UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

Raiane Lima dos Santos

Identificação de parâmetros em modelos estocásticos do sinal glotal usando Redes Neurais Artificiais

NITERÓI

2022

UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

Raiane Lima dos Santos

Identificação de parâmetros em modelos estocásticos do sinal glotal usando Redes Neurais Artificiais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sinais e Sistemas de Comunicações Móveis.

Orientador:
Edson Luiz Cataldo Ferreira

NITERÓI

2022

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S237i Santos, Raiane Lima dos
Identificação de parâmetros em modelos estocásticos do sinal glotal usando Redes Neurais Artificiais / Raiane Lima dos Santos ; Edson Luiz Cataldo Ferreira, orientador. Niterói, 2022.
116 f.

Dissertação (mestrado)–Universidade Federal Fluminense, Niterói, 2022.

DOI: http://dx.doi.org/10.22409/PPGEET.2022.m.14690710708

1. Processamento digital de sinais. 2. Redes Neurais. 3. Produção intelectual. I. Cataldo Ferreira, Edson Luiz, orientador. II. Universidade Federal Fluminense. Escola de Engenharia. III. Título.

CDD –

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Raiane Lima dos Santos

Identificação de parâmetros em modelos estocásticos do sinal glotal usando Redes Neurais Artificiais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sinais e Sistemas de Comunicações Móveis.

Aprovada em 28 de Abril de 2022.

BANCA EXAMINADORA

Prof. Edson Luiz Cataldo Ferreira, D.Sc. – Orientador, UFF

Prof. Maurício Weber Benjó da Silva, D.Sc. – UFF

Prof. Leonardo Alfredo Forero Mendoza, D.Sc. – UERJ

Niterói

2022

Dedico este trabalho a Deus e à minha família.

Agradecimentos

Primeiramente, eu gostaria de agradecer a Deus, pois sem Ele nada disso seria possível. Obrigada por sempre estar ao meu lado em cada etapa da minha vida, e nunca soltar a minha mão mesmo nos momentos mais difíceis.

Agradeço aos meus pais, pois sempre me ensinaram que o estudo é fundamental na vida. Ensinaram, também, que o trabalho duro e a dedicação sempre serão recompensados. Graças a eles tenho essa vontade de estar em constante aprendizado e evolução.

Agradeço ao meu marido, que está ao meu lado nos bons e nos maus momentos, sempre me dando força e ânimo para superar os desafios e obstáculos da vida, me motivando para seguir em frente e continuar nesta caminhada árdua para realizar todos os meu sonhos e objetivos.

Agradecer ao professor Edson Cataldo, meu orientador, por todo o apoio acadêmico fornecido durante esses anos. Pela sua disponibilidade, paciência, incentivo, confiança e motivação que me ajudaram a sempre dar o melhor de mim.

Agradecer à Universidade Federal Fluminense, ao PPGEET por esta oportunidade de me tornar Mestre, e a todo o seu corpo docente, com o qual tive a honra de aprender e trocar conhecimentos.

Resumo

A voz é muito importante para os seres humanos, pois é principalmente através dela que ele pode se comunicar e expressar nossas ideias e sentimentos. Além disso, cada indivíduo possui características únicas em sua voz, como se fosse uma assinatura, o que permite a sua identificação. Essas características podem ser retiradas do sinal de voz ou do chamado sinal glotal, que é um sinal quase-periódico, formado pela vibração das cordas vocais imediatamente após o fluxo de ar passar pela glote. O sinal glotal e, consequentemente, o sinal de voz, podem ser gerados através de modelos determinísticos ou, de forma mais realista, modelos estocásticos, como os usados nesse trabalho. O objetivo dessa dissertação é identificar os parâmetros de um modelo estocástico do sinal glotal usando utilizando Redes Neurais Artificiais. Os modelos matemáticos determinísticos utilizados para o sinal glotal são o de Rosenberg e o de Liljencrants-Fant. O intervalo de tempo entre o fechamento e a abertura das cordas vocais a cada ciclo, chamado de intervalo de tempo glotal, é modelado como um processo estocástico tornando, assim, os modelos do sinal glotal estocásticos. As Redes Neurais Artificiais mostraram-se eficientes na identificação dos parâmetros dos modelos para diferentes casos de densidades espectrais associadas ao processo estocástico considerado.

Palavras-chave: redes neurais, sinal glotal, inteligência artificial, aprendizado de máquina, voz artifical, python, keras, tensorflow, processamento digital de sinais de voz.

Abstract

The voice is very important for human beings, because it is mainly through it that he can communicate and express our ideas and feelings. In addition, each individual has unique characteristics in their voice, as if it were a signature, which allows their identification. These characteristics can be taken from the voice signal or the glottal signal, which is a quasi-periodic signal, formed by the vibration of the vocal cords immediately after the airflow passes through the glottis. The glottal signal and, consequently, the voice signal, can be generated through deterministic models or, more realistically, stochastic models, as used in this work. The objective of this dissertation is to identify the parameters of a stochastic model of the glottal signal using Artificial Neural Networks. The deterministic mathematical models used for the glottal signal are Rosenberg's and Liljencrants-Fant's. The time interval between the closing and opening of the vocal cords at each cycle, called the glottal time interval, is modeled as a stochastic process, thus making the models of the glottal signal stochastic. The Artificial Neural Networks proved to be efficient in the identification of model parameters for different cases of spectral densities associated with the stochastic process considered.

Keywords: neural networks, glottal signal, artificial intelligence, machine learning, artificial voice, python, keras, tensorflow, digital speech signal processing.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

Sumário

Capítulo 1

Introdução

A voz é um dos principais meios de comunicação do ser humano, pois é através dela que é possível se expressar e interagir com outras pessoas. Os estudos relacionados à produção da voz avançam cada vez mais [1] [2] [3] [4]. A cada dia mais se aproxima da síntese de vozes simuladas por robôs, mais próximas de vozes reais [5] [6] [7].

O sinal glotal é um sinal quase-periódico formado pela vibração das cordas vocais imediatamente após o fluxo de ar passar pela glote, posteriormente, filtrado e amplificado pelo trato vocal, gerando a voz audível. Dessa forma, o sinal glotal, juntamente com o sinal de voz, carrega características únicas de seus locutores como se fossem uma assinatura.

Modelos matemáticos do sinal glotal têm sido desenvolvidos ao longo dos anos [8] [9] e essa dissertação tem por objetivo identificar os parâmetros de dois desses modelos usando Redes Neurais Artificiais.

Esse trabalho resulta de um histórico de pesquisa envolvendo alunos de Iniciação Científica, Mestrado e Doutorado do professor orientador dessa dissertação. Em particular, destaca-se o trabalho desenvolvido por Carla Schueler e Felipe M. da Silva [10] durante a Iniciação Científica que tratou da verificação de locutor usando parâmetros extraídos do sinal glotal na emissão de vogais. Em seguida a esse trabalho, eu, autora dessa dissertação, juntamente com Renato Ramos [11] demos continuação ao trabalho deles, tratando da verificação de locutor para palavras completas usando parâmetros extraídos de sinais glotais e de sinais de vozes. Nos dois trabalhos, a ferramenta computacional usada foi a de Modelos Ocultos de Markov. Ambos os trabalhos foram muito importantes e serviram como base para o desenvolvimento dessa dissertação pois trataram da extração de características do sinal glotal. E, um dos principais parâmetros extraídos do sinal glotal é o *jitter*, definido como desvio aleatório dos intervalos de tempo glotais (corresponden-

tes a ciclos glotais). Dessa forma, tornou-se necessário considerar um modelo de sinal glotal que gerasse *jitter*. Para isso, na dissertação de Mestrado de Diego Bahiano [12] e, posteriormente, na publicação [13], considerou-se o intervalo de tempo glotal como um processo estocástico. E, então, dois modelos determinísticos de sinal glotal, usados nessa dissertação, tornaram-se modelos estocásticos.

Para identificar os parâmetros de um modelo estocástico, a partir de sinais de vozes (simulados ou reais) é preciso resolver um problema inverso. No caso descrito aqui, algumas características do sinal de voz são dadas e deseja-se obter os parâmetros do modelo estocástico escolhido. Claramente, esse problema não é simples, pois o problema direto é um sistema formado por equações não-lineares estocásticas. Por isso, a ideia é usar alguma ferramenta computacional para resolver o problema e foram escolhidas as Redes Neurais Artificiais.

Redes Neurais Artificiais têm sido usadas no decorrer dos anos para tratar de problemas de classificação e, também, identificação de parâmetros, inclusive em problemas relacionados à voz humana [14] [15] [16], destacando-se a identificação de patologias e envelhecimento vocal em alguns casos.

Essa dissertação reúne a geração de sinais de voz com *jitter* baseados em modelos estocásticos da produção da voz humana e identifica parâmetros desses modelos, usando sinais de vozes sintetizados mas também de vozes reais, através de Redes Neurais Artificiais.

A ideia é resolver um problema inverso; isto é, a partir de características extraídas dos sinais de voz, procura-se identificar os parâmetros dos modelos. A Fig. 1.1 ilustra o problema direto e o problema inverso.

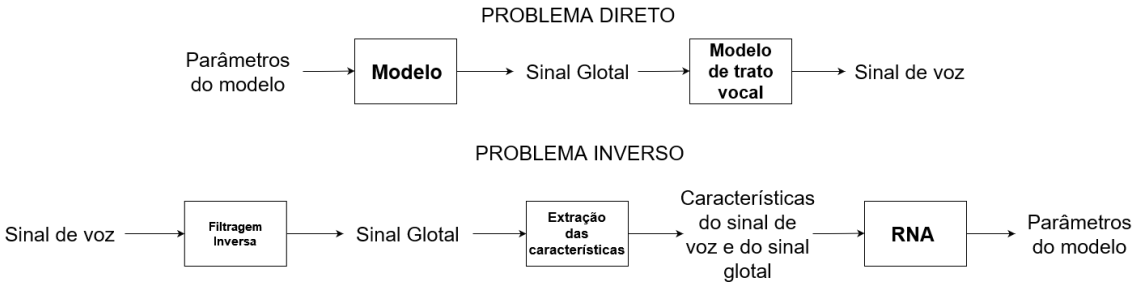

Figura 1.1: Diagrama de blocos representando o problema direto e inverso.

Para simular um sinal de voz foram utilizados dois modelos matemáticos determinísticos, o de Rosenberg [8] e o Liljencrants-Fant [9], que possuem a característica de serem diferenciáveis no tempo, com exceção dos instantes de abertura e fechamento das cordas vocais. Para produzir um sinal glotal mais próximo de um locutor real, foi considerado o intervalo de tempo entre o fechamento e a abertura das cordas vocais como um processo estocástico [12] [13].

Após a geração do sinal glotal, serão calculados alguns parâmetros de tempo e frequência, que serão utilizados como entradas de uma RNA para identificação dos parâmetros do modelo estocástico.

Foi desenvolvida uma metodologia para avaliação e comparação dos resultados obtidos e algumas ferramentas, como as curvas de aprendizagem e matriz de correlação.

A dissertação está estruturada da seguinte maneira: No Capítulo 2 será explicado como funciona a produção do sinal de voz e como é possível obter o sinal glotal através dele, assim como a extração de seus parâmetros de tempo e frequência. Em seguida, no Capítulo 3, serão apresentados os modelos estocásticos do sinal glotal que foram utilizados como base para a geração do sinal de voz. O conceito e a arquitetura da RNA utilizada serão apresentados no Capítulo 4 e o Capítulo 5 trata da metodologia utilizada na identificação dos parâmetros dos modelos de produção de voz usando as RNAs. Os resultados serão apresentados e discutidos no Capítulo 6 e, finalmente, o Capítulo 7 traz as conclusões e as propostas para trabalhos futuros.

Capítulo 2

A produção da voz

A voz é como uma assinatura, pois cada indivíduo possui características únicas em sua voz capaz de identificá-lo. Isso se deve às diferentes medidas dos órgãos que compõem o seu aparelho fonador, além de outras características como a forma de falar, respirar,...

O aparelho fonador humano é o conjunto de órgãos responsáveis pela produção da voz e divide-se em: (i) pulmões, brônquios e traquéia, órgãos respiratórios que fornecem a corrente de ar para a fonação; (ii) laringe, onde se localizam as cordas vocais, que produzem a energia sonora utilizada na fonação; (iii) faringe, boca e fossas nasais que funcionam como caixas de ressonância. A Figura 2.1 [17] ilustra o aparelho fonador.

