UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

RODRIGO RUTSATZ

DESENVOLVIMENTO DE UM PROGRAMA EM LINGUAGEM C

EQUALIZADOR E CONVERSOR DE ARQUIVOS WAVE PARA BWF

Porto Alegre

2014

RODRIGO RUTSATZ

DESENVOLVIMENTO DE UM PROGRAMA EM LINGUAGEM C

EQUALIZADOR E CONVERSOR DE ARQUIVOS WAVE PARA BWF

Trabalho de Conclusão de Curso, apresentado como requisito parcial para a Graduação em Engenharia Elétrica pelo Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul – DELET/UFRGS.

Orientador: Prof. Dr. Carlos Eduardo Pereira

Porto Alegre

2014

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

RODRIGO RUTSATZ

DESENVOLVIMENTO DE UM PROGRAMA EM LINGUAGEM C

EQUALIZADOR E CONVERSOR DE ARQUIVOS WAVE PARA BWF

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação" do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Carlos Eduardo Pereira, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Banca Examinadora:

Prof. Dr. Carlos Eduardo Pereira, UFRGS
Doutor pela Universität Stuttgart – Stuttgart, Alemanha

Prof. Dr. Marcelo Götz, UFRGS
Doutor pela Universität Paderborn – Paderborn, Alemanha

Eng. Eletricista Rafael Susin
Engenheiro pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Porto Alegre

2014

AGRADECIMENTOS

Gostaria de agradecer à todas as pessoas que tornaram a realização deste trabalho possível, o apoio da minha família e dos colegas que durante toda a faculdade se esforçaram juntos para chegar ao mesmo objetivo.

Agradeço pela orientação do Prof. Dr. Carlos Eduardo Pereira e à equipe da DSPro Áudio LTDA, que me apoiaram tanto nos quesitos técnicos quando morais.

RESUMO

Durante os últimos anos, o processamento digital de sinais vem cada vez mais se consolidando e dominando o mercado da geração e transmissão de arquivos de multimídia nos ramos de cinema, televisão e rádio. A padronização de processamento e geração de arquivos está ainda em desenvolvimento, e recentemente vem incluindo novas informações aos arquivos de áudio e vídeo para aprimorar o tratamento dos mesmos entre empresas do ramo. Neste trabalho será desenvolvido um programa, em linguagem C, equalizador genérico de arquivos WAVE que gera um arquivo padronizado de áudio, o BWF (Broadcast WAVE Format).

Palavras-chave: Equalizador. Conversor. WAVE. BWF (Broadcast Wave Format).

ABSTRACT

During the last few years, the processing of digital signals is dominating the market of broadcasting and production of cinema, television and radio data. The standards of processing and generating digital data for audio and video braodcasting has been adding new information to these files, aiming for easier treatment of these files between corporations. This work consists of developing a program, in C language, which equalizes WAVE files and generates a standard audio file, the BWF (Broadcast WAVE Format).

Keywords: Equalizer. Conversor. WAVE. BWF (Broadcast WAVE Format).

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1. Motivação

Nos últimos anos, foi visto que a maior quantidade de reclamações dos espectadores de rádio e televisão recorre de que a potência sonora dos programas não é padronizada, causando a necessidade do espectador de ajustar o volume de acordo com o conteúdo que passa. Propagandas, por exemplo, utilizam volume maior que programas comuns, pois ainda não há uma norma regendo esse assunto.

Devido ao grande número de reclamações, o grupo de transmissão europeu “EBU” – European Broadcast Union – tem se dedicado em padronizar diversos assuntos dentro da radiodifusão e do sistema de televisão, definindo informações adicionais que todos programas devem ter. A padronização da EBU dos arquivos de áudio vem se atualizando constantemente, visando empregar as necessidades do futuro, e é seguida por muitas empresas internacionais do ramo, pois existe a tendência da normalização desses arquivos.

A equalização de som é uma ferramenta vital no tratamento de áudio de radiodifusão e televisão, devido ao controle sobre a inteligibilidade do conteúdo. A geração de arquivos de áudio equalizados e padronizados possibilita, no futuro, a correção de potência sonora dos mesmos, tornando os programas de radiodifusão e televisão de qualidade superior à atual, ainda em desenvolvimento.

1.2. Objetivo

Esse trabalho visa unir as necessidades de equalização e formatação padrão de arquivos de áudio. Para isso, serão desenvolvidos diversos filtros para obter um controle crítico sobre a frequência do áudio a ser formatado. Será feito um estudo de qualidade dos filtros implementados e será gerado um arquivo padrão da EBU, o BWF – Broadcast WAVE Format (Formato WAVE de transmissão).

O padrão BWF possibilita a sincronização de arquivos de áudio e vídeo, além de informar as mais diversas potências do sinal encontradas ao longo do arquivo de som e visa cobrir questões de segurança, como pirataria.

A EBU fornece arquivos de calibração e verificação de alguns dados calculados pertencentes ao arquivo padrão BWF e informa, também, o erro máximo admitido para cada cálculo para estar dentro do padrão. Este trabalho tem como objetivo obter as diversas potências sonoras e picos do sinal de áudio de acordo com os limites estabelecidos.

1.3. Organização do trabalho

O trabalho foi dividido em cinco capítulos e estruturado da seguinte maneira:

- *Capítulo 2:* apresenta o embasamento teórico básico necessário para a compreensão do trabalho;
- *Capítulo 3:* apresenta trabalhos já existentes a respeito do assunto aqui tratado;
- *Capítulo 4:* apresenta a proposta do trabalho, o projeto conceitual e a implementação do programa em linguagem C;

- *Capítulo 5:* apresenta os resultados experimentais obtidos dos filtros e da padronização do arquivo BWF;
- *Capítulo 6:* apresenta as conclusões dos resultados obtidos a partir do estudo descrito nos capítulos anteriores.

2. EMBASAMENTO TEÓRICO

Nesta seção do trabalho será apresentado o embasamento teórico necessário que torna a implementação do programa possível. Serão discutidas questões de arquivos WAVE e BWF, linguagem C e seu funcionamento, a teoria de sinais e filtros, e o modelamento do calculo da potência de sinais.

2.1 Arquivos WAVE e BWF

Em 1991 as empresas “Microsoft” e “IBM” definiram arquivos do tipo “RIFF” – Resource Interchange File Format (Formato de Arquivo para Troca de Recursos). Este tipo de arquivo foi definido como tendo um cabeçalho necessário para o processamento dos dados que o seguem. Arquivos WAVE e BWF, os quais o programa desenvolvido neste trabalho processa, são do tipo “RIFF”, cuja estrutura será apresentada nesta seção.

Arquivos do tipo “RIFF” são mais utilizados para arquivos binários de multimídia e são formados por cabeçalhos e seus respectivos dados em cascata. A Figura 1 foi traduzida e editada da Referência [3] e mostra essa estrutura.

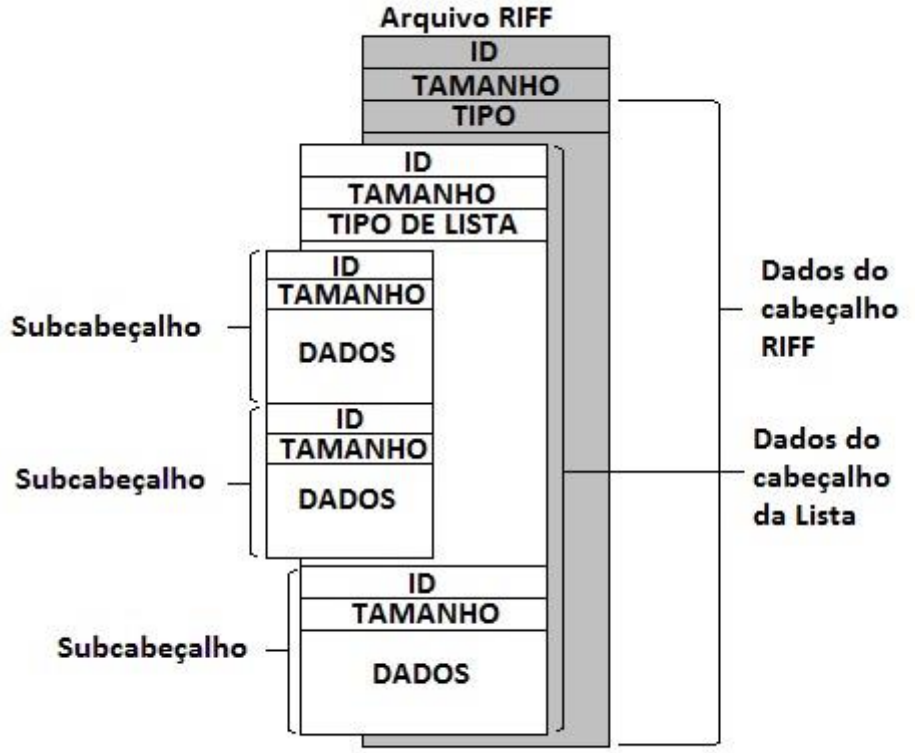

Figura 1: Estrutura de arquivos RIFF.

Os cabeçalhos que formam os arquivos “RIFF” são compostos de duas sequencias de quatro bytes padrão, que são a identificação em formato de quatro caracteres e do tamanho em bytes dos dados que seguem, representado como um inteiro longo.

A estrutura de cascata pode vir a ser dividida em muitos ramos, devido ao cabeçalho do tipo lista. Este cabeçalho contém quatro bytes de informação adicional que são quatro caracteres identificando o tipo da lista. Todos outros tipos de cabeçalho seguem o padrão.

O arquivo “RIFF” mostrado na Figura 1 contém uma lista com três sub-cabeçalho, entretanto, existem arquivos com cabeçalhos de lista que contém outros cabeçalhos tipo lista, aumentando a complexidade de leitura do arquivo.

Dentro dos arquivos do tipo “RIFF” os mais conhecidos são os formatos de vídeo “AVI” e de áudio “WAVE”. Os arquivos “WAVE”, por sua vez, são formados por dois cabeçalhos padrão, podendo conter outros além destes, que

usualmente incluem informações como o nome do artista, produtor, data de gravação, entre outras. Os cabeçalhos padrão são de formato e dados. A Figura 2 mostra a estrutura mais comum dos arquivos “WAVE” [2].

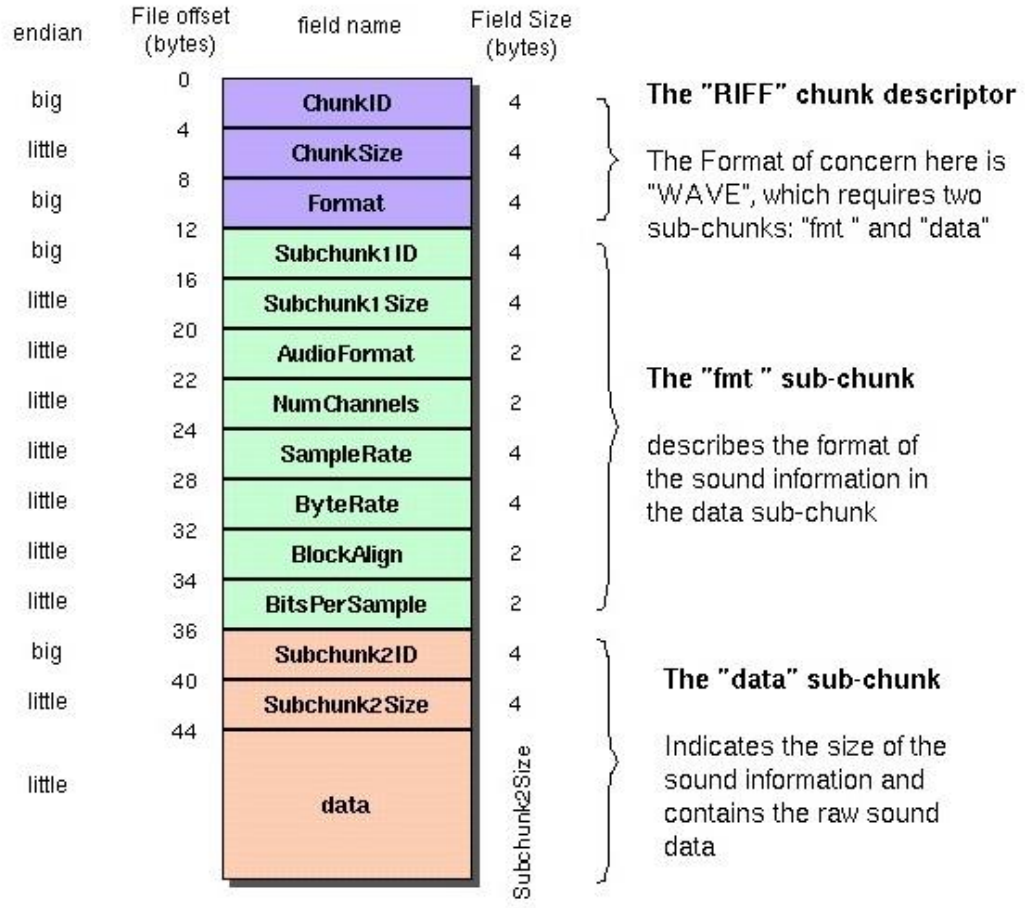

Figura 2: Estrutura do arquivo WAVE.

O primeiro cabeçalho identifica o arquivo “RIFF”, o tamanho do arquivo e o tipo do arquivo, “WAVE”. O primeiro sub-cabeçalho é do formato chamado de “fmt“. Este cabeçalho contém todas as informações necessárias para possibilitar a reprodução do arquivo de áudio, como se existe compressão, o número de canais, a frequência de amostragem, os bits por amostra, entre outros. O segundo e último sub-cabeçalho é dos dados do próprio áudio, chamado “data”.

Todos parâmetros mostrados na Figura 2 são obrigatórios e necessários para o processamento do arquivo “WAVE”. Os dados de áudio são fornecidos conforme a Figura 3[2].

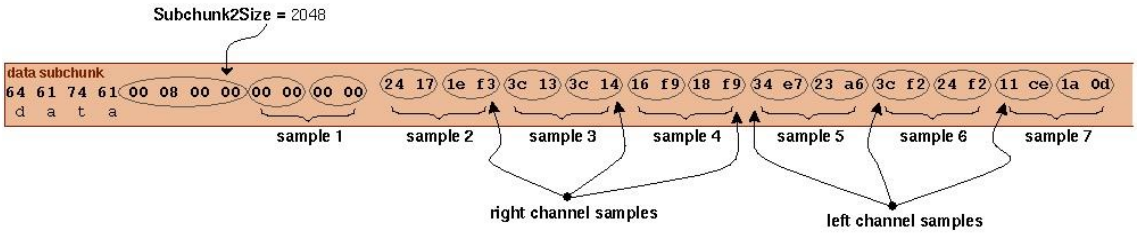

Figura 3: Estrutura dos dados de áudio em arquivos “WAVE”.

Os dados são entregues em blocos de amostras, onde cada bloco contém as amostras de cada canal. Para a melhor compreensão da estrutura, suponha um arquivo de áudio de dois canais. Os dados seriam entregues na ordem: Primeira amostra do primeiro canal, primeira amostra do segundo canal, segunda amostra do primeiro canal, segunda amostra do segundo canal, terceira amostra do primeiro canal, terceira amostra do segundo canal e assim por diante. Para um arquivo com mais canais seriam entregues as amostras número “i” de todos os canais, em ordem, antes de serem fornecidas as amostras “i+1” dos canais.

Os arquivos do tipo BWF (Broadcast WAVE Format) são quase idênticos aos arquivos do tipo WAVE, pois, além de serem arquivos RIFF, contém os cabeçalhos obrigatórios de formatação e dados. A diferença é que existe mais um cabeçalho obrigatório, chamado “bext”, broadcast-audio extension – extensão de áudio de transmissão. Neste cabeçalho são inseridas informações padronizadas para a troca desses arquivos entre usuários ou empresas de tratamento dos mesmos. Arquivos BWF, na verdade, tem a extensão de arquivos WAVE (“.wav”), pois são reproduzidos da mesma forma. Este tipo de arquivos foi criado pelo grupo “EBU”, European Broadcast Union – União de Transmissão Europeia, e é definido e detalhadamente descrito pela EBU Tech – 3285 [1]. A Figura 4 mostra a estrutura descrita dos arquivos BWF.

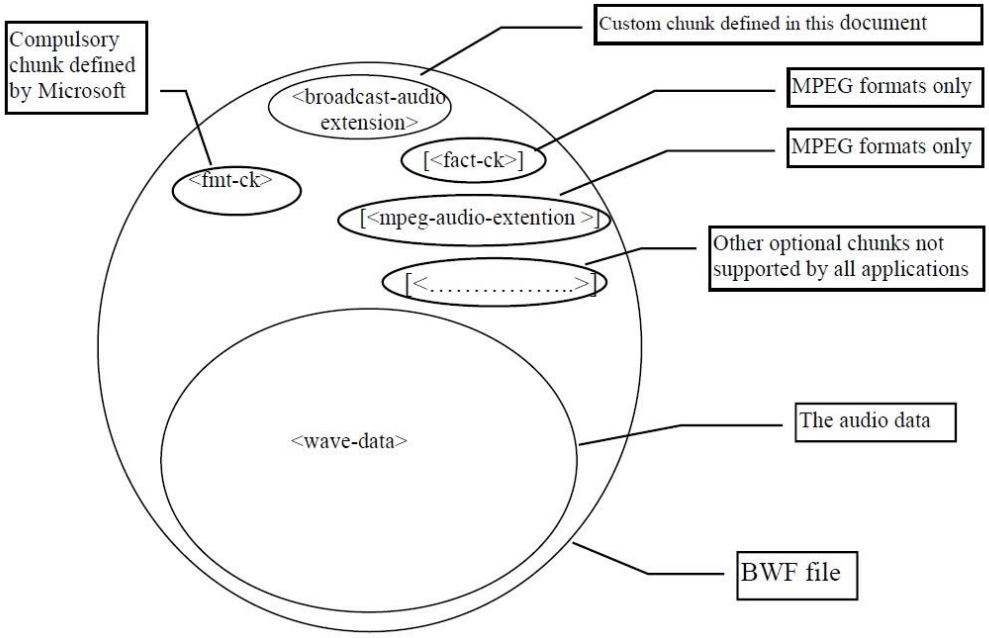

Figura 4: Estrutura de arquivos BWF.

Os cabeçalhos de formatos MPEG não serão descritos neste trabalho, pois o tratamento de áudio digital é feito apenas em arquivos PCM, que são sem compressão, ou seja, sem perda de informações. A compressão é a última etapa antes da transmissão da mídia, usualmente feita em MPEG-2 ou MPEG-4.

A extensão “bext” é formada pelas informações descritas na Tabela 1.

Tabela 1: Estrutura da extensão “bext”.

Informação	Descrição
ID	Contém os caracteres “bext”
Tamanho	Tamanho em bytes do cabeçalho
Descrição	Descrição da sequencia de som
Originador	Nome do originador
Referencia	Referencia do originador

Data	Data de criação
Horário	Horário de criação
Referência de Tempo – Byte menos significativo	Referência de tempo utilizada para a sincronização do arquivo de som. No formato de quantidade de amostras desde a meia noite.
Referência de Tempo – Byte mais significativo	
Versão	Versão do BWF
“UMID”	64 bytes para identificação única do arquivo (utilizado por questões de direitos e segurança)
Valor Sonoro	Similar à potência sonora
Faixa Sonora	Similar à faixa de potência sonora
Verdadeiro Pico de Nível	Valor de pico verdadeiro do áudio
Valor Sonoro Máximo Momentâneo	Similar à potência sonora máxima momentânea
Valor Sonoro Máximo de Curto Prazo	Similar à potência sonora máxima de curto prazo
Reservados	180 bytes reservados para mudanças futuras
Histórico de Código	Histórico de processamento do arquivo

Muitos dos parâmetros mencionados serão descritos com mais detalhe ao longo deste trabalho. Este padrão vem sendo atualizado constantemente, tendo sua segunda versão definida no fim do ano de 2011. O grupo EBU visa

uniformizar a transmissão de áudio de sistemas de televisão e radiodifusão e seus padrões são seguidos por mais de sessenta empresas internacionais.

Não serão descritos em detalhes outros cabeçalhos e informações que podem ser adicionados à arquivos WAVE e BWF, pois estes não tem impacto sobre este trabalho. Mais detalhes podem ser obtidos pelas referências [1], [2] e [3].

2.2 Linguagem C

A linguagem de programação C foi criada em 1972 por Dennis Ritchie no AT&T Bell Labs para o desenvolvimento do sistema operacional Unix. É uma linguagem compilada originalmente desenvolvida em Assembly e é uma das mais populares linguagens de programação.

Serão apresentadas apenas as funcionalidades da linguagem diretamente aplicadas nesse trabalho, explicadas á nível prático de aplicação e a nível baixo, mais próximo do de máquina, para a compreensão completa do sistema a ser implementado. Não serão explicados todos os assuntos da linguagem C que foram utilizados no desenvolvimento do programa, apenas os mais importantes e os que têm impacto direto sobre o resultado final do mesmo.

2.2.1 Declaração de dados

A declaração de dado na linguagem C é vital para qualquer aplicação. Quando uma variável é declarada são atribuídas características à mesma, como a forma que esta será lida, a sua quantidade de bytes e seu valor inicial. Variáveis declaradas globalmente (podem ser lidas em qualquer bloco do programa) são iniciadas nulas caso o programador não defina o valor inicial. Entretanto, para variáveis locais (de funções ou blocos específicos) isso não ocorre, sendo necessária a atribuição de valor explícita, podendo gerar erros no programa. Note o código de declaração na Figura 5:

```
1 int i;
2 unsigned int j;
3 long int k;
4 unsigned long int l;
5
```

Figura 5: Código para a explicação do funcionamento de ponteiros.

