UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Vitor Probst Curtarelli

CARACTERIZAÇÃO DE SISTEMAS NÃO-LINEARES COM DISTORÇÃO HARMÔNICA UTILIZANDO A RESPOSTA À VARREDURA EXPONENCIAL

Florianópolis
2021

Vitor Probst Curtarelli

CARACTERIZAÇÃO DE SISTEMAS NÃO-LINEARES COM DISTORÇÃO HARMÔNICA UTILIZANDO A RESPOSTA À VARREDURA EXPONENCIAL

Trabalho de Conclusão de Curso (TCC) submetido ao Curso de Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Engenharia Elétrica.
Orientador: Prof. Arcanjo Lenzi, PhD.
Coorientador: Ricardo Brum, Me. Eng.

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Curtarelli, Vitor Probst
Caracterização de sistemas não-lineares com distorção harmônica utilizando a resposta à varredura exponencial / Vitor Probst Curtarelli ; orientador, Arcanjo Lenzi, coorientador, Ricardo Brum, 2021.
95 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Elétrica, Florianópolis, 2021.

Inclui referências.

1. Engenharia Elétrica. 2. Sistemas não-lineares. 3. Modelagem. 4. Distorção harmônica. 5. Varredura exponencial. I. Lenzi, Arcanjo. II. Brum, Ricardo. III. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. IV. Título.

Vitor Probst Curtarelli

CARACTERIZAÇÃO DE SISTEMAS NÃO-LINEARES COM DISTORÇÃO HARMÔNICA UTILIZANDO A RESPOSTA À VARREDURA EXPONENCIAL

Este Trabalho de Conclusão de Curso (TCC) foi julgado adequado para obtenção do Título de Bacharel em Engenharia Elétrica e aprovado, em sua forma final, pela Banca Examinadora.

Florianópolis, 24 de setembro de 2021

Documento assinado digitalmente
Jean Vianei Leite
Data: 30/09/2021 14:15:29-0300
CPF: 003.474.909-80
Verifique as assinaturas em https://v.ufsc.br

Prof. Jean Vian Leite, Dr.
Coordenador do Curso de Graduação em
Engenharia Elétrica

Banca Examinadora:

Documento assinado digitalmente
Arcanjo Lenzi
Data: 30/09/2021 17:31:09-0300
CPF: 299.997.669-00
Verifique as assinaturas em https://v.ufsc.br

Prof. Arcanjo Lenzi, PhD.
Orientador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Ricardo Brum
Data: 24/09/2021 17:53:57-0300
CPF: 010.759.680-62
Verifique as assinaturas em https://v.ufsc.br

Ricardo Brum, Me. Eng.
Coorientador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Marcio Holsbach Costa
Data: 25/09/2021 09:12:20-0300
CPF: 572.170.680-53
Verifique as assinaturas em https://v.ufsc.br

Prof. Márcio Holsbach Costa, Dr.
Avaliador
Universidade Federal de Santa Catarina

Documento assinado digitalmente

Stephan Paul
Data: 30/09/2021 13:56:27-0300
CPF: 010.779.519-14
Verifique as assinaturas em https://v.ufsc.br

Prof. Stephan Paul, Dr.
Avaliador
Universidade Federal de Santa Catarina

Documento assinado digitalmente

Erasmo Felipe Vergara Miranda
Data: 24/09/2021 17:48:41-0300
CPF: 005.003.839-79
Verifique as assinaturas em https://v.ufsc.br

Prof. Erasmo Felipe Vergara Miranda, Dr.
Avaliador
Universidade Federal de Santa Catarina

À Claudia, Edenilson, Fernanda e Lucas, por sempre estarem comigo.

AGRADECIMENTOS

Ao professor Arcanjo Lenzi, o Chefinho, por me aceitar como seu orientando e por me ajudar neste trajeto.

Ao meu colega de laboratório e amigo, Ricardo Brum. Sei que não foi fácil me ajudar e me direcionar, sempre te pedindo conselhos e dicas. Este trabalho é tão seu quanto meu, e eu tenho muito a te agradecer por ele. Da escolha do tema à escrita da última linha, você me ajudou.

Aos amigos de curso e de vivência, Hermano, Henrique e Pedro. Vocês não sabem quanto desse trabalho é graças a vocês, de conselhos de escrita a explicar o que é um pedal. Vocês são como uma família para mim, e espero poder ser parte tão importante da vida de vocês quanto vocês são da minha.

Aos meus professores Bartolomeu, Leonardo Resende e Márcio Costa, que me abriram as portas na graduação para a área de processamento de sinais e me deram as ferramentas e chances para chegar onde estou. Este trabalho usa e abusa do conhecimento que vocês me passaram na disciplina de Sistemas Lineares, e se não fosse por ela e pela monitoria que me ofertaram depois, provavelmente este trabalho não sairia do chão. Sem vocês, eu verdadeiramente não teria chego aqui.

À minha família, Claudia, Edenilson, Fernanda e Lucas, por sempre me apoiarem e me ajudarem e estarem ao meu lado. Está implícito que este trabalho não existiria sem vocês e quão importante vocês são para mim.

A todos outros colegas, professores, amigos e quaisquer um que tenham influenciado na escrita deste trabalho ou na minha trajetória até aqui, meu mais sincero e profundo obrigado.

If you want to find the secrets of the universe, think in terms of energy, frequency and vibration.
(Se você quer encontrar os segredos do universo, pense em termos de energia, frequência e vibração.)"
(Nikola Tesla, 1856-1943)

RESUMO

Um sistema não-linear é um sistema cuja saída não é um produto linear da entrada, mas há também a presença de distorções harmônicas que alteram a maneira como o sistema se comporta e afetam a saída. A partir da resposta de um sistema não-linear a uma varredura exponencial, é possível separar os componentes de cada harmônico da distorção uns dos outros, e a partir destes há como montar modelos do sistema de forma a simulá-lo sem ter acesso direto ao sistema. Desta forma, o objetivo do trabalho é a avaliação de métodos para modelagem de sistemas não-lineares através da distorção harmônica caracterizada pela varredura exponencial. Foram testados três modelos, sendo dois já discutidos na literatura e um novo, proposto neste trabalho. Os resultados mostraram que é possível obter as respostas harmônicas do sistema e modelá-lo a partir delas, com diferentes níveis de sucesso para tipos diferentes de sistemas, sendo o método proposto por Novák o que trouxe os melhores resultados, e o método novo neste trabalho o que teve pior desempenho.

Palavras-chave: Sistemas não-lineares. Distorção harmônica. Varredura exponencial. Processamento digital de sinais.

ABSTRACT

A non-linear system is a system whose output is not a linear product of the input, but there is also the presence of harmful distortions that alter the way the system behaves and affect the output. From the response of a non-linear system to an exponential sweep, it is possible to separate the components of each harmonic from the distortion of the others, and from these it is possible to assemble models of the system and simulate it without having direct access to the system. Thus, the objective of this work is the evaluation of methods for modeling non-linear systems through harmonic distortion characterized by exponential sweep. Three models were tested, two of which have already been discussed in the literature and a new one that is proposed in this work. The results showed that it is possible to obtain the system's harmonic responses and model it from them, with different levels of success for different types of systems, being the method proposed by Novák the one that brought the best results, and the new method in this work the one with the worst performance.

Keywords: Non-linear systems. Harmonic distortion. Exponential sweep. Digital signal processing.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AAF	Filtro Anti-Recobrimento
DC	Nível Médio
DFT	Transformada Discreta de Fourier
DSTFT	Transformada Discreta de Fourier de Curto Termo
DUT	Dispositivo sob Testes
FT	Transformada de Fourier
HIR	Resposta Impulsiva Harmônica
IRRS	Resposta Impulsiva Referente à Varredura
LIT	Linear e Invariante no Tempo
LT	Transformada de Laplace
MIMO	Múltiplas Entradas, Múltiplas Saídas
MISO	Múltiplas Entradas, Única Saída
NLIT	Não-Linear, Invariante no Tempo
PSA	Algoritmo de Mudança de Tom
PVA	Algoritmo Vocoder de Fase
SIMO	Única Entrada, Múltiplas Saídas
SISO	Única Entrada, Única Saída
SNR	Razão Sinal-Ruído
STFT	Transformada de Fourier de Curto Termo
TF	Função de Transferência
THD	Distorção Harmônica Total

LISTA DE SÍMBOLOS

f_{sr}	Taxa de amostragem
k_{max}	Ordem máxima extraída
$f[n]$	Sinal no tempo discreto
$\mathbb{Z}$	Conjunto dos números inteiros
$f(t)$	Sinal no tempo contínuo
$\mathbb{R}$	Conjunto dos números reais
$\mathbf{f}(t)$	Vetor de sinais no tempo contínuo
$\|\|\|\|_1$	Operador da distância de Minkowski de ordem 1
$S\{\}$	Notação operador para um sistema
$\xrightarrow{S}$	Notação seta para um sistema
$\mathbb{C}$	Conjunto dos números complexos
$*$	Operador de convolução normal
$\overline{*}$	Operador da convolução vetorial
L_1	Norma de Minkowski de ordem 1
$\delta(t)$	Impulso de Dirac
$h(t)$	Resposta ao impulso de um sistema contínuo
$\mathcal{L}\{\}$	Operador da transformada de Laplace
$F(s)$	Sinal na frequência contínua complexa
$\mathcal{F}\{\}$	Operador da transformada de Fourier
$F(\omega)$	Sinal na frequência contínua imaginária
STFT	Operador da STFT
T_{sr}	Período de amostragem
B_{w}	Largura de banda
L_{s}	Comprimento de sequência
$F[\Omega]$	Sinal na frequência discreta
ψ	Resolução no domínio da frequência
L_{w}	Comprimento da janela discreta
DSTFT	Operador da DSTFT
$\binom{n}{k}$	Coeficiente binomial dado por b e k
Δt_k	Atraso de grupo no tempo contínuo
PSA	Operador do algoritmo PSA
f_c	Frequência de corte
$\mathbb{N}$	Conjunto dos números naturais
$\lfloor \rfloor$	Arredondamento para baixo
Δn_k	Atraso de grupo no tempo discreto
dBFS	Decibel relativo à escala completa
$\delta[n]$	Impulso de Dirac discreto

SUMÁRIO

1 INTRODUÇÃO

O comportamento de um sistema Linear e Invariante no Tempo (LIT) é de comum estudo ao longo de cursos de engenharia e em especial na engenharia elétrica. De fato, grande parte dos circuitos estudados ao longo deste curso e que serão analisados durante a vida profissional, tem comportamento linear até algum limite razoável que seja tomado para aquele estudo. Contudo, nem todo sistema que estuda-se em todas as áreas da engenharia é um sistema linear, e portanto é necessário desenvolver ferramentas que tenham capacidade de ajudar a estudar sistemas não-lineares.

O principal método que se utiliza para estudar sistemas que possuem não-linearidades é a análise da distorção harmônica; isto é, dado um sistema não-linear no qual se aplica uma entrada tonal pura[1] de determinada frequência e amplitude, a saída será também um sinal periódico de mesma frequência fundamental que a entrada, mas com a presença de harmônicas múltiplas desta frequência. Sistemas não-lineares que podem ser modelados utilizando a distorção harmônica deles são chamados sistemas não-lineares determinísticos (AUTOR, 2021c).

Essa mesma análise, utilizando a distorção harmônica do sistema e analisando-o através das harmônicas geradas, pode ser aplicada a qualquer sinal de entrada. Dado algum sinal aplicado à entrada de um sistema não-linear, a saída deste sistema será a entrada, acrescida de harmônicos da entrada causados pela distorção não-linear do sistema. Porém, para sinais não-tonais, esta análise se torna mais complexa, sendo necessária uma análise mais profunda e elaborada do sistema.

O interesse de se modelar um sistema não-linear vem do fato de eles serem de difícil estudo utilizando as ferramentas tradicionais já conhecidas e disseminadas, porém sistemas não-lineares têm se tornado cada vez mais presentes nos âmbitos acadêmico e industrial. Por isso, surge a necessidade de buscar técnicas e métodos que sejam capazes de realizar esta modelagem.

Exemplos de sistemas não-lineares que poderiam ser analisados utilizando-se as técnicas estudadas neste trabalho são pedais de distorção, de uso comum na área musical; conversores AC-DC e conversores chaveados, que têm componentes não-lineares como partes fundamentais da sua construção e operação; e o estudo dos próprios componentes não-lineares, como diodos e transistores, de forma a se obter uma modelagem diferente para o seu comportamento.

1.1 OBJETIVOS

Nas subseções a seguir estão descritos o objetivo geral e os objetivos específicos.

[1] Tonal pura é um sinal que possui uma única frequência, ou pode ser tratado como tal

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é estudar os limites da modelagem de sistemas não-lineares determinísticos por meio de suas respostas impulsivas harmônicas, usando como base a série de Volterra e a resposta do sistema à varredura exponencial.

1.1.2 Objetivos Específicos

- Obter as respostas impulsivasharmônicas do sistema a partir da resposta à varredura exponencial.
- Modelar e reconstruir o sistema a partir das respostas impulsivas harmônicas, usando como base a série de Volterra e diferentes métodos para a modelagem.
- Validar os modelos através de métodos numéricos e experimentos.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em 5 capítulos além da Introdução. No Capítulo 2, será feita uma revisão dos principais conceitos, já de comum conhecimento na engenharia elétrica, com enfoque na área de sinais e sistemas lineares, de forma a explanar os fundamentos que serão necessários para o resto do trabalho.

No Capítulo 3 será feita uma exposição da base de conceitos de sistemas não-lineares que não é tratada ao longo do curso de engenharia elétrica, mas que é de relevância para este trabalho, e portanto é de fundamental conhecimento para ser possível compreender os passos seguintes.

No Capítulo 4 serão tratados os métodos utilizados para aplicar os conhecimentos expostos nos capítulos anteriores, de forma a obter os resultados desejados, além dos dispositivos que serão testados e posteriormente modelados de forma a verificar o funcionamento da teoria exposta.

No Capítulo 5 são mostrados e discutidos os resultados obtidos a partir das modelagens e técnicas apresentadas no Capítulo 4, de forma a verificar se os resultados condizem com o esperado e também se eles foram satisfatórios. Por fim, o Capítulo 6 apresenta uma breve conclusão dos resultados obtidos neste trabalho, bem como uma prévia de possíveis trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo será feita uma breve revisão sobre a teoria de sinais e sistemas lineares, de forma a embasar os conhecimentos sobre sistemas não-lineares que seguem no Capítulo 3 e os experimentos no Capítulo 4.

2.1 SINAIS

Segundo Curtarelli (2020), define-se um sinal como "qualquer entidade portadora de informação". Para o escopo deste trabalho, serão considerados somente sinais no domínio do tempo e suas possíveis variações, como transformação destes sinais para o domínio da frequência.

2.1.1 Sinais contínuos e discretos

Um sinal discreto será considerado como sendo um vetor $f[n]$ com um número finito de elementos, sobre um domínio de tempo discreto, normalmente o domínio dos Inteiros $\mathbb{Z}$, e simbolizado pelos colchetes $[\,]$. Um sinal também pode ser uma função $f(t)$ sobre um tempo contínuo, normalmente o domínio dos Reais $\mathbb{R}$, e simbolizado por parênteses $(\,)$.

Como a maioria das propriedades válidas para tempo contínuo também são válidas para tempo discreto, a notação para tempo contínuo será utilizada neste trabalho, exceto quando necessário, que será indicado.

2.1.2 Sinais singulares e vetoriais

Um sinal pode ser representado como uma única entidade portadora de informação $f(t)$, chamada de sinal singular; ou também um conjunto (finito ou infinito) de sinais singulares, da forma $\mathbf{f}(t) = [f_1(t), f_2(t), ...]$ (um vetor-linha), chamado sinal vetorial. Para diferenciar a notação, sinais vetoriais serão representados em negrito.

Além disso, é possível determinar a norma de um sinal vetorial utilizando a distância de Minkowski de Ordem 1 (SHARMA, 2020) e denotada pela Equação 2.1, em que $||\,||_1$ denota o operador da distância.

$$||\mathbf{f}(t)||_1 = \sum_{k=1}^{\infty} f_k(t) \tag{2.1}$$

2.2 SISTEMAS

Um sistema é qualquer "ferramenta ou método para transformação de um sinal", de acordo com Curtarelli (2020). Como um sistema é um nome genérico para uma operação que é realizada sobre um sinal, pode-se definir genericamente um sistema que opera sobre um sinal $\mathbf{x}(t) = [x_1(t), x_2(t), ...]$, produzindo uma saída $\mathbf{y}(t) = [y_1(t), y_2(t), ...]$. A relação entre a entrada

e a saída é dada por um operador S, definido na Equação 2.2. Ou ainda, pode-se utilizar a notação da Equação 2.3 para representar a conexão entre entrada e saída de um sistema $S\{\}$.

$$\mathbf{y}(t) = S\{\mathbf{x}(t), \mathbf{y}(t)\} \tag{2.2}$$

$$\mathbf{x}(t) \xrightarrow{S} \mathbf{y}(t) \tag{2.3}$$

2.2.1 Sistemas lineares

Um sistema linear é todo sistema que respeita os princípios da linearidade. Isto é, as propriedades da homogeneidade e da aditividade são respeitadas. A homogeneidade dita que, dado um sistema com a relação da Equação 2.3, a relação da Equação 2.4 é válida. Já a aditividade dita que, dada ainda a relação da Equação 2.3, e considerando dois sinais $\mathbf{x}_a(t)$ e $\mathbf{x}_b(t)$ que levam a saídas $\mathbf{y}_a(t)$ e $\mathbf{y}_b(t)$, tem-se a relação da Equação 2.5.

$$k \cdot \mathbf{x}(t) \xrightarrow{S} k \cdot \mathbf{y}(t) \qquad \forall\, k \in \mathbb{C} \tag{2.4}$$

$$\mathbf{x}_a(t) + \mathbf{x}_b(t) \xrightarrow{S} \mathbf{y}_a(t) + \mathbf{y}_b(t) \tag{2.5}$$

Portanto, um sistema não-linear é todo sistema que não respeita alguma das propriedades anteriores. Um exemplo simples de um sistema não-linear é $y(t) = [x(t)]^2$. Este sistema não respeita a homogeneidade, como se mostra nas Equações 2.6 e 2.7, e portanto não é linear. Também, não respeita a aditividade, porém não é necessário mostrar, já que basta um princípio não ser respeitado para poder dizê-lo como sendo não-linear.

$$x(t) \xrightarrow{S} [x(t)]^2 = y(t) \tag{2.6}$$

$$k \cdot x(t) \xrightarrow{S} [k \cdot x(t)]^2 = k^2 \cdot [x(t)]^2 = k^2 \cdot y(t) \neq k \cdot y(t) \tag{2.7}$$

2.2.2 Sistemas invariantes no tempo

Uma operação que se pode realizar sobre um sinal qualquer é o deslocamento no tempo. Isto é, dado um sinal $\mathbf{x}(t)$, tem-se que $\mathbf{x}(t + t_o)$ é o mesmo sinal, adiantado em t_o segundos, com $t_o \in \mathrm{Dom}\{t\}$ (t_o pertence ao domínio de t).

Um sistema é dito invariante no tempo caso, dada a relação da Equação 2.3, a relação da Equação 2.8 é respeitada. Fisicamente, um sistema ser invariante no tempo significa que suas características e seu comportamento não se alteram com a passagem do tempo. Na prática, considera-se um prazo dentro do qual os características do sistema não variam significativamente, quando pensa-se em invariância no tempo.

$$\mathbf{x}(t + t_o) \xrightarrow{S} \mathbf{y}(t + t_o) \qquad \forall\, t_o \in \mathrm{Dom}\{t\} \tag{2.8}$$

Caso a Equação 2.8 não seja respeitada, o sistema é dito variante no tempo. Um exemplo de sistema variante no tempo é $y(t) = x(0) + x(t)$. A invariância no tempo é um pré-requisito para os sistemas que serão analisados neste trabalho, já que essa é uma premissa dos modelos implementados.

2.2.3 Sistemas determinísticos

Define-se como determinístico qualquer sistema em que para uma entrada $\mathbf{x}(t)$ há somente uma única saída $\mathbf{y}(t)$ possível (VIALI, 2021). Sistemas não-determinísticos são sistemas estocásticos ou probabilísticos, cuja saída se torna não um sinal, mas uma distribuição de probabilidade sobre um conjunto de possíveis sinais de saída. Neste trabalho, serão tratados somente sistemas de natureza determinística, já que o objetivo é estudar sistemas bem definidos e de comportamento conhecido. Assim, é possível comparar os resultados obtidos ao final do trabalho com os esperados.

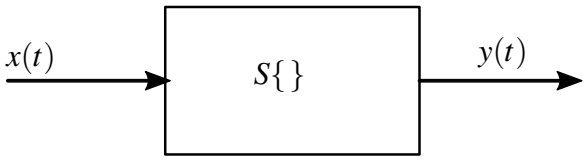
Uma subcategoria de sistemas determinísticos que não está no escopo deste trabalho é a de sistemas caóticos, em que para cada entrada existe somente uma saída, porém pequenas variações nas entradas ou nas condições iniciais do sistema levam a resultados drasticamente diferentes.

2.2.4 Sistemas MISO e SISO

Estes conceitos já foram utilizados na Seção 2.1.2, e agora serão formalmente apresentados. Em um sistema Múltiplas Entradas, Única Saída (MISO)[1], a entrada é um sinal vetorial, mas a saída é um sinal singular. Já sistemas Única Entrada, Única Saída (SISO)[2] são sistemas em que a entrada e saída são sinais singulares. As Figuras 2.1 e 2.2 indicam os esquemáticos de sistemas SISO e MISO, respectivamente.

Há também sistemas Múltiplas Entradas, Múltiplas Saídas (MIMO)[3] e Única Entrada, Múltiplas Saídas (SIMO)[4], porém estes não serão tratados neste trabalho.

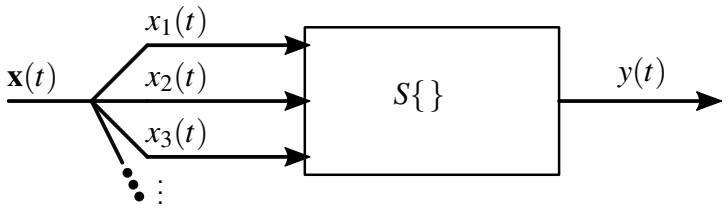
Figura 2.1 – Esquemático de um sistema SISO.



Fonte – do autor.