O fluxo de ar proveniente dos pulmões, após passar pela laringe, produz um sinal (quase) periódico chamado de sinal glotal que, ao passar pela faringe, língua, dentes, lábios e fossas nasais, é filtrado e amplificado, gerando o som da fala.

Um dos órgãos de maior importância para a produção de voz é a laringe, onde estão localizadas as cordas vocais. Ao falar, as cordas vocais fazem um movimento de "abre-e-fecha" em uma determinada frequência chamada de frequência fundamental que produz uma onda de pressão acústica (quase) periódica.

Os órgãos do aparelho fonador por onde passa o sinal glotal funcionam como uma caixa de ressonância, ou seja, alteram a quantidade de energia em função da frequência do sinal, permitindo que algumas frequências sejam enfatizadas enquanto outras são atenuadas. Portanto, a função resposta em frequência do filtro (trato vocal) depende da vogal a ser produzida.

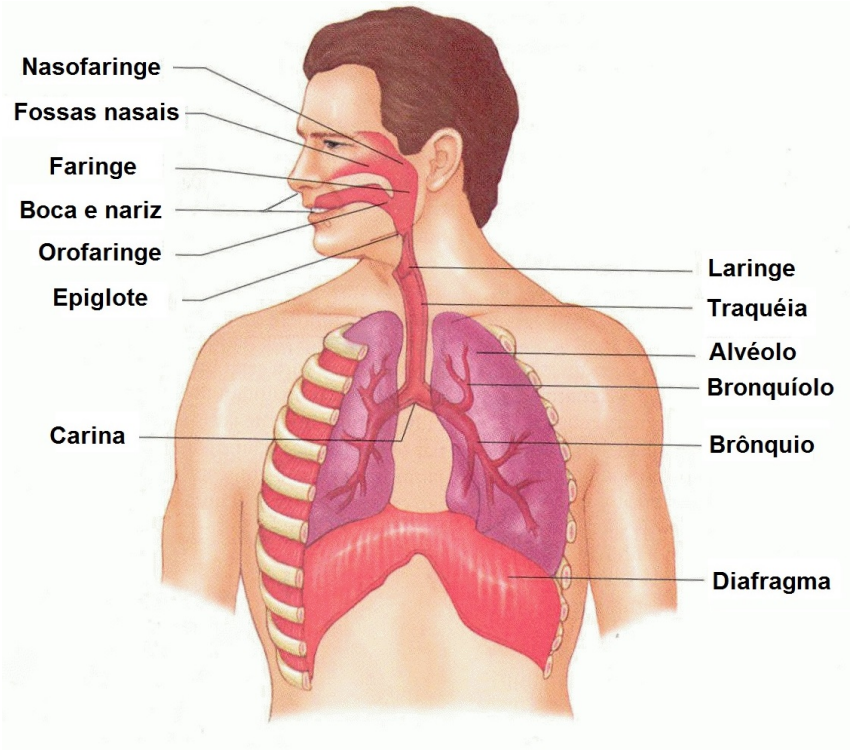

Figura 2.1: Aparelho fonador

2.1 O sinal glotal

O processo da fala se baseia na filtragem do sinal glotal, pelo trato vocal, para a formação das ondas sonoras correspondentes ao sinal de voz. O trecho de um sinal glotal está representado na Fig. 2.2 [18].

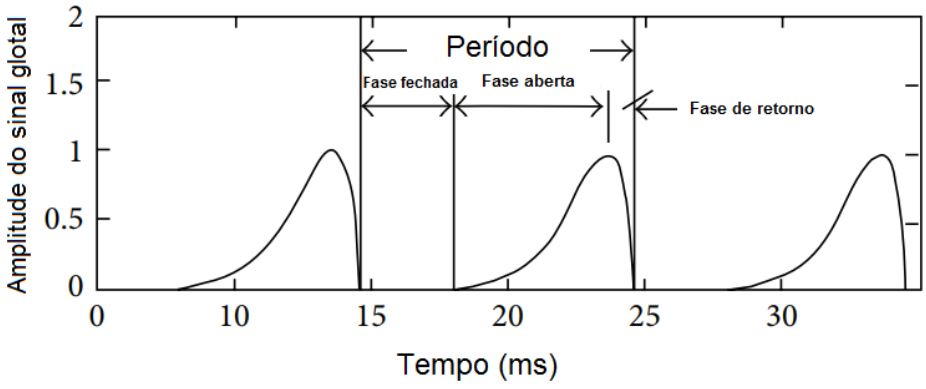

Figura 2.2: Sinal Glotal

Como já foi dito, o sinal glotal gera um sinal (quase) periódico com uma frequência fundamental, chamada comumente de f0. Ao se mover em direção à boca, o sinal encontra certas obstruções que podem ser modeladas como impedâncias que causam ressonâncias em certos harmônicos do sinal glotal. Como os parâmetros das impedâncias variam de

acordo com o movimento da língua e dos dentes, por exemplo, eles se tornam importantes para determinar certas características do locutor.

Para determinar as características relacionadas ao locutor, é preciso, primeiramente, recuperar o sinal glotal [19]. Inicialmente, remove-se os efeitos da filtragem causada pelo aparelho fonador no sinal de voz. Para realizar a filtragem inversa, utiliza-se o algoritmo IAIF (Iterative Adaptive Inverse Filtering) [18] que é um método de filtragem inversa semi-automático e que utiliza um sinal de pressão de fala como entrada, gerando uma estimativa do sinal glotal correspondente. O procedimento tem três partes fundamentais: análise, filtragem e integração. A contribuição glotal para o espectro de fala é inicialmente estimada usando uma estrutura iterativa. Esta contribuição é anulada e, então, a resposta em frequência do trato vocal é construída. Finalmente, a excitação glotal é estimada anulando os efeitos do trato vocal (usando filtragem inversa) e a radiação labial (por integração). Um esquema do modelo de filtragem inversa utilizado nesta abordagem é mostrado na Fig. 2.3.

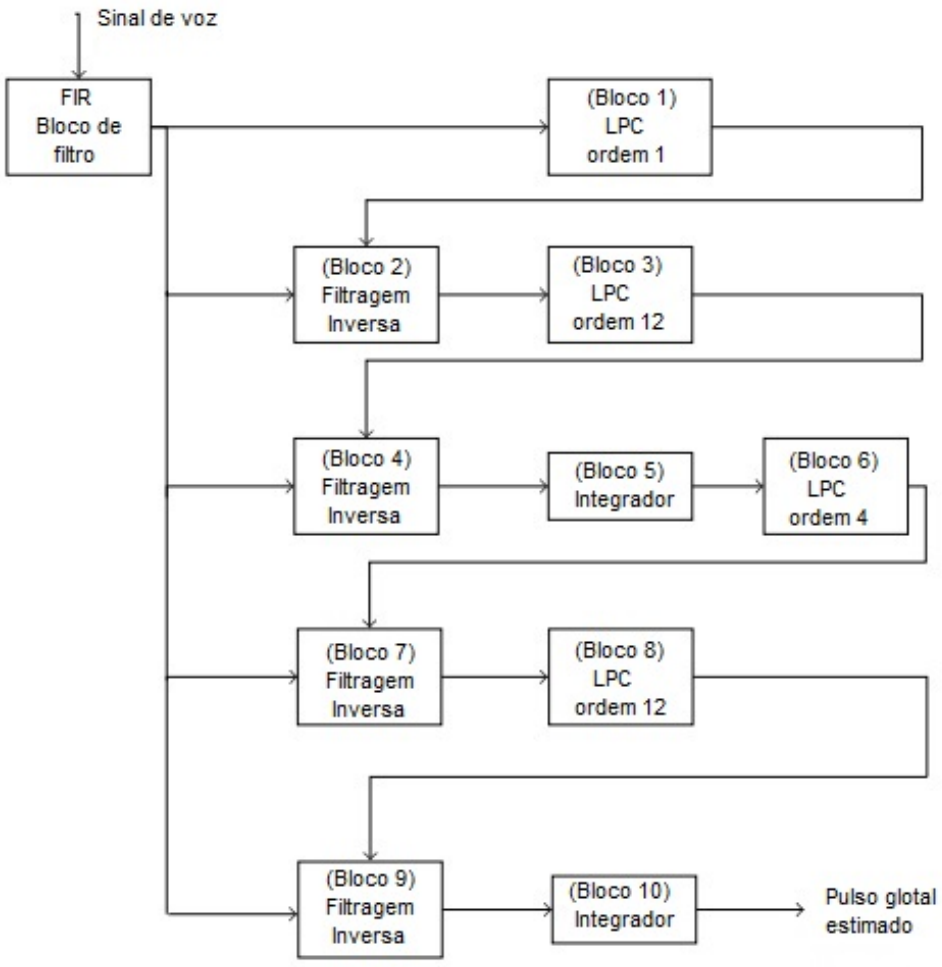

Figura 2.3: Diagrama de blocos do algoritmo IAIF

Primeiramente, o sinal de fala é janelado e, então, é utilizado um filtro passa-altas de resposta ao impulso finita de fase linear (FIR) com uma frequência de corte de 60 Hz para eliminar flutuações de baixa frequência e polarização de corrente contínua. Esse sinal filtrado de alta frequência é utilizado como entrada para os próximos estágios. No bloco 1, o ajuste LPC de ordem 1 é utilizado para calcular a contribuição do pulso glotal para o sinal da fala. No bloco 2, este coeficiente de ordem LPC 1 que simboliza a força do pulso glotal no sinal é utilizado para construir um filtro inverso que é aplicado para anular o efeito glotal do sinal de fala original. Assim, a entrada para o bloco 3 representa o sinal de fala com a componente de fluxo glotal filtrada. Em seguida, no bloco 3, o ajuste de LPC 12 é utilizado para capturar o efeito do filtro de trato vocal em termos de coeficientes de filtro. A ordem 12 é escolhida de acordo com o número de frequências formantes, maior que o número duplo de formantes considerado para a análise. No bloco 4, o efeito do filtro do trato vocal é removido do sinal de voz original, por filtragem inversa. O efeito deste bloco consiste no efeito do fluxo glotal e do efeito da radiação labial. Assim, para limpar a questão da radiação, é utilizado, no bloco 5, um integrador com fuga (com um valor de coeficiente superior a 0,9 e menor que 1), que remove o efeito de radiação labial do fluxo obtido após o bloco 4. A saída do bloco 5 é a primeira estimativa do pulso glotal.

A segunda repetição funciona de forma análoga. A saída do bloco 10 é a estimativa do pulso glotal do sinal de voz original [18].

2.1.1 Características do sinal glotal no domínio do tempo

As características do sinal glotal [18] serão utilizadas neste trabalho como entrada da rede neural. Através delas foi possível identificar os parâmetros correspondentes ao modelo estocástico. As características no domínio do tempo são extraídos (calculados) a partir de amostras em alguns instantes de tempo. Esses instantes podem ser especificados usando o sinal glotal e sua derivada. Esses instantes padronizados em [18], são:

- T - Intervalo de tempo do glotal, que é o intervalo total entre a abertura e o fechamento completo das cordas vocais em um ciclo (glotal);
- f_o - Frequência fundamental do sinal;
- t_{max} - Instante de tempo em que ocorre o valor máximo do sinal glotal;
- t_{min} - Instante de tempo em que ocorre o valor mínimo do sinal glotal;
- A_{ac} - Diferença entre a amplitude máxima e mínima do sinal glotal;

- $A_{d_{max}}$ - Amplitude máxima da derivada do sinal glotal;
- $A_{d_{min}}$ - Amplitude mínima da derivada do sinal glotal;
- t_c - Instante de fechamento das cordas vocais. Pode ser definido como o instante de tempo em que a primeira derivada do pulso glotal cruza o zero depois de um ponto de mínimo;
- t_{o1} e t_{o2} - Instantes de abertura. Para calcular t_{o1}, deve-se considerar o instante de tempo em que a amplitude é 10% ao lado esquerdo do instante t_{max}. Ainda ao lado esquerdo desse ponto, deve ser procurado o instante em que o pulso glotal derivativo cruzar o zero no sentido positivo. Esse é o ponto t_{o1}. Para obter t_{o2}, deve-se marcar o instante de tempo que é 5% maior que t_{o1} e procure o valor máximo positivo do segundo pulso glotal derivativo. Esse é o instante t_{o2};
- t_{qc} e t_{qo} - Instantes de tempo em que a amplitude do sinal é 50% da amplitude pico a pico A_{ac}.

