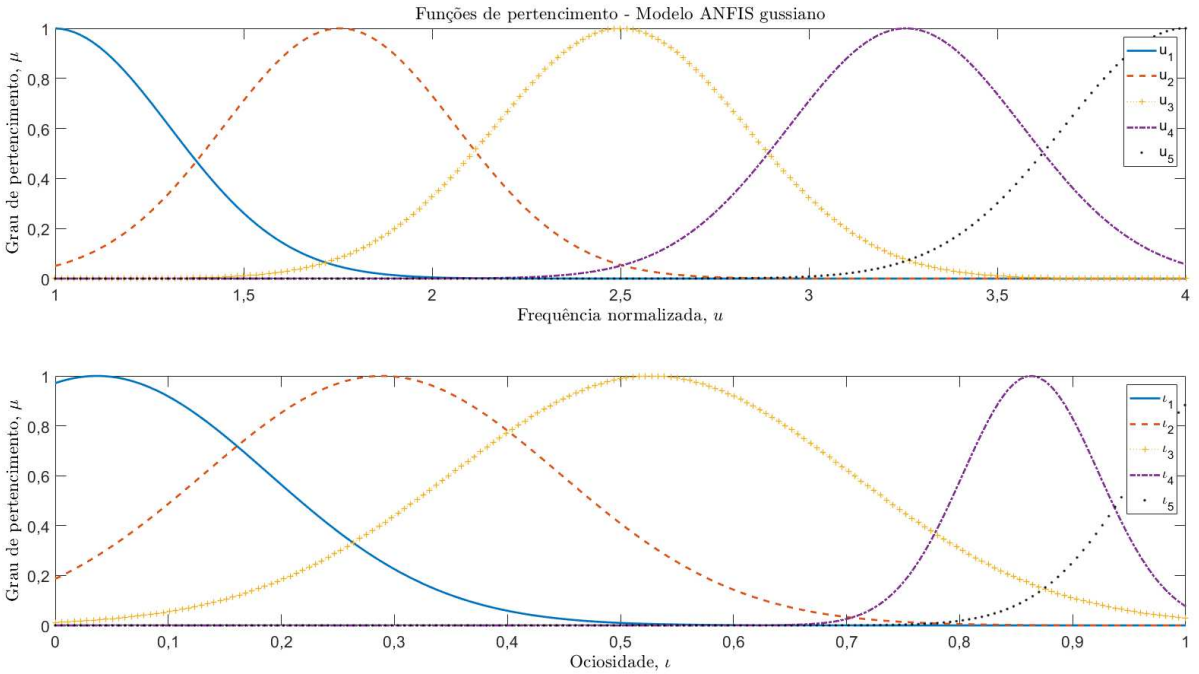

Figura 5.13: Funções de pertencimento para o modelo ANFIS com funções de pertencimento tipo gaussiana.

Fonte: Autor.

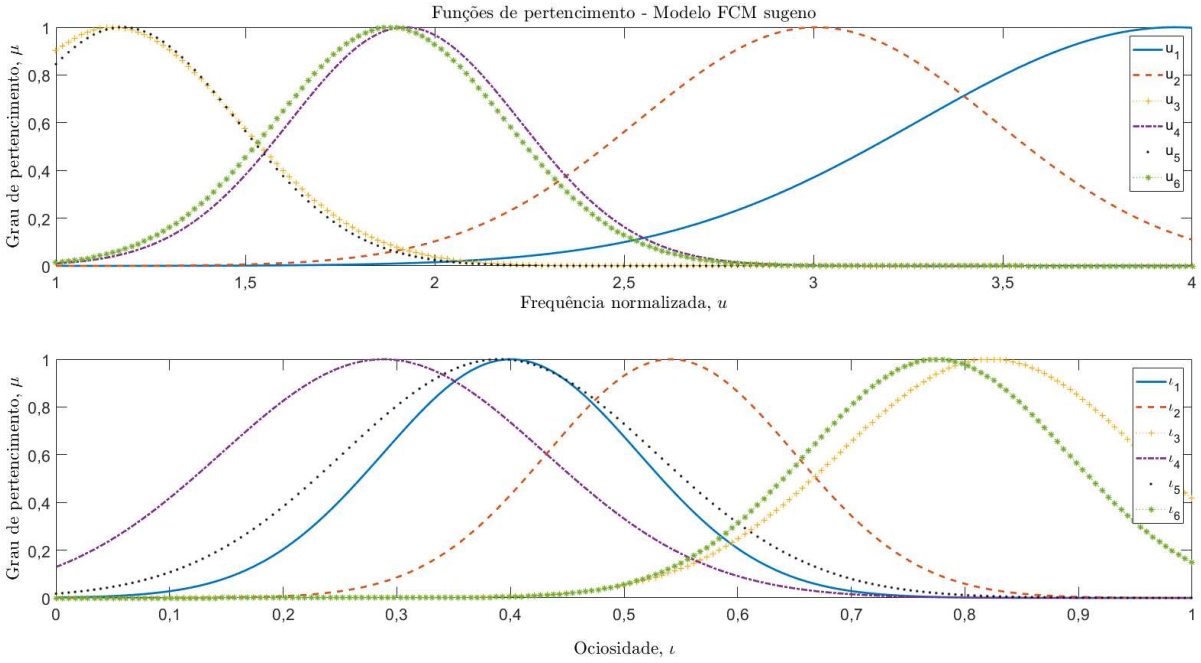

Figura 5.14: Funções de pertencimento para o modelo FCM tipo Sugeno.
Fonte: Autor.

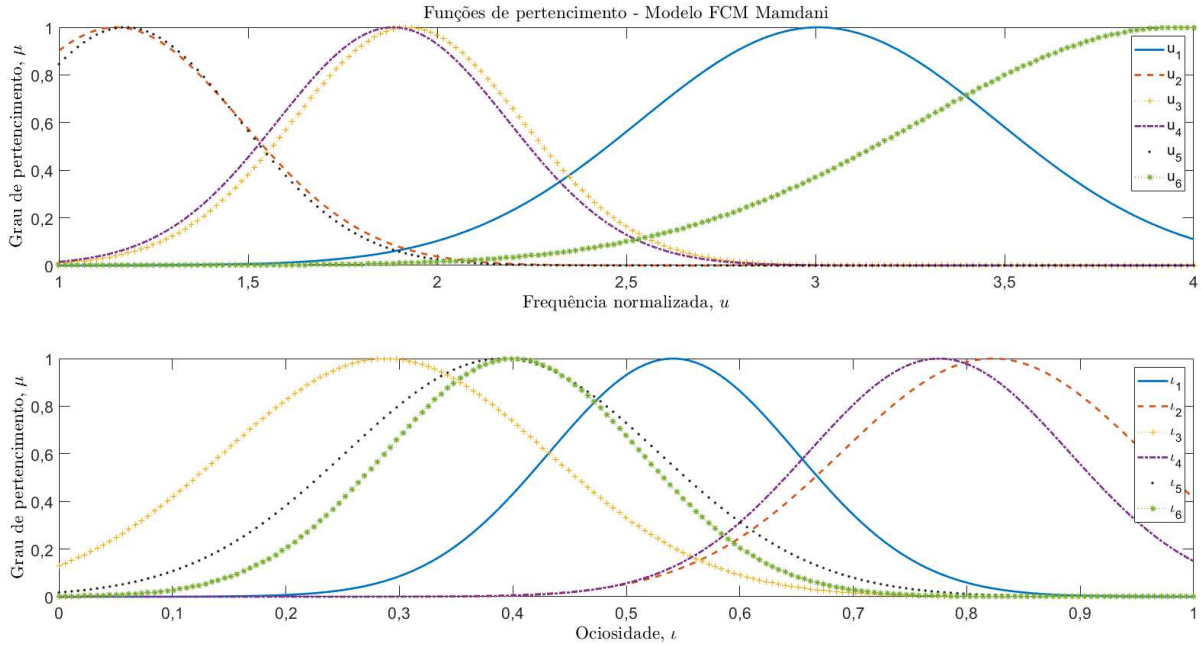

Figura 5.15: Funções de pertencimento para o modelo FCM tipo Mamdani.
Fonte: Autor.

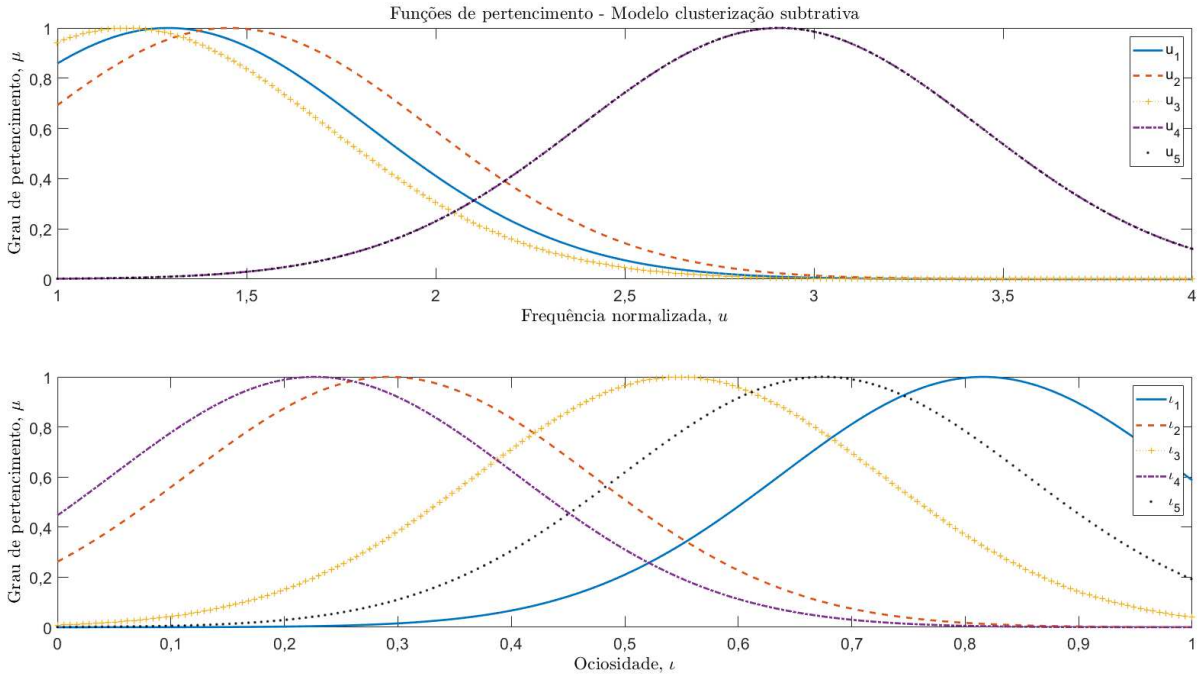

Figura 5.16: Funções de pertencimento para o modelo com clusterização subtrativa.
Fonte: Autor.

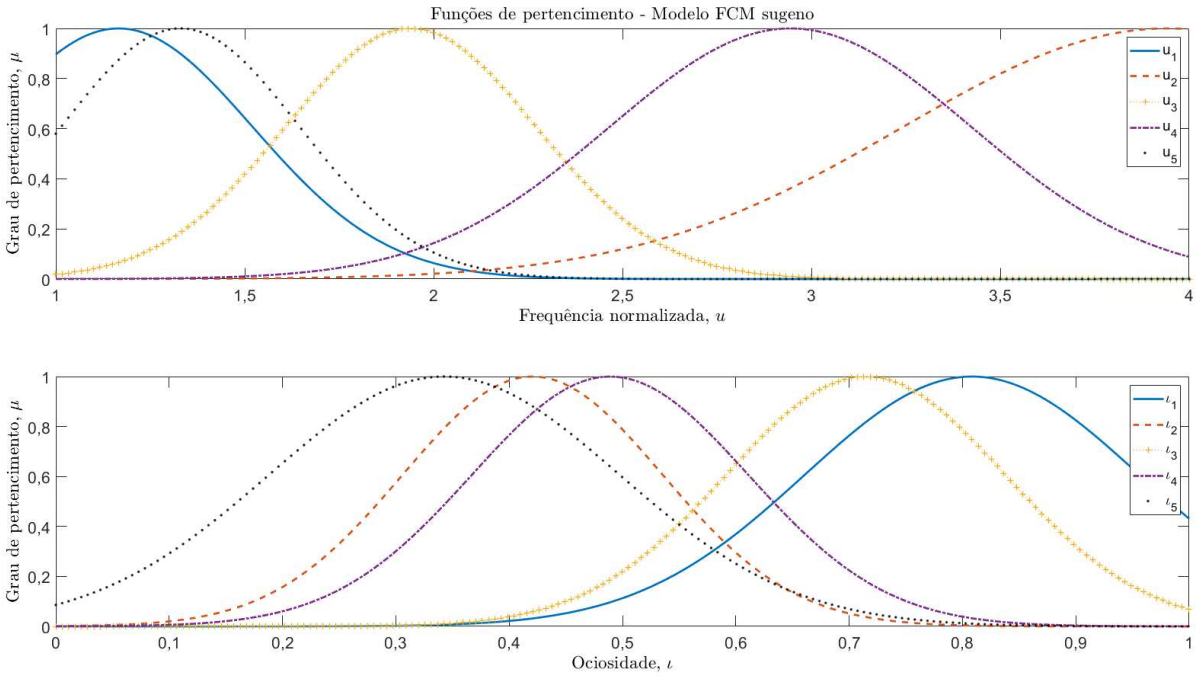

Figura 5.17: Funções de pertencimento para o modelo FCM tipo Sugeno.
Fonte: Autor.

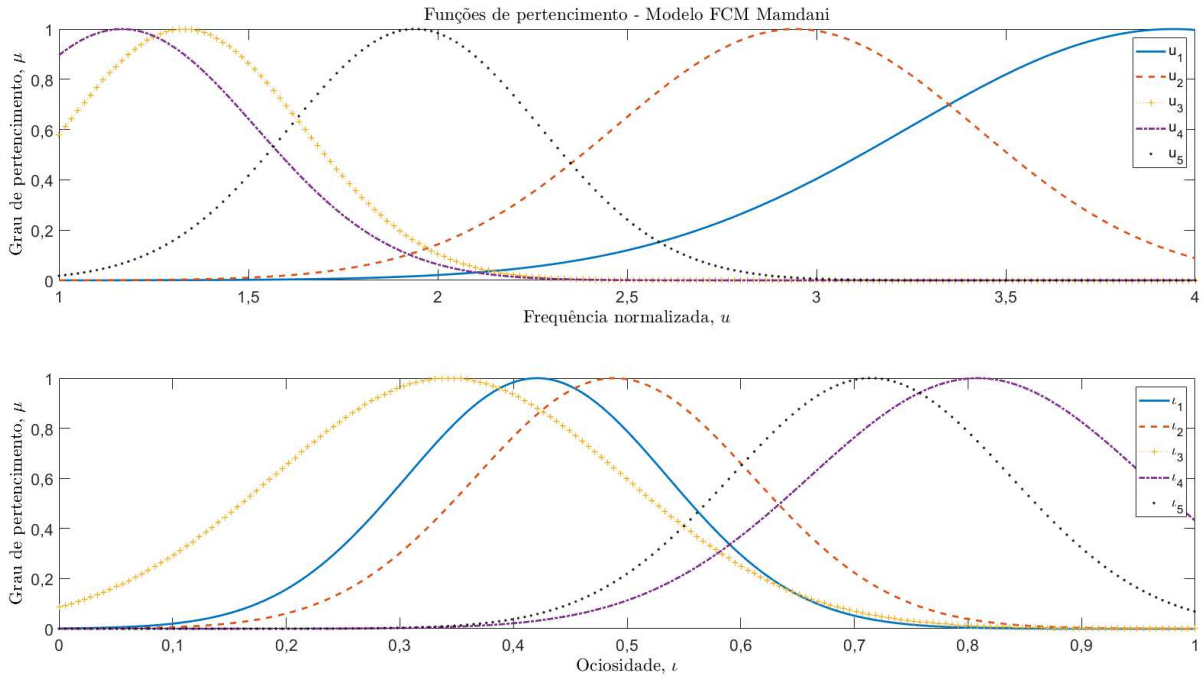

Figura 5.18: Funções de pertencimento para o modelo FCM tipo Mamdani.
Fonte: Autor.

As saídas para os modelos ANFIS, FCM tipo Sugeno e clusterização subtrativa são descritas por (5.11). Apenas o modelo FCM tipo Mamdani possui saída diferente, sendo esta constituída por outras funções de pertencimento.

$$z_i = au(t) + b\iota(t-1) + c \tag{5.11}$$

Os parâmetros a, b e c são apresentados nas Tabelas 5.7, 5.8, 5.9, 5.10 para os modelos ANFIS com funções tipo sino, ANFIS com funções tipo gaussianas, FCM Sugeno e clusterização subtrativa respectivamente. A saída do modelo FCM Mamdani é apresentada na Figura 5.19.

Tabela 5.7: Parâmetros das saídas para o modelo ANFIS com funções tipo sino,

Saídas	a	b	c	Saídas	a	b	c
z_1	0,3428	8,856	-0,332	z_{14}	0,1953	-2,311	1,549
z_2	-0,2348	-0,09932	0,03207	z_{15}	-0,4863	-7,52	8,84
z_3	0,1425	-2,955	2,081	z_{16}	-0,04906	4,865	1,148
z_4	-0,02616	-0,4443	0,6181	z_{17}	-0,03024	1,609	-0,5499
z_5	0,07289	7,739	-7,52	z_{18}	0,1574	-2,198	1,268
z_6	0,04403	-4,491	1,573	z_{19}	0,1103	-1,105	0,8142
z_7	-0,03322	0,76	0,7767	z_{20}	0,06056	18,87	-17,53
z_8	0,07524	-0,1801	0,186	z_{21}	-0,2405	3,964	1,65
z_9	0,0298	-1,624	1,649	z_{22}	-1,307	-1,384	5,648
z_{10}	-0,0157	13,76	-12,93	z_{23}	4,427	-3,555	-15,07
z_{11}	0,04155	-0,1024	0,4635	z_{24}	-0,5756	-8,371	9,262
z_{12}	-0,04887	-2,79	1,672	z_{25}	6,476	-29,69	3,98
z_{13}	0,04445	-1,318	1,261				

Tabela 5.8: Parâmetros das saídas para o modelo ANFIS com funções tipo gaussiana,

Saídas	a	b	c	Saídas	a	b	c
z_1	0,333	6,129	-0,4063	z_{14}	0,07502	-2,077	1,781
z_2	-0,1188	-0,822	0,7365	z_{15}	0,06371	1,266	-1,094
z_3	0,04131	-1,421	1,275	z_{16}	-0,007371	5,763	0,4547
z_4	-0,08795	-0,5492	0,7659	z_{17}	-0,182	0,2527	0,7503
z_5	0,1716	4,617	-4,495	z_{18}	0,1198	-1,254	0,827
z_6	0,1199	11,66	0,2196	z_{19}	-0,04707	-2,796	2,524
z_7	-0,04454	6,713	-1,548	z_{20}	1,267	-6,319	2,764
z_8	0,05642	1,081	-0,68	z_{21}	0,0005992	10,2	0,5808
z_9	-0,04133	2,207	-1,44	z_{22}	-1,5	4,537	4,821
z_{10}	0,1418	22,05	-21,3	z_{23}	1,362	-0,2375	-5,092
z_{11}	-0,04056	-3,013	0,9327	z_{24}	-2,124	-16,94	22,58
z_{12}	-0,02326	-3,805	1,856	z_{25}	13,67	-64,58	7,635
z_{13}	0,01308	-2,09	1,76				

Tabela 5.9: Parâmetros das saídas para o modelo FCM tipo Sugeno,

Saídas	a	b	c
z_1	0,001534	-0,9873	0,9895
z_2	-0,0496	-0,2158	0,9874
z_3	-0,0793	-0,09578	0,5967
z_4	-0,001453	-1,125	1,117
z_5	0,01126	-1,003	0,9195

Tabela 5.10: Parâmetros das saídas para o modelo com clusterização subtrativa,

Saídas	a	b	c
z_1	0,0364	-0,7071	0,6477
z_2	-0,03621	-0,1937	0,922
z_3	-0,08327	-0,09145	0,5501
z_4	-0,001251	-1,099	1,023
z_5	0,01045	-1,024	0,9915

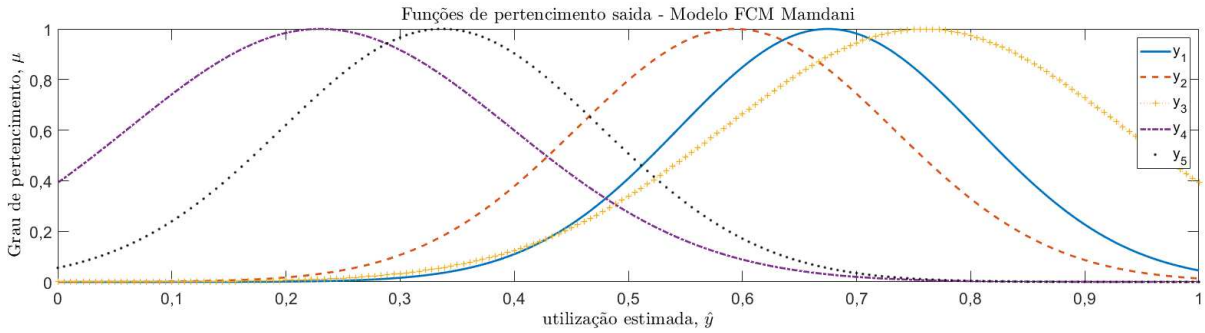

Figura 5.19: Funções de pertencimento da saída para o modelo FCM tipo Mamdani. Fonte: Autor.