Todas varáveis declaradas são lidas como inteiros, entretanto, a forma como o processador efetuará a leitura das mesmas é única para cada uma delas. Na primeira declaração, como nada foi especificado, será efetuada a leitura padrão, que atribui sinal à variável e “short” em bytes (dois bytes). Assim, declarar “int i;” é equivalente a declarar “short signed int i;”. Sempre que não houver os termos de sinal e/ou de quantidade de bytes será efetuada essa leitura padrão.

Quando a variável “j” for lida, o processador lerá dois bytes não atribuindo sinal à variável. As variáveis “k” e “l” tem quatro bytes alocados na memória devido ao atributo “long”, onde “k” pode ser zero, positivo ou negativo e “l” apenas positivo ou zero.

A Tabela 2 mostra alguns dos tipos de declaração e seus respectivos atributos.

Tabela 2: Declaração de variáveis na linguagem C.

Tipo	Faixa de Valores	Tamanho (bits)
char	-127 a 127	8
unsigned char	0 a 255	8
signed char	-127 a 127	8
Int	-32767 a 32767	16
unsigned int	0 a 65535	16
signed int	-32767 a 32767	16
short int	-32767 a 32767	16
unsigned short int	0 a 65535	16

signed short int	-32767 a 32767	16
long int	-2147483647 a 2147483647	32
unsigned long int	0 a 4294967295	32
signed long int	2147483647 a 2147483647	32
float	3.4 E38 a -3.4 E38	32
double	1.7 E-308 a 1.7 E308	64
long double	3.4 E-4932 a 1.1 E4932	80

A declaração de variáveis, como mencionado, é vital dentro da aplicação. Quando se declara um contador inteiro, por exemplo, este tem seus valores mínimo e máximo especificados. Ou seja, ao utilizar um contador de dois bytes em uma estrutura de repetição que contenha um número de repetições maior que 65536, o programa nunca sairá do laço, pois quando o contador estiver no valor máximo e for incrementado, este retornará ao valor mínimo. Esse é um dos possíveis erros gerados por declaração de variáveis inapropriada.

Em trabalhos que o erro de quantização é importante, cada variável a ser utilizada deve ser premeditada, especialmente quando forem feitas operações aritméticas sobre a mesma. Também existe a questão de eficiência de memória, programas eficientes não utilizam variáveis com mais bytes que o necessário para sua devida aplicação.

Dentro de operações aritméticas, deve-se sempre ter em mente a faixa de valores a serem representados pelas variáveis para garantir que o valor final calculado seja verdadeiro. Por exemplo, uma variável “float” não representa números maiores que a ordem de dez elevado na trinta e oito. Ou seja, se durante o processamento do cálculo a variável alcançar tal valor, o resultado não será mais representativo, apesar de ser calculado.

Existem outros tipos de declarações, de constantes e de definições, que não serão tratados neste trabalho por não ter um efeito sobre o resultado final do projeto.

Na próxima seção serão apresentados os ponteiros na linguagem C e será aprofundado um pouco mais as questões qualitativas numéricas.

2.2.2 Ponteiros

Dentro das variadas linguagens de programação, os usuários usualmente estão acostumados a utilizar vetores ou matrizes para diversos objetivos. Na linguagem C, entretanto, não existem vetores nem matrizes, apenas ponteiros.

Um programa após ser compilado utiliza a memória RAM do computador para armazenar a maioria dos dados e variáveis declaradas (isso depende da forma da declaração). Os ponteiros estão alocados na memória e armazenam um endereço de memória, quando são incrementados, atualizam seu valor para o novo endereço de memória. A Figura 6 mostra duas variáveis, “a” e “b”, que estão localizadas nos endereços “0002” e “1008” respectivamente. A variável “a” é um ponteiro que contém o valor “1008”, assim, aponta para a variável “b”.

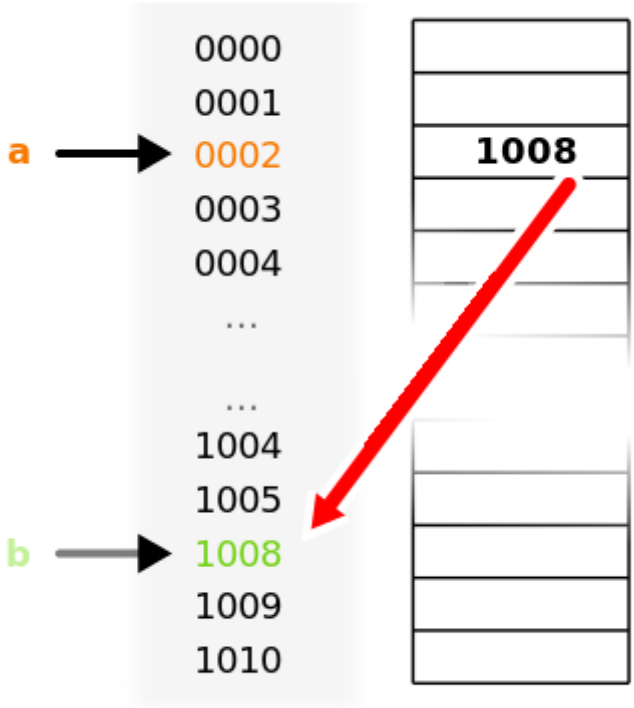

Figura 6: Endereçamento e valor de memória de ponteiros.

Na utilização de ponteiros dois símbolos são necessários: “*” e “&”. O código abaixo visa exemplificar essa utilização:

```
1 int *ponteiro=NULL;
2 int a;
3 ponteiro=&a;
4 *ponteiro=7;
5
```

Figura 7: Código para a explicação do funcionamento de ponteiros.

No código foi declarado um ponteiro que aponta para um objeto do tipo inteiro e foi inicializado nulo, isso é feito por precaução, pois não existe padrão implícito na declaração de ponteiros (apontarão para um endereço aleatório se não inicializados). Um objeto “a” do tipo inteiro é declarado. O ponteiro é, então, atualizado para apontar para o endereço de “a”, eis a utilização do símbolo “&”, que retorna o endereço do objeto. Na linha quatro do código, o

conteúdo do ponteiro é atribuído o valor sete, isso decorre da utilização de “*”, que faz o nexo do valor para o qual o ponteiro aponta.

O código que segue visa exemplificar a utilização de vetores e matrizes na linguagem C para a melhor compreensão do funcionamento a nível de máquina:

```
1 float vetor[100];
2 float matrix[3][3];
3 float *ptr_vet=NULL;
4 float **ptr_mat=NULL;
5 ptr_vet=(float*)malloc(100);
6 ptr_vet=(float**)malloc(3);
7 *ptr_vet=(float*)malloc(3);
8
```

Figura 8: Código para a explicação da relação de vetores e ponteiros.

A função “malloc()” é utilizada para alocar memória para ponteiros. Para isso é necessário determinar o tipo do ponteiro e a quantidade de memória a ser alocada. Essa função será descrita com mais detalhes em seções futuras nesse trabalho.

A primeira declaração no código é de um vetor de cem elementos e a segunda uma matriz quadrada de duas dimensões com três elementos em cada dimensão. A princípio parece que o usuário realmente está trabalhando com vetores e matrizes de dimensões definidas, entretanto, no nível de máquina isso não é verdade. O vetor e a matriz são ponteiros que apontam para a memória onde existe o número definido de elementos reservado e alocado para esses ponteiros.

O que o compilador faz nessa declaração de vetores é equivalente às linhas 3 a 7 do código da Figura 8, onde ponteiros são declarados e as memórias necessárias alocadas para eles. Em outras palavras, como já foi dito, não existem vetores na linguagem C, são ponteiros disfarçados. Isso fica muito claro quando se tenta trabalhar declarando um vetor com tamanho dependente de dados de entrada do programa, por exemplo, o tamanho de um arquivo a ser lido em bytes.

```
1 int tamanho_bytes;
2 int vetor[tamanho_bytes];
3
```

Figura 9: Código de declaração de vetor de tamanho variável.

Não será possível criar o vetor do código da Figura 9, pois “tamanho_bytes” é um valor variável. Assim, a quantidade de memória alocada pelo compilador de linguagem C não seria fixa, e o mesmo não foi desenvolvido para lidar com essa situação. Nesses casos é necessário alocar a memória explicitamente após saber a quantidade de bytes do exemplificado arquivo de entrada, chamada de memória dinâmica. Existem várias linguagens de programação que foram desenvolvidas para aceitar o tipo de tratamento da Figura 9, como C# por exemplo.

O trabalho com ponteiros é muito importante na leitura de arquivos e processamento matemático de variáveis, demandando do programador o controle sobre alocação de memória e trazendo o mesmo mais próximo ao que acontece em nível de máquina.

2.2.3 Estruturas e união de dados

Estruturas e união de dados são duas ferramentas da linguagem C muito utilizadas em leitura e escrita de arquivos. Estruturas são conjuntos de variáveis logicamente relacionadas e associadas a um nome. A Figura 10 mostra a declaração de uma estrutura utilizada para exemplificação.

```
1 struct Pessoa
2 {
3  char sexo[0];
4  short unsigned int idade=0;
5  float altura=0;
6 }
7
8 struct Pessoa X;
9
10 X.sexo='M';
11 X.idade=34;
12 X.altura=1.89;
13
```

Figura 10: Código de exemplo de estrutura.

Nas linhas um a seis do código está a declaração da estrutura “Pessoa”, que contém três variáveis, “Sexo”, “Idade” e “Altura”, que são caracteres, inteiro sem sinal de dois bytes e ponto flutuante, respectivamente. Na linha 8 é declarada uma variável “X” do tipo “Pessoa” e nas linhas dez a doze são atribuídos valores às diferentes variáveis da estrutura. Ao efetuar leituras ou escritas de arquivos binários, usualmente se salva os dados em uma estrutura e a escreve no arquivo binário. A ordem das variáveis escritas e lidas segue a ordem da declaração dentro da estrutura.

Outra ferramenta muito útil é a união de variáveis. Como foi mencionado na Seção 2.2.1 deste trabalho, variáveis de tipos diferentes são lidas diferentemente pelo processador. Assim, não seria possível ler um dado ponto flutuante como inteiro, pois o valor obtido não seria representativo. A união habilita o programador a fazer isso, indicando ao processador que tal dado pode ser lido como tipos diferentes definidos na declaração da união.

```
1 union
2 {
3 long signed int a;
4 float b;
5 }
6
```

Figura 11: Código de exemplo de união.

No código acima, as variáveis “a” e “b” ocupam o mesmo endereço de memória, entretanto, o processador lerá a variável “a” como inteiro com sinal de quatro bytes e “b” como ponto flutuante.

2.2.4 Argumentos “int argc” e “char argv[]”

Os argumentos “int argc” e “char argv[]” são utilizados como comunicação da linha de comando do sistema operacional com a função “main” do programa. São como comandos pré-programados ou variáveis de entrada,

dependendo da utilização. O argumento “argc” representa o número de dados que o usuário do programa forneceu e “argv[]” os valores de entrada correspondentes. Estes argumentos não serão descritos detalhadamente, entretanto, no fim da Seção 2.2 serão fornecidas referências onde o leitor poderá buscar mais informações sobre a linguagem C.

2.2.5 Funções da linguagem C

Nesta seção serão brevemente descritas as funções utilizadas na implementação do programa em linguagem C.

As funções apresentadas na Tabela 3 são pertencentes à biblioteca “stdlib.h”.

Tabela 3: Descrição de funções na linguagem C. Biblioteca “stdlib.h”.

Função	Descrição
fopen	Abre arquivo para a leitura/escrita na memória associando a este um ponteiro.
fclose	Fecha arquivo previamente aberto.
fread	Efetua leitura da localização do ponteiro de um arquivo aberto em uma variável.
fwrite	Efetua a escrita de uma variável na posição do ponteiro de um arquivo aberto.
fseek	Posiciona ponteiro do arquivo.
malloc	Aloca memória à uma variável.
realloc	Reajusta a dimensão de memória alocada à uma variável.

As funções na Tabela 4 pertencem à biblioteca “time.h”.

Tabela 4: Descrição de funções na linguagem C. Biblioteca “time.h”.

Função	Descrição
_strdate	Copia cadeia de caracteres da data definida no computador para uma variável.
_strtime	Copia cadeia de caracteres do horário definido no computador para uma variável.

2.2.6 Outras funcionalidades da linguagem C

Outra funcionalidade importante da linguagem C é o chamado “casting”. Essa ferramenta habilita o programador a converter dados de diferentes tipos e é extremamente simples de se usar, basta escrever entre parêntesis o tipo de dado que se quer converter a variável. A Figura 12 mostra um código de exemplo.

```
1 long signed int a;
2 float b;
3
4 b=(float)a;
5 a=(long signed int)b;
6
```

Figura 12: Código de exemplo de conversão de tipos de variáveis.

Nesta Seção 2.2 foram descritas breve e rapidamente as ferramentas mais utilizadas neste trabalho. Para mais informações a respeito da linguagem C, como laços, condicionamentos ou mais detalhes sobre os assuntos já tratados por exemplo, recorrer às referencias [4], [5] e [6].

2.3 Filtros

Nesta seção será apresentado o embasamento teórico de filtros digitais, incluindo os filtros do tipo FIR e IIR. Então, serão apresentados os equalizadores Shelving e Peak, que são baseados em filtros IIR.

Antes de prosseguir, será feita uma comparação entre filtros FIR e IIR:

- Filtros FIR têm resposta finita ao impulso (Finite Impulse Response), logo, são sempre estáveis. Em filtros IIR, esse não é o caso, pois estes têm resposta infinita ao impulso (Infinite Impulse Response), dessa forma é necessário ter o cuidado de que todos pólos e zeros se encontram no círculo unitário para garantir a estabilidade do filtro (critério de estabilidade de Nyquist).
- Filtros FIR não apresentam distorção de fase por terem fase linear. Este é outro cuidado que se deve ter com filtros IIR, verificar a influência do efeito da distorção de fase que é sempre gerada.
- Filtros FIR requerem maior quantidade de processamento que filtros IIR para uma dada resposta de corte de frequência.
- Filtros FIR podem ser implementados de acordo com o erro de quantização, o que não é possível em filtros IIR.

2.3.1 Filtros FIR

Nesta seção será apresentada a dedução da teoria de sinais dos filtros FIR implementados no programa.

O projeto é feito definindo a resposta em frequência do filtro desejada, então, aplica-se a transformada de Fourier para se obter a resposta no domínio do tempo do filtro. Como a resposta no tempo do filtro será de infinita duração, se passa os filtros FIR por uma janela, limitando os coeficientes não nulos, o que representa a ordem do filtro. Neste ponto é importante diferenciar ordem dos polinômios da resposta em frequência, que são todos de primeira ordem como será mostrado a seguir, do número de coeficientes do filtro, que também é chamado de ordem do filtro.

Serão apresentados filtros FIR passa-baixa, passa-alta, passa-banda e rejeita-banda, mas antes disso, serão descritas as janelas pelas quais os filtros serão definidos.

As janelas são definidas no domínio do tempo de acordo com a Equação (1).

$$w_k[n] = \begin{cases} b_k \; para \; n = 0,1,2,\dots,M \\ 0, caso \; contrário \end{cases} \quad (1)$$

onde *M* representa a ordem da janela e, logo, do filtro. Os coeficientes b_k são diferentes e definidos para cada janela. A resposta do filtro é dada pelas Equações (2) e (3).

$$h_k[n] = F^{-1}\{H(w)\} \quad (2)$$

$$h_{d_k}[n] = w_k[n] \times h_k[n] \quad (3)$$

As janelas utilizadas neste trabalho são as seguintes:

- Retangular, representada pela Equação (4).

$$w_{ret}[n] = \begin{cases} 1 & 0 \geq n \leq M \\ 0, caso \; contrário \end{cases} \quad (4)$$

- Hamming, representada pela Equação (5).

$$w_{ham}[n] = \begin{cases} 0.5 - 0.5cos\frac{2\pi n}{M+1} & 0 \geq n \leq M \\ 0, caso \; contrário \end{cases} \quad (5)$$

- Hanning, representada pela Equação (6).

$$w_{han}[n] = \begin{cases} 0.54 - 0.46cos\frac{2\pi n}{M+1} & 0 \geq n \leq M \\ 0, caso \; contrário \end{cases} \quad (6)$$

- Blackman, representada pela Equação (7).

$$w_{black}[n] = \begin{cases} 0.42 - 0.5cos\frac{2\pi n}{M+1} - 0.080.54 - 0.46cos\frac{4\pi n}{M+1} & 0 \geq n \leq M \\ 0, caso \; contrário \end{cases} \quad (7)$$

Existem outras janelas aplicáveis que não foram utilizadas neste trabalho. A Figura 13 mostra a resposta das janelas apresentadas [7].

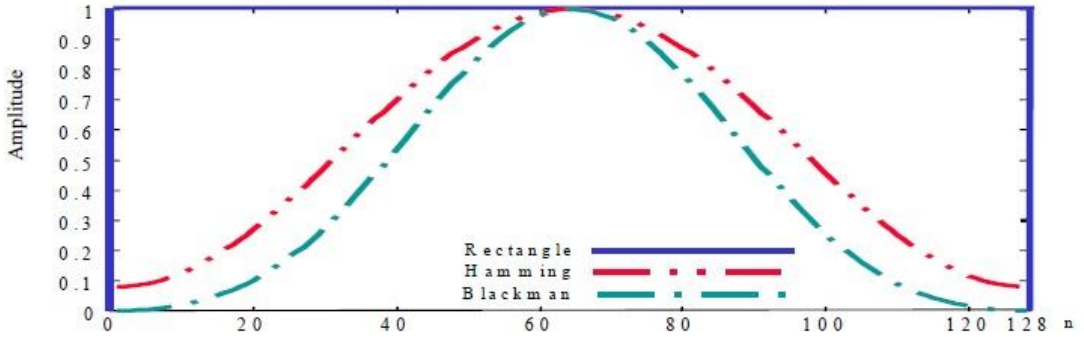

Figura 13: Resposta em frequência das janelas apresentadas.

Como mencionado os filtros foram projetados a partir da resposta em frequência desejada. Então, aplicou-se a transformada para obter a resposta no tempo do filtro, que será convoluída com o sinal de entrada para se obter o sinal filtrado. Essa dedução é mostrada a seguir, mas, antes, é necessário mencionar que para tornar a implementação dos filtros viável, deve-se tornar o filtro causal. A resposta no domínio tempo do filtro é infinita, como já explicado, assim aplica-se a janela para torna-la finita. A causalidade é introduzida por um deslocamento de $\frac{M}{2}$, de tal forma que o sinal tenha seu início em zero. Isto é visto nos termos sinusoidais como $n-\frac{M}{2}$. Outro detalhe importante, é que as frequências de corte das respostas no tempo dos filtros são normalizadas em relação à frequência de amostragem do sinal.

As Equações de (8) a (15) descrevem as respostas em frequência e no tempo, após ter sido aplicada a transformada de Fourier, dos filtros indicados.

Filtro FIR passa-baixa:

$$H_d(f) = \begin{cases} 1 & 0 \leq f \geq f_c \\ 0, caso\ contrário \end{cases} \tag{8}$$

$$h_d(n) = 2f_c \frac{sen\left(2\pi f_c\left(n-\frac{M}{2}\right)\right)}{2\pi f_c\left(n-\frac{M}{2}\right)} \tag{9}$$

Filtro FIR passa-alta:

$$H_d(f) = \begin{cases} 0 & 0 \leq f \geq f_c \\ 1, caso\ contrário \end{cases} \tag{10}$$

$$h_d(n) = (1-2f_c)\left(\frac{sen\left(\pi\left(n-\frac{M}{2}\right)\right)}{\pi\left(n-\frac{M}{2}\right)} - \frac{sen\left(2\pi f_c\left(n-\frac{M}{2}\right)\right)}{2\pi f_c\left(n-\frac{M}{2}\right)}\right) \tag{11}$$

Filtro FIR passa-banda:

$$H_d(f) = \begin{cases} 1 & f_b \leq f \geq f_a \\ 0, & caso\ contrário \end{cases} \quad (12)$$

$$h_d(n) = 2 \times (f_a - f_b)\left(\frac{sen\left(2\pi f_a\left(n-\frac{M}{2}\right)\right)}{2\pi f_a\left(n-\frac{M}{2}\right)} - \frac{sen\left(2\pi f_b\left(n-\frac{M}{2}\right)\right)}{2\pi f_b\left(n-\frac{M}{2}\right)}\right) \quad (13)$$

Filtro FIR rejeita-banda:

$$H_d(f) = \begin{cases} 0 & f_b \leq f \geq f_a \\ 1, & caso\ contrário \end{cases} \quad (14)$$

$$h_d(n) = \left(1 - 2 \times (f_a - f_b)\right)\left(\frac{sen\left(\pi\left(n-\frac{M}{2}\right)\right)}{\pi\left(n-\frac{M}{2}\right)} - \frac{sen\left(2\pi f_a\left(n-\frac{M}{2}\right)\right)}{2\pi f_a\left(n-\frac{M}{2}\right)} + \frac{sen\left(2\pi f_b\left(n-\frac{M}{2}\right)\right)}{2\pi f_b\left(n-\frac{M}{2}\right)}\right) \quad (15)$$

O sinal filtrado será, então, dado pela convolução entre a resposta no tempo do filtro pela janela e o sinal de entrada, e é representada na Equação (16).

$$y[m] = \sum_{n=0}^{M} h_d[n] * x[m-n] \quad (16)$$

O leitor pode encontrar mais informações sobre os filtros FIR utilizados neste trabalho na referência [7], entretanto, é vital informar, que existem diferenças das equações demonstradas nesta seção, particularmente a questão da amplitude dos sinais no domínio do tempo. Apesar disso, essa referência foi escolhida por ser a mais completa no assunto.

