Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Unidade Acadêmica de Engenharia Elétrica

PEDRO MERENCIO PRIMO PASSOS

IMPLEMENTAÇÃO DIGITAL DE EFEITOS DE ÁUDIO:

ATRASO & REVERBERAÇÃO

Campina Grande, Paraíba
Junho de 2015

PEDRO MERENCIO PRIMO PASSOS

IMPLEMENTAÇÃO DIGITAL DE EFEITOS DE ÁUDIO:

ATRASO & REVERBERAÇÃO

Projeto de Engenharia Elétrica
submetido à Unidade Acadêmica de Engenharia
Elétrica da Universidade Federal de Campina
Grande como parte dos requisitos necessários
para a obtenção do grau de Bacharel em
Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento Digital de Sinais

Orientador:
Professor Edmar Candeia Gurjão, D. Sc.

Campina Grande, Paraíba
Junho de 2015

PEDRO MERENCIO PRIMO PASSOS

IMPLEMENTAÇÃO DIGITAL DE EFEITOS DE ÁUDIO:

ATRASO & REVERBERAÇÃO

Projeto de Engenharia Elétrica submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento Digital de Sinais

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor Edmar Candeia Gurjão, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

AGRADECIMENTOS

Primeiramente a Deus.

Agradeço também a minha mãe, Tereza Neuma, ao professor Edmar Candeia Gurjão por toda atenção dada não somente durante as atividades deste trabalho, mas também durante todo meu longo caminho nesta graduação. Por fim, agradeço a todos que, na sinceridade da consciência, sabem que me fizeram bem.

A meus avós e à Adail Silva Paz.

There is no emotion, there is peace.
There is no ignorance, there is knowledge.
There is no passion, there is serenity.
There is no chaos, there is harmony.
There is no death, there is the Force.
—The Jedi Code

RESUMO

Este trabalho utiliza técnicas de processamento digital de sinais para manipular arquivos de áudio, digitalizados em formato padrão WAV, adicionando a eles um efeito de *delay* ou de *reverb*. A programação é realizada na plataforma MATLAB®, versão R2009a, da MathWorks®, com base em algoritmos sugeridos na bibliografia. A implementação destes efeitos dar-se pela utilização de três tipos básicos de filtros digitais, a saber, o filtro de resposta ao impulso finita (FIR), o filtro de resposta ao impulso infinita (IIR) e o filtro passa-tudo (PT). Os códigos desenvolvidos apresentaram resultados satisfatórios, onde foi possível averiguar as características de cada efeito implementado por meio da escuta de um arquivo de áudio editado, bem como pela observação de representações gráficas tão comuns na análise de sinais.

Palavras-chave: PDS, áudio, *delay*, *reverb*.

SUMÁRIO

1. INTRODUÇÃO

Na música existem diversas técnicas de manipulação do som que geram diferentes efeitos audíveis das mais diversas qualidades que, combinados com o crescimento constante da capacidade de processamento dos dispositivos, podem ser implementadas digitalmente por métodos de programação adequados. A teoria da Análise de Sinais e Sistemas e todo o ferramental matemático disponível é o principal conhecimento necessário para trabalhar com estes efeitos. Noções de psicoacústica e de princípios musicais podem facilitar o trabalho de projetar e realizar sistemas de áudio.

Dentre os efeitos de natureza temporal, tem-se o atraso e a reverberação (em inglês, o *delay* e o *reverb*). O primeiro consiste de um retardo temporal do sinal sonoro, a atenuação da sua amplitude e, por fim, sua adição ao sinal que o originou. O atraso do som é denominado como tempo de *delay*, e a variação da sua magnitude é o ganho de *delay*. A manipulação destes parâmetros possibilita gerar outros efeitos temporais de áudio, como o *flanger* e o *echo*, por exemplo [4]. A reverberação é a múltipla repetição do som original. Na natureza ela ocorre quando a onda sonora sofre diversas reflexões nas paredes de um ambiente fechado. Como um efeito de áudio, ela é implementada com o uso do *delay* como principal estrutura. No ramo musical estes dois efeitos são comumente designados pelos seus nomes na língua inglesa. Portanto, estes serão utilizados deste ponto em diante neste trabalho.

Neste trabalho utilizou-se a plataforma MATLAB® para implementar algoritmos que realizam os dois efeitos acima citados. Os códigos foram escritos com base nos métodos descritos segundo Zölzer [4] para serem usados em sistemas em tempo real, como em um pedal de guitarra, por exemplo. Este relatório está dividido em 4 capítulos. Em que o segundo expõe os fundamentos envolvidos no contexto dos efeitos de áudio; o terceiro traz o desenvolvimento dos códigos, bem como os resultados obtidos ao submeter um arquivo digital de áudio a estes métodos; e o ultimo capítulo apresenta as conclusões do trabalho. Os algoritmos escritos veem em anexo ao fim do documento.

2. ÁUDIO E SEUS EFEITOS

Alguns conceitos básicos são pré-requisitos para implementação de efeitos de áudio. Dentre estes estão a definição de som, sua geração, propagação e percepção; o que são efeitos de áudio; e as demais noções contidas no contexto citado. A seguir, estas definições serão detalhadas.

2.1 Som

O som é composto de ondas mecânicas que se propagam em um meio material. As ondas sonoras audíveis são aquelas cujas componentes de frequência encontram-se na faixa de 20 a 20.000 Hertz (Hz). Além disso, a amplitude destas ondas também possuem valores limitantes relacionados com o limiar mínimo para que se possa sentir o efeito do som, e o máximo que pode provocar danos à saúde. Estes, todavia, dependem da frequência do sinal, e são oriundos das características fisiológicas do sistema auditivo humano.

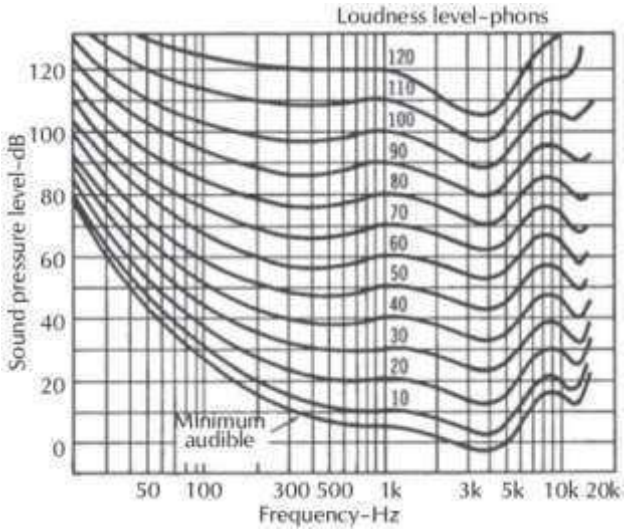

Figura 1: Curvas isofônicas de tons puros em um campo sonoro frontal para humanos com acuidade auditiva média determinadas por Robinson e Dadson.
Fonte: Ballou, Glen. Handbook for Sound Engineers – 4ª ed. Pg. 52.

As curvas isofônicas – como as apresentadas na Figura 1 – são representações gráficas que informam os limites audíveis de amplitude para cada componente de frequência do som [1].

O som é gerado pelas contrações e expansões periódicas do meio material, como o ar ou a água. Sua propagação é de forma longitudinal através do meio, com velocidade proporcional ao comprimento e à frequência da onda, ou seja:

$$v = \lambda \cdot f \tag{1}$$

Onde:
- v = é a velocidade da onda, em metros por segundo (m/s);
- λ = é o comprimento da onda, em metros (m);
- f = é a frequência da onda, em Hz.

A percepção do som acontece através do sistema auditivo. O órgão responsável por esta tarefa é o ouvido, que nos humanos é constituído de três partes: o ouvido externo, o ouvido médio e o ouvido interno, como ilustra a Figura 2 seguinte.

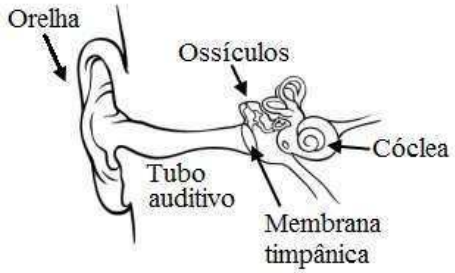

Figura 2: Estrutura do ouvido humano [1]

No ouvido externo o som é captado pela orelha e conduzido ao ouvido médio através do tubo auditivo. Já no ouvido médio a onda sonora é amplificada pela vibração dos ossículos em contato com a membrana timpânica que limita a região. Por fim, no ouvido interno, a informação sonora é convertida em sinal elétrico pela cóclea e transmitida até o cérebro onde é interpretada.

A percepção auditiva é mais do que simplesmente captar os sons ao redor. A interpretação dos sons, bem como todas as sensações provocadas por ele são objetos de estudo da psicoacústica [1]. O estudo dos sons, pois, leva em consideração parâmetros objetivos e subjetivos. Os parâmetros objetivos, tais como a frequência e a amplitude, fornecem características físicas do som. Por outro lado, os parâmetros subjetivos, como a agradabilidade, ou até os sentimentos causados ao ouvir um determinado som, são características cuja avaliação depende do ouvinte.

2.2 Efeitos de áudio

Os efeitos de áudio são fenômenos que ocorrem em um sinal sonoro que modificam seu timbre. Ou seja, alteram as componentes harmônicas do som sem variar, em geral, a sua frequência fundamental. Exemplos destes efeitos são: atraso, reverberação, distorção, equalização, etc.

Alguns efeitos podem originar-se de forma natural, por meio dos fenômenos físicos na propagação de uma onda, como a reflexão e a refração. Contudo, também é possível gerar efeitos de forma artificial com o uso de aparatos que interagem com a onda sonora, como tubos metálicos que ressoam enfatizando uma determinada faixa de frequências, ou por meio de dispositivos eletrônicos, onde a representação sonora, que originalmente dar-se como variação da pressão no meio de propagação, é convertida em uma representação elétrica, geralmente em nível de tensão, através de sensores, a fim de ser manipulada.

É possível realizar tais manipulações utilizando técnicas analógicas ou digitais. Na primeira, a informação sonora é captada, modificada e reproduzida por meio de circuitos analógicos, onde não há a discretização da informação. Um exemplo de tal sistema é o conjunto de pedais de efeito usados por um guitarrista.

No formato digital, o sinal de áudio é processado via um algoritmo capaz de gerar o efeito desejado. Para isso ser possível, a onda sonora de natureza analógica precisa ser convertida para o âmbito digital. Tal conversão é feita por um bloco chamado Conversor Analógico-Digital (ADC), que faz a amostragem e quantização do sinal. A amostragem consiste na extração do valor da amplitude do sinal em instantes de tempo igualmente espaçados entre si. O intervalo entre esses instantes é chamado de período de amostragem. Tal período deve ser, no mínimo, duas vezes mais rápido que a maior frequência presente no sinal de áudio [4].

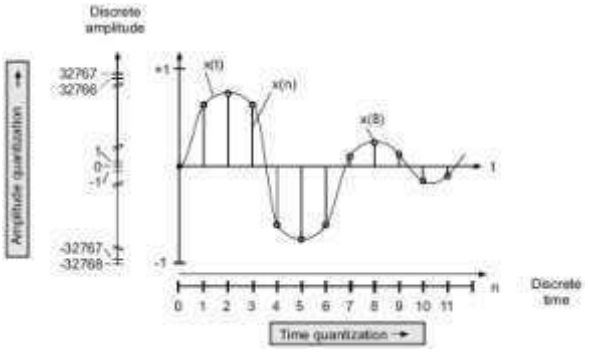

Figura 3: Digitalização de um sinal analógico.