As regras os modelos ANFIS com funções tipo sino e gaussiana são apresentadas na Tabela 5.11 e para os modelos FCM tipo Sugeno e Mamdani e clusterização subtrativa são apresentadas na Tabela 5.12.

Tabela 5.11: Regras para os modelos ANFIS com funções tipo sino e gaussiana.

Regras	Entradas			Saída
	$u(t)$	t-norma	$\iota(t-1)$	$\hat{y}(t)$
R_1	u_1	E	ι_1	y_1
R_2	u_1	E	ι_2	y_2
R_3	u_1	E	ι_3	y_3
R_4	u_1	E	ι_4	y_4
R_5	u_1	E	ι_5	y_5
R_6	u_2	E	ι_1	y_6
R_7	u_2	E	ι_2	y_7
R_8	u_2	E	ι_3	y_8
R_9	u_2	E	ι_4	y_9
R_{10}	u_2	E	ι_5	y_{10}
R_{11}	u_3	E	ι_1	y_{11}
R_{12}	u_3	E	ι_2	y_{12}
R_{13}	u_3	E	ι_3	y_{13}
R_{14}	u_3	E	ι_4	y_{14}
R_{15}	u_3	E	ι_5	y_{15}
R_{16}	u_4	E	ι_1	y_{16}
R_{17}	u_4	E	ι_2	y_{17}
R_{18}	u_4	E	ι_3	y_{18}
R_{19}	u_4	E	ι_4	y_{19}
R_{20}	u_4	E	ι_5	y_{20}
R_{21}	u_5	E	ι_1	y_{21}
R_{22}	u_5	E	ι_2	y_{22}
R_{23}	u_5	E	ι_3	y_{23}
R_{24}	u_5	E	ι_4	y_{24}
R_{25}	u_5	E	ι_5	y_{25}

Tabela 5.12: Regras para os modelos FCM tipo Sugeno e Mamdani e clusterização subtrativa.

Regras	Entradas			Saída
	$u(t)$	t-norma	$\iota(t-1)$	$\hat{y}(t)$
R_1	u_1	E	ι_1	y_1
R_2	u_2	E	ι_2	y_2
R_3	u_3	E	ι_3	y_3
R_4	u_4	E	ι_4	y_4
R_5	u_5	E	ι_5	y_5

5.6 Validação

Foram usados dois processos para a validação dos modelos: o teste de qui-quadrado, para validar que a distribuição das utilizações estimadas condiz com as utilizações esperadas e o erro médio quadrático, para validar uma boa precisão entre os valores estimados e esperados.

5.6.1 Teste Qui-quadrado

Tal teste é utilizado para verificar se a frequência com que um determinado acontecimento observado em uma amostra se desvia significativamente ou não da frequência com que ele é esperado[56]. Para tal, algumas condições devem ser atendidas para a realização do teste:

1. Os dados precisam ser divididos em grupos;
2. Os grupos são independentes;
3. Os itens de cada grupo são selecionados aleatoriamente;
4. As observações devem ser frequências ou contagens;
5. Cada observação pertence a uma e somente uma categoria.

Dessa forma, seja um experimento onde foram realizadas N medições de uma variável X, sendo X como apresentado em (5.12). Para a verificação se a distribuição destes dados é consistente com a distribuição esperada para o fenômeno, é necessário definir intervalos $a_j \leq x \leq b_j$ e calcular o número esperado de medidas que devem estar dentro de cada intervalo j definido, onde $j = 1, 2, ..., n$ e n é o número de intervalos definidos.

$$X = [x_1, x_2, ..., x_N] \tag{5.12}$$

As discrepâncias entre as proporções observadas e esperadas são apresentada em (5.13).

$$\chi^2_{exp} = \sum_{j=1}^{n} \frac{(O_j - E_j)^2}{E_j} \tag{5.13}$$

onde O_j é a frequência observada para cada grupo e E_j é a frequência esperada para cada grupo.

Devem ser então testadas duas hipóteses:

1. Hipótese nula: As frequências observadas não são diferentes das frequências esperadas. Portanto não existe diferença entre as frequências (contagens) dos grupos;

2. Hipótese alternativa: As frequências observadas são diferentes da frequências esperadas. Portanto existe diferença entre as frequências.

Para testar as hipóteses é necessário o cálculo de (5.13) e de um χ^2_{teo} teórico, para comparação e tomada de decisão. Se $\chi^2_{exp} \geq \chi^2_{teo}$ rejeita-se a hipótese nula e aceita-se a hipótese alternativa. Se $\chi^2_{exp} < \chi^2_{teo}$ aceita-se a hipótese nula e rejeita-se a hipótese alternativa.

Para o χ^2_{teo} é necessário estabelecer um nível de significância α, representando o risco de se rejeitar uma hipótese verdadeira. Este valor geralmente é estabelecido menor ou igual a 5%, ou seja, $0 < \alpha \leq 0{,}05$. E além disso, é necessário determinar um grau de liberdade para os intervalos j definidos.

É apresentada na Figura 5.20 a distribuição χ^2 para diferentes graus de liberdade.

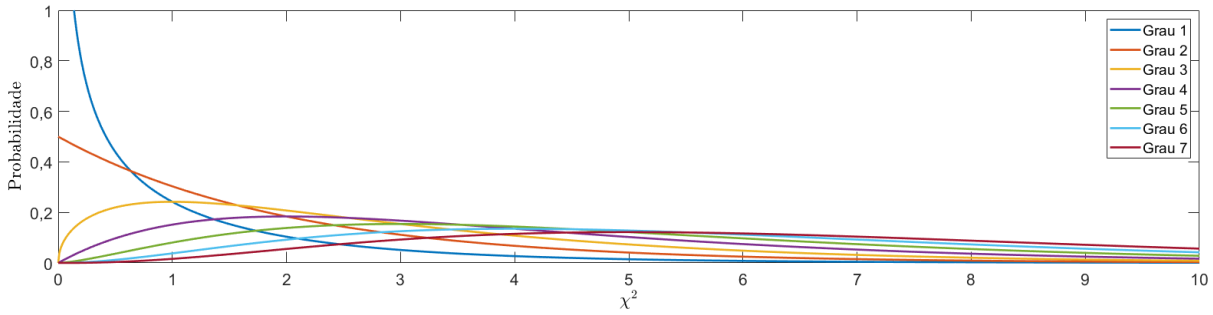

Figura 5.20: Distribuição χ^2.

Fonte: Autor.

Graus de liberdade

Considere um conjunto de dados qualquer. Graus de liberdade é o número de valores deste conjunto de dados que podem variar após terem sido impostas certas restrições a

todos os valores. Desta forma, sendo o número total de células em uma tabela de dados esperados e observados $(l \times c)$ lc. Conhecendo os l totais das linhas, ficam imediatamente conhecidos l dos lc valores das células (um de cada linha). Então o número de células independentes passará a $lc - l$. Se considerarmos os c totais das colunas, percebemos que apenas $c - 1$ deles devem ser considerados. Cada um destes $c - 1$ totais, fixa um valor de célula em cada coluna. Então o número de termos independentes passará a ser $(lc - l) - (c - 1) = (l - 1)(c - 1)$.

Dados esperados e observados

Para validação do modelo, foram feitas medições de frequência normalizada $u(t)$, ociosidade $\iota(t-1)$ e utilizações $y(t)$ para outras cargas de trabalho de mesma natureza, apresentadas na Tabela 5.13. Durante a realização dos testes, as interfaces de rede foram desconectadas, e os processos de atualização e processos não essenciais para operação do sistema operacional foram cancelados.

Tabela 5.13: Cargas de trabalho para validação dos modelos.

Carga de trabalho	Descrição
Carga 1	Vídeo com resolução 1080p (1920x1080) 30 fps
Carga 2	Vídeo com resolução 1080p (1920x1080) 60 fps
Carga 3	Vídeo com resolução 1440p (2560x1440) 30 fps
Carga 4	Vídeo com resolução 4k (3840x2160) 30 fps
Carga 5	Vídeo com resolução 4k (3840x2160) 60 fps

Os dados de validação e estimados são apresentados nas Figuras 5.21, 5.22, 5.23, 5.24, 5.25 para o modelo ANFIS com funções sino, ANFIS com funções gaussianas, FCM tipo Sugeno, FCM tipo Mamdani e com clusterização subtrativa respectivamente.

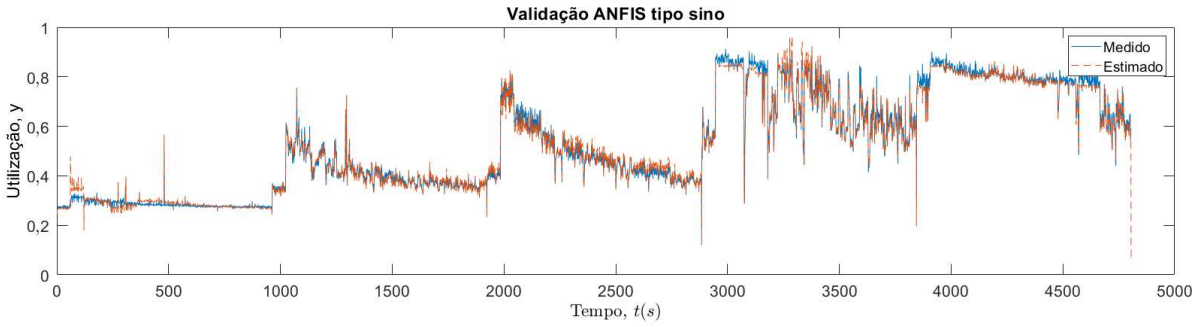

Figura 5.21: Resultado de validação para modelo ANFIS com funções tipo sino.
Fonte: Autor.

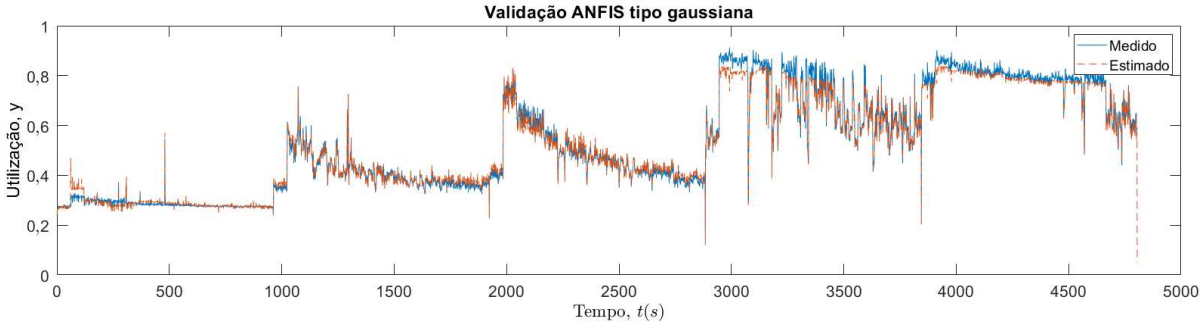

Figura 5.22: Resultado de validação para modelo ANFIS com funções tipo gaussiana.
Fonte: Autor.

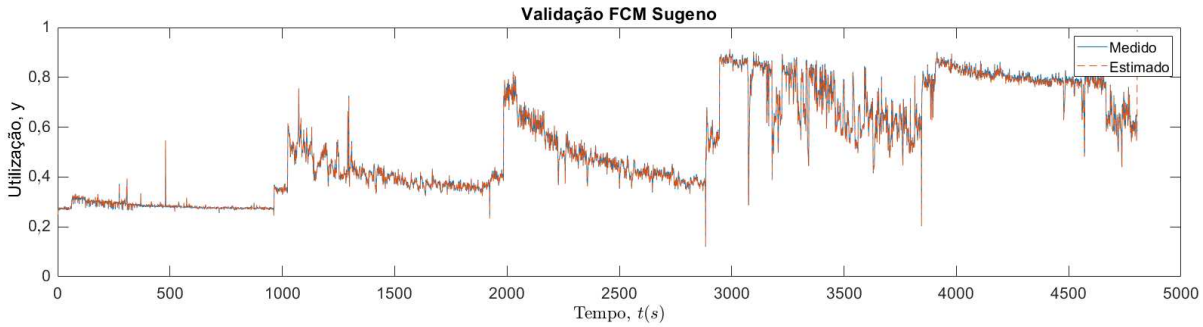

Figura 5.23: Resultado de validação para modelo FCM tipo Sugeno.
Fonte: Autor.

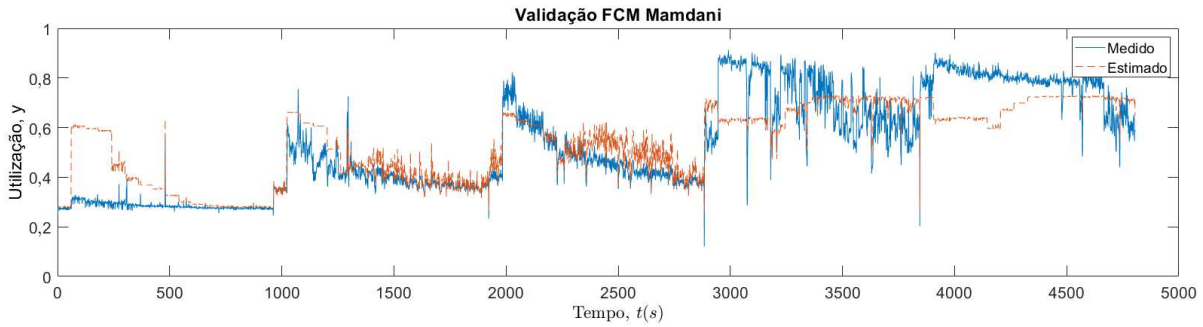

Figura 5.24: Resultado de validação para modelo FCM tipo Mamdani.
Fonte: Autor.

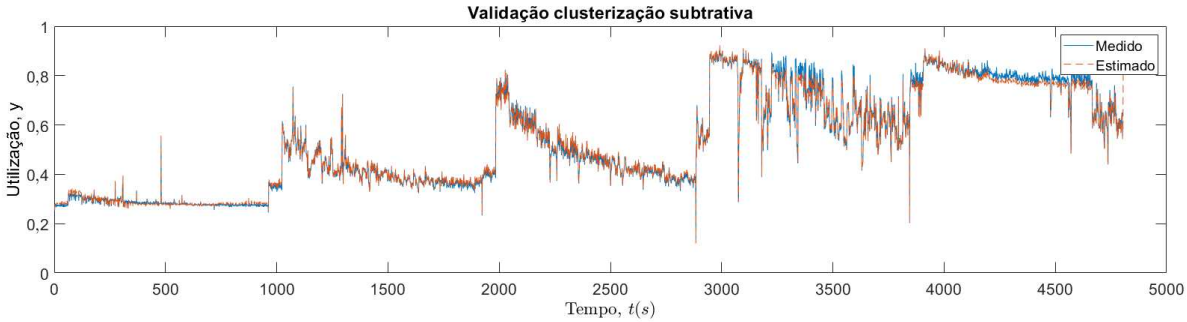

Figura 5.25: Resultado de validação para modelo com clusterização subtrativa. Fonte: Autor.

Os grupos j para as frequências esperadas e observadas adotados são:

1. E_1 e O_1 para y e $\hat{y} \in [0; 0{,}2]$;
2. E_2 e O_2 para y e $\hat{y} \in (0{,}2; 0{,}4]$;
3. E_3 e O_3 para y e $\hat{y} \in (0{,}4; 0{,}6]$;
4. E_4 e O_4 para y e $\hat{y} \in (0{,}6; 0{,}8]$;
5. E_5 e O_5 para y e $\hat{y} \in (0{,}8; 1]$;

As frequências esperadas e observadas são apresentadas nas Tabelas 5.14, 5.15, 5.16, 5.17, 5.18 para os modelos ANFIS com funções tipo sino, ANFIS com funções tipo gaussianas, FCM tipo Sugeno, FCM tipo Mamdani e clusterização subtrativa respectivamente. Usando 5.13 pode-se calcular χ^2_{exp} para cada um dos modelos. O grau de liberdade para cada um dos modelos é $(5-1)(2-1)$, resultando em 4 e adotou-se $\alpha = 0{,}01$ dessa forma existe um risco de 1% de se rejeitar a hipótese verdadeira.

Tabela 5.14: Frequências esperadas e observadas para o modelo ANFIS com funções sino.

j	E_j	O_j
1	1	2
2	1725	1727
3	1290	1303
4	1047	1109
5	742	664

Tabela 5.15: Frequências esperadas e observadas para o modelo ANFIS com funções gaussianas.

j	E_j	O_j
1	1	2
2	1725	1651
3	1290	1439
4	1047	1235
5	742	478

Tabela 5.16: Frequências esperadas e observadas para o modelo FCM do tipo Sugeno.

j	E_j	O_j
1	1	1
2	1725	1728
3	1290	1303
4	1047	1107
5	742	666

Tabela 5.17: Frequências esperadas e observadas para o modelo FCM do tipo Mamdani.

j	E_j	O_j
1	1	0
2	1725	1162
3	1290	1413
4	1047	2230
5	742	0

Tabela 5.18: Frequências esperadas e observadas para o modelo com clusterização subtrativa.

j	E_j	O_j
1	1	1
2	1725	1719
3	1290	1306
4	1047	1112
5	742	667

Temos $\chi^2_{teo} = 13{,}2767$ e para cada modelo temos:

1. Modelo ANFIS com funções sino: $\chi^2_{exp} = 13{,}0042$, logo a hipótese nula é aceita e o modelo é aprovado;

2. Modelo ANFIS com funções gaussianas: $\chi^2_{exp} = 149{,}0719$, logo a hipótese alternativa é aceita e o modelo é reprovado;

3. Modelo FCM tipo Sugeno: $\chi^2_{exp} = 11{,}3590$, logo a hipótese nula é aceita e o modelo é aprovado;

4. Modelo FCM tipo Mamdani: $\chi^2_{exp} = 2275{,}1$, logo a hipótese alternativa é aceita e o modelo é reprovado;

5. Modelo com clusterização subtrativa: $\chi^2_{exp} = 11{,}8355$, logo a hipótese nula é aceita e o modelo é aprovado.

Os modelos aprovados pelo teste qui-quadrado foram os modelos ANFIS com funções sino, FCM tipo Sugeno e clusterização subtrativa. Para a determinação do melhor modelo dentre estes, usou-se o conceito de momentos por serem capaz de caracterizar as distribuições de probabilidade dos sinais estimados.