Todos os instantes acima são calculados com base no valor de t_{max}. E baseados nesses instantes, pode-se calcular algumas características importantes:

- OQ (Open Quotient) - Mede o tempo de abertura com relação ao intervalo de tempo glotal do pulso. Pode ser dividido em OQ_1 e OQ_2:

$$OQ_1 = \frac{(t_c - t_{o1})}{T}; \tag{2.1}$$

$$OQ_2 = \frac{(t_c - t_{o2})}{T}. \tag{2.2}$$

- SQ (Speed Quotient) - Mede a relação entre o instante de abertura e o instante de fechamento. Pode ser divido em SQ_1 e SQ_2:

$$SQ_1 = \frac{(t_{max} - t_{o1})}{(t_c - t_{max})}; \tag{2.3}$$

$$SQ_2 = \frac{(t_{max} - t_{o2})}{(t_c - t_{max})}. \tag{2.4}$$

- CIQ (Closing Quotient) - É a razão entre o período de fechamento e o intervalo de tempo glotal T:

$$CIQ = \frac{(t_c - t_{max})}{T}. \tag{2.5}$$

- AQ (Amplitude Quotient) - É a razão entre o valor pico a pico do pulso glotal e a amplitude mínima do pulso glotal derivativo:

$$AQ = \frac{A_{ac}}{A_{d_{min}}}. \tag{2.6}$$

- NAQ (Normalized Amplitude Quotient) - É o valor de AQ normalizado que pode ser calculado dividindo o valor de AQ pelo intervalo de tempo glotal T:

$$NAQ = \frac{AQ}{T}. \tag{2.7}$$

- QOQ (Quasiopen Quotient) - É o mesmo que OQ, porém é calculado por meio de t_{qc} e t_{qo} em relação à duração do intervalo de tempo glotal:

$$QOQ = \frac{(t_{qc} - t_{qo})}{T}. \tag{2.8}$$

- OQ_a - É o equivalente de OQ para amplitudes:

$$OQ_a = A_{ac}(\frac{\pi}{A_{d_{max}}} + \frac{1}{A_{d_{min}}})f_o. \tag{2.9}$$

2.1.2 Jitter

O *jitter* tem papel fundamental na identificação dos parâmetros do modelo estocástico do sinal glotal, pois ele mede a variação entre os intervalos de tempo glotais de cada pulso glotal, e é isto que de fato faz diferença neste modelo, pois contrariamente ao modelo determinístico, não tem os intervalos de tempo glotais iguais. Por isso, há uma forte relação entre o *jitter* e os parâmetros do modelo estocástico.

Para melhor caracterizar o *jitter* são utilizadas diferentes medidas baseadas na análise do sinal de voz, utilizando o software Praat [20].

$$Jit_{abs} = \frac{1}{N-1}\sum_{i=1}^{N-1}|T_i - T_{i+1}| \tag{2.10}$$

$$Jit_{rel} = \frac{\frac{1}{N-1}\sum_{i=1}^{N-1}|T_i - T_{i+1}|}{\frac{1}{N}\sum_{i=1}^{N}T_i}.100 \tag{2.11}$$

$$Jit_{rap} = \frac{\frac{1}{N-2}\sum_{i=2}^{N-1}|T_i - \frac{1}{3}(T_{i-1} + T_i + T_{i+1})|}{\frac{1}{N}\sum_{i=1}^{N} T_i}.100 \tag{2.12}$$

$$Jit_{ppq5} = \frac{\frac{1}{N-4}\sum_{i=3}^{N-2}|T_i - \frac{1}{5}(T_{i-2} + T_{i-1} + T_i + T_{i+1} + T_{i+2})|}{\frac{1}{N}\sum_{i=1}^{N} T_i}.100 \tag{2.13}$$

$$Jit_{ddp} = \frac{\frac{1}{N-2}\sum_{i=2}^{N-1}|(T_{i+1} - T_i) - (T_i - T_{i-1})|}{\frac{1}{N}\sum_{i=1}^{N} T_i}.100 \tag{2.14}$$

É importante observar que nas Eqs. (2.10), (2.11), (2.12), (2.13) e (2.14), tem-se T_i como o intervalo de tempo glotal do pulso i, $i \in \{1, 2, 3, ..., N\}$, N como o número de pulsos executados e Jit_{abs}, Jit_{rel}, Jit_{rap}, Jit_{ppq5} e Jit_{ddp} como as medidas de dispersão do *jitter*.

2.1.3 Características do sinal glotal no domínio da frequência

Para estimar as características no domínio da frequência do sinal glotal [18], é preciso considerar a densidade espectral de potência do pulso glotal, como na Fig.2.4 [18].

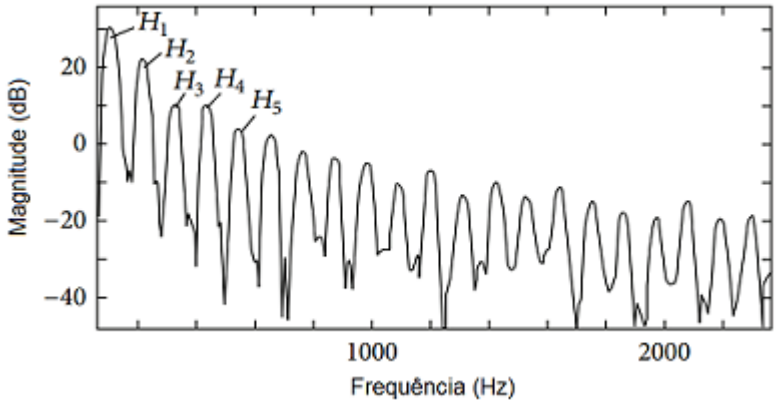

Figura 2.4: Espectro de um pulso glotal.

Existem duas dessas características do pulso glotal que interessam para este estudo.

O primeiro é $H_1 - H_2$ ou dH_{12} que é a diferença entre o primeiro e o segundo harmônicos da forma de onda da densidade espectral de frequência do sinal glotal, em decibel, e o segundo é o fator de riqueza harmônico (HRF), conforme é mostrado na Eq. 2.15, que é definido como a razão entre as somas das amplitudes dos harmônicos acima da frequência fundamental e a magnitude da frequência fundamental ou o primeiro harmônico em decibéis:

$$HRF = \frac{\sum_{r \geq 2} H_r}{H_1}. \tag{2.15}$$

Neste trabalho, todas as caracterísitcas citadas nesse capítulo foram estudadas, porém, apenas algumas dessas medidas foram empregadas na RNA. Será discutido mais sobre isso no Capítulo 6.

2.2 Gerando sinais de voz a partir do sinal glotal

Para a geração do sinal de voz artificial foram utilizados modelos matemáticos que simulam o sinal glotal. O processo de geração do sinal de voz consiste na convolução do sinal glotal com a resposta ao impulso correspondente ao trato vocal. Esse modelo é conhecido como modelo fonte-filtro [19] ilustrado na Fig.2.5.

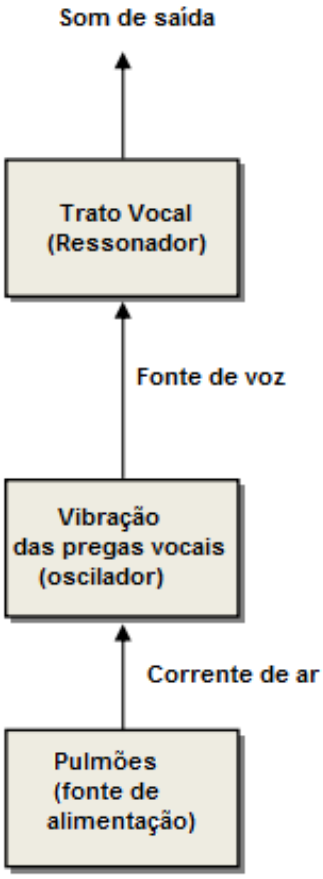

Figura 2.5: Modelo de produção de voz usado (fonte-filtro).

A fonte é o fluxo de ar que sai dos pulmões, passando pelo glote, e o filtro é o trato vocal, inserindo a radiação dos lábios/narinas.

Para gerar as vogais, foi considerada a frequência fundamental no valor de 98 Hz. Outro valor importante a definir são as frequências ou, simplesmente formantes que definem as vogais na modelagem do trato vocal. Dessa maneira, tem-se as formantes definidas na Tabela 2.1 para cada vogal.

Tabela 2.1: Formantes das vogais

Vogal	**F1 (Hz)**	**F2 (Hz)**	**F3 (Hz)**	**F4 (Hz)**	**F5 (Hz)**
a	900	1300	2000	2200	2500
e	450	1700	2000	2200	2310
i	300	1900	2100	2200	2490
o	500	800	2150	2200	2490
u	360	700	2170	2200	2330

O trato vocal é modelado como uma função de transferência que tem apenas pólos, correspondente aos formantes listados acima.

A radiação labial se refere à forma como o sinal se propaga na atmosfera, ou seja, é o efeito de filtragem que acontece quando o som sai da boca e é propagado em todas as direções do espaço. Ela é modelada como um filtro de resposta finita ao impulso (FIR), passa-alta e de primeira ordem. Sendo assim, tem a finalidade de ampliar as componentes de alta frequência do sinal e sua função de transferência está representada na Eq. 2.16:

$$R(z) = 1 - \alpha z^{-1} \tag{2.16}$$

onde α varia de 0,95 a 0,99.

Capítulo 3

Modelos matemáticos do sinal glotal

Os modelos matemáticos do sinal glotal utilizados neste trabalho foram os de Rosenberg [8] e Liljencrants-Fant (LF) [9]. Ao trabalhar com síntese de vozes reais, é necessário que o modelo se adeque da melhor maneira possível à realidade.

Primeiro, serão discutidos os modelos determinísticos. Depois, seguindo o que foi feito por [12], o intervalo de tempo glotal será considerado como um processo estocástico e, dessa forma, será gerado *jitter* no sinal de voz. Com isso, serão criados modelos estocásticos para o sinal de voz.

3.1 Modelo de Rosenberg

O modelo de Rosenberg é descrito pela Eq. (3.1):

$$U_g(t) = \begin{cases} A_v * 0.5[1 - cos(\pi * t/T_p)] & , \ 0 \leq t \leq T_p \\ A_v * cos(\pi * (t - T_p)/(2 * T_n)) & , \ T_p < t \leq T_p + T_n \\ 0 & , \ T_p + T_n < t \leq T_0 \, . \end{cases} \tag{3.1}$$

Esse é um modelo trigonométrico que utiliza duas funções senoidais para representar as fases de abertura e fechamento das cordas vocais, onde A_v é uma constante relacionada à amplitude do pulso, $T_p = \alpha_1 T_0$ é o tempo de abertura e $T_n = \alpha_2 T_0$ é o tempo de fechamento, sendo α_1 e α_2 parâmetros relacionados à porção do pulso ascendente e à porção do pulso descendente, respectivamente. Há uma relação direta estabelecida para este modelo e, para isso, basta tomar $T_p = \alpha_m O_q T_0$; e $T_n = T_e - T_p = (1 - \alpha_m) O_q T_0$. Sendo assim, conclui-se que $O_q = \alpha_1 + \alpha_2$; e $\alpha_m = \alpha_1/(\alpha_1 + \alpha_2)$.

Os parâmetros α_1 e α_2 têm relação direta com os parâmetros de tempo do sinal glotal.

Pela Eq.(3.1) é possível perceber que os tempos de abertura e fechamento do sinal glotal dependem destes paramêtros.

Na Fig. 3.1 pode-se ver um exemplo do sinal glotal determinístico de Rosenberg, os valores de parâmetros utilizados foram A_v=7, T_o=0,009564s, α_1=60, α_2=40.

