Programa de Pós-Graduação em Engenharia Elétrica - PPGEL

Associação ampla UFSJ / CEFET-MG

O Efeito da Precisão Finita na Entropia de Sinais Resultantes de Filtros Digitais IIR

Vinícius da Silva Borges

Orientador: Erivelton Geraldo Nepomuceno

Coorientador: Carlos Augusto Duque

São João del-Rei, 18 de setembro de 2020.

Programa de Pós-Graduação em Engenharia Elétrica - PPGEL

Associação ampla UFSJ / CEFET-MG

O Efeito da Precisão Finita na Entropia de Sinais Resultantes de Filtros Digitais IIR

Vinícius da Silva Borges

Dissertação apresentada à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica, associação ampla entre a Universidade Federal de São João del-Rei e o Centro Federal de Educação Tecnológica de Minas Gerais, como parte dos requisitos necessários à obtenção do grau de Mestre em Engenharia Elétrica.

Área de Concentração: Modelagem e Controle de Sistemas

Linha de Pesquisa: Análise e Modelagem de Sistemas

Grupo de Pesquisa: GCOM - Grupo de Controle e Modelagem

Orientador: Erivelton Geraldo Nepomuceno

Coorientador: Carlos Augusto Duque

São João del-Rei, 18 de setembro de 2020.

Ficha catalográfica elaborada pela Divisão de Biblioteca (DIBIB)
e Núcleo de Tecnologia da Informação (NTINF) da UFSJ,
com os dados fornecidos pelo(a) autor(a)

Borges, Vinicius da Silva .
B732e O Efeito da Precisão Finita na Entropia de Sinais Resultantes de Filtros Digitais IIR / Vinicius da Silva Borges ; orientador Erivelton Geraldo Nepomuceno; coorientador Carlos Augusto Duque. -- São João del-Rei, 2020.
93 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Engenharia Elétrica) -- Universidade Federal de São João del-Rei, 2020.

1. Teoria da Informação. 2. Computação Aritmética. 3. Filtro Digital. 4. Entropia de Shannon. I. Nepomuceno, Erivelton Geraldo , orient. II. Duque, Carlos Augusto , co-orient. III. Título.

Dedico este trabalho à minha querida avó Maria Anísia,

cuja presença foi essencial na minha vida.

AGRADECIMENTOS

A Deus, por me iluminar durante este período.

A minha querida avó, por sempre me apoiar nos meus estudos.

Ao meu orientador professor Erivelton, pelo incentivo, confiança, presteza e contribuições para o desenvolvimento deste trabalho.

Ao meu coorientador Carlos Duque, pelo apoio durante o mestrado.

Aos amigos do GCOM, pela troca de conhecimentos durante esse período.

Aos professores e funcionários do Programa de Pós-Graduação em Engenharia Elétrica.

À UFSJ, GCOM, CEFET-MG, PPGEL, INERGE e FAPEMIG.

Enfim, um agradecimento a todos que contribuíram para conclusão deste trabalho.

“A persistência é o menor caminho do êxito”

Charles Chaplin

SUMÁRIO

RESUMO

A resolução numérica finita da representação digital de números afeta as propriedades dos filtros. Grande número de trabalhos foram dedicados ao desenvolvimento de filtros digitais eficientes, investigando os efeitos na resposta em frequência. No entanto, parece que há menos atenção à influência na entropia pelos sinais filtrados digitais devido à precisão finita. Para contribuir nessa direção, este trabalho apresenta algumas observações sobre a entropia de sinais filtrados. Três tipos de filtros são investigados: Butterworth, Chebyshev e Elíptico. Usando uma técnica de limite, os parâmetros dos filtros são avaliados de acordo com o comprimento da palavra de 16 ou 32 bits. Foi demonstrado que os sinais filtrados têm sua entropia aumentada, mesmo que os filtros sejam lineares. Foi observada correlação positiva significativa ($p < 0{,}05$) entre a ordem e a entropia de Shannon do sinal filtrado usando o filtro elíptico. Comparando com a relação sinal/ruído, a entropia parece mais eficiente na detecção do aumento de ruído em um sinal filtrado. Esse conhecimento pode ser usado como uma condição adicional no projeto de filtros digitais.

Palavras-chave: Teoria da Informação; Computação Aritmética; Filtro Digital; Entropia de Shannon.

ABSTRACT

The finite numerical resolution of digital number representation has an impact on the properties of filters. Much effort has been done to develop efficient digital filters investigating the effects in the frequency response. However, it seems that there is less attention to the influence in the entropy by digital filtered signals due to the finite precision. To contribute in such a direction, this manuscript presents some remarks about the entropy of filtered signals. Three types of filters are investigated: Butterworth, Chebyshev, and elliptic. Using a boundary technique, the parameters of the filters are evaluated according to the word length of 16 or 32 bits. It has been shown that filtered signals have their entropy increased even if the filters are linear. A significant positive correlation ($p < 0.05$) was observed between order and Shannon entropy of the filtered signal using the elliptic filter. Comparing to signal-to-noise ratio, entropy seems more efficient at detecting the increasing of noise in a filtered signal. Such knowledge can be used as an additional condition for designing digital filters.

Keywords: Theory of Information; Computer Arithmetic; Digital Filter; Shannon Entropy.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$x(n)$	Sequência de entrada;
α	Constantes arbitrárias;
$\delta(n)$	Sequência de impulsos;
$h(n)$	Resposta ao impulso;
$\sum$	Somatório;
$y(n)$	Sequência de saída;
$X(n)$	Transformada z de uma sequência de entrada;
ω_c	Frequência de corte;
ω_{c_1}	Limite inferior da banda passante;
ω_{c_2}	Limite superior da banda passante;
α_p	Ondulação da resposta em frequência da banda passante;
α_r	Ondulação da resposta em frequência da faixa de rejeição;
$H(z)$	Função de transferência;
$Y(z)$	Sinal de saída do filtro;
$X(z)$	Sinal de entrada do filtro;
N	Grau do polinômio do numerador;
M	Grau do polinômio do denominador;
a_l	Polos do filtro;
b_k	Zeros do filtro;
ϵ	Fator de ondulação da banda passante;
U_N	Função jacobiana elíptica;
p	Precisão do sistema;
E	Expoente;
S	Mantissa;
B	Base;
ε	Épsilon da máquina;

Ulp	Unidade da última posição;
N_{min}	Menor número positivo;
N_{max}	Maior número positivo;
E_{min}	Expoente mínimo assegurado;
E_{max}	Expoente máximo assegurado;;
x	Número ponto flutuante;
R	Conjunto dos números reais;
x_{inf}	Número arredondado para baixo;
x_{sup}	Número arredondado para cima;
$>$	Maior que;
$<$	Menor que;
$=$	Igual à;
W	Grau de liberdade na banda;
$Y(f)$	Filtro;
$H(X)$	Entropia de Shannon (bits);
X	Símbolo;
L	Tamanho da mensagem;
P_i	Probabilidade;
WL	Comprimento da palavra;
sin	Seno;
t	Tempo;
H_{sinal}	Entropia do sinal;
$H_{ruído}$	Entropia do ruído;

LISTA DE ABREVIAÇÕES

UFSJ	Universidade Federal de São João del-Rei;
GCOM	Grupo de Controle e Modelagem;
CEFET-MG	Centro Federal de Educação Tecnológica de Minas Gerais;
PPGEL	Programa de Pós-Graduação em Engenharia Elétrica;
INERGE	Instituto Nacional de Ciência e Tecnologia em Energia Elétrica;
FAPEMIG	Fundação de Amparo à Pesquisa do Estado de Minas Gerais;
IEEE	Institute of Electrical and Electronic Engineers;
ANSI	American National Standards Institute;
ISO	International Organization for Standardization;
IEC	International Electrotechnical Commission;
Bit	Binary Digit

CAPÍTULO 1

INTRODUÇÃO

Os filtros digitais são mapas de tempo discreto que realizam operações matemáticas em um sinal amostrado [1]. A resposta de frequência é geralmente aplicada para caracterizar filtros [2, 3]. Duas classes principais de filtros digitais são geralmente usadas. Quando a resposta ao impulso não é zero para um número finito de amostras, temos os filtros de resposta de impulso finita (FIR). No caso em que a resposta ao impulso produz um número infinito de amostras diferentes de zero, temos a resposta ao impulso infinita (IIR) [4, 5].

O processo de filtragem digital é amplamente utilizado em muitas aplicações em comunicações, processamento de sinais, engenharia elétrica, biomédica e controle [6–14]; por exemplo, codificação e compressão, aumento de sinal, *denoising*, demodulação de amplitude e frequência, conversões analógico-digitais, detecção de forma e extração [15–24]. Para algumas aplicações, a não linearidade é adaptada a uma finalidade específica [25]. Recentemente, os autores de [26] projetaram um filtro de resposta de impulso infinito truncado sigma-delta digital, que fornece rejeição adequada com um conversor digital-analógico de não mais que 8 bits. A aplicação em [26] está relacionada à comunicação do corpo humano, que para muitos pesquisadores é um tópico de pesquisa promissor, pois desempenha um papel importante nas redes da área do corpo sem fio devido ao seu baixo custo de energia e hardware. Nesta área, parece que os filtros digitais de tamanho médio a baixo atraíram novamente a atenção dos pesquisadores [27, 28].

Quando filtros digitais são empregados em plataformas aritméticas de ponto fixo, por exemplo, microcontroladores, DSP e FPGA, ou com especificações de desempenho muito exigentes, a importância da precisão do coeficiente de filtro aumenta, porque a resposta em frequência do projeto do filtro pode ser modificada de forma significativa [29–31]. Assim, um objetivo comum na análise de precisão finita é escolher um comprimento de palavra para que o sistema digital apresente uma realização suficientemente precisa. Esse formato deve considerar a complexidade e o custo do hardware e software [32].

No processamento de sinal digital, os problemas de comprimento finito de palavras são alguns dos componentes mais significativos quando os pólos discretos estão muito próximos do

círculo unitário. Mullis e Roberts [33] e Hwang [34] demonstraram que a influência dos erros de quantização no desempenho do filtro digital depende da implementação do filtro. Além disso, Rader e Gold [35] mostraram que, para uma determinada implementação de filtro, é possível que pequenos erros nos coeficientes do denominador ou numerador possam causar deslocamentos dos polos e zeros, a ponto de tornar o filtro instável. E ainda, Goodall e Donoghue [36] e Jones et al. [37] observaram uma sensibilidade significativa dos comprimentos dos coeficientes de palavras. Esse fato está relacionado à incapacidade dos computadores de representar a natureza infinita dos conjuntos reais [38]. A influência das limitações do computador abre uma nova perspectiva para a simulação do ambiente do computador. Por exemplo, Nepomuceno [39] apresenta um teorema que identifica a confiabilidade dos cálculos realizados em ponto fixo; em [40, 41] uma técnica foi desenvolvida para recusar uma simulação se uma precisão obrigatória for maior que o erro do limite inferior, aumentando a confiabilidade numérica na simulação e ainda em [42] os autores mostram como é sensível um sistema simulado em diferentes processadores.

Parece claro que muita pesquisa foi dedicada à investigação da influência da precisão finita nos filtros digitais [33, 35, 37, 43, 44]. Nessas investigações, há muitos casos em que a qualidade do filtro é medida usando a resposta do filtro ou a relação sinal-ruído (SNR) [44]. Apesar do fato de o efeito dos filtros na entropia ter sido apontado desde o trabalho de Shannon [45] há menos atenção aos efeitos da precisão finita na entropia dos sinais resultantes de filtros digitais. Um trabalho nessa direção foi realizado por Badii et al. [46] que mostraram a influência de uma resposta de impulso infinito na dimensão fractal do atrator reconstruído a partir de um sinal caótico filtrado. Outros trabalhos empregaram entropia para projetar filtros digitais. Por exemplo, Madan [47] introduziu o uso do método da entropia máxima para o projeto de filtros digitais FIR de fase linear. Em [48] outra tentativa de usar entropia para o projeto de filtros digitais FIR foi observada. No entanto, nenhum trabalho foi encontrado investigando os efeitos na entropia em um sinal filtrado por um filtro IIR. Esta dissertação procura relacionar as limitações computacionais e a variação dos principais parâmetros de um filtro na entropia medida. Como a entropia é um bom índice para detectar aumento de ruído em um sinal, foi utilizada a técnica de contorno para observar os efeitos da precisão finita nos parâmetros dos filtros de acordo com o comprimento da palavra de 16 ou 32 bits. Percebe-se que a entropia é mais sensível que o SNR. Foi importante mostrar que, apesar do filtro linear ideal não aumentar a entropia, experimentos

numéricos usando os filtros elíptico, Butterworth e Chebyshev mostraram um aumento do nível da entropia. Além disso, uma correlação positiva entre ordem e entropia foi observada no filtro elíptico. Essas informações podem ser úteis para projetar ou avaliar filtros digitais em situações em que o ruído de crescimento deve ser mitigado.

1.1 Descrição do Problema

Filtros digitais práticos devem ser implementados com números e aritmética de precisão finita. Como resultado, os coeficientes e os sinais de entrada e saída do filtro estão em forma discretas. Portanto, erros e restrições devido ao tamanho finito das palavras são inevitáveis. Com isso, alguns tipos de efeitos finitos de comprimento de palavra são observados. A quantização dos coeficientes do sistema, erros devido aos arredondamentos na aritmética computacional e uma restrição no nível do sinal, com uma maior exigência de armazenamento [49].

A quantização dos coeficientes do filtro tem o efeito de perturbar a localização dos pólos e zeros do filtro [50]. Como resultado, a resposta real do filtro difere um pouco da resposta de projeto. Esse erro de resposta em frequência é chamado de erro de quantização do coeficiente. Além disso, a quantização dos cálculos do filtro também torna o filtro levemente não linear. Para sinais grandes, essa não linearidade é desprezível e o ruído de arredondamento é a principal preocupação. No entanto, para filtros recursivos com entrada zero ou constante, essa não linearidade pode causar oscilações espúrias chamadas ciclos limite [51].

O uso da aritmética de precisão finita torna necessário quantificar os cálculos de filtro por arredondamento ou truncamento. O ruído de arredondamento é o erro na saída do filtro resultante de cálculos arredondados ou truncados no filtro. Como o nome indica, esse erro parece ruído de baixo nível na saída do filtro. O ruído de arredondamento é potencialmente maior que o ruído de quantização, e ainda, esse erro é um dos principais fatores que determinam a complexidade da implementação do filtro digital [52].

Com a aritmética de ponto fixo é possível que os cálculos do filtro ultrapassem o limite superior do formato da máquina, dessa forma, os resultados são considerados infinito e essa ocorrência é chamada de *overflow*. Este fenômeno ocorre quando os módulos dos sinais internos excedem a faixa dinâmica dos registradores disponíveis, o que, se ocorrer frequentemente em um curto período de tempo, pode acarretar severas distorções no sinal de saída do filtro, iniciando oscilações auto sustentadas denominadas ciclos limite devidos a *overflow* [53].

Esses efeitos de quantização devem ser considerados, tanto na decisão sobre o tamanho do registro necessário para uma determinada implementação quanto na escolha entre várias implementações possíveis do filtro, que serão afetadas diferentemente pela quantização. Uma abordagem para realizações eficientes é considerar a quantização como uma fonte de ruído [54, 55]. Com isso, realizar o cálculo da entropia, visto que, a sua sensibilidade em ruído é mais eficiente à medida que a resolução da especificação do filtro aumenta e ainda o projetista tem a capacidade de relacionar a incerteza com diferentes parâmetros do filtro e do sinal de entrada. O uso da entropia em filtros FIR possibilitou aumentar a ordem de um filtro de forma direta enquanto diminui a entropia. Essa técnica resultou em filtros de ordem superior que tenham melhores características em termos de potência do ruído de quantização da saída [56]. Neste caso, espera-se que os resultados desta dissertação possibilitem o desenvolvimento de filtros IIR através do cálculo da entropia.

1.2 Objetivos

1.2.1 Objetivo Geral

Investigar os efeitos na entropia de um sinal filtrado por um filtro IIR.

1.2.2 Objetivos Específicos

- Implementar diferentes filtros com características Butterworth, Chebyshev e Eliptico;
- Estudo da influência do erro de quantização na entropia medida de filtros digitais;
- Verificar a complexidade e o comportamento da entropia com o aumento da ordem do filtro;
- Comparação entre a entropia de um filtro ideal e a entropia de um filtro implementado.

1.3 Contribuições da Dissertação

Em seguida, são apresentados os trabalhos publicados, frutos dos estudos ao longo do desenvolvimento desta dissertação:

- [57] - V. S. Borges, E. G. Nepomuceno, C. A. Duque, e D. N. Butusov, “Some remarks about entropy of digital filtered signals,” *Entropy*, vol. 22, no. 3, p. 365, 2020

- [58] - V. Borges, E. G. Nepomuceno, A. V. Tutueva, A. I. Karimov, C. Duque, e T. I. Karimov, "Analysis of iir filters by interval response," in *2020 Moscow Workshop on Electronic and Networking Technologies (MWENT)*. IEEE, 2020, pp. 1–5

Em [57], os autores apresentaram algumas observações sobre a entropia de sinais filtrados. Três tipos de filtros são investigados: Butterworth, Chebyshev e elíptico. Usando uma técnica de perturbação dos parâmetros dos filtros são avaliados de acordo com o comprimento da palavra de 16 ou 32 bits. Além disso, os resultados mostram que os sinais filtrados têm sua entropia aumentada, mesmo que os filtros sejam lineares. E ainda, foi observada correlação positiva significativa entre a ordem e a entropia de Shannon do sinal filtrado usando o filtro elíptico, evidenciando que o aumento de coeficientes, consequentemente de operações aritméticas, insere incertezas no processo de filtragem. Comparando com a relação sinal/ruído, a entropia parece mais eficiente na detecção do aumento de ruído em um sinal filtrado. Esse conhecimento pode ser usado como uma condição adicional para projetar filtros digitais.

No artigo [58], foi proposto uma nova abordagem, chamada de abordagem por tamanho de intervalo, facilitando a avaliação da resposta real do filtro digital para vários parâmetros aritméticos de ponto fixo. Forneceram-se exemplos ilustrativos para demonstrar a evolução dos limites da resposta em frequência, conforme a ordem, a frequência de corte e a frequência do sinal quando variados. Sendo assim, mostrou-se que o tamanho do intervalo pode ser usado como um fator de precisão adequado de um filtro digital.