5.6.2 Momentos

Em estatística, momentos são usados para caracterizar funções de densidade de probabilidade quando não se possuem muitas informações sobre as mesmas. O *k-ésimo* momento da variável aleatória discreta x com função de densidade de probabilidade $f_x(x)$ é definido como apresentado em (5.14)[57].

$$\mu_k = E[x^k] = \sum_x [f_x(x)x^k] \tag{5.14}$$

Outra importante medida são os momentos centrais, definidos como apresentado em (5.15). Estes são usados em preferência a momentos comuns por serem calculados em termos de desvios da média em vez de zero.

$$\mu_k = E[x - \mu_1]^k = \sum_x [f_x(x)(x - \mu_1)]^k \tag{5.15}$$

Os principais momentos para os dados obtidos nos modelos são os de primeira até quarta ordem. O momento de primeira ordem, μ_1 é definido como a média, como apresentado em (5.16).

$$\mu_1 = E[x^1] = \sum_x [f_x(x)x^1] \tag{5.16}$$

O momento central de segunda ordem, μ_2 é definido como a variância da variável aleatória discreta e sua raiz é o desvio padrão, denominado σ. Este momento é apresentado em (5.17).

$$\mu_2 = E[x - \mu_1]^2 = E[x - E(x)^2] = Var(x) \tag{5.17}$$

O momento central de terceira ordem, apresentado em (5.18) é definido como obliquidade e pode ser usado como uma medida de assimetria da função de distribuição de probabilidade. Valores próximo de zero significa simetria, caso contrário, uma tendência à esquerda para números negativos e à direita para números positivos.

$$\mu_3 = E[x - \mu_1]^3 = E[x - E(x)^3] \tag{5.18}$$

O momento central de quarta ordem, apresentado em (5.19) é definido como curtose e pode ser usado como uma medida de achatamento da função de distribuição de probabilidade.

$$\mu_4 = E[x - \mu_1]^4 = E[x - E(x)^4] \tag{5.19}$$

O momento de primeira ordem e os momentos centrais de segunda, terceira e quarta ordem dos dados de validação e os modelos aprovados são apresentados na Tabela 5.19. Os erros relativos entre os momentos gerados pelos modelos e os momentos gerados pelos dados de validação são apresentados na Tabela 5.20.

Para o momento de primeira ordem, o modelo por clusterização subtrativa apresentou um erro relativo de 0,21% em relação ao momento de primeira ordem obtido pelos dados de validação, sendo o menor dentre os modelos selecionados. Para o momento central de segunda ordem, o menor erro relativo obtido foi de 2,04% para o modelo FCM Sugeno. Para o momento central de terceira ordem, o menor erro obtido foi de 2,05% para o modelo FCM Sugeno, estando bem abaixo dos erros de 12,65% para o modelo ANFIS tipo sino e o erro de 14,99% do modelo Clusterização Subtrativa. Para o momento central de quarta ordem, o menor erro relativo obtido foi de 3,7%, novamente para o modelo FCM Sugeno.

Com exceção do momento de primeira ordem, os menores erros obtidos para cada momento foram do modelo FCM Sugeno, dessa forma, este modelo foi escolhido para o desenvolvimento e simulação dos controladores Fuzzy.

Tabela 5.19: Momento de primeira ordem e momentos centrais de segunda, terceira e quarta ordem.

	Momentos			
	1ª Ordem	2ª Ordem	3ª Ordem	4ª Ordem
Dados de validação	0,5309	0,0408	0,0025	0,0027
ANFIS Sino	0,5282	0,0385	0,0022	0,0025
FCM Sugeno	0,5293	0,0400	0,0025	0,0026
Clusterização Subtrativa	0,5298	0,0385	0,0021	0,0025

Tabela 5.20: Erros relativos para o momento de primeira ordem e momentos centrais de segunda, terceira e quarta ordem.

	Erros relativos			
	1ª Ordem	2ª Ordem	3ª Ordem	4ª Ordem
ANFIS Sino	0,0051	0,0557	0,1265	0,0840
FCM Sugeno	0,0031	0,0204	0,0205	0,0370
Clusterização Subtrativa	0,0021	0,0560	0,1499	0,0943

Capítulo 6

Controlador Fuzzy

A ideia básica em controle Fuzzy é modelar as ações a partir de conhecimento especialista, ao invés de, necessariamente, modelar o processo em si. Isso nos leva a uma abordagem diferente dos métodos convencionais de controle de processos, onde os mesmos são desenvolvidos via modelagem matemática dos processos de modo a derivar as ações de controle como função do estado do processo. A motivação para esta nova abordagem veio de casos onde o conhecimento especialista de controle era disponível, seja por meio de operadores ou de projetistas, e os modelos matemáticos envolvidos eram muito custosos, ou muito complicados para serem desenvolvidos.

6.0.1 Visão geral do controlador Fuzzy

Durante o projeto de controladores Fuzzy, é necessária a definição de alguns parâmetros. Estes parâmetros são definidos a partir da experiência do projetista ou através de experimentos. Dado um processo, alguns dos parâmetros são fixos, dentro das condições normais de operação, sendo que outros precisam ser alterados de tempos em tempos. Os parâmetros fixos são chamados de parâmetros estruturais e os variáveis são chamados de parâmetros de sintonização:

- Parâmetros estruturais:
    1. Número de variáveis de saída;
    2. Número de variáveis de entrada;
    3. Recursos de operação sobre os dados de entrada (tais como somas e multiplicações);
    4. Variáveis linguísticas;
    5. Funções de pertinência parametrizadas;
    6. Intervalos de discretização e normalização;

7. Estrutura da base de regras;
8. Conjunto básico de regras.

- Parâmetros de sintonização:

1. Universo de discurso das variáveis;
2. Parâmetros das funções de pertinência, tais como altura, largura ou conjunto suporte;
3. ganhos e offset das entradas e saídas.

A sintonização é uma das etapas mais custosas do projeto de um controlador Fuzzy. A grande flexibilidade que decorre da existência de muitos parâmetros exige um grande esforço do projetista de modo a obter o melhor desempenho do controlador.

A estrutura de um processo controlado por um controlador Fuzzy é mostrada na Figura 6.1, enfatizando-se seus componentes básicos: a interface de fuzzyficação, a base de conhecimento, a base de dados, o procedimento de inferência e a interface de defuzzyficação.

Aqui a base de regras ou conhecimento caracteriza a estratégia de controle e suas metas. A base de dados armazena as definições necessárias sobre discretizações e normalizações dos universos de discurso, as partições fuzzy dos espaços de entrada e saída e as definições das funções de pertinência. Os demais processos como fuzzyficação, inferência e deffuzyficação seguem o que foi determinado no Capítulo 3.

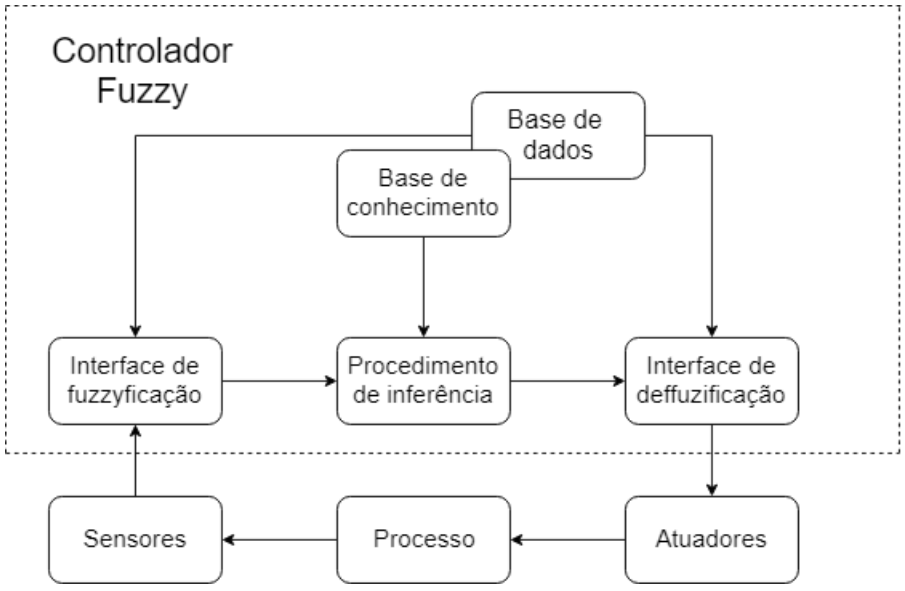

Figura 6.1: Estrutura geral do controlador Fuzzy.
Fonte: Autor.

O controlador deste trabalho é representado pela Figura 6.2. O bloco sensor é responsável pela aquisição da variável controlada $y(k)$, neste trabalho, a utilização do processador. Isto é feito por leituras no arquivo /proc/stat como apresentado em (4.3). O bloco atuador é responsável para mapear o sinal de controle $u(k)$ e modular uma sequência de comandos aplicados em sub-intervalos do período de amostragem $c(k)$, tal como um sinal de modulação por largura de pulso, correspondendo ao sinal de frequência desejado para o processador. Isto é feito por meio do utilitário *indicator-cpufreq*[1].

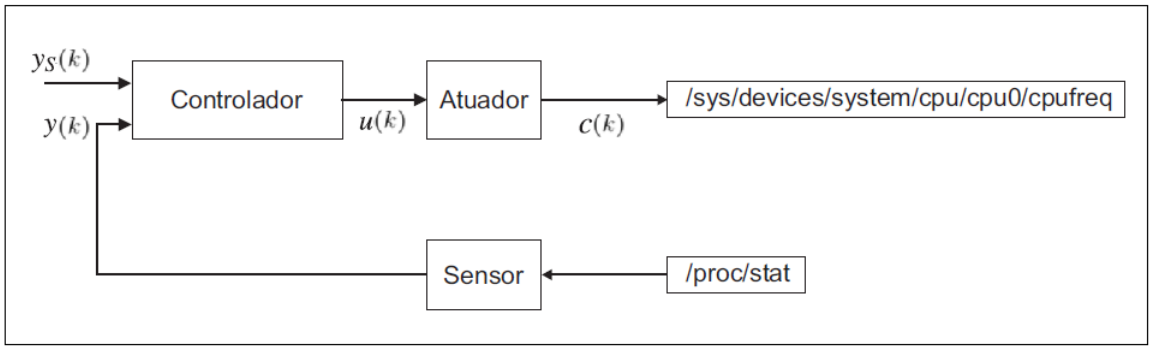

Figura 6.2: Estrutura de controle.
Fonte: Autor.

Foi utilizado um sub-intervalo cinquenta vezes menor que o período de amostragem utilizado para a aplicação da modulação por largura de pulso. Seja o conjunto de frequências normatizadas possíveis para o processador apresentadas em (6.1). São escolhidos os valores imediatamente superior e inferior ao sinal de controle $u(k)$, denominados s_{sup} e s_{inf} respectivamente. Dessa forma, o *duty cicle* $D(k)$ necessário para o modulação por largura de pulso é calculado como apresentado em (6.2).

$$S = \{s_1, s_2, s_3, ..., s_n\} \tag{6.1}$$

$$D(k) = \frac{u(k)}{s_{sup} + s_{inf}} \tag{6.2}$$

6.1 Projeto do controlador Fuzzy

A primeira abordagem realizada para o desenvolvimento dos controladores Fuzzy partiu da adaptação de controladores do tipo PID discretizados com saída de controle (6.3) para controladores Fuzzy, denominados de controladores Fuzzy PID, apresentados por Jantzen [58].

[1] https://launchpad.net/indicator-cpufreq

$$u(k) = u(k-1) + K_p \left[(e(k) - e(k-1)) + \frac{1}{T_i} T_s e(k) + \frac{T_d}{T_s} (e(k) - 2e(k-1) + e(k-2)) \right] \tag{6.3}$$

As entradas do controlador são o erro e a variação do erro. A utilização do processador $y(k)$ é mensurada no $k-$ésimo ponto de amostragem e comparado com a referência de utilização $ys(k)$ para o cálculo do erro $e(k)$, sendo este definido em (6.4). A variação do erro $\Delta e(k)$, é definida em (6.5).

$$e(k) = ys(k) - y(k) \tag{6.4}$$

$$\Delta e(k) = e(k) - e(k-1) \tag{6.5}$$

A saída do controlador é a frequência normatizada, reapresentada em (6.6) por comodidade, onde s_0 é a frequência máxima do processador e $s(k)$ é a frequência no instante k.

$$u(k) = \frac{s_0}{s(k)} \tag{6.6}$$

Os controladores apresentados por Jantzen [58] são apresentados nas Figuras 6.3, 6.4 e 6.5, para o controlador Fuzzy PD, o controlador Fuzzy PD + I e o controlador Fuzzy incremental respectivamente.

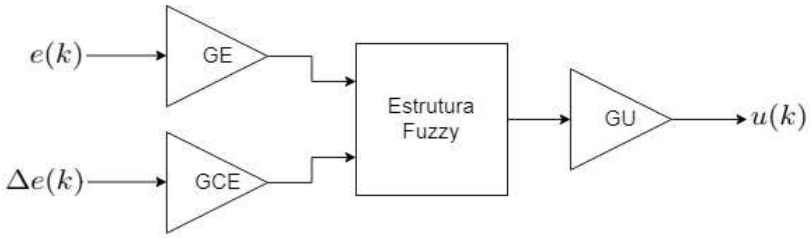

Figura 6.3: Estrutura do controlador Fuzzy PD.

Fonte: Autor.

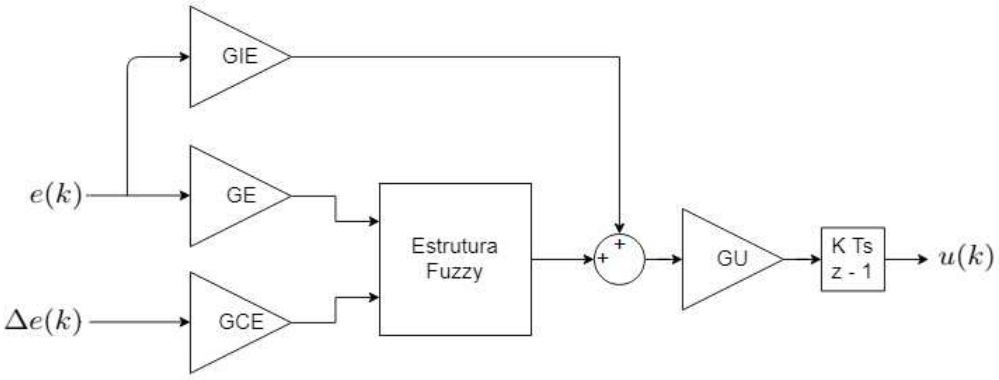

Figura 6.4: Estrutura do controlador Fuzzy PD + I.
Fonte: Autor.

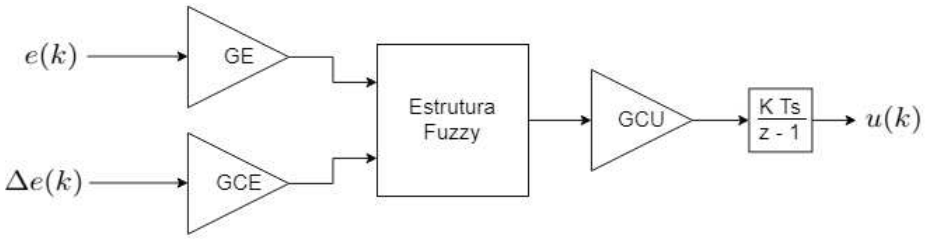

Figura 6.5: Estrutura do controlador Fuzzy incremental.
Fonte: Autor.

As relações entre os ganhos do controlador PID e os ganhos dos controladores Fuzzy abordados são apresentados na Tabela 6.1.

Tabela 6.1: Relações entre ganhos PID e ganhos dos controladores Fuzzy.

Controlador	K_p	$1/T_i$	T_d
Fuzzy PD	$GE.GU$		GCE/GE
Fuzzy PD + I	$GE.GU$	GIE/GE	GCE/GE
Fuzzy incremental	$GCE.GUC$	GE/GCE	

Existem quatro tipos de estruturas Fuzzy para os controladores Fuzzy apresentados nas Figuras 6.3, 6.4 e 6.5, diferenciando-se nas funções de pertinência e conjunto de regras, sendo superfície linear, superfície íngreme, superfície delicadamente inclinada e superfície irregular. A estrutura do tipo superfície linear possui as regras apresentadas na Tabela 6.2 e funções de pertencimento das entradas apresentadas na Figura 6.6 resultando em uma superfície como apresentado em 6.7.

Tabela 6.2: Regras para a estrutura Fuzzy tipo superfície linear.

Regras	Entradas			Saída
	$e(k)$	t-norma	$\Delta e(k)$	$u(k)$
R_1	Neg	E	Neg	-1
R_2	Neg	E	$Zero$	$-0{,}5$
R_3	Neg	E	Pos	0
R_4	$Zero$	E	Neg	$-0{,}5$
R_5	$Zero$	E	$Zero$	0
R_6	$Zero$	E	Pos	0,5
R_7	Pos	E	Neg	0
R_8	Pos	E	$Zero$	0,5
R_9	Pos	E	Pos	1

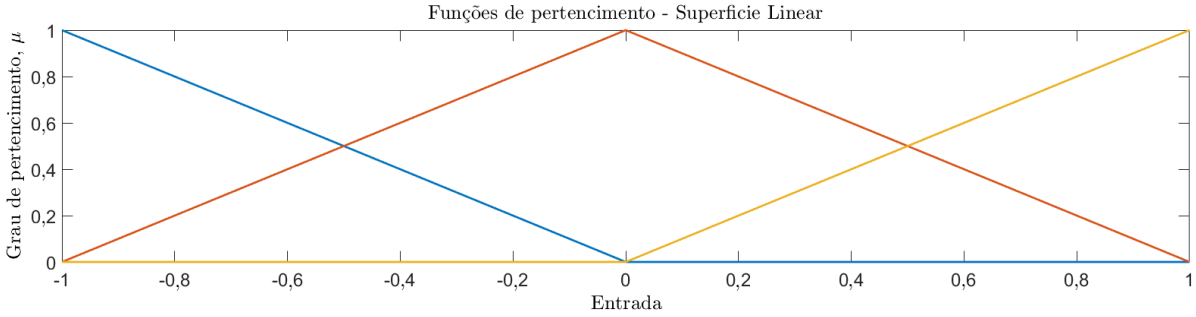

Figura 6.6: Funções de pertencimento das entradas para a estrutura Fuzzy tipo superfície linear.

Fonte: Autor.

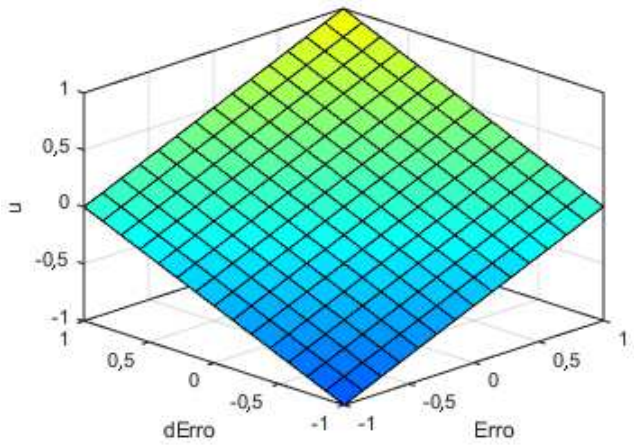

Figura 6.7: Superfície para a estrutura Fuzzy tipo superfície linear.