Fonte: Zölzer, Udo. Digital Audio Signal Processing, 2nd ed. Pg. 25.

A quantização é o truncamento dos valores de amplitude em números discretos que podem ser salvos em registradores digitais.

2.3 *Delay*

Na teoria de sinais, o *delay* – em português, atraso – é um retardo temporal do sinal. Este também pode sofrer variação na sua amplitude, como mostra a Figura 4 seguinte.

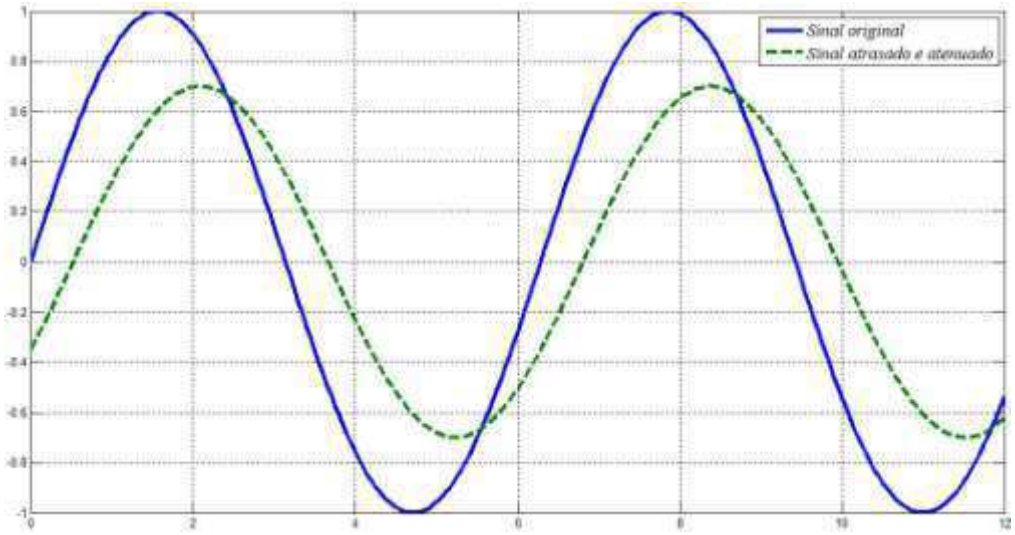

Figura 4: Sinal atrasado e atenuado.

O *delay* pode ocorrer naturalmente devido às reflexões que a onda sonora sofre durante sua propagação no ambiente em que está contida, como mostrado na Figura 5. A distância percorrida por cada onda refletida até chegar ao ouvinte causa o atraso do sinal. Várias condições podem influenciar o *delay*, dentre elas estão: a posição relativa entre a fonte sonora e o ouvinte; a geometria e as dimensões do ambiente de propagação; a amplitude do som; as frequências presentes no sinal; a dissipação da energia das ondas durante suas propagações; etc.

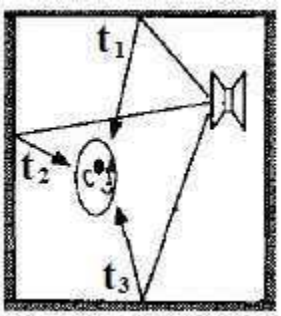

Figura 5: Onda sonora se propagando em um ambiente fechado.

Cada raio t_i representa uma possível trajetória percorrida pelo som.

Do ponto de vista de efeito de áudio, o *delay* é a soma da onda sonora original com suas réplicas atrasadas. Resultando numa onda de mesma frequência fundamental, porém, com suas componentes harmônicas modificadas. A Figura 6 ilustra o resultado da soma da onda original com uma única cópia sua atrasada.

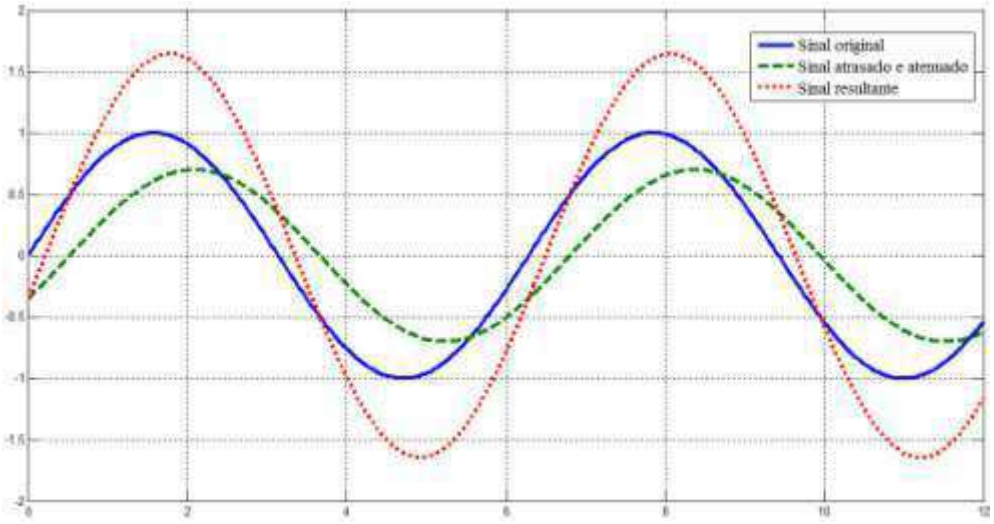

Figura 6: Sinal resultante da soma do sinal original com o sinal atrasado.

A diferente extensão do atraso de cada onda refletida faz com que o som percebido adquira qualidades diferentes. Daí surgem os diversos efeitos de áudio de natureza temporal, tais como o *vibrato*, quando se escuta apenas a onda repetida e com variação periódica do tempo de atraso, tipicamente o tempo de *delay* fica entre 5 e 10 milissegundos (ms) e varia com frequência de 5 até 14 Hz; o *flanger*, com tempo de atraso de até 15 ms e variação de até 1 Hz, e escuta-se também o som original; e o *echo*, quando o tempo de atraso é constante e superior a 50 ms. O *delay* é, portanto, utilizado como operação básica na execução destes diversos efeitos sonoros.

2.4 *Reverb*

O *reverb* – no português, reverberação – é um fenômeno de áudio de natureza temporal. Ocorre em ambientes fechados quando as ondas refletidas continuam propagando-se após a onda original haver cessado. Estas ondas refletidas percorrem um caminho maior que o da onda direta até chegar ao ouvinte. Por isso há o atraso na percepção. As superfícies do ambiente também causam difusão nas ondas sonoras, e estas sofrem dissipação de energia durante a propagação no ar. O tempo necessário para que esta energia seja atenuada de 60 decibéis (dB) é definido como tempo de reverberação. Até que a amplitude da onda sonora

diminua à níveis não audíveis, o som sofre várias reflexões nas paredes do ambiente, podendo passar mais de uma vez por um mesmo ponto. Este fenômeno causa no ouvinte a sensação de ininterrupção, ou seja, ele continua captando o som durante um pequeno intervalo de tempo após sua fonte ter cessado a geração.

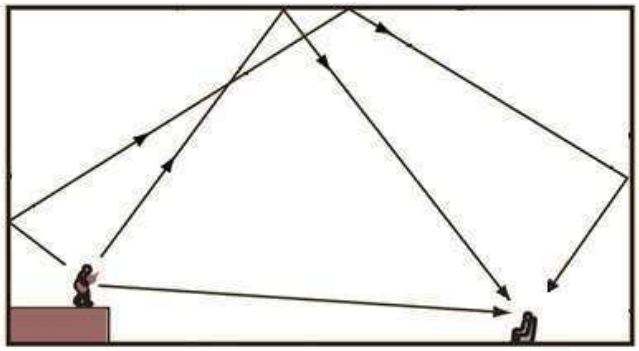

Figura 7: Reflexão do som em um ambiente fechado

A reverberação é, por vezes, confundida com o eco. A diferença entre estes dois fenômenos está na percepção. Quando o tempo de atraso entre o som original e o refletido é grande o suficiente para permitir que o ouvinte possa distingui-los, tem-se o eco. Quando este retardo entre os sons é muito pequeno, a audição humana não consegue identificar qual o som original e qual o refletido, o cérebro, então, interpreta a informação sonora como sendo um único som. Eis, pois, a reverberação.

É possível qualificar acusticamente um determinado ambiente através da sensação causada no ouvinte pela reverberação do som. Para tal, existem parâmetros psicoacústicos que podem ser obtidos experimentalmente. Exemplos destes parâmetros são: o *presence*, um parâmetro de percepção da fonte, relaciona a energia do som direto com o reverberado; o *running reverberance*, um parâmetro da interação entre a fonte e o ambiente, medido pelo tempo de decaimento da resposta impulsiva da sala; e *liveness*, um parâmetro de percepção do ambiente, dado pela variação em alta frequência do tempo de decaimento.

O *reverb* como efeito de áudio é implementado utilizando o *delay* como estrutura básica. Cada onda refletida é interpretada como o sinal original atrasado no tempo e atenuado. Os parâmetros que qualificam acusticamente um ambiente podem ser usados para determinar os tempos de cada *delay* e os seus respectivos ganhos. Todavia, dificilmente é possível definir independentemente cada parâmetro.

3. IMPLEMENTAÇÃO E APLICAÇÃO DOS EFEITOS DE ÁUDIO

Neste trabalho foi realizado a implementação dos efeitos de áudio *delay* e *reverb*. A plataforma MATLAB®, versão R2009a, da MathWorks®, foi usada para a escrita dos algoritmos de cada efeito. Para análise dos códigos foi utilizado primeiramente um sinal de áudio fictício, composto de um único pulso temporal de amplitude unitária e, em seguida, foi feito uso de um arquivo digital de áudio de curta duração e no formato padrão WAV. Exceto para o *reverb*, cujo teste foi diretamente com o arquivo de real áudio. As técnicas de programação aqui usadas foram desenvolvidas a partir dos códigos propostos por Zölzer [4], onde o sinal sonoro é processado amostra por amostra no tempo discreto. Isto permite que estas implementações sejam mais eficientes em sistemas em tempo real, onde o som é processado a medida que ocorre. Os áudios resultantes das aplicações destes métodos podem ser acessados pelo código QR, exposto na Figura 8 seguinte, com o uso de um aplicativo decodificador QR para *smartphone,* a exemplo do *QR Code Reader*, para a plataforma *Android*.

Figura 8: Código QR para acesso aos arquivos de áudio deste trabalho

3.1 Efeito de *delay*

Este efeito de áudio pode ser gerado digitalmente por meio de dois tipos de filtros: um de resposta ao impulso finita (FIR), ou um de resposta ao impulso infinita (IIR). Para tal, dois parâmetros são usados: o tempo de *delay* (Td), que é o intervalo de tempo, em segundos, entre o início do sinal original e o início da primeira repetição atrasada; e o ganho do *delay* (Gd), dado pela razão entre a amplitude do sinal original e a amplitude da primeira reflexão retardada. Uma vez que o sinal de áudio já está discretizado, o tempo de *delay* deve,

idealmente, ser um múltiplo inteiro da frequência de amostragem (Fs) utilizada. Assim, este tempo pode ser diretamente convertido em quantidade de passos (M):

$$\mathbf{M = Td \cdot Fs.} \quad (2)$$

Todavia, o uso deste efeito pode solicitar um tempo de atraso que não seja múltiplo da frequência de amostragem. Para que isto não comprometa o processamento do sinal, a solução mais simples é aproximar o valor de M para o número inteiro (M') mais próximo. Neste caso, o erro percentual ($\varepsilon_{\%}$) do tempo de *delay* é dado por:

$$\mathbf{\varepsilon_{\%} = [(M-M') / M] \cdot 100\%.} \quad (3)$$

Como exemplo: seja a frequência de amostragem de 41,4 Hz e um tempo de *delay* de 53 ms. Pela equação (2), temos que M = 2194,2. O número inteiro mais próximo é, portanto, M' = 2194. O erro percentual cometido é, então, $\varepsilon_{\%}$ = 0,0091%. A aplicação na qual o efeito será utilizado dirá qual o limite aceitável para este erro.