1.4 Organização do Trabalho

Esta dissertação está dividida em cinco capítulos. Após este capítulo de Introdução, o Capítulo 2 apresenta de forma sucinta os conceitos essenciais para compreensão da dissertação. Inicialmente são apresentados conceitos básicos de filtros e, além disso, também são abordadas algumas definições da computação aritmética, com maior detalhamento do padrão IEEE 754-2019 para aritmética de ponto flutuante [38]. Os principais aspectos, propriedades e ferramentas do cálculo e análise da entropia. O Capítulo 3 traz as metodologias empregadas para a realização desta dissertação. O que se pretende nesse capítulo é realizar uma fusão entre os tópicos abordados na publicação realizada. No Capítulo 4 são apresentados os resultados da aplicação das metodologias desenvolvidas. Finalmente, no Capítulo 5 são apresentadas as considerações

finais do trabalho desenvolvido e as perspectivas de trabalhos futuros.

CAPÍTULO 2

CONCEITOS PRELIMINARES

Nesta seção, os filtros de resposta ao impulso infinito (IIR) são abordados. Nas implementações com aritmética finita é possível reduzir o ruído de quantificação inerente através de implementações não recursivas. Os filtros IIR são utilizados em aplicações onde as características de fase lineares não constituem um motivo de preocupação. Ainda será discutida a entropia, com uma breve bibliografia de Shannon com explanação das suas principais realizações e a explicação da entropia da informação. Além disso, a representação numérica será discutida em sistemas computacionais, abordando-se as noções da representação dos números no computador, precisão, épsilon da máquina, ulp e representação IEEE e por fim, arredondamento. Em seguida, o efeito da quantização e o processamento de sinais com o Octave.

2.1 Filtros Digitais

Na esfera tecnológica da atualidade, existe a necessidade de medir grandezas que variam com o tempo. A pressão arterial, a luminosidade de uma estrela, a amplitude de vibração de um terramoto, o fluxo de um líquido num tubo e a pressão acústica de sinais sonoros são alguns exemplos de grandezas que variam com o tempo. Para que estes dados possam ser tratados pelo computador existe a necessidade de converter o sinal em uma componente digital, isto é, a função que varia no tempo que descreve uma grandeza deve ser amostrada. Assim, os dados tratados serão finitos e não infinitos. Dessa forma, resultados ou comportamentos indesejados devido a esta limitação é uma rotina na vida de engenheiros, pesquisadores ou programadores. Uma alternativa para melhorar a qualidade da amostragem está na modificação do número de bits para a representação [59–61].

Sendo assim, uma vez no domínio discreto surge possibilidades de aplicações para o sinal amostrado. Algumas delas são a remoção de ruído, o destaque ou a separação de informações de faixas de frequência diferentes [62], na recuperação de sinais, por exemplo, recuperação de gravações de áudio e melhoramento de imagens [63]. Tais funcionalidades são oriundas dos filtros digitais, circuitos eletrônicos utilizados para processamento de sinal. Eles funcionam como

seletores de frequência, pois através de sua configuração são capazes de eliminar a frequência indesejada presente no sinal. Além disso, os filtros digitais são compostos de multiplexadores, somadores e também podem ser configurados através de lógicas de componentes eletrônicos, como microcontroladores e FPGAs [64, 65].

Para um filtro digital ideal somente as especificações de frequências limites das faixas de passagem seriam necessárias. Neste caso, componentes do sinal processado dentro destas faixas seriam transferidos da entrada para a saída sem nenhuma perda de informação, enquanto as componentes fora desta faixa seriam completamente suprimidas. As principais classes de filtros digitais são os filtros lineares, invariantes no tempo (LTI) e causais. Um filtro digital linear responde a uma soma ponderada de sinais de entrada com a mesma soma ponderada das respostas individuais correspondentes; isso é:

$$F[\alpha_1 x_1(n) + \alpha_2 x_2(n)] = \alpha_1 F[x_1(n)] + \alpha_2 F[x_2(n)],$$

para quaisquer sequencias $x_1(n)$ e $x_2(n)$ e para quaisquer constantes arbitrárias α_1 e α_2. Um filtro digital é considerado invariante no tempo quando sua resposta a uma sequência de entrada permanece a mesma, independentemente do instante em que a entrada é aplicada ao filtro. Sendo,

$$F[x(n - n_0)] = y(n - n_0),$$

para todos os valores n e n_0. Um filtro digital causal é aquele cuja resposta não antecipa o comportamento do sinal de excitação, para quaisquer duas sequências de entrada $x_1(n)$ e $x_2(n)$ de tal modo que $x_1(n) = x_2(n)$ para $n \leq n_0$, as respostas correspondentes do filtro digital são idênticas para $n \leq n_0$; isto é:

$$F[x_1(n)] = F[x_2(n)]. \tag{2.1}$$

Um filtro digital invariante de tempo linear inicialmente em repouso é caracterizado por sua resposta à amostra da unidade ou sequência de impulsos $\delta(n)$. A resposta do filtro, quando excitada por essa sequência, é indicada por $h(n)$ e é referido como resposta ao impulso do filtro digital. Observe que, se o filtro digital for causal, então $h(n) = 0$ para $n < 0$. Uma sequência de entrada arbitrária pode ser expressa como uma soma das sequências de pulso atrasadas e

ponderadas:

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k),$$

e a resposta de um filtro digital LTI para $x(n)$ pode então ser expressa por:

$$\begin{aligned} y(n) &= F\left[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right] \\ &= \sum_{k=-\infty}^{\infty} x(k)F[\delta(n-k)] \\ &= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \\ &= x(n) \quad h(n). \end{aligned} \tag{2.2}$$

O somatório nas duas últimas linhas da expressão acima, chamado soma de convolução representado pelo símbolo " ", relaciona a sequência de saída de um filtro digital à sua resposta de impulso $h(n)$ e à sequência de entrada $x(n)$. A Equação (2.2) pode ser reescrita como:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k). \tag{2.3}$$

A definição da transformação z de uma sequência $x(n)$ é escrita como:

$$X(n) = \sum_{n=-\infty}^{\infty} X(n)z^{-n}. \tag{2.4}$$

A função de transferência de um filtro digital é a razão da transformada z da sequência de saída pela transformada z do sinal de entrada:

$$H(z) = \frac{Y(z)}{X(z)}. \tag{2.5}$$

Tomando a transformação z de ambos os lados da expressão de convolução da Equação (2.3), obtém-se:

$$\sum_{n=-\infty}^{\infty} y(n)z^{-n} = \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} h(k)x(n-k)z^{-n} \tag{2.6}$$

substituindo, $(l = n-k)$, e reescrevendo a equação:

$$\sum_{k=-\infty}^{\infty} y(n)z^{-n} = \sum_{k=-\infty}^{\infty} h(k)z^{-k} \sum_{l=-\infty}^{\infty} x(l)z^{(l)}. \tag{2.7}$$

A seguinte relação entre as transformadas z da saída $Y(z)$, da entrada $X(z)$ e da resposta ao impulso $H(z)$ de um filtro digital é obtida como:

$$Y(z) = H(z)X(z). \tag{2.8}$$

Portanto, a função de transferência de um filtro digital LTI é a transformada z de sua resposta ao impulso.

2.1.1 Resposta em Frequência

A resposta em frequência de um sistema de tempo discreto é definida como a transformada de Fourier de tempo discreto (DTFT) de sua resposta ao impulso $h[n]$. Se uma entrada $x[n]$ for aplicada ao sistema, a saída será a soma da convolução $y[n] = x[n] \quad h[n]$. Com base na propriedade da DTFT de somas de convolução pode-se reescrever a Equação (2.8), como

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega}). \tag{2.9}$$

Se as DTFTs são expressas em termos de magnitude e fase, $Y(e^{j\omega}) = |Y(e^{j\omega})|e^{j\phi_Y(\omega)}$, $X(e^{j\omega}) = |X(e^{j\omega})|e^{j\phi_X(\omega)}$ e $H(e^{j\omega}) = |H(e^{j\omega})|e^{j\phi_H(\omega)}$. Então a Equação (2.9) pode ser reescrita

$$|Y(e^{j\omega})| = |H(e^{j\omega})|.|X(e^{j\omega})|, \tag{2.10}$$

e ainda

$$\phi_Y(\omega) = \phi_H(\omega) + \phi_X(\omega). \tag{2.11}$$

$|H(e^{j\omega})|$ e $\phi_H(\omega)$ são referidos como a resposta em magnitude e a resposta de fase do sistema, respectivamente. Nas Equações (2.10) e (2.11), a resposta da magnitude do sistema representa o ganho ou atenuação que o conteúdo do sinal de entrada na frequência, ω, experimenta quando o sistema o mapeia para a saída. A resposta de fase do sistema representa o deslocamento de fase correspondente na frequência.

Se a resposta da magnitude do sistema, $|H(e^{j\omega})|$, é uma unidade, então diz-se que o sistema passa idealmente o conteúdo do sinal de entrada em frequência, ω_0, para a saída. O conjunto de todas essas frequências é denominado banda passante do sistema. Se $|H(e^{j\omega})|$ for zero, então

o sistema atenua idealmente o conteúdo do sinal de entrada em frequência, ω_0. O conjunto de todas essas frequências é denominado de banda de rejeição.

Se o conjunto de todas as frequências $|\omega| \leq \pi$ é particionado em bandas de passagem e bandas de rejeição, então o sistema é chamado de filtro ideal que passa ou atenua o conteúdo do sinal de uma maneira seletiva de frequência. Por exemplo, um filtro é considerado passa-baixa ideal se sua resposta de frequência for a seguinte:

$$H(e^{j\omega}) = \begin{cases} 1, & |\omega| < \omega_c \\ 0, & |\omega_c| < |\omega| \leq \pi. \end{cases} \tag{2.12}$$

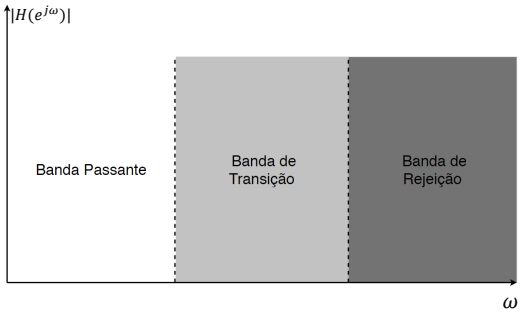
O filtro passa todo o conteúdo de entrada de baixa frequência na banda passante, $|\omega| < \omega_c$, e interrompe todo o conteúdo de entrada de alta frequência na banda de rejeição, $|\omega_c| < |\omega| \leq \pi$.

2.1.2 Classificação dos Filtros

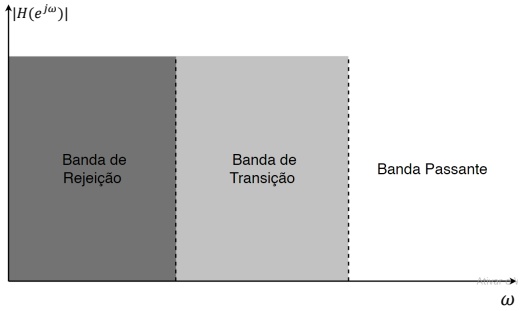
Os filtros podem ser classificados em quatro categorias: passa-baixa, passa-alta, passa-faixa e rejeita-faixa. Cada tipo tem uma aplicação específica no processamento digital de sinal. De um modo geral, um filtro é projetado tendo como base as especificações da banda passante, intervalo de frequência em que ganho em amplitude do filtro atinge aproximadamente o valor de uma unidade, e de rejeição, definida como o intervalo de frequência sobre a qual a resposta de frequência do filtro é atenuada de forma a eliminar o sinal de entrada. Entre a banda passante e a banda de rejeição esta a banda de transição, dentro das quais não há passagem nem rejeição significativas das componentes de frequência. Uma filtragem mais efetiva ocorre em filtros com regiões de transição muito estreitas, algo que ocorre com filtros de ordem superior. Na Fig. 2.1 é ilustrado a distribuição das três faixas para as quatro categorias de filtros.

As especificações do filtro passa-baixa permite a passagem de baixas frequências e atenua as altas. Como é apresentado na Fig. 2.2, ω_c é a frequência de corte, α_p especifica o nível de ondulação da resposta em frequência da banda passante, enquanto o α_r especifica a ondulação da resposta em frequência da faixa de rejeição.

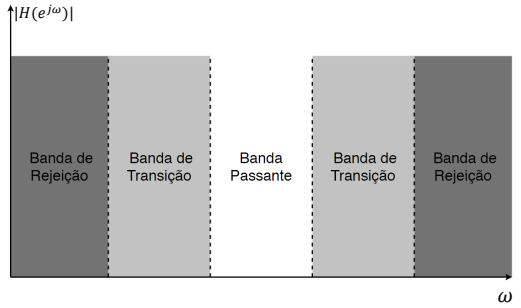
(a)



(b)



(c)



(d)

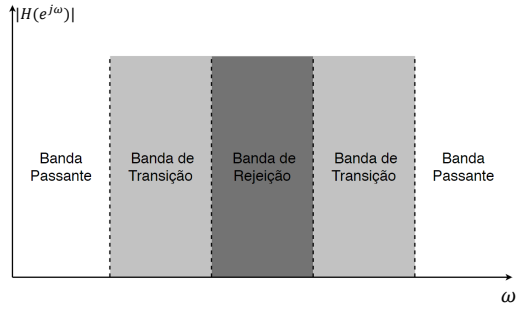


Figura 2.1: **a)** Filtro passa-baixa. **b)** Filtro passa-alta. **c)** Filtro passa-faixa. **d)** Filtro rejeita-faixa. Nas quatro imagens são demostrados o comportamento das bandas de passagem, rejeição e transição. Em filtros ideais a banda de transição é completamente atenuada, tal fato, não é percebido em condições reais. Além disso, uma filtragem mais efetiva ocorre em filtros com regiões de transição estreitas, aumentando a ordem do mesmo. Este comportamento foi alvo de estudo nesta dissertação, com a relação do incremento do valor da ordem e a entropia.

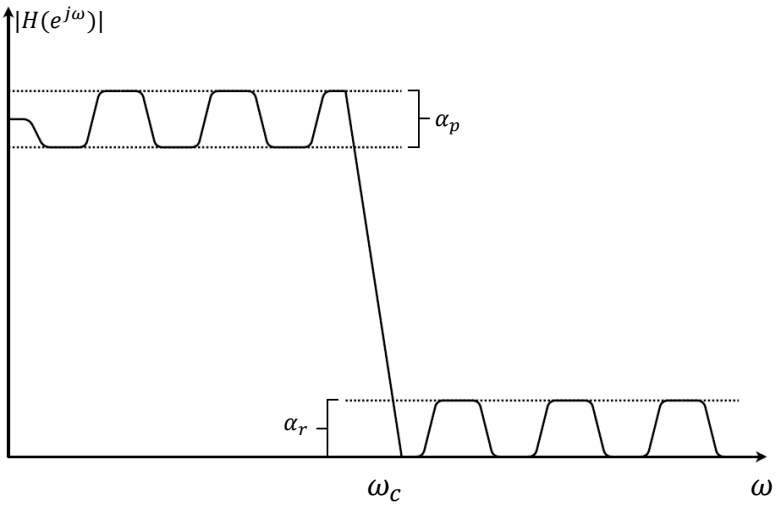


Figura 2.2: Resposta em frequência de um filtro passa-baixo. Sendo ω_c a frequência de corte, α_p a ondulação da banda passante, enquanto o α_r especifica a ondulação da resposta em frequência da faixa de rejeição.

O filtro passa-alta funciona de maneira inversa ao filtro passa-baixa, isto é, permite a passagem de altas frequências e atenua a amplitude das frequências menores. A resposta em frequência para o filtro passa-alta pode ser visualizada na Fig. 2.3.

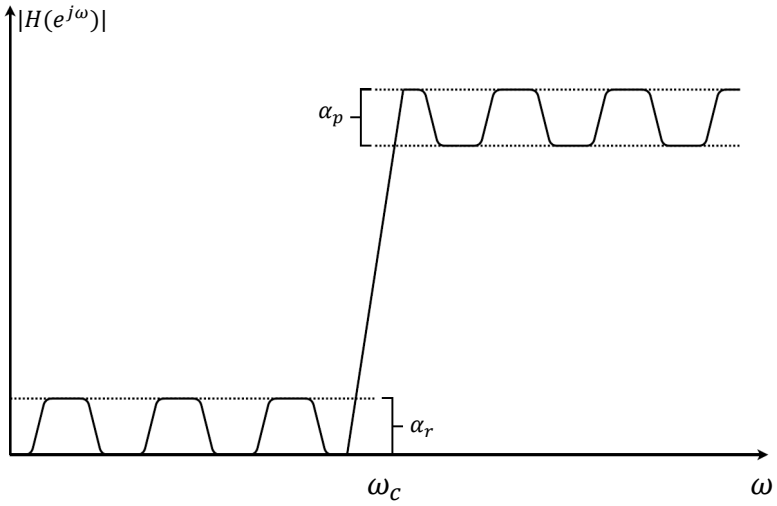


Figura 2.3: Resposta em frequência de um filtro passa-alta. Sendo ω_c a frequência de corte, α_p a ondulação da banda passante, enquanto o α_r especifica a ondulação da resposta em frequência da faixa de rejeição.

O filtro passa-banda, demostrado na Fig. 2.4, permite a passagem de frequências de uma determinada faixa, atenuando as baixas e altas frequências. Como é apresentado na Fig. 2.4, ω_{c_1} e ω_{c_2} são os limites inferiores e superiores da banda passante; respectivamente; α_p é o

parâmetro para especifica a ondulação da resposta em frequência da faixa passante, e o α_r especifica a ondulação da resposta em frequência da faixa de rejeição.

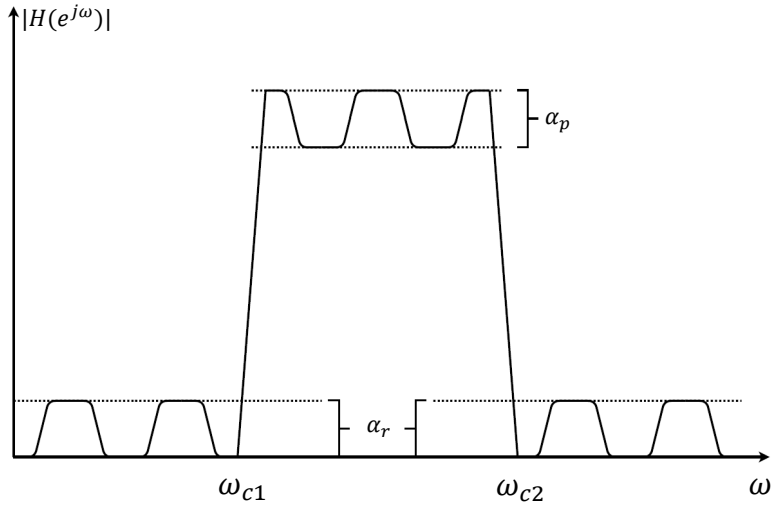


Figura 2.4: Resposta em frequência de um filtro passa-faixa. Sendo ω_{c_1} e ω_{c_2} são os limites inferiores e superiores da banda passante; respectivamente; α_p é o parâmetro para especificar a ondulação da resposta em frequência da faixa passante, e o α_r especifica a ondulação da resposta em frequência da faixa de rejeição.