Fonte: Autor.

A estrutura do tipo superfície íngreme possui as regras R_1, R_3, R_7, e R_9 apresentadas na Tabela 6.2 e funções de pertencimento das entradas apresentadas na Figura 6.8 resultando em uma superfície como apresentado em 6.9.

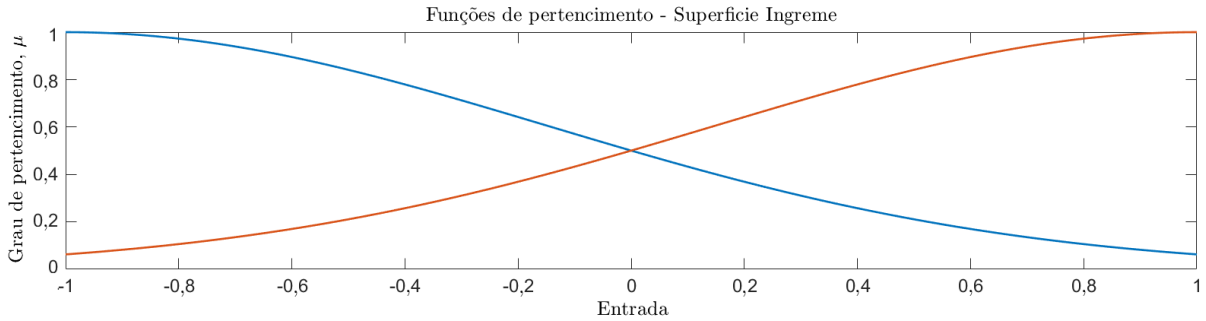

Figura 6.8: Funções de pertencimento das entradas para a estrutura Fuzzy tipo superfície íngreme.

Fonte: Autor.

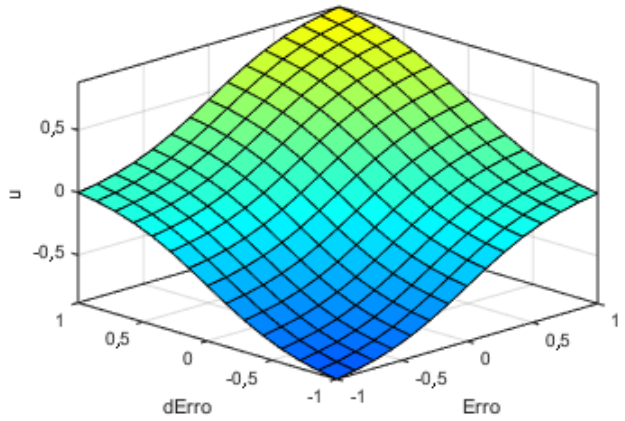

Figura 6.9: Superfície para a estrutura Fuzzy tipo superfície íngreme.
Fonte: Autor.

A estrutura do tipo superfície delicadamente inclinada possui as regras R_1, R_3, R_7, e R_9 apresentadas na Tabela 6.2 e funções de pertencimento das entradas apresentadas na Figura 6.10 resultando em uma superfície como apresentado em 6.11.

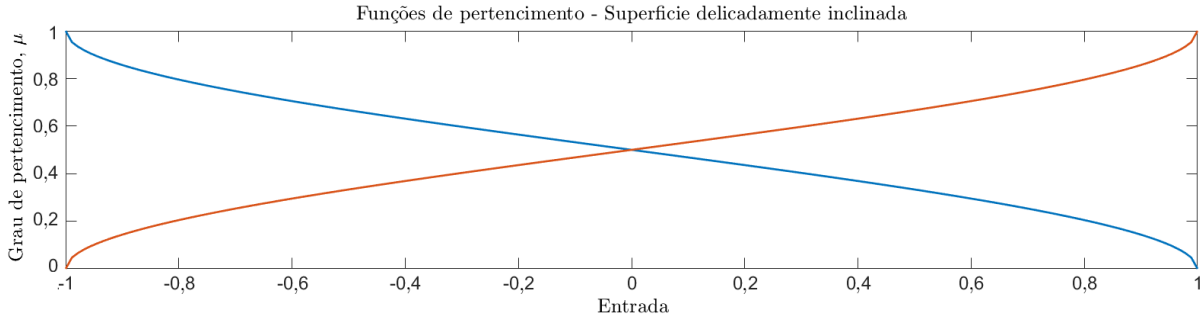

Figura 6.10: Funções de pertencimento das entradas para a estrutura Fuzzy tipo superfície delicadamente inclinada.

Fonte: Autor.

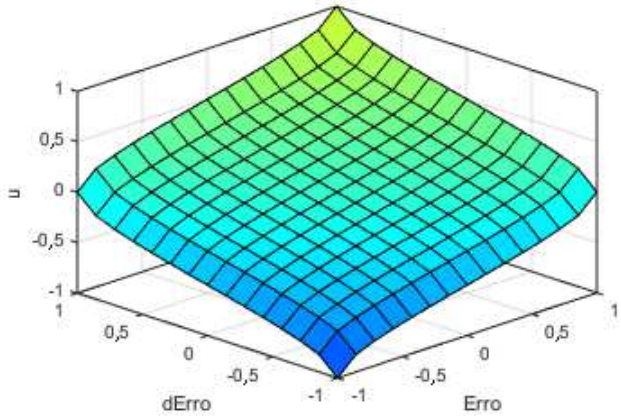

Figura 6.11: Superfície para a estrutura Fuzzy tipo superfície delicadamente inclinada.

Fonte: Autor.

A estrutura do tipo superfície irregular possui as regras apresentadas na Tabela 6.2 e funções de pertencimento das entradas apresentadas na Figura 6.12 resultando em uma superfície como apresentado em 6.13.

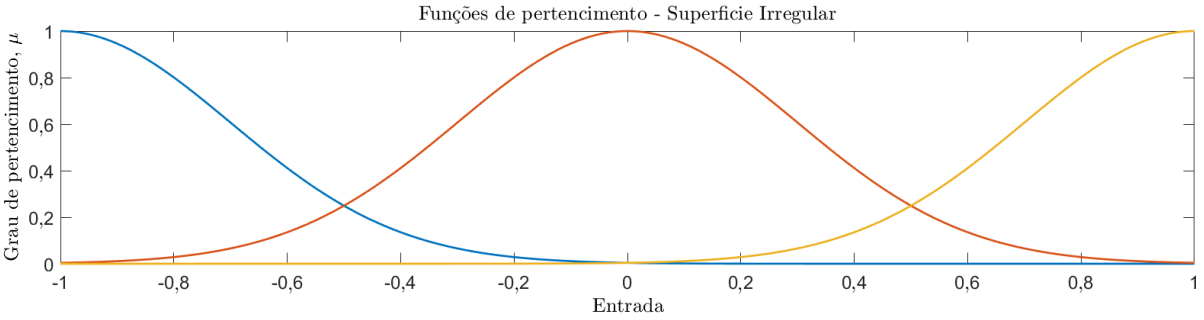

Figura 6.12: Funções de pertencimento das entradas para a estrutura Fuzzy tipo superfície irregular.

Fonte: Autor.

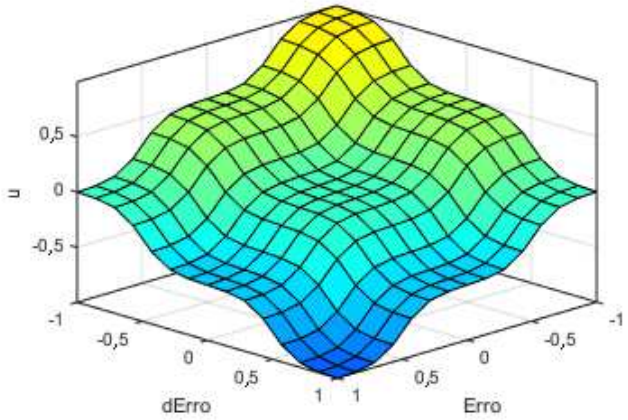

Figura 6.13: Superfície para a estrutura Fuzzy tipo superfície irregular.
Fonte: Autor.

Outra maneira usada no desenvolvimento de regras Fuzzy é apresentado por Wang et al. [59], baseada em dados numéricos. Este método é dividido em cinco passos principais, sendo:

1. Dividir os dados numéricos de entradas e saídas em regiões Fuzzy:

   Seja o conjunto de entradas e saídas apresentadas em (6.7).

   $$(x_1^{(1)}, x_2^{(1)}, y^{(1)}),(x_1^{(2)}, x_2^{(2)}, y^{(2)}), ...,(x_1^{(i)}, x_2^{(i)}, y^{(i)}) \tag{6.7}$$

   Com intervalos $[x_1^-, x_1^+]$, $[x_2^-, x_2^+]$ e $[y^-, y^+]$. E desta forma divide-se cada intervalo em $2N + 1$ regiões, onde N pode ter valores diferentes para cada entrada ou saída diferentes.

2. Gerar as regras Fuzzy a partir destes dados:

   Para isso é necessário determinar os graus de pertencimento das entradas $(x_1^{(i)}, x_2^{(i)})$ e saídas $y^{(i)}$ em diferentes regiões. Considere os conjuntos de dados apresentados na Figura 6.14. Neste caso a entrada $x_1^{(1)}$ possui um grau de 0,8 em $B1$, um grau de 0,2 em $B2$ e um grau zero em todas as outras regiões. De maneira similar, $x_2^{(2)}$ possui um grau de 1 em CE e zero em todas as outras regiões.

   Considerando então cada um dos pares de entrada e saída, pode-se gerar uma regra para cada um destes, sendo:

   $$(x_1^{(1)}, x_2^{(1)}, y^{(1)}) \rightarrow [x_1^{(1)}(0{,}8\ em\ B1, max), x_2^{(1)}(0{,}7\ em\ S1, max); y^{(1)}(0{,}9\ em\ CE, max)] \rightarrow Regra1.$$

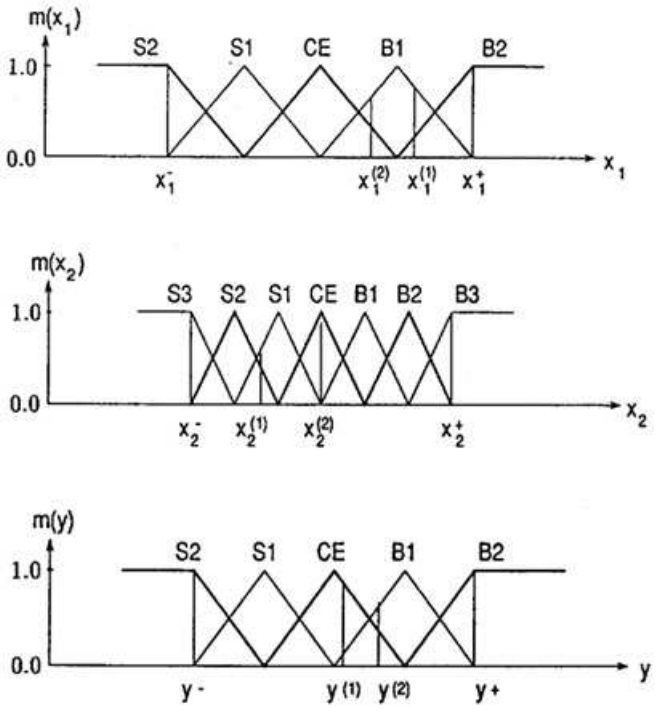

Figura 6.14: Conjunto de entradas e saídas.
Fonte: *Generating Fuzzy Rules by Learning from Examples*, Wang, 1992.

Gerando a regra:

SE x_1 é $B1$ e x_2 é $S1$, ENTÃO y é $B1$.

3. Determinar o grau com que cada uma das regras é geradas, para resolver conflitos entre estas:

   Como usualmente usasse muitos pares de dados, e cada par gera uma regra, é muito provável que existam algumas regras conflitantes, ou seja, regras que possuem a mesma parte SE, porém partes ENTÃO diferentes. Uma maneira de se resolver isso é determinando um grau para cada regra e aceitar a regra do conflito com o grau maior.

   Usa-se a estratégia do produto para determinar o grau de cada regra, por exemplo, para a regra: "SE x_1 é A e x_2 é B, ENTÃO y é C". O grau da regra $D(Regra)$ é definido como apresentado em (6.8).

$$D(Regra) = m_A(x_1)m_B(x_2)m_C(y) \tag{6.8}$$

Sendo assim para a regra 1 definida anteriormente:

$$D(Regra1) = m_{B1}(x_1)m_{CE}(x_2)m_{B1}(y) = (0{,}8)(0{,}7)(0{,}9) = 0{,}504$$

Caso regras também sejam geradas por especialistas humanos, estes também devem atribuir um grau a estas, levando em consideração a quantidade de vezes que a regra em questão ocorre em relação a alguma conflitante.

4. Criar um conjunto de regras Fuzzy, combinando as regras geradas e as regras linguísticas de humanos especialistas em forma de tabela. Os espaços devem ser preenchidos de acordo com o grau de cada regra de dados numéricos ou linguísticos. Caso haja mais de uma regra Fuzzy para cada espaço, escolhe-se a de maior grau.

5. Determinar um mapeamento entre as entradas e saídas, baseado nas combinações de regras Fuzzy usando um processo de defuzzyficação adequado para o tipo de controle e sistema usados.

6.1.1 Controlador Fuzzy pelo método de Wang

Nas Figuras 6.15, 6.16 e 6.17 são apresentadas as funções de pertinência usadas nas duas entradas do controlador Fuzzy para $N = 3$, $N = 2$ e $N = 1$.

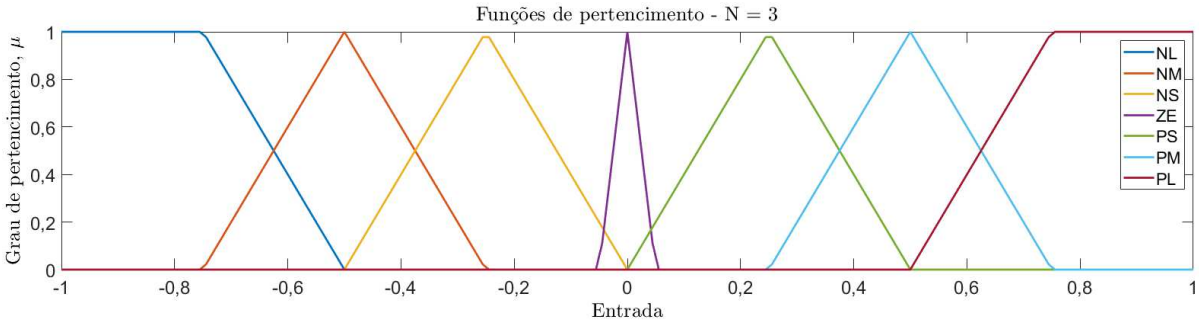

Figura 6.15: Funções de pertencimento para entradas para $N = 3$.
Fonte: Autor.

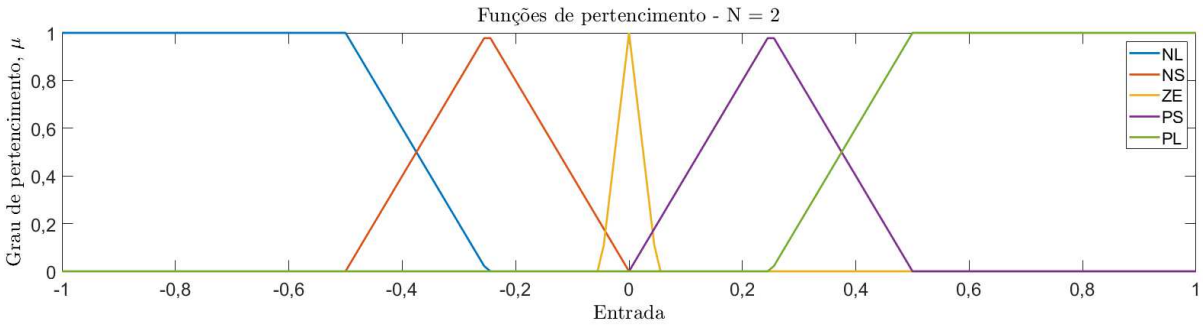

Figura 6.16: Funções de pertencimento para entradas para $N = 2$.
Fonte: Autor.

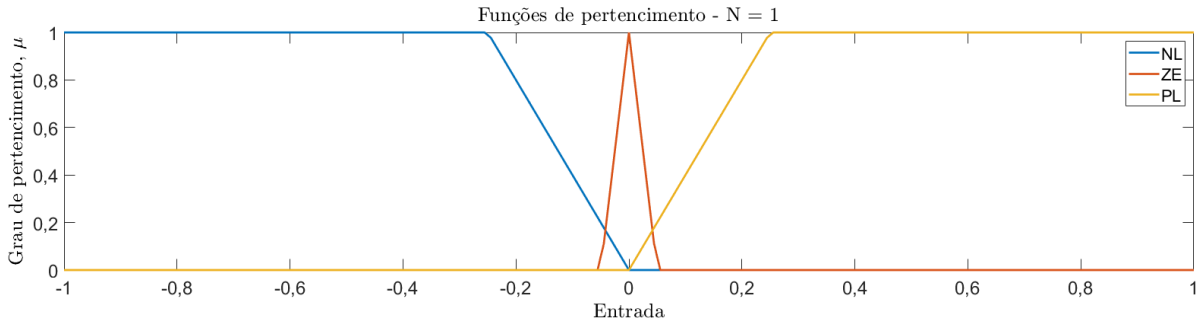

Figura 6.17: Funções de pertencimento para entradas para $N = 1$.
Fonte: Autor.

Os valores lingüísticos associados a estas funções de pertinência são:

1. NL: *Negative Large*
2. NM: *Negative Medium*
3. NS: *Negative Small*
4. ZE: *Zero*
5. PS: *Positive Small*
6. PM: *Positive Medium*
7. PL: *Positive Large*

As saídas possuem os mesmos valores lingüísticos das entradas, sendo estas valores constantes para um sistema Fuzzy tipo Sugeno.

1. Saída NL: -1

2. Saída NM: $-0{,}5$
3. Saída NS: $-0{,}25$
4. Saída ZE: 0
5. Saída PS: 0,25
6. Saída PM: 0,5
7. Saída PL: 1

Base de regras

São apresentadas na Figura 6.18 as cinco zonas que caracterizam a dinâmica existente em sistemas, e é a partir destas zonas que a base de regras do controlador Fuzzy será construída.

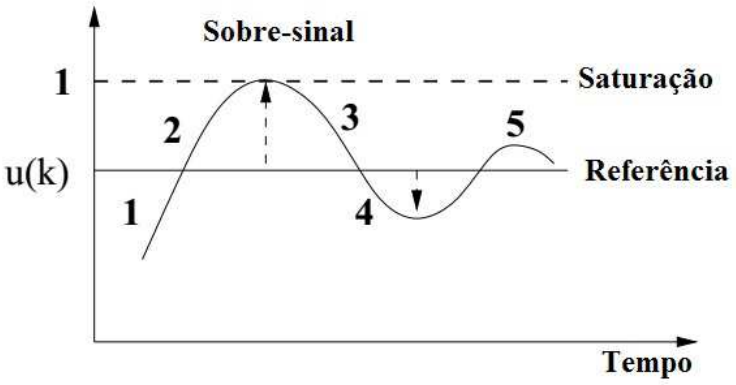

Figura 6.18: Características do controle de utilização.
Fonte: Adaptado de *Utilization Control and Optimization of Real-Time Embedded Systems*, 2015.