Outra solução é através do uso de técnicas de interpolação para estimar o valor da amplitude do sinal num instante de tempo situado entre duas amostras. Esta solução exige maior esforço computacional e modifica a taxa de amostragem do sinal, porém, melhora a precisão do tempo de *delay* quando for necessário.

3.1.1 Filtro FIR

O filtro digital de resposta ao impulso finita gera um *delay* simples do sinal original, ou seja, uma única cópia atrasada do som. Para ser percebido é necessário que tanto o áudio de referência quanto sua réplica sejam ouvidos ao mesmo tempo. O diagrama que representa este sistema é mostrado na Figura 9 seguinte.

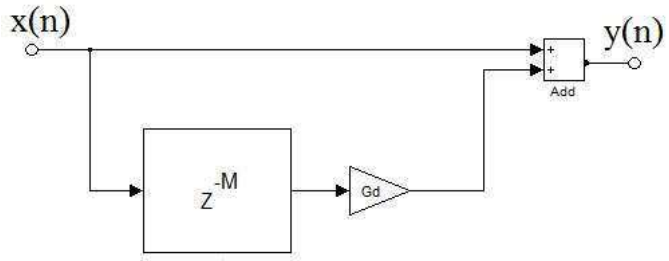

Figura 9: Filtro digital FIR.

A equação de diferença deste filtro e sua função de transferência são respectivamente:

$$\mathbf{y(n) = x(n) + Gd \cdot x(n-M)}\,, \tag{4}$$

$$\mathbf{H(z) = 1 + Gd \cdot z^{-M}}. \tag{5}$$

Sendo x(n) o áudio original e y(n) o áudio processado, ambos no tempo discreto. A reposta temporal deste sistema consiste do sinal original somado a sua versão atrasada, como ilustrado na Figura 10. A resposta deste filtro vista no domínio da frequência é apresentada na Figura 11. O algoritmo deste filtro está no Anexo 5.1.

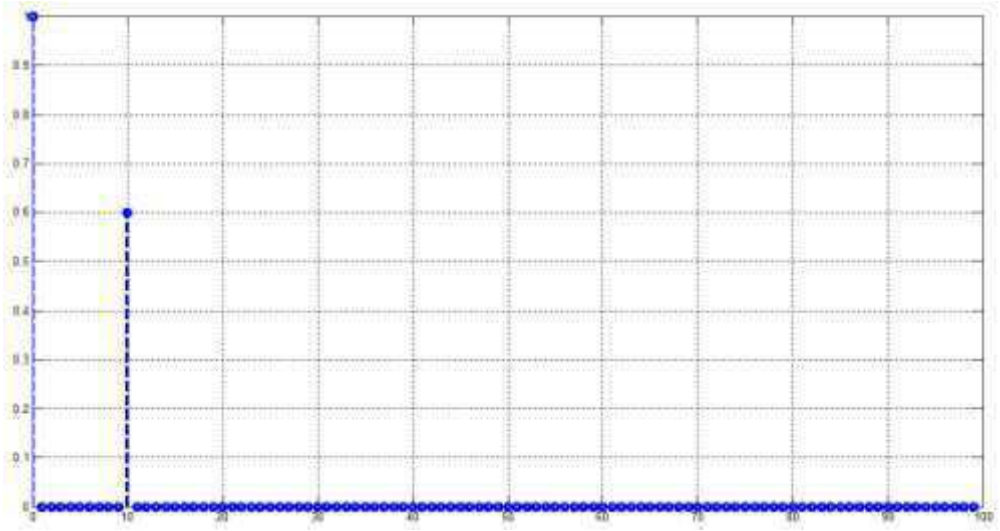

Figura 10: Resposta temporal do *delay* de 10 passos via filtro FIR, com ganho de 0,6, de um pulso unitário.

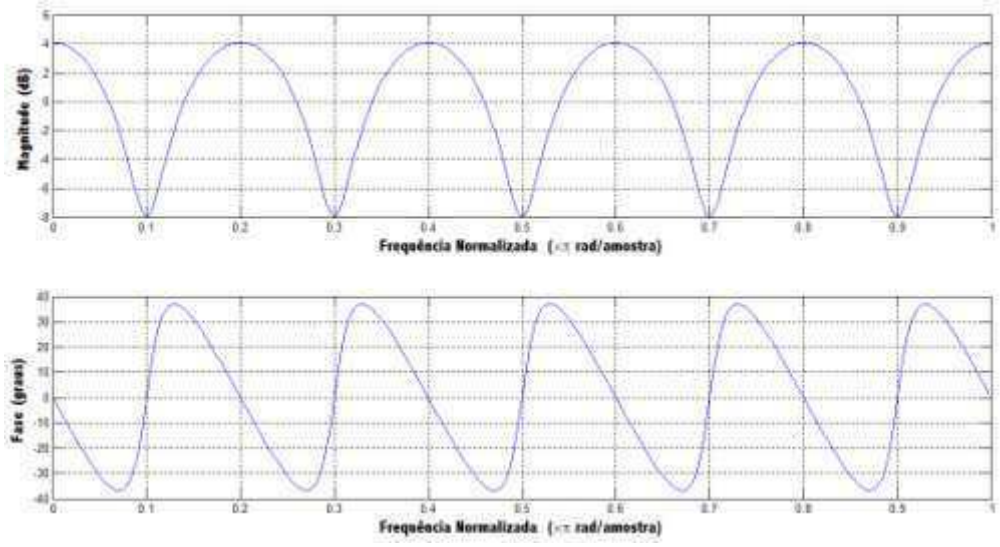

Figura 11: Resposta em frequência do *delay* de 10 passos feito pelo filtro FIR, com ganho de 0,6, de um pulso unitário.

No filtro FIR, quando o ganho Gd é positivo, as frequências múltiplas de (1/Td) são amplificadas, enquanto que as frequências intermediárias são atenuadas. Quando Gd é negativo ocorre o inverso, as frequências múltiplas de (1/Td) são atenuadas e as frequências intermediárias são amplificadas.

Este método, então, foi utilizado para aplicar um *delay* com 500 ms de atraso e ganho de 0,6 a um áudio digital com 9 segundos de duração e amostrado a 44,100 kHz. O algoritmo que realizou esta implementação encontra-se no Anexo 5.2. O resultado obtido está apresentado nas Figuras 12 e 13 seguintes. Graficamente, verifica-se que a mudança causada por este efeito é mais evidente na fase das componentes de frequência do áudio. No áudio gerado é possível identificar a única réplica atrasada do som.

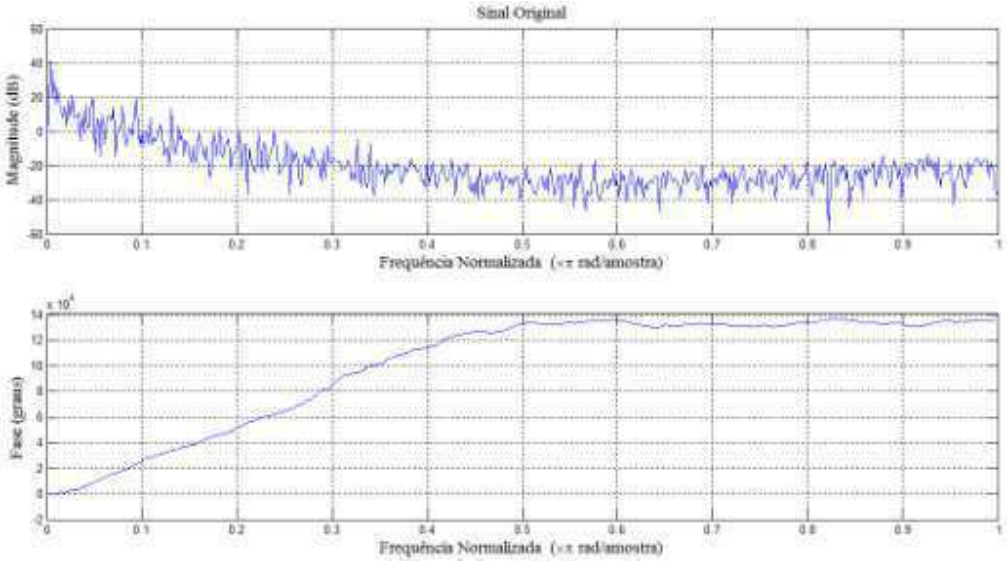

Figura 12: Áudio digital original.

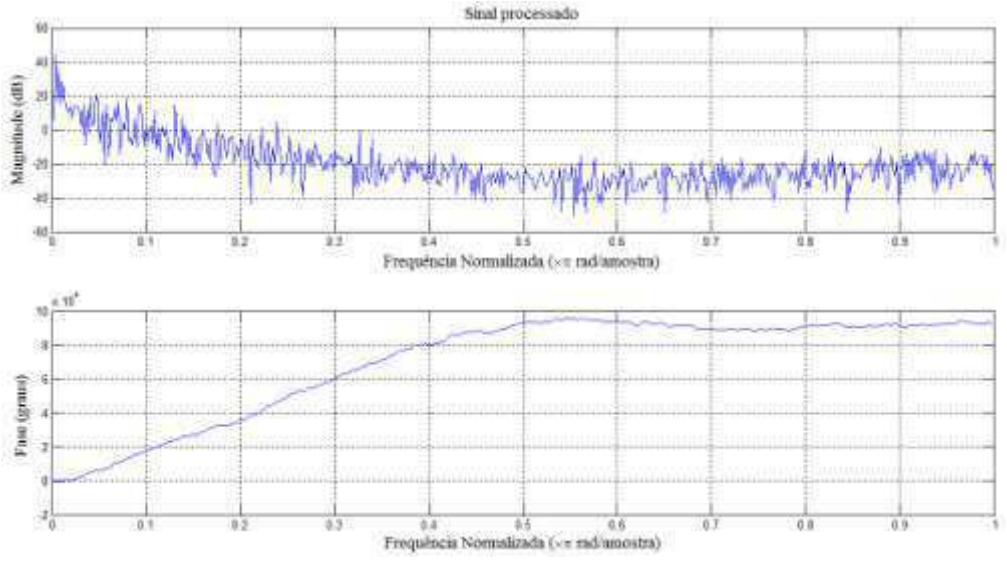

Figura 13: Áudio digital processado via filtro FIR com tempo de delay de 500 ms.

3.1.2 Filtro IIR

Outro filtro digital capaz de implementar o efeito de *delay* é o filtro de resposta ao impulso infinita. Neste a realimentação da saída é atrasada e somada a entrada do sistema. Isto faz com que haja inúmeras réplicas atrasadas do áudio, onde cada cópia sofre uma atenuação de Gd em relação a sua antecessora. O diagrama que representa este sistema é mostrado na Figura 14 seguinte.

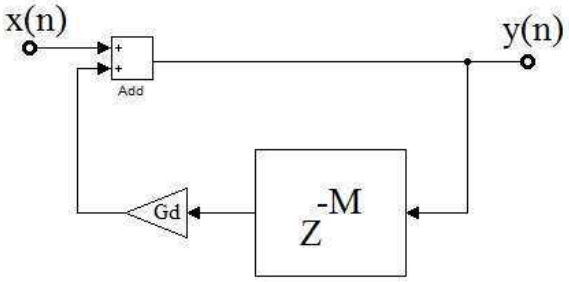

Figura 14: Filtro digital IIR.