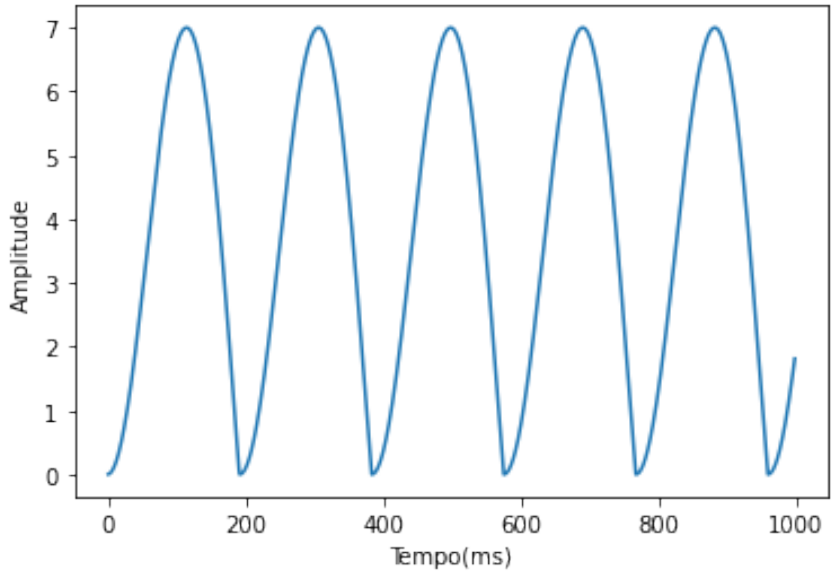

Figura 3.1: Trem de pulsos Rosenberg.

3.2 Modelo de Liljencrants-Fant (LF)

O modelo de LF proposto por [9] é representado pela Eq. (3.2). O modelo é composto por uma parte senoidal que é modulada por uma exponencial crescente (entre 0 e O_qT_0), seguida de uma fase de retorno exponencial decrescente (entre O_qT_0 e T_0).

$$U_g(t) = \begin{cases} -\dfrac{E_e e^{-aO_qT_0}}{sen\left(\frac{\pi}{\alpha_m}\right)\left(a^2+\left(\frac{\pi}{\alpha_m O_qT_0}\right)^2\right)}\left(\dfrac{\pi}{\alpha_m O_qT_0}+ae^{at}sen\left(\dfrac{\pi}{\alpha_m O_qT_0}t\right)-\dfrac{\pi}{\alpha_m O_qT_0}e^{at}cos\left(\dfrac{\pi}{\alpha_m O_qT_0}t\right)\right) & , \ 0 \le t < O_qT_0 \\ -E_e\left(\dfrac{1}{\varepsilon Q_a(1-O_q)T_0}-1\right)\left(T_0-t+\dfrac{1-e^{\varepsilon(T_0-t)}}{\varepsilon}\right) & , \ O_qT_0 \le t \le T_0 . \end{cases} \tag{3.2}$$

A Eq. 3.2 representa o modelo para o caso em que o fechamento das cordas vocais não é instantâneo ($Q_a > 0$), o parâmetro Q_a tem relação com a duração da fase de retorno através de $T_a = Q_a(1 - O_q)T_0$ e é conhecido como o coeficiente da fase de retorno. O parâmetro O_q é o quociente aberto e está relacionado com o instante de fechamento das

cordas vocais. O período fundamental é representando por T_0 e o coeficiente de assimetria por α_m.

É possível observar que os parâmetros Q_a e O_q têm influência sobre os parâmetros de tempo do sinal glotal, assim como α_1 e α_2 no modelo de Rosenberg, pois têm relação direta com os tempos de fechamento e abertura do sinal glotal.

O tipo de modelo do sinal glotal é de extrema importância, pois define a qualidade do sinal. O modelo de Rosenberg é mais simples em relação ao modelo LF, pois no segundo o fechamento das cordas vocais não ocorre de maneira tão abrupta, podendo ter maior controle de fase de fechamento, sendo assim, aproximando-se melhor das vozes reais.

Na Fig. 3.2 pode-se ver um exemplo do sinal glotal determinístico de LF, os valores de parâmetros utilizados foram A_v=7, T_o=0,009564s,α_m=0.75, O_q=0.8, Q_a=0.3.

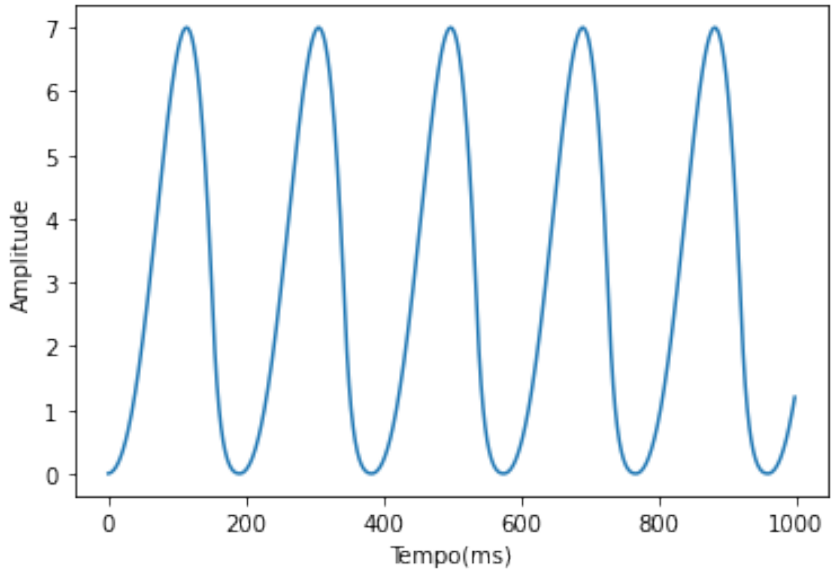

Figura 3.2: Trem de pulsos LF.

Comparando as Figs. 3.1 e 3.2, observou-se que no modelo LF é mais suave a transição entre os pulsos glotais.

3.3 Modelo estocástico do intervalo de tempo glotal

Há diversas formas de simular o *jitter* gerado pela voz humana. Neste trabalho, os modelos determinísticos de sinal glotal considerados são o modelo de Rosenberg [8] e o modelo de LF [9] com parâmetros unificados [21], como já citados anteriormente. Nestes modelos, o intervalo de tempo glotal será considerado como um processo estocástico. Para cada

modelo, duas densidades espectrais distintas são associadas ao processo estocástico [12]. Toda essa modelagem está cuidadosamente descrita na dissertação [12] e no artigo [13]. A teoria será apresentada aqui de forma sucinta.

Primeiro, será apresentado o modelo do processo estocástico associado ao intervalo glotal [13] e a correspondente densidade espectral de tempo, com dois casos considerados. A partir daí, quatro casos serão estudados:

- Modelo 1: Pulso glotal de Rosenberg e densidade espectral a 2 parâmetros;
- Modelo 2: Pulso glotal de Rosenberg e densidade espectral a 3 parâmetros;
- Modelo 3: Pulso glotal de LF e densidade espectral a 2 parâmetros;
- Modelo 4: Pulso glotal de LF e densidade espectral a 3 parâmetros.

O intervalo de tempo associado a um pulso glotal para vozes reais tem o seu comprimento variável, isto é, para cada pulso haverá um intervalo de tempo diferente, por isso pode-se considerar este intervalo como uma variável aleatória denominada T_g.

No caso determinístico, ao discretizar o sinal, o intervalo de tempo glotal pode ser dividido em N intervalos; isto é, $t_g = N\delta t = \sum_{i=1}^{N} \delta t$, sendo δt fixo. No caso aleatório, tem-se $T_g = \sum_{i=1}^{N} \Delta t(t_i)$, sendo $\Delta t(t)$ um processo estocástico.

Considera-se T_{g_i} cada instante glotal, com $i = 1, \ldots, N$. Assim,

$$T_{g_{i+1}} = T_{g_i} + \Delta t(t_i), \quad i = 1, ..., N. \tag{3.3}$$

$\Delta t(t_i)$ são amostras de uma realização do processo estocástico $\Delta t = \{\Delta t(t), t \in \mathbb{R}\}$, construído de acordo com as seguintes características.

1. Para todo t, $0 < \Delta t_0 \leq \Delta t(t)$, onde Δt_0 é uma constante positiva independente de t.
2. É um processo estacionário, não gaussiano, uma vez que só pode ter valores positivos.
3. $E\{(\Delta t(t))^2\} < +\infty$ para todo t (processo estocástico de segunda ordem), tal que $E\{\Delta t(t)\} = \underline{\Delta t(t)} > \Delta t_0 > 0$, sendo considerado contínuo em média-quadrática para garantir a existência de uma unidade espectral de potência.

Considere um processo estocástico real gaussiano de segunda ordem $Y = \{Y(t), t \in \mathbb{R}\}$ centrado e contínuo em média quadrática, estacionário e ergódico, fisicamente realizável. A representação do processo estocástico $\Delta t(t)$ pode ser escrita como:

$$\Delta t(t) = \Delta t_0 + (\underline{\Delta t} - \Delta t_0)(\underline{y} + Y(t))^2, \quad \forall t \in \mathbb{R}, \tag{3.4}$$

onde $\underline{y}$ é um parâmetro tal que:

$$E\{(\underline{y} + Y(t))^2\} = 1 \text{ e } E\{(\underline{y} + Y(t))^4\} < +\infty\,, \tag{3.5}$$

uma vez que $E\{\Delta t(t)\} = \underline{\Delta t}$ e $E\{(\Delta t(t))^2\} < +\infty$.

O processo estocástico gaussiano $Y(t)$ é construído como um filtro linear, $Y = h * N_\infty$, dado pela convolução do ruído branco gaussiano centrado N_∞, cuja função densidade espectral de potência é constante e igual a $S_{N_\infty}(\omega) = 1/(2\pi)$ para todo real w, pelo filtro $h = \mathfrak{F}^{-1}\{H\}$, que é a transformada inversa de Fourier da funçao resposta em frequência $H(w)$.

Dois casos de $H(\omega)$ serão considerados nesse trabalho:

1. $H(\omega) = \dfrac{a^2}{\omega^2 + b^2}$, dependendo de 2 parâmetros: a e b; e
2. $H(\omega) = \frac{a}{-w^2 + 2iw\xi b + b^2}$, dependendo de 3 parâmetros: a, b e ξ.

Baseado neste modelo estocástico, utilizou-se três parâmetros *(a,b,ξ)*, que estão associados às densidades espectrais, como saídas da rede neural desenvolvida. Os modelos a 2 parâmetros levam em consideração apenas os parâmetros a e b, já o modelo a 3 parâmetros faz a inclusão do parâmetro ξ [13].

De acordo com o estudo realizado em [12], estes parâmetros têm relação direta com as características de *jitter*, tanto modelo de Rosenberg como no de LF. Nas Figs. 3.3 e 3.4 pode-se observar o sinal glotal com *jitter*.