De maneira geral, o sinal de erro $e(k)$ define se o sistema está sobrecarregado ou subutilizado e o sinal da mudança do erro $\Delta e(k)$ define se a utilização do sistema cresce ou diminui. Pode-se definir então:

$$\begin{aligned} e(k) < 0, &\text{ Sistema sobrecarregado} \\ e(k) > 0, &\text{ Sistema subutilizado} \end{aligned} \tag{6.9}$$

$$\begin{aligned} \Delta e(k) < 0, &\text{ Utilização cresce} \\ \Delta e(k) > 0, &\text{ Utilização decresce} \end{aligned} \tag{6.10}$$

Usando (6.9) e (6.10) é possível realizar a análise de cada uma das zonas apresentadas na Figura 6.18 para o desenvolvimento das regras usadas no controlador. Desta forma temos:

1. Zona 1: Nesta zona temos $e(k) \geq 0$ e $\Delta e(k) \leq 0$, sendo assim, o sistema se encontra subutilizado e com a utilização crescendo. Nesta região, o sinal de controle deve ser determinado pela comparação das magnitudes da fuzzificação de $e(k)$ e $\Delta e(k)$, chamadas aqui de $F_{e(k)}$ e $F_{\Delta e(k)}$.

   - $|F_{e(k)}| > |F_{\Delta e(k)}|$: aqui temos um sistema subutilizado e com sua utilização crescendo lentamente. Dessa maneira o controlador deve aplicar uma saída mais alta para que a utilização aumente de maneira mais rápida.
   - $|F_{e(k)}| = |F_{\Delta e(k)}|$: aqui temos a situação do sistema na referência desejada, sendo assim, a saída do controlador deve ser zero para que o sistema permaneça no estado em que se encontra.
   - $|F_{e(k)}| < |F_{\Delta e(k)}|$: aqui temos um sistema subutilizado, porém com sua utilização crescendo de maneira muito rápida, sendo assim, o controlador deve aplicar uma saída mais baixa para que a utilização diminua e evite um sobressinal alto.

2. Zona 2: Nesta zona temos $e(k) < 0$ e $\Delta e(k) \leq 0$, sendo assim, o sistema se encontra sobrecarregado e com sua utilização crescendo, dessa forma o controlador deve aplicar uma saída mais baixa para que a utilização diminua.

3. Zona 3: Nesta zona temos $e(k) \leq 0$ e $\Delta e(k) > 0$, sendo assim, o sistema se encontra sobrecarregado e com sua utilização decrescendo. Igualmente a zona 1, o sinal de controle deve ser determinado pela comparação das magnitudes de $F_{e(k)}$ e $F_{\Delta e(k)}$.

   - $|F_{e(k)}| > |F_{\Delta e(k)}|$: aqui temos um sistema sobrecarregado e com sua utilização decrescendo lentamente, dessa maneira o controlador deve aplicar uma saída mais baixa para que a utilização diminua de maneira mais rápida.
   - $|F_{e(k)}| = |F_{\Delta e(k)}|$: aqui temos a situação do sistema na referência desejada, sendo assim, a saída do controlador deve ser zero para que o sistema permaneça no estado em que se encontra.
   - $|F_{e(k)}| < |F_{\Delta e(k)}|$: aqui temos um sistema sobrecarregado, porém com sua utilização decrescendo de maneira muito rápida, sendo assim, o controlador deve aplicar uma saída mais alta para que a utilização aumente e encontre a referência do sistema de maneira mais branda.

4. Zona 4: Nesta zona temos $e(k) > 0$ e $\Delta e(k) > 0$, sendo assim, o sistema se encontra subutilizado e com sua utilização decrescendo, dessa forma o controlador deve aplicar uma saída mais alta para que a utilização aumente.

5. Zona 5: Nesta zona temos $|e(k)| < \epsilon$ e $|\Delta e(k)| < \epsilon$, onde ϵ é a tolerância adotada. Neste cenário, temos a utilização do sistema na referência desejada, sendo assim, a saída do controlador deve ser zero para que o sistema permaneça no estado em que se encontra.

As regras para os controladores com $N = 3$, $N = 2$ e $N = 1$ são apresentadas nas Tabelas 6.3, 6.4 e 6.5 usando como base cada uma das cinco regiões descritas, sendo estas regras do tipo SE-ENTÃO. De acordo com cada uma das entradas é mostrado a função de pertinência assumida pela saída. Onde as saídas na cor vermelha pertencem à *Zona 1*, na cor azul à *Zona 2*, na cor verde à *zona 3*, na cor laranja à *zona 4* e na cor preta à *Zona 5*.

Tabela 6.3: Base de regras Fuzzy para $N = 3$.

		$\Delta e(k)$						
		NL	**NM**	**NS**	**ZE**	**PS**	**PM**	**PL**
$e(k)$	**NL**	NL	NL	NL	NL	NM	NM	ZE
	NM	NL	NL	NL	NM	NS	ZE	PS
	NS	NL	NL	NM	NS	ZE	PS	PM
	ZE	NL	NM	NS	ZE	PS	PM	PL
	PS	NM	NS	ZE	PS	PM	PL	PL
	PM	NS	ZE	PS	PM	PL	PL	PL
	PL	ZE	PS	PM	PL	PL	PL	PL

Tabela 6.4: Base de regras Fuzzy para $N = 2$.

		$\Delta e(k)$				
		NL	**NS**	**ZE**	**PS**	**PL**
$e(k)$	**NL**	NL	NL	NL	NM	ZE
	NS	NL	NM	NS	ZE	PM
	ZE	NL	NS	ZE	PS	PL
	PS	NM	ZE	PS	PM	PL
	PL	ZE	PM	PL	PL	PL

Tabela 6.5: Base de regras Fuzzy para $N = 1$.

		$\Delta e(k)$		
		NL	**ZE**	**PL**
$e(k)$	**NL**	NL	NL	ZE
	ZE	NL	ZE	PL
	PL	ZE	PL	PL

6.2 Validação

Para a validação dos controladores, foram realizadas simulações para todos os controladores desenvolvidos e o modelo desenvolvido no Capítulo 5. O *setpoint* escolhido para as simulações foi de 70% de utilização com tempo de amostragem $T_s = 0{,}1s$ e simulação da carga de trabalho pesada apresentada na Tabela 5.1. O valor de *setpoint* foi determinado para que exista uma margem de segurança para o caso de serem exigidas mais tarefas ao processador e não ocorra penalidades de desempenho por utilizações iguais a 100%. O tempo de amostragem usado é o mesmo apresentado no Capítulo 4.

Primeiramente serão apresentados os resultados obtidos para os controladores Fuzzy adaptados a partir do controlador PID e em seguida os controladores desenvolvidos pelo método de Wang.

6.2.1 Controladores Fuzzy adaptados a partir do controlador PID

O controlador PID desenvolvido foi sintonizado pelo método de Ziegler Nichols. Seus ganhos são apresentados na Tabela 6.6. A simulação para este controlador é apresentada na Figura 6.19.

Tabela 6.6: Ganhos do controlador PID.

Ganho	**Valor**
K_p	2,5
T_i	0,1572
T_d	0,04

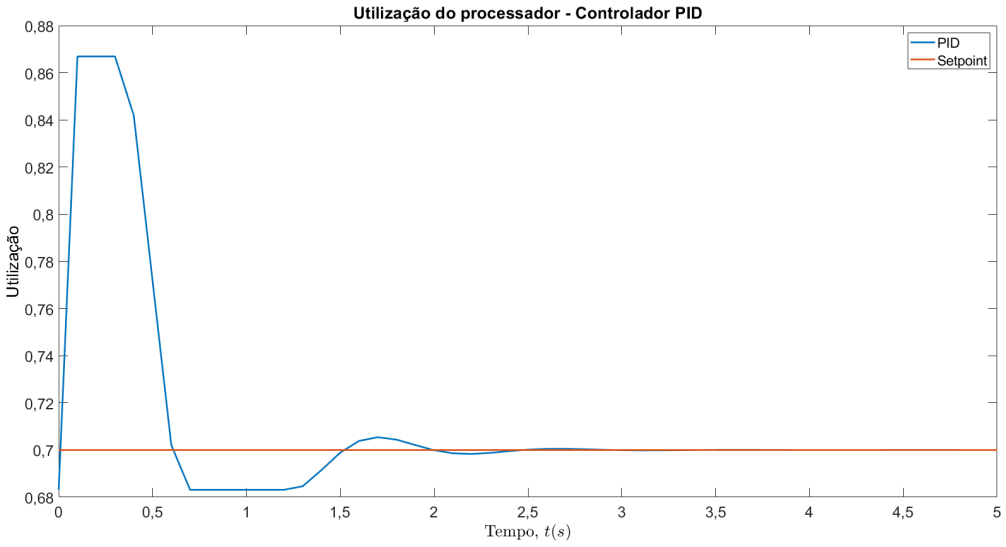

Figura 6.19: Curva de controle para controlador PID.
Fonte: Autor.

Controladores desenvolvidos com superfície linear

São apresentadas nas Figuras 6.20, 6.21 e 6.22 as curvas de controle obtidas para os controladores Fuzzy PD, Fuzzy PD + I e Fuzzy Incremental respectivamente. Os controladores Fuzzy PD e Fuzzy PD + I realizam o controle da carga na utilização desejada, porém com tempo de acomodação acima do obtido pelo controlador PID. O controlador Fuzzy Incremental não realizou o controle da carga, obtendo utilização acima do *setpoint* desejado.

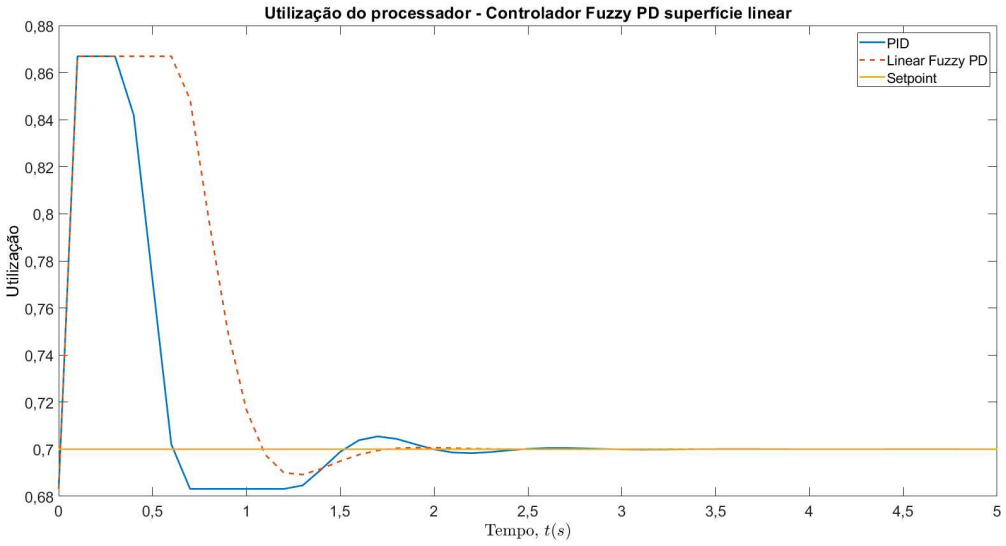

Figura 6.20: Curva de controle para controlador Fuzzy PD com superfície linear.
Fonte: Autor.

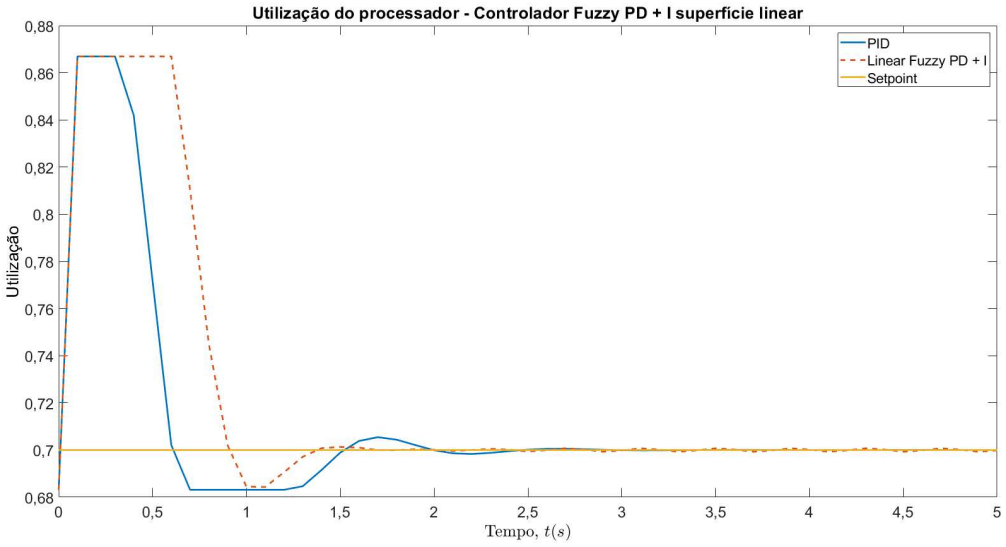

Figura 6.21: Curva de controle para controlador Fuzzy PD + I com superfície linear. Fonte: Autor.

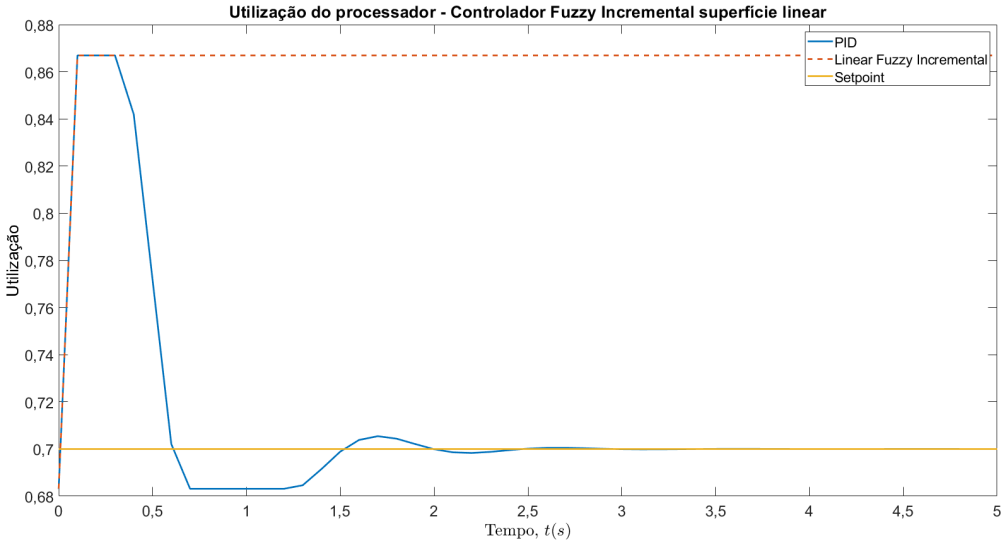

Figura 6.22: Curva de controle para controlador Fuzzy Incremental com superfície linear. Fonte: Autor.

Controladores desenvolvidos com superfície íngreme

São apresentadas nas Figuras 6.23, 6.24 e 6.25 as curvas de controle obtidas para os controladores Fuzzy PD, Fuzzy PD + I e Fuzzy Incremental respectivamente. O controlador Fuzzy PD realiza o controle da carga na utilização desejada, porém com tempo de acomodação acima do obtido pelo controlador PID. O controlador Fuzzy PD + I e Fuzzy Incremental não realizam o controle da carga na utilização desejada, com o controlador

Fuzzy PD + I apresentando oscilações e o controlador Fuzzy Incremental apresentado utilização acima da utilização desejada.

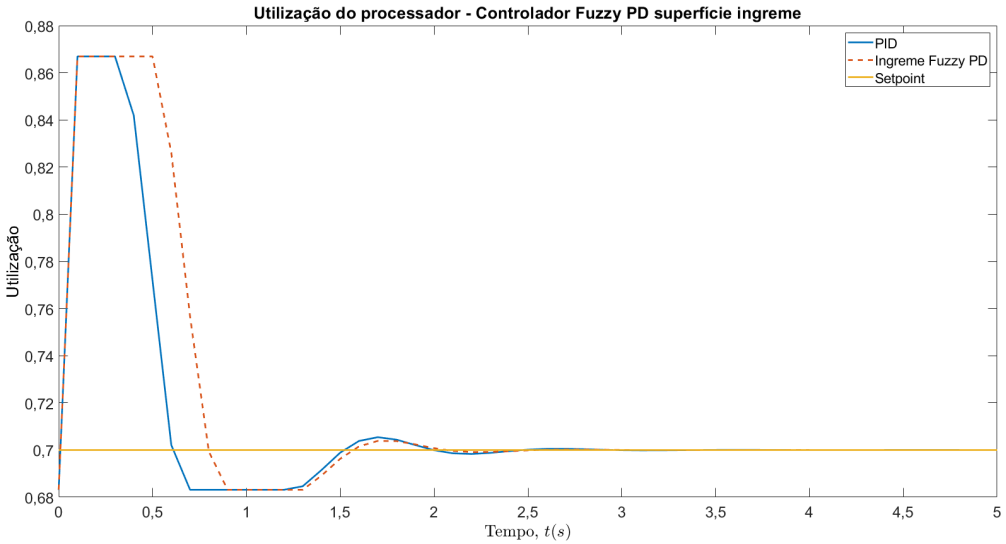

Figura 6.23: Curva de controle para controlador Fuzzy PD com superfície íngreme.
Fonte: Autor.

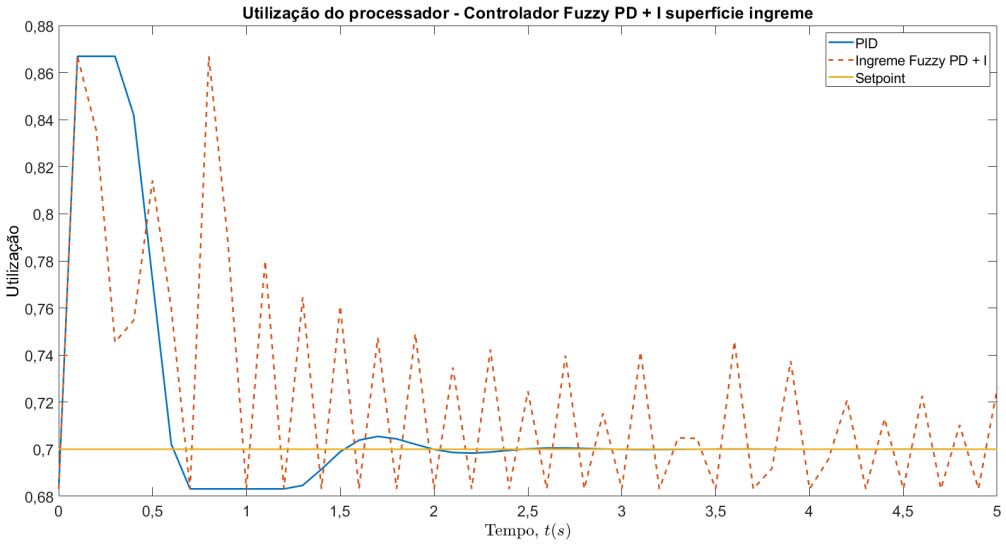

Figura 6.24: Curva de controle para controlador Fuzzy PD + I com superfície íngreme.
Fonte: Autor.