Por último, como pode ser visualizado na Fig. 2.6, o filtro rejeição de banda atenua as frequências de uma determinada faixa, permitindo as baixas e altas frequências.

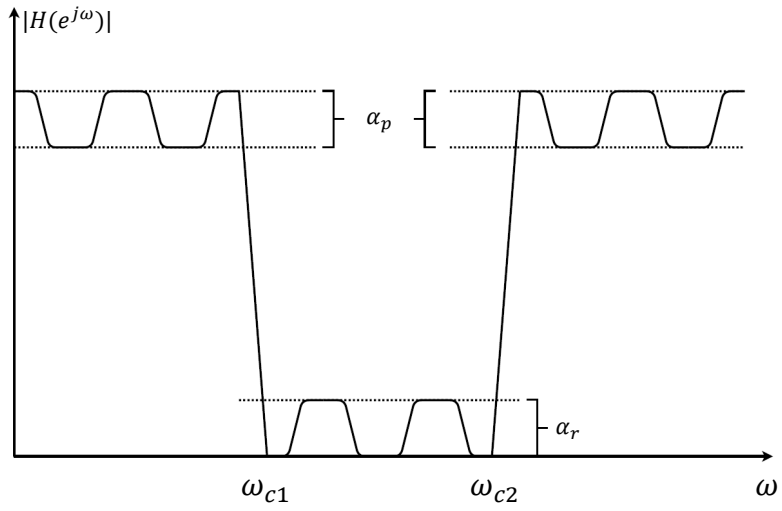


Figura 2.5: Resposta em frequência de um filtro rejeita-faixa. Sendo ω_{c_1} e ω_{c_2} são os limites inferiores e superiores da banda de rejeição; respectivamente; α_p é o parâmetro para especificar a ondulação da resposta em frequência da faixa passante, e o α_r especifica a ondulação da resposta em frequência da faixa de rejeição.

2.1.3 Filtros IIR

Os filtros digitais IIR são caracterizados por ter resposta infinita ao impulso [66]. Eles têm realimentação de saída, o que os torna interessantes porque permitem obter uma resposta de frequência mais seletiva com menor número de coeficientes. Dessa forma, considerando a função de transferência de um filtro digital LTI sendo a transformada z de sua resposta ao impulso e reescrevendo a Equação (2.8),

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{l=0}^{N} a_l z^{-l}} \tag{2.13}$$

$$\sum_{l=0}^{N} a_l z^{-l} Y(z) = \sum_{k=0}^{M} b_k z^{-k} X(z), \tag{2.14}$$

em que N e M são o grau do polinômio do numerador e do denominador, respectivamente; b_k e a_l são os coeficientes do filtro, e por fim $Y(z)$ e $X(z)$ as transformadas z da saída e da entrada. Para encontrar a equação da diferença do filtro, é feita a transformação z inversa de cada lado da Equação (2.14). Obtendo o resultado:

$$\sum_{l=0}^{N} a_l y[n-l] = \sum_{k=0}^{M} b_k x[n-k], \tag{2.15}$$

sendo $y[n-l]$ e $x[n-k]$ a saída e a entrada do sinal. Uma forma mais condensada da equação da diferença é:

$$y[n] = \frac{1}{a_0}\left(\sum_{k=0}^{M} b_k x[n-k] - \sum_{l=1}^{N} a_1 y[n-l]\right). \tag{2.16}$$

Por fim, tomando $a_0 = 1$, temos:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{l=1}^{N} a_1 y[n-l]. \tag{2.17}$$

2.1.4 Projeto de Filtros IIR

As funções de transferência pertencentes aos filtros eletrônicos analógicos IIR foram extensivamente estudadas e otimizadas por suas características de amplitude e fase. As soluções desejadas podem ser transferidas para o caso de filtros de tempo discreto cujas funções de trans-

ferência são expressas no domínio z, como foi visto nas Equações (2.4)-(2.8). Dessa forma, o uso de técnicas matemáticas, como o método de transformação bilinear ou invariância de impulso possibilita que os filtros IIR digitais sejam baseados em soluções conhecidas para filtros analógicos, como o filtro Butterworth, Chebyshev e Elíptico, herdando as características dessas soluções.

O filtro de processamento de sinal que apresenta resposta de frequência plana na banda passante pode ser denominado como filtro Butterworth. Em 1930, o físico e engenheiro britânico Stephen Butterworth descreveu o filtro Butterworth em seu artigo *"On the theory of filter amplifiers"* [67] pela primeira vez.

O filtro Butterworth tem resposta de frequência o mais plana possível matematicamente, portanto, também é chamado de filtro de magnitude máxima plana (de $0Hz$ à frequência de corte a $-3dB$ sem ondulações). O filtro Butterworth muda de banda passante para a banda de rejeição, alcançando o nivelamento da banda passante às custas da banda de transição amplas e é considerado a sua principal desvantagem. A única maneira de diminuir a banda de transição é aumentando o número de pólos do filtro, o que implica no aumento da ordem do filtro [68]. Por fim, esse filtro é usado em aplicações de conversão de dados como um filtro anti-aliasing devido à sua natureza de banda de passagem plana. Além disso, outras aplicações são em radares, como no projeto da exibição da faixa de alvo do radar, em de áudio de alta qualidade.

O nome dos filtros Chebyshev recebeu o nome de Pafnuty Chebyshev porque suas características matemáticas são derivadas dos polinômios de Chebyshev. Um dos seus principais aspectos é a banda de transição estreita. Dessa forma, a principal característica deste filtro é sua velocidade, isso implica no custo da ondulação em uma das bandas de resposta em frequência. Sendo assim, os filtros Chebyshev possuem uma banda de rejeição monotônica e outra banda com ondulação, que pode ser diminuído com o aumento da ordem, ou na diminuição da banda de transição. A capacidade de manipular a ondulação e o tamanho da banda de transição sem alterar a ordem, tornam os filtros Chebyshev mais flexíveis que os filtros Buttherworth. Entretanto, não é possível especificar todos os parâmetros ao mesmo tempo e, portanto, o projetista deve escolher entre fixar a ondulação ou fixar a ordem do filtro [61].

O último tipo de filtro utilizado foi o Elíptico. Como características esses filtros são os que possuem a menor banda de transição, como consequência respostas mais rápidas entre as bandas de passagem e rejeição. O custo disso é a ocorrência de ondulações nas duas bandas de

resposta em frequência. Este tipo de filtro apresenta resposta em fase fortemente não linear, o que dificulta no processo de equalização para aplicações que necessitam de fases lineares [69].

2.2 Entropia

Na mecânica estatística, a entropia é uma propriedade extensiva de um sistema termodinâmico. Quantifica o número de configurações microscópicas, conhecidas como microestados, que são consistentes com as quantidades macroscópicas que caracterizam o sistema, como volume, pressão e temperatura [70]. Por ser determinada pelo número de microestados aleatórios, a entropia está relacionada à quantidade de informações adicionais necessárias para especificar o estado físico exato de um sistema, dada sua especificação macroscópica. Por esse motivo, costuma-se dizer que a entropia é uma expressão da aleatoriedade de um sistema, ou da falta de informações sobre ele. Quando vista em termos da teoria da informação, a função de estado de entropia é simplesmente a quantidade de informação, no sentido de Shannon, que seria necessária para especificar o microestado completo do sistema. Isso não é especificado pela descrição macroscópica.

2.2.1 Quem foi Shannon?

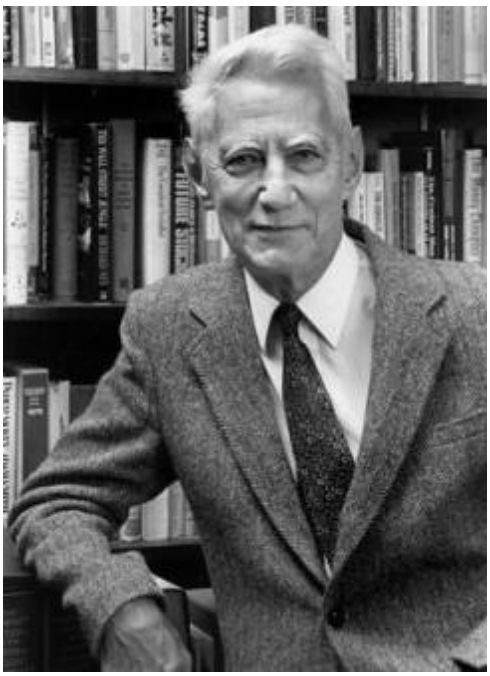


Figura 2.6: Claude Elwood Shannon [71].

Claude Elwood Shannon nasceu em 1916 em Michigan, EUA, e cresceu na pequena cidade de Gaylord. Ele era uma criança curiosa, inventiva, brincalhona e especialmente apaixonado por quebra-cabeças intelectuais, enigmas, criptogramas e malabarismo. Além disso, Shannon construiu modelos com controle remoto e montou seu próprio sistema de telégrafo. Ele entrou na universidade de Michigan aos 16 anos, onde estudou engenharia elétrica e matemática. Mais tarde, ele descreveria sua teoria da informação como "a mais matemática das ciências da engenharia"[72].

Shannon se formou em 1936. Ele encontrou um estágio no MIT como programador assistente do “analisador diferencial” - uma máquina analógica para resolver equações diferenciais de segunda ordem - sob a supervisão de Vannevar Bush, que se tornaria seu mentor. Interruptores de relé controlavam a máquina, o que leva Shannon a um estudo sistemático dos circuitos de relé. Usando seu conhecimento matemático, ele estabeleceu a ligação entre a álgebra booleana. Com apenas 21 anos, sua tese de mestrado [73] revolucionou o uso de circuitos lógicos ao fundar a teoria do projeto de circuitos digitais. Foi descrito como "possivelmente a tese de mestrado mais importante e também a mais famosa do século"[74]. Pelo trabalho de seu mestre, Shannon recebeu o prêmio Alfred Nobel em 1940, concedido pela Sociedade Americana de Engenheiros Civis. No entanto, o principal trabalho de Shannon ainda estava por vir, teoria da informação.

A tese de doutorado de Shannon [75], defendida no MIT em 1940, desenvolve uma álgebra aplicada à genética. Neste mesmo ano, Shannon tornou-se pesquisador nacional no Instituto de Estudos Avançados de Princeton, Nova Jersey. Em Princeton, Shannon teve a oportunidade de discutir suas ideias com cientistas influentes e matemáticos como Hermann Weyl e John von Neumann. Logo após, ele foi trabalhar para os laboratórios de telefonia da Bell. Shannon leu os trabalhos de Harry Nyquist [76] e Ralph Hartley [77], publicados no final da década de 1920 no *Bell System Technical Journal*, o jornal de pesquisa especializada dos Laboratórios Bell. Seus trabalhos terão uma influência decisiva na teoria da informação de Shannon.

Shannon publicou *A Mathematics The Communication of Communication*, em duas partes nas edições de julho e outubro do jornal técnico da Bell System [45]. A teoria é apresentada em forma completa neste único artigo. Resolve inteiramente os problemas de compressão e transmissão de dados, fornecendo os limites fundamentais de desempenho. Pela primeira vez, está provado que para uma comunicação confiável deve ser essencialmente digital. E ainda, em

1948 o artigo é republicado como um livro [78]. Em 1956, Shannon ingressou na faculdade do MIT para trabalhar no Laboratório de Pesquisa em Eletrônica. Por fim, ele desenvolveu a doença de Alzheimer e passou os últimos anos de sua vida em um lar de idosos. Shannon morreu em 2001.

2.2.2 Conceito de Entropia

A entropia reflete uma relação direta entre o comprimento da informação e sua incerteza. Como a entropia quantifica eventos probabilísticos e repetitivos, é utilizada geralmente em diferentes campos [79]. O amadurecimento da ideia de entropia de variáveis e processos aleatórios por Claude Shannon forneceu as origens da teoria da informação. De fato, o primeiro nome de Shannon para esse conceito era *incerteza* e esse foi o motivo de muitos definirem entropia como "medida da incerteza sobre o resultado de um processo aleatório"[80].

No início, Shannon desenvolveu uma forma matemática de medir o quanto de informação existe na transmissão de uma mensagem. Tal proposta baseava-se em conceitos estatísticos, expressando a ideia de que o aumento da probabilidade do próximo símbolo diminuiria o tamanho da informação [81]. Assim sendo, há uma relação muito forte entre o conceito de entropia de Shannon e o conceito de incerteza. Por isso, a entropia pode ser definida como a quantidade de incerteza que há em uma mensagem, a qual diminui à medida que os símbolos são transmitidos, ou seja, à medida que a mensagem vai sendo conhecida, tendo-se então a informação, que pode ser vista como redução da incerteza [45].

Como exemplo didático, ao utilizar como idioma a língua portuguesa, como fonte de informação, e ao transmitir como símbolo a letra “q”, a probabilidade do próximo símbolo ser a letra “u” é maior se comparado com outro símbolo, enquanto a probabilidade de ser novamente a letra “q” é praticamente nula. Pois, a ocorrência de palavras formadas pelo conjunto "qu"é maior se comparados com outros tipos de combinação. A Fig. 2.7 ilustra este exemplo didático.

A definição de quantidade de informação ou incerteza probabilística, proposta por Shannon, utiliza um tratamento probabilístico e faz uso de uma função logarítmica, tendo sua forma funcional sendo representada por [45]:

$$H(P_i,...,PL) = \sum_{i=1}^{L} P_i log_2 \frac{1}{P_i}, \tag{2.18}$$

em que, L é o tamanho da mensagem e P_i a probabilidade de ocorrência de cada evento i.

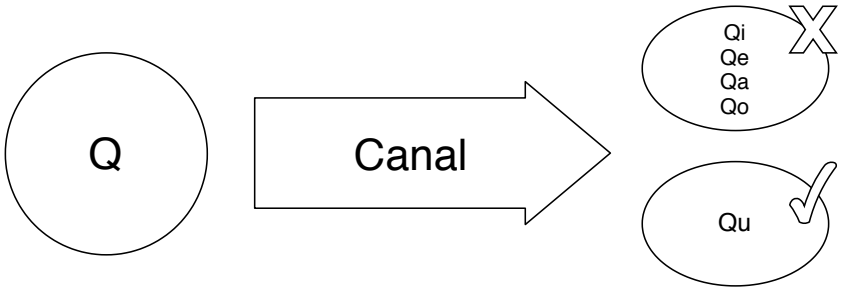


Figura 2.7: Exemplo didático da relação entre entropia e incerteza baseada na estatística. De forma intuitiva, a medida que a informação é transmitida, nesse caso a letra "q", é possível prever a próxima letra, no exemplo acima é mais provável a ocorrência da letra "u" do que as letras "a", "e", "i" e "o". Exemplo retirado de [82].

Com isso, a Equação (2.18) trata estatisticamente uma mensagem levando em consideração os símbolos que a compõe, da mesma forma que a mecânica estatística descreve um sistema macroscópico, levando em consideração os seus constituintes microscópicos. Assim, o termo entropia ganha um novo significado, o de indicar a medida de incerteza probabilística em uma dada distribuição de probabilidade, passando a ser denominada entropia de Shannon. Dessa forma, algumas características interessantes da Teoria da Informação, são definidas como: entropia máxima só é atingida quando a ocorrência de todos os símbolos é equiprovável, a medida que a ocorrência de um grupo de símbolos se torna mais provável que a dos outros símbolos do repertório, a entropia decresce e quando existe certeza sobre qual símbolo vai ser transmitido, a entropia é zero.

Se uma fonte de transmissão tiver 4 caracteres "A", "B", "C" e "D", uma possível mensagem poderá ser "ABADDCAB". A teoria da informação fornece uma maneira de calcular a menor quantidade possível de informações para tal mensagem. Se todas as quatro letras são igualmente prováveis, neste caso 25%, a melhor maneira, em um canal binário, de representar cada caractere é codificar em 2 bits cada letra: "A" pode ser codificada como "00", "B" como "01", "C" como "10" e "D" como "11". No entanto, se "A" ocorrer com 70% de probabilidade, "B" com 26% e "C" e "D" com 2% cada, é possível atribuir códigos de comprimento variável, de modo que receber um ’1’ decida olhar para outro bit a menos que 2 bits de 1s sequenciais já tenham sido recebidos. Nesse caso, "A" seria codificado como "0", um bit, "B" como "10" e "C" e "D" como "110" e "111". É fácil ver que 70% das vezes apenas um bit precisa ser enviado, 26% das vezes dois bits e apenas 4% das vezes 3 bits. Em

média, são necessários menos de 2 bits, pois a entropia é menor, por causa da alta prevalência de "A" seguida de "B" juntos 96% dos caracteres. O cálculo da soma das probabilidades de log na base 2 ponderadas por probabilidade mede e captura esse efeito. Dada a Equação (2.18):

$$H = 0{,}7log_2\frac{1}{0{,}7} + 0{,}26log_2\frac{1}{0{,}26} + 0{,}02log_2\frac{1}{0{,}02} + 0{,}02log_2\frac{1}{0{,}02}$$

$$H = 1{,}0913.$$

2.3 Representação Numérica

Desde meados da década de 1950 o sistema de ponto flutuante é usado, com isso, o primeiro computador comercial com hardware de ponto flutuante foi o computador Z4 desenvolvido por Konrad Zuse, projetado em 1942-1945. Em 1946, a Bell Laboratories introduziu a Mark V, que implementava números decimais de ponto flutuante [83]. Inicialmente, os computadores usavam muitas representações diferentes para números de ponto flutuante. A falta de padronização era um problema contínuo no início da década de 1970 para aqueles que escreviam e mantinham código-fonte de alto nível; esses padrões de ponto flutuante do fabricante diferiam nos tamanhos das palavras, nas representações, no comportamento de arredondamento e na precisão geral das operações [84]. Como consequência, alguns sistemas numéricos eram menos precisos que outros e isso ocasionava incompatibilidade na forma como um programa se comportava em diferentes máquinas. Com isso, no início dos anos 80 foi criado o padrão IEEE 754. Esse padrão foi significativamente baseado em uma proposta da Intel, que estava projetando o i8087 coprocessador numérico; A Motorola, que estava projetando o 68000 na mesma época, também deu uma contribuição significativa.