[1] Do inglês, *Multiple Inputs, Single Output*
[2] Do inglês, *Single Input, Single Output*
[3] Do inglês, *Multiple Inputs, Multiple Outputs*
[4] Do inglês, *Single Input, Multiple Outputs*

Figura 2.2 – Esquemático de um sistema MISO.



Fonte – do autor.

2.2.5 Sistemas LIT

Como já dito no Capítulo 1, um sistema LIT é um sistema que é simultaneamente linear e invariante no tempo. De forma geral, sistemas LIT podem ser descritos e modelados através de equações diferenciais homogêneas a coeficientes constantes; isto é, um sistema LIT MISO pode ser descrito seguindo a Equação 2.9, em que $x_k(t)$ é o k-ésimo elemento do vetor de sinais de entrada $\mathbf{x}(t)$, e os elementos a_n e $b_{(m,k)}$ são constantes. Um sistema LIT SISO é uma particularização da Equação 2.9 para $K = 1$, em que o vetor de sinais de entrada possui somente um elemento.

$$\sum_{n=0}^{N} a_n \cdot \frac{\mathrm{d}^n y(t)}{\mathrm{d}t^n} = \sum_{k=1}^{K} \left(\sum_{m=0}^{M} b_{(m;k)} \cdot \frac{\mathrm{d}^m x_k(t)}{\mathrm{d}t^m} \right) \tag{2.9}$$

2.2.6 Convolução e impulso de Dirac

A convolução normal é uma operação sobre dois sinais singulares, $a(t)$ e $b(t)$, que gera um terceiro sinal $c(t)$ a partir deles, é definida pela Equação 2.10, e simbolizada por $*$. Além disso, define-se também a convolução para sinais vetoriais conforme a Equação 2.11 que é caracterizada pelo operador $\overline{*}$, tem como saída também um vetor, e é chamada de convolução vetorial.

Por questão de reduzir notação, define-se que a operação de convolução normal resulta em um sinal singular, e portanto a convolução normal entre dois sinais vetoriais é representada pela junção da convolução vetorial e da norma L_1, conforme a Equação 2.12.

$$c(t) = a(t) * b(t) = \int_{-\infty}^{\infty} a(\tau) b(t-\tau) \mathrm{d}\tau \tag{2.10}$$

$$\mathbf{c}(t) = \mathbf{a}(t) \,\overline{*}\, \mathbf{b}(t) = [c_1(t), c_2(t), ...] \qquad c_k(t) \stackrel{\Delta}{=} \int_{-\infty}^{\infty} a_k(\tau) b_k(t-\tau) \mathrm{d}\tau \tag{2.11}$$

$$c(t) = \mathbf{a}(t) * \mathbf{b}(t) = ||\mathbf{c}(t)||_1 \stackrel{\Delta}{=} \sum_{k=1}^{\infty} \int_{-\infty}^{\infty} a_k(\tau) b_k(t-\tau) \mathrm{d}\tau \tag{2.12}$$

O impulso de Dirac $\delta(t)$ é uma distribuição matemática teórica, comumente tratada como uma função, fundamental para a área de processamento de sinais (LATHI, 2007), que é definido como $\delta(t) = 0$ para $t \neq 0$, tendo "amplitude infinita" para $t = 0$, e respeita a Equação 2.13.

$$\int_{-\infty}^{\infty} \delta(t)\mathrm{d}t = \int_{0^-}^{0^+} \delta(t)\mathrm{d}t = 1 \tag{2.13}$$

2.2.7 Resposta ao impulso

A resposta ao impulso $h(t)$ de um sistema é a resposta deste quando a sua entrada é o impulso de Dirac. Para um sistema LIT SISO, diz-se que a resposta ao impulso caracteriza o sistema (LATHI, 2007). De maneira genérica, para um sistema LIT MISO com K entradas, haverá K respostas ao impulso $h_k(t)$, que compõem o sinal $\mathbf{h}(t) = [h_1(t), h_2(t)..., h_k(t)]$, sendo que esta agora caracteriza o sistema. Isto é, dado um sistema caracterizado pela Equação 2.9, dos quais foram obtidas as K respostas ao impulso $h_k(t)$ do sistema, pode-se descrevê-lo conforme a Equação 2.14.

$$y(t) = \mathbf{x}(t) * \mathbf{h}(t) \tag{2.14}$$

2.3 TRANSFORMADAS

Uma transformada é uma operação que se realiza sobre um sinal singular com o objetivo de mudar a estrutura com a qual ele se apresenta, de forma a obter informações diferentes ou facilitar sua manipulação matemática.

Transformadas integrais são um tipo de transformada, em que a transformação é realizada através de uma integral. Toda transformada integral é caracterizada pelo seu núcleo $V(u,v)$, em que u é o domínio do sinal original e v é o domínio do sinal transformado, e representada pela Equação 2.15. Por notação, sinais com letra minúscula serão no domínio original, e sinais com letra maiúscula serão a transformada do sinal original.

$$F(v) = \mathcal{T}\{f(u)\}(v) = \int_{-\infty}^{\infty} f(u) \cdot V(u,v)\,\mathrm{d}u \tag{2.15}$$

De forma geral, a transformação integral é uma operação reversível. Dado um núcleo $V(u,v)$, existe um núcleo $\Lambda(v,u)$ que reverte a transformada anterior; isto é, se a transformada direta é dada pela Equação 2.15, o operador $\Lambda(v,u)$ realiza a operação inversa, conforme a Equação 2.16.

$$f(u) = \int_{-\infty}^{\infty} F(v) \cdot \Lambda(v,u)\,\mathrm{d}v \tag{2.16}$$

2.3.1 Transformada de Laplace

A Transformada de Laplace (LT) é uma transformada integral, cujo núcleo é descrito por $V(t,s) = \mathrm{e}^{-st}$, com $s = \sigma + \mathrm{j}\omega \in \mathbb{C}$; ou seja, é definida pela Equação 2.17, e simbolizada por $\mathcal{L}\{\,\}$.

$$F(s) = \mathcal{L}\{f(t)\}(s) = \int_{-\infty}^{\infty} f(t)\mathrm{e}^{-st}\mathrm{d}t \tag{2.17}$$

Como o argumento da exponencial do núcleo da transformada deve ser adimensional, e t é expresso no tempo, s é expresso como uma frequência. Porém, como s é um número complexo, deve-se tratá-lo como sendo uma frequência complexa, em que a parte real representa a atenuação e a parte imaginária a oscilação.

Além de respeitar o princípio da linearidade, a transformada de Laplace possui duas propriedades que fazem com que ela seja de grande interesse: a primeira é ser capaz de transformar operadores diferenciais em uma equação diferencial (STEWART, 2008) em operadores algébricos; e a segunda é que ela transforma a operação de convolução no tempo em uma operação de multiplicação na frequência, conforme o Teorema da Convolução (LATHI, 2007).

Aplicando a transformada de Laplace em cada parcela da equação Equação 2.9, chega-se na Equação 2.18.

$$\sum_{n=0}^{N} a_n \cdot s^n Y(s) = \sum_{k=1}^{K} \left(\sum_{m=0}^{M} b_{(m;k)} \cdot s^m X_k(s) \right) \tag{2.18}$$

Nota-se que no lado esquerdo o termo $Y(s)$ é comum a todos os termos, e no lado direito o termo $X_k(s)$ é comum a cada termo do somatório interno. Portanto, pode-se reescrever a Equação 2.18, obtendo a Equação 2.19, em que $Q(s)$ e todos os $P_k(s)$ são polinômios em função de s.

$$Y(s) \cdot \underbrace{\sum_{n=0}^{N} a_n \cdot s^n}_{Q(s)} = \sum_{k=1}^{K} \left(X_k(s) \cdot \underbrace{\sum_{m=0}^{M} b_{(m;k)} \cdot s^m}_{P_k(s)} \right) \tag{2.19}$$

Considerando que $Q(s)$ não é um polinômio identitariamente nulo, já que isso implicaria na inexistência do sistema, pode-se dividir ambos os lados da Equação 2.19 por $Q(s)$, isolando o termo $Y(s)$ na equação. Com isso, tem-se a Equação 2.20, em que define-se

$\mathbf{H}(s) = [H_1(s), H_2(s), ...]$ como a Função de Transferência (TF)[5] do sistema.

$$Y(s) = \sum_{k=1}^{K} \left(X_k(s) \cdot \underbrace{\frac{P_k(s)}{Q(s)}}_{H_k(s)} \right) \tag{2.20}$$

Por meio do teorema da Convolução, e tomando que $H_k(s)$ é a transformada de Laplace de $h_k(t)$ (LATHI, 2007), obtém-se a Equação 2.21.

$$y(t) = \sum_{k=1}^{K} \big(x_k(t) * h_k(t) \big) = \mathbf{x}(t) * \mathbf{h}(t) \tag{2.21}$$

Portanto, conhecendo-se a resposta ao impulso $\mathbf{h}(t)$ do sistema, é possível modelá-lo e reescrevê-lo em função dela, sem que seja necessário acesso ao sistema. Neste trabalho, todas as implementações de convolução serão realizadas através do teorema da convolução, por ser computacionalmente mais eficiente e não causar problemas relevantes para este trabalho.

2.3.2 Deconvolução

Seguindo o teorema da convolução (LATHI, 2007), tem-se a relação da Equação 2.22. Com isso, dado que $c(t) = a(t) * b(t)$, pode-se definir a operação de deconvolução conforme a Equação 2.23, fazendo uso da transformada de Laplace. Esta operação tem o papel de inverter o que a operação de convolução faz. Isto é, conhecendo-se $b(t)$ e $c(t)$, e a relação da Equação 2.22, pode-se determinar $a(t)$.

$$a(t) * b(t) \xrightarrow{\mathcal{L}\{\}} A(s) \cdot B(s) \tag{2.22}$$

$$a(t) = \mathcal{L}^{-1} \left\{ \frac{\mathcal{L}\{c(t)\}}{\mathcal{L}\{b(t)\}} \right\} \tag{2.23}$$

2.3.3 Transformada de Fourier

Semelhante à transformada de Laplace, a Transformada de Fourier (FT)[6] é definida pelo seu núcleo $V(t, \omega) = \mathrm{e}^{-\mathrm{j}\omega t}$ ($\omega \in \mathbb{R}$) descrita na Equação 2.24, e simbolizada por $\mathcal{F}\{\}$. Ela pode ser pensada como um caso particular da transformada de Laplace em que $s = \mathrm{j}\omega$, ou seja a análise da transformada de Laplace sobre o eixo imaginário do plano complexo.

$$F(\omega) = \mathcal{F}\{f(t)\}(\omega) = \int_{-\infty}^{\infty} f(t) \mathrm{e}^{-\mathrm{j}\omega t} \, \mathrm{d}t \tag{2.24}$$

Como explicado na Seção 2.3.1, s representa uma frequência complexa, em que a parte imaginária representa a oscilação sem decaimento. Portanto, analisar a transformada de

[5] Do inglês, *Transfer Function*
[6] Do inglês, *Fourier Transform*

Laplace projetada sobre o eixo imaginário, que por definição é idêntico a analisar a transformada de Fourier, significa analisar as componentes de frequência puras de um sinal. Enquanto a principal utilidade da transformada de Laplace é estudar sistemas e facilitar a sua manipulação, a transformada de Fourier é fundamental para analisar, visualizar e compreender sinais. Chama-se a transformada de Fourier de um sinal de espectro.

2.3.4 Transformada de Fourier de curto termo

Janelamento é o processo de truncar um sinal de comprimento desconhecido em um intervalo finito no tempo, de forma a analisá-lo separadamente (OPPENHEIM; SCHAFER; BUCK, 1999). A janela é a função que é utilizada para janelar o sinal original, normalmente representada por $w(t)$, e a partir disso pode-se definir a operação de janelamento pela Equação 2.25, em que $f(t)$ é o sinal original, $f_w(t)$ é o sinal janelado, e τ é o instante onde a origem da janela estará centrada.

$$f_w(t) = f(t) \cdot w(t - \tau) \tag{2.25}$$

A Transformada de Fourier de Curto Termo (STFT)[7] é uma aplicação da FT com um sinal janelado (ROYER, 2019). Portanto, a STFT possui dois parâmetros no domínio transformado: ω, que indica a frequência; e τ que indica o instante do janelamento. A STFT é caracterizada pela equação da Equação 2.26.

$$F(\omega, \tau) = \mathbf{STFT}\{f(t)\}(\omega, \tau) = \int_{-\infty}^{\infty} f(t) \cdot w(t - \tau)\, \mathrm{e}^{-\mathrm{j}\omega t} \mathrm{d}t \tag{2.26}$$

2.4 SISTEMAS DISCRETOS

Conforme exposto na Seção 2.1.1, boa parte das propriedades que são válidas para sistemas contínuos, também são válidas para sistemas discretos. Porém há alguns detalhes e nuances em sistemas discretos que é importante que sejam destacados, já que eles não possuem um equivalente no domínio contínuo, e também não são óbvios e triviais.

2.4.1 Processo de amostragem

A amostragem é o processo de obter um sinal contínuo, mas que só é conhecido em alguns valores, a partir de um sinal contínuo. Mesmo que os detalhes de como funcionam os diferentes tipos de amostragem não sejam discutidos, é conveniente rever esse conceito e alguns efeitos. A primeira noção a se apresentar é a do período de amostragem T_{sr}[8]. Dado um sinal contínuo singular $f(t)$, pode-se extrair seu valor a cada T_{sr} segundos, de forma a saber

[7] Do inglês, *Short-time Fourier Transform*
[8] Do inglês, *sampletime*

o valor da função naquele instante. O inverso de T_{sr} é a taxa de amostragem f_{sr}[9]. Utilizando as propriedades do impulso de Dirac, obtém-se Equação 2.27, que define a amostragem ideal (LATHI, 2007).

$$f_a(t) = f(t) \cdot \sum_{k=-\infty}^{\infty} \delta(t - kT_{\mathrm{sr}}) = \sum_{k=-\infty}^{\infty} f(kT_{\mathrm{sr}}) \cdot \delta(t - kT_{\mathrm{sr}}) \tag{2.27}$$

Com isso, o sinal amostrado $f_a(t)$ pode ser representado por uma sequência infinita de impulsos, cada um realizado a cada T_{sr} segundos, em que a magnitude do impulso é o valor da função naquele ponto.

Um efeito que a discretização causa é que a transformada de Fourier do sinal torna-se um sinal periódico (CURTARELLI, 2020), em que o espectro original do sinal repete-se infinitas vezes no domínio da frequência. O período do sinal no domínio da frequência é a taxa de amostragem f_{sr}. Em função disso, é possível que haja sobreposição dos espectros, o que causaria distorção do sinal amostrado e este não seria uma representação correta do sinal original. Portanto, é necessário escolher uma taxa de amostragem alta o suficiente de forma a evitar este recobrimento e, assim, evitar esse tipo de distorção. O teorema de amostragem de Nyquist-Shannon (LATHI, 2007) dita que é necessário que, se o sinal for limitado em frequência com largura de banda B_{w}, tal que $X(\omega > B_{\mathrm{w}}) = 0$, então $f_{\mathrm{sr}} = 2 \cdot B_{\mathrm{w}}$ garante que não haverá recobrimento na frequência. Esta frequência de $2 \cdot B_{\mathrm{w}}$ é chamada de taxa de Nyquist (LATHI, 2007).

Na prática não se adapta a taxa de amostragem ao sinal, mas sim o contrário. Utilizando filtros passa-baixas analógicos (NOCETI FILHO, 2020) antes do processo de amostragem, pode-se limitar a banda do sinal à metade da taxa de amostragem. Este filtro específico é chamado de Filtro Anti-Recobrimento (AAF)[10]. Embora sua utilização também cause alterações na forma do sinal, diferentes aplicações utilizam larguras de banda diferentes do espectro de frequência, então informação do sinal acima de B_{w} é desnecessária para aquela aplicação. Um exemplo é o processamento de áudio: a audição humana comumente limita-se à faixa $20 - 20$k Hz (PUJOL; TRIGUEIROS-CUNHA, 2018). Portanto, ao se gravar um sinal de áudio, não é necessário preocupar-se com frequências acima de 20k Hz. Na indústria fonográfica, normalmente é utilizada a taxa de amostragem de 44.1k Hz, um pouco acima da taxa de Nyquist que seria 40k Hz. Como filtros passa-baixa analógicos não são ideiais[11], deve-se evitar que a taxa de Nyquist caia no meio da banda de transição do filtro (LATHI, 2007). Por isso, não se utiliza $f_{\mathrm{sr}} = 2 \cdot B_{\mathrm{w}}$, mas sim um pouco acima, para levar em conta a banda de transição do filtro.

[9] Do inglês, *samplerate*

[10] Do inglês, *Anti-Aliasing Filter* (AUTOR, 2021a)

[11] Filtros passa-baixa ideais são filtros em que $F(s \leq f_c) = 1$ e $F(s > f_c) = 0$, onde f_c é a frequência de corte (NOCETI FILHO, 2020), e $F(s)$ é a função de transferência do filtro

2.4.2 Discretização

A discretização é o passo de, a partir dos valores amostrados de $f_a(t)$ obtidos de $f(t)$ através da amostragem, registrar estes valores em algum meio discreto, como disco rígido ou fita magnética, como uma sequência de valores $f[n]$.

Na teoria esta seria uma sequência infinita de valores, com $n \in \mathbb{Z}$, porém isto é impossível de se realizar fisicamente. Na prática possui-se uma sequência finita de valores, com um comprimento de vetor representado por L_s.

Este passo é fundamental pois com isso é possível armazenar os dados obtidos a partir da amostragem em hardware, para que se possa realizar um pós-processamento digital sobre o sinal. O passo de armazenar em hardware também implicaria em distorções não-lineares devido à quantização dos valores, chamada de ruído de quantização que pode ser calculado pela Equação 2.28 (WIDROW; KOLLÁR, 2008), em que Q indica o número de bits utilizado na digitalização. Portanto, com um número suficientemente grande de bits/amostra essa distorção é desprezível (BONCELET, 2009) e menor do que o ruído de fundo presente na medição.

$$\mathrm{SQNR} = 20\log_{10}\left(2^Q\right) \approx 6.02Q\,\mathrm{dB} \tag{2.28}$$

2.4.3 Transformada Discreta de Fourier e DSTFT

A Transformada Discreta de Fourier (DFT)[12] é o equivalente discreto à transformada de Fourier, definida pela Equação 2.29 (LATHI, 2007).

$$F[\Omega] = \sum_{n=0}^{L_\mathrm{s}-1} f[n]\mathrm{e}^{-\frac{\mathrm{j}2\pi}{L_\mathrm{s}}\Omega\cdot n} \tag{2.29}$$

Um detalhe relevante sobre a DFT é que o sinal $F[\Omega]$ também será um sinal discreto e periódico, com período L_s. Por causa disso, $F[\Omega]$ normalmente só é definido para $\Omega \in [0, L_\mathrm{s}-1] \in \mathbb{Z}$ (OPPENHEIM *et al.*, 1996). É comum visualizar a DFT em função não de L_s, mas sim f_sr, em que há um mapeamento linear de $L_\mathrm{s}-1$ para f_sr. Isso ajuda a visualizar e entender o espectro do sinal discreto em função de frequências tangíveis. Por fim, tem-se que $\psi = f_\mathrm{sr}/L_\mathrm{s}$ é a resolução no domínio da frequência da DFT (ROYER, 2019). Ou seja, $\Omega\cdot\psi$ indica o equivalente a uma frequência contínua (em Hz), da frequência discreta de Ω. Outra particularidade da DFT é que a implementação de convolução utilizando dela utilizando o teorema da convolução faz com que a convolução seja cíclica (OPPENHEIM *et al.*, 1996). Embora isto possa ser um problema, já que o comportamento da convolução linear é diferente do comportamento da convolução cíclica, pode-se evitar distorções adicionando-se uma sequência de zeros ao final de ambos os sinais.

Além da DFT, também é possível realizar uma análise discreta da STFT, obtendo a Transformada Discreta de Fourier de Curto Termo (DSTFT)[13] (ROYER, 2019), que é descrita

[12] Do inglês, *Discrete Fourier Transform*
[13] Do inglês, *Discrete Short-time Fourier Transform*

pela Equação 2.30. Em função do janelamento discreto, o sinal a ser analisado será limitado não à L_{s}, mas sim ao comprimento da janela, denotado L_{w}. Como a janela representa somente uma parte do sinal, tem-se $L_{\mathrm{w}} \leq L_{\mathrm{s}}$.

$$F[\Omega,\eta] = \mathbf{DSTFT}\{f[n],w[n]\}(\Omega,\eta) = \sum_{n=0}^{L_{\mathrm{w}}-1} f[n+\eta]\cdot w[n]\mathrm{e}^{-\frac{\mathrm{j}2\pi}{L_{\mathrm{w}}}\Omega\cdot n} \tag{2.30}$$

3 SISTEMAS NÃO-LINEARES

Como explicado na Seção 2.2.1, um sistema não-linear não respeita a homogeneidade e/ou a aditividade. Existem dois tipos de não-linearidades: não-linearidades determinísticas e não-linearidades probabilísticas, a diferença tendo sido explicada na Seção 2.2.3.

3.1 DISTORÇÃO HARMÔNICA

Dado um sistema Não-Linear, Invariante no Tempo (NLIT) SISO e determinístico, pode-se aplicar uma entrada senoidal neste sistema, da forma $x(t) = A \cdot \cos(\omega_0 t)$. Considerando que o sinal existe deste $t = -\infty$, pode-se considerar que a saída já alcançou seu regime estacionário. Devido ao fato de o sistema ser determinístico e invariante no tempo, tem-se que cada período da resposta do sistema será idêntico em forma, e portanto o sinal de saída $y(t)$ também será um sinal periódico, com mesmo período do sinal de entrada. Assim, pode-se escrever a saída $y(t)$ em função da sua série de Fourier de cossenos defasados (CURTARELLI, 2020).

$$y(t) = \sum_{k=0}^{\infty} D_k \cdot \cos\left(k\left[\omega_0 t + \frac{\phi_k}{k}\right]\right) \tag{3.1}$$

Com isso, tem-se que para uma entrada senoidal, um sistema NLIT resultará em uma saída também periódica, de mesmo período, representado por uma sequência infinita de senoides com frequência múltipla da frequência fundamental do sinal de entrada. Este efeito do surgimento de harmônicas do sinal de entrada, a partir da sua frequência fundamental, é chamado de distorção harmônica (KRARTI, 2018).