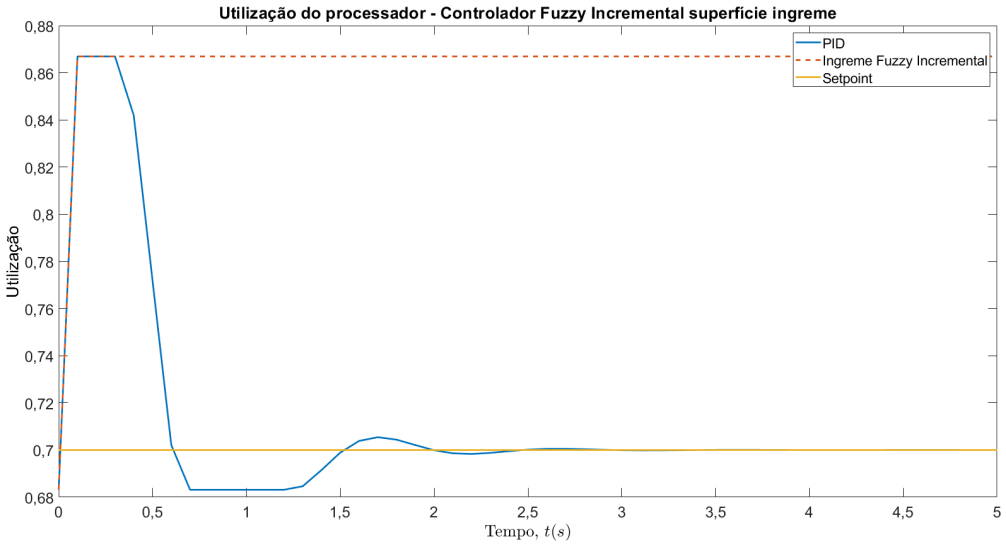

Figura 6.25: Curva de controle para controlador Fuzzy Incremental com superfície íngreme.

Fonte: Autor.

Controladores desenvolvidos com superfície delicadamente inclinada

São apresentadas nas Figuras 6.26, 6.27 e 6.28 as curvas de controle obtidas para os controladores Fuzzy PD, Fuzzy PD + I e Fuzzy Incremental respectivamente. Os controladores Fuzzy PD e Fuzzy PD + I realizam o controle da carga na utilização desejada, porém com tempo de acomodação acima do obtido pelo controlador PID. O controlador Fuzzy Incremental não realizou o controle da carga, obtendo utilização acima do *setpoint* desejado.

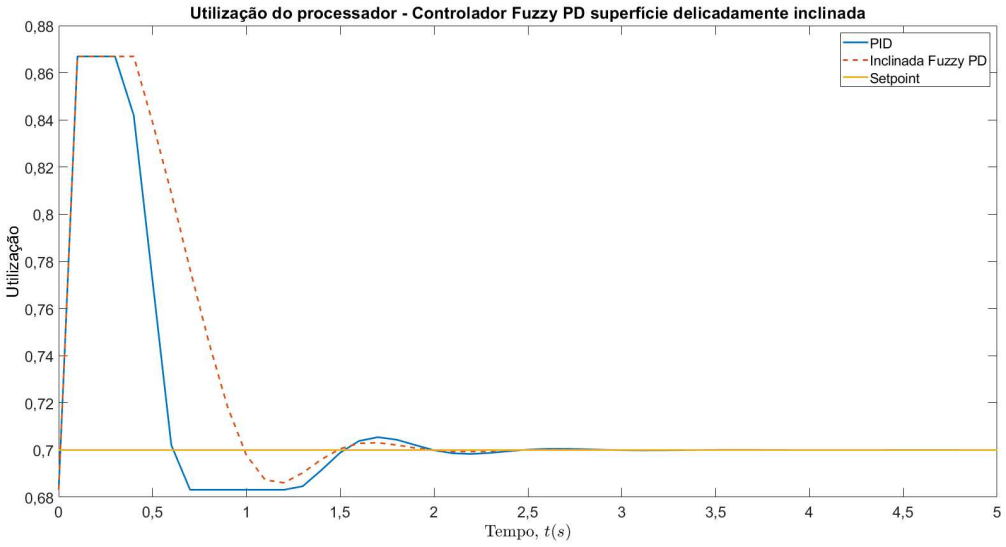

Figura 6.26: Curva de controle para controlador Fuzzy PD com superfície delicadamente inclinada.

Fonte: Autor.

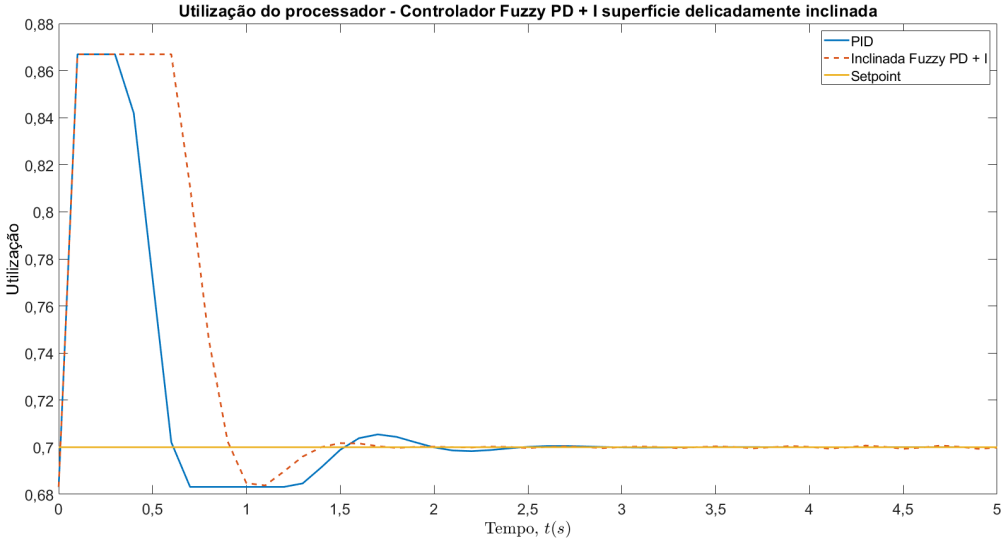

Figura 6.27: Curva de controle para controlador Fuzzy PD + I com superfície delicadamente inclinada.

Fonte: Autor.

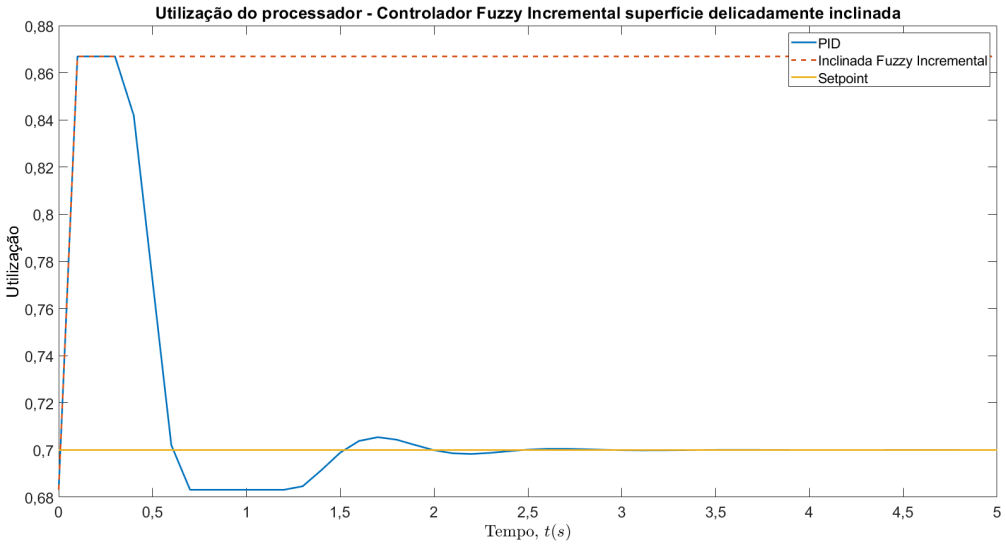

Figura 6.28: Curva de controle para controlador Fuzzy Incremental com superfície delicadamente inclinada.

Fonte: Autor.

Controladores desenvolvidos com superfície irregular

São apresentadas nas Figuras 6.29, 6.30 e 6.31 as curvas de controle obtidas para os controladores Fuzzy PD, Fuzzy PD + I e Fuzzy Incremental respectivamente. O controlador Fuzzy PD realiza o controle da carga na utilização desejada, porém com tempo de acomodação acima do obtido pelo controlador PID. O controlador Fuzzy PD + I também realiza o controle da carga na utilização desejada, porém apresenta oscilações até se estabilizar e o controlador Fuzzy Incremental não realiza o controle da carga, apresentado utilização acima da utilização desejada.

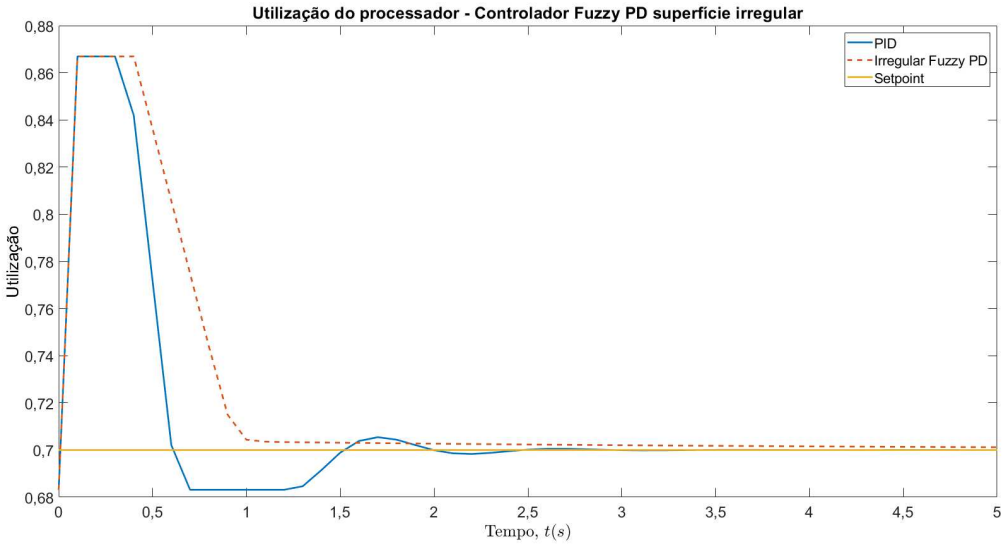

Figura 6.29: Curva de controle para controlador Fuzzy PD com superfície delicadamente inclinada.

Fonte: Autor.

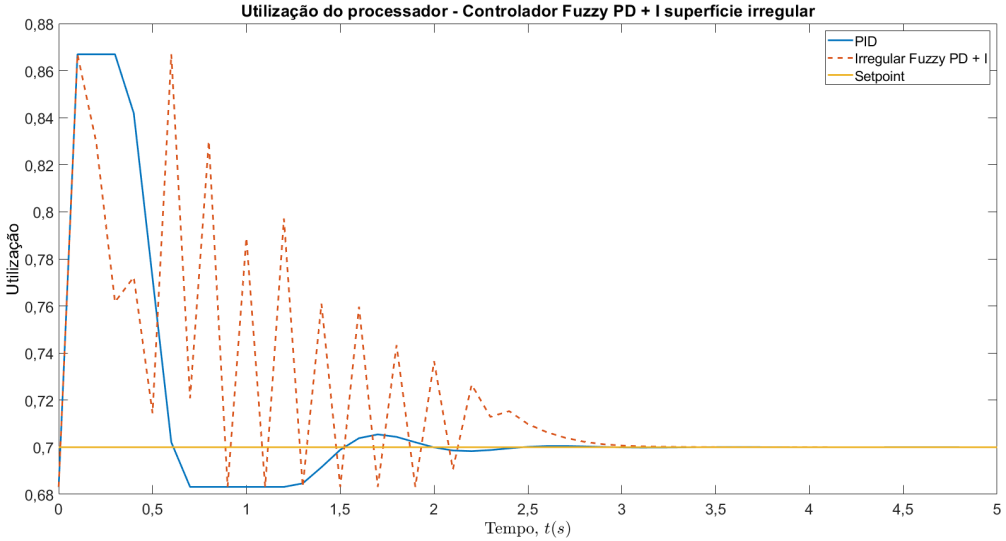

Figura 6.30: Curva de controle para controlador Fuzzy PD + I com superfície delicadamente inclinada.

Fonte: Autor.

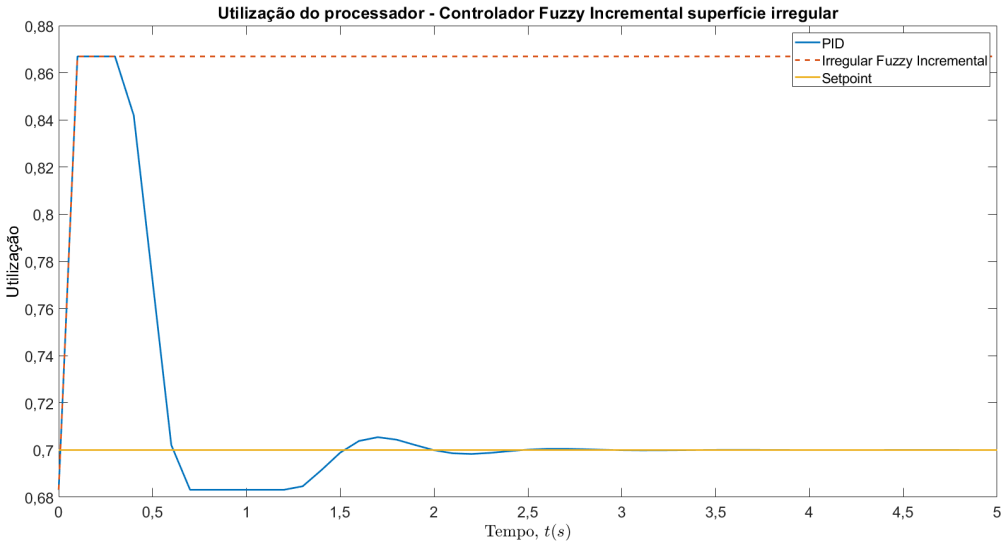

Figura 6.31: Curva de controle para controlador Fuzzy Incremental com superfície delicadamente inclinada.

Fonte: Autor.

Controladores desenvolvidos com método de Wang

Para a validação dos controladores desenvolvidos pelo método de Wang foram realizadas simulações com as mesmas condições usadas para os controladores Fuzzy adaptados do controlador PID. O *setpoint* foi de 70% de utilização com tempo de amostragem $T_s = 0{,}1s$ e simulação da carga de trabalho pesada apresentada na Tabela 5.1. Os resultados são apresentados nas Figuras 6.32, 6.33 e 6.34, para os controladores com $N = 3$, $N = 2$ e $N = 1$ respectivamente.

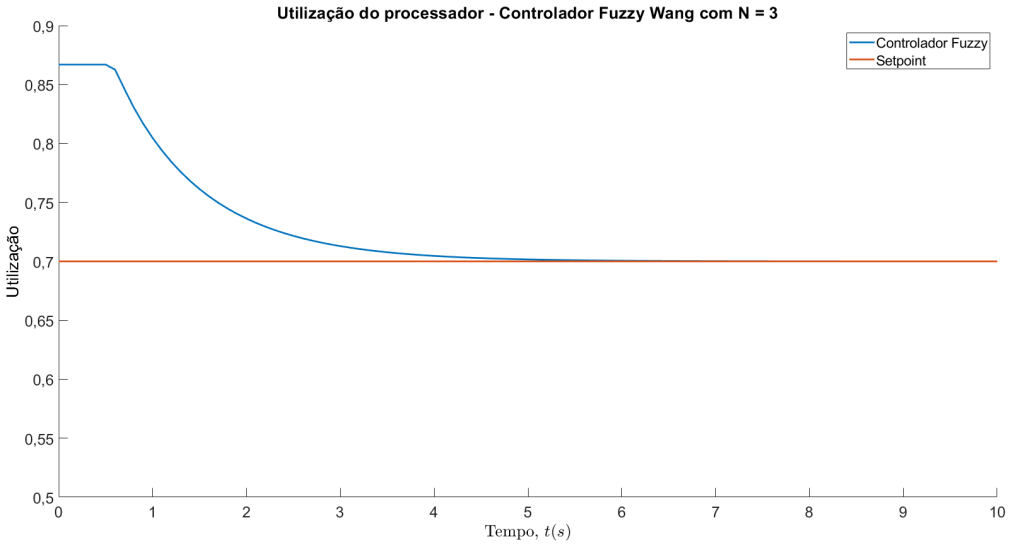

Figura 6.32: Curva de controle para controlador Fuzzy com método de Wang e $N = 3$. Fonte: Autor.

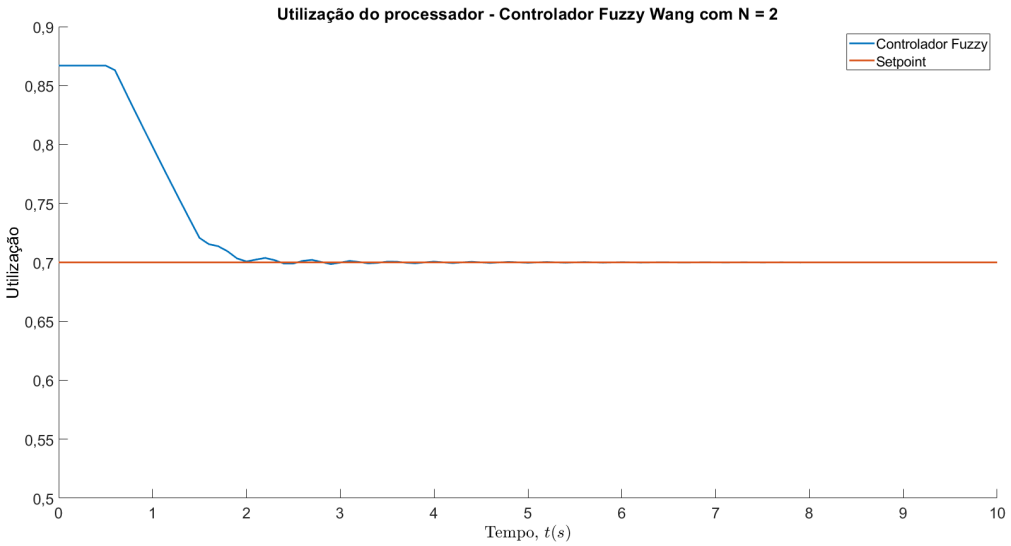

Figura 6.33: Curva de controle para controlador Fuzzy com método de Wang e $N = 2$. Fonte: Autor.

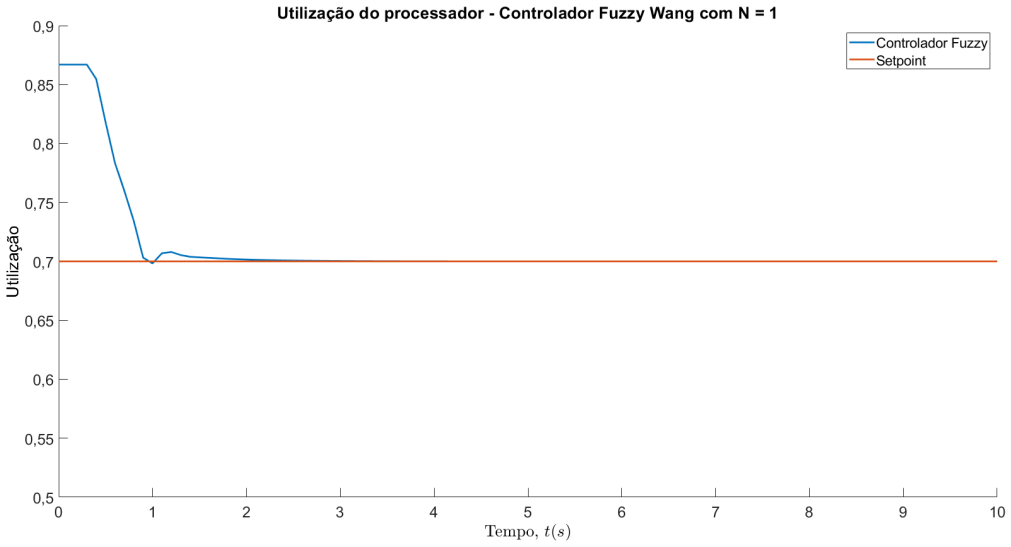

Figura 6.34: Curva de controle para controlador Fuzzy com método de Wang e $N = 1$. Fonte: Autor.