A equação de diferença deste filtro e sua função de transferência são respectivamente:

$$\mathbf{y(n) = x(n) + Gd \cdot y(n-M),} \tag{6}$$

$$\mathbf{H(z) = 1 / (1 - Gd \cdot z^{-M}).} \tag{7}$$

A reposta temporal deste sistema consiste do sinal original somado a suas diversas cópias atrasadas e atenuadas, como ilustrado na Figura 15. A resposta deste filtro vista no domínio da frequência é apresentada na Figura 16. O algoritmo está no Anexo 5.3.

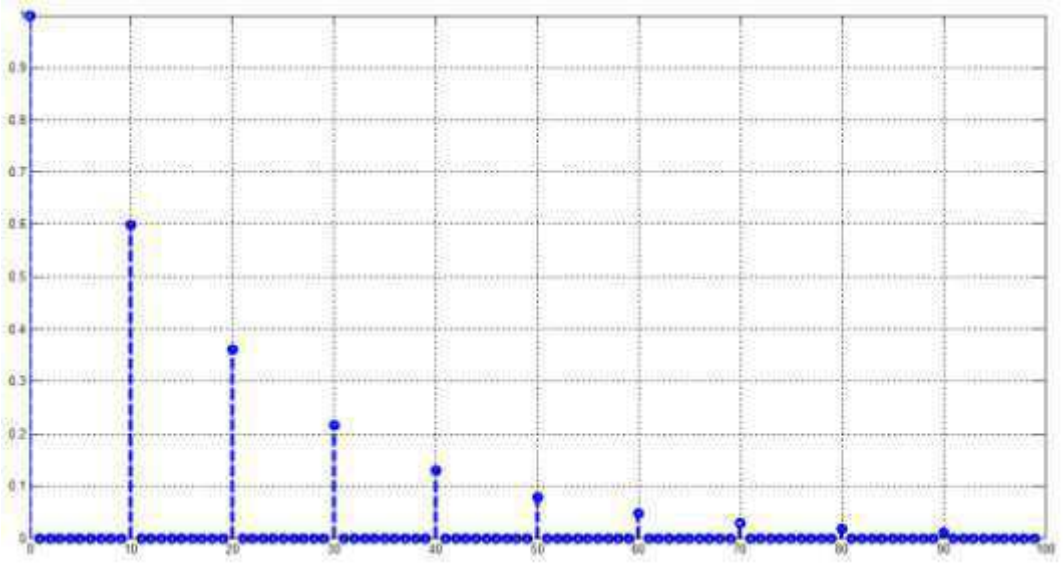

Figura 15: Resposta temporal do *delay* de 10 passos via filtro IIR, com ganho de 0,6, de um pulso unitário.

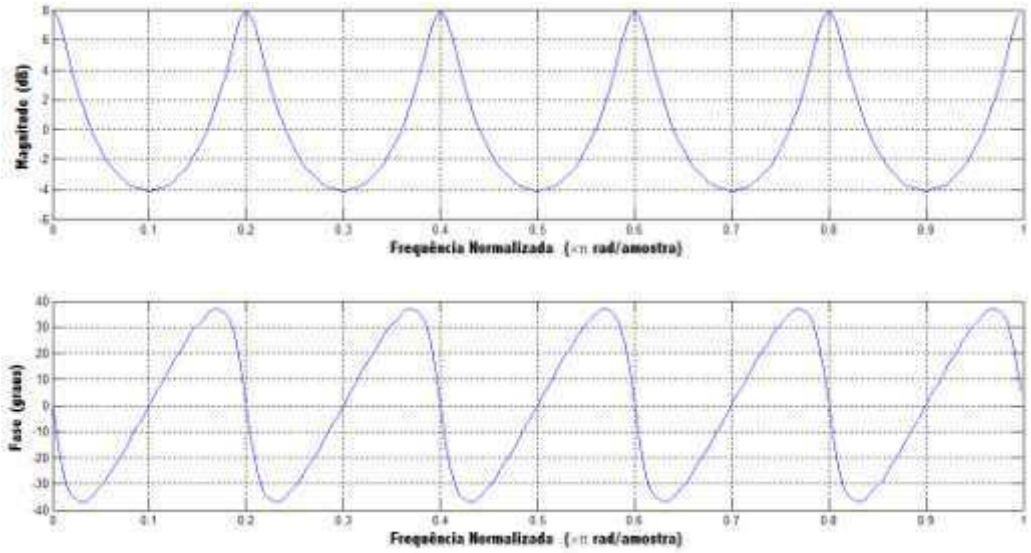

Figura 16: Resposta em frequência do *delay* de 10 passos feito pelo filtro IIR, com ganho de 0,6, de um pulso unitário.

Por haver uma malha de realimentação neste filtro, o ganho Gd deve possuir módulo ≤ 1, para garantir a estabilidade do sistema. A amplificação e atenuação das componentes de frequência são como no filtro FIR.

O filtro IIR foi utilizado para aplicar um *delay* com 500 ms de atraso e ganho de 0,6 ao mesmo arquivo de áudio ilustrado na Figura 12. O algoritmo desta implementação encontra-se no Anexo 5.4. O resultado obtido está exposto na Figura 17. Novamente é possível obervar o efeito na fase do sinal resultante. As inúmeras réplicas atrasadas e gradativamente atenuadas podem ser notadas no áudio gerado.

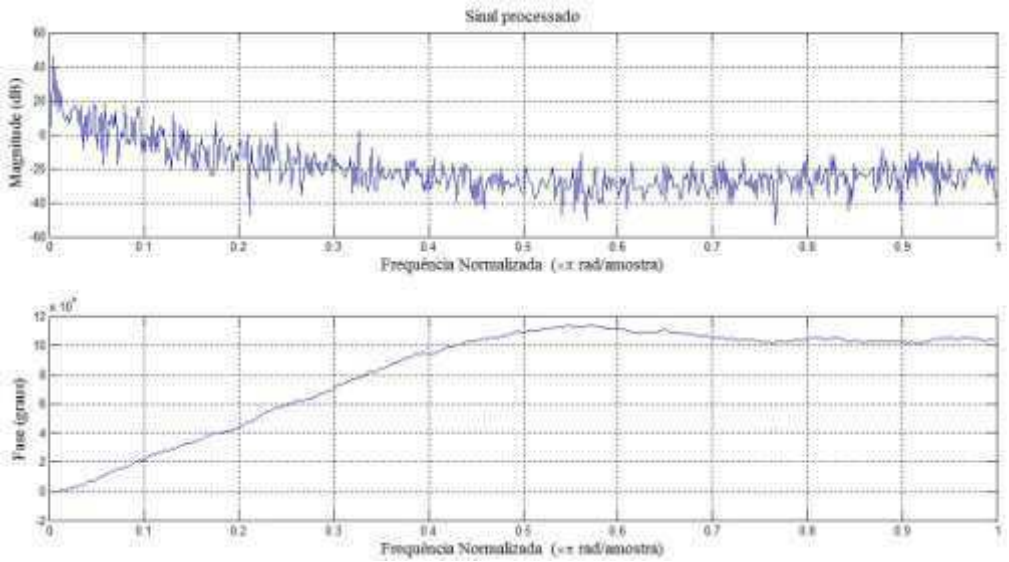

Figura 17: Áudio digital processado via filtro IIR com tempo de delay de 500 ms.

3.1.3 Filtro Universal

É possível combinar as estruturas dos filtros FIR e IIR, apresentadas respectivamente nas Figuras 09 e 14, em um único bloco universal que pode ser configurado para assumir as características desejadas através da manipulação dos seus parâmetros. A Figura 18 a seguir ilustra o caso.

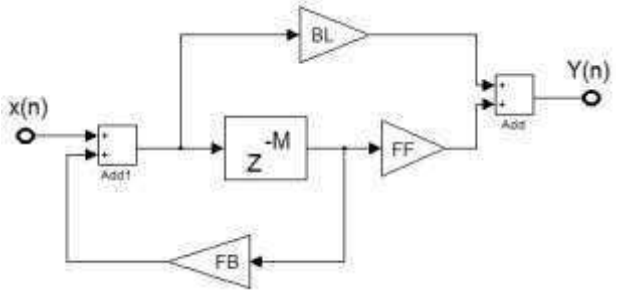

Figura 18: Filtro Digital Universal.

Sendo: BL é o ganho de mesclagem; FB é o ganho de realimentação; e FF é o ganho de alimentação direta. A equação de diferenças e a função de transferência são respectivamente:

$$\mathbf{y(n) = BL \cdot x(n) + FF \cdot x(n-M) + BL \cdot FB \cdot y(n-M),} \quad (8)$$

$$\mathbf{H(z) = (BL + FF \cdot z^{-M}) / (1 - BL \cdot FB \cdot z^{-M}).} \quad (9)$$

Esta estrutura também permite a implementação de um filtro digital passa-tudo. A Tabela 1 seguinte mostra quais valores cada ganho deve assumir para resultar no filtro desejado.

Tabela 1: Configuração do filtro digital universal.

	BL	**FB**	**FF**
Filtro FIR	1	0	Gd
Filtro IIR	1	Gd	0
Filtro passa-tudo	Gd	-Gd	1
***Delay* puro**	0	0	1

Assim sendo, é possível repetir o experimento realizado com os filtros FIR e IIR, utilizando um arquivo de áudio, e obter o efeito de áudio de *delay* também com tempo de atraso de 500 ms e ganho de 0,6. Os algoritmos que utilizam a estrutura do filtro universal

para gerar os filtros de resposta ao impulso finita e infinita estão, respectivamente, nos Anexos 5.5 e 5.6. Os resultados obtidos são idênticos aos expostos nas Figuras 13 e 17.

A estrutura do filtro universal foi, então, usada para implementar os efeitos de *vibrato*, *flanger* e *echo*, derivados diretamente do *delay*, no mesmo arquivo de áudio original da Figura 12. Na conversão do tempo de atraso em quantidade de passos discretos foi utilizado, agora, o método de interpolação linear. Visto que nestes casos é preciso variar o tempo de *delay* em uma frequência baixa, exceto no efeito de *echo*. Os algoritmos de cada um estão nos Anexos 5.7, 5.8 e 5.9 respectivamente. As Figuras 19, 20 e 21, seguintes, trazem os resultados de cada efeito. Nos áudios gerados percebe-se que os efeitos estão com intensidade exagerada, ou seja, os tempos de *delay* estão altos demais. Em especial, no *flanger* surge um ruído agudo não característico do efeito. Indício da necessidade de um filtro passa-baixas na implementação deste.

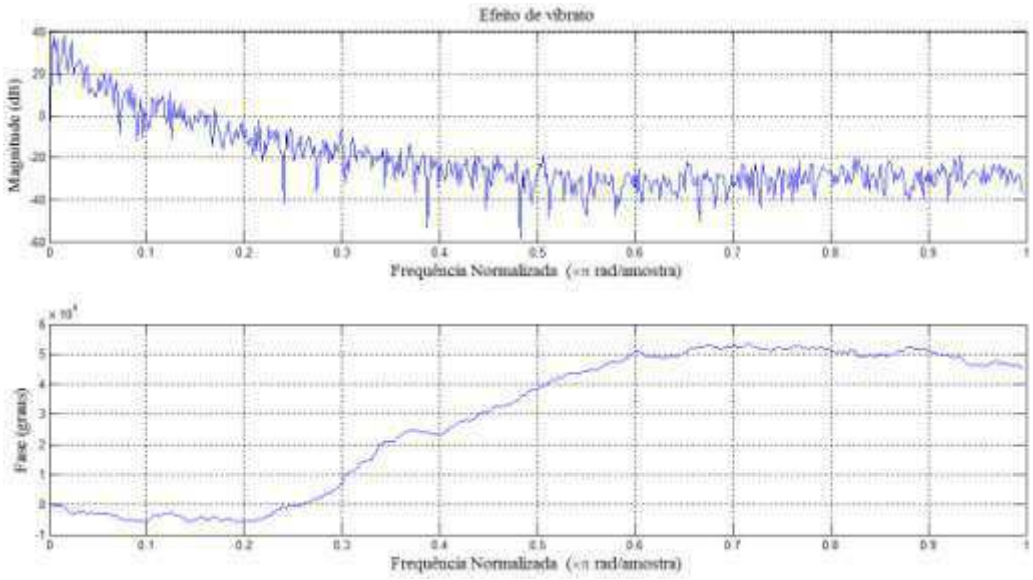

Figura 19: Efeito de *vibrato*, com tempo de *delay* de 6 ms variando em 5 Hz.