O padrão IEEE para ponto flutuante binário e aritmética foi publicado em 1985, sendo denominado oficialmente como ANSI/IEEE Std 754-1985. Em 1989, ele recebeu reconhecimento internacional como IEC 559. Em 2008, uma versão atualizada foi publicada, sendo denominada IEEE 754-2008. Nesta revisão estendeu o padrão anterior, adicionou aritmética e formatos decimais, reforçou certas áreas do padrão original que foram deixadas indefinidas e fundiu-se no IEEE 854 (o padrão de ponto flutuante independente de raiz). Em 2011, o padrão internacional ISO/IEC/IEEE 60559:2011 (com conteúdo idêntico ao IEEE 754) foi aprovado para uso. E por

fim, a última atualização ocorreu em 2019, sendo denominada IEEE Std 754-2019 [38].

O padrão Institute of Electrical and Electronics Engineers (IEEE) [38] normalizou aspectos essenciais para os fabricantes de desenvolvedores de *softwares*. Este padrão consiste representações como a operação de arredondamento adequado para o ponto flutuante; tratamento consistente de exceções, tais como a divisão por zero [84]. A seguir são apresentadas noções da representação dos números no computador de acordo aritmética de ponto flutuante.

2.3.1 Noções da Representação dos Números no Computador

A aritmética de ponto flutuante é baseada na notação exponencial ou científica, e é utilizada por computadores na representação dos números e execução das operações [38, 85]. Um número x é representado como ponto flutuante da seguinte maneira:

$$x = \pm S \times B^E, \tag{2.19}$$

sendo $1 \leq S < 10$ e E é inteiro, chamado de expoente. S é chamado de significante ou mantissa e B é a base utilizada. Normalmente os computadores utilizam a base binária ($B = 2$) [86]. Assim, o número x é escrito como:

$$x = \pm S \times B^E, \tag{2.20}$$

em que $1 \leq S < 2$, Consequentemente, a expansão binária do significante do número x é dada por

$$S = (b_0 b_1 b_2 b_3 ... b_{p-1}), \tag{2.21}$$

com $b_0 = 1$ e p é a precisão do sistema.

O padrão IEEE 754-2008 define que o primeiro bit do significante é diferente de zero, como apresentado na Equação (2.21). Normalmente, esse primeiro bit é chamado de bit escondido. Os bits após o ponto binário são chamados de parte fracionária do significante e as Equações (2.20) e (2.21) são representações normalizadas do número x e o processo de obtenção desta representação chama-se normalização. Para representar um número normalizado, a sua representação binária é dividida em três partes: sinal, expoente E e o significante S. Se $x \in \mathbb{R}$ pode

ser armazenado exatamente em um computador então x é chamado de número ponto flutuante. Senão, x deve ser arredondado [84].

2.3.2 Precisão, Épsilon da Máquina e Ulp

De acordo com Overton [84] a precisão (p) de um sistema de ponto flutuante é o número de bits do significante, incluindo o bit escondido. No sistema de 32 bits, a precisão é $p = 24$, sendo 1 bit escondido e 23 bits do significante. Um ponto flutuante normalizado com precisão p é expresso por

$$S = (b_0 b_1 b_2 b_3 ... b_{p-1}) \times 2^E. \tag{2.22}$$

O menor ponto flutuante x que é maior do que 1 é dado por:

$$(1{,}00...01)_2 = 1 + 2^{-(p-1)}. \tag{2.23}$$

Denomina-se épsilon da máquina (*machine epsilon*) a distância entre este número e o número 1:

$$\varepsilon = (0{,}00...01)_2 = 2^{-(p-1)} \tag{2.24}$$

O padrão IEEE especifica dois formatos básicos: *single* e *double*. O formato *single* disponibiliza 1 bit para o sinal, 8 bits para o expoente e 23 bits para o significante. E, o formato *double* possui 1 bit para armazenar o sinal, 11 bits para o expoente e 52 bits para o significante. O sinal sempre é representado com 1 bit e pode ser negativo se o bit tem valor 1 ou positivo se o bit tem valor 0 [84]. A Tabela 2.1 mostra a precisão para diferentes formatos.

Tabela 2.1: Precisão dos formatos de ponto flutuante do IEEE. Para os formatos *half precision*, *single* e *double* a precisão corresponde, respectivamente, 11, 24 e 53. Dessa forma, com a Equação (2.24) é possível estipular os valores de ε.

Formato	Precisão	Épsilon da Máquina
Half Precision	$p = 11$	$\varepsilon = 2^{-10} \approx 9{,}8 \times 10^{-4}$
Single	$p = 24$	$\varepsilon = 2^{-23} \approx 1{,}2 \times 10^{-7}$
Double	$p = 53$	$\varepsilon = 2^{-52} \approx 2{,}2 \times 10^{-16}$

De modo mais geral, para qualquer ponto flutuante pode-se determinar a distância dele para o próximo número que a máquina é capaz de armazenar. Essa distância é chamada de *ulp (unit*

in the last place) e é dada por:

$$ulp(x) = (0{,}00...01)_2 \times 2^E = 2^{-(p-1)} \times 2^E = \varepsilon \times 2^E \tag{2.25}$$

Com base na Equação (2.25), observa-se que quanto maior o valor de x maior também é sua *ulp*, ou seja, maior é a faixa de valores ao redor de x não representáveis no computador. Se $x > 0$ então *ulp(x)* é a distância entre x e o próximo maior ponto flutuante. Se $x < 0$ então *ulp(x)* é a distância entre x e o próximo menor ponto flutuante.

2.3.3 Representação IEEE

O expoente utiliza uma representação deslocada (*biased representation*). Para o formato *single*, o expoente é a representação binária de $E + 127$. A faixa de E vai do número binário de 1 a 254, o que corresponde a $E_{min} = -126$ e $E_{max} = 127$ [86]. O menor número positivo normalizado é denotador por

$$N_{min} = (0{,}00...01)_2 \times 2^{-126} = 2^{-126} \approx 1{,}2 \times 10^{-38}. \tag{2.26}$$

O maior número positivo normalizado é

$$N_{max} = (1{,}111...1)_2 \times 2^{127} = \left(2 - 2^{-23}\right) \times 2^{127} \approx 3{,}4 \times 10^{38}. \tag{2.27}$$

Para o formato *double*, os expoentes variam de $E_{min} = -1022$ a $E_{max} = 1023$ [84]. Sendo assim, o menor número positivo normalizado corresponde por:

$$N_{min} = (0{,}00...01)_2 \times 2^{-1022} = 2^{-1022} \approx 2{,}2 \times 10^{-380}. \tag{2.28}$$

O maior número positivo normalizado é:

$$N_{max} = (0{,}00...01)_2 \times 2^{1023} = 2^{1023} \approx 1{,}8 \times 10^{380}. \tag{2.29}$$

A Tabela 2.2 apresenta a faixa de representação dos formatos de ponto flutuante do IEEE.

Números maiores que $zero$ e menores do que o menor valor possível de ser representado pelo formato utilizado são considerados iguais a zero pelo computador. A esse arredondamento chama-se *underflow*. Da mesma forma, números que ultrapassam o limite superior do formato

Tabela 2.2: Faixa de representação dos formatos de ponto flutuante do IEEE. Para o formato *single* o intervalo de variação do expoente corresponde à $-126 < E < 127$. Com isso, através das Equações (2.26) e (2.27) é possível calcular o menor número positivo normalizado, N_{min}, e o maior número positivo normalizado, N_{max}. Da mesma maneira, Para o formato *double* o intervalo de variação do expoente corresponde à $-1022 < E < 1023$. Dessa forma, as Equações (2.28) e (2.29) fornecem o menor número positivo normalizado, N_{min}, e o maior número positivo normalizado, N_{max}. Seguindo a mesma metodologia é possível calcular o menor e maior números positivos normalizado para o formato *half precision*. Números maiores que zero e menores do que o menor valor possível de ser representado são arredondados para zero. Da mesma forma, números que ultrapassam o maior número representável são considerados infinito.

Formato	E_{min}	E_{max}	N_{min}	N_{max}
Half Precision	-6	7	$2^{-6} \approx 1{,}6 \times 10^{-2}$	$2^{7} = 128$
Single	-126	127	$2^{-126} \approx 1{,}2 \times 10^{-38}$	$2^{128} \approx 3{,}4 \times 10^{38}$
Double	-1022	1023	$2^{-1032} \approx 2{,}2 \times 10^{-380}$	$2^{1023} \approx 1{,}8 \times 10^{380}$

da máquina, são considerados infinito e essa ocorrência é chamada de *overflow*. Essa faixa de valores é simétrica, ou seja, vale tanto para números positivos quanto para negativos [38]. Quando qualquer computação tenta produzir um valor que é indefinido, mesmo no sistema de números reais, o resultado é um valor excepcional conhecido como Not-a-Number ou NaN [84].

Existem alguns números que não podem ser normalizados, ou seja, a normalização resulta em um expoente fora da faixa de representação suportada pelo formato da máquina. Esses números usam a combinação de expoente zero e a parte fracionária não zero e são chamados de números subnormalizados. Números subnormalizados têm sua precisão reduzida à medida que o número decresce [84].

2.3.4 Arredondamentos

Por sua natureza, todos os números expressos em formato de ponto flutuante são números racionais com uma expansão final na base relevante (por exemplo, uma expansão decimal final na base 10 ou uma expansão binária final na base 2). Números irracionais, como π ou $\sqrt{2}$ devem ser aproximados. O número de dígitos (ou bits) de precisão também limita o conjunto de números racionais que podem ser representados exatamente. Por exemplo, o número decimal 123456789 não pode ser representado exatamente se apenas oito dígitos decimais de precisão estiverem disponíveis (seria arredondado para 123456790 ou 123456780, onde o dígito mais à direita, 0, não está explicitamente armazenado).

Quando um número é representado em algum formato (como uma sequência de caracteres) que não é uma representação de ponto flutuante nativa suportada em uma implementação de

computador, será necessária uma conversão antes de poder ser usado nessa implementação. Se o número puder ser representado exatamente no formato de ponto flutuante, a conversão será exata. Se não houver uma representação exata, a conversão exigirá uma escolha de qual número de ponto flutuante usar para representar o valor original. A representação escolhida terá um valor diferente do original, e o valor assim ajustado é chamado de valor arredondado. O padrão IEEE define o valor arredondado correto de x, denotado por *round(x)* da seguinte forma.

Definição 2.3.1. *(Modos de arredondamentos). Se x é um número flutuante então $round(x) = x$. Senão, o valor depende do modo de arredondamento:*

- *Arredondamento para baixo:* $round(x) = x_{inf}$.
- *Arredondamento para cima:* $round(x) = x_{sup}$.
- *Arredondamento em direção a zero:* $round(x) = x_{inf}(x > 0)$ *ou* $round(x) = x_{sup}(x < 0)$.
- *Arredondamento para o mais próximo:* $round(x)$ *é tanto* x_{inf} *ou* x_{sup}*, dependendo de qual for mais próximo de* x*. Se houver um empate, aquele com o último bit significativo igual a zero é escolhido.*

Os modos de arredondamento são úteis quando a quantidade de erro que está sendo introduzida deve ser limitada, também são úteis no diagnóstico de instabilidade numérica [87]. Além disso, um método interessante que usa os modos de arredondamento é a aritmética intervalar [88]. Na Fig. 2.8 o número x será representado pelo intervalo do arrendondamento para baixo, x_{inf}, e para cima, x_{sup}, dessa forma, constituindo o intervalo do número x. Tal conceito está representado no experimento computacional I, pois os pólos e zeros são arredondados pelo seu valor de quantização, sendo neste caso, a representação em ponto flutuante.

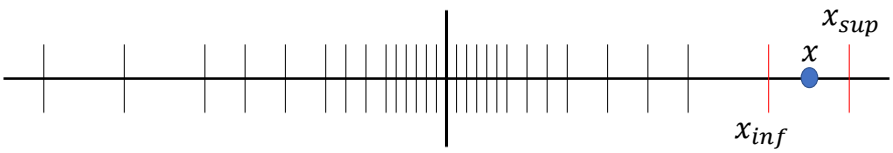


Figura 2.8: Representação de ponto flutuante e arredondamento. Os valores arredondados, x_{inf} e x_{sup}, constituem o intervalo do número x. Este método de representação numérica foi utilizado em [89] para obter a resposta em frequência de filtros digitais.

Exemplo 2.3.1. *Para a representação de número em ponto flutuante, considere o número* 45,45 *em um sistema de precisão simples, sendo o seu total de 32 bits, 1 bit para o sinal, 8 bits para o expoente e 23 bits para a representação da mantissa, e ainda as conversões para binário:*

$45 \xrightarrow{bin\acute{a}rio} 101101$

$0{,}45 \xrightarrow{bin\acute{a}rio} 0{,}0111001100....$

Dessa forma, a representação em binário do número 45,45 *é:*

$101101{,}0111001100....$

Normalizando o número,

$1{,}01101011100 \times 2^5$.

Com isso, o expoente é igual à 5 *e a mantissa* 01101011100. *Seguindo a metodologia de cálculo das Equações* (2.26)-(2.29), E + 127, *desse forma,* 5 + 127 = 132. *Transformando* 132 *em binário,* 10000100, *é possível escrever o número* 45,45 *em seu formato de ponto flutuante.*

$$\overbrace{0}^{Bit\ de\ Sinal}\ \overbrace{10000100}^{Expoente}\overbrace{01101011100110011001101}^{Mantissa}$$

Neste caso, o número 45,45 em binário é uma dízima, com isso para a representação em ponto flutuante este valor foi arredondado para a representação mais próxima. Dessa forma, atribuiu-se '1' no último bit da mantissa. Como consciência, convertendo o número armazenado em ponto flutuante para decimal obtém-se 45.450000762939453125.

2.4 Erro de Quantização

Para a implementação de filtros digitais, necessita-se encontrar os coeficientes do filtro. Para calcular e avaliar os coeficientes do sistema digital, usa-se a aritmética de precisão finita. Sendo esta basicamente limitada apenas pela representação numérica disponível do sistema. Isso é feito quando a velocidade dos cálculos não é um problema, e sim a precisão. Portanto, traduzindo essa definição de aritmética de precisão finita para sistemas digitais, pode-se dizer que o número de bits que usamos no projeto de um filtro é limitado pelo tamanho da palavra do registro usado para armazená-los.

O fato, no entanto, é que a maioria dos sistemas digitais tem um número fixo de bits em seus registros, sendo assim, a capacidade dos registros é limitada. Dessa forma, um dos métodos para quantificar a limitação computacional é quantizar os coeficientes do filtro para o tamanho

da palavra do registro, como arredondamento ou truncamento [38, 84]. A localização dos pólos e zeros de qualquer filtro digital depende diretamente do valor dos seus coeficientes. Mas, como os valores estão sendo quantizados, para ajustá-los ao registrador, haverá uma mudança nos valores dos pólos e zeros. Isso, por sua vez, faz com que a localização dos pólos e zeros mude da localização desejada. Assim, a quantização dos coeficientes do filtro cria um desvio na resposta em frequência do sistema [89].

Exemplo 2.4.1. *Considere a seguinte função de transferência:*

$$H(z) = \frac{z^2}{(z - 0{,}9)(z - 0{,}8)}, \tag{2.30}$$

convertendo os polos para binário, truncando em 3 bits e reescrevendo o seu valor arredondado em decimal:

$$0{,}9 \rightarrow 0.1110 \rightarrow 0.111 \rightarrow 0{,}875$$

$$0{,}8 \rightarrow 0.1100 \rightarrow 0.110 \rightarrow 0{,}75.$$

Com esse exemplo fica evidente a influência do processo de quantização na representação de pólos em funções de transferência.

Esse erro de quantização pode ser visto de uma maneira alternativa se considerarmos os coeficientes do filtro delimitados por um intervalo. Assim, a quantização pode ser vista como a adição de uma certa quantidade de incerteza. Quanto menos bits for usado na quantização; mais ruído é adicionado. Essa é precisamente a fonte de ruído mostrada na Fig. 3.1.

Usando uma representação de ponto fixo, o erro de quantização é dado por:

$$|\epsilon_r| \leq \frac{Q}{2}, \tag{2.31}$$

em que $Q = 2^b$, b é o número de bits. Assim, os coeficientes da Equação (2.17) apresentam limites mais baixos dado por:

$$\underline{a}_k = a_k - Q/2 \tag{2.32}$$

$$\underline{b}_k = b_k - Q/2, \tag{2.33}$$

enquanto os limites superiores são dados por

$$\overline{a}_k = a_k + Q/2 \tag{2.34}$$

$$\overline{b}_k = b_k + Q/2. \tag{2.35}$$

Isso equivale a dizer que o erro de quantização produz um intervalo em torno do valor desejado dos coeficientes. Em outras palavras, o valor aproximado dos coeficientes $\hat{a}_k$ e $\hat{b}_k$ é dado por:

$$\underline{a}_k < \hat{a}_k < \overline{a}_k \tag{2.36}$$

$$\underline{b}_k < \hat{b}_k < \overline{b}_k. \tag{2.37}$$

2.5 Processamento de Sinais com o Octave

Para a realização das simulações, o toolbox do Octave $signal$ fornece funções e comados para analisar, pré-processar e extrair recursos de sinais amostrados de maneira uniforme e não uniforme. Os recursos do *software* inclui ferramentas para projeto e análise de filtro, reamostragem, suavização, degradação e estimativa do espectro de potência. Dessa forma, projeta-se um filtro Butterworth da seguinte forma:

```
[b,a] = butter(n,W,ftype);
```

em que, b e a são respectivamente os coeficiente do numerador e denominados, n é a ordem do filtro e w a frequência de corte, deve ser especificada em radianos para filtros analógicos. Além disso, $ftype$ é o tipo de classificação do filtro, neste caso o comando low representa o filto passa-baixa, $high$, $stop$ e $pass$, respectivamente, filtro passa-alta, rejeita-faixa e passa-faixa. Para a simulação dos dois últimos é preciso representar w_1 e w_2, onde $w_1 < w_2$.

Para o projeto de filtros Chebyshev usa-se o comando:

```
[b,a] = cheby1(n,Rp,W,ftype);
```

sendo n, w e $ftype$ os mesmos comandos do Butterworth e Rp são os decibéis de ondulação de banda passante de pico a pico.

Para o projeto de filtros Elípticos:

```
[b,a] = ellip(n,Rp,Rs,Wp,ftype);
```

em que n, w, Rp e $ftype$ os mesmos comandos dos filtros anteriores. Além disso, Rs são os decibéis de atenuação da banda de parada.