A distorção harmônica também pode ser caracterizada por meio das potências da senoide fundamental. Utilizando a fórmula de De Moivre (SCHNEIDER, 2011), pode-se reescrever um cosseno $\cos(nx)$ como na equação Equação 3.2[1]. Com isso, é possível passar a série de Fourier da Equação 3.1 para a forma da Equação 3.3, sem perda de informação. Os coeficientes ϑ_n no momento não são relevantes, mas é importante saber que tal conversão é possível. Assim, pode-se modelar distorções harmônicas através de uma série de Fourier, e também por meio de uma série de potências conforme as Equações 3.1 e 3.3 respectivamente.

$$\cos(nx) = \sum_{k=0}^{n} \binom{n}{k} [\cos(x)]^n \cdot [\sin(x)]^{n-k} \cdot \cos\left(\frac{(n-k)\pi}{2}\right) \tag{3.2}$$

$$y(t) = \sum_{n=0}^{\infty} \vartheta_n \cdot \left[\cos\left(\omega_0 t + \frac{\phi_n}{n}\right)\right]^n \tag{3.3}$$

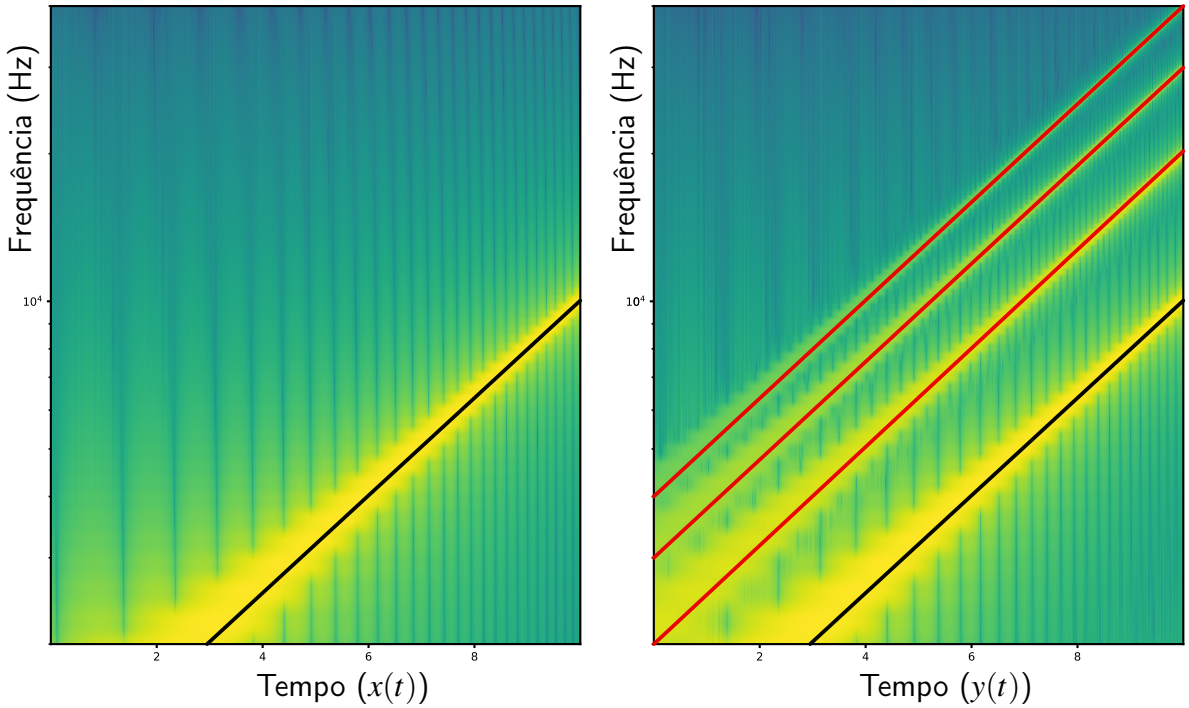
A Figura 3.1 mostra um exemplo do surgimento de harmônicas. O sistema das figuras é um sistema NLIT sem memória[2]. Em ambas as subfiguras, o eixo das abscissas é o sinal

[1] $\binom{n}{k}$ simboliza o coeficiente binomial dado por b e k (AUTOR, 2021b)

[2] Sistemas sem memória são sistemas em que a saída depende da entrada instantaneamente, e não depende da entrada no passado (CURTARELLI, 2020)

no tempo, e o eixo das ordenadas é o seu conteúdo espectral em torno daquele instante, obtido por meio da DSTFT do sinal. As figuras estão em escala de cores, de forma que cores mais escuras indicam valores menores, e cores mais escuras valores maiores. As linhas foram adicionadas artificialmente para facilitar visualização. Na figura da saída (Figura 3.1b), a linha preta corresponde à parte linear da resposta do sistema, e as linhas vermelhas à distorção harmônica do sistema não-linear.

Figura 3.1 – Espectrograma dos sinais de entrada e saída de um sistema NLIT.



(a) Sinal de entrada $x(\omega, \tau)$ (b) Sinal de saída $y(\omega, \tau)$

Fonte – do autor.

3.2 VARREDURAS EXPONENCIAIS

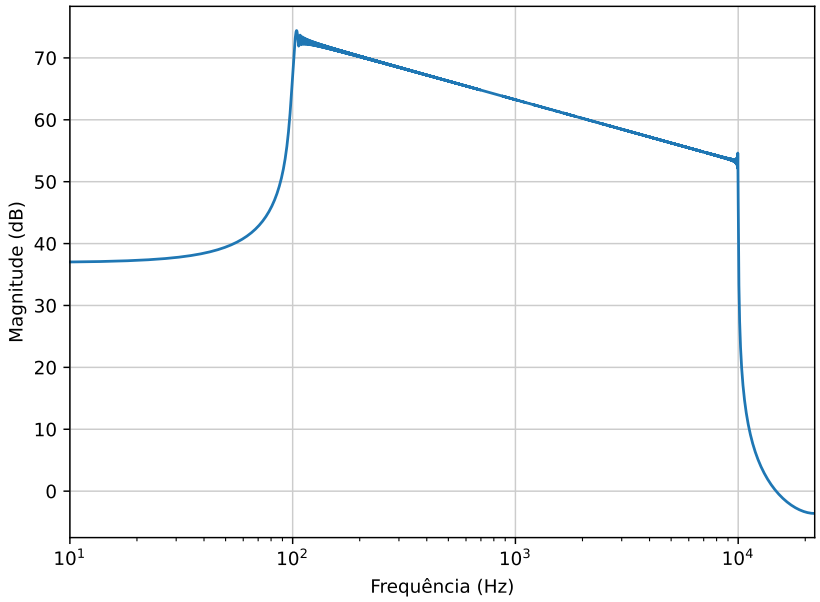
Um varredura é uma função $g(t)$ oscilatória com frequência variável, tal que a frequência da portadora contemple todo o espectro de frequências (ou uma faixa desejada). Uma varredura exponencial é, como indica o nome, uma varredura cuja variação de frequência ocorre exponencialmente. Farina (2000) define a equação da varredura exponencial pela expressão da Equação 3.4, em que ω_1 é a frequência angular inicial da varredura; ω_2 é a frequência angular final da varredura; e T é a duração da varredura (em segundos).

$$x(t) = \sin\left(\frac{\omega_1 \cdot T}{\ln\left(\frac{\omega_2}{\omega_1}\right)} \cdot \left(e^{\frac{t}{T}\cdot\ln\left(\frac{\omega_2}{\omega_1}\right)} - 1\right)\right) \tag{3.4}$$

O espectrograma de uma varredura com $f_1 = 100$ Hz, $f_2 = 10000$ Hz e $T = 10$ s é mostrado na Figura 3.1a, em que a frequência cresce exponencialmente, já que o eixo das

ordenadas está em escala logarítmica. Além disso, o seu espectro em frequência se encontra na Figura 3.2, onde é visível que há pouca informação presente no sinal fora da banda de interesse, entre f_1 e f_2, além de haver mais energia no sinal em baixas frequências, resultado que é esperado da varredura função do comportamento exponencial da variação da frequência do sinal, o que faz com que as baixas frequências sejam mais relevantes, tanto no tempo quanto na frequência.

Figura 3.2 – Módulo do espectro de uma varredura com $f_1 = 100$ Hz, $f_2 = 10000$ Hz e $T = 10$ s



Fonte – do autor.

A utilização de uma varredura para realizar medições de forma a se caracterizar um sistema físico se dá pela razão que ela é capaz de separar a resposta impulsiva referente à varredura em suas respostas impulsivas harmônicas (FARINA, 2000; MÜLLER; MASSARANI, 2008). É possível entender isso intuitivamente da seguinte maneira: em um dado instante de tempo t_o, a varredura exponencial terá uma frequência instantânea ω_o e, ao se passar a varredura $x(t)$ por um sistema NLIT, obtendo a saída $y(t)$, este causará harmônicas, e no instante t_o, surgirão harmônicas $n\,\omega_o$.

Como a frequência instantânea de $x(t)$ é monótona crescente, a frequência instantânea $n\,\omega_o$ só será aplicada na entrada em um instante no futuro; portanto, pode-se considerar que há uma não-causalidade[3] na resposta do sistema à varredura. Esta não-causalidade é um resultado matemático, que representa as não-linearidades do sistema físico causal.

Assim, ao se fazer a deconvolução de $y(t)$ com $x(t)$ (NOVÁK *et al.*, 2010), obtém-se o sinal $\tilde{h}(t)$ chamado de Resposta Impulsiva Referente à Varredura (IRRS). Como todo sistema

[3] Um sinal $f(t)$ é não-causal se $f(t) \neq 0$ para algum $t < 0$ (OPPENHEIM *et al.*, 1996)

físico é um sistema causal (OPPENHEIM; WILLSKY; NAWAB, 1996), para um sistema linear o sinal $\tilde{h}(t)$ também seria um sinal causal.

Para o sistema não-linear, este sinal possui uma parte causal, que é a parte da saída referente à primeira harmônica do sistema, porém também surgirá uma parte não-causal. Em função da causalidade do sistema, esta parte não será devido aos comportamentos lineares do sistema, e portanto tem-se que eles são devidos às não-linearidades do mesmo. $\tilde{h}(t)$ é chamado de Resposta Impulsiva Harmônica (HIR) por um abuso de notação, devido ao fato de que para um sistema linear ele seria realmente a resposta ao impulso.

O atraso temporal é o atraso no tempo das respostas não-lineares do sistema (NOVÁK *et al.*, 2010). Para a varredura exponencial, o atraso temporal é dado pela Equação 3.5 (FARINA, 2000). A principal vantagem da varredura exponencial sobre outras é que o atraso temporal é constante, já que Δt_k depende somente de k e não do instante t e, portanto, para cada k-ésima harmônica a resposta dessa está deslocada no tempo por uma constante fixa, e não ocorre um espalhamento dela no tempo.

$$\Delta t_k = T \cdot \frac{\ln(k)}{\ln\left(\frac{\omega_2}{\omega_1}\right)} \tag{3.5}$$

Ou seja, é possível separar cada HIR $\tilde{h}_k(t)$ a partir de $\tilde{h}(t)$, até um certo limite. Embora Δt_k cresça com a ordem da harmônica, este atraso no tempo cresce logaritmicamente. Desta maneira, a partir de certo ponto as HIR começarão a sobrepôr-se, e não será possível separá-las.

Sabendo que cada HIR estará atrasada no tempo com um atraso Δt_k, pode-se reescrever $\tilde{h}(t)$ em função da soma das HIR, atrasadas, conforme a Equação 3.6, em que $\tilde{h}_k(t)$ é a HIR do sistema para harmônica de grau k. Além disso, também é possível descrever a HIR $\tilde{\mathbf{h}}(t)$ conforme a Equação 3.7, o que faz com que haja uma relação entre o sinal $\tilde{h}(t)$ que é um sinal singular e $\tilde{\mathbf{h}}(t)$ que é um sinal múltiplo.

$$\tilde{h}(t) = \sum_{k=1}^{\infty} \tilde{h}_k(t + \Delta t_k) \tag{3.6}$$

$$\tilde{\mathbf{h}}(t) = [h_1(t), h_2(t), ...,] \tag{3.7}$$

3.3 SÉRIE DE VOLTERRA

Seja um sistema NLIT sem memória. Dado que é um sistema sem memória, a partir da Equação 2.2 pode-se concluir que $y(t) = S\{x(t), y(t)\} = w(x(t))$, em que $w(x(t))$ é uma função matemática que opera no valor de $x(t)$ instantaneamente para determinar sua saída. Supõe-se que $w(x(t))$ seja uma função suave (MALTA *et al.*, 1993). Assim, pode-se descrever o sistema S em termos da sua série de Taylor (BOYD, 1980), escrevendo-o em função das

potências de $x(t)$, conforme a equação Equação 3.8.

$$S\{x(t)\} = w(x(t)) = w(0) + w'(0) \cdot x(t) + \frac{w''(0)}{2} \cdot [x(t)]^2 + ... = \sum_{k=0}^{\infty} \frac{w^{(n)}(0)}{k!} \cdot [x(t)]^k \quad (3.8)$$

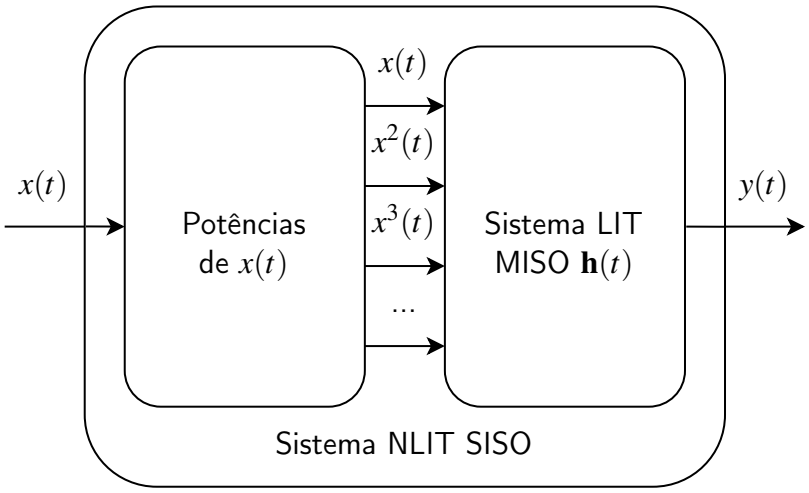
Escolhe-se $w(x = 0)$ como sendo o ponto central da expansão por conveniência de notação, já que qualquer valor poderia ser utilizado. Esta expansão, porém, é válida somente para sistemas sem memória, já que $w(x(t))$ depende de $x(t)$ instantaneamente, e não de instantes passados e/ou futuros (LATHI, 2007). É possível emular um sistema com memória a partir da Equação 3.8 ao se convoluir esta resposta que possui distorção harmônica, com um filtro $h'(t)$ (NOVÁK *et al.*, 2010), responsável pela reverberação (AUTOR, 2021e) do sistema, ou seja, pelo efeito de memória. A partir disso, tem-se a Equação 3.10 em que $\mathbf{x}(t) = [x(t), x(t)^2, x(t)^3, ...]$.

$$\begin{aligned} f(x(t)) = w(x(t)) * h'(t) &= \sum_{k=0}^{\infty} \left(\frac{w^{(k)}(0)}{k!} \cdot [x(t)]^k * h'(t) \right) \\ &= \sum_{n=0}^{\infty} \left(\frac{w^{(k)}(0)}{k!} \cdot h'(t) \right) * [x(t)]^k \end{aligned} \quad (3.9)$$

$$y(t) = f(x(t)) = \sum_{k=0}^{\infty} h_k(t) * [x(t)]^k = \mathbf{h}(t) * \mathbf{x}(t) \quad (3.10)$$

A Equação 3.10 é uma formulação possível para expressar a série de Volterra. A série de Volterra pode ser tratada como um conjunto de infinitas funções $h_k(t)$ que descrevem um sistema não-linear com distorções harmônicas (HÉLIE, 2015). Enquanto no caso de um sistema linear é possível obter uma única função $h(t)$ que descreve o sistema, para um sistema não-linear obtém-se uma família de funções $h_k(t)$ denominada núcleo (BOYD, 1980), que capturam tanto a informação da distorção harmônica do sistema, bem como transportam a informação da reverberação e do perfil na frequência da resposta linear do sistema.

Figura 3.3 – Modelagem de sistema NLIT SISO como um sistema LIT MISO com entradas potências da entrada do sistema SISO.



Fonte – do autor.

Comparando a Equação 3.10 com a Equação 2.21, percebe-se que é possível descrever um sistema NLIT SISO como sendo um sistema LIT MISO, em que $\mathbf{x}(t) = [x(t), x^2(t), x^3(t), ...]$, e $\mathbf{h}(t) = [h_1(t), h_2(t), ...]$. Um esquemático desta modelagem está na Figura 3.3.

3.4 RESPOSTAS IMPULSIVAS

Até então foram tratados dois tipos de respostas impulsivas do sistema: as respostas impulsivas $h_k(t)$ por meio de potências do sinal, necessários para aplicação da série de Volterra; e as HIR $\tilde{h}_k(t)$ para múltiplos da frequência instantânea do sinal, obtidos a partir da resposta do sistema não-linear à varredura exponencial. Estas respostas impulsivas não necessariamente são iguais, embora seja possível que haja uma correlação entre elas, e portanto é necessário encontrar uma solução para compatibilizá-las, além de produzir um sinal $\mathbf{x}(t)$ adequado.

3.4.1 Solução equivocada

A primeira ideia de solução para este problema de incompatibilidade entre os tipos de respostas impulsivas seria não convertê-las, tratando $\tilde{\mathbf{h}}(t) = \mathbf{h}(t)$, e utilizar $\mathbf{x}(t) = [x(t), x(2t), ...]$. Esta abordagem resulta em dois problemas:

O primeiro, que utilizando as propriedades da FT o sinal $x(kt)$ não seria um sinal com k vezes a frequência, mas sim ele teria uma duração k vezes menor, tanto no tempo contínuo quanto no discreto.

O segundo que as HIR $\tilde{h}_k(t)$ são respostas impulsivas à frequência instantânea do sinal. Ou seja, cada harmônico que surge, surge de forma instantânea, não levando em conta todo o espectro do sinal mas sim o espectro dele em torno daquele instante de tempo.

3.4.2 Soluções tradicional e de Novák, e intermodulação

A solução tradicional para este problema é considerar que $\tilde{\mathbf{h}}(t) = \mathbf{h}(t)$, e utilizar $x_k(t) = [x(t)]^k$ (FARINA, 2000). O principal problema desta solução é que esta equivalência entre ambas as respostas não é tão simples, já que uma se trata de múltiplos de frequência, e outra de potências do sinal. Considerando como exemplo um cosseno $\cos(\omega_0 t)$ e sua terceira potência, tem-se a relação da Equação 3.11 em que uma potência do cosseno gera harmônicos diferentes do grau da potência, como seria esperado devido à fórmula de De Moivre da Equação 3.2. Este mesmo efeito seria esperado para outras potências, em que surgiriam harmônicos de ordem inferior.

$$[\cos(\omega_0 t)]^3 = \frac{\cos(3\omega_0 t)}{4} + \frac{3\cos(\omega_0 t)}{4} \tag{3.11}$$

Em seu artigo, Novák *et al.* (2010) propõem uma solução para este problema, que consiste em converter de $\tilde{\mathbf{h}}(t)$ para $\mathbf{h}(t)$ por meio de uma matriz A, definida pela Equação 3.12[4], em que a conversão é dada pelas Equações 3.13 e 3.14 (AUTOR, 2021d). Essa conversão segue a mesma ideia da fórmula de De Moivre.

$$A_{n,m} = \begin{cases} \frac{1}{2^{n-1}} \cdot \binom{n}{\lfloor \frac{n}{2}+m \rfloor} & \text{, se } n \geq m \text{ e } (n+m) \bmod 2 = 0, \\ 0 & \text{, caso contrário} \end{cases} \tag{3.12}$$

$$\begin{bmatrix} H_{d1}(\omega) \\ H_{d2}(\omega) \\ H_{d3}(\omega) \\ \vdots \end{bmatrix} = (A^T)^{-1} \cdot \begin{bmatrix} \tilde{H}_{d1}(\omega) \\ \tilde{H}_{d2}(\omega) \\ \tilde{H}_{d3}(\omega) \\ \vdots \end{bmatrix} \tag{3.13}$$

$$\mathbf{H}_d(\omega) = \left((A^T)^{-1} \cdot \tilde{\mathbf{H}}_d(\omega)^T \right)^T = \tilde{\mathbf{H}}_d(\omega) \cdot A^{-1} \tag{3.14}$$

O subscrito d nos espectros indica que se trata somente do semieixo direito da reta imaginária. Tomando que os sinais $\mathbf{h}(t)$ e $\tilde{\mathbf{h}}(t)$ são reais, tem-se que seus espectros são sinais simétrico-conjugados; isto é, $H_k(\omega) = H_k(-\omega)^*$ para qualquer k, e também válido para $\tilde{H}_k$ qualquer. Portanto, pode-se analisar somente o semieixo direito da reta imaginária, e depois extrapolar para a metade esquerda. Este método, fazendo uso da matriz A, não leva em consideração o efeito de intermodulação.

O efeito de intermodulação (RUMSEY; MCCORMICK, 2006), ou distorção de intermodulação, surge quando uma ou mais senoides de frequências diferentes excitam um sistema não-linear. Seja um sistema não-linear $y(t) = x(t)^2$. Ao se entrar com uma senoide $x(t) = \cos(\omega_0 t)$, tem-se que a saída do sistema será conforme a Equação 3.15, através de

[4] a mod b é o resto da divisão inteira de a por b (MODULO..., 2021)

propriedades trigonométricas. À primeira vista, este resultado é promissor, já que é o desejado exceto por um fator Nível Médio (DC).

$$y(t) = [\cos(\omega_0 t)]^2 = \frac{\cos(2\omega_0 t) + 1}{2} \tag{3.15}$$

Porém, ao se colocar um sinal $x(t) = \cos(\omega_0 t) + \cos(\omega_1 t)$[5] na entrada do sistema, a saída não será da forma da Equação 3.15, mas sim da forma Equação 3.16.

$$\begin{aligned} y(t) &= [\cos(\omega_0 t) + \cos(\omega_1 t)]^2 \\ &= \underbrace{\frac{\cos(2\omega_0 t) + 1}{2}}_{[1]} + \underbrace{\frac{\cos(2\omega_1 t) + 1}{2}}_{[2]} + \underbrace{4 \cdot \left(\cos\frac{(\omega_0 + \omega_1)t}{2} + \cos\frac{(\omega_0 - \omega_1)t}{2} \right)}_{[3]} \end{aligned} \tag{3.16}$$

Os termos [1] e [2] da Equação 3.16 são o resultado desejado, em que cada frequência sai dobrada, porém há também o termo [3] que é o resultado da intermodulação. Esta mesma lógica pode ser aplicada a qualquer sinal no tempo, ao se pensar na transformada de Fourier. Das propriedades da FT (OPPENHEIM *et al.*, 1996), pode-se utilizar a da Equação 3.17. Ou seja, $x(t)^2 \xrightarrow{\mathcal{F}\{\}} X(\omega) * X(\omega)$ e, voltando à Equação 2.10, conclui-se que para uma frequência ω_0 a saída não dependerá somente de $X(\omega_0)$, mas de todo o espectro de frequência (em função da convolução).

$$a(t) \cdot b(t) \xrightarrow{\mathcal{F}\{\}} \frac{1}{2\pi} A(\omega) * B(\omega) \tag{3.17}$$

3.4.3 Algoritmo de mudança de tom

Um Algoritmo de Mudança de Tom (PSA)[6] (ROYER, 2019) é um tipo de algoritmo empregado amplamente na área musical, utilizado para harmonização e correção de tom em partes específicas de uma música ou de um sinal. De forma geral, um PSA funciona aumentando a frequência do sinal em um trecho, sem alterar a sua duração, utilizando diversos métodos para alcançar este efeito.