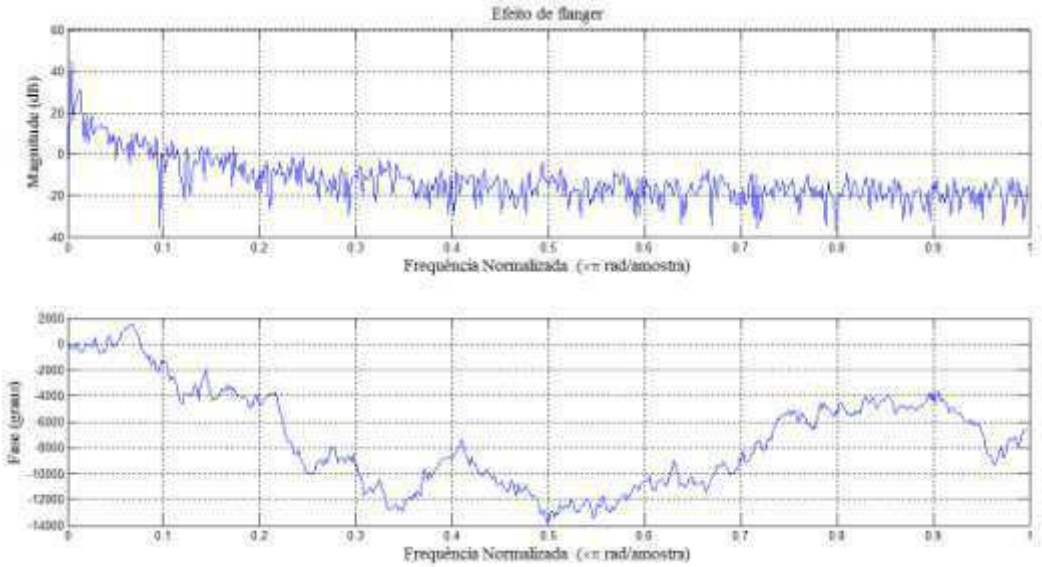

Figura 20: Efeito de *flanger*, com tempo de *delay* de 13,5 ms variando em 0,5 Hz.

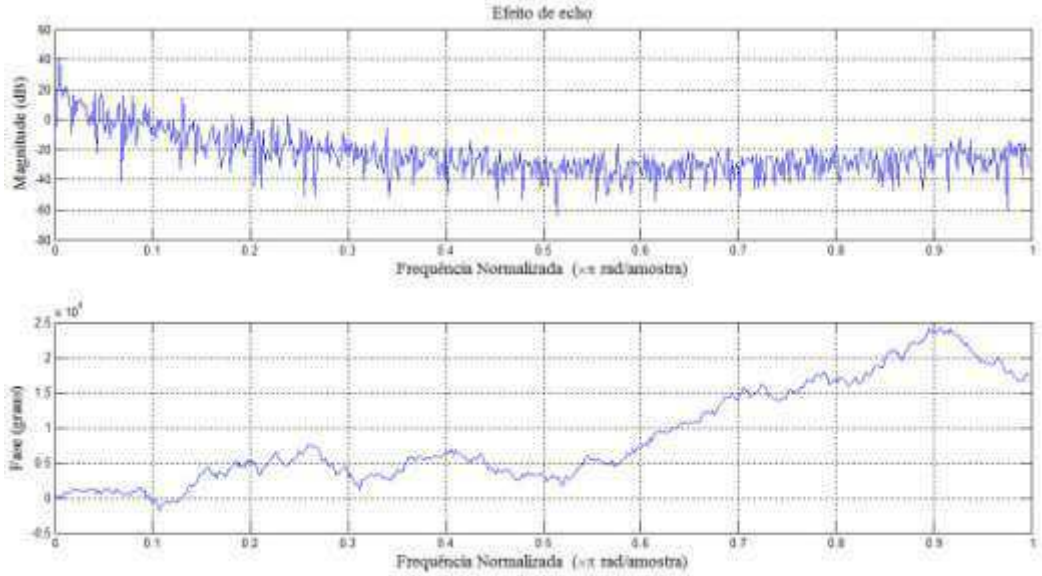

Figura 21: Efeito de *echo*, com tempo de *delay* de 200 ms.

3.2 Efeito de *reverb*

O *reverb* é gerado pela combinação dos filtros FIR, IIR e passa-tudo. Uma das possíveis formas de agrupar estes blocos foi proposta por Manfred R. Schroeder, da *Bell Laboratories*. Futuramente ele propôs uma modificação do seu próprio modelo de implementação artificial do *reverb*. James A. Moorer, da *Institut de Recherche et Coordination Acoustique/Musique*, fez melhorias ao *reverb* de Schroeder. A literatura contem outros inumeros métodos de implementação deste efeito. Estes três citados, por serem os pioneiros e bastante populares, serão utilizados neste trabalho.

Cada filtro utilizado possui seu próprio tempo de *delay*. Cada atraso é determinado pelo perfil acústico do ambiente que deseja-se simular no *reverb* artificial. O ganho de cada filtro é determinado em função do seu tempo de atraso e do tempo de reverberação do ambiente.

3.2.1 Primeiro *reverb* de Schroeder

Neste método, o sinal de entrada passa primeiramente por quatro filtros IIR associados em paralelo. As saídas destes são somadas e passam por dois filtros passa-tudo associados em série[2]. Como ilustra a Figura 22.

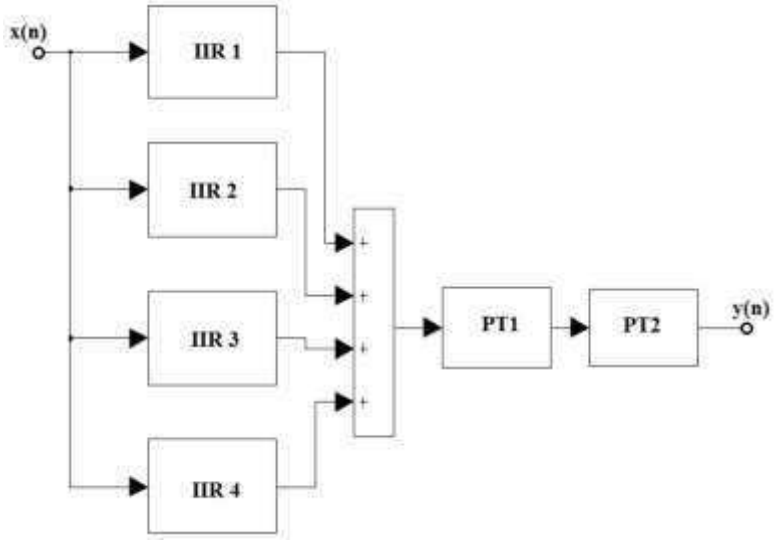

Figura 22: Primeiro *reverb* de Schroeder.

Os tempos de atraso foram estipulados para uma sala de concerto de médio porte[3]. E todos os filtros foram implementados a partir do algoritmo do filtro digital universal. O mesmo arquivo digital de áudio do caso do efeito de *delay* foi utilizado como sinal de entrada para o *reverb*. O algoritmo deste método está no Anexo 5.10. O resultado obtido está na Figura 24 seguinte. O áudio gerado, porém, indica que este método não é eficiente com a parametrização utilizada, pois o timbre de seu som é muito semelhante ao original.

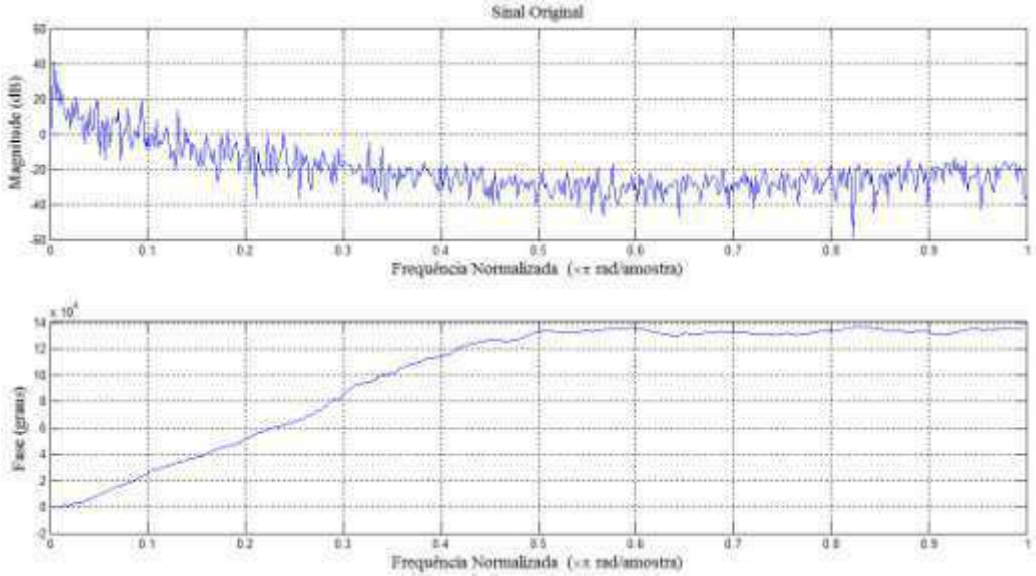

Figura 23: Áudio digital original.

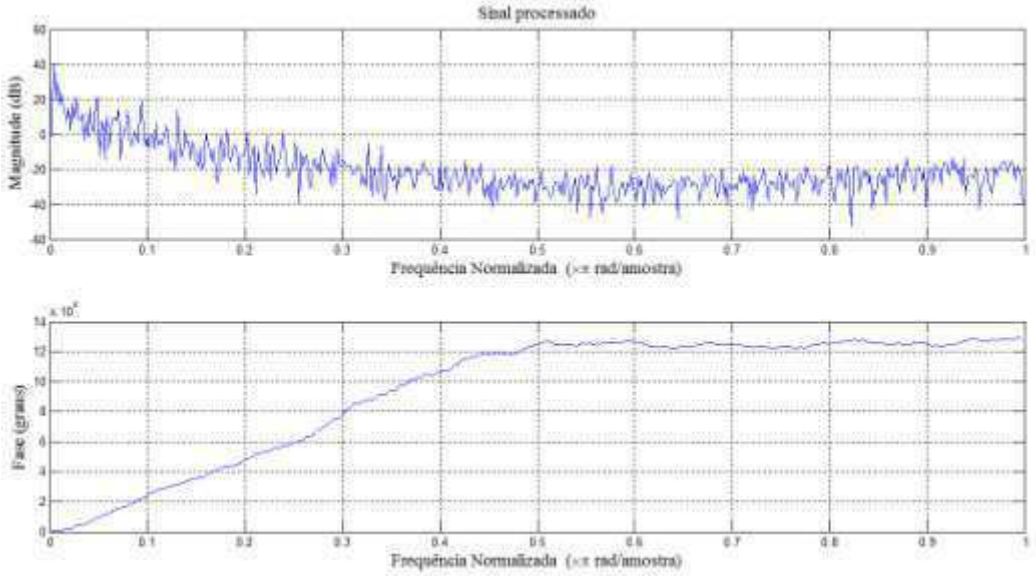

Figura 24: Áudio digital processado pelo *reverb* de Schroeder.