2.3.2 Filtros IIR

O projeto de filtros IIR é formado pela representação da resposta desejada no domínio da frequência. Diferentemente do caso dos filtros FIR, não será feita a convolução entre o sinal de entrada e a resposta do filtro, mas, sim, calculado diretamente com base na transformada Z, que no caso discreto representa o atraso do sinal.

Neste trabalho são utilizados todos possíveis filtros IIR de primeira e segunda ordem: passa-tudo, passa-baixa, passa-alta, passa-banda e rejeita-

faixa, que serão apresentados nesta seção. Só é possível projetar filtros passa-banda e rejeita-faixa em filtros IIR de no mínimo segunda ordem.

A construção dos filtros é feita a partir dos filtros IIR de primeira e segunda ordem passa-tudo. Primeiro será descrito o projeto dos filtros IIR de primeira ordem para então prosseguir para os de segunda ordem.

A resposta de um filtro IIR passa-tudo de primeira ordem é dada pela Equação (17), cujos coeficientes são definidos pela Equação(18).

$$A(Z) = \frac{z^{-1}+c}{1+cz^{-1}} \tag{17}$$

$$c = \frac{\tan(\pi f_c)-1}{\tan(\pi f_c)+1} \tag{18}$$

onde a frequência de corte é normalizada em relação à frequência de amostragem. A resposta no tempo de um sinal aplicado à função de transferência da Equação (17) é dada pela Equação (19).

$$y_f(n) = cx(n) + x(n-1) - cy(n-1) \tag{19}$$

Os filtros IIR passa-baixa e passa-alta são derivados do filtro passa-tudo pela Equação (20).

$$H(Z) = \frac{1}{2}(1 \pm A(Z)) \tag{20}$$

onde o filtro passa-baixa é obtido pela soma e o filtro passa-alta pela subtração da resposta do filtro passa-tudo, resultando na resposta final dada pela Equação (21).

$$y(n) = \frac{1}{2}\Big(x(n) \pm y_f(n)\Big) \tag{21}$$

Os filtros de segunda ordem são baseados no filtro passa-tudo de segunda ordem dado pela Equação (22).

$$A(Z) = \frac{-c+d(1-c)z^{-1}+z^{-2}}{1+d(1-c)z^{-1}-cz^{-2}} \tag{22}$$

onde os coeficientes são dados pelas Equações (23) e (24).

$$c = \frac{\tan(\pi f_b) - 1}{\tan(\pi f_b) + 1} \tag{23}$$

$$d = -\cos(2\pi f_c) \tag{24}$$

onde f_b representa a faixa de frequência de banda e f_c a frequência de corte do filtro, ambas normalizadas em relação à frequência de amostragem do sinal.

A resposta no tempo de um sinal aplicado ao filtro passa-tudo de segunda ordem é dada pela Equação (25).

$$y_f(n) = -cx(n) + d(1-c)x(n-1) + x(n-2) - d(1-c)y(n-1) + cy(n-2) \tag{25}$$

Os filtros passa-banda e rejeita-faixa são obtidos a partir do filtro passa-tudo de segunda ordem conforme a Equação (26).

$$H(Z) = \frac{1}{2}(1 \mp A(Z)) \tag{26}$$

onde o passa-banda é obtido pela subtração e o rejeita-faixa pela adição do filtro passa-tudo, resultando na resposta final dada pela Equação (27).

$$y(n) = \frac{1}{2}\Big(x(n) \pm y_f(n)\Big) \tag{27}$$

Os filtros passa-baixa e passa-alta de segunda ordem utilizados neste trabalho foram obtidos pela mesma derivação da transformada Z, entretanto, não a partir do filtro passa-tudo. Dessa forma, eles obedecem à Equação (28).

$$H(Z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} - a_2 z^{-2}} \tag{28}$$

onde os coeficientes são representados na Tabela 5, obtida da referência [8].

Tabela 5: Coeficientes dos filtros passa-baixa e passa-alta ordem dois.

Passa-Baixa de segunda ordem com $\mathrm{K} = \tan(\pi f_c)$				
b_0	b_1	b_2	a_1	a_2
$\frac{K^2}{1+\sqrt{2}K+K^2}$	$\frac{2K^2}{1+\sqrt{2}K+K^2}$	$\frac{K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$

Passa-Alta de segunda ordem com $\mathrm{K} = \tan(\pi f_c)$				
b_0	b_1	b_2	a_1	a_2
$\frac{1}{1+\sqrt{2}K+K^2}$	$\frac{-2}{1+\sqrt{2}K+K^2}$	$\frac{1}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$

A resposta de um sinal aplicado a esses filtros é então calculada de acordo com a Equação (29).

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) - a_1y(n-1) + a_2y(n-2) \quad (29)$$

2.3.3 Equalizadores

Equalizadores são conjuntos de filtros com ganho aplicados a sinais. Nesta seção serão detalhados os filtros Shelving e Peak. Filtros Shelving são filtros passa-baixa e passa-alta IIR com uma variável de ganho na implementação em dB. Os filtros Peak são os filtros passa-faixa e rejeita-faixa IIR com variável de ganho em dB adicionada. Note que os filtros Peak são de ordem mínima dois, já os filtros Shelving podem ser de qualquer ordem.

O conjunto de filtros Shelving passa-baixa e passa-alta com pelo menos um filtro Peak representam um equalizador básico. Em equalizadores de áudio de maior qualidade utiliza-se um conjunto de 28 filtros Peak, um passa-baixa Shelving e um passa-alta Shelving, tendo, assim, uma faixa para cada terço de oitava.

Os filtros Shelving passa-baixa e passa-alta de primeira ordem são regidos pelas Equações (30) a (34).

$$H(Z) = 1 + \frac{H_0}{2}(1 \pm A(Z)) \quad (30)$$

$$A(Z) = \frac{c+z^{-1}}{1+cz^{-1}} \quad (31)$$

$$c = \frac{\tan(\pi f_c) - 1}{\tan(\pi f_c) + 1} \tag{32}$$

$$H_0 = V_0 - 1 \tag{33}$$

$$V_0 = 10^{G/20} \tag{34}$$

onde *G* é o ganho desejado em dB e V_0 e H_0 os fatores de ganho aplicados.

A implementação destes filtros é feita de forma análoga aos IIR de primeira ordem, sendo modificado o último passo, que é a aplicação da Equação (30) à resposta do filtro passa-tudo, resultando na resposta final dos filtros Shelving apresentada na Equação (35).

$$y(n) = x(n) + \frac{H_0}{2}\Big(x(n) \pm y_f(n)\Big) \tag{35}$$

Os filtros Shelving de segunda ordem têm comportamento análogo aos filtros IIR passa-baixa e passa-alta de segunda ordem, com a integração dos fatores de ganhos nas equações. A Figura 14 mostra os coeficientes do filtro Shelving passa-baixa e a Figura 15 do filtro Shelving passa-alta com o ganho integrado, obtidas da referência [9].

Low-frequency shelving (boost $V_0 = 10^{G/20}$)				
a_0	a_1	a_2	b_1	b_2
$\frac{1+\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(V_0K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Low-frequency shelving (cut $V_0 = 10^{-G/20}$)				
a_0	a_1	a_2	b_1	b_2
$\frac{1+\sqrt{2}K+K^2}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{2(V_0K^2-1)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2V_0}K+V_0K^2}$

Figura 14: Coeficientes do filtro Shelving passa-baixa.

High-frequency shelving (boost $V_0 = 10^{G/20}$)				
a_0	a_1	a_2	b_1	b_2
$\frac{V_0+\sqrt{2V_0}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-V_0)}{1+\sqrt{2}K+K^2}$	$\frac{V_0-\sqrt{2V_0}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
High-frequency shelving (cut $V_0 = 10^{-G/20}$)				
a_0	a_1	a_2	b_1	b_2
$\frac{1+\sqrt{2}K+K^2}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{2(K^2-1)}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{2(K^2/V_0-1)}{1+\sqrt{2/V_0}K+K^2/V_0}$	$\frac{1-\sqrt{2/V_0}K+K^2/V_0}{1+\sqrt{2/V_0}K+K^2/V_0}$

Figura 15: Coeficientes do filtro Shelving passa-alta.

A resposta final desses filtros é regida pela Equação (29) em conjunto com os coeficientes recém apresentados.

Os filtros Peak deste trabalho são formados a partir do filtro passa-tudo IIR de segunda ordem e são definidos pelas Equações (36) a (40).

$$H(Z) = 1 + \frac{H_0}{2}(1 - A(Z)) \tag{36}$$

$$A(Z) = \frac{-c_{g/c} + d(1 - c_{g/c})z^{-1} + z^{-2}}{1 + d(1 - c_{g/c})z^{-1} - c_{g/c}z^{-2}} \tag{37}$$

$$c_g = \frac{\tan(\pi f_b) - 1}{\tan(\pi f_b) + 1} \tag{38}$$

$$c_c = \frac{\tan(\pi f_b) - V_0}{\tan(\pi f_b) + V_0} \tag{39}$$

$$d = -\cos(2\pi f_c) \tag{40}$$

onde V_0 e H_0 são os fatores de ganho já mencionados, Equações (30) e (31). Os coeficientes c_g e c_c correspondem aos filtros Peak de ganho ou de corte, respectivamente, e são utilizados substituindo $c_{g/c}$, de acordo com o efeito desejado.

A resposta final do filtro é dada conforme a Equação (41).

$$y_f(n) = -c_{g/c}x(n) + d(1 - c_{g/c})x(n-1) + x(n-2) - d(1 - c_{g/c})y(n-1) + c_{g/c}y(n-2) \tag{41}$$

Para mais informações sobre filtros IIR e equalizadores o leitor pode recorrer às referências [8] e [9].

3. TRABALHOS RELACIONADOS

Neste capítulo serão descritos alguns trabalhos relacionados a este trabalho.

Existem muitos trabalhos acadêmicos sobre filtros e equalizadores de mais variados tipos, entretanto, todos os trabalhos analisados realizavam a implementação dos filtros de estudo na ferramenta “MatLab”, onde o usuário define apenas a função transferência do filtro e plota os gráficos para fazer a análise dos resultados. No quesito de geração de arquivos BWF não foram encontrados trabalhos acadêmicos.

Foram encontrados diversos trabalhos comerciais relacionados, assim, serão apresentados alguns softwares, suas funções básicas serão descritas, analisadas e comparadas aos objetivos aqui propostos.

3.1 Wave Agent

Este software é disponível tanto para sistemas operacionais Windows como Macintosh, onde é possível ver e editar cabeçalhos de arquivos WAVE e BWF de forma quase completa (não é possível editar informações de potência, entre outras). Neste programa também é possível reproduzir os arquivos de áudio, mixa-los com controle de volume e faixa, entretanto, não existe nenhuma função de equalização dos arquivos de som.

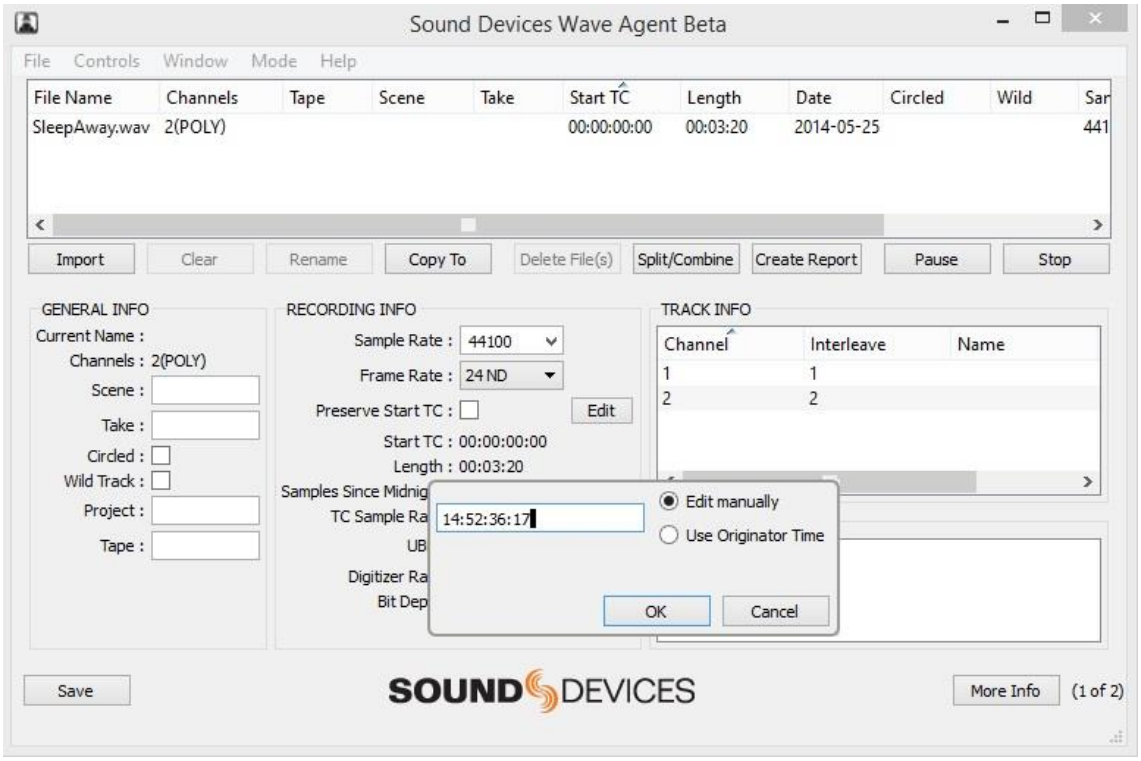

Figura 16: Janela principal do software Wave Agent.

Na Figura 16 é apresentada a tela principal do programa, onde está sendo editada a referencia de tempo para sincronização do cabeçalho “bext”, descrito no Capítulo 2 deste trabalho. Este programa, na verdade, é um freeware, ou seja, não tem custo. Existem muitos softwares similares à este mencionado, entretanto, não foi encontrado nenhum com equalização de som implementada e controle dos dados dos cabeçalhos ao ponto que é proposto por este trabalho.

3.2 soundBlade e Reaper

Existem muitos softwares de tratamento de áudio, que incluem inúmeras funções de efeitos como equalização, compressão, entre outros. Aqui serão apresentados softwares de nível profissional, como o soundBlade e Reaper, que incluem a edição de cabeçalhos de arquivos RIFF e equalização.

Tanto soundBlade quanto Reaper dispõe aos usuários diversos equalizadores de áudio, desde os mais básicos até os equalizadores gráficos, cobrindo a equalização proposta neste trabalho, entretanto, estes programas

não são eficientes na leitura e escrita de cabeçalhos, especialmente do cabeçalho BWF. Esses softwares são programados para ignorar qualquer parte desconhecida de cabeçalhos, e, além disto, escrevem apenas a versão zero do BWF, muito mais simples que a versão dois apresentada no Capítulo 2, com a qual este trabalho lida.

As Figuras 17 e 18 mostram as telas principais desses softwares.

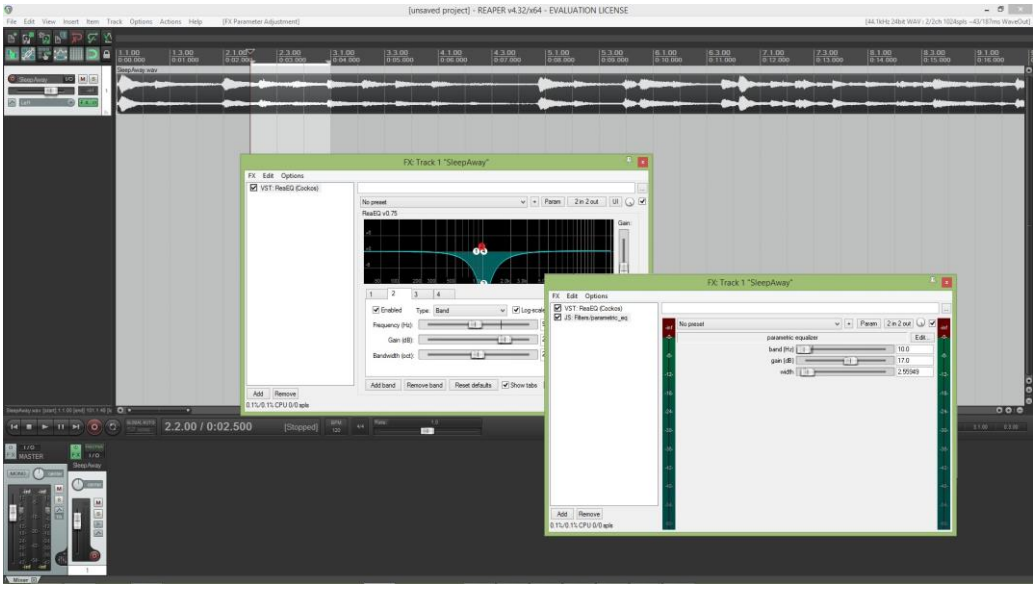

Figura 17: Janela principal do Reaper - Equalizador Gráfico e Peak abertos.

Figura 18: Janela principal do software soundBlade SE.

Estes softwares são gratuitos para testar, entretanto, tem custo em torno de 600 dólares, dependendo da versão desejada.

4. PROPOSTA DO TRABALHO

Este trabalho resume-se ao desenvolvimento de um programa em linguagem C equalizador e conversor de arquivos WAVE para arquivos BWF para a plataforma operacional Windows.

O ambiente de desenvolvimento utilizado foi o “Microsoft Visual Studio Express 2012”, que é um software gratuito para usuários cadastrados. O programa será executado no próprio computador, dessa forma, os requisitos de memória, velocidade e consumo de energia são muito flexíveis. É necessário que seja possível processar arquivos de tamanho da ordem de dezenas de mega bytes em segundos e que seja viável toda a alocação de memória que o programa utiliza.

4.1 Implementação do Programa

Neste capítulo serão apresentados os fluxogramas que descrevem o programa, que serão explicados e detalhados. O código desenvolvido não será apresentado devido a questões de segurança de informação.

A Figura 19 apresenta o fluxograma geral do programa, que será aprofundado etapa por etapa.

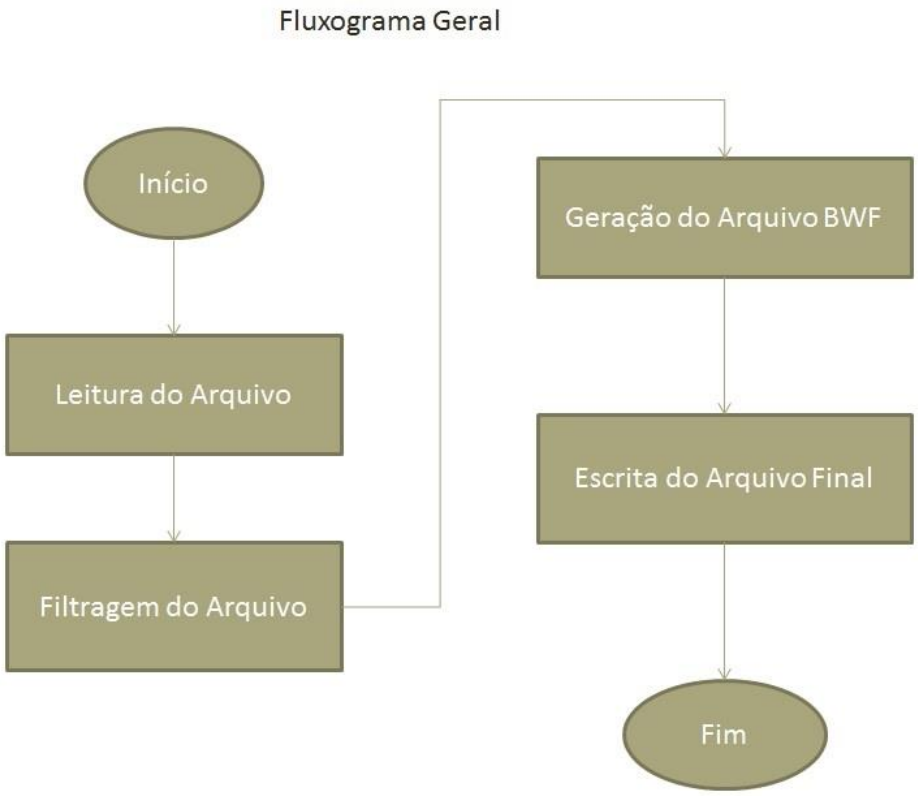

Figura 19: Fluxograma geral do programa desenvolvido.

O programa consiste de quatro principais etapas: a leitura do arquivo WAVE, a filtragem do áudio, a determinação de todas as informações necessárias para a geração do arquivo BWF e, finalmente, a escrita do arquivo final.

4.1.1 Leitura do Arquivo

A leitura do arquivo inicia-se por ler oito bytes em dois grupos de quatro, que devem representar a identificação e o tamanho do arquivo. Então, a identificação do arquivo é verificada, esta deve conter as letras “RIFF”. Se o arquivo for do tipo esperado, serão lidos mais quatro bytes, que devem conter as letras “WAVE”, que representam o tipo de arquivo aqui tratado. Sendo o arquivo do tipo desejado, inicia-se a leitura conforme o fluxograma da Figura 20.