O objetivo da utilização de um PSA aqui é gerar sinais $x_k(t)$ que possuam frequências instantâneas k vezes maior que $x_1(t) = x(t)$ ao longo de todo a duração do sinal, para que sejam utilizados com as respostas impulsivas de $\mathbf{h}(t) = \tilde{\mathbf{h}}(t)$ obtidas a partir da IRRS. Isto é modelado utilizando as Equações 3.18 e 3.19.

$$\mathbf{x}_k(t) = [x_1(t), x_2(t), ...] \text{ , em que } x_k(t) = \mathbf{PSA}\{x(t), k\} \tag{3.18}$$

$$y(t) = \mathbf{x}(t) * \mathbf{h}(t) \tag{3.19}$$

O algoritmo a ser usado é um Algoritmo Vocoder de Fase (PVA)[7], que opera no

[5] com $\omega_0 \neq \omega_1$
[6] Do inglês, *Pitch Shifting Algorithm*
[7] Do inglês, *Phase Vocoder Algorithm*

domínio da frequência através da DSTFT (ROYER, 2019), e utiliza uma abordagem semelhante à apresentada na Seção 3.4.1, tratando o espectro de frequências do sinal dentro de cada janelamento da DSTFT como sendo as frequências instantâneas do sinal, conforme supõe-se que causa o surgimento da distorção harmônica. Um diagrama da operação do PVA pode ser encontrado na Figura B.5 (TEMKO *et al.*, 2014).

Este algoritmo foi escolhido para a implementação do PSA por já haver implementações dele disponíveis na internet, não sendo necessário recriá-lo para este trabalho mas somente adaptá-lo e reimplementá-lo, além de ser uma técnica robusta e que tem os parâmetros necessários para uma implementação eficiente neste trabalho.

4 PROCESSOS METODOLÓGICOS

O propósito deste trabalho consiste em testar a eficiência e precisão da modelagem de sistemas não-lineares utilizando a varredura exponencial. Para isto, será seguida uma cadeia de aquisição, análise, processamento e síntese, de forma a obter um modelo do Dispositivo sob Testes (DUT)[1] que será comparado ao mesmo, com o objetivo de verificar os resultados.

4.1 SISTEMAS E DISPOSITIVOS

Haverá dois tipos de DUT neste trabalho: digitais e analógicos. Para os sistemas digitais, todo o procedimento será realizado em um computador doméstico, fazendo uso de simulações utilizando a linguagem de programação Python. O DUT será simulado utilizando funções próprias da linguagem e suas bibliotecas, de forma a se ter controle sobre a sua operação, com o objetivo de poder realizar testes do funcionamento dos algoritmos necessários para os passos posteriores.

Para os sistemas analógicos, o sinal de entrada salvo como um arquivo no formato ".wav" será passado ao DUT através de cabo de áudio P2-P10, e adquirido dele através de cabo de áudio P10-P2, e gravado utilizando software Audacity® no mesmo formato de arquivo. A conversão entre sinais discretos e sinais contínuos é realizada por uma placa de som integrada do computador. Não-linearidades e distorções que a placa de som pode causar nas medições não foram consideradas pois como todos os sinais passam por ela, ela não causará erros ou diferenças nos resultados já que todos serão afetados igualmente por ela, ou considera-se que serão.

O objetivo dos sistemas analógicos é ter uma aplicação prática para a modelagem. Dos 5 DUT escolhidos para este trabalho, três são digitais e dois são analógicos. Ambos os DUT analógicos são pedais de distorção (NOCETI FILHO, 2021), dispositivos utilizados amplamente na área musical para aumentar o conteúdo harmônico da saída de um instrumento, alterando o seu timbre e, com isso, a maneira como ele soa.

4.1.1 Sistema A

O sistema A (S_A) será um sistema não-linear simulado, que gera uma quantidade finita de harmônicos, sem memória. O comportamento deste sistema é regido pela Equação 4.1. O objetivo da utilização dele é verificar o funcionamento das teorias expostas na Capítulo 2 utilizando de um sistema simples e conhecido.

$$y[n] = S_A\{x[n], y[n]\} = x[n] - 0,5\,x[n]^2 + 0,2\,x[n]^3 \tag{4.1}$$

[1] Do inglês, *Device Under Test*, é o sistema a ser modelado

4.1.2 Sistema B

O sistema B (S_B) é um sistema não-linear, também construido para que criasse uma quantidade finita de harmônicos, mas agora com a presença de memória, sendo seu comportamento determinado pela Equação 4.2, em que $f_1[n]$ e $f_2[n]$ são filtros digitais que tem o propósito de inserir o efeito de memória na resposta do sistema. O objetivo da utilização dele é ainda verificar a teoria porém agora com um sistema mais complexo, com o objetivo de verificar a capacidade dos algoritmos de capturar os efeitos de memória e reverberação.

$$y[n] = S_B\{x[n]\} = x[n] * f_1[n] + 0,3x[n]^3 * f_2[n] \tag{4.2}$$

Ambos os filtros $f_1[n]$ e $f_2[n]$ são filtros Butterworth (NOCETI FILHO, 2020) digitais desenvolvidos com a biblioteca SciPy em Python. O filtro $f_1[n]$ é um filtro passa-altas de ordem $N = 1$ e frequência de corte $f_c = 1$k Hz. O filtro $f_2[n]$ é um filtro passa-baixas de ordem $N = 1$ e $f_c = 100$ Hz. Os diagramas de Bode (OPPENHEIM *et al.*, 1996) dos filtros podem ser vistos nas Figuras B.1 e B.2. Em todas as figuras, as linhas cinzas verticais indicam a f_c do respectivo filtro. O diagrama de módulo da Figura B.1 apresenta um decaimento a mais no final dele em função do AAF passa-baixas.

4.1.3 Sistema C

O sistema C (S_C) é um sistema não-linear, sem memória, mas com uma quantidade infinita de harmônicos, sendo seu comportamento determinado pela Equação 4.3. O objetivo deste sistema é também verificar o funcionamento da teoria com um sistema complexo, porém agora ao invés de adicionar memória ao sistema, deseja-se analisar o efeito de uma quantidade infinita de respostas impulsivas, e os efeitos de utilizar-se somente uma quantidade finita destas, verificando a acurácia do modelo.

$$y[n] = S_C\{x[n]\} = \sqrt[3]{x[n]} \tag{4.3}$$

4.1.4 Sistema D

O sistema D (S_D) é o primeiro dos dois sistemas analógicos a serem testados. Ele é um pedal de distorção *Fuzz Face*, cujo esquemático encontra-se na Figura B.3. Para propósito deste trabalho, ele será tratado como um sistema *Black Box* (PARKER, 2020), desconsiderando quaisquer mecanismos do seu funcionamento interno e se preocupando somente com suas entrada e saída.

4.1.5 Sistema E

O sistema E S_E é um *MXR Distortion Plus*, também um pedal de distorção, que também será tratado como um sistema *Black Box*. O propósito dele neste trabalho, bem

como do sistema D, é verificar em conjunto as capacidades dos algoritmos desenvolvidos de modelarem um sistema que produz infinitas harmônicas e com memória[2]. Seu esquemático encontra-se na Figura B.4.

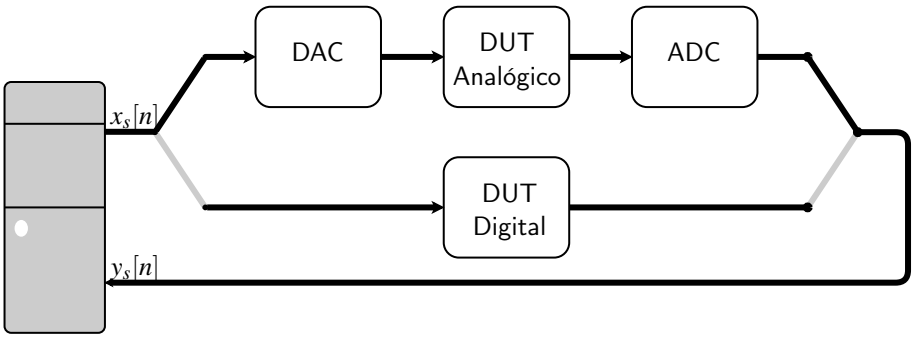
4.2 AQUISIÇÃO DE DADOS

Este processo consiste em excitar o sistema com um sinal conhecido, e observar a sua saída. O sinal a ser utilizado é uma varredura exponencial discreto $s[n]$, com parâmetros $f_1 = 20$ Hz, $f_2 = 20$k Hz, $T = 10$ s, com um $f_{\mathrm{sr}} = 44,1$k Hz. Além disso, uma sequência de zeros será adicionada ao final da varredura chamada *Stop Margin* (DIETRICH *et al.*, 2013), com o mesmo comprimento dele, de forma que haja espaço para a gravação capturar as possíveis reverberações do sistema e também ser possível capturar o ruído de fundo do sistema. Desta maneira, o sinal de entrada $x_s[n]$ dos sistemas a serem testados pode ser descrito pela Equação 4.4[3].

$$x_s[n] = \begin{cases} \sin\left(\dfrac{2\pi f_1 \cdot T}{\ln\left(\frac{f_2}{f_1}\right)} \cdot \left(\mathrm{e}^{\frac{n/f_{\mathrm{sr}}}{T}\cdot\ln\left(\frac{f_2}{f_1}\right)} - 1\right)\right) & n \leq \lfloor L_{\mathrm{s}}/2 \rfloor, \\ 0 & n > \lfloor L_{\mathrm{s}}/2 \rfloor, \\ n \in [0, L_{\mathrm{s}} - 1] \in \mathbb{N} \end{cases} \tag{4.4}$$

A Figura 4.1 mostra um diagrama da etapa de aquisição de dados, em que o chaveamento indica se o sistema for digital ou analógico.

Figura 4.1 – Diagrama de blocos da etapa de aquisição de dados.



Fonte – do autor.

4.3 PROCESSAMENTO DIGITAL DOS DADOS

A partir dos dados obtidos na etapa de aquisição, tem-se a etapa de processamento de dados. Tendo $y_s[n]$ a resposta do DUT ao sinal de excitação $x_s[n]$ (Equação 4.4), pode-se

[2] Ambos os pedais possuem uma quantidade infinita de harmônicos em função dos componentes utilizados para sua construção, como diodos e transistores; e possuem memória também pelos componentes, como capacitores, que causam efeito de memória

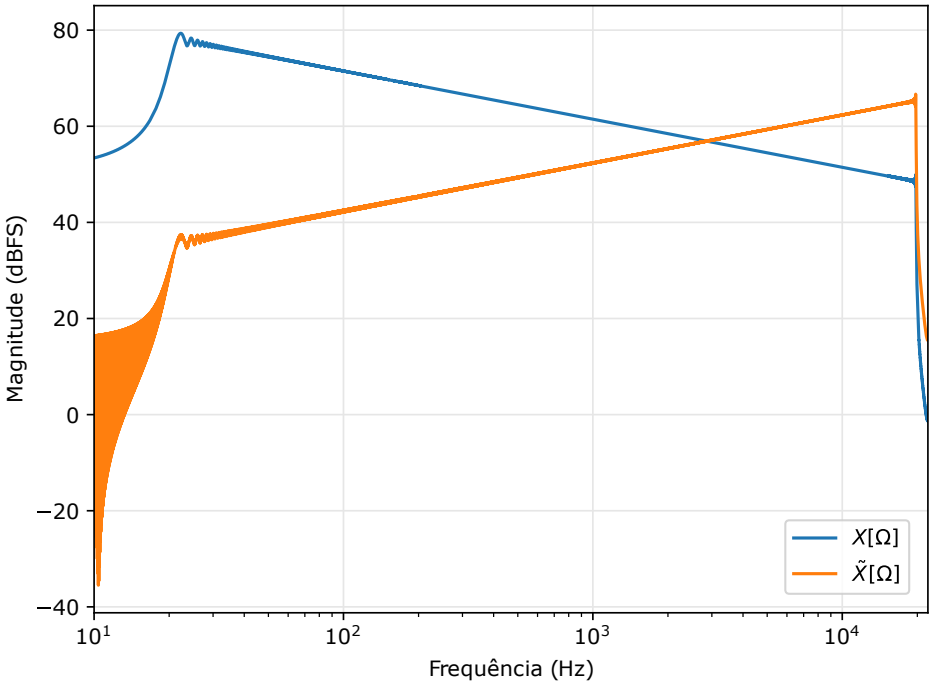
[3] $\lfloor\ \rfloor$ denota a operação de arredondamento para baixo

realizar a deconvolução entre os dois sinais de forma a IRRS do sistema, $\tilde{h}[n]$. Este processo será realizado conforme sugerido por Novák *et al.* (2010), em que ao invés de deconvoluir $y_s[n]$ com $x_s[n]$, realiza-se a convolução de $y_s[n]$ com $\tilde{x}_s[n]$, sinal este que possui o comportamento inverso na frequência em relação a $x_s[n]$, na faixa de interesse $[\omega_1, \omega_2]$. Este procedimento evita problemas como valores muito pequenos de $X[\omega]$ com ω fora da faixa de interesse. A formulação do sinal $\tilde{x}_s[n]$ em função de $x_s[n]$ se dá pela Equação 4.5.

$$\tilde{x}_s[n] = \begin{cases} \frac{f_1 \cdot \ln\left(\frac{f_2}{f_1}\right)}{T} \cdot \mathrm{e}^{\frac{-n/f_{\mathrm{sr}}}{T}} \cdot x_s[L_{\mathrm{s}}/2 - n] & n \leq \lfloor L_{\mathrm{s}}/2 \rfloor, \\ 0 & n > \lfloor L_{\mathrm{s}}/2 \rfloor, \\ n \in [0, L_{\mathrm{s}} - 1] \in \mathbb{N} \end{cases} \tag{4.5}$$

Os espectros de $x_s[n]$ e $\tilde{x}_s[n]$ que serão utilizados nos procedimentos podem ser vistos na Figura 4.2, em que nota-se que o comportamento deles ao longo da faixa de interesse é inverso, o que indica que eles são inversamente proporcionais ao longo dela, já que o gráfico é em escala logarítmica. Um diagrama de blocos da etapa de obtenção da IRRS $\tilde{h}[n]$ encontra-se na Figura 4.3.

Figura 4.2 – Módulo do espectro em frequência dos sinais $x_s[n]$ e $\tilde{x}_s[n]$, com parâmetros $f_1 = 20$ Hz, $f_2 = 20$k Hz, $T = 10$ s, com um $f_{\mathrm{sr}} = 44,1$k Hz.



Fonte – do autor.

4.3.1 Decomposição dos sinais

Após a convolução de $y_s[n]$ com $\tilde{x}_s[n]$, obtendo-se $\tilde{h}[n]$, é necessário obter cada HIR $\tilde{h}_k[n]$ do sistema, de forma a poder prosseguir com o algoritmo. Utilizando o fato de que a

DFT faz com que a convolução seja cíclica, é necessário determinar em que instantes de tempo discreto inicia-se cada resposta harmônica $\tilde{h}_k[n]$, e isto pode ser obtido a partir da Equação 3.5 e de f_{sr}.

Como ($f_{\mathrm{sr}} = \text{amostras/segundo}$), tem-se que ($\text{amostra} = \text{instante} \cdot f_{\mathrm{sr}}$). Com isso, obtém-se a Equação 4.6, em que a ordem máxima que será considerada é k_{max}, já que a partir de certo ponto os Δn_k ficam próximos demais para poder obter cada resposta separadamente.

$$\Delta n_k = \left\lfloor T \cdot \frac{\ln(k)}{\ln\left(\frac{\omega_2}{\omega_1}\right)} \cdot f_{\mathrm{sr}} \right\rfloor \tag{4.6}$$

Por se estar utilizando uma implementação digital dos processos, a origem do sinal $\tilde{h}[n]$ encontra-se não em $n = 0$, mas em $n = \lfloor L_{\mathrm{s}}/2 \rfloor$, e portanto esta será tratada como a origem quando se pensar em causalidade e conceitos semelhantes.

Sabendo que a primeira HIR em $\tilde{h}[n]$ ocorre em $n = \lfloor L_{\mathrm{s}}/2 \rfloor$[4], e o atraso de cada HIR em relação à primeira, consegue-se extraí-las até k_{max} e construir o sinal $\tilde{\mathbf{h}}[n]$.

No processo adotado, desloca-se o sinal $\tilde{h}[n]$ ($\lfloor L_{\mathrm{s}}/2 \rfloor - \Delta n_k$ amostras para a esquerda), de forma a posicionar o início da resposta impulsiva da k-ésima harmônica na origem. Após isso, aplica-se uma janela de forma a zerar os valores do sinal deslocado a partir do próximo impulso, e utiliza-se uma janela mista (SOARES, 2006) com um decaimento Hann de forma a suavizar a separação. Este tipo de decaimento foi escolhido por não causar muito vazamento e espalhamento na frequência (OPPENHEIM; SCHAFER; BUCK, 1999).

4.3.2 Processamento e algoritmos

Uma vez dispondo da resposta do sistema à varredura, busca-se construir uma modelagem do sistema original a partir da resposta $\tilde{\mathbf{h}}[n]$, e como discutido na Seção 3.4, essa resposta não pode ser prontamente utilizada. Serão testadas as três soluções propostas neste trabalho: a solução tradicional (FARINA, 2000); a solução proposta por Novák *et al.* (2010); e a solução proposta neste trabalho, baseada no PSA.

Para a solução tradicional, não é necessário nenhum preparo dos sinais de entrada ou de resposta do sistema, já que nela considera-se que $\tilde{\mathbf{h}}[n] = \mathbf{h}[n]$, e tem-se que as entradas da modelagem MISO do sistema Figura 3.3 são $\mathbf{x}[n] = [x[n], x^2[n], ...]$.

Para a solução proposta por Novák, é necessário realizar a conversão de $\tilde{\mathbf{h}}[n]$ para $\mathbf{h}[n]$ através da matriz A, conforme as Equações 3.12 até 3.14, e tem-se que o vetor de entrada $\mathbf{x}[n]$ é o mesmo da solução tradicional.

Para a solução proposta neste trabalho, tem-se que $\mathbf{h}[n] = \tilde{\mathbf{h}}[n]$, e é necessário implementar um algoritmo PSA para transformar $x[n]$ em $\mathbf{x}[n] = [x_1[n], x_2[n], ...]$ (Seção 3.4.3). O algoritmo utilizado foi originalmente implementado por Woodall (2015), e adaptado para este

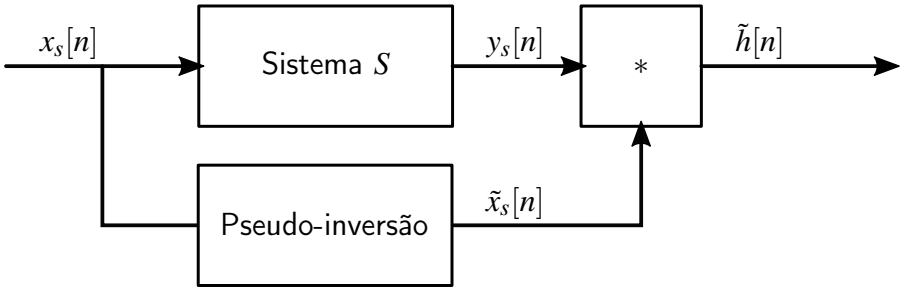
[4] No caso dos sistemas analógicos, isto não ocorre em função de os momentos de início de sinal e início de gravação não são sincronizados, mas pode ser trabalhado para que o primeiro impulso ocorra no instante desejado

trabalho. O funcionamento de cada uma das implementações e suas vantagens e desvantagens teóricas já foram expostos nas Seções 3.4.2 and 3.4.3.

4.3.3 Modelagem e síntese da saída

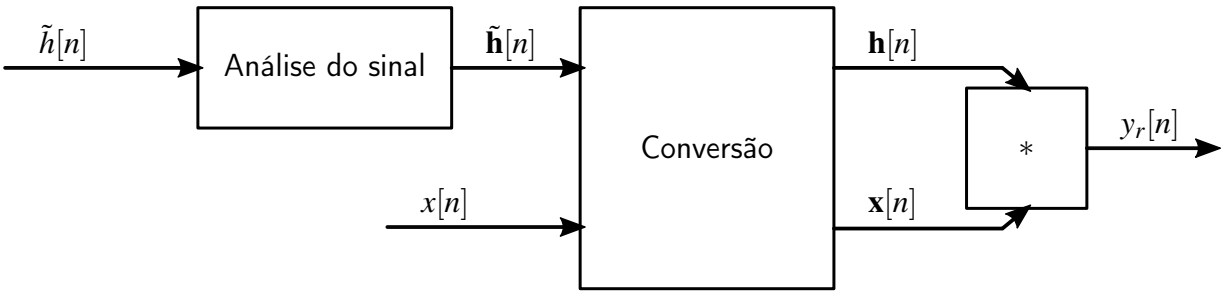
A partir de $\mathbf{h}[n]$ e um sinal de entrada $\mathbf{x}[n]$ qualquer que já tenha sido preparado para o algoritmo escolhido, é possível realizar a etapa de síntese do sinal de saída e simular o comportamento do sistema caracterizado por $\mathbf{h}[n]$ sem ter acesso a ele, por meio da Equação 2.11. O diagrama de blocos das etapas de processamento, conversão e síntese encontra-se na Figura 4.4.