3.2.2 Segundo *reverb* de Schroeder

Este método consiste de uma associação em série de cinco filtros passa-tudo, cada um com seu próprio tempo de *delay*[2]. Novamente, os ganhos de cada filtro é função do seu próprio atraso e do tempo de reverberação.

Figura 25: Segundo *reverb* de Schroeder.

Para fins de simulação, os mesmo tempos de atraso dos filtros em paralelo do modelo anterior foram utilizados. O código escrito está no Anexo 5.11. A aplicação deste *reverb* no mesmo arquivo de áudio dos casos anteriores resultou no exposto na Figura 26 seguinte. O áudio resultante deste método apresentou qualidade bastante satisfatória.

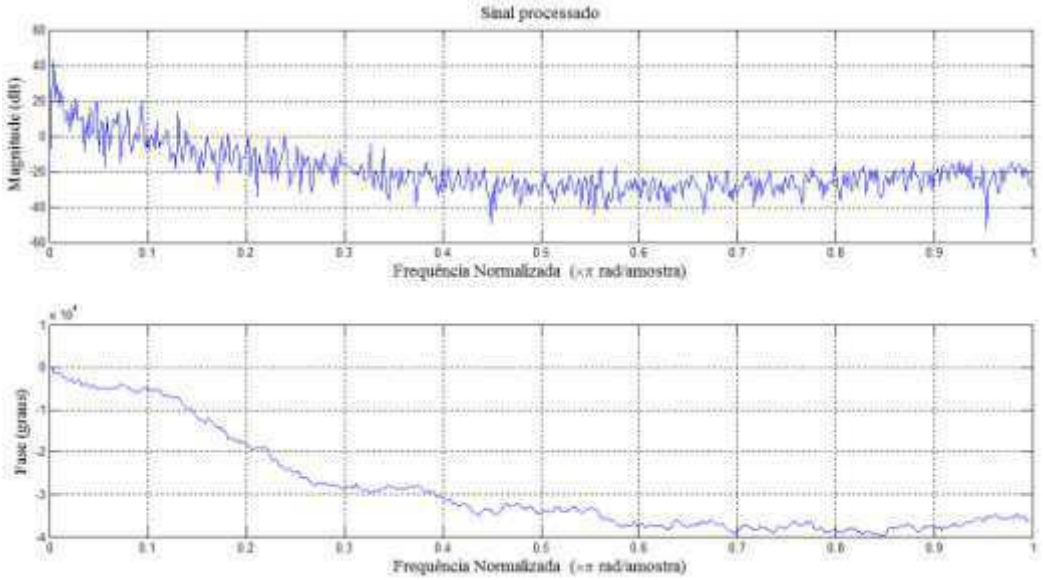

Figura 26: Áudio digital processado pelo *reverb* de Schroeder.

3.2.3 *Reverb* de Moorer

Moorer expandiu a primeira estrutura proposta por Schroeder. No primeiro estágio o sinal de entrada passa agora por uma associação em cadeia de filtros FIR. A saída destes filtros é somada ao próprio sinal de entrada. No estágio seguinte o resultante do anterior é alimentado diretamente para saída e também realimentado para uma associação em paralelo de seis filtros IIR seguidos por um filtro passa-tudo. A saída deste ultimo filtro é alimentada para a saída e somada ao sinal resultante do primeiro estágio[2]. A Figura 27 ilustra este método.

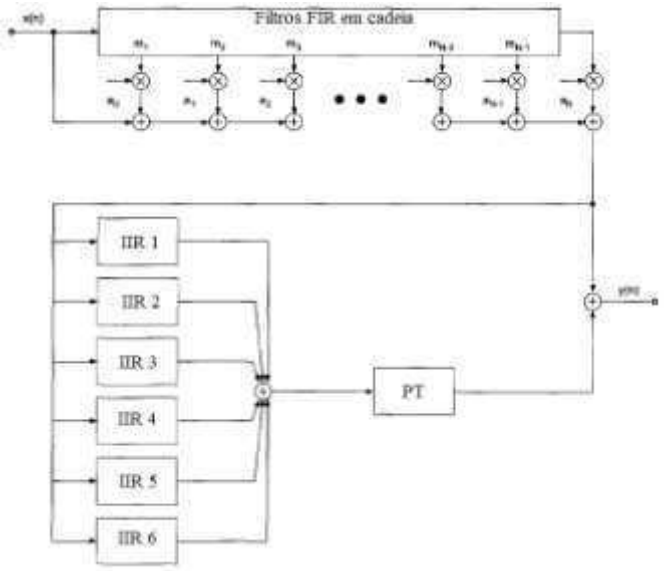

Figura 27: *Reverb* de Moorer.

O algoritmo que realiza este método segue no Anexo 5.12. O sinal resultante deste método encontra-se ilustrado na Figura 28. O arquivo de áudio resultante deste método apresentou resultado semelhante ao primeiro método proposto por Schroeder.

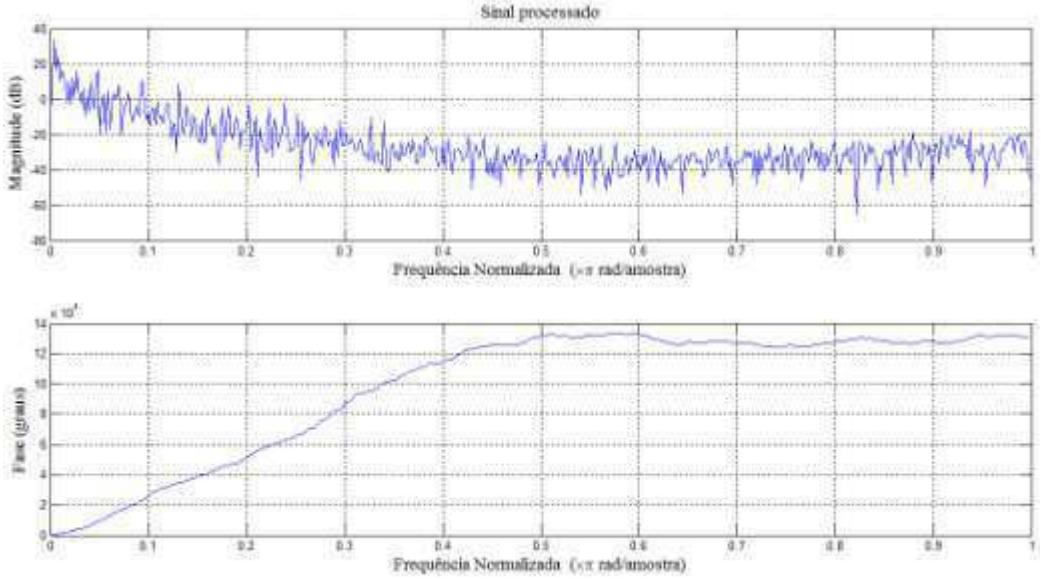

Figura 28: Áudio digital processado pelo *reverb* de Moorer.

4. CONCLUSÃO

Neste trabalho foi apresentado métodos para implementação digital de efeitos de áudio de natureza temporal para uso musical, mais precisamente, efeitos que podem ser implementados através de uma ação de retardo no tempo do sinal de entrada. Os dois efeitos aqui abordados foram o *delay* e o *reverb*. No primeiro, uma ou mais cópias do áudio são atrasadas, atenuadas e somadas ao próprio sinal de entrada para gerar um efeito sonoro desejado. Para implementação digital deste efeito algoritmos de três tipos de filtros digitais foram utilizados, estes são os de resposta ao impulso finita, de resposta ao impulso infinita e passa-tudo. O tanto do atraso – tempo de *delay* – caracteriza a ação resultante. A exemplos, temos o *vibrato*, o *flanger* e o *echo*. O *reverb*, por sua vez, é a repetição contínua do sinal de origem, onde cada cópia sofre atenuação proporcional ao seu tempo de atraso. Este fenômeno surge da reflexão das ondas sonoras nas paredes de um ambiente fechado. A implementação do *reverb* foi realizada via três métodos propostos por Manfred Schroeder e James Moorer.

Os arquivos digitais de áudios gerados por cada simulação demonstraram a capacidade dos métodos aqui utilizados. Pôde-se perceber, também, que os parâmetros dos efeitos não podem ser escolhidos ao acaso. Pois o resultado sonoro pode não ser agradável à audição, ou simplesmente não condizer com o modelo físico dos efeitos. Uma vez que os efeitos aqui estudados ocorrem normalmente na natureza. A exemplo, dos resultados obtidos para os efeitos *flanger*, 1º *reverb* de Schroeder e o *reverb* de Moorer. Por fim, a realização desta atividade permite enfatizar a importância dos conteúdos oferecidos nas disciplinas do curso de Engenharia Elétrica da UFCG, desde aquelas classificadas como sendo de conhecimentos básicos até as de ênfase que expõe diretamente as teorias usadas neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BALLOU, G. (editor). Handbook for Sound Engineers. 4ª edição, 2008. Focal Press.

[2] DORNEAN, I.; TOPA, M.; KIREI, B. S. Digital Implementation of Artificial Reverberation Algorithms. ACTA TECHNICA NAPOCENSIS, Vol 49, Número 4, 2008.

[3] LIU, L. Case Study: Schroeder Reverberator. 2012. Departament of Computer Science, ETH Zürich. Reconfigurable Computing Systems (252-2210-00L), Outono 2012.

[4] Zölzer, U (Editor). *DAFX, Digital Audio Effects*. 1ª Edição, 2002. Josn Wiley & Sons.

5. ANEXOS

5.1 Filtro digital de resposta finita ao impulso (FIR)

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para Filtro Digital de Resposta Finita ao Impulso

close all;
clear all;
clc;

% ============================ SINAL ORIGINAL ============================

x=zeros(100,1);
x(1)=1;

% ==================== DELAY USANDO O FIR COMB FILTER ====================
M = 10;
Gd=0.6;

Delayline=zeros(M,1);  %memory allocation for lenght 10

for n=1:length(x);
    y(n)=x(n)+Gd*Delayline(M);
    Delayline=[x(n);Delayline(1:M-1)];
end;

figure
stem (0:length(y)-1,y)
grid
figure
freqz(y,1)
```

5.2 Uso do filtro FIR em um arquivo digital de áudio

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
```

```
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para Filtro Digital de Resposta Finita ao Impulso

close all
clear all;
clc;

% ======================= ARQUIVO DE ÁUDIO ORIGINAL =======================

[x, Fs] = wavread('audio');


% ==================== DELAY USANDO O FIR COMB FILTER ====================

Td = 500e-3;        % Tempo de delay em segundos

M = round(Td*Fs);   % Delay em quantidade de passos

Gd = 0.6;           % Ganho do delay


Delayline = zeros(M,1);  % Alocação de memória para o delay
Y = zeros(1,length(x));  % Alocação de memória para a saída

for n = 1:length(x);
    y(n) = x(n) + Gd*Delayline(M);
    Delayline = [x(n); Delayline(1:M-1)];
end;


% ====================== ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y,Fs,'audio_FIR_delayed');

freqz(x,1)
title('Sinal Original')
figure
freqz(y,1)
title('Sinal processado')
```

5.3 Filtro digital de resposta infinita ao impulso (IIR)

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
```

```
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para Filtro Digital de Resposta Infinita ao Impulso

close all;
clear all;
clc;