Além disso, outros comandos foram utilizados paro o desenvolvimento do Algoritmo 3.1, para o comando $size$ retorna o tamanho da dimensão N no enésimo argumento. O número de linhas é retornado no primeiro argumento, e o número de colunas é retornado no segundo argumento.

```
size([1, 2; 3, 4; 5, 6])
>> [3, 2]
```

Além disso, utilizou-se o comando:

```
A = [9 2 9 5];
C = unique(A)
>> 2   5   9
```

O comando $unique$ retorna os elementos exclusivos de A , classificados em ordem crescente. Se a entrada A for um vetor, a saída também será um vetor com a mesma orientação (linha ou coluna) da entrada. Para uma entrada de matriz, a saída é sempre um vetor de coluna. Por fim, o último comando:

```
A = 1:10;
S = sum(A)
>> 55
```

Soma dos elementos ao longo da dimensão do vetor A.

CAPÍTULO 3

METODOLOGIA

Neste capítulo, o conceito de entropia aliado a filtros será discutido e três experiências numéricas são descritas. Para cada experimento, as etapas principais são apresentadas. Todas as experiências numéricas foram realizadas no Octave [90] em um computador com Windows 10, processador Intel i7 e sistema operacional de 64 bits. Essas experiências foram projetadas para verificar alguns efeitos de precisão finita na entropia de sinais filtrados digitalmente. Na Experiência Numérica 1, pólos e zeros são perturbados por erro de quantização devido a uma representação de ponto fixo de 16 e 32 bits. O experimento numérico 2 visa examinar o processo de filtragem digital com o uso do filtro elíptico. A correlação entre aumento de ordem e entropia é verificada no experimento numérico 3.

3.1 Entropia em Filtros Digitais

A conexão com o filtro digital fica clara quando o esquema original proposto por Shannon é observado. Este esquema foi adaptado na Figura 3.1. Shannon estava interessado em como uma mensagem poderia ser transmitida através de um canal de um transmissor para um destino. Nesse processo, um recurso necessário é considerar a presença de ruído. Aqui, vemos esse esquema da perspectiva da filtragem. Assim, o canal será o filtro que através de uma entrada definida a altera na saída. A fonte de ruído neste caso vem do hardware/software de precisão finita, a quantização e os arredondamentos em sistemas computacionais, onde o filtro digital é implementado. É evidente que em aplicações reais muitas outras fontes de ruído devem ser consideradas. No entanto, para os fins desta dissertação, considera-se apenas a operação do filtro como fonte de ruído.

Na seção 22, Shannon [45] afirma “A operação do filtro é essencialmente uma transformação linear de coordenadas". Shannon deduziu isso considerando o fato de que se um conjunto com uma entropia H_1 por grau de liberdade na banda W é passado através de um filtro com a característica $Y(f)$, o conjunto de saída tem uma entropia H_2 dada pela Equação (3.1). Em outras palavras, as frequências dos novos componentes são apenas as antigas multiplicadas por

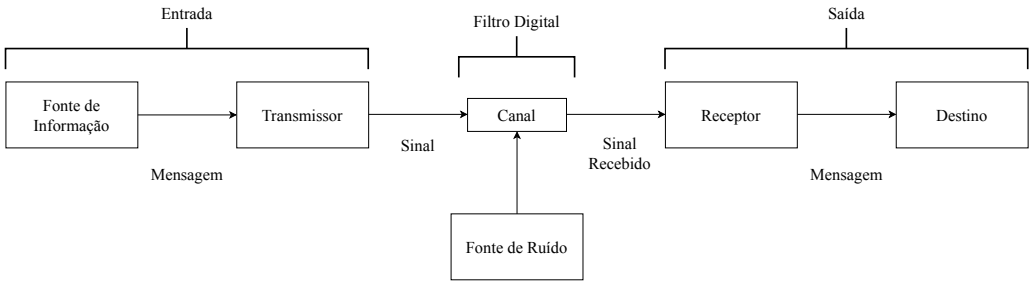


Figura 3.1: Adaptação do "Diagrama esquemático de um sistema de comunicação geral"em [45]. Nesta dissertação, o foco está em ver o canal como uma fonte de filtro e ruído como consequência da implementação de precisão finita dos filtros digitais.

um ganho. Além disso, Shannon descreveu de tal maneira que um filtro apresenta um impacto direto na entropia de um sinal. Fica claro na ideia de Shannon que os sinais filtrados pelos filtros ideais passa-alta, passa-baixo, banda passante ou faixa de parada devem ter sua entropia diminuída, como pode ser visto em [45, p. 40]

$$H_2 = H_1 + \frac{1}{W}\int_W log|Y(f)|^2 df. \tag{3.1}$$

Existem alguns tipos de entropia caracterizados na literatura. No que diz respeito à termodinâmica, a entropia alude à medida de desordem. Na mecânica estatística, refere-se à quantidade de incerteza no sistema. Na teoria da informação, é uma proporção da incerteza relacionada a uma variável aleatória [45, 91]. Shannon fornece o número ideal de dígitos binários para representar cada evento de uma determinada mensagem, para que o número médio de bits/eventos da mensagem seja o menor possível. A entropia de Shannon é definida por [92],

$$H(X) = \sum_{i=1}^{L} P_i log_2 \frac{1}{P_i}, \tag{3.2}$$

onde $H(X)$ é a entropia (bits), X é um símbolo para o sinal, P_i é o valor de probabilidade do símbolo X; L é o tamanho do sinal. Nesta dissertação, a entropia foi medida para comprimentos de palavras de 16 e 32 bits. Em um sinal aleatório completo representado por um comprimento de palavra de 16 bits, a entropia é exatamente 16 bits.

Para prosseguir com o cálculo da entropia de Shannon, aplicamos o seguinte processo de padronização ao sinal de saída da seguinte forma:

$$S_k = \text{ceil}\left(\frac{y_k - \min(y_k)}{\max(y_k - \min(y_k))} \times 2^{WL}\right), \tag{3.3}$$

onde y_k é o sinal; ceil(x) é uma função que retorna o menor número inteiro não inferior a x; min e max retornam o menor e o maior valor de um vetor, respectivamente; WL é o comprimento da palavra dado em bits.

A Figura 3.2(a) e (b) apresenta uma onda senoidal

$$y(t) = 2\sin(2\pi 2t)$$

amostrado em $\delta = 0.01$ para ilustrar esse procedimento. Para este seno, a entropia calculada, Algoritmo 3.1, usando $WL = 8$ e Equação (3.2) é $H = 4.71$. Um sinal aleatório distribuído uniforme é mostrado na Figura 3.2(c), para o qual a entropia calculada é $H(k) = 7.59 \pm 0.03$. Aumentando o número de amostras do sinal aleatório, o valor da entropia se aproxima de 8, conforme o esperado.

Uma última observação sobre esse procedimento está relacionada à necessidade de descartar o transitório e limitar o número de amostras para o cálculo da entropia dos sinais filtrados. O número de amostras foi adotado como 2^{10}, o que limita a medida da entropia até 10. Somente em uma tabela 4.3, adotamos 2^{12} amostras. Testes feitos com maior número de amostras nos mostraram que esse limite é suficiente para uma estimativa confiável da entropia de Shannon neste trabalho.

3.1.1 Entropia para Detectar Ruído

A entropia tem sido amplamente utilizada para detectar ruídos em sinais e imagens [91, 93–95]. Para mostrar a eficácia da entropia como uma maneira de detectar o crescimento de ruído em um sinal, calculamos a entropia alterando a variação do ruído gaussiano de $\sigma = 0{,}01$ para $\sigma = 0{,}02$. A média foi mantida como $\mu = 0... \geq$. Uma onda senoidal é mostrada na Figura 3.3a. Ruído gaussiano com $\sigma = 0{,}01$ e $\sigma = 0{,}02$ foi adicionado a essa onda senoidal e representado na Figura 3.3b,c, respectivamente. A entropia calculada são (a) 5,66, (b) $5{,}95 \pm 0{,}03$, and (c) $6{,}23 \pm 0{,}04$. O nível do ruído gaussiano é pouco visível, mas a entropia tem sido sensível ao aumento do ruído.

A entropia é uma maneira sensível de medir a incerteza. Para mostrar ainda mais essa

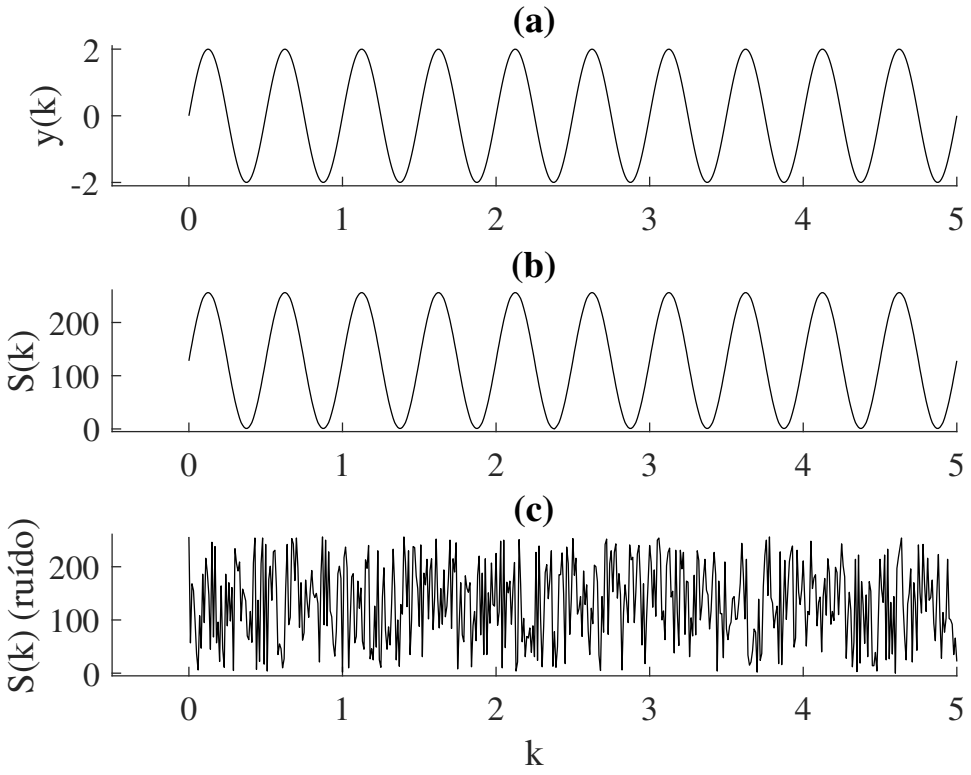


Figura 3.2: Procedimento de padronização do vetor $y(k)$ em (a) para cálculo de entropia usando Eq. (3.3). O resultado $S(k)$ em (b) é composto apenas por números inteiros dentro 0 até $2^{WL} - 1$. Nesse caso, foi utilizado $WL = 8$ bits. (c) mostra um sinal aleatório com distribuição uniforme. 50 execuções desse sinal produzem uma entropia de $H = 7{,}59 \pm 0{,}03$. Aumentando o número de amostras, esse valor se aproxima de 8, conforme o esperado.

propriedade, vamos comparar esta medida com a conhecida relação sinal-ruído (SNR) dada em dB pela seguinte equação

$$SNR = 20 \log_{10} \frac{A_{\text{sinal}}}{A_{\text{ruído}}}, \tag{3.4}$$

onde A é a amplitude do quadrado médio da raiz (RMS). Seja a relação entre a entropia de sinal e ruído (ESN)

$$ESN = 20 \log_{10} \frac{H_{\text{sinal}}}{H_{\text{ruído}}}, \tag{3.5}$$

onde H é a entropia do sinal e do ruído. Usando essas duas equações, vamos comparar a sensibilidade em uma pequena variação de ruído. A Tabela 3.1 mostra a diferença entre SNR e ESN para o sinal da Figura 3.1b (onda senoidal com ruído gaussiano de $\sigma = 0{,}01$) e o mesmo sinal, mas com um σ dada na primeira coluna da Tabela 3.1. A mensagem desta Tabela

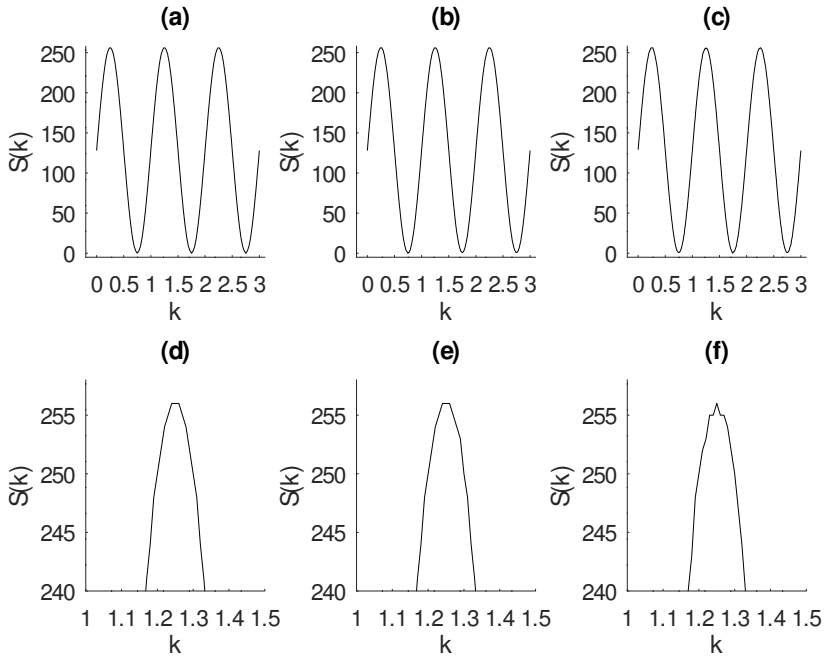


Figura 3.3: Cálculo da entropia de Shannon para três sinais. Todos os sinais foram padronizados de acordo com o procedimento descrito na Equação (3.3). (**a**) Onda senoidal $2sin(2\pi 2t)$. (**b**) Onda senoidal adicionada com ruído gaussiano de $\sigma = 0{,}01$ e $\mu = 0$. (**c**) Onda senoidal adicionou ruído gaussiano de $\sigma = 0{,}02$ e $\mu = 0$. A entropia calculada é (**a**) $5{,}66$, (**b**) $5{,}95 \pm 0{,}03$, e (**c**) $6{,}23 \pm 0{,}04$. O nível do ruído gaussiano é baixo; no entanto, a entropia tem sido sensível ao aumento do ruído. Painéis (**d**–**f**) mostram um zoom na figura acima para ver a presença de um pequeno nível de ruído nos sinais dos painéis (**b**,**c**).

é simples. Para o caso de $\sigma = 0{,}0200$, o SNR dá uma diferença de $2{,}6359 \pm 0{,}6920$ dB, Considerando que a entropia para esta diferença é $15{,}9343 \pm 3{,}3038$ dB. Quando a diferença entre a variação do ruído desses dois sinais é apenas $0{,}0125 - 0{,}01 = 0{,}0025$, temos mais confiança para usar o ESN para detectar esse nível de diferença de ruído, pois a diferença entre o SNR desses dois sinais é $0{,}527 \pm 0{,}589$, considerando que para o ENS temos $4{,}023 \pm 2{,}866$. Para o caso SNR, o intervalo dado por um σ é $(-0{,}062; 1{,}116)$ e perdemos a confiança para garantir que um dos sinais apresente um nível de ruído mais alto que o outro.

3.2 Métodos de Simulação

Para a realização dos objetivos desta dissertação, alguns processos de simulação foram desenvolvidos. Para a entropia de Shannon o método de cálculo é apresentado no Algoritmo 3.1, seguindo os seguintes passos:

Passo 1: Estabelecer tamanho dos dados.

Passo 2: Organização dos dados em ordem crescente através do comando $unique$.

Tabela 3.1: Sensibilidade à variação do ruído para SNR e ESN. O sinal de referência foi adicionado com o ruído gaussiano de $\sigma = 0{,}01$. Aqui, mostramos a diferença entre a medida de SNR e ESN em relação a este sinal. Observe que um aumento de 0,01 no desvio padrão do sinal é capaz de aumentar 15,934 na ESN (entropia) e apenas 2,635 na relação sinal/ruído (SNR). Foram utilizadas 50 execuções para calcular a média e o desvio padrão dessas quantidades em dB. Quando a diferença entre esses dois sinais é de apenas 0,002, o ESN seria mais robusto para detectar essa diferença que o SNR.

σ	**Diferença de SNR (dB)**	**Diferença de ESN (dB)**
0,0200	2,635 ± 0,692	15,934 ± 3,303
0,0192	2,571 ± 0,664	14,928 ± 2,710
0,0183	2,104 ± 0,673	14,110 ± 2,858
0,0175	1,905 ± 0,538	11,525 ± 2,835
0,0167	1,648 ± 0,582	11,481 ± 3,356
0,0158	1,509 ± 0,728	9,620 ± 3,438
0,0150	1,265 ± 0,741	8,498 ± 4,123
0,0142	0,870 ± 0,650	6,720 ± 3,219
0,0133	0,745 ± 0,623	4,817 ± 3,229
0,0125	0,527 ± 0,589	4,023 ± 2,866

Passo 3: Cálculo da frequência com que cada elemento aparece no vetor X.

Passo 4: Estabelecimento da probabilidade P.

Passo 5: Cálculo da entropia H.

Algoritmo 3.1 Pseudo-código para o cálculo da entropia.

X dados a serem analisados
para `k=1:m` **faça**
$Alphabet \leftarrow unique(X(:,k))$

    para `i:length(Alphabet)` **faça**
$Frequency(i) \leftarrow sum(X(:,k) == Alphabet(i))$
    fim para
$P \leftarrow \frac{Frequency}{sum(Frequency)}$
$H(Column) = sum(P. \quad log2(\frac{1}{P}))$

fim para
Saída H

Além disso, o próximo passo foi a implementação do pacote *signal* do Octave. Dessa forma, os comandos `butter`, `cheby` ou `ellip`, discutidos na Seção 2, são usados para o cálculo dos polos e zeros. Para o filtro butterworth são necessários as especificações de ordem, frequência de corte e o tipo de classificação. No caso do filtro chebyshev, além dos parâmetros definidos anteriormente ainda é necessário especificar o ripple da banda passante. Por fim, no elíptico o valor da banda de rejeição é necessário.

Dessa forma, para a realização do experimento numérico I o sistema digital precisa ser

definido, neste caso 16 ou 32 bits (b). Sendo assim, os coeficientes serão arredondados, $\hat{b}$ e $\hat{a}$, seguindo as Equações (2.32)-(2.35), baseado no erro de quantização, q, definido na Equação (2.31). Por fim, a Equação (2.17) filtra o sinal com os coeficientes arredondados. O Algoritmo 3.2 apresenta a metodologia de simulação.

No experimento numérico II, somente as especificações do filtro elíptico precisam ser determinadas para os cálculos dos polos e zeros. Além disso, o sinal de entrada está apresentado na Tabela 3.3 na coluna sinal completo. O Algoritmo 3.3 ilustra a metodologia de simulação.