Figura 4.3 – Diagrama de blocos da etapa de obtenção da resposta impulsiva referente à varredura $\tilde{h}[n]$.



Fonte – do autor.

Figura 4.4 – Diagrama de blocos da etapa de análise, conversão e síntese.



Fonte – do autor.

5 DISCUSSÃO DE RESULTADOS

Neste capítulo serão apresentados e comentados os resultados obtidos a partir da metodologia descrita no Capítulo 4, discutindo-os e avaliando suas implicações e conclusões, guiando-se pelos objetivos do trabalho expostos na Seção 1.1.2.

5.1 DECONVOLUÇÃO E ANÁLISE

Como explicado na Seção 4.3, o resultado da deconvolução do sinal de saída $y_s[n]$ com o sweep inverso $\tilde{x}_s[n]$ gera a IRRS $\tilde{h}[n]$. Estas respostas são mostradas na Figura 5.1, sendo que cada subfigura indica o sinal $\tilde{h}[n]$ de um sistema diferente. Nelas, é possível notar a separação de cada HIR no tempo, sendo que estas estão atrasadas da resposta impulsiva da harmônica fundamental, como era esperado de se conseguir. Embora a duração de todas as IRRS $\tilde{h}[n]$ seja de 40 s, o eixo do tempo foi limitado a 25 segundos para melhorar a visualização já que nenhum dos sistemas possui informação além dos 25 s.

Além disso, foi necessário obter elas como sinais separados que compunham o sinal $\tilde{\mathbf{h}}[n]$. A Figura 5.2 mostra o módulo dos espectros de cada HIR de $\tilde{\mathbf{h}}[n]$ de cada um dos sistemas, até $k_{max} = 5$, exceto pelos sistemas A e B que possuem uma quantidade finita de harmônicos. Em todos os gráficos na frequência a escala de amplitude é em dBFS[1]. Também, em todos os sistemas espera-se que não haja informação das HIR abaixo de 20 Hz, já que a varredura exponencial não estaria excitando o sistema nesta região da frequência.

No sistema A (Figura 5.2a), nota-se que a resposta impulsiva da 1ª harmônica possui resposta plana ao longo de praticamente toda a faixa de interesse, exceto por alguns artefatos que surgiram em baixas e altas frequências. A 3ª harmônica também possui uma resposta plana a partir de 60 Hz, que seria esperado já que a varredura inicia em 20 Hz e o sistema não conseguiria reproduzir informação sobre a 3ª harmônica nesta faixa já que não houve excitação dele. Também, é notável a presença de alguns artefatos, especialmente em altas frequência. Uma hipótese levantada que pode ser causa disso são efeitos de intermodulação ou recobrimento, após potências dos sinais serem tomadas. Estes artefatos também surgem nas respostas dos outros sistemas.

Contudo, é possível ver que a 2ª harmônica não apresenta uma resposta plana sendo que isto era esperado. Levando em conta a função que caracteriza o sistema (Equação 4.1), a resposta ao impulso esperada de cada uma das HIR seria também um impulso $\delta[n]$. Este efeito é causado por cada uma das respostas impulsivas não ser uma função causal, e ter um "espalhamento" anticausal que também carrega informação sobre ela que se perde ao considerá-la como começando no instante Δn_k. Atrasar o início da HIR faria com que seu espectro fosse mais plano, mas afastaria o resultado da modelagem do resultado verdadeiro já que isso implicaria em um atraso artificial na saída daquela harmônica.

[1] Do inglês, *Decibels relative to Full Scale*

No sistema B (Figura 5.2b), nota-se que as respostas possuem um comportamento semelhante ao esperado, que seria as Figuras B.1 e B.2 que representam os filtros aplicados sobre as 1ª e 3ª potências do sinal. Contudo, há um resquício de f_2, que é a resposta da 3ª harmônica, sobre a 1ª harmônica. Isto se deve ao efeito explicitado por Novák, em função das potências do cosseno, e é esperado que a solução proposta por ele seja capaz de reverter esta distorção indesejada.

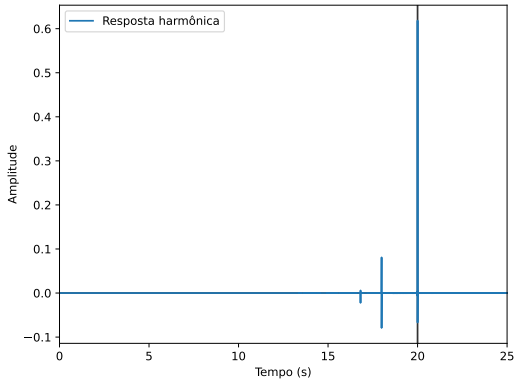
No sistema C (Figura 5.2c), nota-se que somente os harmônicos de 1ª, 3ª e 5ª ordem são relevantes, o que condiz com o esperado por a função $\sqrt[3]{x}$ ser uma função ímpar. Nota-se, porém, que os harmônicos de 2ª e 4ª ordem são vísiveis, e isto se deve a ruídos numéricos e erros da modelagem, possivelmente tendo sido causados pelos começo e fim da varredura. Os picos que são visíveis em ambos são devidos à frequência inicial ω_1 do sinal de excitação, que gera harmônicos de todas as ordens pelo fato do sistema ainda ter as condições iniciais dele agindo.

Os sistemas D e E (Figuras 5.2d e 5.2e) tem um comportamento na frequência desconhecido, e portanto analisar se os resultados são coerentes com o esperado é difícil. Porém, é relevante perceber a presença de picos e antipicos em ambos. Isto possivelmente se deve às ressonâncias que ocorrem dentro do circuito de cada um dos DUT, que poderia causar esses picos na resposta em frequência e estes ocorreriam ao longo das diversas harmônicas.

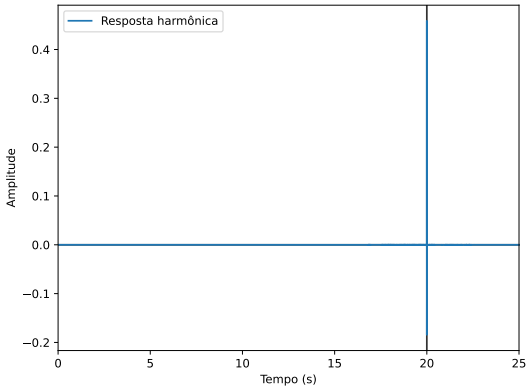
A Figura 5.3 mostra o módulo dos espectros de cada resposta impulsiva $\mathbf{h}[n]$, obtidas a partir de $\tilde{\mathbf{h}}[n]$ e da matriz A (Equação 3.12) para o método de Novák, para todos os DUT. Os sistemas mais relevantes são os sistemas B (Figura 5.3b) e D (Figura 5.3d).

No sistema B, nota-se que mesmo utilizando a matriz A não foi possível separar completamente o comportamento de cada um dos filtros aplicados, sendo ainda prevalente um pouco do filtro do harmônico de 3º ordem sobre o harmônico de 1ª. No sistema D, nota-se que a maioria dos picos teve sua amplitude reduzida drasticamente, sendo relevante ainda somente o antipico em 1k Hz no harmônico de 5ª ordem.

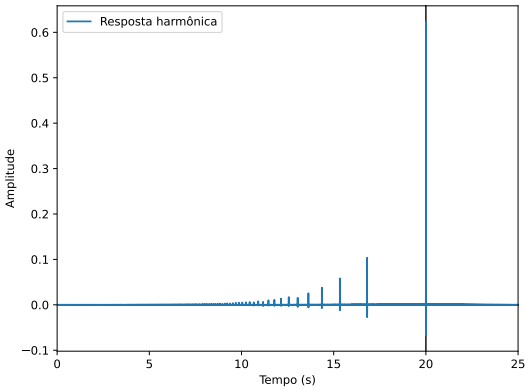
Figura 5.1 – Respostas impulsivas à varredura $\tilde{h}[n]$ dos sistemas.



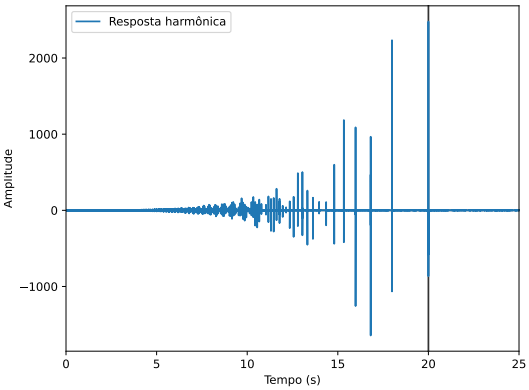
(a) Resposta impulsiva à varredura do sistema A



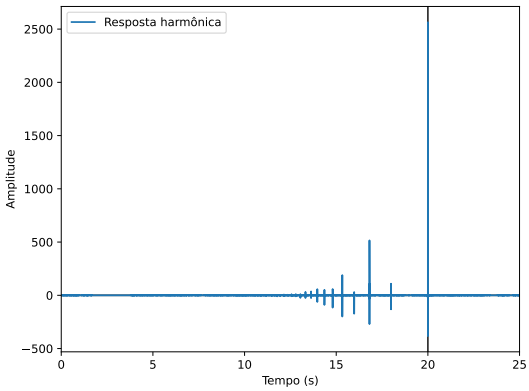
(b) Resposta impulsiva à varredura do sistema B



(c) Resposta impulsiva à varredura do sistema C



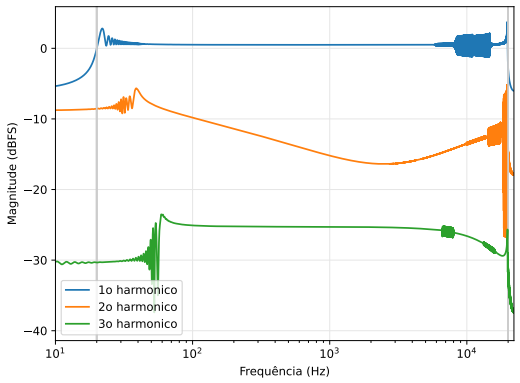
(d) Resposta impulsiva à varredura do sistema D



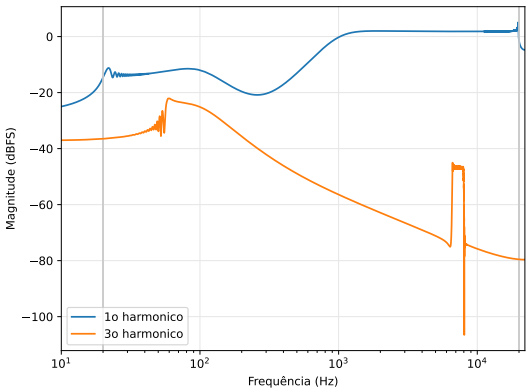
(e) Resposta impulsiva à varredura do sistema E

Fonte – do autor.

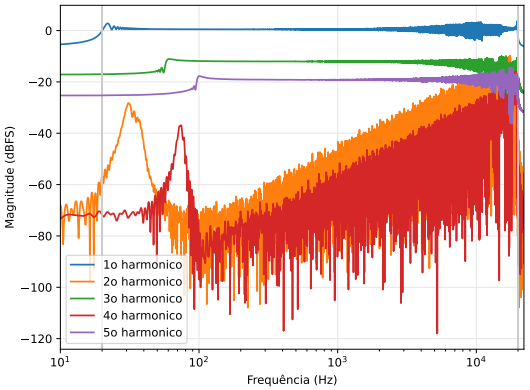
Figura 5.2 – Módulo dos espectros de $\tilde{\mathbf{h}}[n]$ dos sistemas até $k_{max} = 5$.



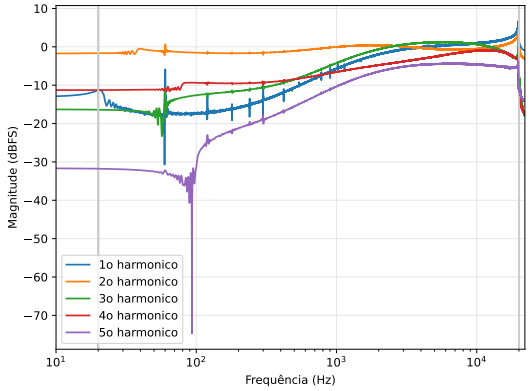
(a) Módulo dos espectros de $\tilde{\mathbf{h}}[n]$ do sistema A



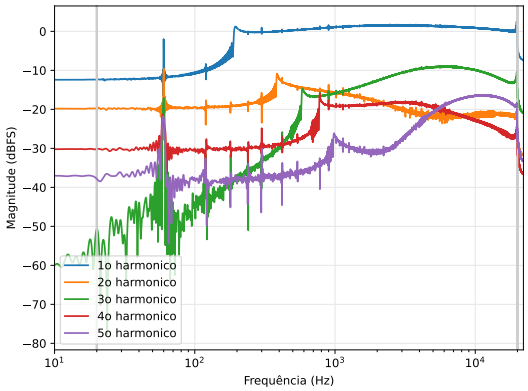
(b) Módulo dos espectros de $\tilde{\mathbf{h}}[n]$ do sistema B



(c) Módulo dos espectros de $\tilde{\mathbf{h}}[n]$ do sistema C



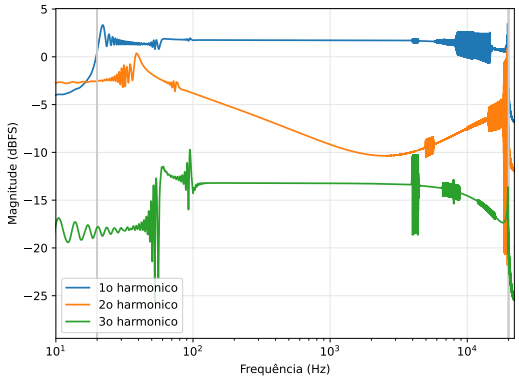
(d) Módulo dos espectros de $\tilde{\mathbf{h}}[n]$ do sistema D



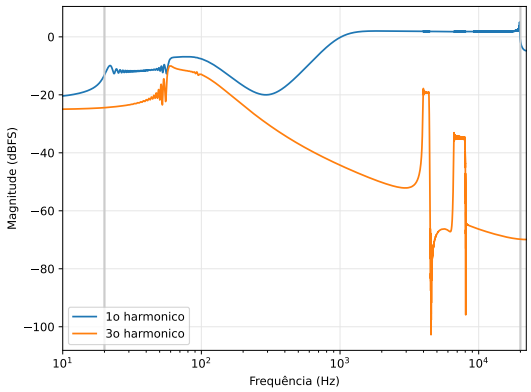
(e) Módulo dos espectros de $\tilde{\mathbf{h}}[n]$ do sistema E

Fonte – do autor.

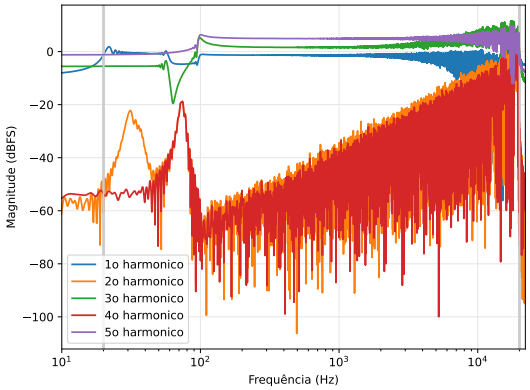
Figura 5.3 – Módulo dos espectros de $\mathbf{h}[n]$ dos sistemas até $k_{max} = 5$ para a solução de Novák.



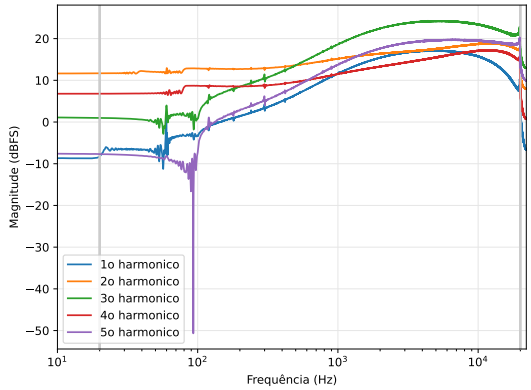
(a) Módulo dos espectros de $\mathbf{h}[n]$ do sistema A



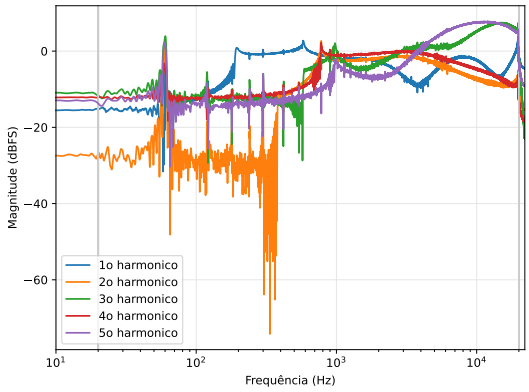
(b) Módulo dos espectros de $\mathbf{h}[n]$ do sistema B



(c) Módulo dos espectros de $\mathbf{h}[n]$ do sistema C



(d) Módulo dos espectros de $\mathbf{h}[n]$ do sistema D



(e) Módulo dos espectros de $\mathbf{h}[n]$ do sistema E

Fonte – do autor.

5.2 SÍNTESE DA SAÍDA

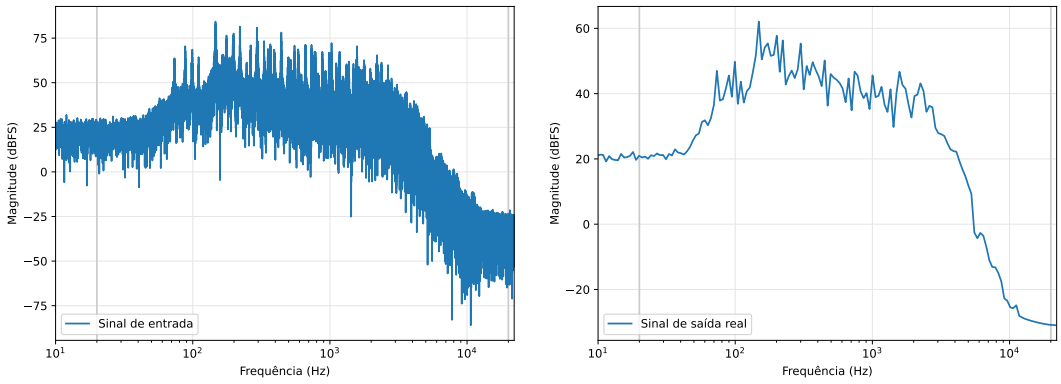
Para a etapa de síntese e modelagem, um trecho de um sinal de áudio foi utilizado como sinal de entrada $x[n]$ (YOUTUBE, 2018). O propósito de se utilizar um sinal de áudio musical é a possibilidade de aproveitar os mesmos resultados obtidos neste trabalho para possíveis pesquisas futuras.

O espectro do sinal $x[n]$ encontra-se na Figura 5.4a. Como ele é um sinal extremamente complexo e de difícil visualização por ter grandes variações de magnitude, foi realizada uma subamostragem em seu espectro de forma que se perde um pouco da precisão e dos detalhes, mas ganha-se visibilidade e compreensibilidade. Além disso, o pico da magnitude do sinal subamostrado foi normalizada em 0 dBFS. O espectro subamostrado consta na Figura 5.4b.

Este sinal foi escolhido por ser um áudio "limpo" de uma guitarra, sem distorções ou efeitos já aplicados, de forma que viria a ser possível utilizar os mesmos resultados em outros trabalhos futuros que poderiam depender disso.

Os espectros dos sinais de saída reconstruídos junto do sinal de saída verdadeiro, para cada sistema, encontram-se na Figura 5.5. Da mesma maneira que foi implementado para o sinal $x[n]$, os espectros de $y[n]$ são todos apresentados com a subamostragem, de forma a facilitar a visualização.

Figura 5.4 – Módulo do espectro do sinal de entrada teste $x[n]$

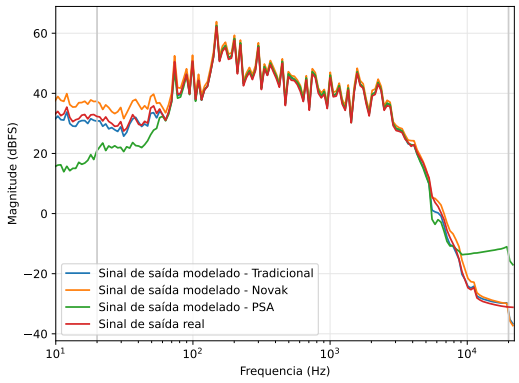


(a) Módulo do espectro completo do sinal $x[n]$

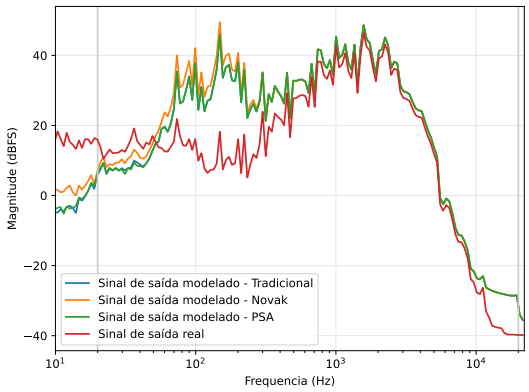
(b) Módulo do espectro subamostrado do sinal $x[n]$

Fonte – do autor.

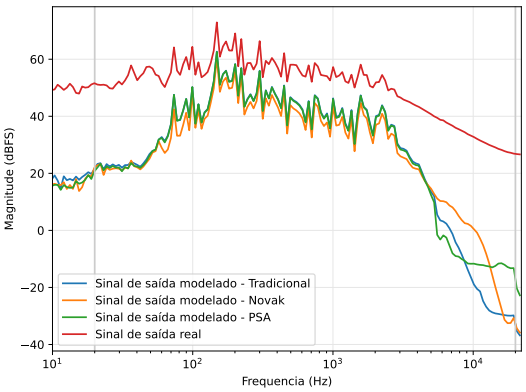
Figura 5.5 – Módulo do espectro das saídas reais e modeladas dos sistemas - $k_{max} = 5$.