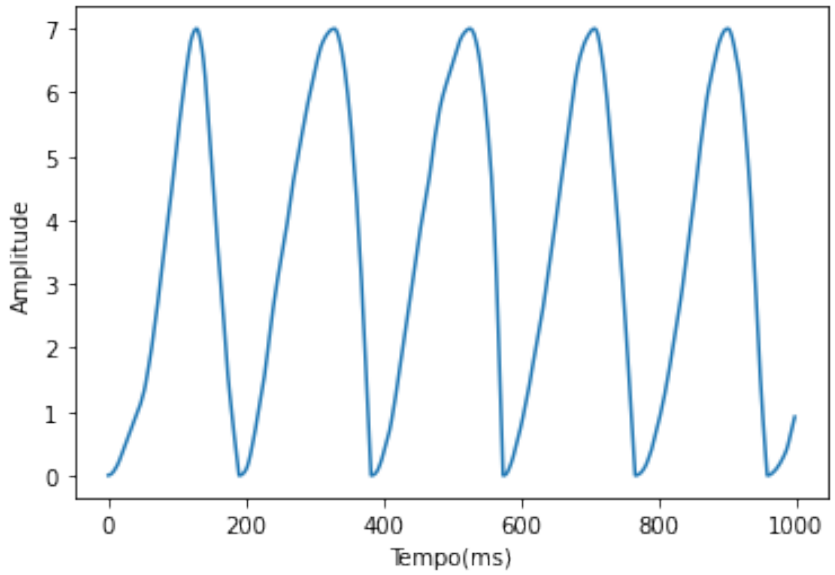

Figura 3.3: Trem de pulsos Rosenberg - Modelo 1 (a=10, b=300)

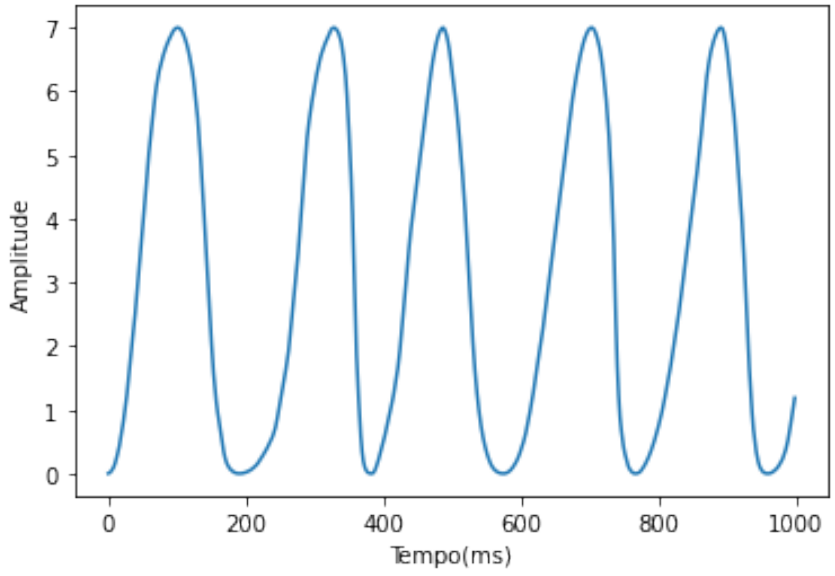

Figura 3.4: Trem de pulsos LF - Modelo 3 (a=14, b=100)

Capítulo 4

Redes Neurais Artificiais

O tema Inteligência Artificial (IA) nunca foi tão relevante quanto na última década [22]. Há um crescente interesse em entender e implementar soluções utilizando IA. Dentro do campo de IA existem algumas ferramentas para implementação da tecnologia, uma das principais é o Aprendizado de Máquina (Machine Learning - ML) [23]. Neste trabalho, serão utilizadas Redes Neurais Artificiais que são um tipo de Aprendizado de Máquina.

As Redes Neurais Artificiais (RNAs) já existem desde a década de 1950 [24] [25], e embora a arquitetura tenha evoluído ainda faltavam componentes para que a tecnologia avançasse, um destes componentes é o grande volume de dados (*Big Data*) [26]. O grande volume de dados que são gerados atualmente em um curto período de tempo permitiu que as redes atingissem alto nível de precisão. Aliando o *Big Data* ao desenvolvimento de programação paralela utilizando GPUs, para dar conta do processamento do volume grande de dados, foi possível chegar ao nível de evolução atual nos modelos de ML e, de fato, colocar o termo Inteligência Artifical em prática.

RNAs são basicamente um conjunto de algoritmos que tentam imitar o funcionamento do cérebro humano. Tarefa bem difícil devido à complexidade dessa máquina. O cérebro se desenvolve através de experiências adquiridas em situações vividas e, por isso, os modelos são baseados em treinamentos com dados conhecidos, ou seja, experiências vividas. Espera-se que o mesmo aprenda com elas. Os neurônios são as principais unidades que o cérebro utiliza para transmitir e processar informações.

4.1 O Perceptron - neurônio matemático

O neurônio é formado de 3 partes principais: **corpo celular**, que possuem ramificações chamadas **dendritos** e outra ramificação mais extensa, denominada **axônio**. Nas extremidades dos axônios encontram-se os nervos terminais, por eles é possível realizar a transmissão das informações para outros neurônios. Esta transmissão é chamada de sinapse, como pode ser observado na Fig. 4.1 [27].

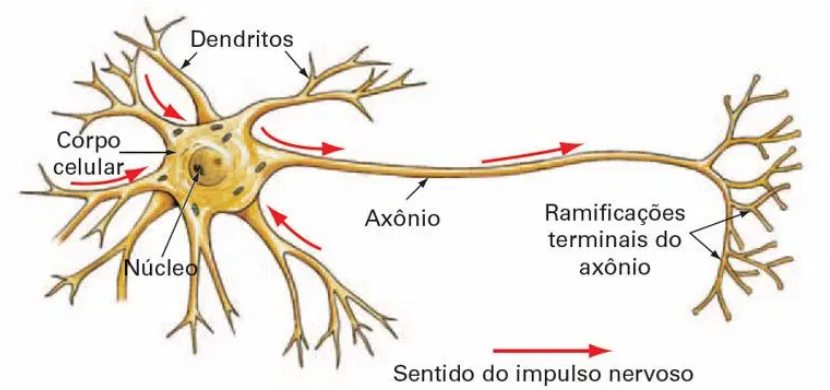

Figura 4.1: Representação de um neurônio biológico.

O cérebro é composto por bilhões de neurônios que estão conectados entre si formando uma enorme rede de comunicação, a rede neural. É importante observar que o corpo e os dentritos formam a superfície de entrada do neurônio, e o axônio a superfície de saída da informação. As informações transmitidas pelos neurônios são na verdade impulsos elétricos, ou seja, são a propagação de um estímulo ao longo dos neurônios que pode ser qualquer sinal captado pelos receptores nervosos.

A partir do conhecimento da estrutura e do funcionamento do neurônio biológico, pesquisadores tentaram simular este sistema em computador. Basicamente, um neurônio matemático de uma rede neural artificial é um componente que calcula a soma ponderada de várias entradas, aplica uma função de ativação e passa o resultado adiante.

As funções de ativação são de extrema importância nas redes neurais, pois são elas que decidem se o neurônio deve ser ativado. Ou seja, se a informação que o neurônio recebeu é relevante para seguir adiante ou se deve ser ignorada.

Este neurônio matemático é conhecido como Perceptron. O corpo da célula é representado pela soma das entradas multiplicadas pelos pesos sinápticos e, no final, pela função de ativação que definirá qual será a saída. O axônio é representado pela saída obtida da função de ativação. Da mesma maneira que no modelo biológico, os estímulos podem ser

de excitação ou inibição, representados por pesos positivos ou negativos, respectivamente.

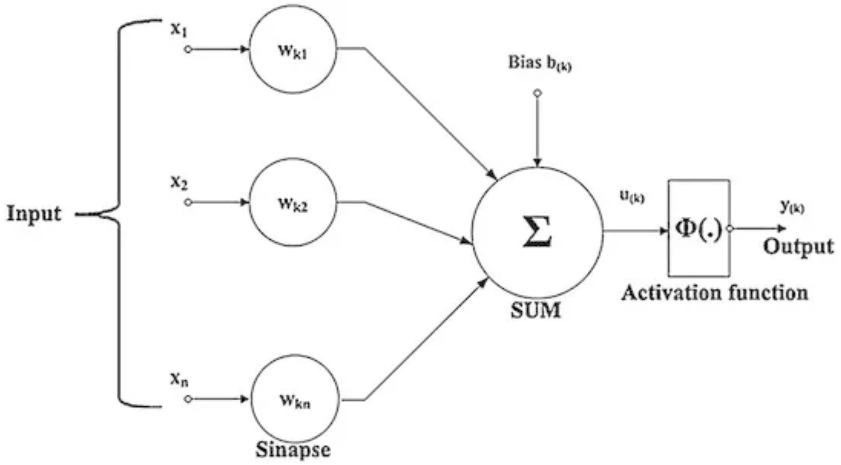

Figura 4.2: Representação de um neurônio matemático (Perceptron).

Como mostrado na Fig. 4.2 [27], as entradas são x_{k1}, x_{k2}, ..., x_n. Os pesos sinápticos que serão aplicados na entrada são representados como w_{k1}, w_{k2}, ..., w_{kn}.

O peso representa a importância de uma entrada, pois quanto maior o peso desta entrada, mais ela influenciará na rede neural. Isso significa que o peso decide com que rapidez a função de ativação será ativada. O *Bias* é usado para ajustar a saída junto da soma ponderada das entradas para o neurônio, ou seja, é uma constante que ajuda o modelo a se adaptar aos dados fornecidos.

Os pesos utilizados na rede neural precisam ser iniciados de alguma maneira, pois isso faz com que a rede aprenda mais rápido. Neste trabalho, a inicialização dos pesos de cada camada foi feita utilizando a distribuição normal.

Outro parâmetro que pode-se controlar na Rede Neural é a sua taxa de aprendizagem, ou seja, quanto devo variar meus pesos após as iterações para chegar no menor valor de erro possível. Esta taxa pode ser definida na inicialização da rede e quanto maior o seu valor então mais rápido chegará no peso ideal, porém se o valor for muito grande é possível ter problemas no processo de aprendizado.

Existem vários tipos de funções de ativação, e cada uma deve ser aplicada em determinada situação. A função de ativação que melhor se ajustou aos dados deste trabalho foi a ReLU. Esta função foi apresentada pela primeira vez em [28]. Desde então se popularizou e tem sido umas das funções mais utilizadas para RNAs.

A função ReLU é a unidade linear rectificada. É definida como:

$$f(x) = max(0, x) \tag{4.1}$$

Pela Eq.4.1, pode-se perceber que a função irá retornar o valor máximo no intervalo entre 0 e o dado de entrada, ou seja, se a entrada for negativa, ela será convertida em zero e o neurônio não será ativado. Isso significa que, ao mesmo tempo, apenas alguns neurônios são ativados, tornando a rede esparsa, eficiente e fácil para a computação.

Um único perceptron, assim como um único neurônio, não é capaz de resolver os problemas complexos de RNAs. Por isso, será utilizada uma arquitetura de rede neural de forma a tornar o modelo robusto para realizar as identificações dos parâmetros.

4.2 Arquitetura das Redes Neurais Artificiais

Uma das arquiteturas mais utilizadas para trabalhar com sinais de voz é a *Long Short Term Memory* (LSTM), ela se baseia em uma arquitetura de Rede Neural Recorrente que memoriza valores em intervalos aleatórios. A LSTM funciona muito bem para prever séries temporais com intervalos de tempo de duração desconhecida. Porém, neste trabalho o objetivo é identificar parâmetros de modelos estocásticos do sinal glotal, que não tem o comportamento de séries temporais.

A arquitetura da RNA utilizada neste trabalho foi a Perceptron de Múltiplas Camadas (Multilayer Perceptrons - MLP) [29] [30]. Como já explicado anteriormente, um Perceptron é um classificador linear e possui uma única saída com base em várias entradas de valor real. Como pode-se ver na Fig. 4.3, o MLP é uma arquitetura composta por uma camada de entrada, camadas ocultas formadas por mais de um perceptron ligados entre si e, por fim, a camada de saída.

Os MLPs são muito utilizados em problemas de aprendizado supervisionado, que é o caso deste trabalho. O que caracteriza um aprendizado supervisionado é possuir o conhecimento prévio dos dados reais de saída da rede. A rede treina um conjunto de pares entrada-saída e aprende a modelar a correlação (ou dependências) entre essas entradas e saídas.

O treinamento envolve o ajuste dos parâmetros, ou os pesos e bias, do modelo para minimizar o erro. A rede executa dois movimentos: ida e volta. Na ida, o sinal se move da camada de entrada até a camada de saída e na volta, usando o *backpropagation* faz os ajustes dos pesos e de bias em relação ao erro, e o próprio erro pode ser medido de várias

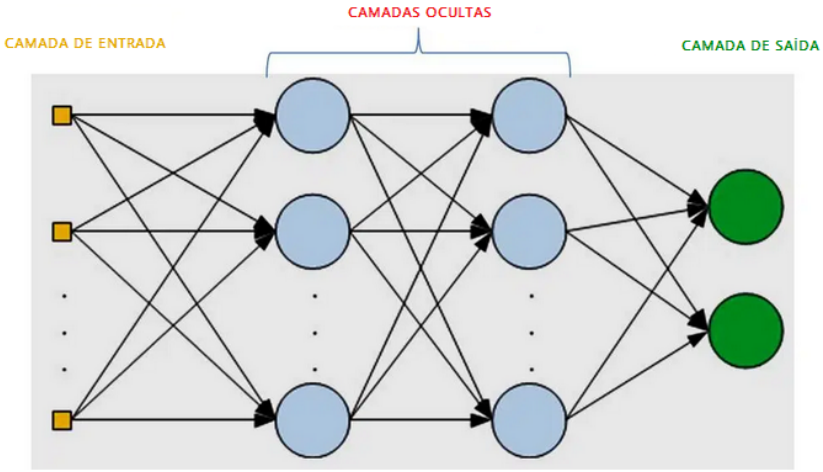

Figura 4.3: Arquitetura MLP da RNA.

maneiras. A rede repete estes movimentos até que o erro não possa mais ser reduzido, pois atingiu seu valor mínimo, ou seja, o estado de convergência.