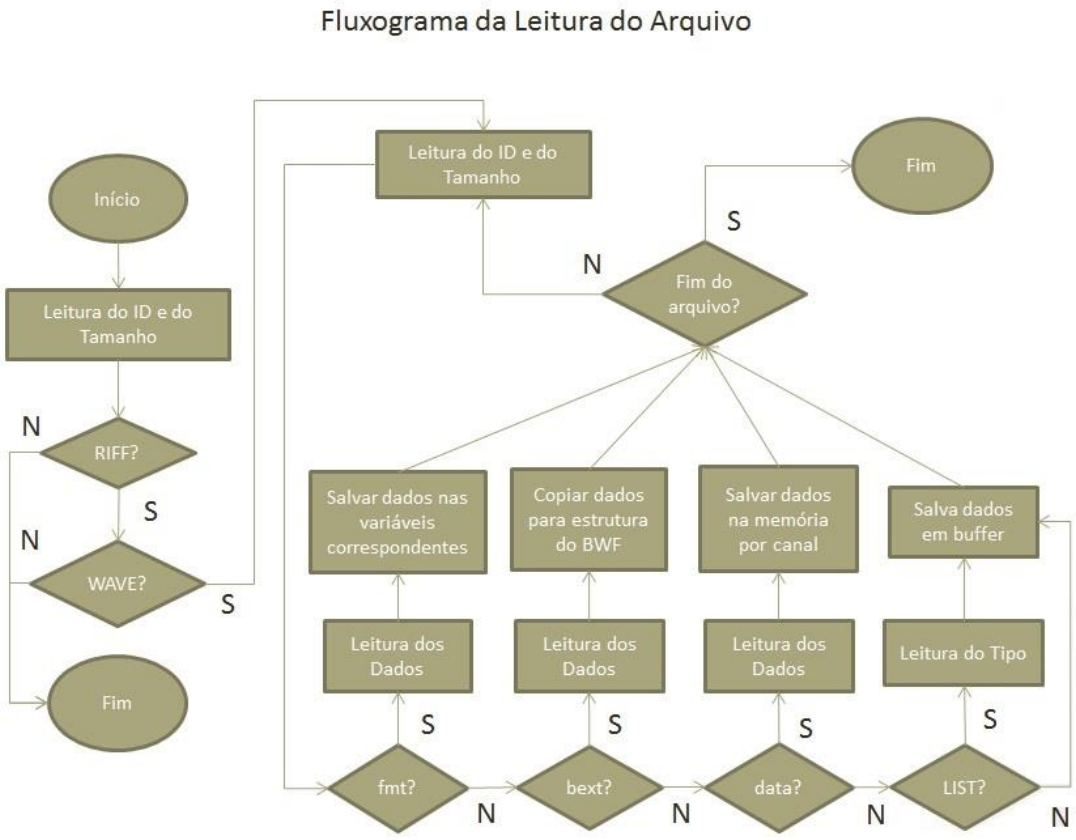

Figura 20: Fluxograma de leitura do programa desenvolvido.

A leitura do resto dos dados é prosseguida por ler mais oito bytes, correspondentes à identificação e tamanho do primeiro cabeçalho. A identificação do cabeçalho é comparada aos cabeçalhos de interesse, formatação, extensão BWF, dados e listas (estas devem ser comparadas por ter um formato diferente). Caso o cabeçalho não seja nenhum dos mencionados, os dados do mesmo são salvos em um buffer para a escrita dos mesmos no arquivo final.

Cada cabeçalho de interesse tem um tratamento específico, em geral, os dados são salvos para o tratamento ao que devem servir. Quando um cabeçalho é lido por completo, são lidos os oito bytes de identificação e tamanho do próximo cabeçalho. Este processo se repete até a leitura final do arquivo.

Os dados do sinal de áudio são salvos em memória para serem filtrados. As informações necessárias para o tratamento desses dados são salvas na estrutura da formatação, obtidas do cabeçalho “fmt”.

4.1.2 Filtragem do Áudio

A Figura 21 apresenta a próxima etapa do programa, a filtragem. Nesta etapa o filtro desejado é selecionado e aplicado aos dados de áudio. A escolha do filtro é definida por comando de linha ao chamar o programa, o que não será detalhado neste trabalho, pois esta interface ainda sofrerá muitas mudanças.

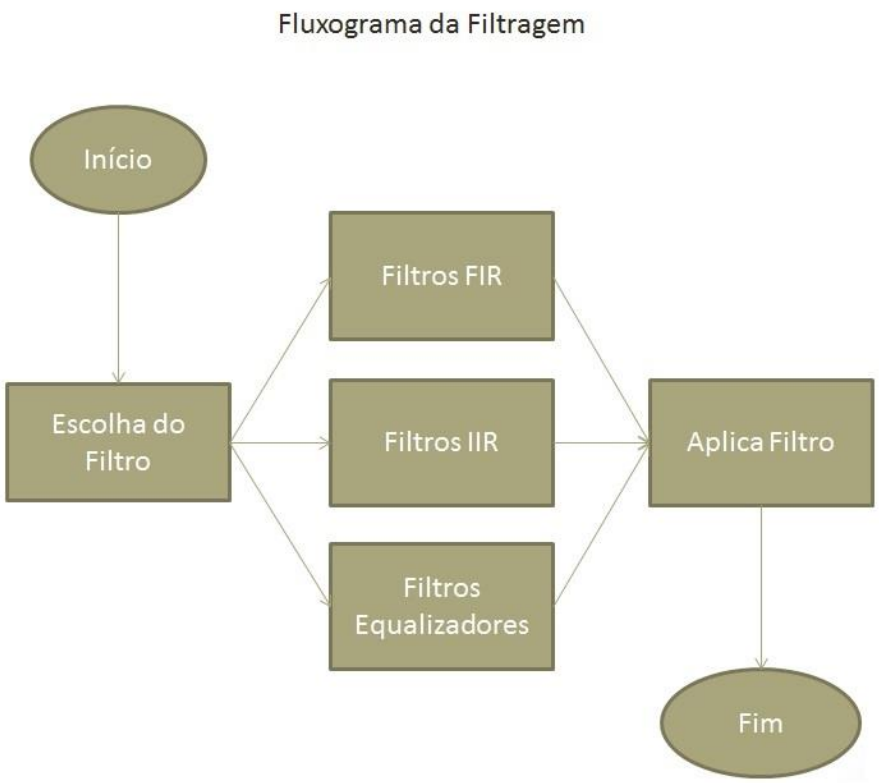

Figura 21: Fluxograma geral da filtragem do arquivo.

Os filtros implementados no programa são de três classes: FIR, IIR e equalizadores. Segue a lista completa dos filtros programados:

- FIR
    - Passa-baixa

    - Passa-alta
    - Passa-faixa
    - Rejeita-faixa
- IIR
    - Primeira Ordem
        - Passa-tudo
        - Passa-baixa
        - Passa-alta
    - Segunda Ordem
        - Passa-tudo
        - Passa-baixa
        - Passa-alta
        - Passa-faixa
        - Rejeita-faixa
- Equalizadores
    - Shelving
        - Primeira Ordem
            - Passa-baixa
            - Passa-alta
        - Segunda Ordem
            - Passa-baixa
            - Passa-alta
    - Peak
        - Segunda Ordem
            - Passa-faixa
            - Rejeita-faixa

Estes filtros englobam todos os filtros de primeira e segunda ordem digitais com ganho aplicável. A programação de cada um deles tem seus respectivos detalhes, entretanto, para tornar este trabalho mais objetivo, a implementação dos filtros será explicada considerando apenas os grupos.

Os filtros FIR, que apresentam resposta finita ao impulso, são definidos matematicamente no domínio da frequência, então, é feita a transformada de

Fourier para obter a resposta do filtro no domínio do tempo. A função do filtro no domínio da frequência é de primeira ordem, entretanto, esta não representa a ordem do filtro, que é dada pelo número de amostras aplicadas ao filtro, o que torna a resposta do filtro finita. A resposta do filtro é não causal, então, é necessário torná-la finita e causal.

A aplicação de janelas é utilizada para suavizar o filtro, obtida pela multiplicação da janela com a resposta do filtro. A causalidade é bastante objetiva, trata-se de um deslocamento do sinal resultante para que a este inicie no ponto zero.

A convolução é, então, feita com o sinal de áudio salvo na memória. Aqui é vital o cuidado dos limites do somatório da convolução e dos elementos dos sinais a serem somados. Como a convolução baseia-se na soma de sinais passados com a resposta do filtro, é necessário a verificação de que estes sinais atrasados são existentes e o programador deve estar atento a este aspecto. A Figura 22 mostra o fluxograma descrito.

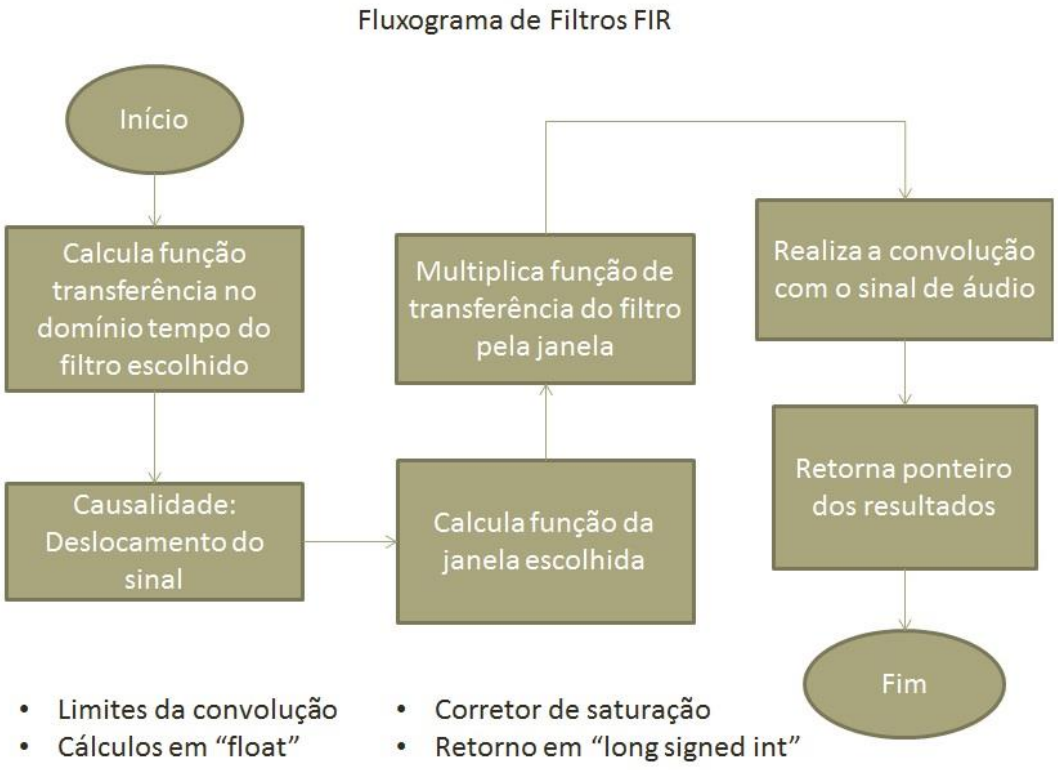

Figura 22: Fluxograma da filtragem por filtros FIR.

Os cálculos foram feitos utilizando dados do tipo ponto flutuante para garantir um resultado representativo. Os dados de áudio são inteiros de um a quatro bytes de tamanho, dependendo do arquivo. O sinal convoluído, que representa o sinal de áudio filtrado, é, então, convertido ao inteiro respectivo e retornado à função principal.

A programação dos filtros IIR é similar aos filtros FIR, pois, também, inicia-se definindo a resposta em frequência do filtro desejada. Entretanto, ao invés de transformar essa resposta para o domínio do tempo, aplica-se a transformada Z, que resultará em uma equação de diferenças como resposta do filtro.

Os parâmetros são calculados e, então, a resposta final do filtro. A equação utilizada é da forma da Equação 25. Veja que sinais passados fazem parte do cálculo, assim, nesta etapa, também, deve-se ter o cuidado de limitar os elementos tratados.

Como o caso dos filtros FIR, os cálculos são realizados utilizando dados do tipo ponto flutuante, que são convertidos a inteiros e retornados à função principal. A Figura 23 apresenta o fluxograma da implementação descrita.

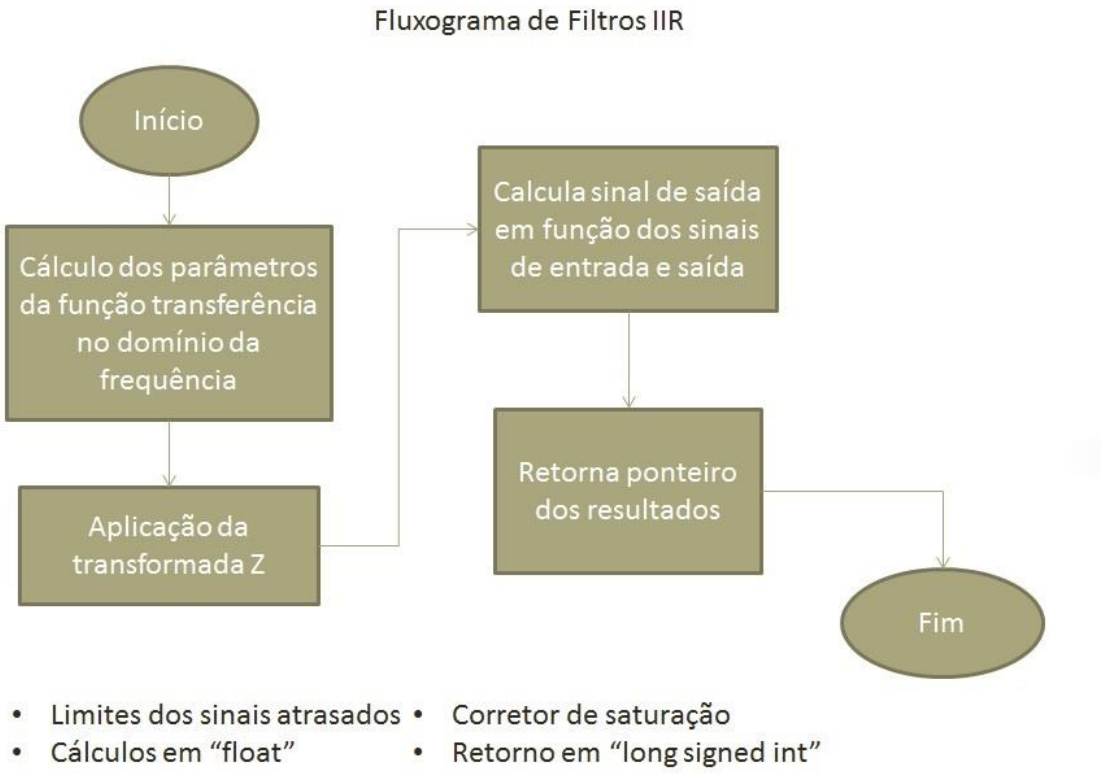

Figura 23: Fluxograma da filtragem por filtros IIR.

Os filtros equalizadores Shelving e Peak são baseados nos filtros IIR. Dessa forma, a implementação destes é quase idêntica a dos filtros IIR, a única diferença está nos parâmetros "a" e "b" da equação final, nos quais foi adicionada uma variável de ganho.

Os filtros Shelving passa-alta e passa-baixa são derivados dos filtros IIR passa-alta e passa-baixa, enquanto os filtros Peak são derivados dos filtros IIR passa-faixa e rejeita-faixa. A Figura 24 apresenta o fluxograma da implementação destes filtros.

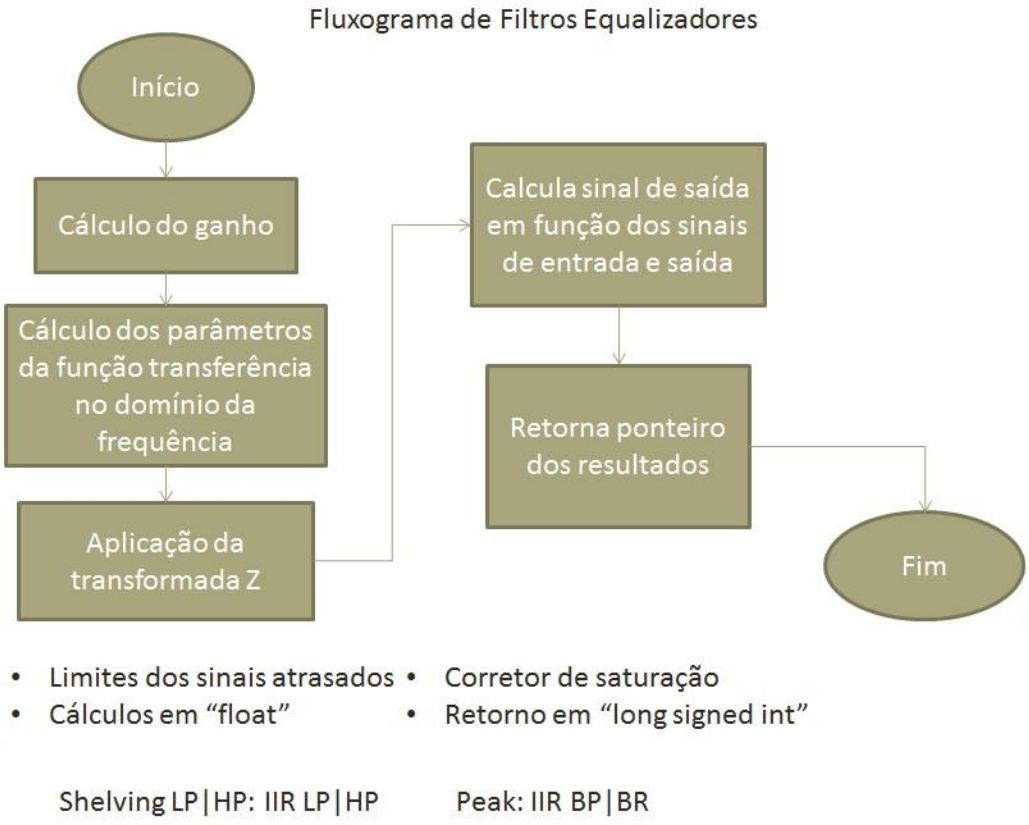

Figura 24: Fluxograma da filtragem por filtros equalizadores.

4.1.3 Geração do Arquivo BWF

A próxima etapa do programa consiste em gerar o arquivo BWF, sendo necessário determinar os parâmetros apresentados na Tabela 1. Muitos dos parâmetros já são definidos ou devem ser criados pelo programador, como o nome do criador do arquivo, sua referencia e local, a data e hora da criação do arquivo, a descrição do arquivo, registros, a versão da extensão e o histórico de código.

Os parâmetros mais importantes são pertencentes à versão dois do padrão BWF, relativos à potência sonora e picos do sinal de áudio. Estes parâmetros e o tempo de referência serão descritos com maior detalhamento. A Figura 25 apresenta o fluxograma para a geração do arquivo BWF levando em conta apenas estes parâmetros mais importantes.

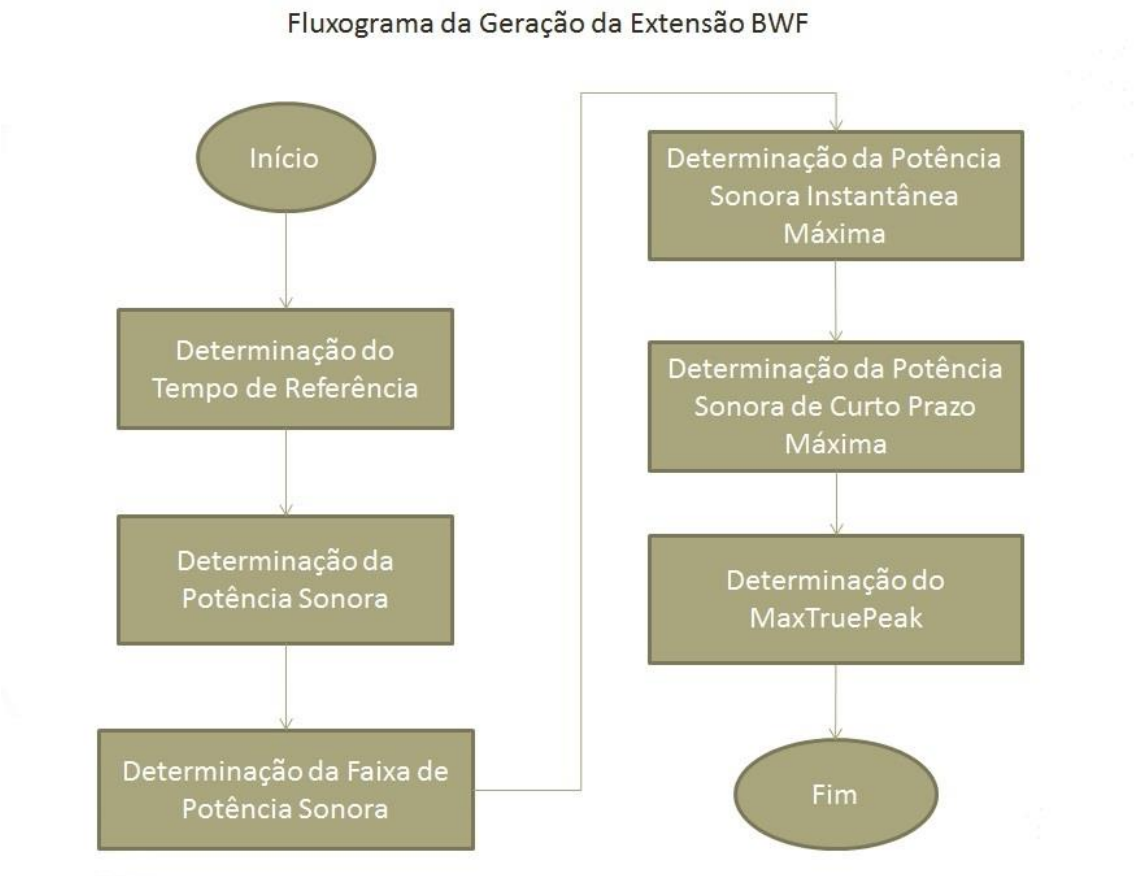

Figura 25: Fluxograma para a geração do arquivo BWF – Versão 2.