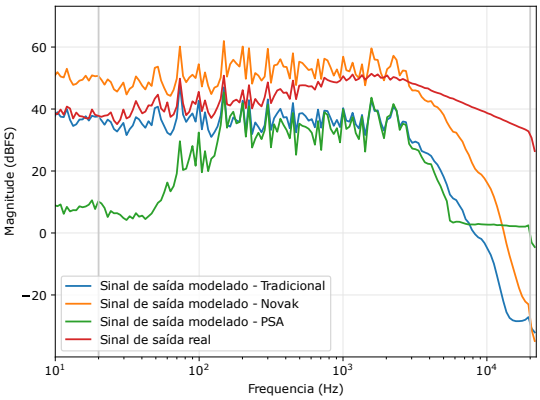
(a) Módulo do espectro das saídas verdadeira e modeladas do sistema A - $k_{max} = 5$



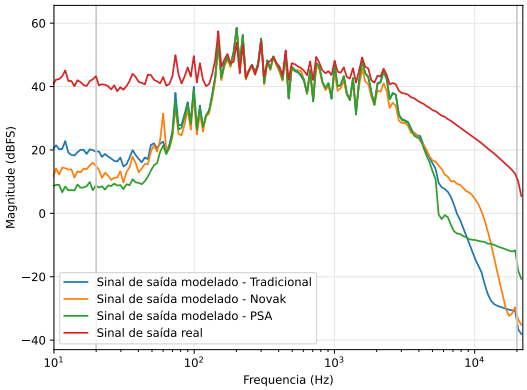
(b) Módulo do espectro das saídas verdadeira e modeladas do sistema B - $k_{max} = 5$



(c) Módulo do espectro das saídas verdadeira e modeladas do sistema C - $k_{max} = 5$



(d) Módulo do espectro das saídas verdadeira e modeladas do sistema D - $k_{max} = 5$



(e) Módulo do espectro das saídas verdadeira e modeladas do sistema E - $k_{max} = 5$

Fonte – do autor.

Em todos os sistemas, nota-se uma diferença significativa entre as saídas modeladas e a saída verdadeira, abaixo de 100 Hz. A primeira explicação possível é que isto ocorre pela mesma razão de a k-ésima HIR não possuir informação até $k \cdot 20$ Hz, e portanto a parte inferior

do espectro não foi completamente excitada de forma a capturar em detalhes as harmônicas geradas, o que poderia explicar a discrepância. Outra explicação possível é a presença de ruído em baixas frequências no sinal verdadeira que não foi capturado ao longo da modelagem, o que explicaria também esta diferença. Contudo esse não é o caso, já que para os sistemas A, B e C o seu comportamento é simulado, e portanto não há presença de ruído; e para os sistema D e E, suas Razão Sinal-Ruído (SNR)[2] são de 77,5 dB e 66,9 dB, respectivamente.

Nota-se que no sistema A, todas as 3 modelagens mostraram resultados extremamente satisfatórios na faixa de 100-5k Hz, sendo seus espectros praticamente idênticos ao da saída verdadeira. A que melhor se aproximou acima de 5k Hz foi a modelagem de Novák, e ainda nesta faixa a diferença foi pequena, sendo relevante notar que acima de 10k Hz a modelagem através do PSA apresentou uma subida novamente.

Para o sistema B, é perceptível que enquanto que acima de 1k Hz as 3 modelagens tiveram resultados semelhantes ao do sistema A, na faixa de 10-1k Hz todas elas mostraram-se incapazes de capturar o comportamento original do sistema. Voltando-se à definição do sistema B na Seção 4.1.2, tinha-se que ele era composto por um filtro passa-baixas em 100 Hz, e um filtro passa-altas em 1k Hz, o que indica que há alguma correlação entre os filtros escolhidos e a faixa onde as modelagens não foram eficázes. Olhando as Figuras 5.2a e 5.3a é possível perceber que não foi possível separar os dois filtros nas HIR, que é a razão de haver este erro nas modelagens.

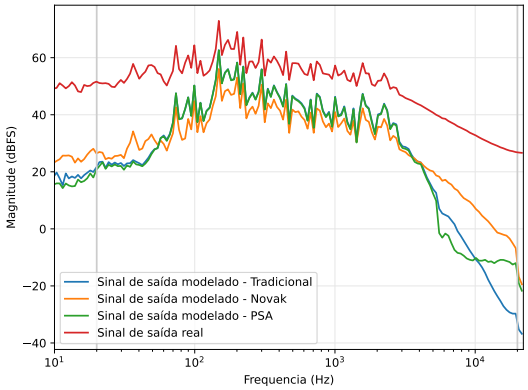
No sistema C, as 3 modelagens trouxeram resultados muito semelhantes ao longo de toda a faixa de frequências, mas nenhuma delas foi capaz de capturar corretamente as informações de baixa frequência, e nem seguir os detalhes do contorno ao longo de todo o espectro, pela mesma razão do sistema A. Agora, também, tem-se que enquanto os espectros das modelagens se aproximaram do espectro verdadeiro, nenhum deles chegou muito próximo, sempre tendo diferenças notáveis graficamente. Este mesmo erro ocorre para os sistemas D e E, o que traz a suspeita de que talvez esta diferença seja devida ao fato de se estar utilizando uma quantidade extremamente limitada de harmônicos do sistema, já que para estes 3 sistemas há uma quantidade infinita de harmônicos, e considerou-se somente até $k_{max} = 5$.

A Figura 5.6 mostra o módulo dos espectros dos sinais de saída reconstruídos e do sinal de saída verdadeira para os sistemas C, D e E da mesma forma que Figuras 5.5c até 5.5e, porém agora com $k_{max} = 15$. No sistema C, nota-se que o resultado da modelagem de Novák aproxima-se do esperado, o que indica que com uma quantidade ainda maior de harmônicos seria possível alcançar um resultado suficientemente satisfatório, porém aumentar o número de harmônicos utilizados de maneira incauta faz com que o sistema modelado seja mais sensível ao ruído, já que a magnitude da IRRS $\tilde{h}[n]$ tende a diminuir conforme aumenta-se o número de harmônicos e aproximar-se de um ruído de fundo e também as respostas ao impulso em $\tilde{h}[n]$ aproximam-se no tempo, diminuindo a janela temporal entre elas e a informação que cada uma carrega.

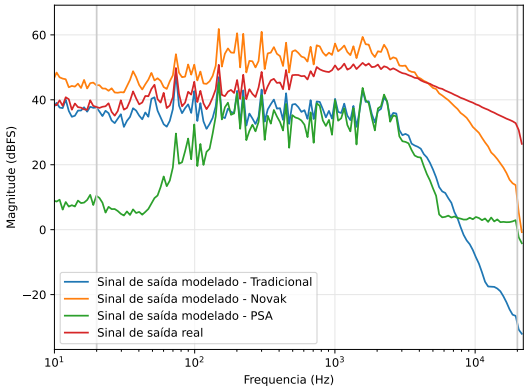
[2] Do inglês, *Signal-to-Noise Ratio*

Nos sistemas D e E, nota-se que o espectro da modelagem de Novák aproxima-se do esperado porém ainda não tanto quanto nos sistemas A e C, o que indica que há muita informação em harmônicos mais altos não-capturada pela modelagem. Este resultado é coerente com o que foi obtido anteriormente nos experimentos, já que ao se olhar as Figuras 5.1c até 5.1e nota-se que os sistemas D e E são os que mais possuem informação em altas frequências, em especial o sistema E, então é esperado que seria necessário considerar mais harmônicas.

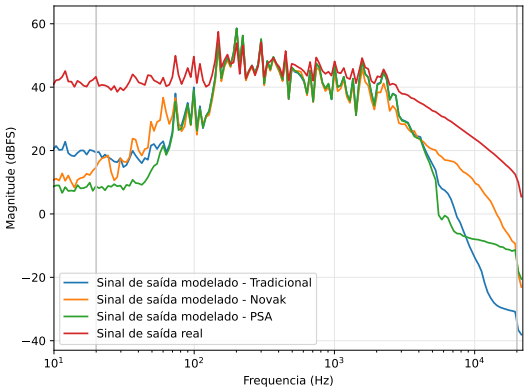
Figura 5.6 – Módulo do espectro das saídas reais e modeladas dos sistemas - $k_{max} = 15$.



(a) Módulo do espectro das saídas verdadeira e modeladas do sistema C - $k_{max} = 15$



(b) Módulo do espectro das saídas verdadeira e modeladas do sistema D - $k_{max} = 15$



(c) Módulo do espectro das saídas verdadeira e modeladas do sistema E - $k_{max} = 15$

Fonte – do autor.

5.3 COMPARAÇÃO ENTRE MODELAGENS

A partir dos resultados anteriores, é possível determinar que o método que melhor modelou os sistemas de forma geral foi o método de Novák, através da matriz de conversão. O problema de intermodulação, explicitado na Seção 3.4.2 não se mostrou impactante no resultado final dos experimentos, ou pelo menos foi menos relevante que outros fatores como a utilização de uma quantidade insuficiente de harmônicos para a modelagem, ou o surgimento de harmônicos nas potências do cosseno, que era o efeito que Novák propunha resolver.

Nota-se que, de forma geral, os resultados do método tradicional e o utilizando o PSA foram pouco afetados pelo aumento de k_{max}, o que era esperado pela sobreposição de harmônicos, enquanto o método de Novák melhorou ao se incluir mais harmônicos. Além disso, o método com PSA foi o que menos se aproximou da resposta verdadeira, o que reforça que o problema de intermodulação, que ele foi imaginado para contornar, não é relevante o suficiente para que sua utilização se justifique, e que o problema de cruzamento de harmônicos é mais impactante.

Além disso, há também o problema visto no sistema B, em que nem mesmo o método de Novák conseguiu separar completamente as respostas dos dois filtros e a modelagem distanciou-se notavelmente do resultado esperado. Contudo, este efeito é de pouca relevância prática já que na maioria dos casos reais o comportamento de cada HIR terá ao menos um comportamento semelhante na frequência como pode ser visto nas Figuras 5.2d e 5.2e, e não será tão drástico quanto o que foi imposto sobre o sistema B.

6 CONCLUSÕES E CONSIDERAÇÕES

Neste trabalho, tinha-se por propósito testar diferentes métodos (tradicional, de Novák e com PSA) para modelagem de sistemas não-lineares a partir da sua resposta à varredura exponencial de forma a verificar seus efeitos e verificar se os problemas teóricos pensados seriam relevantes quando aplicados à prática.

Dentre os métodos utilizados foi proposto um método novo através um algoritmo PSA, com o objetivo de contornar um problema de intermodulação que poderia surgir nos outros métodos pensados. O método que foi mais eficaz em modelar os sistemas utilizados foi o método proposto por Novák *et al.* (2010), que se pensava que poderia ter problemas em função do efeito de intermodulação, mas este não mostrou-se relevante.

Dos objetivos expostos no início do trabalho, foi possível alcançar todos com sucesso, verificando o funcionamento da varredura exponencial para obtenção das respostas harmônicas e as capacidades dos diferentes métodos de modelagem para simular um sistema não-linear a partir das respostas harmônicas.

Embora as modelagens não tenham sido perfeitas para todos os sistemas, foram alcançados resultados relevantes, além de ter sido possível visualizar os efeitos que a modelagem com diferentes quantidades de harmônicas exerce sobre o resultado final, e quão mais próximo os resultados chegam ao se incluir um número maior de harmônicos na modelagem.

6.1 SUGESTÕES DE TRABALHOS FUTUROS

Como notou-se nas Figuras 5.5 e 5.6, um aumento na quantidade de harmônicos considerados resultou em uma melhora na modelagem do sistema, porém espera-se que este resultado tenha um limite devido ao ruído presente e na aproximação dos Δn_k. Dessa maneira, encontrar um algoritmo que seja capaz de encontrar o valor ideal de k_{max} é interessante, do ponto de vista de pesquisa.

Outro estudo a ser realizado futuramente é um trabalho prático, em que se deseja avaliar psicoacusticamente os métodos utilizados, já que verificou-se somente por meio de dados e espectros quão próximas ficaram as respostas modeladas em relação à real, e é interessante fazer testes com indivíduos para averiguar se a metodologia analisada é aplicável também a sistemas reais, em especial sistemas musicais como os pedais que foram utilizados como DUT neste trabalho, para sua modelagem e simulação digitais.

Um último trabalho a se pensar para o futuro, voltado a uma perspectiva teórica, é analisar se há relação entre a amplitude do sweep utilizado para caracterizar o sistema e a amplitude das harmônicas obtidas. Além disso, pode-se pensar em estudar a relação entre as respostas harmônicas do sistema que compunham $\mathbf{h}[n]$ e a Distorção Harmônica Total (THD)[1] (RAZAVI, 2010) de um sinal que saia deste, de forma a verificar se há alguma correlação entre ambas e se é possível definir um valor que caracterize a distorção de um sistema.

[1] Do inglês, *Total Harmonic Distortion*

REFERÊNCIAS

AUTOR, Sem. **Anti-aliasing filter (Inglês) [Filtro anti-recobrimento]**. [*S.l.*: *s.n.*]. Disponível em: https://en.wikipedia.org/wiki/Anti-aliasing_filter. Acesso em: 23 ago. 2021.

AUTOR, Sem. **Binomial coefficient (Inglês) [Coeficiente binomial]**. [*S.l.*: *s.n.*]. Disponível em: https://en.wikipedia.org/wiki/Binomial_coefficient. Acesso em: 21 ago. 2021.

AUTOR, Sem. **Sistema dinâmico não linear**. [*S.l.*: *s.n.*]. Disponível em: https://pt.wikipedia.org/wiki/Sistema_din%C3%A2mico_n%C3%A3o_linear. Acesso em: 20 ago. 2021.

AUTOR, Sem. **Trigonometric Power Formulas (Inglês) [Fórmulas de Potências Trigonométricas]**. [*S.l.*: *s.n.*]. Disponível em: https://mathworld.wolfram.com/TrigonometricPowerFormulas.html. Acesso em: 29 ago. 2021.

AUTOR, Sem. **What Is Reverberation of Sound? (Inglês) [O Que É Reverberação do Som?]** [*S.l.*: *s.n.*], 2021e. Disponível em: https://www.soundproofcow.com/reverberation/. Acesso em: 23 ago. 2021.

BARCELLOS, João Cláudio Elsen. **Montagem de um Fuzz Face e um MXR Distortion Plus**. [*S.l.*: *s.n.*], 2021.

BONCELET, Charles. Chapter 7 - Image Noise Models (Inglês) [Capítulo 7 - Modelos para Ruído em Imagem]. *In:* BOVIK, Al (Ed.). **The Essential Guide to Image Processing (Inglês) [O Guia Essencial para Processamento de Imagens]**. Boston: Academic Press, 2009. P. 143–167. Disponível em: https://www.sciencedirect.com/science/article/pii/B978012374457900007X. Acesso em: 21 ago. 2021.

BOYD, Stephen Poythress. **Volterra Series: Engineering Fundamentals (Inglês) [Série de Volterra: Fundamentos para Engenharia]**. 1980. PhD em Engenharia – University of California. Disponível em: https://web.stanford.edu/~boyd/papers/pdf/boyd_phd_thesis.pdf. Acesso em: 4 mai. 2021.

CURTARELLI, Vitor Probst. **Análise de Sinais e Sistemas Lineares**. [*S.l.*: *s.n.*], 2020.

DIETRICH, Pascal; MASIERO, Bruno; VORLANDER, Michael. **On the Optimization of the Multiple Exponential Sweep Method (Inglês) [Sobre a Otimização do Método da Varredura de Múltiplo Exponencial]**. [*S.l.*: *s.n.*], 2013. Disponível em: https://www.researchgate.net/publication/236024151_On_the_Optimization_of_the_Multiple_Exponential_Sweep_Method. Acesso em: 12 set. 2021.

FARINA, Angelo. Simultaneous Measurement of Impulse Response and Distortion with a Swept-Sine Technique. (Inglês) [Medição Simultânea de Resposta Impulsiva e Distorção com uma Técnica de Seno-Varrido]. **Audio Engineering Society**, 2000. Disponível em: https://www.researchgate.net/publication/2456363_Simultaneous_Measurement_of_Impulse_Response_and_Distortion_With_a_Swept-Sine_Technique. Acesso em: 16 ago. 2021.

HÉLIE, Thomas. **Introduction to Volterra series and applications to physical audio signal processing (Inglês) [Introdução à série de Volterra e aplicações ao processamento de sinais de áudio físicos]**. [*S.l.*: *s.n.*], 2015. Disponível em: http://s3.ircam.fr/wp-content/uploads/2015/10/DAFx_Tutorial_Helie.pdf. Acesso em: 4 mai. 2021.

KRARTI, Moncef. **Optimal Design and Retrofit of Energy Efficient Buildings, Communities, and Urban Centers (Inglês) [Design Ótimo e Retroajuste de Construções, Comunidades e Centros Urbanos Energeticamente Eficientes]**. [*S.l.*]: Butterworth-Heinemann, 2018. P. 189–245. Disponível em: https://www.sciencedirect.com/science/article/pii/B9780128498699000041. Acesso em: 12 set. 2021.

LATHI, B. P. **Sinais e Sistemas Lineares**. [*S.l.*]: Bookman, 2007.

MALTA, Iaci P.; SALDANHA, Nicolau C.; TOMEI, Carlos. **Geometria e análise numérica dde funções do plano no plano**. [*S.l.*: *s.n.*], 1993. Disponível em: https://impa.br/wp-content/uploads/2017/04/19_CBM_93_02.pdf. Acesso em: 28 set. 2021.

MATH VAULT. **Modulo**. [*S.l.*: *s.n.*]. Disponível em: https://mathvault.ca/math-glossary/. Acesso em: 21 ago. 2021.

MÜLLER, Swen; MASSARANI, Paulo. Transfer-Function Measurement with Sweeps (Inglês) [Medição de Funções de Transferência com Varreduras], 2008. Disponível em: https://www.aes.org/e-lib/browse.cfm?elib=10189. Acesso em: 4 mai. 2021.

NOCETI FILHO, Sidnei. **EFEITOS DE ÁUDIO DIGITAIS E ANALÓGICOS - PARTE 3**. [*S.l.*: *s.n.*]. Disponível em: http://www.linse.ufsc.br/~sidnei/filtros-efeito-audio.html. Acesso em: 23 ago. 2021.

NOCETI FILHO, Sidnei. **Filtros Seletores de Sinais**. [*S.l.*]: Editora da UFSC, 2020. v. 4.

NOVÁK, Antonín; SIMON, Laurent; KADLEC, František; LOTTON, Pierrick. Nonlinear System Identification Using Exponential Swept-Sine Signal (Inglês) [Identificação de Sistemas Não-Lineares usando Sinal Seno-Varrido Exponencialmente]. **IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT**, 2010. Disponível em: https://ieeexplore.ieee.org/document/5299278. Acesso em: 4 mai. 2021.

OPPENHEIM, Alan V.; SCHAFER, Ronald W.; BUCK, John R. **Discrete-time signal processing (Inglês) [Processamento digital de tempo discreto]**. [*S.l.*: *s.n.*], 1999. Disponível em: https://d1.amobbs.com/bbs_upload782111/files_24/ourdev_523225.pdf. Acesso em: 21 ago. 2021.

OPPENHEIM, Alan V.; WILLSKY, Alan S.; NAWAB, S. Hamid. **Sinais e Sistemas (2nd Ed.)** [*S.l.*]: Pearson, 1996. Disponível em: https://www.academia.edu/38640412/Alan_V_Oppenheim_Alan_S_Willsky. Acesso em: 21 ago. 2021.

PARKER, Clara. **A General Black Box Theory (Inglês) [Uma Teoria Geral de Caixa Preta]**. [*S.l.*: *s.n.*], 2020. Disponível em: https://www.journals.uchicago.edu/doi/10.1086/287954. Acesso em: 23 ago. 2021.

PUJOL, Rémy; TRIGUEIROS-CUNHA, Nuno. **Campo Auditivo Humano**. [*S.l.*: *s.n.*], 2018. Disponível em: http://www.cochlea.org/po/som/campo-auditivo-humano. Acesso em: 17 ago. 2021.

RAZAVI, Behzad. **Fundamentos de Microeletrônica**. [*S.l.*]: LTC, 2010. Disponível em: https://www.academia.edu/17663639/B_Razavi_Fundamentos_de_Microeletr%C3%B4nica. Acesso em: 3 set. 2021.

ROYER, Théo. Pitch-shifting algorithm design and applications in music (Inglês) [Design de algoritmo de mudança de tom e aplicações na música], 2019. Disponível em: http://kth.diva-portal.org/smash/get/diva2:1381398/FULLTEXT01.pdf. Acesso em: 2 ago. 2021.

RUMSEY, Francis; MCCORMICK, Tim. **Sound and Recording: An Introduction (Inglês) [Som e Gravação: Uma Introdução]**. [*S.l.*: *s.n.*], 2006. P. 538.

SCHNEIDER, Cynthia. **De Moivre's Theorem: A Literature and Curriculum Project on Roots, Powers, and Complex Numbers. (Inglês) [Teorema de De Moivre: Um Projeto de Currículo e Literatura sobre Raízes, Potências, e Números Complexos]**. 2011. Masters of Science in Teaching Mathematics – Portland State University. Disponível em: http://web.pdx.edu/~caughman/Cindy%5C%20501%5C%20Final.pdf. Acesso em: 16 ago. 2021.

SHARMA, Pulkit. **4 Types of Distance Metrics in Machine Learning (Inglês) [4 Tipos de Métricas de Distância em Aprendizado de Máquina]**. [*S.l.*: *s.n.*], 2020. Disponível em: https://www.analyticsvidhya.com/blog/2020/02/4-types-of-distance-metrics-in-machine-learning/. Acesso em: 28 set. 2021.

SOARES, Zemar Martins Defilippo. **Calibração de microfones com resposta impulsiva**. 2006. Doutor em engenharia mecânica – Universidade Federal de Santa Catarina.

Disponível em: http://repositorio.ufsc.br/xmlui/handle/123456789/88630. Acesso em: 29 ago. 2021.

STEWART, James. **Cálculo**. [*S.l.*]: CENGAGE Learning, 2008. v. 1.

TEMKO, Andriy; MARNANE, William; BOYLAN, Geraldine; O' TOOLE, John; LIGHTBODY, Gordon. Neonatal EEG audification for seizure detection (Inglês) [Audificação de EEG neonatal para detecção de convulsão]. **2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014**, v. 2014, ago. 2014.

VIALI, Lorí. **Tipos de Modelos de Simulação**. [*S.l.*: *s.n.*]. Disponível em: http://www.mat.ufrgs.br/~viali/estatistica/mat2274/material/laminas/Comp_2.pdf. Acesso em: 16 ago. 2021.

WIDROW, Bernard; KOLLÁR, István. **Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications (Inglês) [Ruído de Quantização: Erros de Arredondamento em Computação Digital, Processamento de Sinais, Controle e Comunicações]**. [*S.l.*]: Cambridge University Press, 2008. Disponível em: http://oldweb.mit.bme.hu/books/quantization/. Acesso em: 28 set. 2021.