O algoritmo de *backpropagation* [31] [32] é composto de dois passos: Primeiro o passo para frente (*forward pass*), onde as entradas são passadas através da rede e as previsões de saída obtidas, depois o passo para trás (*backward pass*), onde calcula-se o gradiente da função de perda na camada de previsão da rede e usa-se esse gradiente para aplicar recursivamente a regra da cadeia para atualizar os pesos na rede (etapa também conhecida como retro-propagação).

O gradiente estocástico descendente (SGD) [33] é um algoritmo que encontra, de maneira iterativa, os valores dos pesos e bias que minimizam a função de interesse, como o erro entre os dados originais e os previstos.

Sendo assim, para minimizar estes erros, foi utilizado o otimizador chamado Adam, que é uma variação do SGD. Sua principal inovação é o cálculo não só das taxas de aprendizagem individuais para cada parâmetro, mas também do primeiro e segundo momento dos gradientes. Este otimizador é largamente utilizado em modelos de *machine learning*.

Os modelos de *machine learning* são funções de parâmetros que mudam durante o treinamento, como o peso e o *bias*, e também de hiperparâmetros que são configurados antes do treinamento, como a taxa de aprendizado e a função de ativação. A quantidade de épocas é o número de vezes em que o modelo realizará o treinamento, ou seja, quantas passagens completas do conjunto de dados devem ser feitas. Para isso foi utilizado o *Early Stopping* (Parada antecipada) [34], método disponível na biblioteca do *Keras*, ao usar este recurso é monitorada a perda nos dados de treino e validação. O *Keras* é uma

biblioteca de rede neural de código aberto escrita em Python. Quando a perda parar de melhorar, termina-se o treinamento. Em particular, isso significa que não é preciso se preocupar em descobrir explicitamente como o número de épocas depende dos outros hiperparâmetros, pois isso é feito automaticamente. Além disso, a Parada Antecipada também impede automaticamente o *overfitting* [35]. Será falado mais sobre este tema no próximo capítulo.

Capítulo 5

Identificação dos parâmetros dos modelos

Neste capítulo será descrita toda a metodologia utilizada para a identificação dos parâmetros dos modelos e a utilização das métricas e estratégias para análise dos resultados obtidos.

5.1 Geração dos dados

Para a criação da base de dados utilizada na RNA foram geradas 200 amostras do sinal glotal correspondente à emissão de uma vogal com tempo de sustentação de 3 segundos. Os sinais foram gerados para todos os modelos considerados.

Estes modelos serão referenciados como: Modelo 1, Modelo 2, Modelo 3 e Modelo 4 como descrito no Capítulo 3, de acordo com o modelo matemático do sinal glotal (Rosenberg ou LF) e da densidade espectral associada ao processo estocástico do intervalo de tempo glotal (2 ou 3 parâmetros). Sendo assim, no total tem-se quatro tipos de sinal glotal a serem estudados.

Pode-se destacar alguns passos que ajudaram a avaliar qual seria o melhor intervalo de valores para os parâmetros do modelo estocástico a serem utilizados na geração da base de dados:

Passo 1 - A sensibilidade do sinal glotal é analisado ao variar os parâmetros a e b, de acordo com os valores nas Tabelas 6.1 e 6.3, para os modelos 1 e 3. Dessa maneira, entende-se como estes parâmetros influenciam na geração do sinal glotal.

Passo 2 - Os valores dos parâmetros a, b são fixados em 100 e 1000000, respectivamente. Varia-se o parâmetro ξ nos modelos 2 e 4 para observar sua influência no sinal

glotal.

Passo 3 - Foi definido o intervalo de valores dos três parâmetros citados acima (a, b e ξ) que deram amostras do sinal glotal com um valor considerável de jitter para cada modelo.

Passo 4 - Foram incluídos os parâmetros do modelo estocástico que influenciam as características de tempo extraídas do sinal glotal. São eles: α_1 e α_2, O_q e Q_a, para os modelos de Rosenberg e LF, respectivamente. Os valores para estes parâmetros também foram definidos experimentalmente.

Passo 5 - Geração do banco de dados dos sinais glotais usando os parâmetros definidos.

Passo 6 - Para cada amostra de sinal glotal gerado foi desenvolvido um código para a extração das suas características. São elas: de tempo (OQ, SQ, CIQ, QOQ, OQa), *jitter* (absoluto, relativo, rap, ddp e ppq5) e frequência (dH12 e HRF) explicados no Capítulo 2.

Os valores finais utilizados para os parâmetros dos modelos estão descritos nas Tabelas 6.1, 6.2, 6.3 e 6.4.

Desta forma, os parâmetros de cada modelo estocástico serão identificados utilizando as características extraídas do sinal glotal como entrada da RNA (explicada a seguir). No Capítulo 6 será explicado como foram utilizadas estas combinações de características. De acordo com a teoria dos modelos, já era esperado que as características de *jitter* teriam maior influência sobre os parâmetros a, b e ξ, e as características de tempo sobre os parâmetros α_1, α_2, O_q e Q_a.

5.2 Construção da RNA

A partir das características extraídas do sinal glotal, deseja-se identificar os parâmetros a, b, ξ, α_1, α_2, O_q e Q_a dos modelos estocásticos. Para isso, foram testados alguns frameworks/bibliotecas em python e, por fim, foi desenvolvida uma RNA utilizando o *keras* [36]. A mesma fornece uma variedade de camadas para a construção da arquitetura de rede neural ideal para o trabalho, completamente costumizável. Algumas arquiteturas de rede foram testadas e a que teve o melhor desempenho foi a MLP (explicado no Capítulo 4). Dentro desta arquitetura também foram testadas diferentes configurações da RNA.

Após ser definido que o modelo utilizado para a criação da RNA seria o Perceptron

multicamadas (Multi Layer Perceptron - MLP), os próximos passos foram as configurações desta rede. Foi preciso definir a função de ativação, o otimizador e qual a taxa de aprendizagem que seria utilizada. Todos estes conceitos já foram explicados no Capítulo 4. Além disso, támbém foi preciso definir o número de épocas para o treinamento, ou seja, quantas passagens completas do conjunto de dados devem ser feitas. Outra configuração importante a ser feita é a definição do número de neurônios e de camadas ocultas do modelo.

As características de tempo, *jitter* e frequência calculadas a partir do sinal glotal são os dados de entrada da RNA, pois serão utilizadas para realizar a identificação dos parâmetros do modelo estocástico, como já ilustrado na Fig. 1.1.

5.3 Pré-processamento dos dados

A etapa de pré-processamento dos dados é uma das mais importantes quando se trata de aprendizado de máquina. Muitas técnicas podem ser aplicadas de acordo com os dados e o problema que se deseja resolver.

No caso deste trabalho, há parâmetros a serem identificados pela rede que possuem ordens de grandezas com diferenças bem significantes. Por isso, os dados serão normalizados e isso fará com que o cálculo do erro seja mais coerente e ajudará no desempenho da rede para realizar a identificação dos parâmetros. A normalização dos dados segue a Eq. 5.1:

$$y_{normalizado} = \frac{y}{|y_{max}|} \tag{5.1}$$

onde $y_{normalizado}$ é o vetor da saída normalizada, y_{max} é o valor máximo do vetor de saída e y é o vetor com os valores reais. Com isso, todos os valores estarão entre -1 e 1.

5.4 Estratégias para a geração e análise dos resultados

Devido à grande quantidade de características do sinal glotal que foram calculadas, foi necessário estabelecer um critério para a combinação destas características na entrada da RNA para cada parâmetro identificado de cada um dos modelos. Sabe-se que alguns parâmetros do modelo possuem uma maior correlação com as características de *jitter* e outros com as de tempo. Para auxiliar de maneira mais assertiva nesta combinação de

características foi gerada a matriz de correlação dos dados, que indica o grau de relação entre as características e os parâmetros, podendo variar entre -1 e 1. Quanto mais perto de 1 maior será a correlação entre eles. Na Figura 5.1, pode-se ver um exemplo da matriz de correlação.

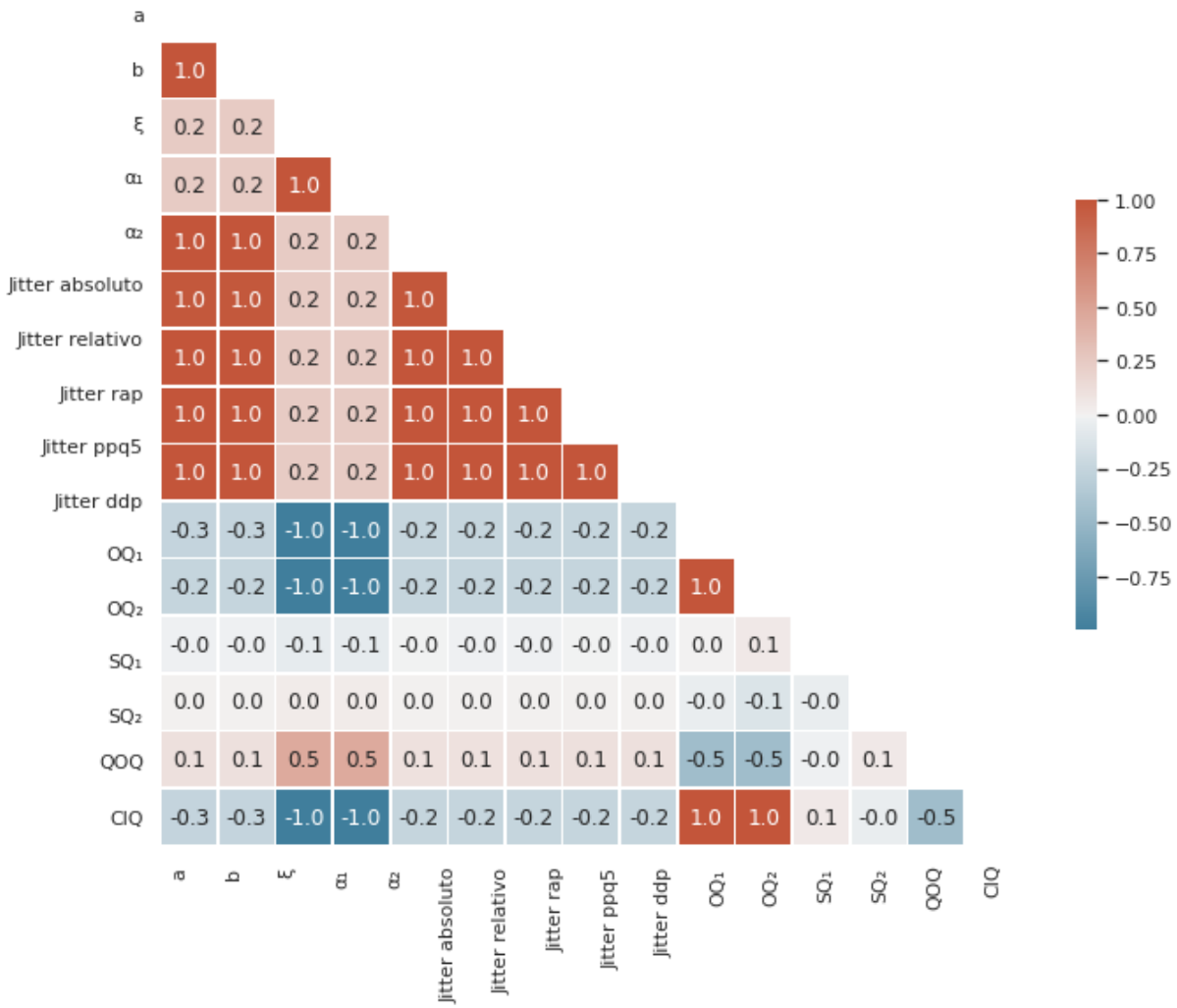

Figura 5.1: Matriz de correlação entre os parâmetros do Modelo 1 e as características extraídas do sinal glotal.