No último experimento, as especificações dos polos e zeros seguem as mesmas diretrizes do primeiro experimento. No entanto, a ordem será alterada 1 à 8. O Algoritmo 3.4 mostra a metodologia de simulação.

3.3 Experimento Numérico I

Como foi visto, o aumento do número de bits na representação em máquina, torna o número mais próximo do seu valor real e ainda diminui o efeito de quantização. Além disso, o efeito dos processos de arredondamento pode deslocar, de modo significativo, os polos e zeros desejados. Com isso, a pertubação dos coeficientes do filtro com 16 e 32 bits, espera-se um maior nível de entropia em formatos de sistemas com menores bits.

Dessa forma, o experimento numérico 1 tem o propósito de comparar a pertubação dos polos e zeros com sistemas de 16 e 32 bits. Para a realização da ideia proposta foi desenvolvido os algorítimos, apresentados no `Apêndice A`, de filtragem do sinal para os filtros O esquema proposto pode ser resumido nas seguintes etapas:

Passo 1: Use os comandos `butter`, `cheby` ou `ellip` do Octave para gerar os polos e zeros da função de transferência de acordo com a Equação (2.13). Essa etapa é representada pelas linhas de 1 à 6 do Algoritmo 3.1, sendo os parâmetros ordem (n), frequência da borda passante (ω_c), ripple da banda passante (α_p), ripple da banda de parada (α_r) e tipo de filtros (ft), neste caso ft pode assumir os caracteres low, passa faixa, $high$ passa alta, $pass$ passa faixa e $stop$ rejeita faixa.

Passo 2: Escolha o número de bits e calcule o erro de quantização de acordo com a Equação (2.31). No Algoritmo 3.1, b representa o número de bits e q o erro de quantização. Além disso, $\hat{b}$ e $\hat{a}$ são os coeficientes arredondados baseado no valor quantizado.

Algoritmo 3.2 Pseudo-código da metodologia proposta para o experimento numérico I

1: $b \leftarrow número\ de\ bits$
2: $q \leftarrow \frac{2^{-b}}{2}$
3: $\hat{p} \leftarrow p \pm q$
4: $\hat{z} \leftarrow z \pm q$
5:
6: **para** `n=z-1:k` **faça**
7: $y_1[k] = \sum_{k=0}^{M} \hat{p}_k x[n-k]$
8: $y_2[k] = \sum_{l=1}^{N} \hat{z}_l y[n-l]$
9: $y[k] = y_1[k] - y_2[k]$
10: **fim para**
11: **Saída** $y[k]$
12: **para** `k=1:1:50` **faça**
13: Equação (3.3)
14: Algoritmo 3.1
15: **fim para**
16: **Saída** $H[k]$
17: média($H[k]$)
18: desvio($H[k]$)

Passo 3: Insira o erro de quantização nos polos e zeros, como pode ser visto nas linhas de 8 à 9 do pseudo código. Com isso, $\hat{b}$ e $\hat{a}$ são os coeficientes arredondados baseado no valor quantizado [89]. Dessa forma, a Equação (2.17) pode ser reescrita da seguinte maneira:

$$y[k] = \sum_{k=0}^{M} \hat{b_k} x[n-k] - \sum_{l=1}^{N} \hat{a}_l y[n-l]. \tag{3.6}$$

Passo 4: O sinal é filtrado usando 50 combinações diferentes descritas pelas Equações (2.32)-(2.35). Por exemplo, utilizando a Equação (2.31):

$$y[k] = \underline{b_1} x[1] + \underline{b_2} x[2] + ... + \underline{b_k} x[k] - \underline{a}_1 y[1] - \underline{a}_2 y[2] - \underline{a}_k y[k]. \tag{3.7}$$

$$y[k] = \overline{b_1} x[1] + \underline{b_2} x[2] + ... + \underline{b_k} x[k] - \underline{a}_1 y[1] - \underline{a}_2 y[2] - \underline{a}_k y[k]. \tag{3.8}$$

$$y[k] = \overline{b_1} x[1] + \overline{b_2} x[2] + ... + \underline{b_k} x[k] - \underline{a}_1 y[1] - \underline{a}_2 y[2] - \underline{a}_k y[k]. \tag{3.9}$$

$$y[k] = \overline{b_1} x[1] + \overline{b_2} x[2] + ... + \overline{b_k} x[k] - \underline{a}_1 y[1] - \underline{a}_2 y[2] - \underline{a}_k y[k]. \tag{3.10}$$

As Equações (3.7)-(3.10) correspondem à combinações dos coeficientes arredondados para

a realização da simulação.

Passo 5: Aplique o procedimento de padronização ao sinal filtrado de acordo com a Equação (3.3).

Passo 6: Calcule a média e o desvio padrão da entropia dos 50 sinais filtrados.

Na Experiência Numérica 1, os pólos e zeros do filtro são perturbados com os efeitos da quantização de 16 e 32 bits. O sinal de entrada é composto como uma soma de sinais sinusoidais de 50, 75, 125, 150 Hz. A ordem dos filtros é dada na Tabela 3.2. Além disso, a frequência de corte adotada é de 100 Hz para os casos de passa-baixo e alto, para os filtros passa faixa e rejeita faixa o intervalo de frequência de corte adotado foi de 70 e 130 Hz.

Tabela 3.2: Ordem dos filtros para a experiência numérica 1. Sendo a frequência de corte de 100 Hz para os casos passa-baixo e passa-alto. Para os filtros passa faixa e rejeita faixa foram projetados com 70 e 130 Hz. A taxa de amostragem igual a 0,001, e por fim, a ordem do filtro foi definida, para o máximo valor antes de se observar instabilidade numérica, ou seja, a maior ordem que a implementação numérica permite simular.

Tipo de Filtros	**Butterworth**	**Chebyshev**	**Elliptic**
Passa alta	14	8	6
Passa baixa	14	8	6
Passa faixa	5	4	3
Rejeita faixa	5	4	3

3.4 Experimento Numérico II

Na Experiência Numérica 2, a entropia foi calculada para o sinal original 3 e o sinal filtrado usando filtros elípticos. Para comparar, o sinal de entrada foi simulado sem os componentes de frequência filtrados. Dessa forma, a expectativa dos resultados está no aumento da entropia com a adição de componentes de frequência, e a mesma entropia para o sinal filtrado e o sinal ideal. A descrição completa do sinal de entrada e do sinal idealmente filtrado pode ser vista na Tabela 3.3. A variação do comprimento do sinal foi usada aqui para calcular a média e o desvio padrão da entropia.

Passo 1: Use o comando `ellip` do Octave para gerar os pólos e zeros da função de transferência Equação (2.13). Essa etapa é representada pelas linhas de 1 à 2 do Algoritmo 3.3.

Passo 2: Escolha do sinal de entrada (3.3).

Passo 3: Filtrar sinal usando a Equação (2.17).

Algoritmo 3.3 Pseudo-código da metodologia proposta para o experimento numérico II.

1: **Sinal de entrada na Tabela 3.3**
2:
3: **para** `n=z-1:k` **faça**
4: $y_1[k] = \sum_{k=0}^{M} \hat{p_k} x[n-k]$
5: $y_2[k] = \sum_{l=1}^{N} \hat{z_l} y[n-l]$
6: $y[k] = y_1[k] - y_2[k]$
7: **fim para**
8: **Saída** $y[k]$
9: **para** `k=1:1:50` **faça**
10: Equação (3.3)
11: Algoritmo 3.1
12: **fim para**
13: **Saída** $H[k]$
14: média($H[k]$)
15: desvio($H[k]$)

Tabela 3.3: Sinais de entrada para a experiência numérica 2. Nós projetamos três tipos de sinais compostos por diferentes somas harmônicas. Os valores das frequências de 1 a 6 são 40 Hz, 60 Hz, 80 Hz, 130 Hz, 150 Hz e 170 Hz, respectivamente. Para comparar, o sinal de entrada foi simulado sem os componentes de frequência filtrados, como mostrado na terceira coluna. Isso é equivalente a produzir uma saída por um filtro ideal. Em todos os casos, foi adotada uma taxa amostral de 0,001 s. Valores diferentes ou mesmo taxa de mostragem variável não foram investigados neste trabalho.

Sinal	Sinal completo	Sinal idealmente filtrado
1	$sin(w_1 t) + sin(w_6 t)$	$sin(w_1 t)$
2	$sin(w_1 t) + sin(w_2 t) + sin(w_5 t) + sin(w_6 t)$	$sin(w_1 t) + sin(w_2 t)$
3	$sin(w_1 t) + sin(w_2 t) + ... + sin(w_5 t) + sin(w_6 t)$	$sin(w_1 t) + sin(w_2 t) + sin(w_3 t)$

Passo 4: O sinal é filtrado usando 50 valores de *wl* dentro `1024` to `6024`. Esta etapa é representado pelas linhas de 9 à 11 do pseudo-código.

Passo 5: Aplique o procedimento de padronização ao sinal filtrado de acordo com a Equação (3.3), sendo demostrado na linha 10 do Algoritmo 3.2.

Passo 6: Calcule a média e o desvio padrão da entropia do sinal filtrado.

3.5 Experimento Numérica III

Nesta experiência, a ordem do filtro foi variada para um sinal de entrada com frequências de 20, 60 e 80 Hz e a frequência de corte de 60 Hz. Com isso, o incremento da ordem do filtro aumenta os números de polos e zeros, como consequência, aumenta-se as operações no processo de filtragem e os processos de arredondamento na representação numérica. Dessa

forma, espera-se um aumento da entropia com a varição da ordem.

Algoritmo 3.4 Pseudo-código da metodologia proposta para experiência numérica III.

1: $n \leftarrow 1$ á 8
2:
3: **para** `n=z-1:k` **faça**
4: $y_1[k] = \sum_{k=0}^{M} \hat{p}_k x[n-k]$
5: $y_2[k] = \sum_{l=1}^{N} \hat{z}_l y[n-l]$
6: $y[k] = y_1[k] - y_2[k]$
7: **fim para**
8: **Saída** $y[k]$
9: **para** `k=1:1:50` **faça**
10: Equação (3.3)
11: Algoritmo 3.1
12: **fim para**
13: **Saída** $H[k]$
14: média($H[k]$)
15: desvio($H[k]$)

Passo 1: Use o comando `butter`, `cheby` ou `ellip` de Octave para gerar os pólos e zeros da função de transferência, Equação (2.13).

Passo 2: Filtrar sinal usando Equações (2.17).

Passo 3: O sinal é filtrado usando 50 valores diferentes de WL de 1024 até 6024.

Passo 4: Aplique o procedimento de padronização ao sinal filtrado de acordo com a Equação (3.3).

Passo 5: Calcule a média e o desvio padrão da entropia do sinal filtrado.

Passo 6: Altere a ordem do filtro de 1 à 8 para cada uma das etapas de 1 a 5.

CAPÍTULO 4

RESULTADOS

Filtros digitais em sistemas discretos são implementados com precisão finita. Dessa forma, erros e limitações devido ao tamanho finito das palavras são inevitáveis. Alguns exemplos de consequências da precisão finita em filtros são a quantização dos coeficientes do sistema, erros devido aos arredondamentos na aritmética computacional e uma restrição no nível do sinal, com uma maior exigência de armazenamento. A quantização dos coeficientes do filtro tem o efeito de perturbar a localização dos pólos e zeros do filtro, dessa forma, resposta real do filtro difere um pouco da resposta ideal. Outra restrição consiste em quantificar os cálculos de filtro por arredondamento ou truncamento. O ruído de arredondamento é o erro na saída do filtro resultante de cálculos arredondados ou truncados no filtro. Como o nome indica, esse erro parece ruído debaixo nível na saída do filtro. Além disso, com a aritmética de ponto fixo, é possível que os cálculos do filtro sejam excedidos.

Esses efeitos de quantização devem ser considerados para a realização de um projeto. Como foi vista na secção 3, a entropia tem sido amplamente utilizada para detectar ruídos em sinais e imagens, devido, a sua eficácia como uma maneira de detectar o crescimento de ruído em um sinal. Dessa forma, uma abordagem para realizações eficientes é considerar a quantização como uma fonte de ruído. Para relacionar as limitações computacionais e a variação dos principais parâmetros de um filtro na entropia medida, foi utilizado uma técnica de contorno para observar os efeitos da precisão finita nos parâmetros dos filtros de acordo com o comprimento da palavra de 16 ou 32 bits. Com isso, os resultados, da Experiência Numérica 1 são mostrados na Tabela 4.1. Além disso, a Tabela 4.2 e a Figura 4.1 mostram o resultado da Experiência Numérica 2, evidenciando um aumento de entropia no processo de filtragem, dessa forma a inserção de incertezas. Por fim, a Tabela 4.3 mostra os resultados da experiência numérica 3, relacionando o aumento da ordem do filtro, logo adicionando operações de aritméticas, com o cálculo da entropia.

4.1 Experimento Numérico I

Com os resultados do experimento I, através da Tabela 4.1, o aumento do nível de desordem ocorreu em todos os casos se comparado com à entropia do sinal original, H = 4,9255. Neste mesmo experimento fica evidente que o aumento do número de bits não é garantia para o nível mais baixo de desordem, por exemplo, no caso do filtro Butterworth todas as simulações de 32 bits tiveram entropia maior que no 16 bits, com desvio padrão também maior.

Tabela 4.1: Resultados da Experiência Numérica 1. Cálculo de entropia para Butterworth, Chebyshev e filtro elíptico. Aplicamos nosso teste em 50 tipos de filtros em duas palavras (WL): 16 e 32 bits. A média μ o desvio padrão σ dos 50 resultados são mostrados. Valores de σ mostrados como 0,0000 significa que os valores calculados são inferiores a 0,00005. A medida da entropia para o sinal original é $H = 4{,}9255$. A partir deste resultado, é evidente o aumento da entropia medida em todos os sinais filtrados.

Tipos de Filtros	WL	Butterworth	Chebyshev	Elliptic
Passa Baixa	16	5,3276 ± 0,0000	5,3256 ± 0,0099	5,3775 ± 0,1639
	32	5,5451 ± 0,0487	5,3529 ± 0,0176	5,3297 ± 0,0055
Passa Alta	16	5,3266 ± 0,0070	5,3266 ± 0,0072	5,3290 ± 0,0068
	32	5,7350 ± 0,0594	5,3287 ± 0,0060	5,3297 ± 0,0058
Passa Faixa	16	5,3276 ± 0,0000	5,3266 ± 0,0070	5,3276 ± 0,0000
	32	5,3726 ± 0,0302	5,3371 ± 0,0156	5,3284 ± 0,0031
Rejeita Faixa	16	5,3266 ± 0,0072	5,3276 ± 0,0000	5,3266 ± 0,0070
	32	5,4125 ± 0,0410	5,3314 ± 0,0079	5,3287 ± 0,0048

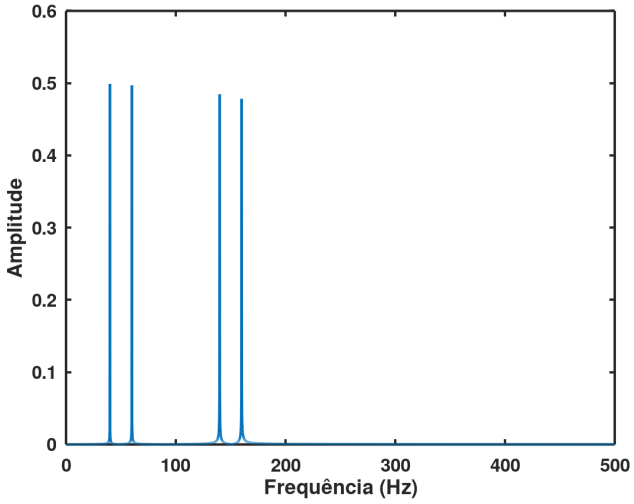
4.2 Experimento Numérico II

No experimento número II, a entropia do sinal original se mantém maior que o sinal filtrado pelo filtro ideal com desvio padrão também baixo. Mas a entropia aumenta em todos os sinais de entrada, tal característica evidencia a influência do processo de filtragem. Mas a entropia não varia proporcionalmente com a adição de componentes de frequência. Além disso, a Figura 4.1 mostra que o processo de filtragem também afetou a amplitude do sinal.

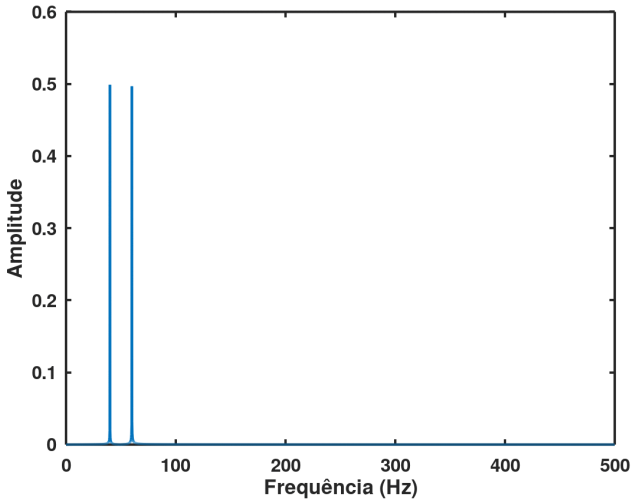
Tabela 4.2: Entropia do sinal original, o sinal simulado sem os componentes de frequência e o sinal filtrado. A terceira coluna é uma filtragem ideal. Como esperado, a entropia é reduzida. O mesmo não ocorre com o uso de filtro projetado com base no tipo elíptico. Nestes testes foram utilizados 32 bits. Resultados semelhantes foram obtidos para Butterworth e Chebyshev. A média e o desvio padrão foram calculados em 50 execuções para comprimentos de 1024 a 6024 amostras do sinal.

Sinal	Original	Filtro Ideal	Elliptic
1	6,5658 ± 0,0006	6,5435 ± 0,0004	7,9828 ± 0,0240
2	6,5131 ± 0,0012	6,4839 ± 0,0009	7,9586 ± 0,0235
3	6,5654 ± 0,0008	6,5434 ± 0,0004	7,9568 ± 0,0239

(a)



(b)



(c)

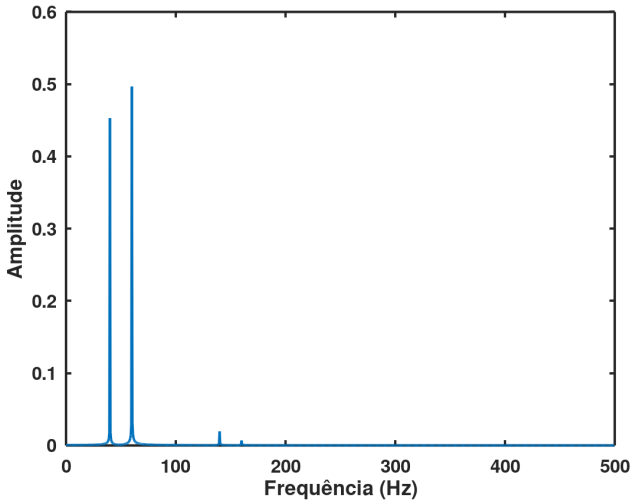


Figura 4.1: (**a**) FFT do sinal 2, Tabela 3.3—Sinal completo; (**b**) FFT do sinal 2 na Tabela 3.3—Sinal filtrado ideal; (**c**) FFT do filtro Chebyshev. A FFT calcula a similaridade esperada entre os sinais. Esse é outro ponto que torna relevante investigar o efeito do filtro digital na entropia do sinal filtrado.