% ============================ SINAL ORIGINAL ============================

x=zeros(100,1);
x(1)=1;

% ==================== DELAY USANDO O IIR COMB FILTER ====================
M = 10;
Gd=0.6;

Delayline=zeros(M,1);  %memory allocation for lenght 10

for n=1:length(x);
    y(n)=x(n)+Gd*Delayline(M);
    Delayline=[y(n);Delayline(1:M-1)];
end;

figure
stem (0:length(y)-1,y)
figure
freqz(y,1)
```

5.4 Uso do filtro IIR em um arquivo digital de áudio

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
```

```
%
% Algoritmo para Filtro Digital de Resposta Infinita ao Impulso


close all;
clear all;
clc;

% ======================= ARQUIVO DE ÁUDIO ORIGINAL =======================

[x, Fs] = wavread('audio');


% ==================== DELAY USANDO O IIR COMB FILTER ====================

Td = 500e-3;        % Tempo de delay em segundos

M = round(Td*Fs);   % Delay em quantidade de passos

Gd = 0.6;           % Ganho do delay


Delayline = zeros(M,1);  % Alocação de memória para o delay
Y = zeros(1,length(x));  % Alocação de memória para a saída

for n = 1:length(x);
    y(n) = x(n) + Gd*Delayline(M);
    Delayline = [y(n); Delayline(1:M-1)];
end;

% ======================= ARQUIVO DE ÁUDIO PROCESSADO =======================

wavwrite(y,Fs,'audio_IIR_delayed');

freqz(x,1)
title('Sinal Original')
figure
freqz(y,1)
title('Sinal processado')
```

5.5 Filtro FIR via filtro digital universal

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
```

```
%
%
% Algoritmo para Filtro Digital de Resposta Finita ao Impulso
% via o filtro digital universal


close all;
clear all;
clc;

% ======================= ARQUIVO DE ÁUDIO ORIGINAL ======================

[x, Fs]=wavread('audio');


% ========================== PARÂMETROS DO DELAY ==========================

Td = 500e-3;      % Tempo de delay em segundos

M=round(Td*Fs);   % Delay em quantidade de passos

Gd = 0.6;         % Ganho do delay


% ============================ FILTRO UNIVERSAL ============================

% Configurando o filtro para atuar como um FIR
BL=1;    % Ganho de mesclagem
FB=0;    % Ganho de realimentação
FF=Gd;   % Ganho de alimentação direta

Delayline=zeros(M,1);  % Alocação de memória para o delay
y=zeros(1,length(x));  % Alocação de memória para a saída

for n=1:length(x);
    xh=x(n)+FB*Delayline(M);
    y(n)=FF*Delayline(M)+BL*xh;
    Delayline=[xh;Delayline(1:M-1)];
end;


% ======================= ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y,Fs,'audio_uni_FIR_delayed');

freqz(x,1)
title('Sinal Original')
figure
freqz(y,1)
title('Sinal processado')
```

5.6 Filtro IIR via filtro digital universal

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
```

```
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para Filtro Digital de Resposta Infinita ao Impulso
% via o filtro digital universal


close all;
clear all;
clc;

% ======================= ARQUIVO DE ÁUDIO ORIGINAL =======================

[x, Fs]=wavread('audio');


% ========================== PARÂMETROS DO DELAY ==========================

Td = 500e-3;      % Tempo de delay em segundos

M=round(Td*Fs);   % Delay em quantidade de passos

Gd = 0.6;         % Ganho do delay

% ============================ FILTRO UNIVERSAL ============================

% Configurando o filtro para atuar como um IIR
BL=1;    % Ganho de mesclagem
FB=Gd;   % Ganho de realimentação
FF=0;    % Ganho de alimentação direta


Delayline=zeros(M,1);  % Alocação de memória para o delay
y=zeros(1,length(x));  % Alocação de memória para a saída

for n=1:length(x);
    xh=x(n)+FB*Delayline(M);
    y(n)=FF*Delayline(M)+BL*xh;
    Delayline=[xh;Delayline(1:M-1)];
end;


% ====================== ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y,Fs,'audio_uni_FIR_delayed');

freqz(x,1)
```

```
title('Sinal Original')
figure
freqz(y,1)
title('Sinal processado')
```

5.7 Efeito vibrato.

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para efeito VIBRATO


close all;
clear all;
clc;

% ======================= ARQUIVO DE ÁUDIO ORIGINAL =======================

[x, Fs]=wavread('audio');


% ========================== PARÂMETROS DO FILTRO ==========================

Td = 6e-3;          % Tempo de delay em segundos

Mi = floor(Td*Fs);  % Delay em quantidade de passos
frac = (Td*Fs)-Mi;

Gd = 0.6;           % Ganho do delay

Modfreq = 5;        % Variação do delay em Hz
MODFREQ = Modfreq/Fs;



% ============================ FILTRO UNIVERSAL ============================

% Configurando o filtro para atuar como um passa-tudo
BL=0;   % Ganho de mesclagem
FB=0;   % Ganho de realimentação
FF=1;   % Ganho de alimentação direta
```

```
LEN=length(x);
L=2+Mi+Mi*2;
Delayline=zeros(L,1);  %memory allocation
y=zeros(size(x));

for n=1:(LEN-1);
    M=MODFREQ;
    MOD=sin(M*2*pi*n) ;
    ZEIGER=1+Mi+Mi*MOD;
    i=floor(ZEIGER);
    frac=ZEIGER-i;

    xh=x(n)+FB*Delayline(L);

    y(n)=FF*Delayline(i+1)*frac+Delayline(i)*(1-frac)+BL*xh;

    Delayline=[xh;Delayline(1:L-1)];
end;


% ====================== ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y,Fs,'audio_vibrato');

freqz(x,1)
title('Sinal Original')
figure
freqz(y,1)
title('Efeito de vibrato')
```

5.8 Efeito flanger.

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para efeito FLANGER


close all;
clear all;
clc;
```

```
% ====================== ARQUIVO DE ÁUDIO ORIGINAL ======================

[x, Fs]=wavread('audio');


% ========================== PARÂMETROS DO FILTRO ==========================

Td = 13.5e-3;       % Tempo de delay em segundos

Mi = floor(Td*Fs);  % Delay em quantidade de passos
frac = (Td*Fs)-Mi;

Gd = 0.6;           % Ganho do delay

Modfreq = 0.5;      % Variação do delay em Hz
MODFREQ = Modfreq/Fs;



% ============================ FILTRO UNIVERSAL ============================

% Configurando o filtro para atuar como um passa-tudo
BL=1;   % Ganho de mesclagem
FB=Gd;  % Ganho de realimentação
FF=0;   % Ganho de alimentação direta


LEN=length(x);
L=2+Mi+Mi*2;
Delayline=zeros(L,1);  %memory allocation
y=zeros(size(x));

for n=1:(LEN-1);
    M=MODFREQ;
    MOD=sin(M*2*pi*n) ;
    ZEIGER=1+Mi+Mi*MOD;
    i=floor(ZEIGER);
    frac=ZEIGER-i;

    xh=x(n)+FB*Delayline(L);

    y(n)=FF*Delayline(i+1)*frac+Delayline(i)*(1-frac)+BL*xh;

    Delayline=[xh;Delayline(1:L-1)];
end;

y = y/2.5;

% ====================== ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y,Fs,'audio_flanger');

freqz(x,1)
title('Sinal Original')
figure
freqz(y,1)
title('Efeito de flanger')
```

5.9 Efeito echo.

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para efeito ECHO


close all;
clear all;
clc;

% ======================= ARQUIVO DE ÁUDIO ORIGINAL ======================

[x, Fs]=wavread('audio');


% ========================== PARÂMETROS DO FILTRO ==========================

Td = 200e-3;         % Tempo de delay em segundos

Mi = floor(Td*Fs);  % Delay em quantidade de passos
frac = (Td*Fs)-Mi;

Gd = 0.6;            % Ganho do delay

Modfreq = 0;         % Variação do delay em Hz
MODFREQ = Modfreq/Fs;



% ============================ FILTRO UNIVERSAL ============================

% Configurando o filtro para atuar como um passa-tudo
BL=1;   % Ganho de mesclagem
FB=0;   % Ganho de realimentação
FF=Gd;  % Ganho de alimentação direta


LEN=length(x);
L=2+Mi+Mi*2;
Delayline=zeros(L,1);  %memory allocation
y=zeros(size(x));
```

```
for n=1:(LEN-1);
    M=MODFREQ;
    MOD=sin(M*2*pi*n) ;
    ZEIGER=1+Mi+Mi*MOD;
    i=floor(ZEIGER);
    frac=ZEIGER-i;

    xh=x(n)+FB*Delayline(L);

    y(n)=FF*Delayline(i+1)*frac+Delayline(i)*(1-frac)+BL*xh;

    Delayline=[xh;Delayline(1:L-1)];
end;

y=y/1.5;

% ====================== ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y,Fs,'audio_echo');

freqz(x,1)
title('Sinal Original')
figure
freqz(y,1)
title('Efeito de echo')
```

5.10 Reverb de Schroeder – Método 1º.

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para efeito REVERB
% Modelo I proposto por Schroeder


close all;
clear all;
clc;

% ====================== ARQUIVO DE ÁUDIO ORIGINAL ======================

[x, Fs]=wavread('audio');
```

```
% ========================= PARÂMETROS DOS DELAYS ========================

% Tempo de reverberação
% Menor que 0.2s -> muito curto
% Maior que 0.5s -> muito longo

T60 = 0.2;

% Tempos de delay dos filtros em segundos

TdC1 = 29.7e-3;
TdC2 = 37.1e-3;
TdC3 = 41.1e-3;
TdC4 = 43.7e-3;

TdA1 = 5e-3;
TdA2 = 1.7e-3;

% Ganhos dos filtros

GdC1 = 0.001^(TdC1/T60);
GdC2 = 0.001^(TdC2/T60);
GdC3 = 0.001^(TdC3/T60);
GdC4 = 0.001^(TdC4/T60);

GdA1 = 0.001^(TdA1/T60);
GdA2 = 0.001^(TdA2/T60);


% ========================= PARÂMETROS DOS FILTROS ========================

BLC1 = 1; FBC1 = GdC1; FFC1 = 0;
BLC2 = 1; FBC2 = GdC2; FFC2 = 0;
BLC3 = 1; FBC3 = GdC3; FFC3 = 0;
BLC4 = 1; FBC4 = GdC4; FFC4 = 0;

BLA1 = GdA1; FBA1 = -GdA1; FFA1 = 1;
BLA2 = GdA2; FBA2 = -GdA2; FFA2 = 1;


% ========================== FILTROS IIR PARALELO =========================

% Alocação de memórias
MC1 = round(TdC1*Fs); MC2 = round(TdC2*Fs);
MC3 = round(TdC3*Fs); MC4 = round(TdC4*Fs);

DelaylineC1 = zeros(MC1,1); DelaylineC2 = zeros(MC2,1);
DelaylineC3 = zeros(MC3,1); DelaylineC4 = zeros(MC4,1);
z1 = zeros(1,length(x)); z2 = zeros(1,length(x));
z3 = zeros(1,length(x)); z4 = zeros(1,length(x));


for n=1:length(x);
    % IIR 1
    xh1 = x(n) + FBC1*DelaylineC1(MC1);
    z1(n) = FFC1*DelaylineC1(MC1) + BLC1*xh1;
    DelaylineC1 = [xh1; DelaylineC1(1:MC1-1)];
```

```
    % IIR 2
    xh2 = x(n) + FBC2*DelaylineC2(MC2);
    z2(n) = FFC2*DelaylineC2(MC2) + BLC2*xh2;
    DelaylineC2 = [xh2; DelaylineC2(1:MC2-1)];
    % IIR 3
    xh3 = x(n) + FBC3*DelaylineC3(MC3);
    z3(n) = FFC3*DelaylineC3(MC3) + BLC3*xh3;
    DelaylineC3 = [xh3; DelaylineC3(1:MC3-1)];
    % IIR 4
    xh4 = x(n) + FBC4*DelaylineC4(MC4);
    z4(n) = FFC4*DelaylineC4(MC4) + BLC4*xh4;
    DelaylineC4 = [xh4; DelaylineC4(1:MC4-1)];
end;

% Soma dos sinais de saída de cada filtro
z = (z1 + z2 + z3 + z4)/4;


% ======================== FILTROS PASSA-TUDO SÉRIE ======================

% Alocação de memórias
MA1 = round(TdA1*Fs); MA2 = round(TdA2*Fs);

DelaylineA1 = zeros(MA1,1); DelaylineA2 = zeros(MA2,1);

y1 = zeros(1,length(z)); y2 = zeros(1,length(z));


for n=1:length(z);
    zh=z(n)+FBA1*DelaylineA1(MA1);
    y1(n)=FFA1*DelaylineA1(MA1)+BLA1*zh;
    DelaylineA1=[zh;DelaylineA1(1:MA1-1)];
end;
for n=1:length(y1);
    yh=y1(n)+FBA2*DelaylineA2(MA2);
    y2(n)=FFA2*DelaylineA2(MA2)+BLA2*yh;
    DelaylineA2=[yh;DelaylineA2(1:MA2-1)];
end;


% ====================== ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y2,Fs,'audio_schroeder1');

freqz(x,1)
title('Sinal Original')
figure
freqz(y2,1)
title('Sinal processado')
```