WOODALL, Christopher. **Python Pitch Shifter (Inglês) [Alterador de Tom em Python]**. [*S.l.*: *s.n.*], 2015. Disponível em: https://github.com/cwoodall/pitch-shifter-py. Acesso em: 6 mar. 2021.

YOUTUBE. **Clean Guitar Sound Sample**. [*S.l.*: *s.n.*], 2018. Disponível em: https://www.youtube.com/watch?v=76ttovQLKSE. Acesso em: 6 mar. 2021.

Apêndices

APÊNDICE A – CÓDIGOS

```
# ------------------------------------------------------------------------
# Autor: Vitor Probst Curtarelli
# Data: 29 de setembro de 2021
# main() do codigo para leitura e processamento de dados para modelagem
# de sistemas nao-lineares
# ------------------------------------------------------------------------

import SIS_A, SIS_B, SIS_C, SIS_D, SIS_E
import processo as PRO
from f_imports import *

sistemas = [
    SIS_A.SistemaA(),
    SIS_B.SistemaB(),
    SIS_C.SistemaC(),
    SIS_D.SistemaD(),
    SIS_E.SistemaE(),
]
samplerate = 44100

for sistema in sistemas:
    processo = PRO.Processo(f1=20, f2=20000, samplerate=samplerate, t2=20)
    processo.eval(sistema)
    processo.separar_harmonicos(n_harmonicos=5)
    processo.load()
    processo.conversao(processo.xn, "Novak")
    processo.sintese()

    processo.plot_Xo()
    processo.plot_hn_()
    processo.plot_vec_Ho_()
    processo.plot_vec_Ho()

    plt.figure("Plot Yo")
    plt.clf()
    processo.plot_Yro(processo.xn, "Trad")
    processo.plot_Yro(processo.xn, "Novak")
    processo.plot_Yro(processo.xn, "Novo")
    processo.plot_Yo()
    plt.legend(loc="lower left")
    plt.xlim(10, processo.samplerate//2)
    plt.xlabel("Frequencia (Hz)")
    plt.ylabel("Magnitude (dBFS)")
    plt.grid(color=[0.9, 0.9, 0.9])
    plt.tight_layout()
    plt.savefig(\'../Figuras_TCC/Sinais/{}/{}_plot_Yo_{}.pdf\'.format(
        processo.S__.name,
        processo.S__.shortname,
        processo.n_harmonicos))
    # plt.show()
    print("{} concluido".format(sistema.name))
```

```python
# ------------------------------------------------------------------------
# Autor: Vitor Probst Curtarelli
# Data: 29 de setembro de 2021
# Codigo da classe Processo para processamento dos sinais e sistemas
# ------------------------------------------------------------------------

from f_imports import *
import pitchshifter_v2 as psv2


class Processo:
    def __init__(self, t1=0, t2=10, f1=20, f2=20000, samplerate=44100):
        self.t1 = t1
        self.t2 = t2
        self.T = t2-t1
        self.f1 = f1
        self.f2 = f2
        self.samplerate = samplerate

        self.L = (1/self.f1)*np.round(self.T*self.f1/np.log(f2/f1))
        self.t_1 = t_1 = np.linspace(t1, t2, (t2-t1)*self.samplerate)
        self.pure_sweep = np.sin(2 * np.pi * f1 * self.L * (np.exp(t_1 / self.L) - 1))
        self.pure_asweep = (f1 / self.L) * np.exp(-t_1 / self.L) * np.flip(self.pure_sweep)

        self.sweep = np.hstack([self.pure_sweep, 0 * self.pure_sweep[1:]])
        self.asweep = np.hstack([self.pure_asweep, 0 * self.pure_asweep[1:]])
        self.asweep /= np.sqrt(np.sum(self.asweep**2)*np.sum(self.sweep**2))
        self.t_2 = np.arange(0, self.sweep.shape[0])/self.samplerate

        self.samples = self.sweep.shape[0]

        self.freqs = np.linspace(0, self.samplerate, self.samples)
        self.S__ = None
        self.xn = None
        self.yn = None
        self.yrn = None
        self.vec_xn = None
        self.metodo = None
        self.n_harmonicos = None
        self.bins = 200

        self.c = {"light": [0.8, 0.8, 0.8],
                  "dark":  [0.2, 0.2, 0.2]}

        self.metodos = ["Trad", "Novak", "Novo"]
        self.full_metodos = {"Trad":    "Tradicional",
                             "Novak":   "Novak",
                             "Novo":    "PSA",}

    def gen_sweep(self):
        wavfile.write("sweep_{}_{}_{}.wav".format(self.f1, self.f2, self.samplerate),
                      self.samplerate, self.sweep.astype(np.float32))

    def eval(self, sistema):
        xn = self.sweep
        ysweep = sistema.simulate(xn)
        hn_ = convol(ysweep, self.asweep)
```

```
58          self.S__ = sistema
59          self.S__.ysweep = ysweep
60          self.S__.hn_ = np.copy(hn_)
61          self.centralizar()
62
63      def centralizar(self):
64          hn_ = self.S__.hn_
65          pico = np.where(hn_ == np.amax(hn_))[0]
66          comp = hn_.shape[0]
67          hn_ = np.roll(hn_, -(pico - comp//2))
68          self.S__.hn_ = hn_
69
70      def separar_harmonicos(self, n_harmonicos=5):
71          hn_ = self.S__.hn_
72          L = self.L
73          t2 = self.t2
74          samplerate = self.samplerate
75          ts = np.arange(n_harmonicos, 0, -1)
76          delta_ts = L*np.log(ts)
77          e_delta_ts = t2-delta_ts
78          e_delta_ns = np.floor(e_delta_ts*samplerate)
79          vec_hn_ = []
80          for i in range(n_harmonicos):
81              aux = np.copy(hn_)
82              aux = np.roll(aux, -int(e_delta_ns[i])+1)
83              to_zero = abs(int(e_delta_ns[(i+1) % n_harmonicos] - e_delta_ns[i]) // 2)
84              aux = janelamento(aux, to_zero, int(0.1*to_zero))
85              aux = aux/np.amax(hn_)
86              vec_hn_.append(aux)
87
88          vec_hn_.reverse()
89          self.n_harmonicos = n_harmonicos
90          self.S__.vec_hn_ = vec_hn_
91
92      def load(self, xn="clean_guitar.wav"):
93          _, xn = wavfile.read(xn)
94          xn = xn[:self.sweep.shape[0], 0].astype(np.float32)
95          self.xn = xn/np.amax(xn)
96
97      def conversao(self, xn, metodo):
98          if metodo not in self.metodos:
99              metodo = "Novak"
100
101         if metodo == "Trad":
102             self.conversao_trad(xn)
103         elif metodo == "Novak":
104             self.conversao_novak(xn)
105         elif metodo == "Novo":
106             self.conversao_novo(xn)
107         self.metodo = metodo
108
109     def conversao_trad(self, xn):
110         vec_hn = self.S__.vec_hn_
111         vec_xn = []
112         for i in range(self.n_harmonicos):
113             vec_xn.append(xn**(i+1))
114
```

```
115         self.S__.vec_hn = vec_hn
116         self.vec_xn = vec_xn
117 
118     def conversao_novak(self, xn):
119         A = np.zeros([self.n_harmonicos, self.n_harmonicos])
120 
121         for m_ in range(self.n_harmonicos):
122             m = m_+1
123             for eta in range(self.n_harmonicos):
124                 if (m_ + eta) % 2 == 0 and m >= eta:
125                     A[m_, eta] = 1/(2**(m-1)) * choose(m, np.floor((m-eta)/2))
126 
127         A_ti = np.linalg.inv(A.T)
128         vec_Ho_ = [fft(f) for f in self.S__.vec_hn_]
129         arr_Ho_ = np.array(vec_Ho_)
130         arr_Holeft_ = arr_Ho_[:, :(arr_Ho_.shape[1]//2 + 1)]
131         arr_Horght_ = arr_Ho_[:, (arr_Ho_.shape[1]//2 + 1):]
132 
133         arr_Holeft = A_ti@arr_Holeft_
134         arr_Horght = np.conj(A_ti)@arr_Horght_
135 
136         arr_Ho = np.hstack([arr_Holeft, arr_Horght])
137         vec_Ho = [arr_Ho[i, :] for i in range(arr_Ho.shape[0])]
138         vec_hn = [np.real(ifft(f)) for f in vec_Ho]
139         vec_xn = []
140         for i in range(self.n_harmonicos):
141             vec_xn.append(xn**(i+1))
142 
143         self.S__.vec_hn = vec_hn
144         self.vec_xn = vec_xn
145 
146     def conversao_novo(self, xn):
147         vec_hn = self.S__.vec_hn_
148         vec_xn = [xn]
149         for i in range(1, self.n_harmonicos):
150             _, xkn, _ = psv2.PSA(factor=i+1, source=(self.samplerate, xn), chunk_size=2**13)
151             vec_xn.append(xkn)
152         self.S__.vec_hn = vec_hn
153         self.vec_xn = vec_xn
154 
155     def sintese(self):
156         yrn = 0
157         for i in range(self.n_harmonicos):
158             yrn += convol(self.vec_xn[i], self.S__.vec_hn[i])
159         self.yrn = yrn
160 
161     def plot_hn_(self):
162         plt.figure("Plot hn_")
163         plt.clf()
164         hn_ = self.S__.hn_
165         plt.axvline(x=self.t_2[-1]/2, c=self.c["dark"])
166         plt.axhline(y=0, c=self.c["dark"])
167         plt.plot(self.t_2, hn_, label="Resposta harmonica")
168         plt.legend(loc="upper left")
169         plt.xlim([0, 25])
170         plt.xlabel("Tempo (s)")
171         plt.ylabel("Amplitude")
```

```
172         plt.tight_layout()
173         plt.savefig(\'../Figuras_TCC/Sinais/{}/{}_plot_hn_.pdf\'.format(
174             self.S__.name, self.S__.shortname))
175 
176     def plot_Ho_(self):
177         plt.figure("Plot Ho_")
178         plt.clf()
179         Ho_ = fft(self.S__.hn_)
180         vlogHo_ = 20*np.log10(np.abs(Ho_))
181         plt.axvline(x=self.f1, c=self.c["light"])
182         plt.axvline(x=self.f2, c=self.c["light"])
183         plt.semilogx(self.freqs[:self.samples//2], vlogHo_[:self.samples//2],
184                      label="Resposta harmonica")
185         plt.legend(loc=\'lower left\')
186         plt.xlim([10, self.samplerate//2])
187         plt.xlabel("Frequencia (Hz)")
188         plt.ylabel("Magnitude (dBFS)")
189         plt.grid(color=[0.9, 0.9, 0.9])
190         plt.tight_layout()
191         plt.savefig(\'../Figuras_TCC/Sinais/{}/{}_plot_Ho_.pdf\'.format(
192             self.S__.name, self.S__.shortname))
193 
194     def plot_vec_Ho_(self):
195         plt.figure("Plot vec Ho_")
196         plt.clf()
197         for index in range(self.n_harmonicos):
198             if (index+1) in self.S__.filtros or "all" in self.S__.filtros:
199                 h = self.S__.vec_hn_[index]
200                 H = fft(h)
201                 vlogH = 20*np.log10(np.abs(H))
202                 plt.axvline(x=self.f1, c=self.c["light"])
203                 plt.axvline(x=self.f2, c=self.c["light"])
204                 plt.semilogx(self.freqs[:self.samples//2], vlogH[:self.samples//2],
205                              label="{}o harmonico".format(index+1))
206         plt.legend(loc="lower left")
207         plt.xlim(10, self.samplerate//2)
208         plt.xlabel("Frequencia (Hz)")
209         plt.ylabel("Magnitude (dBFS)")
210         plt.grid(color=[0.9, 0.9, 0.9])
211         plt.tight_layout()
212         plt.savefig(\'../Figuras_TCC/Sinais/{}/{}_plot_vec_Ho_.pdf\'.format(
213             self.S__.name, self.S__.shortname))
214 
215     def plot_vec_Ho(self):
216         plt.figure("Plot vec Ho")
217         plt.clf()
218         for index in range(self.n_harmonicos):
219             if (index+1) in self.S__.filtros or "all" in self.S__.filtros:
220                 h = self.S__.vec_hn[index]
221                 H = fft(h)
222                 vlogH = 20*np.log10(np.abs(H))
223                 plt.axvline(x=self.f1, c=self.c["light"])
224                 plt.axvline(x=self.f2, c=self.c["light"])
225                 plt.semilogx(self.freqs[:self.samples//2], vlogH[:self.samples//2],
226                              label="{}o harmonico".format(index+1))
227         plt.legend(loc="lower left")
228         plt.xlim(10, self.samplerate//2)
```

```
229         plt.xlabel("Frequencia (Hz)")
230         plt.ylabel("Magnitude (dBFS)")
231         plt.grid(color=[0.9, 0.9, 0.9])
232         plt.tight_layout()
233         plt.savefig(\'../Figuras_TCC/Sinais/{}/{}_plot_vec_Ho.pdf\'.format(
234             self.S__.name, self.S__.shortname))
235 
236     def plot_Xo(self):
237         plt.figure("Plot xn")
238         plt.clf()
239         Xo = fft(self.xn)
240         vlogXo = 20*np.log10(np.abs(Xo))
241         plt.axvline(x=self.f1, c=self.c["light"])
242         plt.axvline(x=self.f2, c=self.c["light"])
243         plt.semilogx(self.freqs[:self.samples//2], vlogXo[:self.samples//2],
244                      label="Sinal de entrada")
245         plt.legend(loc="lower left")
246         plt.xlim(10, self.samplerate//2)
247         plt.xlabel("Frequencia (Hz)")
248         plt.ylabel("Magnitude (dBFS)")
249         plt.grid(color=[0.9, 0.9, 0.9])
250         plt.tight_layout()
251         plt.savefig(\'../Figuras_TCC/Sinais/plot_xn.pdf\')
252 
253         freqs = self.freqs[:self.samples//2]
254         vlogYo = vlogXo[:self.samples//2]
255         plt.figure("Plot xn bins")
256         plt.clf()
257         plt.axvline(x=self.f1, c=self.c["light"])
258         plt.axvline(x=self.f2, c=self.c["light"])
259         Yobins = stats.binned_statistic(freqs, vlogYo,
260                                         bins=np.logspace(0, np.log10(self.samplerate//2),
261                                                          self.bins))
262         plt.plot(np.convolve(Yobins.bin_edges, np.ones(2), \'valid\')/2,
263                  Yobins.statistic,
264                  label="Sinal de saida real")
265         plt.xscale(\'log\')
266         plt.legend(loc="lower left")
267         plt.xlim(10, self.samplerate//2)
268         plt.xlabel("Frequencia (Hz)")
269         plt.ylabel("Magnitude (dBFS)")
270         plt.grid(color=[0.9, 0.9, 0.9])
271         plt.tight_layout()
272         plt.savefig(\'../Figuras_TCC/Sinais/plot_xn_bins.pdf\')
273 
274     def plot_Yo(self):
275         self.yn = self.S__.simulate(self.xn, "saida")
276         Yo = fft(self.yn)
277         vlogYo = 20*np.log10(np.abs(Yo))
278         plt.axvline(x=self.f1, c=self.c["light"])
279         plt.axvline(x=self.f2, c=self.c["light"])
280 
281         freqs = self.freqs[:self.samples//2]
282         vlogYo = vlogYo[:self.samples//2]
283 
284         Yobins = stats.binned_statistic(freqs, vlogYo,
285                                         bins=np.logspace(0, np.log10(self.samplerate//2),
```

```
286                                                         self.bins))
287         plt.plot(np.convolve(Yobins.bin_edges, np.ones(2), \'valid\')/2,
288                  Yobins.statistic, # - np.amax(Yobins.statistic),
289                  label="Sinal de saida real")
290         plt.xscale(\'log\')
291 
292     def plot_Yro(self, xn, metodo):
293         self.conversao(xn, metodo)
294         metodo = self.full_metodos[metodo]
295         self.sintese()
296         freqs = self.freqs[:self.samples//2]
297 
298         Yro = fft(self.yrn)
299         vlogYro = 20*np.log10(np.abs(Yro))
300         vlogYro = vlogYro[:self.samples//2]
301         Yrobins = stats.binned_statistic(freqs, vlogYro,
302                                          bins=np.logspace(0, np.log10(self.samplerate//2),
303                                                           self.bins))
304         plt.plot(np.convolve(Yrobins.bin_edges, np.ones(2), \'valid\')/2,
305                  Yrobins.statistic, # - np.amax(Yrobins.statistic),
306                  label="Sinal de saida modelado - {}".format(metodo))
```

```
#-------------------------------------------------------------------------
# Autor: Vitor Probst Curtarelli
# Data: 03 de setembro de 2021
# Imports e funcoes genericas a serem utilizadas
#-------------------------------------------------------------------------


import numpy as np
from scipy.fftpack import fft
from scipy.fftpack import ifft
import scipy as sp
import scipy.signal as signal
from scipy import stats
from scipy.io import wavfile
import matplotlib.pyplot as plt
import math


def convol(x, h):
    X = fft(x)
    H = fft(h)
    Y = X*H
    y = ifft(Y)
    y = np.real(y)
    return y


def choose(n, m):
    return math.factorial(n)/(math.factorial(m)*math.factorial(n-m))


def janelamento(f, c1, c2):
    c2 = c2/c1
    n = np.arange(f.shape[0])
    n_hann = np.arange(int(c1*2*c2))
    hann = np.sin(np.pi*n_hann/n_hann.shape[0])**2
    hann_right = hann[hann.shape[0]//2:]
    janela = np.ones_like(n, dtype=float)
    janela[(c1 - hann_right.shape[0]):c1] = hann_right
    janela[c1:] = 0
    f = f*janela
    return f
```

```
##
# pitch-shifter-cli.py: Pitch Shifter Command Line Tool
#
# Author(s): Chris Woodall <chris@cwoodall.com>
# Adapted by: Vitor Probst Curtarelli <vitor.curtarelli@gmail.com>
# MIT License 2015-2021 (c) Chris Woodall <chris@cwoodall.com>
##
import scipy
import scipy.interpolate
import scipy.io.wavfile
import sys

from PSA_stft import *
from PSA_vocoder import *
from PSA_utilities import *
from PSA_resampler import linear_resample


def PSA(factor, source="sample1.wav", out="out.wav", blend=1, chunk_size=4096,
        overlap=0.9, no_resample=False):
    if type(source) == str:
        try:
            source = scipy.io.wavfile.read(source)
        except:
            print("File {0} does not exist".format(source))
            sys.exit(-1)
    elif type(source) in [list, tuple]:
        pass
    else:
        print("Invalid file format")
        sys.exit(-1)

    RESAMPLING_FACTOR = factor
    HOP = int((1-overlap)*chunk_size)
    HOP_OUT = int(HOP*RESAMPLING_FACTOR)

    audio_samples = source[1].tolist()

    rate = source[0]
    mono_samples = stereoToMono(audio_samples)
    frames = stft(mono_samples, chunk_size, HOP)
    vocoder = PhaseVocoder(HOP, HOP_OUT)
    adjusted = [frame for frame in vocoder.sendFrames(frames)]

    merged_together = istft(adjusted, chunk_size, HOP_OUT)

    if no_resample:
        final = merged_together
    else:
        resampled = linear_resample(merged_together,
                                    len(mono_samples))
        final = resampled * blend + (1-blend) * mono_samples
        final = (final/np.amax(final))*np.amax(mono_samples)

    if len(source[1].shape) == 2:
        input = np.average(source[1], 1)
    else:
```

```
58          input = source[1]
59      return input, final, rate
```

```
1  #!/usr/bin/env python
2
3  import numpy as np
4  from scipy.interpolate import interp1d
5
6  def linear_resample(samples, out_len):
7      """
8      Resamples samples to have length equal to out_len.
9
10     Uses a 1d linear interpolator
11
12     Args:
13       samples: samples to resample using an interpolator
14       out_len: Length of output sample size.
15
16     Returns:
17         resampled and interpolated output.
18     """
19     sample_size = len(samples)
20     interpolator = interp1d(np.arange(0, sample_size), samples, kind='linear')
21
22     resample_n = np.linspace(0, sample_size-1, out_len)
23     return interpolator(resample_n)
```

```
1  #!/usr/bin/env python
2
3  import numpy as np
4  from scipy.fft import fft, ifft
5  from scipy.signal import hanning
6  from numpy import array, zeros, real
7
8
9  def stft(x, chunk_size, hop, w=None):
10     """
11     Takes the short time fourier transform of x.
12
13     Args:
14       x: samples to window and transform.
15       chunk_size: size of analysis window.
16       hop: hop distance between analysis windows
17       w: windowing function to apply. Must be of length chunk_size
18
19     Returns:
20       STFT of x (X(t, omega)) hop size apart with windows of size chunk_size.
21
22     Raises:
23       ValueError if window w is not of size chunk_size
24     """
25     if not w:
26         w = hanning(chunk_size)
27     else:
28         if len(w) != chunk_size:
29             raise ValueError("window w is not of the correct length {0}.".format(chunk_size))
30     X = array([fft(w*x[i:i+chunk_size])
```

```
                for i in range(0, len(x)-chunk_size, hop)]) /\
            np.sqrt(((float(chunk_size)/float(hop))/2.0))
    return X


def istft(X, chunk_size, hop, w=None):
    """
    Naively inverts the short time fourier transform using an overlap and add
    method. The overlap is defined by hop

    Args:
      X: STFT windows to invert, overlap and add.
      chunk_size: size of analysis window.
      hop: hop distance between analysis windows
      w: windowing function to apply. Must be of length chunk_size

    Returns:
      ISTFT of X using an overlap and add method. Windowing used to smooth.

    Raises:
      ValueError if window w is not of size chunk_size
    """

    if not w:
        w = hanning(chunk_size)
    else:
        if len(w) != chunk_size:
            raise ValueError("window w is not of the correct length {0}.".format(chunk_size

    x = zeros(len(X) * (hop))
    i_p = 0
    for n, i in enumerate(range(0, len(x)-chunk_size, hop)):
        x[i:i+chunk_size] += w*real(ifft(X[n]))
    return x
```

```
#!/usr/bin/env python

import numpy as np
import scipy as sp
import collections

def scalar_len(a):
    """

    Return
    """
    if isinstance(a, collections.Iterable):
        return len(a)
    else:
        return 1

def complex_polarToCartesian(r, theta):
    """
    Convert a polar representation of a complex number to a cartesian
    representation.

    Can be used with a numpy array allowing for block conversions
```

```
23
24      Example Usage:
25
26      results = complex_polarToCartesian(1.4142, 0.7854)
27
28      results approx. 1+1j
29      """
30      return r * np.exp(theta*1j)
31
32  def complex_cartesianToPolar(x):
33      """
34      Convert a cartesian representation of a complex number to a polar
35      representation.
36
37      Can be used with a numpy array allowing for block conversions
38
39      Example Usage:
40
41      results = complex_cartesianToPolar(1 + 1j)
42
43      results approx. (1.4142, 0.7854)
44      """
45      return (np.abs(x), np.angle(x))
46
47  def stereoToMono(audio_samples):
48      """
49      Takes in an 2d array of stereo samples and returns a mono numpy
50      array of dtype np.int16.
51      """
52      LEFT = 0
53      RIGHT = 1
54      channels = scalar_len(audio_samples[0])
55      if channels == 1:
56          mono_samples = np.asarray(audio_samples,
57              dtype=np.int16)
58
59      elif channels == 2:
60          mono_samples = np.asarray(
61              [(sample[RIGHT] + sample[LEFT])/2 for sample in audio_samples],
62              dtype=np.int16
63          )
64
65      else:
66          raise Exception("Must be mono or stereo")
67
68
69
70      return mono_samples
```

```
1  #!/usr/bin/env python
2  from PSA_utilities import *
3  import numpy as np
4
5
6  class PhaseVocoder(object):
7      """
8      Implements the phase vocoder algorithm.
```

```
9
10      Usage:
11          from phaseshifter import PhaseVocoder, stft
12          vocoder = PhaseVocoder(HOP, HOP_OUT)
13          phase_corrected_frames = [frame for frame in vocoder.sendFrames(frames)]
14
15      Attributes:
16          input_hop: Input hop distance/size
17          output_hop: Output hop distance/size
18          last_phase: numpy array of all of the previous frames phase information.
19          phase_accumulator: numpy array of accumulated phases.
20      """
21
22      def __init__(self, ihop, ohop):
23          """
24          Initialize the phase vocoder with the input and output hop sizes desired.
25
26          Args:
27              ihop: input hop size
28              ohop: output hop size
29          """
30          self.input_hop = int(ihop)
31          self.output_hop = int(ohop)
32          self.reset()
33
34      def reset(self):
35          """
36          Reset the phase accumulator and the previous phase stored to 0.
37          """
38          self.last_phase = 0
39          self.phase_accumulator = 0
40
41      def sendFrame(self, frame):
42          """
43          Send a single frame to the phase vocoder
44
45          Args:
46              frame: frame of FFT information.
47
48          Returns: phase corrected frame
49          """
50          omega_bins = 2*np.pi*np.arange(len(frame))/len(frame)
51          magnitude, phase = complex_cartesianToPolar(frame)
52
53          delta_phase = phase - self.last_phase
54          self.last_phase = phase
55
56          delta_phase_unwrapped =  delta_phase - self.input_hop * omega_bins
57          delta_phase_rewrapped = np.mod(delta_phase_unwrapped + np.pi, 2*np.pi) - np.pi
58
59          true_freq = omega_bins + delta_phase_rewrapped/self.input_hop
60
61          self.phase_accumulator += self.output_hop * true_freq
62
63          return complex_polarToCartesian(magnitude, self.phase_accumulator)
64
65      def sendFrames(self, frames):
```

```
66          """
67          A generator function for processing a group of frames.
68
69          Args:
70              frames: an array of numpy arrays containing frequency domain information.
71
72          Returns: Each iteration yields the phase correction for the current frame.
73          """
74          for frame in frames:
75              yield self.sendFrame(frame)
```

```python
# ------------------------------------------------------------------------
# Autor: Vitor Probst Curtarelli
# Data: 03 de setembro de 2021
# Codigo de modelagem do sistema A para o trabalho
# ------------------------------------------------------------------------

from f_imports import *


class SistemaA:
    # variables
    def __init__(self, samplerate=44100):
        self.samplerate = samplerate
        self.filtros = [1, 2, 3]
        self.name = "Sistema A"
        self.shortname = "sis_A"

        self.ysweep = None
        self.hn_ = None
        self.vec_hn_ = None
        self.vec_hn = None

    def simulate(self, x, type_="entrada"):
        # y = signal.sosfilt(self.f1, x) + 0.3*signal.sosfilt(self.f2, x**3)
        y = x - 0.5*x**2 + 0.2*x**3
        return y


if __name__ == "__main__":
    sistema = SistemaA()
    sistema.bode()
    # t = np.linspace(0, 10, 44100*10)
    # x = np.sin(t)
    # y = sistema.simulate(x)
    # plt.plot(t, x)
    # plt.plot(t, y)
    # plt.show()
```

```python
# ------------------------------------------------------------------------
# Autor: Vitor Probst Curtarelli
# Data: 03 de setembro de 2021
# Codigo de modelagem do sistema B para o trabalho
# ------------------------------------------------------------------------

from f_imports import *


class SistemaB:
    # variables
    def __init__(self, samplerate=44100, fc_hgh=1000, fc_low=100):
        self.samplerate = samplerate
        self.fc_hgh = fc_hgh
        self.fc_low = fc_low

        self.f1 = signal.butter(N=3, Wn=self.fc_hgh, btype='high',
                                analog=False, output='sos', fs=self.samplerate)
        self.f2 = signal.butter(N=3, Wn=self.fc_low, btype='low',
                                analog=False, output='sos', fs=self.samplerate)
        self.filtros = [1, 3]
        self.FRFs = None
        self.name = "Sistema B"
        self.shortname = "sis_B"

        self.ysweep = None
        self.hn_ = None
        self.vec_hn_ = None
        self.vec_hn = None

    def bode(self):
        samplerate = self.samplerate
        fc1 = self.fc_hgh
        fc2 = self.fc_low
        f1 = self.f1
        f2 = self.f2
        cline = [0.7, 0.7, 0.7]
        cgrid = [0.9, 0.9, 0.9]
        caxis = [0.2, 0.2, 0.2]

    # filters f1 and f2

    # transfer function of filters f1 and f2
        _, FRF_f1 = signal.sosfreqz(f1, worN=samplerate, whole=True, fs=samplerate)
        _, FRF_f2 = signal.sosfreqz(f2, worN=samplerate, whole=True, fs=samplerate)

        # transfer function of filters f1 and f2 prepared for plots
        Tf_f1 = [20*np.log10(np.abs(FRF_f1)), (180/np.pi)*np.angle(FRF_f1)]
        Tf_f2 = [20*np.log10(np.abs(FRF_f2)), 2*np.angle(FRF_f2)]
        Tf_f2[1] = np.unwrap(Tf_f2[1])*180/(2*np.pi)

        self.FRFs = [Tf_f1[0], 0, Tf_f2[0]]
        # # Bode plot of f1
        # Magnitude plot
        plt.subplot(2, 1, 1)
        plt.xlim([1, samplerate//2])
        plt.axvline(x=fc1, c=cline)
```

```
58          plt.axhline(y=0,   c=caxis)
59          plt.semilogx(Tf_f1[0])
60          plt.grid(c=cgrid)
61          # plt.title("Diagrama de Bode do filtro $f_1(t)$")
62          plt.xlabel("Frequencia (Hz)")
63          plt.ylabel("Modulo (dB)")
64
65          # Phase plot
66          plt.subplot(2, 1, 2)
67          plt.xlim([1, samplerate//2])
68          plt.yticks([-90, -45, 0, 45, 90])
69          plt.ylim([-50, 100])
70          plt.axvline(x=fc1, c=cline)
71          plt.axhline(y=0, c=caxis)
72          plt.semilogx(Tf_f1[1])
73          plt.grid(c=cgrid)
74          plt.xlabel("Frequencia (Hz)")
75          plt.ylabel("Fase (o)")
76          plt.tight_layout()
77          plt.show()
78
79          # # Bode plot of f2
80          # Magnitude plot
81          plt.subplot(2, 1, 1)
82          plt.xlim([1, samplerate//2])
83          plt.gca().set_ylim(bottom=-50)
84          plt.axvline(x=fc2, c=cline)
85          plt.axhline(y=0,   c=caxis)
86          plt.semilogx(Tf_f2[0])
87          plt.grid(c=cgrid)
88          # plt.title("Diagrama de Bode do filtro $f_2(t)$")
89          plt.xlabel("Frequencia (Hz)")
90          plt.ylabel("Modulo (dB)")
91
92          # Phase plot
93          plt.subplot(2, 1, 2)
94          plt.xlim([1, samplerate//2])
95          plt.yticks([-90, -45, 0, 45, 90])
96          plt.ylim([-100, 50])
97          plt.axvline(x=fc2, c=cline)
98          plt.axhline(y=0, c=caxis)
99          plt.semilogx(Tf_f2[1])
100         plt.grid(c=cgrid)
101         plt.xlabel("Frequencia (Hz)")
102         plt.ylabel("Fase (o)")
103         plt.tight_layout()
104         plt.show()
105
106     def simulate(self, x, type_="entrada"):
107         y = signal.sosfilt(self.f1, x) + 0.3*signal.sosfilt(self.f2, x**3)
108         # y = signal.sosfilt(self.f1, x) + signal.sosfilt(self.f2, x**3)
109         return y
110
111
112 if __name__ == "__main__":
113     sistema = SistemaB()
114     sistema.bode()
```

```
115     # t = np.linspace(0, 10, 44100*10)
116     # x = np.sin(t)
117     # y = sistema.simulate(x)
118     # plt.plot(t, x)
119     # plt.plot(t, y)
120     # plt.show()
```