4.3 Experimento Numérico III

No experimento 3, a ordem do filtro é alterada, logo com a adição de polos e zeros maior será o número de operações aritméticas para filtrar o sinal. Dessa forma, tal correlação entre o aumento de operações e a entropia só foi encontrada no filtro Elíptico com valor de $p = 0{,}030$.

Tabela 4.3: Cálculo de entropia com variação de ordem para três tipos de filtros. Utilizamos o software PSPP para realizar a análise de regressão e significância. Uma correlação positiva significativa entre os tipos de ordem e filtro é encontrada apenas para o filtro elíptico. Embora exista $r = 0.7$ para Chebyshev, seu valor $p = 0.052$, o que significa que não há significância estatística. A correlação é significativa no nível 0,05 (bicaudal) para elíptico com valor de p igual a 0,030. Realizamos nossos cálculos usando comprimento de palavra de 32 bits. Os valores são em média de 50 execuções para diferentes períodos de séries temporais em 1024 a 6024 amostras.

Order	**Butterworth**	**Chebyshev**	**Elliptic**
1	6,6458 ± 0,0006	6,6458 ± 0,0006	6,6458 ± 0,0006
2	6,6458 ± 0,0006	6,6458 ± 0,0006	6,6458 ± 0,0006
3	6,6458 ± 0,0006	6,6470 ± 0,0009	6,6458 ± 0,0006
4	6,6458 ± 0,0006	6,6458 ± 0,0006	6,6458 ± 0,0006
5	6,6458 ± 0,0006	6,6458 ± 0,0006	6,6458 ± 0,0006
6	6,6458 ± 0,0006	6,6458 ± 0,0006	6,6508 ± 0,0026
7	6,6458 ± 0,0006	6,6533 ± 0,0012	11,6121 ± 0,7048
8	6,6458 ± 0,0006	6,6683 ± 0,0056	11,6121 ± 0,7048
r	–	0,700	0,760
p-value	–	0,052	0,030

CAPÍTULO 5

CONCLUSÕES

Este trabalho investigou o efeito da precisão finita na entropia de sinais resultantes de filtros digitais IIR. Isso permite quantificar a introdução de ruído devido à ação de tais filtros. Mostramos que a entropia é uma boa alternativa para identificar a presença de ruído. Apresentou um resultado melhor do que a relação sinal/ruído para uma pequena quantidade de variação. Para observar os efeitos da entropia em sinais filtrados, projetamos três experimentos numéricos. Na Experiência Numérica 1, evidenciamos o aumento da entropia em todos os tipos de filtros investigados (Butterworth, Chebyshev e elíptico) por 16 e 32 bits. A entropia do sinal de entrada é $H = 4{,}9255$, enquanto em todo o sinal filtrado a entropia é $H > 5{,}32$. Não é o que se espera de um filtro linear ideal (consulte [45]). Devemos notar, de acordo com a Tabela 3.2, que o aparelho elíptico foi configurado com a ordem mais baixa. Mesmo em tais circunstâncias, esse tipo de filtro mostrou praticamente o mesmo nível de entropia no sinal filtrado.

Os resultados da Experiência Numérica II são mostrados na Tabela 4.2. Nesse caso, um filtro ideal é simulado retirando alguns dos componentes de frequência do sinal. A entropia do sinal filtrado aumentou significativamente, variando de 6,5 a quase 8. A Figura 4.1 mostra a FFT dos sinais. É possível notar uma pequena diferença entre as subfiguras (b) e (c), com isso, interpreta-se que o processo de filtragem também afetou a amplitude do sinal em uma frequência selecionada.

Na Experiência Numérica III, notamos outra característica, conforme descrito na Tabela 4.3. Essa experiência mostra uma correlação positiva significativa no nível 0,05 (bicaudal) para elíptica com valor de p igual a 0,030. A partir dessas experiências, parece claro que o filtro elíptico introduz mais incerteza, isto é, entropia, ao sinal filtrado quando comparado aos filtros Butterworth e Chebyshev.

As observações feitas neste manuscrito são coerentes com as apresentadas por DeBrunner et al. [48]. Como estamos concentrando nossa atenção no ruído da fonte fornecido por operações aritméticas (veja a Figura 3.1), estratégias de projeto que buscam maneiras mais eficientes de implementar expressões matemáticas podem ser úteis para reduzir a entropia.

Em trabalho futuro, pretende-se testar diferentes topologias de filtro (direta ou em cascata, por exemplo) para verificar sua influência no aumento da entropia, como é feito neste manuscrito. Isso parece um caminho bastante razoável, pois a ordem está relacionada ao aumento do número de operações matemáticas, que é uma fonte bem conhecida do ruído. Verificar a influência da taxa de amostragem e o número de amostras no cálculo da entropia. Além disso, investigar a relação da entropia com instabilidade em filtros, esta perspectiva é evidente com a realização do Experiência Numérica 3, pois antes do filtro se tornar instável sua entropia já era máxima. Por fim, a diferença observada na Figura 4.1 relaciona o processo de filtragem e limitação computacional com a amplitude do sinal filtrado, perspectiva ainda pouco explorada.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] E. Lai, *Practical digital signal processing*. Elsevier, 2003.

[2] V. Giurgiutiu, *Structural health monitoring: with piezoelectric wafer active sensors*. Elsevier, 2007.

[3] A. Antoniou, *Digital signal processing*. McGraw-Hill, 2016.

[4] I. Grout, *Digital systems design with FPGAs and CPLDs*. Elsevier, 2011.

[5] M. Wolf, *Embedded System Interfacing: Design for the Internet-of-Things (IoT) and Cyber-Physical Systems (CPS)*. Morgan Kaufmann, 2019.

[6] S. K. Mitra e Y. Kuo, *Digital signal processing: a computer-based approach*. McGraw-Hill New York, 2006, vol. 2.

[7] D. R. Frey, "Chaotic digital encoding: An approach to secure communication," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 10, pp. 660–666, 1993.

[8] S. M. Kuo e D. R. Morgan, *Active noise control systems*. Wiley, New York, 1996, vol. 4.

[9] A. T. Johns e S. K. Salman, *Digital protection for power systems*. IET, 1995.

[10] E. N. Bruce, *Biomedical signal processing and signal modeling*. Wiley New York:, 2001.

[11] R. J. Cameron, C. M. Kudsia, e R. R. Mansour, *Microwave filters for communication systems: fundamentals, design, and applications*. John Wiley & Sons, 2018.

[12] P. Shukl e B. Singh, "Recursive digital filter based control for power quality improvement of grid tied solar pv system," in *2018 8th IEEE India International Conference on Power Electronics (IICPE)*. IEEE, 2018, pp. 1–6.

[13] K. C. Wu, *Power converters with digital filter feedback control*. Academic Press, 2016.

[14] E. Ozpolat, B. Karakaya, T. Kaya, e A. Gulten, "FPGA-based digital filter design for biomedical signal," in *2016 XII International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. IEEE, 2016, pp. 70–73.

[15] R. C. Daniels e V. Gregers-Hansen, "Code inverse filtering for complete sidelobe removal in binary phase coded pulse compression systems," in *IEEE International Radar Conference, 2005.* IEEE, 2005, pp. 256–261.

[16] P. Gaydecki, "A real time programmable digital filter for biomedical signal enhancement incorporating a high-level design interface," *Physiological Measurement*, vol. 21, no. 1, pp. 187–196, 2000.

[17] K. S. Kumar, B. Yazdanpanah, e P. R. Kumar, "Removal of noise from electrocardiogram using digital fir and iir filters with various methods," in *2015 International Conference on Communications and Signal Processing (ICCSP)*. IEEE, 2015, pp. 0157–0162.

[18] J. J. Patel, K. R. Parmar, e H. N. Mewada, "Design of fir filter for burst mode demodulator of satellite receiver," in *2016 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 2016, pp. 0686–0690.

[19] M. Cernak, A. Asaei, e A. Hyafil, "Cognitive speech coding: examining the impact of cognitive speech processing on speech compression," *IEEE Signal Processing Magazine*, vol. 35, no. 3, pp. 97–109, 2018.

[20] M. C. Saxena, M. V. Upadhyaya, H. K. Gupta, e A. Sharma, "Denoising of ECG signals using FIR & IIR filter: A performance analysis," in *Proceedings on International Conference on Emerg*, vol. 2, 2018, pp. 51–58.

[21] M. Liu, H. Hao, P. Xiong, F. Lin, Z. Hou, e X. Liu, "Constructing a guided filter by exploiting the butterworth filter for ecg signal enhancement," *Journal of Medical and Biological Engineering*, vol. 38, no. 6, pp. 980–992, 2018.

[22] X. Tian, H. Daigle, e H. Jiang, "Feature detection for digital images using machine learning algorithms and image processing," in *Unconventional Resources Technology Conference, Houston, Texas, 23-25 July 2018*. Society of Exploration Geophysicists, American Association of Petroleum, 2018, pp. 1562–1575.

[23] S. Sun, L. Xu, Z. Cao, A. Huang, e W. Yang, "Digital recursive demodulator based on Kalman filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 12, pp. 3138–3147, 2017.

[24] S. Wang, G. Wu, F. Su, e J. Chen, "Simultaneous microwave photonic analog-to-digital conversion and digital filtering," *IEEE Photonics Technology Letters*, vol. 30, no. 4, pp. 343–346, 2018.

[25] W.-K. Ling, *Nonlinear digital filters: analysis and applications*. Academic Press, 2010.

[26] B. Zhao, Y. Lian, A. M. Niknejad, e C. H. Heng, "A Low-Power Compact IEEE 802.15.6 Compatible Human Body Communication Transceiver With Digital Sigma–Delta IIR Mask Shaping," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 2, pp. 346–357, 2019.

[27] D. Darmawardana, S. Perera, D. Robinson, J. Meyer, M. Klatt, e U. Jayatunga, "A digital zero-phase filter for measuring high frequency emissions (supraharmonics) in electrical distribution networks," in *2018 Australasian Universities Power Engineering Conference (AUPEC)*. IEEE, 2018, pp. 1–6.

[28] V. Ariyarathna, V. A. Coutinho, S. Pulipati, A. Madanayake, R. T. Wijesekara, C. U. Edussooriya, L. T. Bruton, T. K. Gunaratne, e R. J. Cintra, "Real-time 2-d fir trapezoidal digital filters for 2.4 ghz aperture receiver applications," in *2018 Moratuwa Engineering Research Conference (MERCon)*. IEEE, 2018, pp. 350–355.

[29] B. P. V. Davidek, M. Antosova, "Finite word-length effects in digital state-space filters," *Radioengineering*, vol. 8, no. 4, pp. 7–10, 1999.

[30] B. Liu, "Effect of finite word length on the accuracy of digital filters–a review," *IEEE Transactions on Circuit Theory*, vol. 18, no. 6, pp. 670–677, 1971.

[31] H.-J. Butterweck, J. Ritzerfeld, e M. Werter, "Finite wordlength effects in digital filters," *AEÜ*, vol. 43, pp. 76–89, 1989.

[32] H.-M. Cheng e G.-C. Chiu, "Finite precision controller implementation-limitation on sample rate," in *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on*, vol. 1. IEEE, 2003, pp. 634–639.

[33] C. Mullis e R. Roberts, "Synthesis of minimum roundoff noise fixed point digital filters," *IEEE Transactions on Circuits and Systems*, vol. 23, no. 9, pp. 551–562, 1976.

[34] S. Hwang, "Minimum uncorrelated unit noise in state-space digital filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 4, pp. 273–281, 1977.

[35] C. M. Rader e B. Gold, "Digital filter design techniques in the frequency domain," *Proceedings of the IEEE*, vol. 55, no. 2, pp. 149–171, 1967.

[36] R. M. Goodall e B. Donoghue, "Very high sample rate digital filters using the δ operator," *IEE Proceedings G (Circuits, Devices and Systems)*, vol. 140, no. 3, pp. 199–206, 1993.

[37] S. Jones, R. Goodall, e M. Gooch, "Targeted processor architectures for high-performance controller implementation," *Control Engineering Practice*, vol. 6, no. 7, pp. 867–878, 1998.

[38] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[39] E. G. Nepomuceno, "Convergence of recursive functions on computers," *The Journal of Engineering*, vol. 2014, no. 10, pp. 560–562, 2014.

[40] E. G. Nepomuceno e S. A. M. Martins, "A lower bound error for free-run simulation of the polynomial NARMAX," *Systems Science & Control Engineering*, vol. 4, no. 1, pp. 50–58, 2016.

[41] E. Nepomuceno, S. Martins, G. Amaral, e R. Riveret, "On the lower bound error for discrete maps using associative property," *Systems Science & Control Engineering*, vol. 5, no. 1, pp. 462–473, 2017.

[42] F. L. Milani, W. R. Lacerda Júnior, S. A. M. Martins, e E. G. Nepomuceno, "Influência de softwares e sistemas operacionais na simulação de modelos dinâmicos não lineares," *XXI Congresso Brasileiro de Automática*, 2016.

[43] L. S. DeBrunner, "Reducing Complexity of FIR Filter Implementations for Low Power Applications," in *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*. IEEE, 2007, pp. 1407–1411.

[44] A. Mehrnia e A. N. Willson, "A Lower Bound for the Hardware Complexity of FIR Filters," *IEEE Circuits and Systems Magazine*, vol. 18, no. 1, pp. 10–28, 2018.

[45] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[46] R. Badii, G. Broggi, B. Derighetti, M. Ravani, S. Ciliberto, A. Politi, e M. A. Rubio, "Dimension increase in filtered chaotic signals," *Physical Review Letters*, vol. 60, no. 11, pp. 979–982, 1988.

[47] R. N. Madan, "Maximum entropy method and digital filter design," in *Maximum Entropy and Bayesian Methods*. Springer Netherlands, 1993, pp. 49–54.

[48] V. E. DeBrunner, L. S. DeBrunner, S. Coone, e Xiaojuan Hu, "Using entropy to build efficient fir digital filters," in *3rd IEEE Signal Processing Education Workshop. 2004 IEEE 11th Digital Signal Processing Workshop, 2004.*, 2004, pp. 97–101.

[49] A. Venetsanopoulos, B. Mertzios, e S. Mneney, "Effects of finite precision in two-dimensional recursive digital filters," *International journal of electronics*, vol. 58, no. 1, pp. 159–174, 1985.

[50] C. M. Rader e B. Gold, "Effects of parameter quantization on the poles of a digital filter," *Proceedings of the IEEE*, vol. 55, no. 5, pp. 688–689, 1967.

[51] T. Claasen, W. Mecklenbrauker, e J. Peek, "Effects of quantization and overflow in recursive digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 6, pp. 517–529, 1976.

[52] C. Mullis e R. Roberts, "Roundoff noise in digital filters: Frequency transformations and invariants," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 6, pp. 538–550, 1976.

[53] A. C. Davies, "Nonlinear oscillations and chaos from digital filter overflow," *Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences*, vol. 353, no. 1701, pp. 85–99, 1995.

[54] W. R. Bennett, "Spectra of quantized signals," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 446–472, 1948.

[55] B. Eckhardt, "On the roundoff error of a multiplier," *ArEIU*, vol. 29, pp. 162–164, 1975.

[56] L. S. D. Victor E. DeBrunner, "Using entropy to build efficient FIR digital filters," in *3rd IEEE Signal Processing Education Workshop. 2004 IEEE 11th Digital Signal Processing Workshop, 2004.* IEEE, 2004, pp. 97–101.

[57] V. S. Borges, E. G. Nepomuceno, C. A. Duque, e D. N. Butusov, "Some remarks about entropy of digital filtered signals," *Entropy*, vol. 22, no. 3, p. 365, 2020.

[58] V. Borges, E. G. Nepomuceno, A. V. Tutueva, A. I. Karimov, C. Duque, e T. I. Karimov, "Analysis of iir filters by interval response," in *2020 Moscow Workshop on Electronic and Networking Technologies (MWENT)*. IEEE, 2020, pp. 1–5.

[59] L. Tan e J. Jiang, *Digital signal processing: fundamentals and applications*. Academic Press, 2018.

[60] B. Porat, *A course in digital signal processing*. Wiley New York, 1997, vol. 1.

[61] P. S. Diniz, E. A. Da Silva, e S. L. Netto, *Digital signal processing: system analysis and design*. Cambridge University Press, 2010.

[62] S.-Y. Peng, Y.-H. Lee, T.-Y. Wang, H.-C. Huang, M.-R. Lai, C.-H. Lee, e L.-H. Liu, "A power-efficient reconfigurable OTA-C filter for low-frequency biomedical applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 2, pp. 543–555, 2018.

[63] W. Barkouti, L. Salhi, e A. Chérif, "Digital audio watermarking using psychoacoustic model and cdma modulation," *Signal & Image Processing: An International Journal (SIPIJ) Vol*, vol. 2, 2011.

[64] Y. Lenaphet e P. Meemon, "The implementation of digital filter on FPGA for the spectral fusing gabor domain optical coherence microscopy," in *2020 8th International Electrical Engineering Congress (iEECON)*. IEEE, 2020, pp. 1–4.

[65] K. Thesni, K. Praveen, e L. Srivani, "Implementation and performance comparison of digital filter in FPGA," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 2020, pp. 589–594.

[66] A. V. Oppenheim e C. J. Weinstein, "Effects of finite register length in digital filtering and the fast fourier transform," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 957–976, 1972.

[67] S. Butterworth *et al.*, "On the theory of filter amplifiers," *Wireless Engineer*, vol. 7, no. 6, pp. 536–541, 1930.

[68] S. K. Mitra e Y. Kuo, *Digital signal processing: a computer-based approach*. McGraw-Hill New York, 2006, vol. 2.

[69] A. V. Oppenheim, *Discrete-time signal processing*. Pearson Education India, 1999.

[70] R. Ligrone, *Biological Innovations that Built the World: A Four-billion-year Journey Through Life and Earth History*. Springer, 2019.