5.11 Reverb de Schroeder – Método 2º.

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
% Unidade Acadêmica de Engenharia Elétrica
%
%
```

```
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para efeito REVERB
% Modelo II proposto por Schroeder


close all;
clear all;
clc;

% ======================= ARQUIVO DE ÁUDIO ORIGINAL =======================

[x, Fs]=wavread('audio');


% ========================= PARÂMETROS DOS DELAYS =========================

% Tempo de reverberação
% Menor que 0.2s -> muito curto
% Maior que 0.5s -> muito longo

T60 = 0.3;

% Tempos de delay dos filtros em segundos

TdA1 = 29.7e-3;
TdA2 = 37.1e-3;
TdA3 = 41.1e-3;
TdA4 = 43.7e-3;
TdA5 = 5e-3;

% Ganhos dos filtros

GdA1 = 0.001^(TdA1/T60);
GdA2 = 0.001^(TdA2/T60);
GdA3 = 0.001^(TdA3/T60);
GdA4 = 0.001^(TdA4/T60);
GdA5 = 0.001^(TdA5/T60);


% ======================== PARÂMETROS DOS FILTROS =========================

BLA1 = GdA1; FBA1 = -GdA1; FFA1 = 1;
BLA2 = GdA2; FBA2 = -GdA2; FFA2 = 1;
BLA3 = GdA3; FBA3 = -GdA3; FFA3 = 1;
BLA4 = GdA4; FBA4 = -GdA4; FFA4 = 1;
BLA5 = GdA5; FBA5 = -GdA5; FFA5 = 1;
```

```
% ======================== FILTROS PASSA-TUDO SÉRIE ========================

% Alocação de memórias
MA1 = round(TdA1*Fs); MA2 = round(TdA2*Fs);
MA3 = round(TdA3*Fs); MA4 = round(TdA4*Fs);
MA5 = round(TdA5*Fs);

DelaylineA1 = zeros(MA1,1); DelaylineA2 = zeros(MA2,1);
DelaylineA3 = zeros(MA3,1); DelaylineA4 = zeros(MA4,1);
DelaylineA5 = zeros(MA5,1);

y1 = zeros(1,length(x)); y2 = zeros(1,length(y1));
y3 = zeros(1,length(y2)); y4 = zeros(1,length(y3));
y5 = zeros(1,length(y4));


for n=1:length(x);
    xh=x(n)+FBA1*DelaylineA1(MA1);
    y1(n)=FFA1*DelaylineA1(MA1)+BLA1*xh;
    DelaylineA1=[xh;DelaylineA1(1:MA1-1)];
end;
for n=1:length(y1);
    yh=y1(n)+FBA2*DelaylineA2(MA2);
    y2(n)=FFA2*DelaylineA2(MA2)+BLA2*yh;
    DelaylineA2=[yh;DelaylineA2(1:MA2-1)];
end;
for n=1:length(y2);
    yh=y2(n)+FBA2*DelaylineA2(MA2);
    y3(n)=FFA2*DelaylineA2(MA2)+BLA2*yh;
    DelaylineA2=[yh;DelaylineA2(1:MA2-1)];
end;
for n=1:length(y3);
    yh=y3(n)+FBA2*DelaylineA2(MA2);
    y4(n)=FFA2*DelaylineA2(MA2)+BLA2*yh;
    DelaylineA2=[yh;DelaylineA2(1:MA2-1)];
end;
for n=1:length(y4);
    yh=y4(n)+FBA2*DelaylineA2(MA2);
    y5(n)=FFA2*DelaylineA2(MA2)+BLA2*yh;
    DelaylineA2=[yh;DelaylineA2(1:MA2-1)];
end;


% ====================== ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y5,Fs,'audio_schroeder2');

freqz(x,1)
title('Sinal Original')
figure
freqz(y5,1)
title('Sinal processado')
```

5.12 Reverb de Moorer.

```
% Universidade Federal de Campina Grande
% Centro de Engenharia Elétrica e Informática
```

```
% Unidade Acadêmica de Engenharia Elétrica
%
%
% Trabalho de Conclusão de Curso
%
% Período: 2015.1
%
% Implementação Digital de Efeitos de Áudio: Delay & Reverb
%
% Aluno: Pedro Merencio Primo Passos
% Matrícula: 112150175
%
% Orientador: Prof. Edmar Candeia Gurjão, D. Sc.
%
%
% Algoritmo para efeito REVERB
% Modelo proposto por Moorer


close all;
clear all;
clc;

% ======================== ARQUIVO DE ÁUDIO ORIGINAL ========================

[x, Fs]=wavread('audio');


% ========================== PARÂMETROS DOS DELAYS ==========================

% Tempo de reverberação
% Menor que 0.2s -> muito curto
% Maior que 0.5s -> muito longo

T60 = 0.3;

% Tempos de delay dos filtros em segundos

TdB1 = 29.7e-3;
TdB2 = 37.1e-3;

TdC1 = 50.0e-3;
TdC2 = 53.3e-3;
TdC3 = 64.1e-3;
TdC4 = 75.0e-3;
TdC5 = 77.5e-3;
TdC6 = 80.0e-3;

TdA1 = 500e-3;
TdA2 = 85e-3;

% Ganhos dos filtros

GdB1 = 0.001^(TdB1/T60);
GdB2 = 0.001^(TdB2/T60);

GdC1 = 0.001^(TdC1/T60);
GdC2 = 0.001^(TdC2/T60);
```

```
GdC3 = 0.001^(TdC3/T60);
GdC4 = 0.001^(TdC4/T60);
GdC5 = 0.001^(TdC5/T60);
GdC6 = 0.001^(TdC6/T60);

%GdA1 = 0.001^(TdA1/T60);
GdA1 = 0.7;


% ========================= PARÂMETROS DOS FILTROS ========================

BLB1 = 1; FBB1 = 0; FFB1 = GdB1;
BLB2 = 1; FBB2 = 0; FFB2 = GdB2;

BLC1 = 1; FBC1 = GdC1; FFC1 = 0;
BLC2 = 1; FBC2 = GdC2; FFC2 = 0;
BLC3 = 1; FBC3 = GdC3; FFC3 = 0;
BLC4 = 1; FBC4 = GdC4; FFC4 = 0;

BLA1 = GdA1; FBA1 = -GdA1; FFA1 = 1;


% ============================ FILTROS FIR ENTRADA ========================

% Alocação de memórias
MB1 = round(TdB1*Fs);
MB2 = round(TdB2*Fs);

DelaylineB1 = zeros(MB1,1);
DelaylineB2 = zeros(MB2,1);

p1 = zeros(1,length(x));
p2 = zeros(1,length(x));


for n=1:length(x);
    % FIR 1
    ph1 = x(n) + FBB1*DelaylineB1(MB1);
    p1(n) = FFB1*DelaylineB1(MB1) + BLB1*ph1;
    DelaylineB1 = [ph1; DelaylineB1(1:MB1-1)];
    % FIR 2
    ph2 = x(n) + FBB2*DelaylineB2(MB2);
    p2(n) = FFB2*DelaylineB2(MB2) + BLB2*ph2;
    DelaylineB2 = [ph2; DelaylineB2(1:MB2-1)];
end;

p = (p1+p2)/2;
% ========================= FILTROS IIR PARALELO ========================

% Alocação de memórias
MC1 = round(TdC1*Fs); MC2 = round(TdC2*Fs);
MC3 = round(TdC3*Fs); MC4 = round(TdC4*Fs);

DelaylineC1 = zeros(MC1,1); DelaylineC2 = zeros(MC2,1);
DelaylineC3 = zeros(MC3,1); DelaylineC4 = zeros(MC4,1);

z1 = zeros(1,length(p)); z2 = zeros(1,length(p));
z3 = zeros(1,length(p)); z4 = zeros(1,length(p));
```

```
for n=1:length(p);
    % IIR 1
    xh1 = p(n) + FBC1*DelaylineC1(MC1);
    z1(n) = FFC1*DelaylineC1(MC1) + BLC1*xh1;
    DelaylineC1 = [xh1; DelaylineC1(1:MC1-1)];
    % IIR 2
    xh2 = p(n) + FBC2*DelaylineC2(MC2);
    z2(n) = FFC2*DelaylineC2(MC2) + BLC2*xh2;
    DelaylineC2 = [xh2; DelaylineC2(1:MC2-1)];
    % IIR 3
    xh3 = p(n) + FBC3*DelaylineC3(MC3);
    z3(n) = FFC3*DelaylineC3(MC3) + BLC3*xh3;
    DelaylineC3 = [xh3; DelaylineC3(1:MC3-1)];
    % IIR 4
    xh4 = p(n) + FBC4*DelaylineC4(MC4);
    z4(n) = FFC4*DelaylineC4(MC4) + BLC4*xh4;
    DelaylineC4 = [xh4; DelaylineC4(1:MC4-1)];
end;

z = (z1 + z2 + z3 + z4)/4;


% ======================== FILTROS PASSA-TUDO SÉRIE ======================

% Alocação de memórias
MA1 = round(TdA1*Fs);
MA2 = round(TdA2*Fs);

DelaylineA1 = zeros(MA1,1);
DelaylineA2 = zeros(MA2,1);

y1 = zeros(1,length(z));
y2 = zeros(1,length(y1));



for n=1:length(z);
    zh=z(n)+FBA1*DelaylineA1(MA1);
    y1(n)=FFA1*DelaylineA1(MA1)+BLA1*zh;
    DelaylineA1=[zh;DelaylineA1(1:MA1-1)];
end;
for n=1:length(y1);
    yh=y1(n);
    y1(n)=DelaylineA2(MA2);
    DelaylineA2=[yh;DelaylineA2(1:MA2-1)];
end;

y = (p + y2)/2;

% ======================== ARQUIVO DE ÁUDIO PROCESSADO ======================

wavwrite(y,Fs,'audio_moorer1');

freqz(x,1)
title('Sinal Original')
figure
freqz(y,1)
title('Sinal processado')
```