O tempo de referência é utilizado para a sincronização de arquivos de vídeo e/ou áudio. Este parâmetro é representado por oito bytes, sendo decomposto em dois grupos de quatro bytes denominados tempo de referência “low” e “high”, representando os bytes menos e mais significativos respectivamente.

Existe mais um detalhe intrigante a respeito desse parâmetro, pois o seu valor não representa diretamente o tempo, mas sim número de amostras desde a meia noite. Então, é necessário saber a frequência de amostragem para poder calcular o número de amostras correspondentes ao horário desejado.

A Figura 26 apresenta o fluxograma do cálculo do tempo de referência, onde *s*, *m* e *h* representam segundos, minutos e horas respectivamente e *T* e *Fs* o total de amostras e a frequência de amostragem.

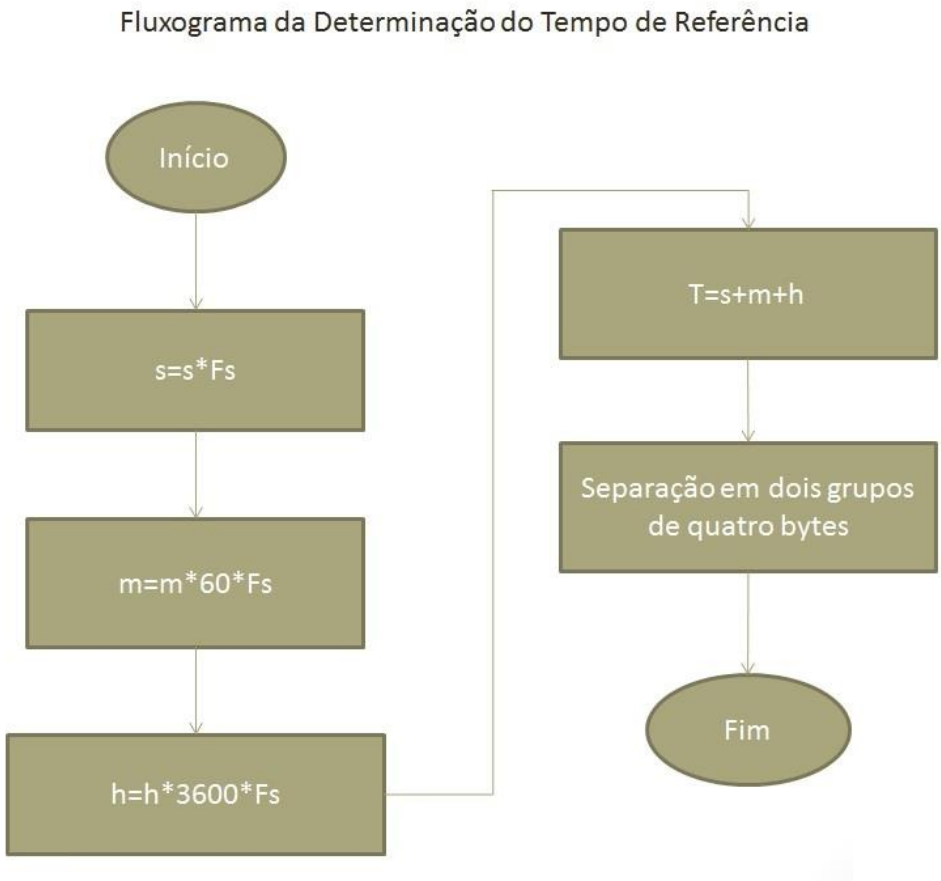

Figura 26: Fluxograma do cálculo do tempo de referência.

A determinação da potência sonora requer um processamento avançado. A ITU (International Telecomunication Union – União Internacional de Telecomunicação) determinou um algoritmo que define a forma de calcular a potência instantânea, de curto prazo e do programa [10].

Primeiramente, com o objetivo de simular a resposta em frequência do ouvido humano, o sinal de áudio, cuja potência será calculada, é passado por um filtro chamado “Filtro-K”. Este filtro, na verdade, é composto por dois filtros, um filtro IIR de segunda ordem passa alta com frequência de corte em cerca de 100Hz e um filtro Shelving passa alta com ganho de 4dB e frequência de corte próxima a 1000Hz. Estes filtros estão apresentados na Figura 27 [10].

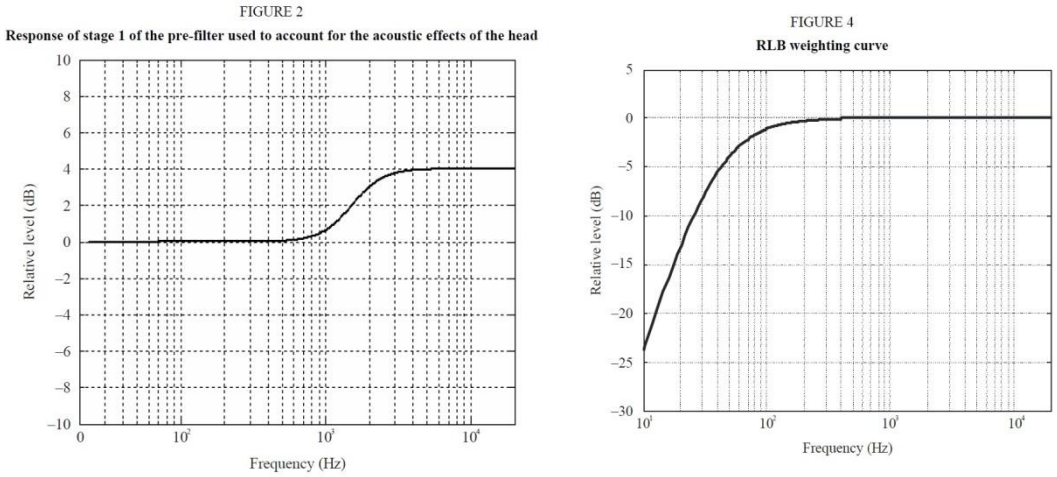

Figura 27: Formação do filtro tipo K.

A potência será calculada com base no sinal de áudio filtrado. O algoritmo definido pela ITU consiste em calcular potências janeladas no tempo. Em outras palavras, calcular a potência dos primeiros 400ms de áudio corresponde à potência da primeira janela de 400ms. Essas janelas são sobrepostas, 75% para ser exato, e a potência é calculada para cada uma das janelas até um tempo determinado, "*T*".

Os valores de potência janelados são comparados a dois limiares, um absoluto e outro relativo. O limite absoluto corresponde a -70dB. Todas as janelas cuja potência é menor que esse limiar são descartadas do cálculo da potência final. Após as janelas terem sido comparadas ao primeiro limite, se calcula um novo limite, relativo ao absoluto.

Este novo limiar é relativo à potência ao longo do arquivo e pode variar bastante seu valor. Ele é determinado pelo somatório das potências não descartadas pelo primeiro limite e por uma adição de 10dB do limiar absoluto. Os valores de potência janelados, menores que este novo limiar relativo, também, são descartados do cálculo da potência final. A Figura 28 apresenta o janelamento do sistema, onde "*Tg"* representa a duração das janelas e "*Overlap"* o quanto das janelas é sobreposto.

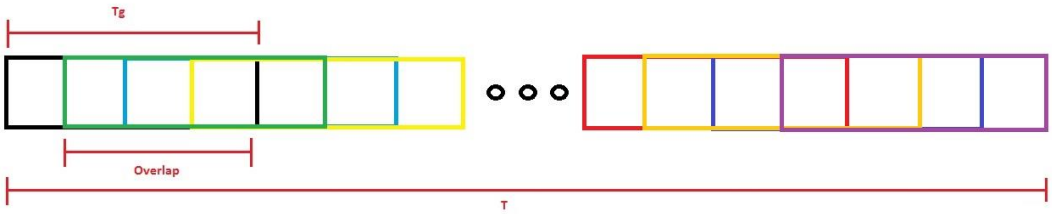

Figura 28: Janelamento para o cálculo de potência.

A sobreposição das janelas é feita para correlação das potências calculadas, assim garante o cálculo da maioria dos valores possíveis. O tempo de duração das janelas "*Tg*" é fixado pela IBU em 400ms, como o "Overlap" de 75%. Existe um segundo valor de tempo "T", já mencionado, que representa a faixa de tempo varrida pelo algoritmo. Este parâmetro determina o tipo de potência calculada. A potência instantânea consiste em calcular a potência durante o tempo total de 400ms (seria equivalente a calcular a potência de apenas uma janela). A potência de curto prazo corresponde a aplicação do algoritmo para três segundos e a potência sonora do programa, ou simplesmente potência sonora, aplica o algoritmo ao longo de todo o arquivo de áudio.

A Figura 29 apresenta o fluxograma do cálculo da potência sonora descrito. A EBU fornece arquivos padrão para teste e calibração destes cálculos de potência, o que será mais detalhado no Capítulo 5, que se trata dos resultados obtidos.

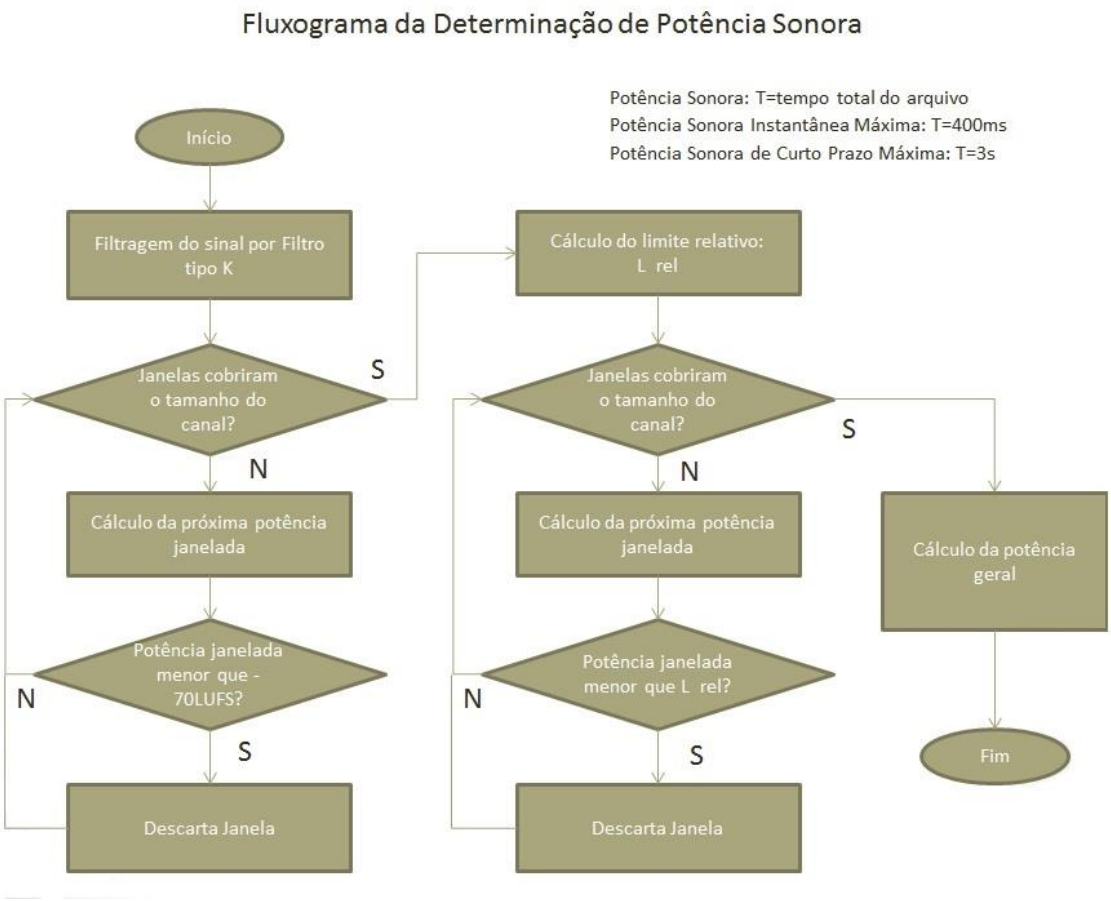

Figura 29: Fluxograma da determinação de potência sonora.

A faixa de potência sonora é calculada a partir dos valores obtidos de potência sonora. Para garantir que eventos curtos, mas de alto volume, não afetem a faixa de potência sonora calculada, se utiliza um método análogo ao “Interquartile Range”. Este método consiste em utilizar limites de distribuição estatística para se determinar a faixa de variação de potência. Ele considera que a potência do arquivo segue uma distribuição normal e define limites para a determinação da verdadeira população de potência. Em outras palavras, este método retira exceções.

A EBU define o nonagésimo quinto e o décimo percentual como os limites que caracterizam a faixa, assim retirando do cálculo os valores que representam silêncio ou eventos de volume extremo de curtíssimo prazo. Para se aplicar esse método, organiza-se os valores de potência calculados em ordem crescente, então, verifica-se o valor correspondente ao nonagésimo quinto e o décimo percentual. A diferença entre esses valores representará a

faixa de potência sonora desejada. A Figura 30 apresenta o fluxograma da determinação da faixa de potência sonora.

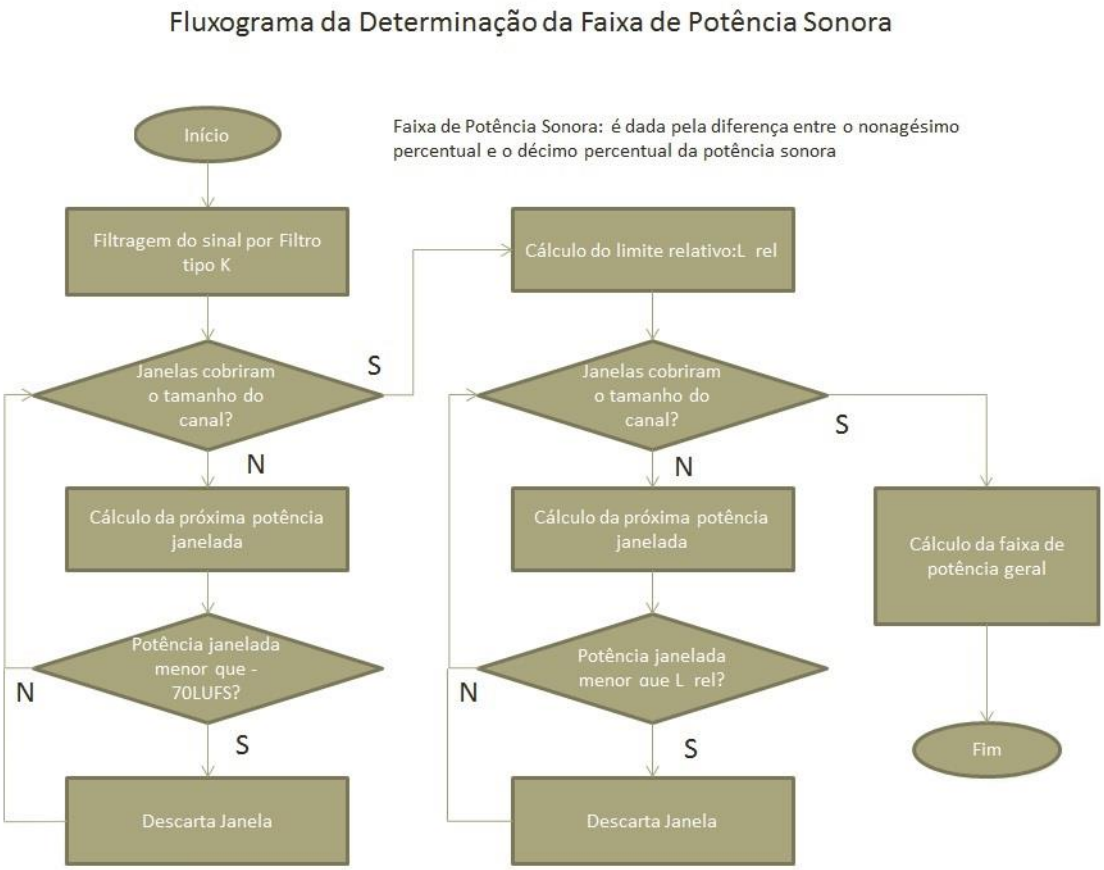

Figura 30: Fluxograma da determinação da faixa de potência sonora.

Existem algumas diferenças no cálculo da potência janelada na determinação da faixa de potência sonora e da potência sonora. A potência janelada na determinação da faixa de potência é calculada utilizando um "Overlap" mínimo de 66%. O limite relativo é determinado a +20dB do limite absoluto, após terem sido retiradas as janelas cuja potência fosse menor que o limite absoluto de -70dB. As janelas utilizadas têm duração de três segundos.

Estas mudanças resultam em grandes diferenças nos valores calculados e não podem ser ignoradas.

A determinação do pico máximo do sinal de áudio é uma questão que vem sendo discutida por muitos grupos a tempo. Existem diversos medidores de pico, como RMS, Peak, VU, entre outros. A EBU levanta o questionamento

da veracidade destes picos calculados por esses métodos. A principal questão se trata de que os verdadeiros picos podem se encontrar entre duas amostras, ao se considerar um comportamento analógico do sinal digital. Assim, a EBU em conjunto com a ITU definiu o valor de pico máximo verdadeiro do sinal ("Maximum True Peak"). O algoritmo para o cálculo desse valor é fornecido pela ITU [10].

Antes de prosseguir para a explicação do cálculo e deste parâmetro é importante mencionar que esse assunto ainda é muito recente e ainda não foi definido determinadamente. Mesmo dentro do algoritmo fornecido pela ITU existem muitas etapas opcionais, e nem a ITU, nem a EBU fornecem algum material padrão para a verificação e calibração do algoritmo. A Figura 31 apresenta o fluxograma da determinação do pico máximo verdadeiro do sinal.

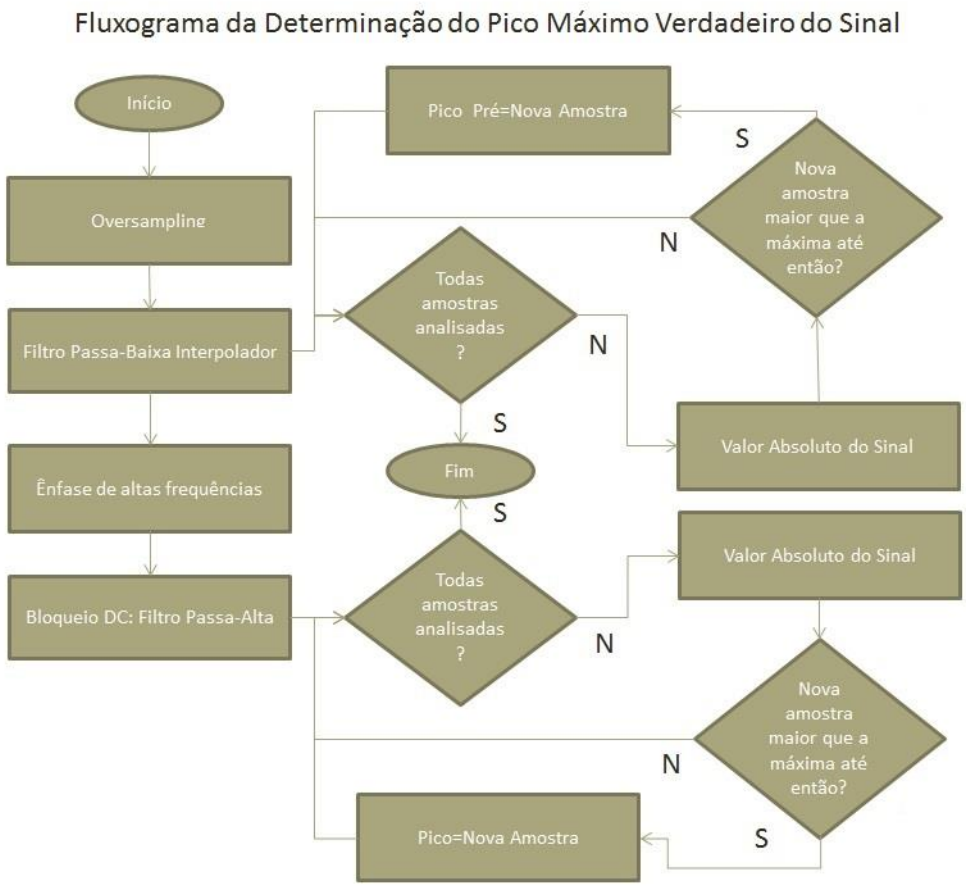

Figura 31: Fluxograma da determinação do pico máximo do sinal de áudio.

A ideia geral do algoritmo é de fazer uma sobreamostragem do sinal, de tal forma que frequências mais altas possam ser mais bem representadas, pois a ITU alega que o erro de pico de altas frequências chega facilmente a -3dB de leitura. Essa sobreamostragem é feita por inserir zeros entre as amostras do sinal. Então, é aplicado um filtro passa baixa com o objetivo de interpolar o sinal. A ITU não especifica exatamente este filtro, apenas ressalta que ele deve cortar frequências acima da máxima desejada. Assim, o filtro utilizado apresenta frequência de corte igual a metade da frequência de amostragem (a maior frequência que pode ser representada de acordo com o critério de Nyquist).

Após o filtro interpolador existem as etapas opcionais, que consistem de um filtro de ênfase de altas frequências e do bloqueio DC do sinal. O filtro de ênfase é implementado por um filtro Shelving com um zero em 14.1kHz e um polo em 20kHz (também opcional). O ganho deste filtro utilizado foi de *G=3dB*. O bloqueio DC não é especificado pela ITU, e foi feito pela aplicação de um filtro IIR passa-alta de segunda ordem com frequência de corte em 100Hz.

Após estas etapas o valor absoluto do sinal é calculado. Finalmente, varre-se este sinal de áudio tratado procurando-se a amostra de maior valor, que é convertida para dB. A ITU aconselha implementar o filtro com e sem as etapas opcionais, calculando ambos valores de pico e selecionando o maior valor como sendo o pico verdadeiro.