Nos eixos é possível observar as características calculadas do sinal glotal como mostrado no Capítulo 2 e os parâmetros do modelo que deseja-se identificar. Neste exemplo pode-se perceber pela primeira coluna da matriz que o parâmetro a tem alta correlação com todas as características de *jitter* e pouca correlação com as de tempo. Assim como α_1 tem correlação alta com as características de tempo e pouca correlação com as características de *jitter*.

A partir deste estudo na matriz de correlação foi possível definir quais características seriam utilizadas na entrada da rede para previsão dos parâmetros do modelo estocástico.

Como já dito anteriormente, foram geradas 200 amostras de sinal glotal em cada modelo. Destes 200 áudios, 80% foram usados para treinar o modelo e 20% foram utilizados para teste. E, dentro do universo dos 80% foram retirados 20% dos dados para validação.

Foi utilizada a técnica de *shuffle* (embaralhar) para garantir que os dados tenham bastante variação de valores, evitando assim que uma parte dos dados não esteja representada no conjunto de treino.

Um dos métodos mais utilizados para a avaliação do resultado de desempenho de uma rede neural são as curvas de aprendizado. Para este trabalho foram utilizadas as curvas de aprendizado para os dados de treino e validação, como ilustrado na Fig. 5.2.

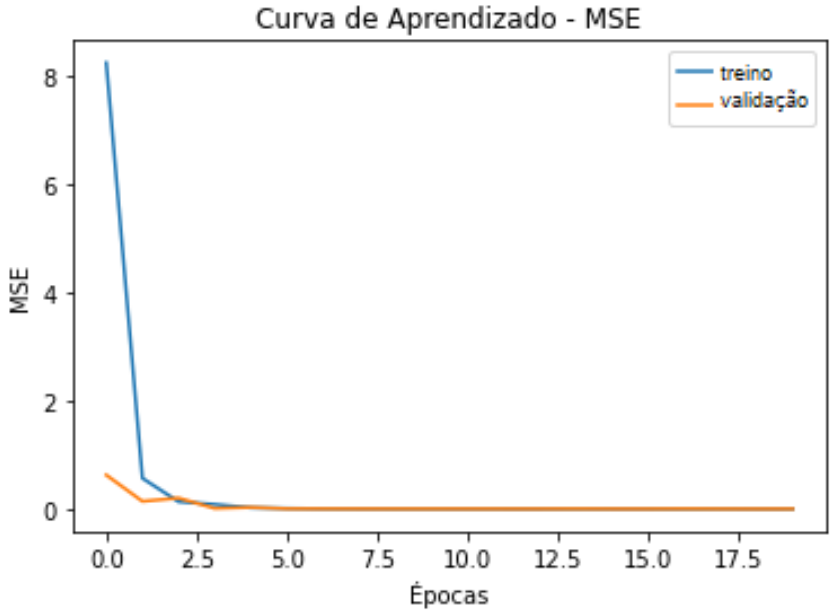

Figura 5.2: Exemplo de Curva de aprendizado para uma Rede Neural

No gráfico da Fig. 5.2 tem-se o valor do erro médio quadrático (MSE) de cada curva no eixo das ordenadas e as épocas de treinamento nos eixos das abscissas. A Eq. 5.2 mostra o cáculo deste erro.

$$MSE(y, \bar{y}) = \frac{1}{n_{amostras}} \sum_{i=0}^{n_{amostras}-1} (y_i - \bar{y}_i)^2 \tag{5.2}$$

onde y é o valor real da base de dados que será previsto, $\bar{y}$ é o valor previsto pela rede neural e $n_{amostras}$ é o número total de amostras.

Neste gráfico, pode-se observar que o erro para o conjunto de treino começa muito alto mas ao longo do treinamento e com maior números de épocas as curvas convergem para o mesmo ponto, é o que se espera em um modelo bem treinado. Como está sendo

utilizada uma métrica de erro, quanto menor seu valor melhor será o desempenho na rede.

Outra informação importante que pode ser retirada deste gráfico é em relação à qualidade do treinamento do modelo, isto porque, as curvas de validação e treino indicam se o modelo está realmente aprendendo, detectando o caso de *overfitting* (sobre-ajuste).

O *overfitting* ocorre quando o modelo tem um desempenho muito bom para os dados do conjunto de treino, porém ao utilizar os dados de validação o resultado é ruim, ou seja, a curva de validação tem um desempenho pior em relação à curva de treino porque o modelo não consegue generalizar o seu aprendizado, ele apenas decorou os dados de treino.

Após a validação dos dados é realizada a etapa de testes. Nesta etapa serão utilizados 20% dos dados separados inicialmente do conjunto de treino, ou seja, estes dados nunca foram utilizados anteriormente pelo modelo. Estes dados são submetidos ao modelo gerado pela base de treinos e, a partir dos valores de previsão obtidos neste conjunto de testes foi possível analisar a qualidade dos resultados. Para isso a métrica utilizada também será o erro quadrático médio, mostrado na Eq. 5.2. Foi considerado como resultado satisfatório valores abaixo de 10% em termos de taxa de erro.

Desta forma, tem-se todos os insumos necessários para analisar o desempenho do modelo, para isso foram feitas as análises representadas acima, no capítulo a seguir serão abordados os resultados mais significantes e o trabalho com estas métricas.

Capítulo 6

Resultados

Neste capítulo serão apresentados e analisados os resultados obtidos de acordo com a metodologia desenvolvida no Capítulo 5. Os resultados serão apresentados separadamente de acordo com os modelos e parâmetros utilizados:

- Modelo 1: Pulso glotal de Rosenberg e densidade espectral a 2 parâmetros;
- Modelo 2: Pulso glotal de Rosenberg e densidade espectral a 3 parâmetros;
- Modelo 3: Pulso glotal de LF e densidade espectral a 2 parâmetros;
- Modelo 4: Pulso glotal de LF e densidade espectral a 3 parâmetros.

6.1 Rede Neural Artificial usada

Para a obtenção dos resultados apresentados neste capítulo foi utilizada a arquitetura de Rede Neural chamada de MLP. Dentro desta rede existem algumas configurações que foram ajustadas:

- Função de ativação: relu, que é a função de unidade linear retificada, que retorna $f(x) = max(0, x)$;
- Otimizador: Adam, é uma extensão popular para o gradiente estocástico descendente (SGD) ;
- Taxa de aprendizado: 0.001;
- Número de camadas ocultas: 4;

- Número de perceptrons, neurônio matemático, em cada camada oculta: 32;
- Épocas: Utilizou-se o *Early Stopping* para definir a melhor quantidade de épocas.

Na Fig.6.1 tem-se uma ilustração de como a RNA ficou estruturada.

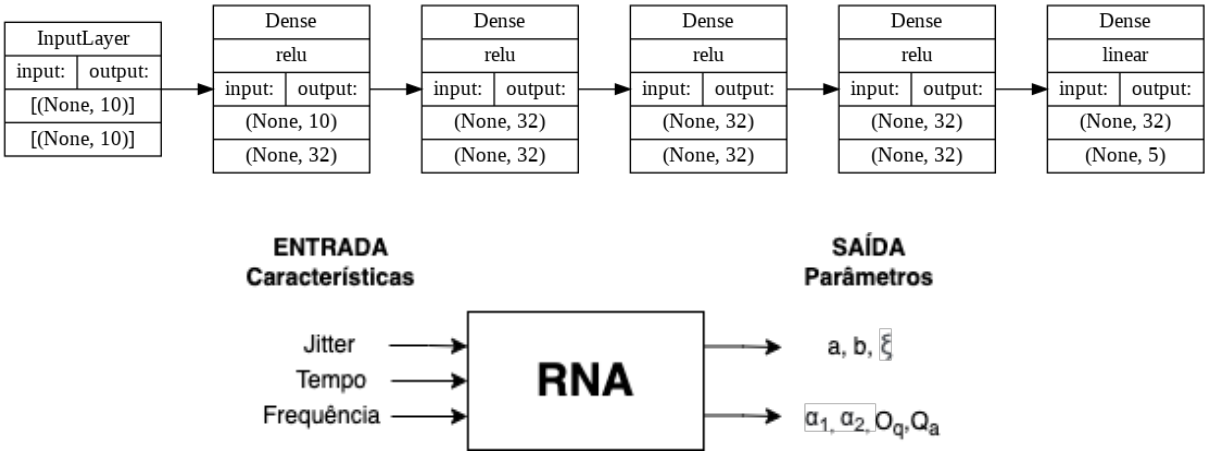

Figura 6.1: Diagrama de blocos da RNA usada.

A primeira linha corresponde ao tipo de camada utilizada. Para as camadas do tipo densas também há a especificação da função de ativação utilizada. Por fim, as duas últimas linhas correspondem à dimensão de entrada e saída, respectivamente. A dimensão *None* significa que esta é variável. Sendo assim, ela não afeta o tamanho da rede, apenas denota que é possível selecionar o número de amostras de sua entrada durante o teste/treinamento da RNA.

Todos os parâmetros citados acima foram definidos de forma experimental. De acordo com [37], muitos problemas de otimização podem ser soluciandos utilizando o algoritmo Adam. O valor da taxa de aprendizado costuma variar entre 0.1, 0.01 e 0.001 e neste trabalho o valor que retornou o melhor resultado foi o de 0.001. Os números de camadas e neurônios foram variados até ser encontrada uma combinação que funcionasse bem com os dados de todos os quatro modelos estudados. Por fim, foi utilizado o *Early Stopping*, método disponível na biblioteca do *Keras* para monitorar a função de custo nos dados de treino e validação. Este método controla o número de épocas até o momento em que a rede para de aprender, evitando assim o *overfitting*. Como já explicado, o *overfitting* ocorre quando a rede tem um ótimo desempenho para o conjunto de treino, mas ao utilizar o conjunto de validação o desempenho não fica tão satisfatório, ou seja, a rede não aprendeu.

6.2 Base de dados

Conforme as Tabelas 6.1, 6.2, 6.3 e 6.4, tem-se as variações dos parâmetros do modelo estocástico, de acordo com o modelo utilizado. Algumas dessas variações foram definidas segundo o estudo prévio em [12] e outras foram definidas de maneira experimental.

Em particular, o parâmetro b está relacionado com o expoente da parte exponencial e influencia diretamente a convergência da EDO estocástica. Por esse motivo, os valores são tão maiores que a, que é relacionado com a condição inicial e, assim, à solução particular da EDO estocástica.

O cálculo do erro do modelo é feito considerando todas as saídas da rede e, por isso, foi preciso normalizar as saídas, visto que principalmente os parâmetros a e b têm uma grande diferença em suas ordens de grandeza. Além disso, a normalização ajuda a rede neural a melhorar a sua previsão, dado que a variação das amostras fica em torno de -1 e 1.

Tabela 6.1: Variação dos parâmetros para o Modelo 1.

Parâmetros	**Valor inicial**	**Passo**	**Valor final**
a	100	2	500
b	1000000	20000	5000000
α_1	50	1	100
α_2	20	1	70

Tabela 6.2: Variação dos parâmetros para o Modelo 2.

Parâmetros	**Valor inicial**	**Passo**	**Valor final**
a	100	2	500
b	1000000	20000	5000000
ξ	0.1	0.1	20
α_1	50	1	100
α_2	20	1	70

Tabela 6.3: Variação dos parâmetros para o Modelo 3.

Parâmetros	**Valor inicial**	**Passo**	**Valor final**
a	100	2	500
b	1000000	20000	5000000
O_q	0.1	0.1	0.4
Q_a	0.1	0.1	0.5

Tabela 6.4: Variação dos parâmetros para o Modelo 4.

Parâmetros	**Valor inicial**	**Passo**	**Valor final**
a	100	2	500
b	1000000	20000	5000000
ξ	0.1	0.1	20
O_q	0.2	0.1	1
Q_a	0.1	0.1	0.9

Com as variações dos parâmetros apresentadas nas tabelas, foram geradas 200 amostras do sinal de voz. Para cada sinal de voz foram extraídas as características que servirão de entrada para RNA, conforme já discutido, para a solução do problema inverso.

É importante destacar que, nesse momento, todo o desenvolvimento está controlado, isto é, o banco de dados criado e as escolhas para a resolução do problema inverso são bem definidas. Depois, será apresentada a discussão de um caso para voz real.