Todos os controladores realizaram o controle da carga na utilização desejada, praticamente sem *overshoot* e com tempos de acomodação diferentes, onde o maior foi obtido pelo controlador com $N = 3$ e o menor com o controlador $N = 1$, isto deve-se ao último controlador possuir apenas as saídas *Positive Large*, *Zero* e *Negative Large*, dessa forma conseguindo ações de controle mais rápidas que os controladores anteriores.

6.3 Conclusão

Este capítulo apresentou o desenvolvimento dos controladores usados neste trabalho. Apresentando o diagrama de blocos geral dos controladores, suas entradas e saídas. As funções de pertinências das entradas e da saída como também todos os conceitos usados no desenvolvimento da base de regras do controladores. A validação dos mesmos foi realizada por simulações utilizando o modelo desenvolvido no Capítulo 5. Os melhores resultados foram obtidos com os controladores desenvolvidos com o método de Wang e estes foram selecionados para aplicações reais e comparações com o *Governor Ondemand*.

Capítulo 7

Resultados Experimentais

Neste capítulo encontram-se os resultados práticos obtidos com os controladores desenvolvidos em comparação ao *Governor Ondemand*. Os experimentos foram divididos em diversas cargas de trabalho e foram usadas cargas de mesma natureza as apresentadas no Capítulo 5.

7.1 Experimentos com cargas fixas

São apresentadas na Tabela 7.1 as cargas utilizadas em cada teste. Os controladores são atualizados a cada $100ms$ para garantir a atualização dos valores numéricos do arquivo /proc/stat, porém aquisições de dados foram realizadas a cada $1s$ devido a limitações na taxa de amostragem do *WattsUp*. O consumo de energia para cada uma das cargas pode ser quantificado como apresentado em (7.1). Onde $E(t)$ é o consumo de energia em Wh e $P(t)$ é a potência consumida ao longo do tempo em W . Cada teste tem duração de 2 minutos e os resultados obtidos são apresentados a seguir.

$$E(t) = \int_0^t P(t)dt \tag{7.1}$$

Tabela 7.1: Descrição das cargas de trabalho usadas nos experimentos com carga fixa.

Experimento	Descrição da carga de trabalho
Carga 1	Vídeo com resolução 4K (3840x2160) 60 fps
Carga 2	Vídeo com resolução 1080p (1920x1080) 60 fps
Carga 3	Vídeo com resolução 1080p (1920x1080) 30 fps
Carga 4	Vídeo com resolução 720p (1280x720) 30 fps
Carga 5	Vídeo com resolução 360p (640x360) 30 fps

São apresentadas nas Figuras 7.1, 7.2, 7.3, 7.4 e 7.5 os resultados obtidos para os experimentos com as cargas 1, 2, 3, 4 e 5 respectivamente para o *Governor Ondemand*.

As potências, utilizações e consumo de energia médias obtidas em cada experimento são apresentados na Tabela 7.2.

É possível observar que potência, utilização e frequência são maiores para a primeira carga, já que esta é a que mais exige processamento e com isso o consumo de energia da mesma é superior as outras cargas. A segunda carga é processada basicamente na mesma frequência que a primeira carga, porém, por ser uma carga mais leve, a utilização do processador é menor que na primeira carga e com isso o consumo de potência também é menor para esta. Este comportamento se repete para as demais cargas de trabalho tanto para a potência quanto para a utilização, já que o *Governor Ondemand* fez o controle de todas as cargas utilizando frequências próximas a frequência máxima do processador.

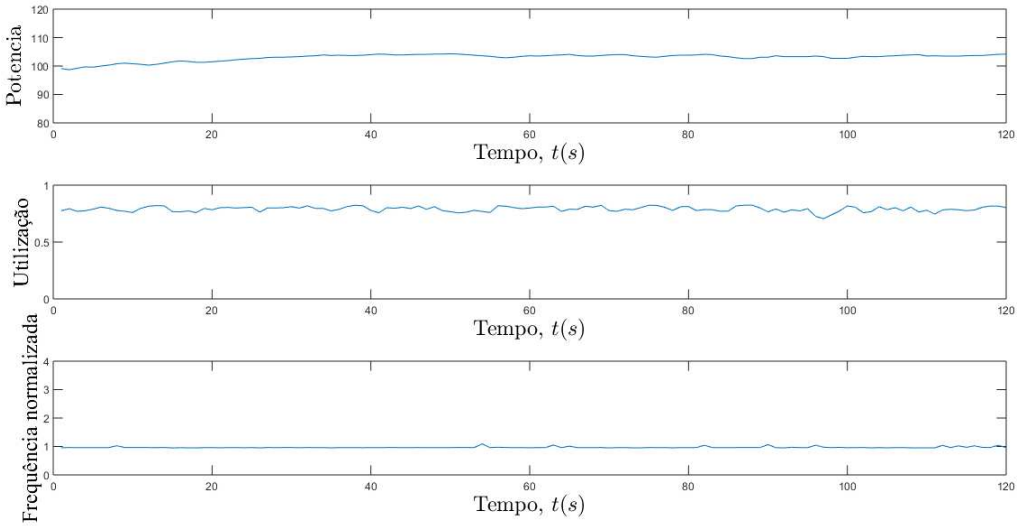

Figura 7.1: Resultados para o experimento com carga 1 para o *Governor Ondemand*.
Fonte: Autor.

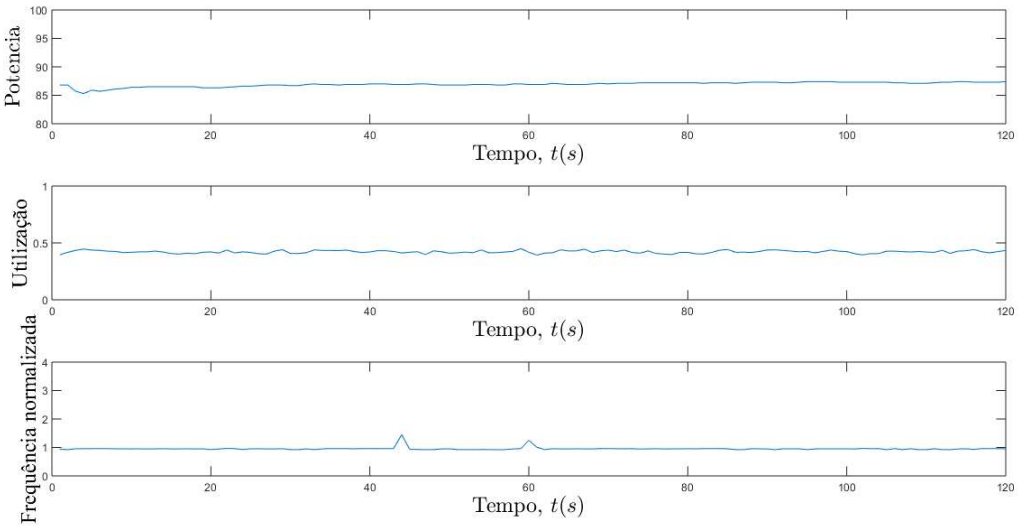

Figura 7.2: Resultados para o experimento com carga 2 para o *Governor Ondemand*. Fonte: Autor.

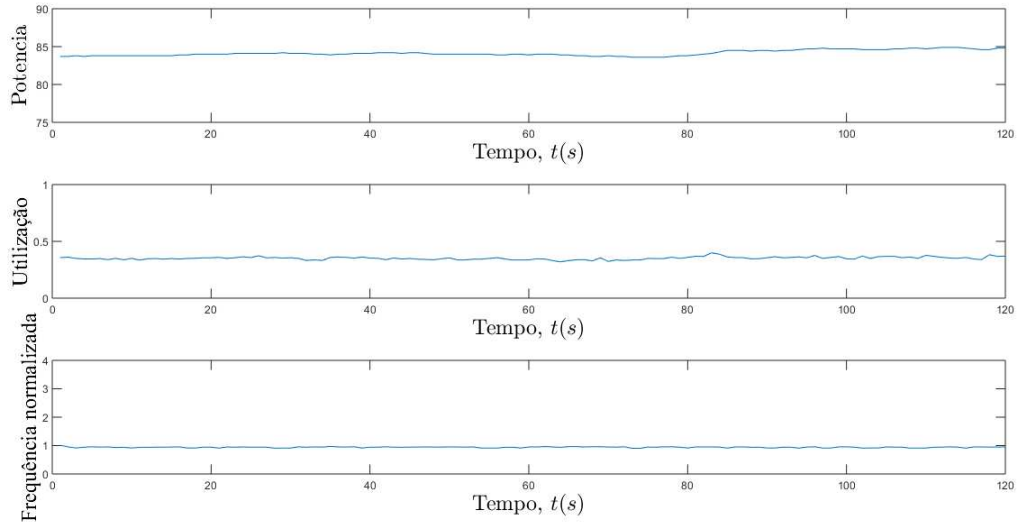

Figura 7.3: Resultados para o experimento com carga 3 para o *Governor Ondemand*. Fonte: Autor.

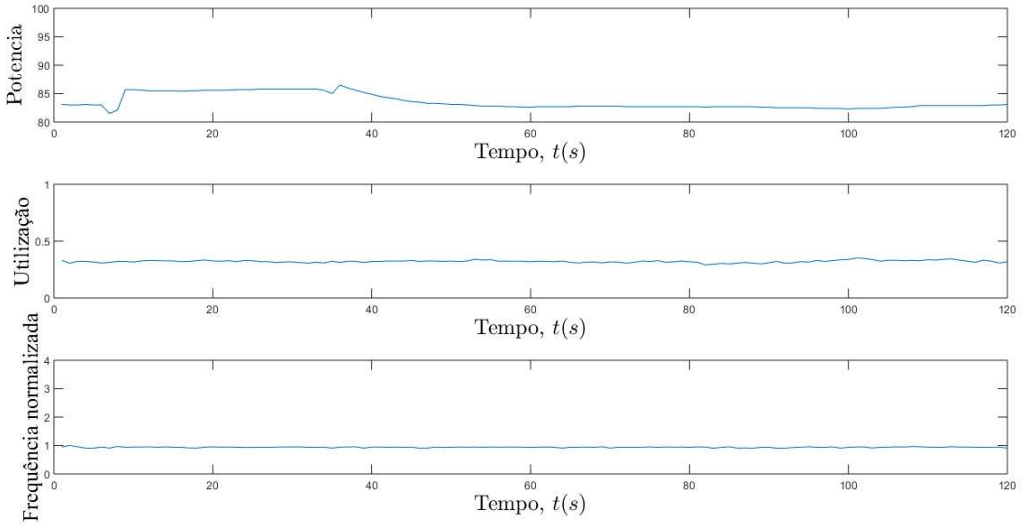

Figura 7.4: Resultados para o experimento com carga 4 para o *Governor Ondemand*.
Fonte: Autor.

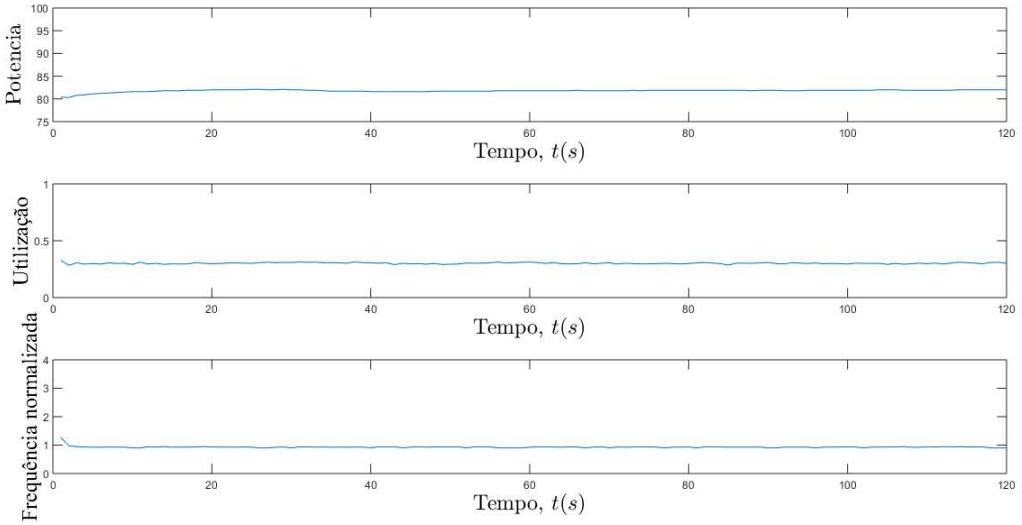

Figura 7.5: Resultados para o experimento com carga 5 para o *Governor Ondemand*.
Fonte: Autor.

São apresentadas nas Figuras 7.6, 7.7, 7.8, 7.9 e 7.10 os resultados obtidos para os experimentos com as cargas 1, 2, 3, 4 e 5 respectivamente para o controlador Fuzzy desenvolvido. As potências e utilizações médias obtidas em cada experimento são apresentados na Tabela 7.3.

Novamente é possível observar que potência, utilização e frequência são maiores para a primeira carga, já que esta é a que mais exige processamento e com isso o consumo de energia da mesma é superior as outras cargas. Para a segunda carga já existe uma grande redução na frequência empregada, já que esta carga é mais leve que a primeira e com

Tabela 7.2: Potência, utilização e consumo de energia médias para as cargas 1, 2, 3, 4, 5 e *Governor Ondemand*.

Ondemand			
Experimento	Potência (W)	Utilização	Energia (kWh)
Carga 1	103,0033	0,7899	0,2600
Carga 2	86,9208	0,4214	0,1738
Carga 3	84,1508	0,3515	0,1683
Carga 4	83,5625	0,3213	0,1671
Carga 5	81,7808	0,3019	0,1636

isso exige menos processamento, resultando em um menor consumo de potência. Para a terceira carga, a redução da frequência se repete, novamente por ser uma carga mais leve e necessitar de menos processamento. Para as demais cargas esse comportamento se repete, dessa forma são usadas frequências menores, capazes de realizar o processamento destas cargas sem causar penalidade de desempenho.

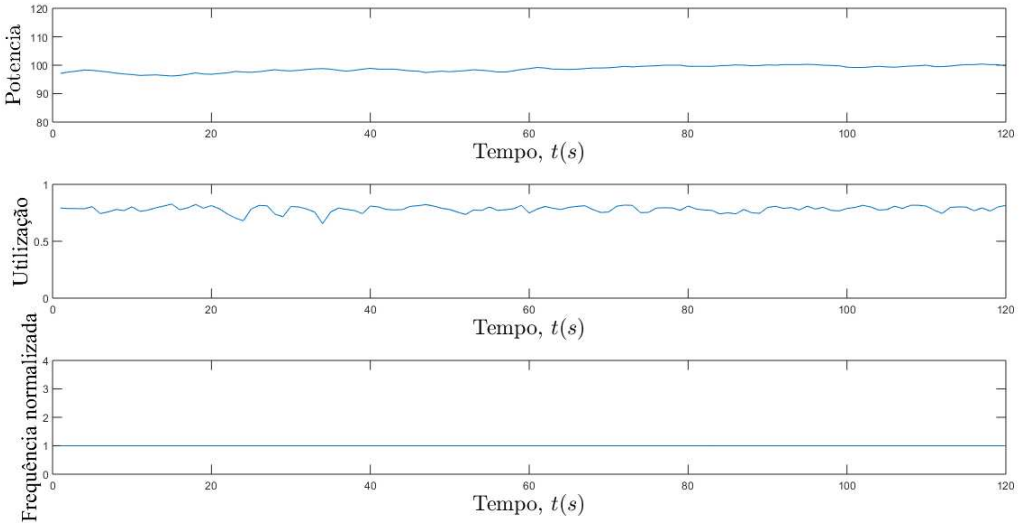

Figura 7.6: Resultados para o experimento com carga 1 para o controlador Fuzzy.
Fonte: Autor.

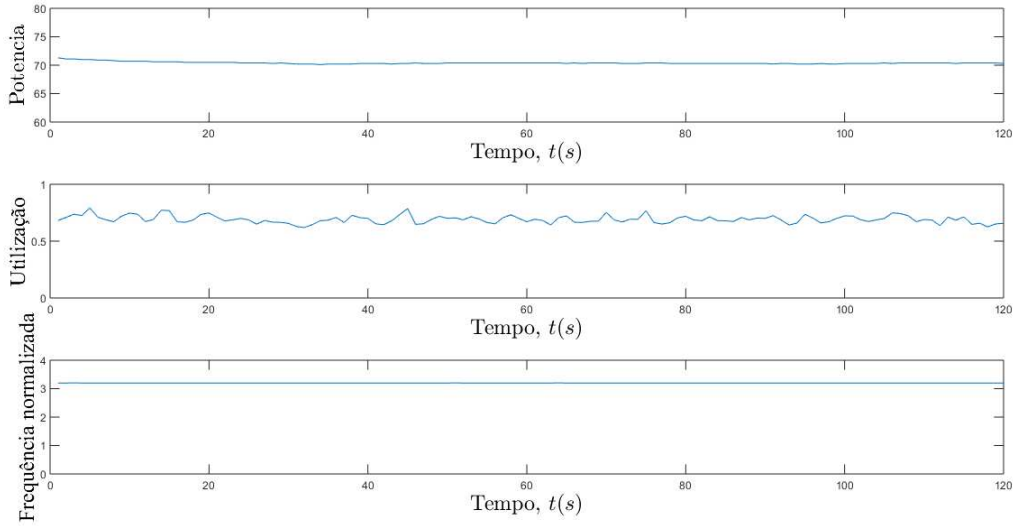

Figura 7.7: Resultados para o experimento com carga 2 para o controlador Fuzzy. Fonte: Autor.

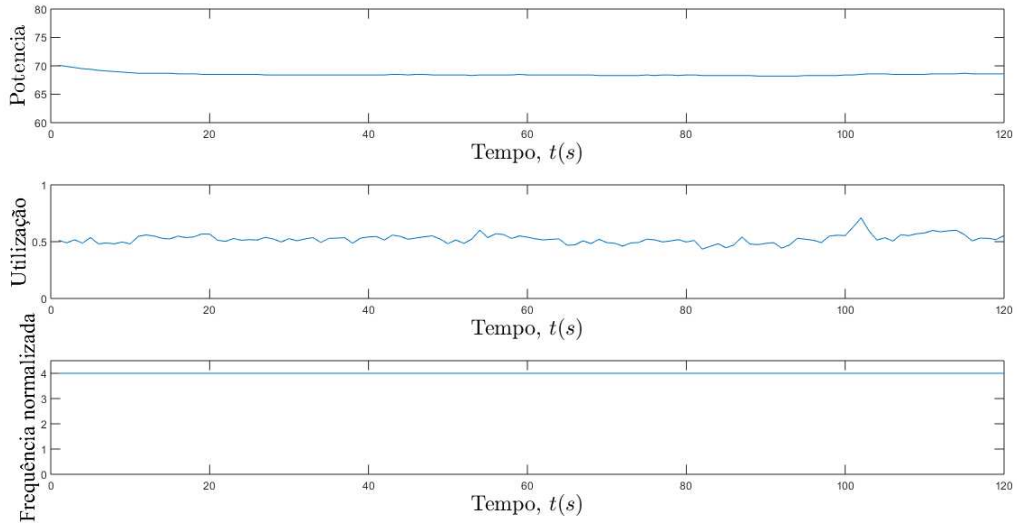

Figura 7.8: Resultados para o experimento com carga 3 para o controlador Fuzzy. Fonte: Autor.