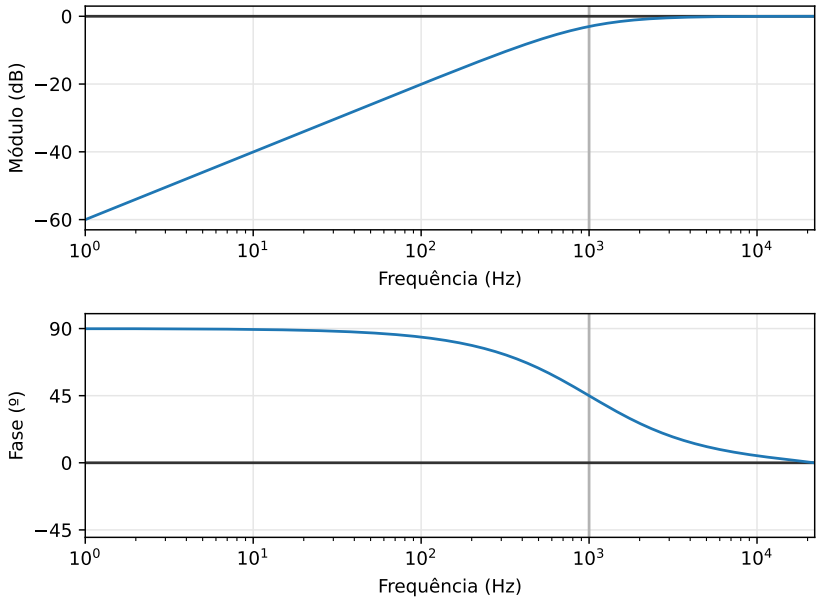
```python
# ------------------------------------------------------------------------
# Autor: Vitor Probst Curtarelli
# Data: 03 de setembro de 2021
# Codigo de modelagem do sistema C para o trabalho
# ------------------------------------------------------------------------

from f_imports import *


class SistemaC:
    # variables
    def __init__(self, samplerate=44100):
        self.samplerate = samplerate
        self.name = "Sistema C"
        self.shortname = "sis_C"
        self.filtros = ["all"]

    def simulate(self, x, type_="entrada"):
        # y = signal.sosfilt(self.f1, x) + 0.3*signal.sosfilt(self.f2, x**3)
        y = np.cbrt(x)
        return y


if __name__ == "__main__":
    sistema = SistemaC()
    sistema.bode()
    # t = np.linspace(0, 10, 44100*10)
    # x = np.sin(t)
    # y = sistema.simulate(x)
    # plt.plot(t, x)
    # plt.plot(t, y)
    # plt.show()
```

```python
# ------------------------------------------------------------------------
# Autor: Vitor Probst Curtarelli
# Data: 03 de setembro de 2021
# Codigo de modelagem do sistema D para o trabalho
# ------------------------------------------------------------------------

from f_imports import *


class SistemaD:
    # variables
    def __init__(self, samplerate=44100):
        self.samplerate = samplerate
        self.filtros = ["all"]
        self.name = "Sistema D"
        self.shortname = "sis_D"

        self.ysweep = None
        self.hn_ = None
        self.vec_hn_ = None
        self.vec_hn = None

    def simulate(self, x, type_="entrada"):
        if type_ == "entrada":
            rate, data = wavfile.read("Sistema_D/Sistema_D_medicao.wav")
            data = data[:x.shape[0], 0]
            return data
        elif type_ == "saida":
            rate, data = wavfile.read("Sistema_D/Sistema_D_comparacao.wav")
            data = data[:x.shape[0], 0]
            data = data/np.sqrt(np.sum(data**2))
            return data
```

```python
# ------------------------------------------------------------------------
# Autor: Vitor Probst Curtarelli
# Data: 03 de setembro de 2021
# Codigo de modelagem do sistema E para o trabalho
# ------------------------------------------------------------------------

from f_imports import *


class SistemaE:
    # variables
    def __init__(self, samplerate=44100):
        self.samplerate = samplerate
        self.filtros = ["all"]
        self.name = "Sistema E"
        self.shortname = "sis_E"

        self.ysweep = None
        self.hn_ = None
        self.vec_hn_ = None
        self.vec_hn = None

    def simulate(self, x, type_="entrada"):
        if type_ == "entrada":
            rate, data = wavfile.read("Sistema_E/Sistema_E_medicao.wav")
            data = data[:x.shape[0], 0]
            return data
        elif type_ == "saida":
            rate, data = wavfile.read("Sistema_E/Sistema_E_comparacao.wav")
            data = data[:x.shape[0], 0]
            data = data/np.sqrt(np.sum(data**2))
            return data
```

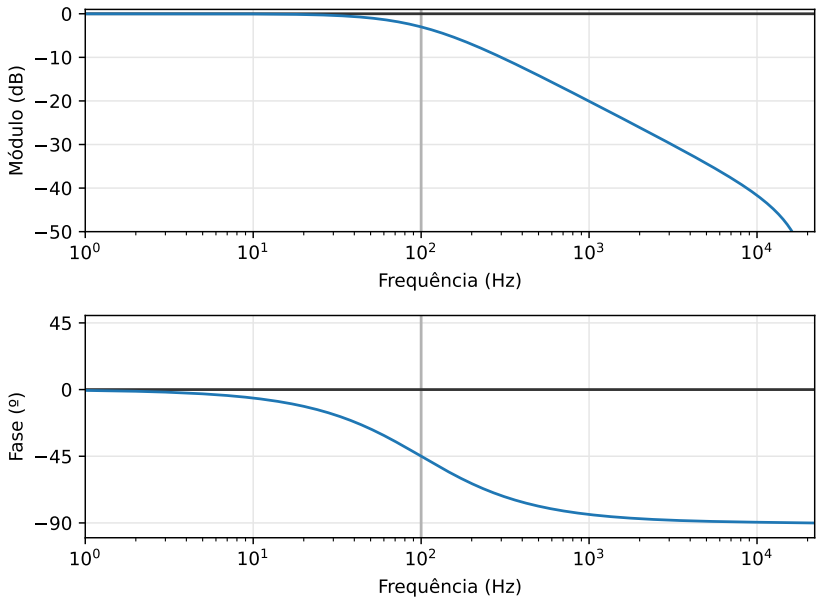
APÊNDICE B – DIAGRAMAS E ESQUEMÁTICOS

Figura B.1 – Diagramas de Bode do filtro $f_1(t)$ do sistema C.



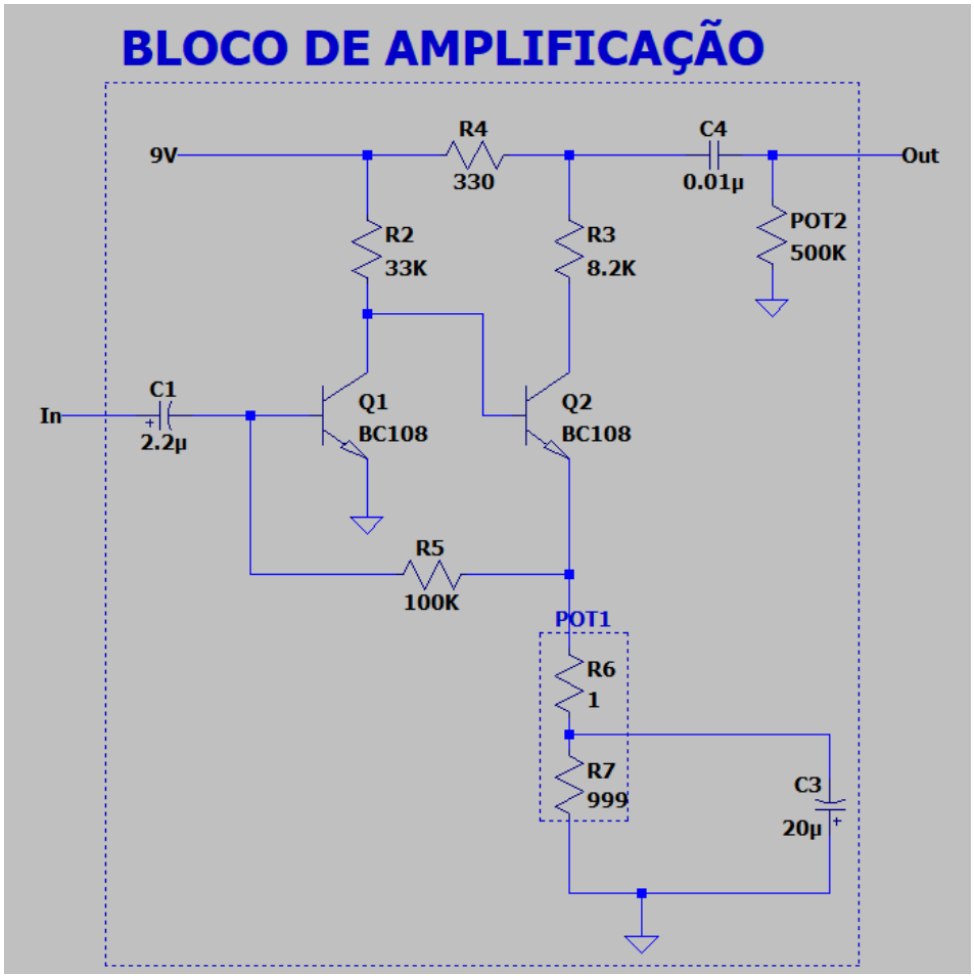
Fonte – do autor.

Figura B.2 – Diagramas de Bode do filtro $f_2(t)$ do sistema C.



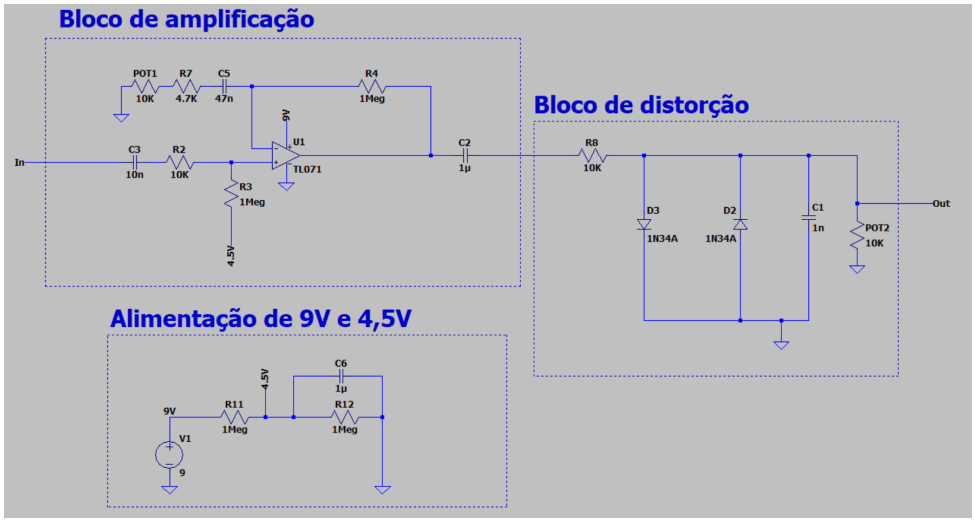
Fonte – do autor.

Figura B.3 – Esquemático do circuito do sistema D.



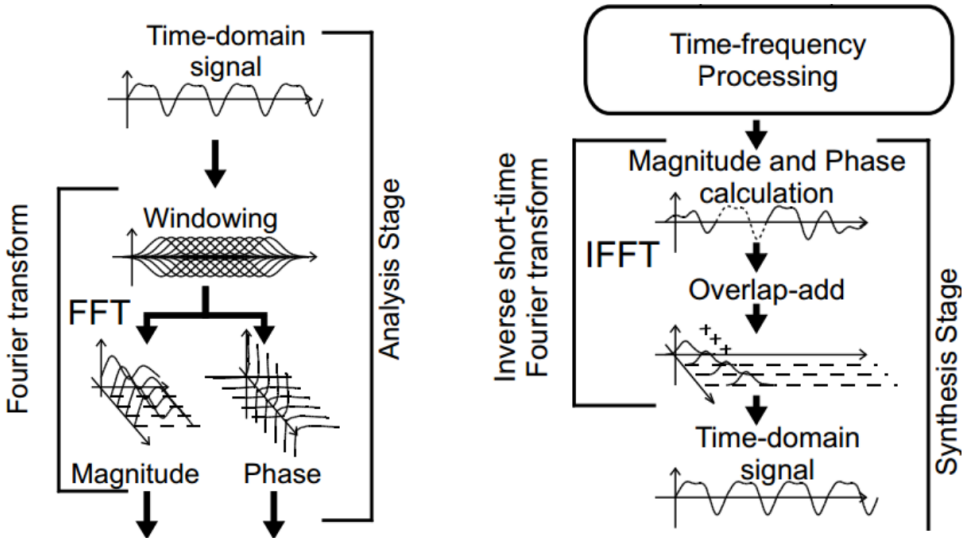
Fonte – Fornecido por João Cláudio Elsen Barcellos (BARCELLOS, 2021).

Figura B.4 – Esquemático do circuito do sistema E.



Fonte – Fornecido por João Cláudio Elsen Barcellos (BARCELLOS, 2021).

Figura B.5 – Diagrama de operação do algoritmo PVA.



Fonte – Fornecido por Andriy Temko *et. al.* (TEMKO *et al.*, 2014)