[71] S. W. Golomb, E. Berlekamp, T. M. Cover, R. G. Gallager, J. L. Massey, e A. J. Viterbi, "Claude Elwood Shannon," *Notices of the AMS*, vol. 49, no. 1, 2002.

[72] O. Rioul, "This is it: A primer on Shannon's entropy and information," *L'Information, Seminaire Poincare*, vol. 23, pp. 43–77, 2018.

[73] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Electrical Engineering*, vol. 57, no. 12, pp. 713–723, 1938.

[74] H. Gardner, *The mind's new science: A history of the cognitive revolution*. Basic books, 1987.

[75] C. E. Shannon, "An algebra for theoretical genetics," Tese de Doutorado, Massachusetts Institute of Technology, 1940.

[76] H. Nyquist, "Certain factors affecting telegraph speed," *Transactions of the American Institute of Electrical Engineers*, vol. 43, pp. 412–422, 1924.

[77] R. V. Hartley, "Transmission of information 1," *Bell System technical journal*, vol. 7, no. 3, pp. 535–563, 1928.

[78] C. Shannan e W. Weaver, "The mathematical theory of communication," *Urbana, USA*, vol. 117, 1963.

[79] S. Wan, X. Zhang, e L. Dou, "Shannon entropy of binary wavelet packet subbands and its application in bearing fault extraction," *Entropy*, vol. 20, no. 4, p. 260, 2018.

[80] J. P. Noonan e P. Basu, "Information Theory Preliminaries," in *Signal and Image Restoration: Information-Theoretic Approaches*. 1000 20th Street, Bellingham, WA 98227-0010 USA: SPIE, 2011, pp. 3–10.

[81] E. C. Cherry, "A history of the theory of information," *Proceedings of the IEE-Part III: Radio and Communication Engineering*, vol. 98, no. 55, pp. 383–393, 1951.

[82] J. R. Paviotti *et al.*, "Considerações sobre o conceito de entropia na teoria da informação," *Tese (Mestrado em Tecnologia) – Universidade Estadual de Campinas, Faculdade de Tecnologia. Limeira - SP*, 2019.

[83] B. Randell, *The origins of digital computers: selected papers*. Springer, 2013.

[84] M. L. Overton, *Numerical computing with IEEE floating point arithmetic*. Siam, 2001, vol. 76.

[85] S. Boldo, F. Faissole, e V. Tourneur, "A formally-proved algorithm to compute the correct average of decimal floating-point numbers," in *2018 IEEE 25th Symposium on Computer Arithmetic (ARITH)*. IEEE, 2018, pp. 69–75.

[86] J.-J. Fernández, I. Garcıa, e E. M. Garzón, "Floating point arithmetic teaching for computational science," *Future Generation Computer Systems*, vol. 19, no. 8, pp. 1321–1334, 2003.

[87] L. Titolo, C. A. Muñoz, M. A. Feliú, e M. M. Moscato, "Eliminating unstable tests in floating-point programs," in *International Symposium on Logic-Based Program Synthesis and Transformation*. Springer, 2018, pp. 169–183.

[88] S. Boldo e G. Melquiond, *Computer Arithmetic and Formal Proofs: Verifying Floating-point Algorithms with the Coq System*. Elsevier, 2017.

[89] N. K. Bose e K. Kim, "Boundary implications for frequency response of interval FIR and IIR filters," *IEEE Transactions on signal processing*, vol. 39, no. 10, pp. 2167–2173, 1991.

[90] J. W. Eaton, D. Bateman, e S. Hauberg, *Gnu Octave*. Network Thoery London, 1997.

[91] A. W. Jayawardena, P. Xu, e W. K. Li, "Modified correlation entropy estimation for a noisy chaotic time series," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 2, p. 023104, 2010.

[92] Y. Luo, M. Du, e J. Liu, "A symmetrical image encryption scheme in wavelet and time domain," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 20, no. 2, pp. 447–460, 2015.

[93] A. Beghdadi e A. Khellaf, "A noise-filtering method using a local information measure," *IEEE Transactions on Image Processing*, vol. 6, no. 6, pp. 879–882, 1997.

[94] T. Figlus, J. Gnap, T. Skrúcaný, B. Šarkan, e J. Stoklosa, "The use of denoising and analysis of the acoustic signal entropy in diagnosing engine valve clearance," *Entropy*, vol. 18, no. 7, p. 253, 2016.

[95] L. G. Nardo, E. G. Nepomuceno, J. Arias-Garcia, e D. N. Butusov, "Image encryption using finite-precision error," *Chaos, Solitons & Fractals*, vol. 123, pp. 69–78, 2019.

APÊNDICE A

ROTINAS COMPUTACIONAIS

Os algoritmos que foram elaborados ao longo deste trabalho foram desenvolvidos via Octave. No experimento I, a representação de ponto fixo é representado pela variável b. No experimento computacional II, a variável x representa o sinal de entrada, enquanto a variável $x2$ representa o sinal idealmente filtrado. Tais programas podem ser encontrados no site: `https://ufsj.edu.br/gcom`.

A.1 Experimento Computacional I - Filtro Butterworth

```
pkg load signal

% Sinal teste

t=0:0.001:15;
fs1=50; % Frequencia do sinal
fs2=75; % Frequencia do sinal
fs3=125; % Frequencia do sinal
fs4=150; % Frequencia do sinal
x=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs3)+sin(2*pi*t*fs4);

n=6; % Ordem
rp=1; % Decibeis de ondulacao de banda passante pico a pico
rs=20; % Decibeis de atenuacao da banda de parada abaixo do valor de pico da banda passante.
wc=100; % Frequencia de borda de banda passante
wp=(wc/500); % Frequencia de borda de banda passante normalizada (wc/500)
b=16; % Numero de bits

% parametros do filtro (n,rp,rs,wp,(low, high, pass, or stop))

[p,z] = ellip(n,rp,rs,wp,'low'); % parametros do filtro (ordem,wc normalizado(wc/500))

q=(2^(-b))/2; % numero quantizado

pin = p.-q; % Polos arredondado para baixo
psu = p.+q; % Polos arredondado para cima
```

```
zin = z.-q; % Zeros arredondado para cima
zsu = z.+q; % Zeros arredondado para cima

% Combinacao 1
pp(1,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) pin(6) pin(7)];
zz(1,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7)];
% Combinacao 2
pp(2,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) pin(6) psu(7)];
zz(2,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7)];
% Combinacao 3
pp(3,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) psu(6) psu(7)];
zz(3,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7)];
% Combinacao 4
...

for am=1:1:50

p=pp(am,:);
z=zz(am,:);

% Filtragem do sinal

for k=1:1:(length(z)-1)
y(k)=0;
end

for k=length(z):1:length(t)
v1=x(k-(length(z)-1):k);
y1=p*v1';

v2=y(k-(length(z)-1):k-1);
y2=fliplr(z(2:length(z)))*v2';

y(k)=y1-y2;
end
% Calculo da entropia

r11=y(10001:10001+2^10);
aux=r11-min(r11);
er1=ceil(aux./max(aux)*2^16);
entr(am)=myntropy(er1(:));

endfor

media=mean(entr)
```

```
desvio=std(entr)
```

A.2 Experimento Computacional I - Filtro Chebyshev

```
clear all
close all
clc

pkg load signal

% Sinal teste

t=0:0.001:15;
fs1=50; % Frequencia do sinal
fs2=75; % Frequencia do sinal
fs3=125; % Frequencia do sinal
fs4=150; % Frequencia do sinal
x=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs3)+sin(2*pi*t*fs4);

n=8; % Ordem
rp=2; % Decibeis de ondulacao de banda passante pico a pico
wc=100; % Frequencia de borda de banda passante
wp=(wc/500); % Frequencia de borda de banda passante normalizada (wc/500)
b=16; % Numero de bits

% parametros do filtro (n,rp,wp,(low, high, pass, or stop))

[p,z] = cheby1(n,rp,wp,'high');

q=(2^(-b))/2; % numero quantizado

pin = p.-q; % Polos arredondado para baixo
psu = p.+q; % Polos arredondado para cima

zin = z.-q; % Zeros arredondado para cima
zsu = z.+q; % Zeros arredondado para cima

% Combinacao 1
pp(1,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) pin(6) pin(7) pin(8) pin(9)];
zz(1,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7) zin(8) zin(9)];
% Combinacao 2
```

```
pp(2,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) pin(6) pin(7) pin(8) psu(9)];
zz(2,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7) zin(8) zin(9)];
% Combinacao 3
pp(3,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) pin(6) pin(7) psu(8) psu(9)];
zz(3,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7) zin(8) zin(9)];
% Combinacao 4
...

for am=1:1:50

p=pp(am,:);
z=zz(am,:);

% Filtragem do sinal

for k=1:1:(length(z)-1)
y(k)=0;
end

for k=length(z):1:length(t)
v1=x(k-(length(z)-1):k);
y1=p*v1';

v2=y(k-(length(z)-1):k-1);
y2=fliplr(z(2:length(z)))*v2';

y(k)=y1-y2;
end
% Calculo da entropia

r11=y(10001:10001+2^10);
aux=r11-min(r11);
er1=ceil(aux./max(aux)*2^16);
entr(am)=myntropy(er1(:));

endfor

media=mean(entr)
desvio=std(entr)
```

A.3 Experimento Computacional I - Filtro Elíptico

```
clear all
close all
clc

pkg load signal

% Sinal teste

t=0:0.001:15;
fs1=50; % Frequencia do sinal
fs2=75; % Frequencia do sinal
fs3=125; % Frequencia do sinal
fs4=150; % Frequencia do sinal
x=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs3)+sin(2*pi*t*fs4);

n=6; % Ordem
rp=1; % Decibeis de ondulacao de banda passante pico a pico
rs=20; % Decibeis de atenuacao da banda de parada abaixo do valor de pico da banda passante.
wc=100; % Frequencia de borda de banda passante
wp=(wc/500); % Frequencia de borda de banda passante normalizada (wc/500)
b=16; % Numero de bits

% parametros do filtro (n,rp,rs,wp,(low, high, pass, or stop))

[p,z] = ellip(n,rp,rs,wp,'high'); % parametros do filtro (ordem,wc normalizado(wc/500))

q=(2^(-b))/2; % Numero quantizado

pin = p.-q; % Polos arredondado para baixo
psu = p.+q; % Polos arredondado para cima

zin = z.-q; % Zeros arredondado para cima
zsu = z.+q; % Zeros arredondado para cima

% Combinacao 1
pp(1,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) pin(6) pin(7)];
zz(1,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7)];
% Combinacao 2
pp(2,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) pin(6) psu(7)];
zz(2,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7)];
% Combinacao 3
pp(3,:)=[pin(1) pin(2) pin(3) pin(4) pin(5) psu(6) psu(7)];
zz(3,:)=[zin(1) zin(2) zin(3) zin(4) zin(5) zin(6) zin(7)];
% Combinacao 4
...
```

```
for am=1:1:50

p=pp(am,:);
z=zz(am,:);

% Filtragem do sinal

for k=1:1:(length(z)-1)
y(k)=0;
end

for k=length(z):1:length(t)
v1=x(k-(length(z)-1):k);
y1=p*v1';

v2=y(k-(length(z)-1):k-1);
y2=fliplr(z(2:length(z)))*v2';

y(k)=y1-y2;
end
% Calculo da entropia

r11=y(10001:10001+2^10);
aux=r11-min(r11);
er1=ceil(aux./max(aux)*2^16);
entr(am)=myntropy(er1(:));

endfor

media=mean(entr)
desvio=std(entr)
```

A.4 Experimento Computacional II

```
pkg load signal

% Sinal teste

t=(0:0.001:90);
fs1=40; % Frequencia do sinal
fs2=60; % Frequencia do sinal
fs3=80; % Frequencia do sinal
```

```
fs4=130; % Frequencia do sinal
fs5=150; % Frequencia do sinal
fs6=170; % Frequencia do sinal
%x=sin(2*pi*t*fs1)+sin(2*pi*t*fs6);
x=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs5)+sin(2*pi*t*fs6);
%x=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs3)+sin(2*pi*t*fs4)+sin(2*pi*t*fs5)+sin(2*pi*t*
    fs6);


%x2=sin(2*pi*t*fs1);
x2=sin(2*pi*t*fs1)+sin(2*pi*t*fs2);
%x2=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs3);

n=5; % Ordem
rp=1; % Decibeis de ondulacao de banda passante pico a pico
rs=10; % Decibeis de atenuacao da banda de parada abaixo do valor de pico da banda passante.
wc=100; % Frequencia de borda de banda passante
wp=(wc/500); % Frequencia de borda de banda passante normalizada (wc/500)

% parametros do filtro (n,rp,rs,wp,(low, high, pass, or stop))

[p,z] = ellip(n,rp,rs,wp,'low'); % parametros do filtro (ordem,wc normalizado(wc/500))

[H,w] = freqz(p,z); % FT sem arredondamento

% Filtragem do sinal

for k=1:1:(length(z)-1)
y(k)=0;
end

for k=length(z):1:length(t)
v1=x(k-(length(z)-1):k);
y1=p*v1';

v2=y(k-(length(z)-1):k-1);
y2=fliplr(z(2:length(z)))*v2';

y(k)=y1-y2;
end

cont=0;

for d=1:1:50

r11=y(10001:10001+2^16+cont);
```

```
aux1=r11-min(r11);
er1=ceil(aux1./max(aux1)*2^32);
trop1(d)=myntropy(er1(:));

rx=x(10001:10001+2^10+cont);
aux2=rx-min(rx);
erx=ceil(aux2./max(aux2)*2^32);
tropx(d)=myntropy(erx(:));

rx2=x2(10001:10001+2^10+cont);
aux3=rx-min(rx2);
erx2=ceil(aux3./max(aux3)*2^32);
tropx2(d)=myntropy(erx2(:));

%trops=[ tropx' tropx2' trop1'];

cont=cont+100;

end

mtr1=mean(tropx);
dtr1=std(tropx);
result1=[mtr1 dtr1]

mtr2=mean(tropx2);
dtr2=std(tropx2);
result2=[mtr2 dtr2]

mtr3=mean(trop1);
dtr3=std(trop1);
result3=[mtr3 dtr3]
```

A.5 Experimento Computacional III - Filtro Butterworth

```
clear all
close all
clc

pkg load signal
```

```
% Sinal teste

t=0:0.001:20;
fs1=20; % Frequencia do sinal
fs2=60; % Frequencia do sinal
fs3=90; % Frequencia do sinal
fs4=150; % Frequencia do sinal
x=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs3)+sin(2*pi*t*fs4);

for d=1:1:12

n=d; % Ordem
wc=100; % Frequencia da banda passante
wp=(wc/500); % Frequencia da banda passante normalizada (wc/500)

% parametros do filtro (n,wp,(low, high, pass or stop))


[p,z] = butter(n,wp,'low'); % parametros do filtro (ordem,wc normalizado(wc/500))

[H,w] = freqz(p,z); % FT sem arredondamento

% Filtragem do sinal

for k=1:1:(length(z)-1)
y(k)=0;
end

for k=length(z):1:length(t)
v1=x(k-(length(z)-1):k);
y1=p*v1';

v2=y(k-(length(z)-1):k-1);
y2=fliplr(z(2:length(z)))*v2';

y(k)=y1-y2;
end

cont=0;

for am=1:1:50

r11=y(10001:10001+2^10+cont);
aux=r11-min(r11);
er1=ceil(aux./max(aux)*2^32);
trop1(am)= myntropy(er1(:));
```

```
cont=cont+100;

end

media(d)=mean(trop1);
desvio(d)=std(trop1);

end

[media' desvio']
```

A.6 Experimento Computacional III - Filtro Chebyshev

```
clear all
close all
clc

pkg load signal


t=0:0.001:20;
fs1=20; % Frequencia do sinal
fs2=60; % Frequencia do sinal
fs3=90; % Frequencia do sinal
fs4=150; % Frequencia do sinal
x=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs3)+sin(2*pi*t*fs4);

for d=1:1:12
n=d; % Ordem
rp=2; % Decibeis de ondulacao de banda passante pico a pico
wc=100; % Frequencia da banda passante
wp=(wc/500); % Frequencia da banda passante normalizada (wc/500)

% parametros do filtro (n,rp,wp,(low, high, pass, or stop))



[p,z] = cheby1(n,rp,wp,'low');

[H,w] = freqz(p,z); % FT sem arredondamento
```

```
% Filtragem do sinal

for k=1:1:(length(z)-1)
y(k)=0;
end

for k=length(z):1:length(t)
v1=x(k-(length(z)-1):k);
y1=p*v1';

v2=y(k-(length(z)-1):k-1);
y2=fliplr(z(2:length(z)))*v2';

y(k)=y1-y2;
end
cont=0;

for am=1:1:50

r11=y(10001:10001+2^10+cont);
aux=r11-min(r11);
er1=ceil(aux./max(aux)*2^32);
trop1(am)= myntropy(er1(:));

cont=cont+100;

end

media(d)=mean(trop1);
desvio(d)=std(trop1);

end

[media' desvio']
```

A.7 Experimento Computacional III - Filtro Elíptico

```
% Sinal teste

t=0:0.001:50;
fs1=20; % Frequencia do sinal
```

```
fs2=60; % Frequencia do sinal
fs3=80; % Frequencia do sinal
fs4=150; % Frequencia do sinal
x=sin(2*pi*t*fs1)+sin(2*pi*t*fs2)+sin(2*pi*t*fs3)+sin(2*pi*t*fs4);

for d=1:1:8
n=d; % Ordem
rp=1; % Decibeis de ondulacao de banda passante pico a pico
rs=20; % Decibeis de atenuacao da banda de parada abaixo do valor de pico da banda passante.
wc=100; % Frequencia de borda de banda passante
wp=(wc/500); % Frequencia de borda de banda passante normalizada (wc/500)

% parametros do filtro (n,rp,rs,wp,(low, high, pass, or stop))


[p,z] = ellip(n,rp,rs,wp,'low'); % parametros do filtro (ordem,wc normalizado(wc/500))

[H,w] = freqz(p,z); % FT sem arredondamento

% Filtragem do sinal

for k=1:1:(length(z)-1)
y(k)=0;
end

for k=length(z):1:length(t)
v1=x(k-(length(z)-1):k);
y1=p*v1';

v2=y(k-(length(z)-1):k-1);
y2=fliplr(z(2:length(z)))*v2';

y(k)=y1-y2;
end

cont=0;

for am=1:1:50

r11=y(10001:10001+2^10+cont);
aux=r11-min(r11);
er1=ceil(aux./max(aux)*2^32);
trop1(am)= myntropy(er1(:));

cont=cont+100;
```

```
end

media(d)=mean(trop1);
desvio(d)=std(trop1);

end

[media' desvio']
```