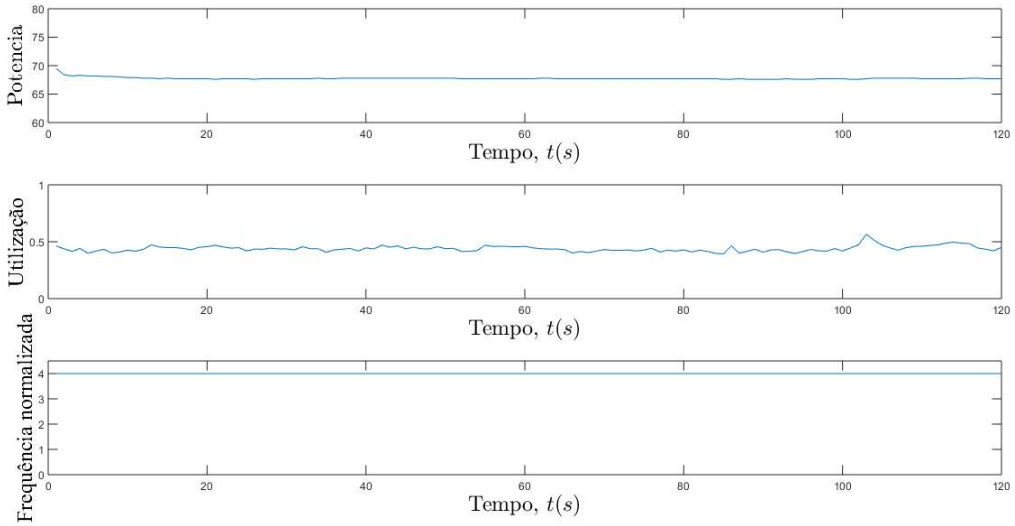

Figura 7.9: Resultados para o experimento com carga 4 para o controlador Fuzzy. Fonte: Autor.

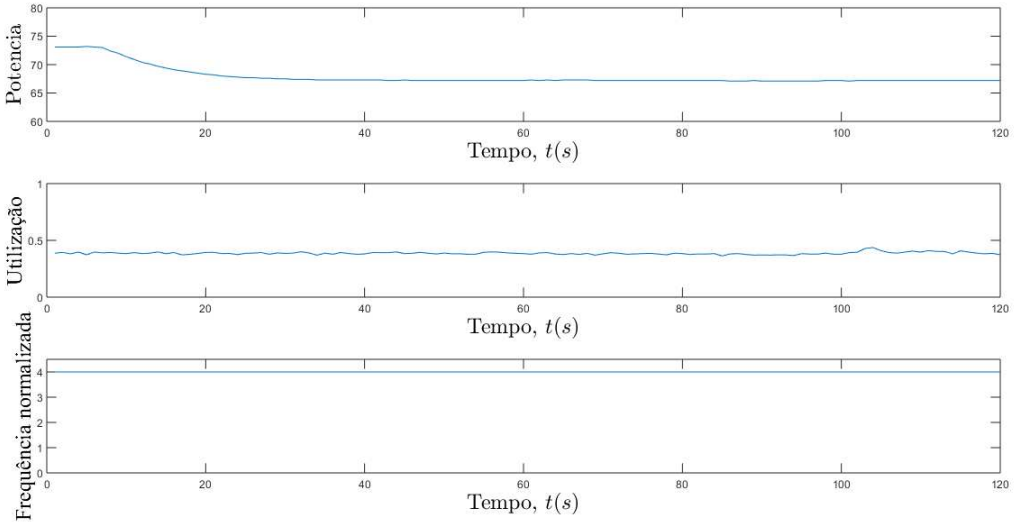

Figura 7.10: Resultados para o experimento com carga 5 para o controlador Fuzzy. Fonte: Autor.

Para a carga 1, o *setpoint* não é possível de ser alcançado, já que mesmo em frequência máxima a carga gera utilizações maiores que 70%, neste cenário a ação do *Governor Ondemand* e do controlador Fuzzy são as mesmas, executar a carga em máxima frequência. Para a carga 2, é possível alcançar o *setpoint*, desse forma o controlador Fuzzy realiza esta ação, o *Governor Ondemand* executa a carga em frequências maiores, mantendo a utilização do processador mais baixa, porém com maior consumo de potência. Para as cargas 3, 4 e 5 o *setpoint* não pode ser alcançado pois mesmo na frequência mínima, as cargas geram utilizações menores que o mesmo, dessa forma, o controlador Fuzzy executa

Tabela 7.3: Potência, utilização e consumo de energia médias para as cargas 1, 2, 3, 4, 5 e controlador Fuzzy.

Controlador Fuzzy			
Experimento	Potência (W)	Utilização	Energia (kWh)
Carga 1	101,7025	0,7820	0.2034
Carga 2	70,4042	0,6984	0.1408
Carga 3	68,5083	0,5233	0.1370
Carga 4	67,7633	0,4383	0.1355
Carga 5	65,9000	0,3863	0.1318

estas cargas em mínima frequência, reduzindo o consumo de potência, já o *Governor Ondemand* executa estas cargas com frequências maiores, aumentando o consumo de potência. A porcentagem de redução de potência para cada carga é apresentada na Tabela 7.4. Para a carga 1, a redução foi de 1,2629% praticamente não reduzindo. Para as cargas 2, 3, 4 e 5 a redução foi de 19,0019%, 18,5887%, 18,9070% e 19,4187% respectivamente, representando uma redução média de 18,9791% para cargas pesadas, médias, leves e mínimas.

Tabela 7.4: Porcentagem de redução de potência.

Experimento	Redução de potência
Carga 1	1,2629%
Carga 2	19,0019%
Carga 3	18,5887%
Carga 4	18,9070%
Carga 5	19,4187%

Capítulo 8

Conclusões

Nas condições em que foram realizados os experimentos e baseando-se na proposta inicialmente estabelecida, este trabalho mostrou a possibilidade do uso de controladores Fuzzy para a realização do gerenciamento de energia em processadores para computadores de propósito geral.

Neste trabalho foram apresentados tópicos importantes para o entendimento do consumo de energia em processadores de computadores de propósito geral. Primeiramente foram apresentados os conceitos sobre consumo de energia em processadores, enfatizando sua parte construtiva e destacando como principais causas a tensão de alimentação, chaveamento e corrente de fuga. Em seguida foram apresentados conceitos de escalonamento dinâmico de tensão e frequência, mostrando que o controle de frequência é efetivo no gerenciamento de energia no processador.

Foi apresentado um modelo de tarefas periódicas, enfatizando os tempos de execução de cada tarefa, sendo necessário a estimação destes parâmetros em tempo de execução para o uso deste modelo em políticas de gerenciamento de energia. A partir deste modelo de tarefas, foi desenvolvido um modelo Fuzzy para o processador, tornando-o mais genérico, já que não é necessária as estimações dos tempos de execução, sendo possível estimar a utilização do processador apenas por sua frequência atual e ociosidade atrasada.

Para o modelo do processador foram apresentados conceitos sobre o sistema ANFIS, uma técnica híbrida, juntando redes neurais e lógica Fuzzy, além do uso de algoritmos de clusterização para reconhecimento de padrões entre frequências, utilizações e ociosidades em processadores. Foram desenvolvidos cinco modelos para o processador da plataforma experimental, validando o melhor pelo conjunto do teste qui-quadrado, usado para verificar se a frequência de ocorrência de uma amostra se desvia significantemente da ocorrência esperada e o erro médio quadrático, usada para determinar o melhor modelo entre os modelos validados pelo teste qui-quadrado.

Para a aquisição de dados, foi desenvolvida uma plataforma experimental, unindo

algoritmos para medir utilizações e ociosidades do processador em sistemas *Linux* a um medidor de energia para a aquisição da potência consumida pelo computador. Além de medições, os algoritmos desenvolvidos permitem a aplicação de diferentes frequências em cada núcleo do processador, sendo possível o estudo da dinâmica de execução de diferentes cargas de trabalho em diferentes frequências.

Os controladores foram desenvolvidos usando-se técnicas de conversão de controladores PID em controladores Fuzzy e controladores construídos por conhecimento especialista, validados por simulações utilizando o próprio modelo desenvolvido no trabalho. Escolhendo-se o melhor com base nos resultados obtidos nestas simulações.

O controlador se mostrou eficiente na redução do consumo de energia quando comparado ao *Governor Ondemand*, já que adota frequências mais baixas para algumas cargas de trabalho que o adotado pelo *Governor* sem comprometer o desempenho. Porém políticas de escolhas de *setpoints* ideais para cada carga de trabalho aplicada, ou métodos mais robustos para o desenvolvimento de controladores Fuzzy, podem trazer bons benefícios na relação consumo e desempenho.

8.1 Principais contribuições desta dissertação

1. Desenvolvimento de modelos Fuzzy de processadores capazes de estimar a utilização usando a frequência atual aplicada ao processador e sua ociosidade atrasada, não sendo necessário o uso de técnicas de estimações em tempo real.

2. Desenvolvimento de uma plataforma experimental para estudo e controle de processadores em sistemas *Linux*.

3. Estudo sobre o comportamento de processadores na execução de cargas de trabalho de diferentes intensidades.

4. Desenvolvimento de controladores Fuzzy para o gerenciamento de energia em processadores.

8.2 Sugestões para trabalhos futuros

Existe muito ainda a ser desenvolvido para um melhor gerenciamento de energia em processadores. As recomendações a seguir podem ser úteis para o seguimento da pesquisa.

1. Uso de outras técnicas para o gerenciamento de energia em processadores, como redes neurais ou algoritmos genéticos;

2. Desenvolvimento do modelo inverso do processador, para a aplicação de outras técnicas de controle.

3. Desenvolvimento de política de gerenciamento com *setpoint* ajustável para cada tipo de carga de trabalho;

4. Como determinar a base de regras com o objetivo de minimizar o consumo de energia, satisfazendo uma restrição de penalidade de desempenho.

Referências Bibliográficas

1 TIWARI, V. et al. Dynamic power management for microprocessors: A case study. In: IEEE. *VLSI Design, 1997. Proceedings., Tenth International Conference on*. 1997. p. 185–192.

2 LUIZ, S. O. D. Gerenciamento dinâmico de energia em processadores com cargas de trabalho variantes no tempo. *Campina Grande, PB, UFCG*, 2012.

3 LOVE, R. *Linux kernel development*. : Pearson Education, 2010.

4 BARROS, C. A. de. Redução do consumo energético de aplicações paralelas eficientes em arquiteturas multicore. *Rio Grande do Norte, RN, UFRN*, 2014.

5 BOSNACKI, D. et al. Parallel probabilistic model checking on general purpose graphics processors. v. 13, p. 21–35, 01 2011.

6 DARGIE, W. A stochastic model for estimating the power consumption of a processor. v. 64, 04 2014.

7 DARGIE, W.; WEN, J. A probabilistic model for estimating the power consumption of processors and network interface cards. p. 845–852, 2013.

8 BENINI, L. et al. Policy optimization for dynamic power management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 18, n. 6, p. 813–833, 1999.

9 LUIZ, S. O. D. et al. Técnica de gerenciamento dinâmico de energia orientada à autonomia da bateria para sistemas embarcados. In: *Anais do XVII Congresso Brasileiro de Automática (CBA 2008)*. Juiz de Fora, Brazil: Sociedade Brasileira de Automática, 2008.

10 SRIDHARAN, R.; GUPTA, N.; MAHAPATRA, R. Feedback-controlled reliability-aware power management for real-time embedded systems. In: IEEE. *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*. 2008. p. 185–190.

11 WANG, X. et al. Power-aware cpu utilization control for distributed real-time systems. In: IEEE. *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*. 2009. p. 233–242.

12 CHEN, X. et al. Dynamic voltage and frequency scaling for shared resources in multicore processor designs. In: ACM. *Proceedings of the 50th Annual Design Automation Conference*. 2013. p. 114.

13 ANGHEL, I. et al. Dynamic frequency scaling algorithms for improving the cpu's energy efficiency. In: IEEE. *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*. 2011. p. 485–491.

14 ENOMOTO, T.; KOBAYASHI, N. A low power multimedia processor implementing dynamic voltage and frequency scaling technique. In: IEEE. *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*. 2013. p. 75–76.

15 HE, X.-B. et al. An adaptive fuzzy scheduling and fuzzy-pid perfomance control model being suitable to soft real-time systems. In: IEEE. *Information Assurance and Security, 2009. IAS'09. Fifth International Conference on*. 2009. v. 1, p. 149–153.

16 RATTANATAMRONG, P.; FORTES, J. A. Fuzzy scheduling of real-time ensemble systems. In: IEEE. *High Performance Computing & Simulation (HPCS), 2014 International Conference on*. 2014. p. 146–153.

17 ZERKAOUI, S.; HAJJAJI, A. E.; BOSCHE, J. On-line control strategy for instantaneous power management of hybrid power system based on dynamic fuzzy logic controller. In: IEEE. *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. 2012. p. 1130–1136.

18 BASARAN, C. et al. Robust fuzzy cpu utilization control for dynamic workloads. *Journal of Systems and Software*, Elsevier, v. 83, n. 7, p. 1192–1204, 2010.

19 JACOMET, M.; WALTI, R. A vlsi fuzzy processor with parallel rule execution. In: *Proceedings of IEEE 5th International Fuzzy Systems*. 1996. v. 1, p. 554–558 vol.1.

20 YANG, Z. et al. An electricity data cluster analysis method based on saga-fcm algorithm. In: *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*. 2017. p. 389–393.

21 O'NEAL, K.; BRISK, P. Predictive modeling for cpu, gpu, and fpga performance and power consumption: A survey. In: *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2018. p. 763–768. ISSN 2159-3477.

22 GU, L. A novel subtractive clustering by using k-harmonic means clustering for initialization. In: *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 2016. p. 840–843. ISSN 2327-0594.

23 AHMED, A.; LOO, C. K.; OBO, T. Neuro-fuzzy model with subtractive clustering optimization for arm gesture recognition by angular representation of kinect data. In: *2017 6th International Conference on Informatics, Electronics and Vision 2017 7th International Symposium in Computational Medical and Health Technology (ICIEV-ISCMHT)*. 2017. p. 1–6.

24 WOLF, W. *Computers as Components*: Principles of embedded computing system design. : Morgan Kaufmann, 2016.

25 CHANDRAKASAN, A. P.; SHENG, S.; BRODERSEN, R. W. Low-power cmos digital design. *IEICE Transactions on Electronics*, The Institute of Electronics, Information and Communication Engineers, v. 75, n. 4, p. 371–382, 1992.

26 BURD, T. D.; BRODERSEN, R. W. Energy efficient cmos microprocessor design. In: IEEE. *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*. 1995. v. 1, p. 288–297.

27 MUDGE, T. Power: A first-class architectural design constraint. *Computer*, IEEE Computer Society Press, v. 34, n. 4, p. 52–58, 2001.

28 E SALUTA K, L. M. W. Minimizing energy consumption for high-performance processing. *Design Automation Conference*, IEEE Computer Society Press, p. 199–204, 2002.

29 KIM, N. S. et al. Leakage current: Moore's law meets static power. *computer*, IEEE, v. 36, n. 12, p. 68–75, 2003.

30 VENKATACHALAM, V.; FRANZ, M. Power reduction techniques for microprocessor systems. *ACM Computing Surveys (CSUR)*, ACM, v. 37, n. 3, p. 195–237, 2005.

31 CHANG, L. et al. Practical strategies for power-efficient computing technologies. *Proceedings of the IEEE*, IEEE, v. 98, n. 2, p. 215–236, 2010.

32 KORTHIKANTI, V. A.; AGHA, G. Towards optimizing energy costs of algorithms for shared memory architectures. In: ACM. *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*. 2010. p. 157–165.

33 USMAN, S.; KHAN, S. U.; KHAN, S. A comparative study of voltage/frequency scaling in noc. In: *EIT*. 2013. p. 1–5.

34 COHEN, D. et al. A car racing based strategy for the dynamic voltage and frequency scaling technique. In: IEEE. *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*. 2012. p. 774–779.

35 SHIN, D.; KIM, J.; LEE, S. Intra-task voltage scheduling for low-energy hard real-time applications. *IEEE Design & Test of Computers*, v. 18, n. 2, p. 20–30, 2001.

36 LEBLEBICI, Y. *CMOS Digital Integrated Circuits: Analysis and Design*. : McGraw-Hill College, 1996.

37 LEE, I.; LEUNG, J. Y.; SON, S. H. *Handbook of real-time and embedded systems*. : CRC Press, 2007.

38 CHOI, K. et al. Dynamic voltage and frequency scaling under a precise energy model considering variable and fixed components of the system power dissipation. In: IEEE COMPUTER SOCIETY. *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*. 2004. p. 29–34.

39 CHENCI, G. P.; RIGNEL, D. G.; LUCAS, C. A. Uma introdução á lógica fuzzy. *Revista Eletrônica de Sistemas de Informação e de Gestão Tecnológica*, v. 1, n. 1, 2011.

40 RAMOS, D. B. Controlador nebuloso para motor de ignição por compressão operando com gás natural e óleo diesel. *Santa Maria, RS, UFSM*, 2006.

41 NELLES, O. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. : Springer Science & Business Media, 2013.

42 SANDRI, S.; CORREA, C. Lógica nebulosa. *Instituto Tecnológico da Aeronáutica–ITA, V Escola de Redes Neurais, pp. C073-c090, São José dos Campos*, 1999.

43 SHAW, I. E. S. O.; oES, E. S. Mg controle e modelagem fuzzy. *Edgard Blücher Ltda. Primeira Edição*, 1999.

44 DERIVATION of Fuzzy Control Rules from Human Operator's Control Actions. *IFAC Proceedings Volumes*, v. 16, n. 13, p. 55 – 60, 1983. IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis, Marseille, France, 19-21 July, 1983.

45 JANG, J. S. R.; SUN, C.-T. Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, v. 83, n. 3, p. 378–406, 1995.

46 TEIXEIRA, R. de A.; CORREA, M. V.; ASSIS, J. C. de O. Redes neuro fuzzy: algoritmo de treinamento e aplicações.

47 HUAMANI, I. R. L. et al. Redes neurais fuzzy aplicadas em identificação e controle de sistemas. [sn], 2003.

48 JANG, J. S. R. Anfis: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 23, n. 3, p. 665–685, 1993.

49 XU, R.; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, v. 16, n. 3, p. 645–678, 2005.

50 YAGER, R.; FILEV, D. Generation of fuzzy rules by mountain clustering. v. 2, p. 209–219, 01 1994.

51 CHIU, S. Fuzzy model identification based on cluster estimation. v. 2, p. 267–278, 01 1994.

52 SILVEIRA, T. Maria Andrade da. Modelos de previsão de carga elétrica em curto prazo desenvolvidos com redes neurais artificiais e lógica fuzzy considerando a variável temperatura. Universidade Federal de Pernambuco, 2010.

53 FIRMO, A. J. S.; LUIZ, S. O. D.; LIMA, A. M. N. Determining a processor utilization model based on fuzzy c-means clustering. *Congresso Brasileiro de Automática*, 2018.

54 NARENDRA, K. S.; PARTHASARATHY, K. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, v. 1, n. 1, p. 4–27, 1990.

55 RICHARDSON, I. E. *The H. 264 advanced video compression standard.* : John Wiley & Sons, 2011.

56 BEIGUELMAN, B. *Curso de Bioestatástica Básica. 4ed.* : Sociedade Brasileira de Genética., 1996.

57 HYVARINEN JUHA KARHUNEN, E. O. A. *Independent Component Analysis.* : John Wiley & Sons, 2001.

58 JANTZEN, J. *Tuning of Fuzzy PID Controllers*. 1998.

59 WANG, L.-X.; MENDEL, J. M. Generating fuzzy rules by learning from examples. *IEEE Transactions on systems, man, and cybernetics*, IEEE, v. 22, n. 6, p. 1414–1427, 1992.