Como foi mencionado, muito neste algoritmo ainda precisa ser detalhado e definido, além de serem fornecidos arquivos padrões para a verificação e calibração da implementação do mesmo. Entretanto, este algoritmo foi desenvolvido de tal forma que o valor calculado como pico verdadeiro será provavelmente maior que o valor de pico real. Assim, tem-se a vantagem de ter valores representativos do verdadeiro limite de pico do sinal.

4.1.4 Escrita do Arquivo Final

A escrita do arquivo final resume-se a união de todas as etapas apresentadas até o momento. Os cabeçalhos lidos tinham sido salvos de

alguma forma (isso dependia do cabeçalho específico), que são então escritos em um novo arquivo. Os parâmetros calculados do arquivo BWF são escritos em seguida na estrutura de extensão “bext”. Caso já existisse uma extensão “bext” estas seriam fundidas em uma única extensão, que é então escrita no novo arquivo.

Os dados filtrados se encontram salvos na memória RAM do computador neste ponto, e são os últimos dados a serem escritos. A ordem geral dos dados é sempre mantida a mesma, a única possível mudança é a adição da extensão do arquivo BWF, caso essa já não existisse, que é adicionada após o cabeçalho de formatação.

5. RESULTADOS EXPERIMENTAIS

Neste capítulo serão apresentados e discutidos os resultados obtidos dos filtros implementados e dos parâmetros calculados do arquivo BWF. Os resultados são apresentados de tal forma a cobrir todos os assuntos tratados o mais objetivamente possível.

5.1 Resultados do Desempenho dos Filtros

Nesta seção serão apresentados, discutidos e analisados os resultados das respostas em frequência dos filtros implementados. Para obter os resultados foi gerado um arquivo WAVE de ruído branco pelo software "AudaCity". Esse ruído branco é apresentado na Figura 32 e foi o sinal aplicado aos filtros para se obter a respostas dos mesmos.

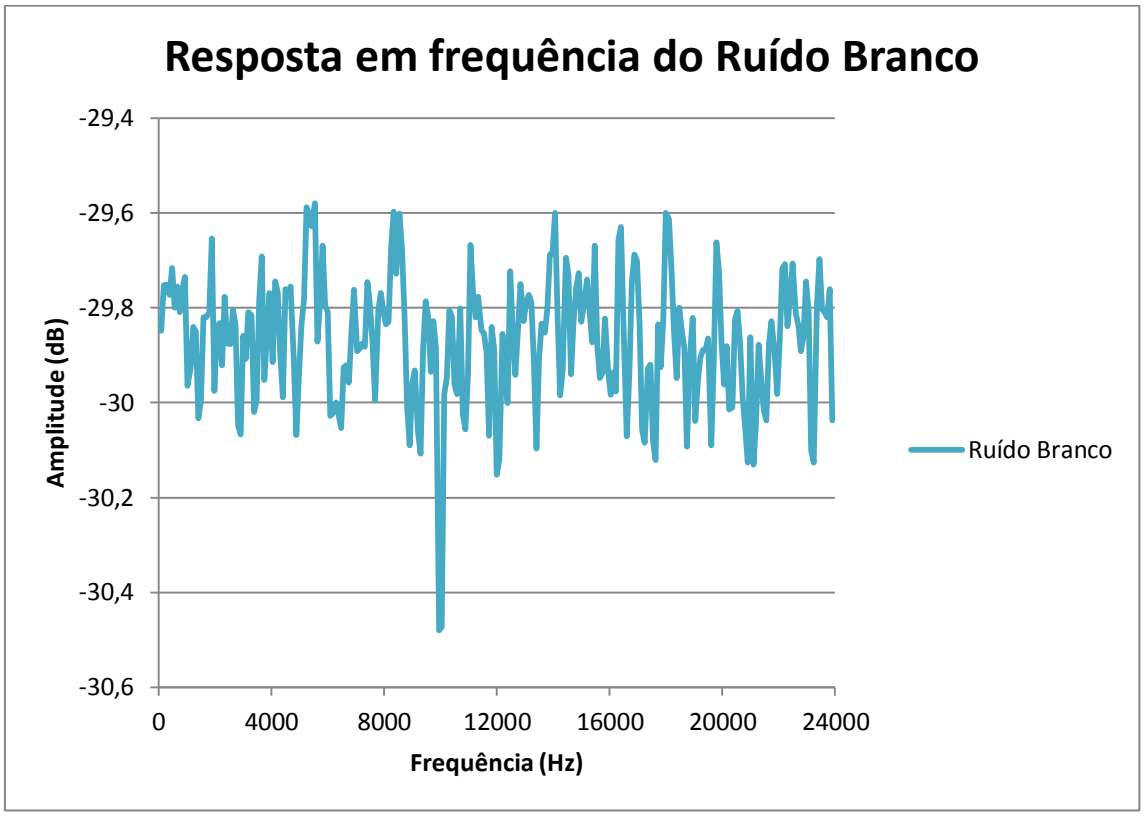

Figura 32: Resposta em frequência do Ruído Branco.

Note que o ruído gerado não é ideal, pois oscila em média cerca de 0,4dB. Na frequência próxima a 10kHz existe uma queda em amplitude de cerca de 0,7dB da média. Estas não idealidades podem distorcer consideravelmente as respostas em frequência dos filtros, especialmente na região próxima aos 10kHz.

As respostas em frequência que serão apresentadas neste capítulo também foram obtidas utilizando o software mencionado. Estas foram exportadas e então plotadas utilizando a ferramenta “Microsoft – Excel”.

Para tornar a apresentação dos resultados mais objetiva, foram selecionadas figuras cobrindo todos os parâmetros dos filtros disponíveis no programa, entretanto, não serão apresentados os gráficos de todos os parâmetros para cada filtro, mas, sim, um parâmetro por filtro, apenas o necessário para tratar de todos os assuntos. Isso ficará claro ao longo deste capítulo.

5.1.1 Resultados dos Filtros FIR

Nesta seção serão apresentados os resultados correspondentes aos filtros FIR. A Figura 33 mostra a resposta em frequência do filtro FIR passa-alta, onde foi variada a frequência de corte do filtro. A janela utilizada foi a retangular, o ganho nulo e a ordem do filtro (amostras filtradas por amostra de saída) de 100 unidades.

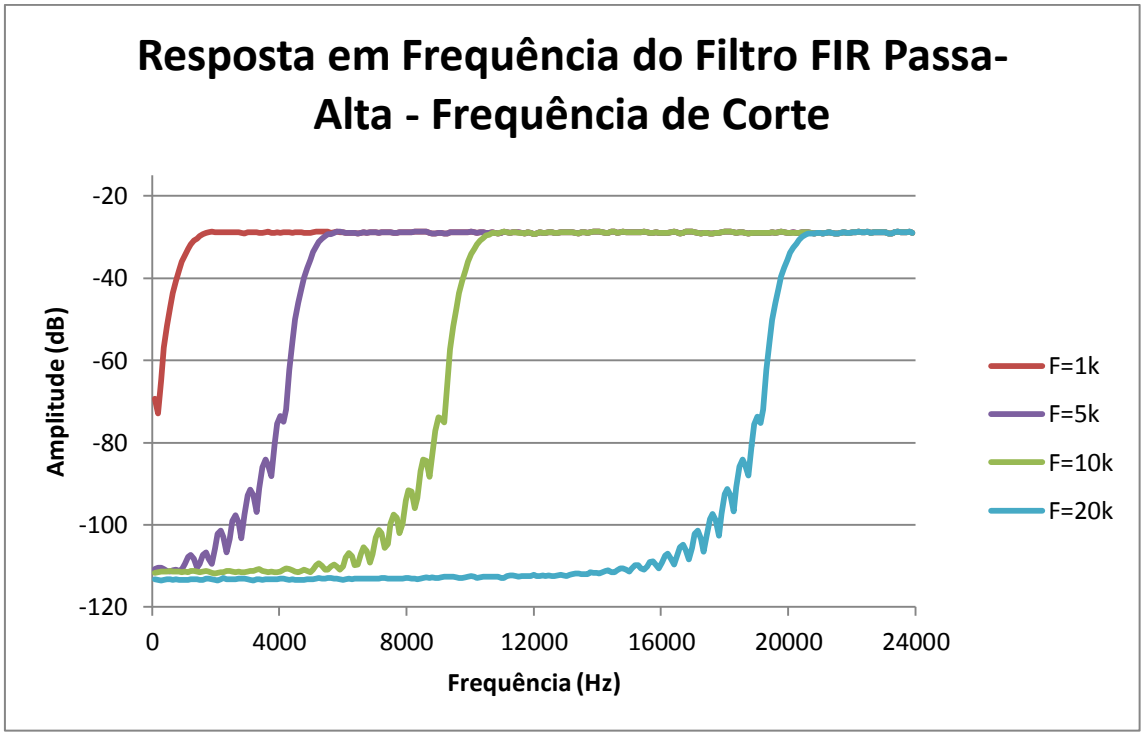

Figura 33: Resposta em frequência do filtro FIR passa-alta – Frequência de Corte.

Filtros FIR janelados são projetados para que a frequência de corte corresponda a uma atenuação de -6dB, diferentemente da maioria dos filtros. Foi verificado que em todas as frequências de corte da Figura 33 a atenuação correspondia a -6dB, como esperado.

A Figura 34 apresenta a resposta em frequência de um filtro FIR Passa-Baixa, onde foram mantidos constantes a frequência de corte de 12kHz, o ganho nulo e a janela Blackman. Note que na frequência de corte todas as curvas se cruzam em -6dB, relativo à magnitude base do sinal.

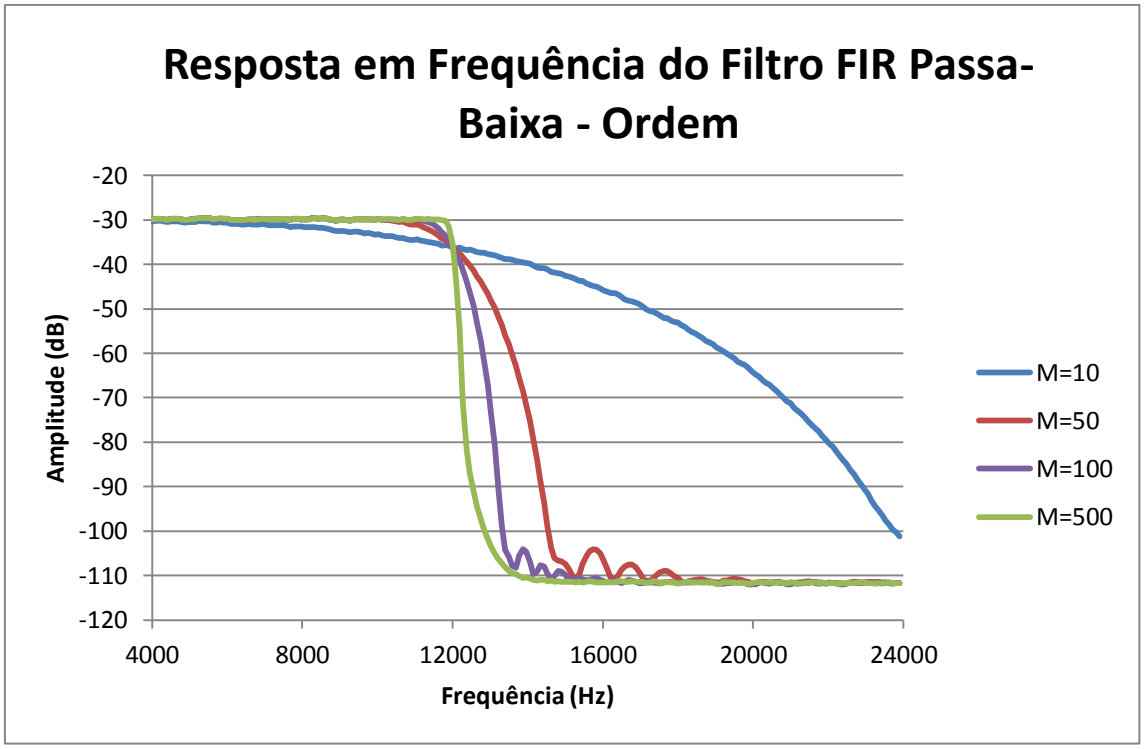

Figura 34: Resposta em Frequência do Filtro FIR Passa-Baixa – Ordem.

A queda em magnitude por década ou oitava é dependente da ordem do filtro FIR, conforme mostra a Figura 34. Para ordens baixas, próximas a dezenas de unidades, a resposta do filtro é ineficiente, não atenuando consideravelmente as frequências superiores a de corte. Já para ordens maiores, de centenas de unidades, essa resposta é excelente, atenuando rapidamente o sinal a ser filtrado.

A ordem dos filtros FIR também tem influência sobre o “ripple” da janela utilizada. Quanto maior a ordem (quantia de amostras processadas por amostra filtrada) menor o “ripple” da janela. Filtros FIR de ordens altas exigem alto processamento computacional, entretanto, é possível obter respostas em frequência quase ideais, sem “ripple”, com altas atenuações e com distorção de fase linear.

A Figura 35 apresenta as diferentes janelas implementadas no programa para os filtros FIR, no caso, um filtro FIR passa-faixa com frequência de corte inferior de 7kHz e superior de 12kHz, com ganho nulo e ordem de 100 unidades.

A janela com a maior oscilação em magnitude é a retangular, que também tem a menor atenuação. A janela de Hamming tem uma resposta muito similar à retangular, mas com uma atenuação maior. As janelas de Blackman e Hanning são muito similares, entretanto, a de Blackman tem o melhor desempenho, demonstrando uma atenuação maior e a oscilação um pouco menor.

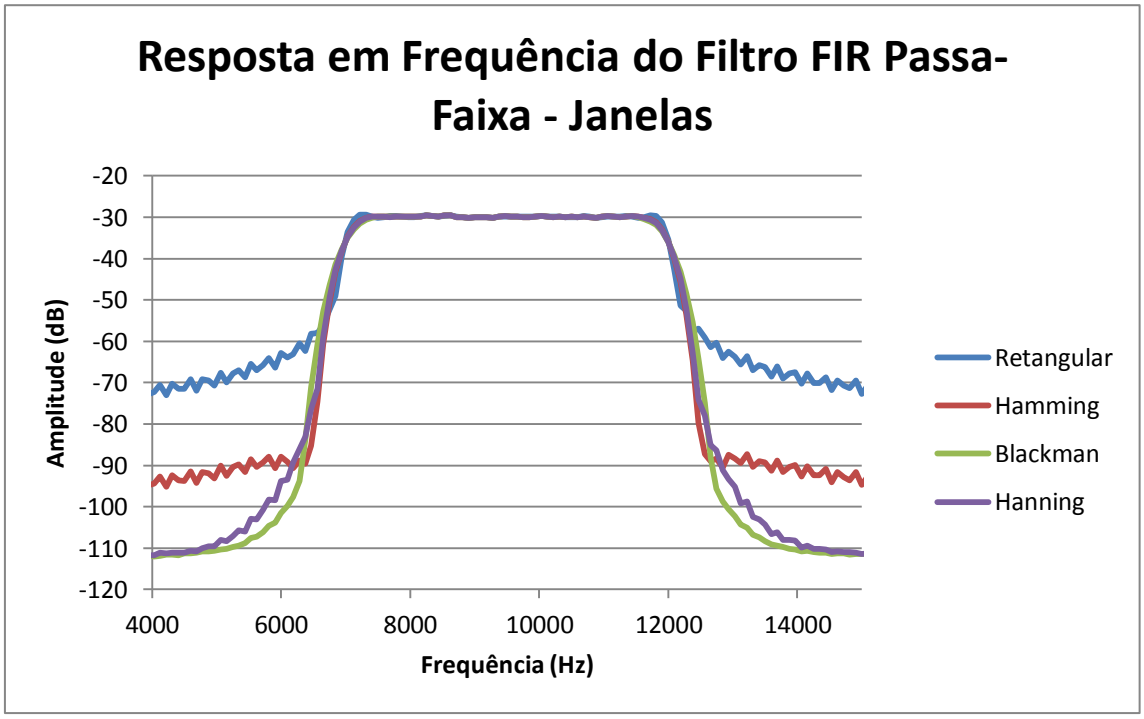

Figura 35: Resposta em Frequência do Filtro FIR Passa-Faixa – Janelas.

Antes de prosseguir é importante mencionar que inicialmente a atenuação mais brusca é da janela retangular, seguida pela janela Hamming, Hanning e Blackman respectivamente. Ou seja, na frequência de corte essas janelas podem ser consideradas mais efetivas, entretanto, suas atenuações finais são mais fracas como já mencionado.

A Figura 36 apresenta o último parâmetro dos filtros FIR sendo variado, o ganho. A figura mostra a resposta em frequência do filtro FIR rejeita-faixa, com frequências de corte inferior de 7kHz e superior de 12kHz, de ordem de 100 unidades e janela Hanning aplicada.

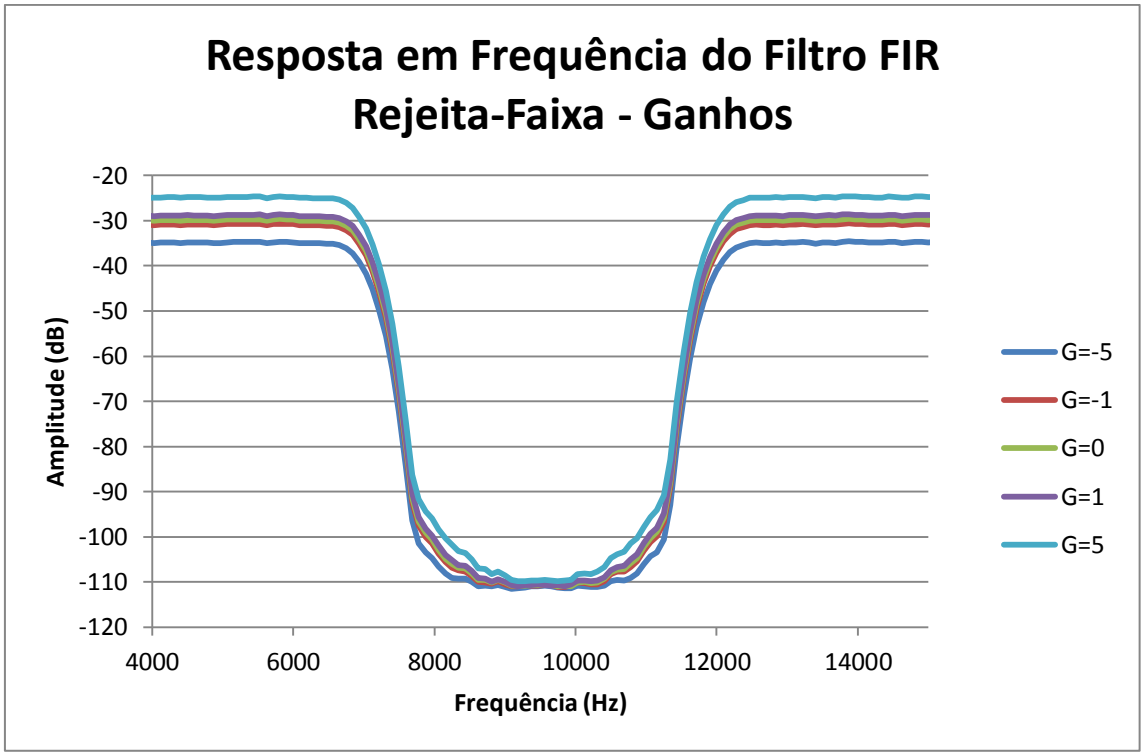

Figura 36: Resposta em Frequência do Filtro FIR Rejeita-Faixa – Ganhos.

O efeito do ganho foi programado de tal forma que todo o sinal fosse amplificado ou atenuado. Em outras palavras, todas as frequências tem o mesmo ganho aplicado em dB. Quando o ganho é nulo, os filtros FIR programados correspondem aos filtros FIR comuns (onde não há variável de ganho implementada).

Note que o conjunto de gráficos apresentados nesta seção engloba todos os filtros FIR implementados (passa-baixa, passa-alta, passa-faixa e rejeita-faixa) e todos os parâmetros variáveis dos mesmos (frequência de corte, ganho, janela aplicada e ordem do filtro), cobrindo todas as possíveis combinações de parâmetros e filtros. Na próxima seção serão apresentados os resultados dos filtros IIR.

5.1.2 Resultados dos Filtros IIR

Nesta seção serão apresentados os gráficos de alguns filtros IIR e diferentes parâmetros, visando cobrir todas questões tratadas neste trabalho de modo mais objetivo possível.

A Figura 37 apresenta o gráfico da resposta em frequência de um filtro IIR passa-alta de primeira ordem, onde as frequências de corte foram variadas. Foi verificado que na frequência de corte todas as curvas se encontravam a uma atenuação de -3dB, como esperado de um filtro IIR.

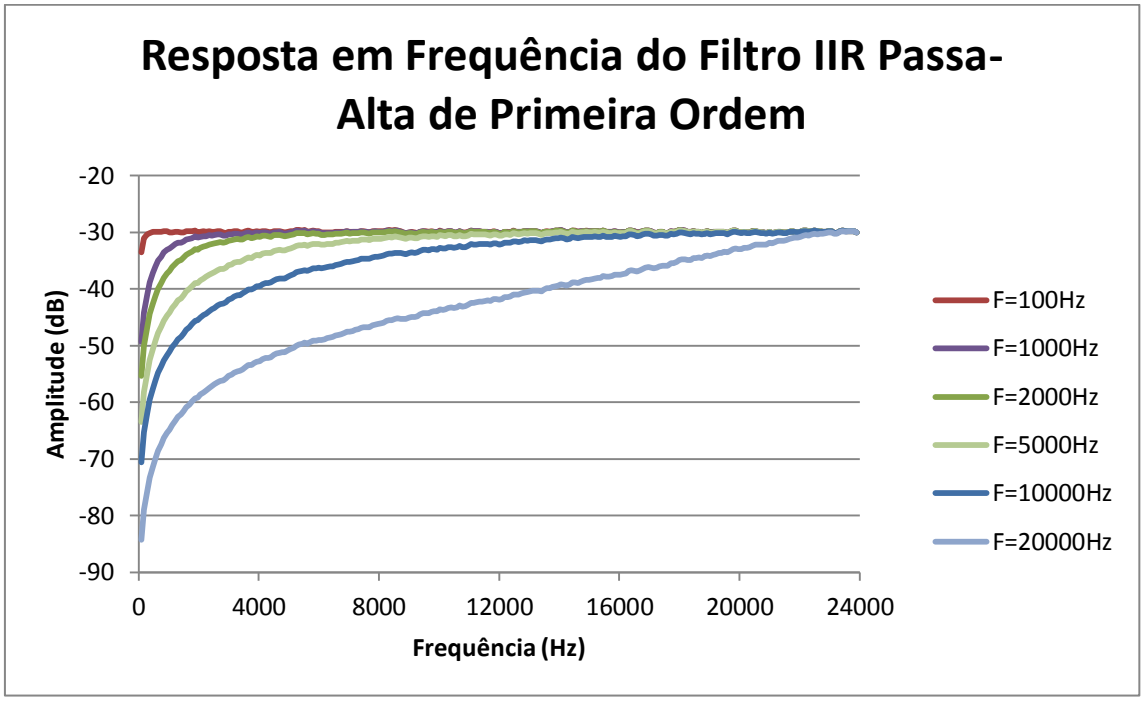

Figura 37: Resposta em Frequência do Filtro IIR Passa-Alta de Primeira Ordem.

Utilizando uma escala logarítmica foi possível confirmar que a atenuação do filtro realmente corresponde a -20dB/década, garantindo o comportamento esperado de um filtro IRR de primeira ordem.

A Figura 38 apresenta a resposta em frequência de um filtro IIR passa-baixa de segunda ordem, onde foi variada a frequência de corte do mesmo. Os mesmos procedimentos foram adotados, garantindo que a atenuação nas variadas frequências de corte é de -3dB e a atenuação analisada no gráfico logarítmico demonstrou ser -40dB/década, o esperado de um filtro IIR de segunda ordem.

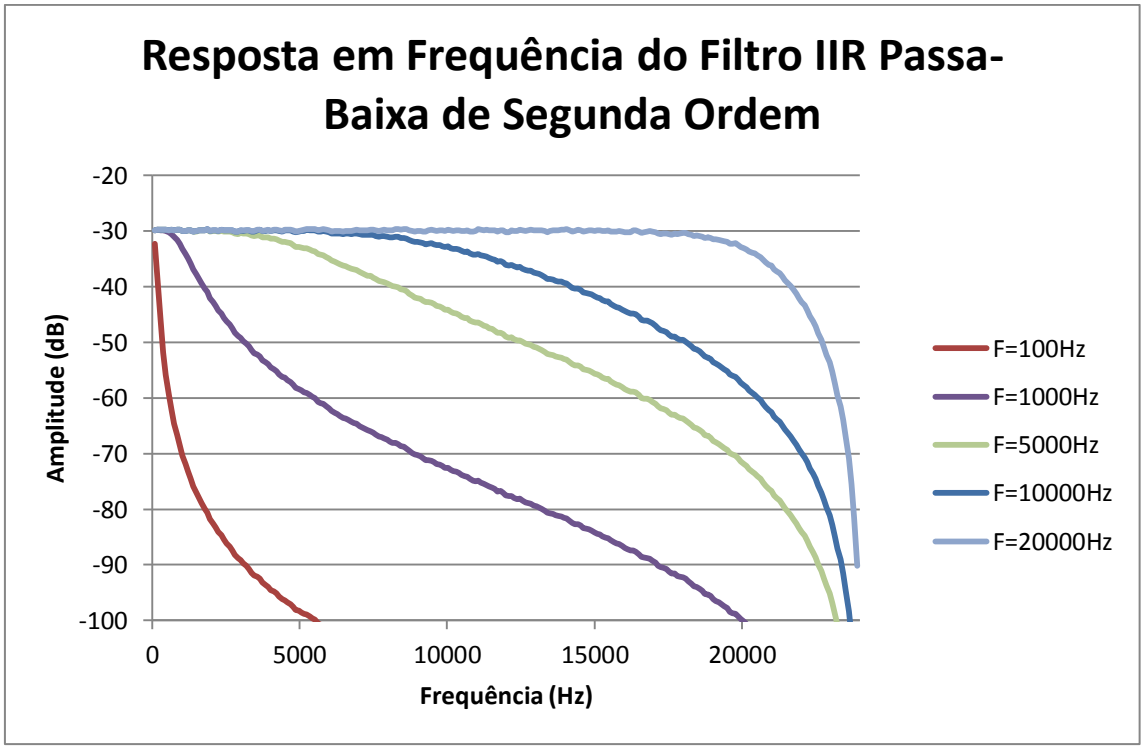

Figura 38: Resposta em Frequência do Filtro IIR Passa-Baixa de Segunda Ordem.

Existe uma peculiaridade em filtros digitais. A formulação matemática desses filtros é feita utilizando frequência normalizada em relação à frequência de amostragem. Isso tem alguns efeitos visíveis nesses filtros.

O primeiro é que próximo aos extremos das frequências, o comportamento linear não é mais garantido (a atenuação nessa região não será -20dB/década para um filtro de primeira ordem por exemplo). Isso ocorre, pois os pontos extremos são utilizados como valores definidos, são pontos fixos dependentes do filtro aplicado e seus parâmetros utilizados.

Outro efeito dessa formulação é que os filtros têm certo comportamento de simetria em relação à frequência normalizada, isso é visível nas Figuras 39, 43, 44 e 45, discutidas adiante.

A Figura 39 apresenta a resposta em frequência de um filtro IIR passa-banda com frequência de corte em 10kHz, onde a banda de frequência passante é variada.

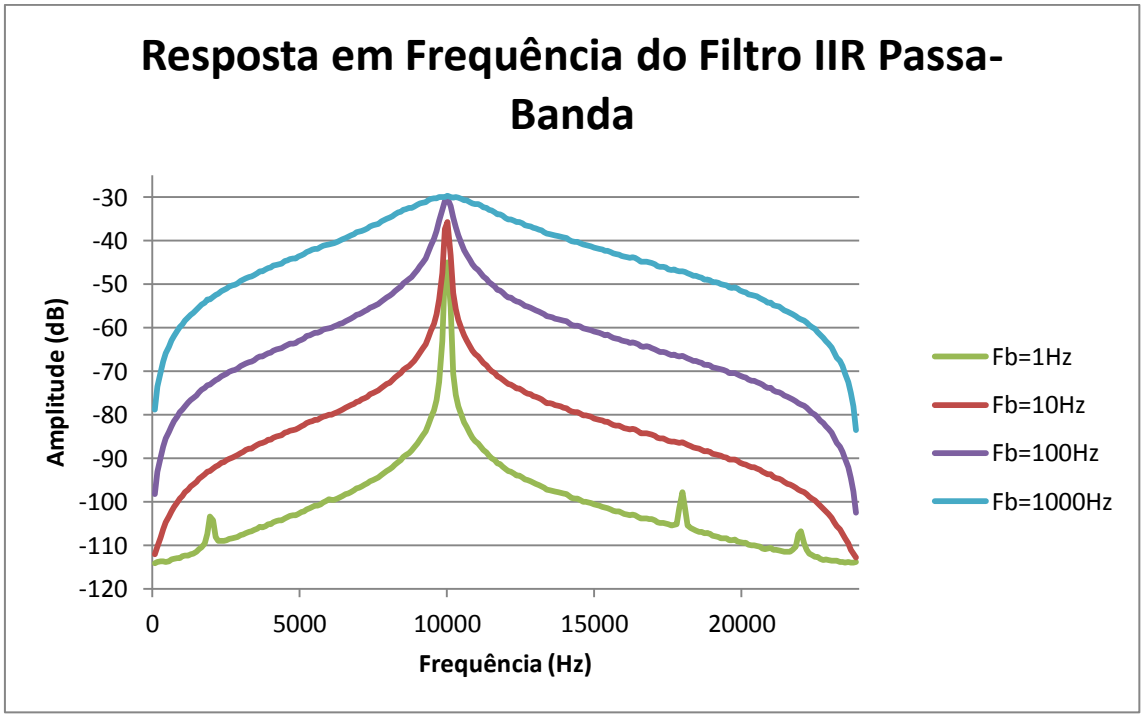

Figura 39: Resposta em Frequência do Filtro IIR Passa-Banda.

Note que a simetria é em relação à frequência normalizada e não a frequência central como deveria ser para um filtro passa-banda. Isso fica claro ao analisar a linha de -60dB para a banda passante de 1000Hz. Veja que os pontos que cruzam esta linha são em cerca de 1kHz e 22kHz. Se a simetria fosse em relação à frequência de central, esses pontos deveriam se encontrar à uma mesma distancia de 10kHz.

Outro fato importante é de que para bandas passantes muito curtas, o filtro não demonstra o comportamento ideal, atenuando também as frequências que deveriam ser passantes e induzindo picos em casos extremos como no de banda passante de 1Hz.

A Figura 40 apresenta a resposta em frequência do filtro IIR rejeita-faixa com banda de rejeição constante de 100Hz e frequências de corte variadas.

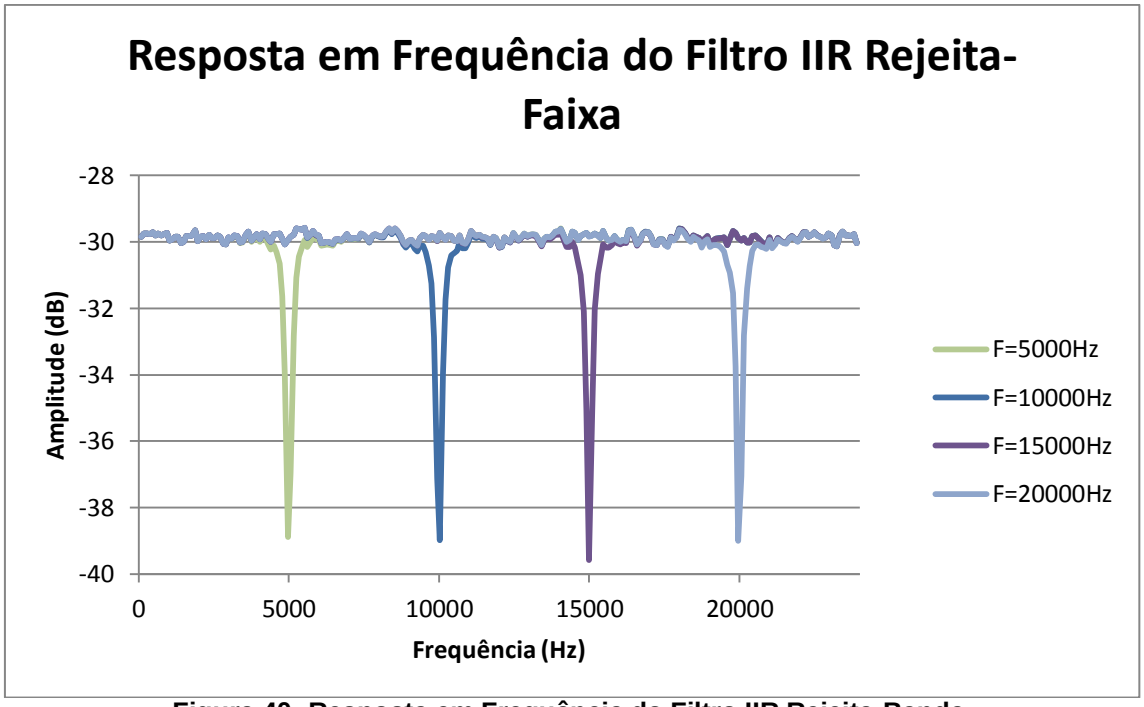

Figura 40: Resposta em Frequência do Filtro IIR Rejeita-Banda.

É importante ressaltar que há uma pequena variação entre as amplitudes de pico dos sinais, chegando a uma variação máxima de aproximadamente 0,69dB. Uma possível explicação para isso seria as não idealidades do ruído branco utilizado para testar os filtros, entretanto, nenhum experimento foi feito para confirmar essa hipótese.

Conforme proposto, os gráficos apresentados nesta seção cobrem todos os filtros IIR implementados no programa ao longo com os parâmetros variáveis dos mesmos, com exceção dos filtros IIR passa-tudo, que não foram apresentados, pois sem um estudo de distorção de fase não faz sentido apresentar os resultados apenas de magnitude dos mesmos. Na próxima seção são analisados os resultados dos filtros equalizadores.

5.1.3 Resultados dos Filtros Equalizadores

Nesta seção serão apresentados os resultados dos filtros Shelving e Peak disponíveis no programa deste trabalho.

As Figuras 41 e 42 apresentam filtros Shelving passa-alta de primeira ordem e passa-baixa de segunda ordem respectivamente. Nestas figuras as

frequências de corte e os ganhos são variados de forma a cobrir todo o espectro de frequência.

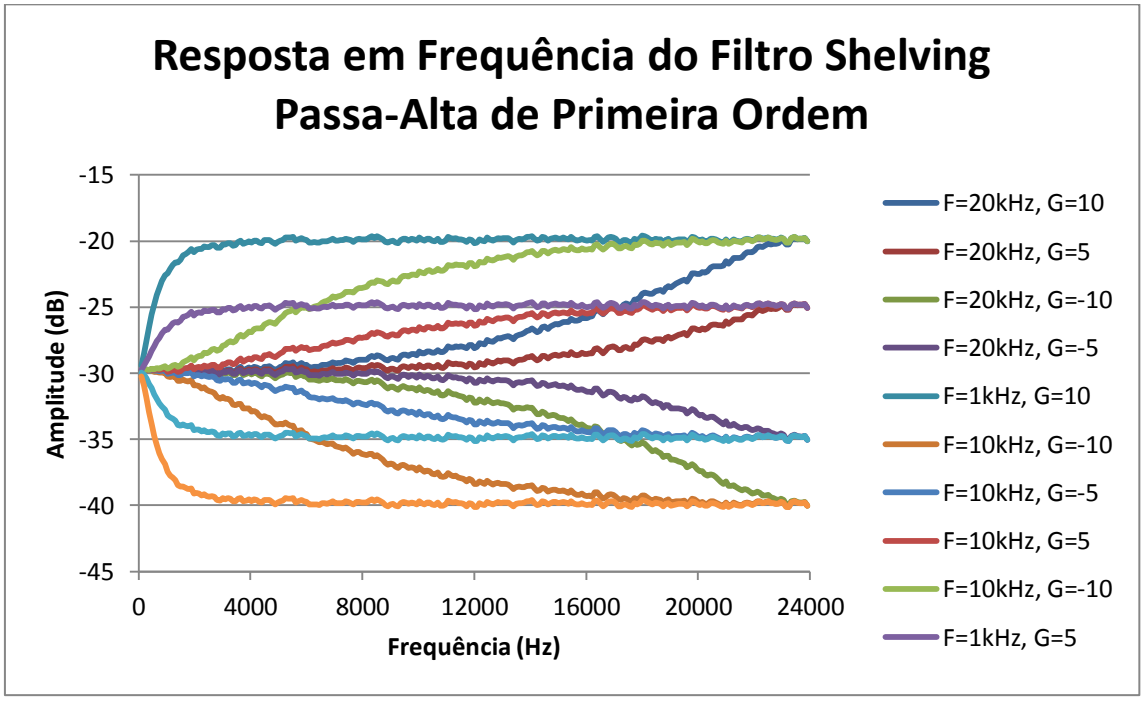

Figura 41: Resposta em Frequência do Filtro Shelving Passa-Alta de Primeira Ordem.

Antes de explicar mais profundamente esses filtros vale ressaltar um detalhe. Ambos os gráficos das Figuras 41 e 42 têm a mesma variação de parâmetros, frequências de corte de 1kHz, 10kHz e 20kHz, e ganhos de -10dB, -5dB, 5dB e 10dB. De primeiro relance, a Figura 41 aparenta ter mais curvas que a Figura 42, o que não é verdade. Essa falsa impressão é gerada pela velocidade de convergência dos filtros, que é mais rápida no filtro de segunda ordem.

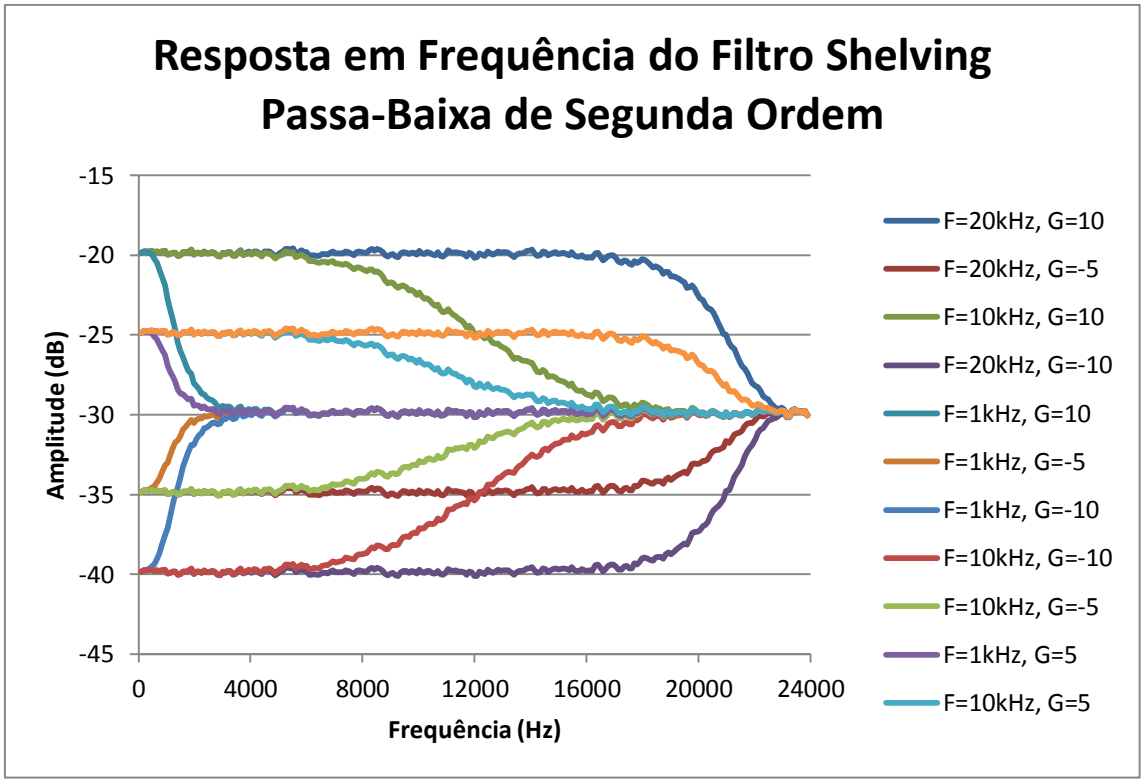

Figura 42: Resposta em Frequência do Filtro Shelving Passa-Baixa de Segunda Ordem.

Os filtros Shelving não obedecem à relação clássica de atenuação de magnitude de -20dB/década por ordem do filtro. A inclinação do comportamento da amplitude na frequência é dependente da frequência de corte e do ganho aplicado, que são as prioridades dos filtros Shelving.

Os ganhos definem pontos fixos na resposta em frequência desses filtros, pontos iniciais nos filtros do tipo passa-baixa e finais nos do tipo passa-alta, o que pode ser facilmente notado nas Figuras 41 e 42. O segundo ponto fixado é na frequência de corte, ponto que se encontra a -3dB do valor de ganho definido.

Como mencionado a atenuação por década dos filtros Shelving não é constante, mas é possível concluir que para dados o ganho e a frequência de corte, o filtro de segunda ordem apresenta o dobro da atenuação por década quando comparado ao filtro de primeira ordem.

O último filtro implementado no programa é o chamado “Peak”. Este filtro tem o comportamento mais complexo de todos apresentados, pois suas

características dependem dos parâmetros de entrada, como frequência de corte, banda passante ou rejeitada e ganho.

As Figuras 43 a 46 apresentam o comportamento deste filtro no domínio da frequência com diversos parâmetros sendo variados, cada figura com o objetivo de demonstrar características específicas do filtro.

A Figura 43 apresenta a resposta em frequência do filtro Peak para diversas frequências de corte, com ganho de 15dB e banda passante de 100Hz. Nesta figura é novamente notável a simetria em relação a frequência normalizada, especialmente nas curvas de 5kHz e 20kHz.

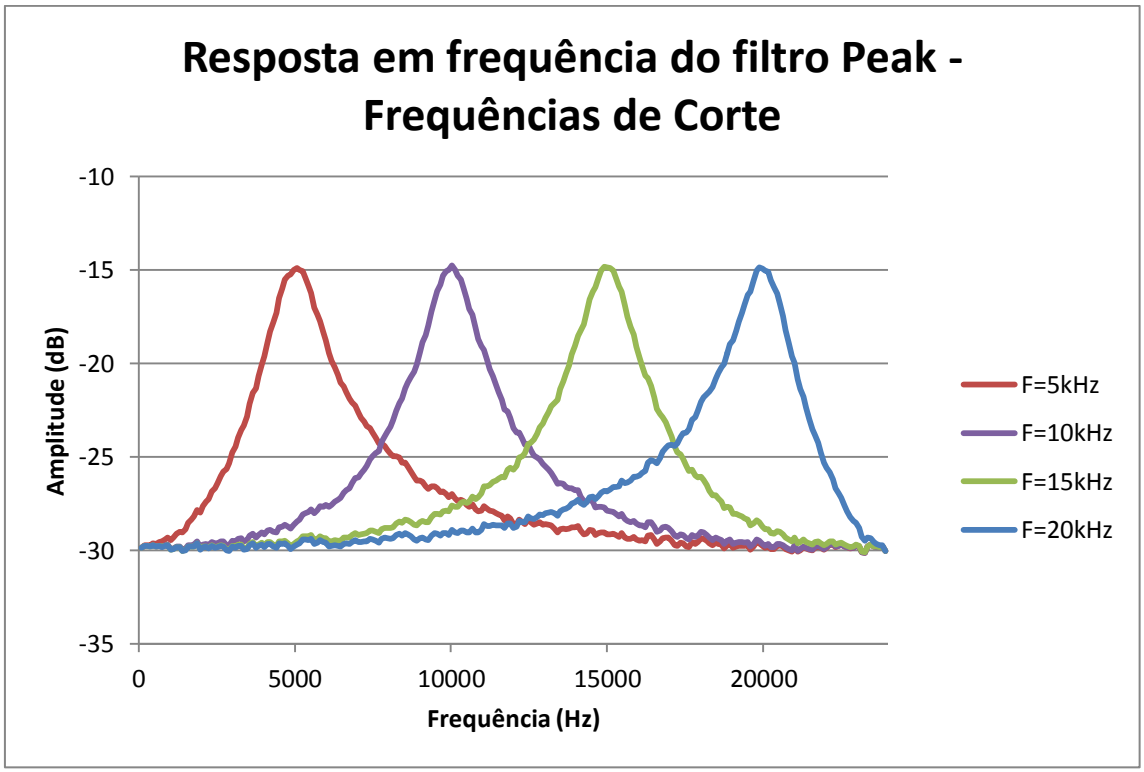

Figura 43: Resposta em frequência do filtro Peak - Frequências de Corte.

A resposta apresentada é a ideal do filtro, com todos os parâmetros sendo representativos (ganho, frequência de corte e banda passante).

Na Figura 44 o ganho aplicado foi constante e de 10dB, a frequência de corte permaneceu inalterada em 10kHz e a banda passante foi variada de 1Hz até 5kHz.

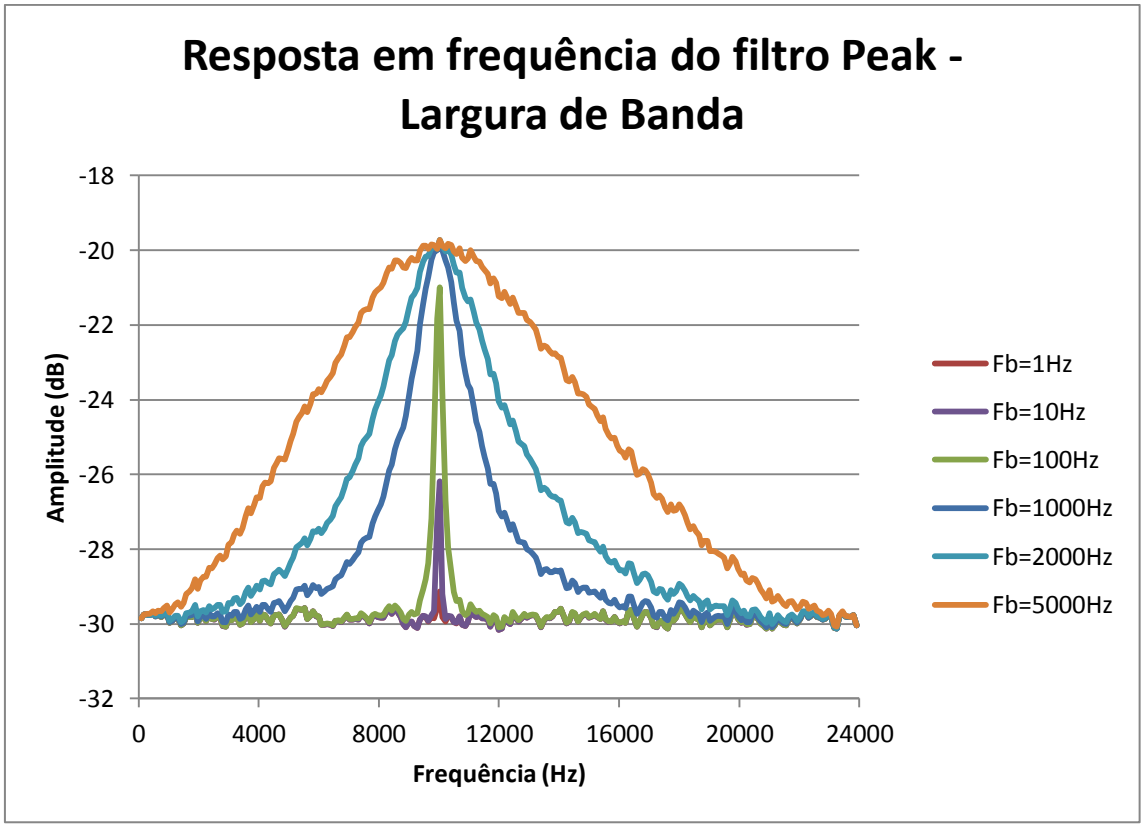

Figura 44: Resposta em frequência do filtro Peak - Largura de Banda.

O primeiro fato a destacar é que o ganho não permaneceu constante e representativo em todas as faixas passantes. Para bandas de 1kHz ou mais, este ganho era verdadeiro, entretanto, quanto menor a banda a partir deste valor, menor o ganho. Neste caso, apesar de o ganho não ser representativo, a banda passante é. Em outras palavras, os sinais amplificados estão dentro das bandas especificadas.

A questão é que o ganho e a faixa de frequência passante são diretamente relacionados. Assim, ao definir um desses parâmetros se restringe o outro.

A Figura 45 apresenta variados ganhos, de -20dB a +20dB, para frequência de corte fixa em 10kHz e frequência de banda passante de 1kHz. O intuito deste gráfico é demonstrar que para valores utilizáveis na prática de ganho, o filtro é sempre representativo com a faixa de 1kHz de banda passante.

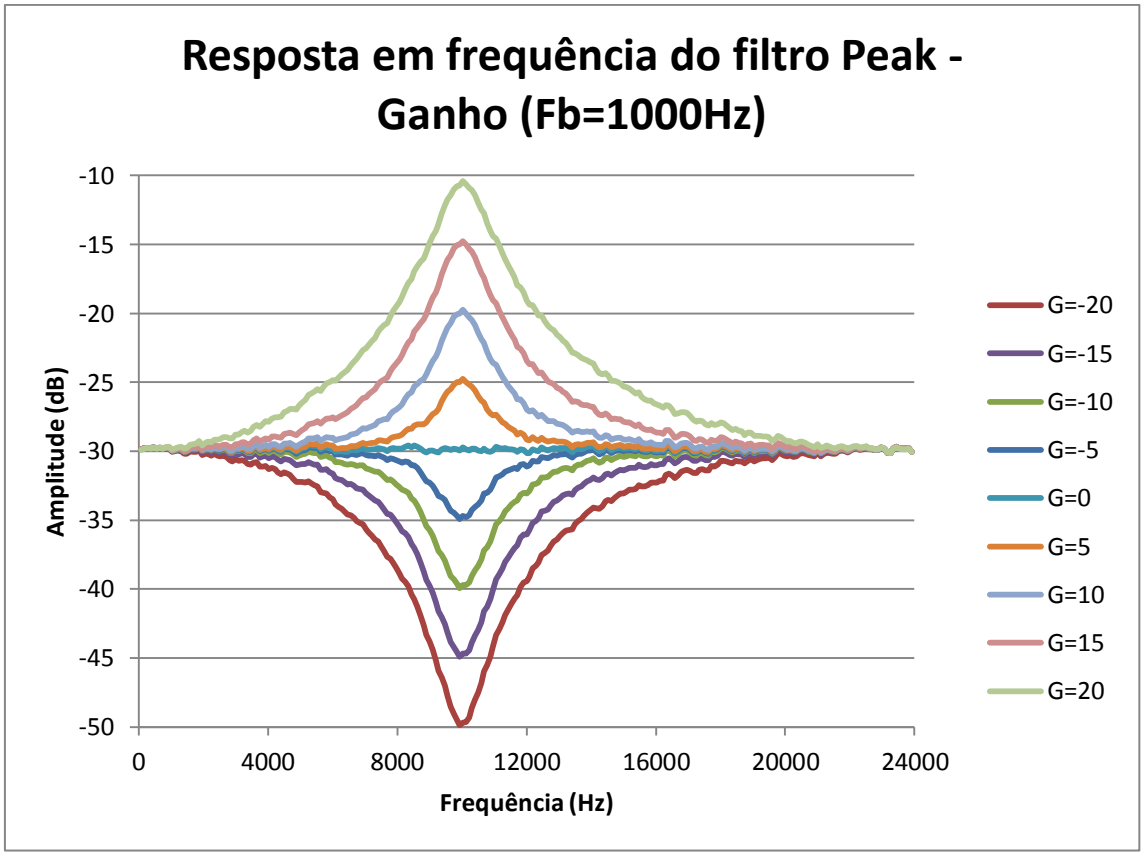

Figura 45: Resposta em frequência do filtro Peak - Ganho (Fb=1000Hz).

A Figura 46 visa explorar um pouco mais os limites da faixa passante sobre o desempenho do filtro. Nesta figura a frequência de corte foi mantida em 10kHz e a faixa passante em 10Hz. O ganho foi variado de -30dB a +30dB.

O pico de ganho alcançado foi em -8,8dB, o que implica em um ganho de +21,2dB quando o parâmetro ganho foi definido como +30dB. Note que as curvas não são simétricas a linha de -30dB (potência do ruído aplicado). A maior atenuação obtida com o ganho em -30dB foi de -12,1dB. Conclui-se que para ganhos negativos o filtro é menos eficaz que para ganhos positivos em bandas passantes curtas.

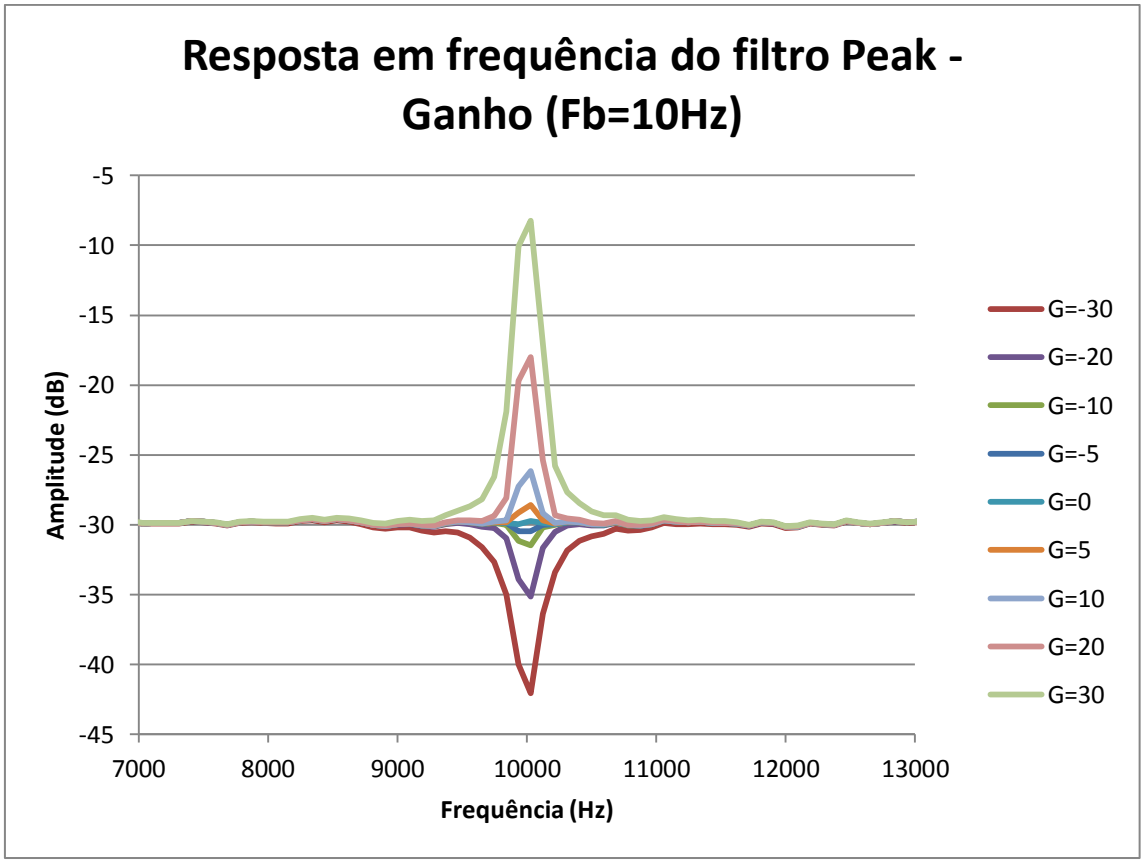

Figura 46: Resposta em frequência do filtro Peak - Ganho (Fb=10Hz).

O formato dos picos, entretanto, é sempre representativo e idêntico em todas as curvas. Logo, podemos concluir que é possível aplicar filtros com todos os ganhos desejados na prática (usualmente utiliza-se de -15dB a +15dB) em equalizadores, mesmo com faixas passantes pequenas. Entretanto, os valores dos parâmetros aplicados não serão mais representativos. Seria necessário aplicar o filtro variando os parâmetros até encontrar a resposta em frequência desejada.

Este comportamento é encontrado em todos os softwares que dispõe de filtros Peak para o usuário. Normalmente avisando o cliente de que os valores aplicados podem não ser verdadeiros.

5.2 Resultados dos Parâmetros do Arquivo BWF

Nesta seção serão apresentados os resultados obtidos dos parâmetros calculados para gerar o arquivo padrão BWF, especificamente, a potência sonora e a faixa de potência sonora.

A EBU fornece arquivos para a verificação e calibração da programação dos algoritmos fornecidos pela ITU. Junto com os arquivos padrões eles dispõe uma ficha técnica especificando os detalhes importantes.

A Tabela 6 apresenta os resultados do cálculo da potência sonora cuja norma de verificação e calibração pode ser encontrada na referência [11].

Tabela 6: Resultados do Cálculo da Potência Sonora.

Sinal de Teste	Potência (LUFS)	Resposta e Tolerância Aceita (LUFS)
Seno 1kHz -20LUFS	-19,998	-20 ± 0,1 LUFS
Seno 1kHz -26LUFS	-26,009	-26 ± 0,1 LUFS
Seno 1kHz -40LUFS	-40,076	-40 ± 0,1 LUFS
Sequência 1	-23,005	-23 ± 0,1 LUFS
Sequência 2	-33,026	-33 ± 0,1 LUFS
Sequência 3	-23,066	-23 ± 0,1 LUFS
Sequência 4	-23,066	-23 ± 0,1 LUFS
Sequência 5	-23,030	-23 ± 0,1 LUFS
Sequência 6-5C	-23,071	-23 ± 0,1 LUFS
Sequência 6-6C	-23,071	-23 ± 0,1 LUFS

Sequência 7	-22,980	-23 ± 0,1 LUFS
Sequência 8	-23,013	-23 ± 0,1 LUFS

Os resultados apresentados não foram obtidos diretamente após programação do algoritmo. Foi necessária uma etapa de calibração do filtro tipo K (filtro que simula a resposta do ouvido humano). Este filtro foi detalhado no Capítulo 4 e será descrito novamente por conveniência. O filtro tipo K é formado por dois filtros, um passa-alta com frequência de corte em 100Hz e um Shelving com ganho de 4dB e frequência de corte em 1kHz.

Os arquivos que continham a senóide de 1kHz foram primeiramente utilizados para calibrar o ganho e a frequência exata do filtro Shelving que compõe o filtro tipo K. Os sinais de teste correspondentes a sequencia 1 a 6C, são arquivos de som contendo variações de senos de 1kHz. Estes foram utilizados para a calibração fina do filtro Shelving e dos ganhos dos canais (detalhe não especificado neste trabalho).

As sequências 7 e 8, que são áudio de programas de rádio, são, então, utilizadas para calibrar a frequência de corte exata do filtro passa alta que também compõe o filtro tipo K. Note que a qualidade e correta implementação dos filtros foi vital para poder realizar esta etapa deste processo.

Os resultados apresentados na Tabela 6 foram os obtidos após esta calibração ter sido feita. Todos os valores calculados se encontram dentro das faixas aceitas definidas pela EBU.

A Tabela 7 apresenta os resultados do cálculo da faixa de potência sonora. O filtro tipo K utilizado neste algoritmo foi constituído dos filtros calibrados do cálculo da potência e os resultados apresentados foram obtidos diretamente ao rodar o programa. Todos os valores gerados pelo programa se encontram dentro da faixa especificada pela EBU, garantindo, assim, o padrão do arquivo BWF.

Tabela 7: Resultados do Cálculo da Faixa de Potência Sonora.

Sinal de Teste	Faixa de Potência (LU)	Resposta e Tolerância Aceita (LU)
Sequência 1	10,008	10± 1 LU
Sequência 2	5,000	5 ± 1 LU
Sequência 3	20,024	20 ± 1 LU
Sequência 4	15,013	15 ± 1 LU
Sequência 5	4,618	5 ± 1 LU
Sequência 6	15,212	15 ± 1 LU

Como já mencionado anteriormente, os outros parâmetros que fazem parte do arquivo BWF são gerados pelo usuário ou pelo criador do arquivo, não tendo especificações como os parâmetros acima.

É provável que nos próximos anos a EBU forneça arquivos e uma norma para a verificação e calibração do algoritmo de detecção do pico máximo verdadeiro do sinal, o que não foi apresentado até o momento.

No Capítulo 6 são apresentadas as conclusões deste trabalho.

6. CONCLUSÕES

Neste trabalho foi apresentado o desenvolvimento de um programa que equaliza arquivos WAVE e gera o arquivo padrão BWF. Foi visto que os resultados de todos os filtros foram de acordo com os esperados e que o conjunto de filtros implementados concede um alto controle da resposta em frequência desejada. Os resultados dos parâmetros do BWF estavam dentro dos limites estabelecidos pelo padrão fornecido pela EBU, assim, o programa gera a versão dois completa deste arquivo.

Existem melhorias a serem feitas no programa, principalmente um estudo de memória e velocidade de processamento do mesmo. Estes quesitos foram levados em consideração durante a programação, entretanto, nenhuma forma de medição foi aplicada para se obter resultados claros a respeito dos limites de memória e velocidade do programa.

A programação foi feita de tal forma que o arquivo era mapeado por inteiro na memória, significando que para arquivos extensos (horas de áudio) o programa provavelmente não será hábil a alocar toda a memória necessária. A utilização de buffers circulares resolveria o problema da memória, entretanto, comprometeria a velocidade de processamento do programa. Assim, um estudo seria vital para aumentar a eficiência do programa nestes quesitos.

A distorção de fase dos filtros não foi descrita, pois se sabe da teoria, que filtros FIR têm desempenho linear de fase, diferentemente dos filtros IIR que apresentam distorção não linear da fase do sinal. Então, para aplicação direta, caso a fase seja uma questão importante utiliza-se filtros FIR, caso contrário os filtros IIR também estão disponíveis. O estudo detalhado da distorção de fase é um trabalho a ser empregado no futuro, com o objetivo de melhor compreender o comportamento dos filtros utilizados na equalização do áudio.

Outro trabalho futuro a ser desenvolvido é a implementação de compressores e expansores no programa. O que habilitará o programa a não

só aumentar drasticamente a inteligibilidade do áudio, mas, também, gerar um arquivo com potência sonora uniforme, contribuindo aos espectadores de rádio e televisão que querem um volume padronizado nas programações.

7. REFERÊNCIAS

[1] EBU – TECH 3285: Specification of the Broadcast Wave Format (BWF). Acesso em 5 de abril de 2014, disponível em http://www.ebu.com.

[2] WAVE PCM Soundfile Format. Acesso em 11 de abril de 2014, disponível em https://ccrma.stanford.edu/courses/422/projects/WaveFormat/.

[3] RIFF File Structure. Acesso em 5 de maio de 2014, disponível em http://www.johnloomis.org/cpe102/asgn/asgn1/riff.html.

[4] GOMES, A.; VERUNICHI, E. (2004). *Fundamentos da Programação de Computadores. Prentice-Hall.*

[5] MANZANO, J. (2003). *Linguagem C*. Editora Érica.

[6] The C Resources Network. Primeiro acesso em 4 de abril de 2014, disponível em http://www.cplusplus.com/.

[7] Digital Filters. Primeiro acesso em 23 de abril de 2014, disponível em http://dea.brunel.ac.uk/cmsp/Home_Saeed_Vaseghi/Chapter05-DigitalFilters.pdf.

[8] ZOLZER, U. (2002). Digital Audio Effects. John Willey & Sons, Ltd.

[9] ZOLZER, U. (2008). Digital Audio Signal Processing – Second Edition. John Willey & Sons, Ltd.

[10] ITU-R BS.1770-2 (03/2011): Algorithms to measure audio programme loudness and true-peak audio level. Acesso em 15 de maio de 2014, disponível em www.itu.int.

[11] EBU – TECH 3241: 'EBU Mode' metering to supplement Loudness normalisation. Acesso em 20 de maio de 2014, disponível em http://www.ebu.com.

[12] EBU – TECH 3242: Loudness Range measure to supplement loudness normalisation. Acesso em 20 de maio de 2014, disponível em http://www.ebu.com.