6.3 Modelo 1

Para realizar a identificação dos parâmetros e escolher parâmetros de entrada que mais influenciaram as características dos sinais, foi analisada a matriz de correlação apresentada na Fig. 6.2.

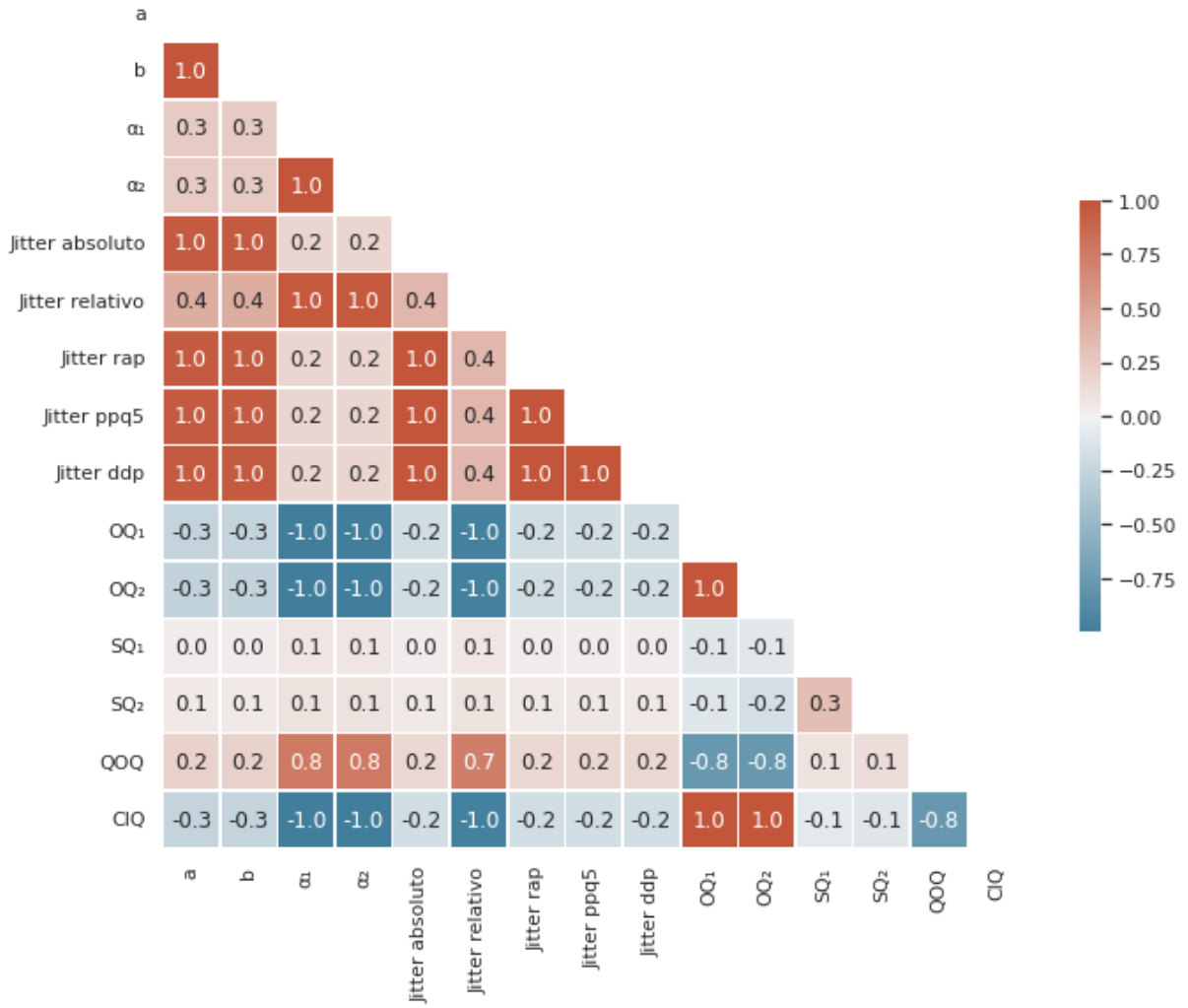

Figura 6.2: Matriz de correlação do Modelo 1.

A matriz de correlação indica o grau de relação entre as variáveis. Quanto mais próximo de 1 maior será a correlação entre elas. É possível observar na matriz acima que as características que obtiveram os melhores resultados de correlação foram as de *jitter*. Por isso, combinações destas medidas, juntamente com as características de tempo, foram utilizadas como entradas da rede e os melhores resultados foram obtidos com a combinação das seguintes características: *jitter* absoluto, *jitter* relativo, *jitter* rap, *jitter* ppq5, *jitter* ddp e QOQ.

Após definir as características de entrada, foi realizado o treinamento da rede e a curva de aprendizado para os dados de treino e testes está ilustrada na Fig. 6.3.

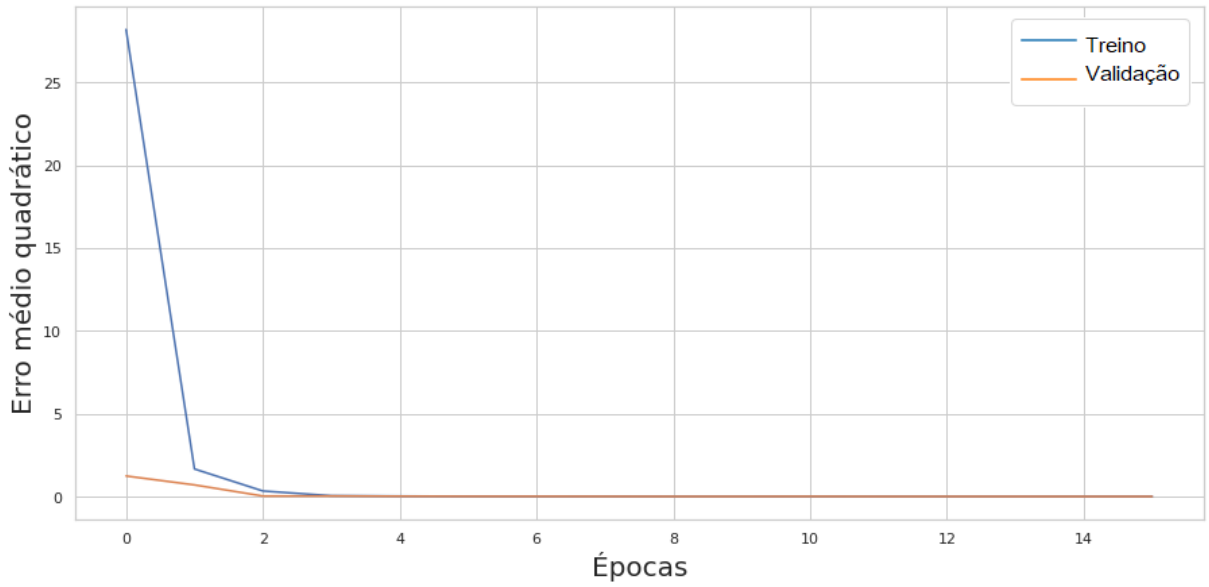

Figura 6.3: Curva de aprendizado para o Modelo 1.

De acordo com a Fig. 6.3, pode-se observar que a partir da quarta época o erro de treino e validação são praticamente iguais. O melhor resultado de MSE foi obtido na época 16, onde o erro foi igual a 1,2% para o conjunto de treino e 1,2% para o conjunto de validação. Não foi observado problemas de *overfitting* neste modelo.

Para o conjunto de testes, o valor do MSE ficou em torno de 1,2%. O resultado para o conjunto de testes foi o mesmo do conjunto de validação, ou seja, a identificação dos parâmetros ficou muito boa.

6.4 Modelo 2

Na Fig. 6.4, tem-se a matriz de correlação dos parâmetros do modelo com as características extraídas do sinal glotal.

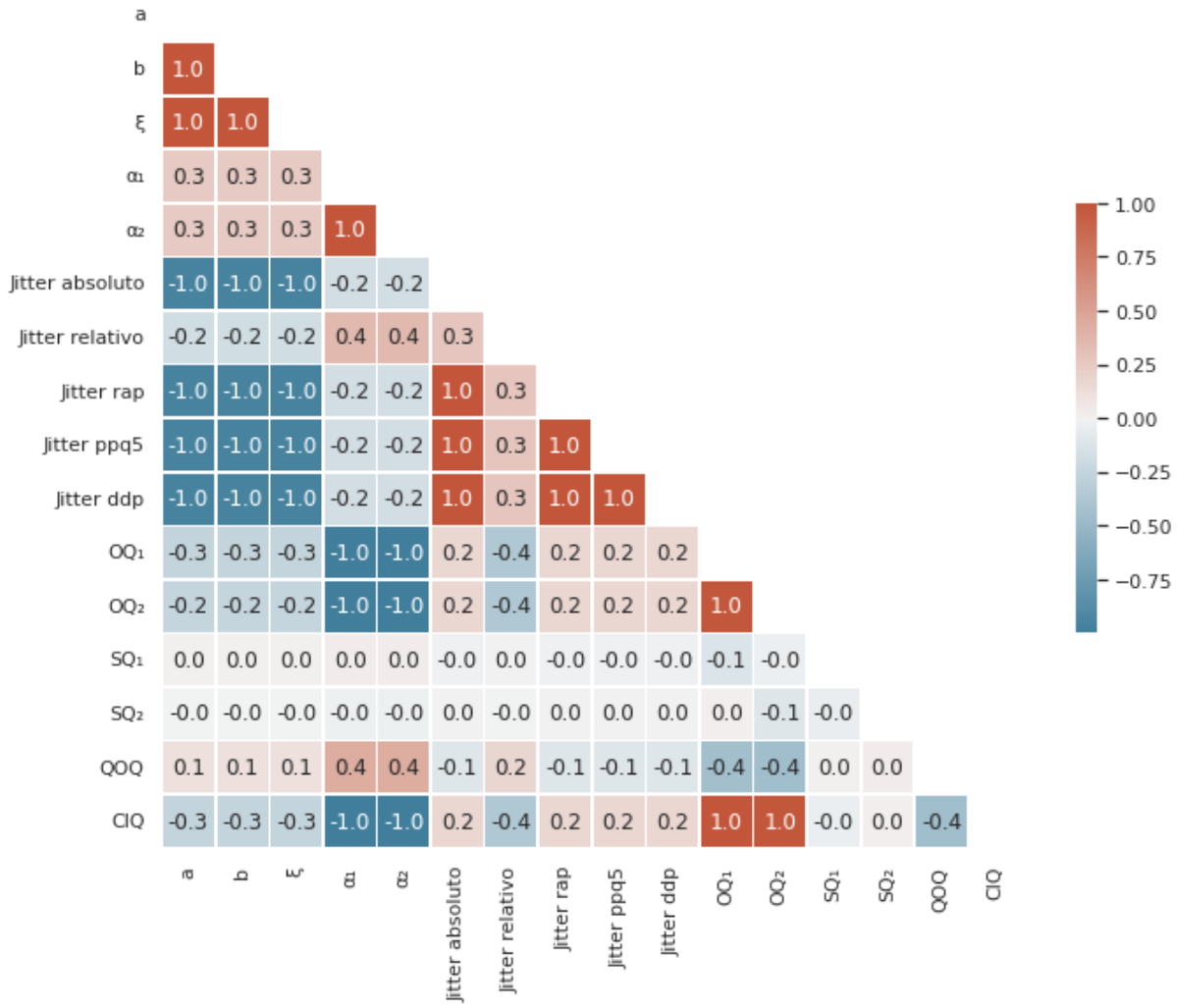

Figura 6.4: Matriz de correlação Modelo 2.

Apesar da característica QOQ ter maior correlação do que as outras características de tempo, ao utilizar apenas ela juntamente com os parâmetros de *jitter* os resultados não foram satisfatórios. O erro ficou maior ao treinar a rede apenas com esta característica. Neste caso, foram utilizadas todas as características de *jitter* e de tempo como entrada da rede, são elas: *Jitter* absoluto, relativo, ddp e ppq5, OQ_1, OQ_2, SQ_1, SQ_2, QOQ, CIQ. A curva de aprendizado para os dados de treino e testes está ilustrada na Fig. 6.5.

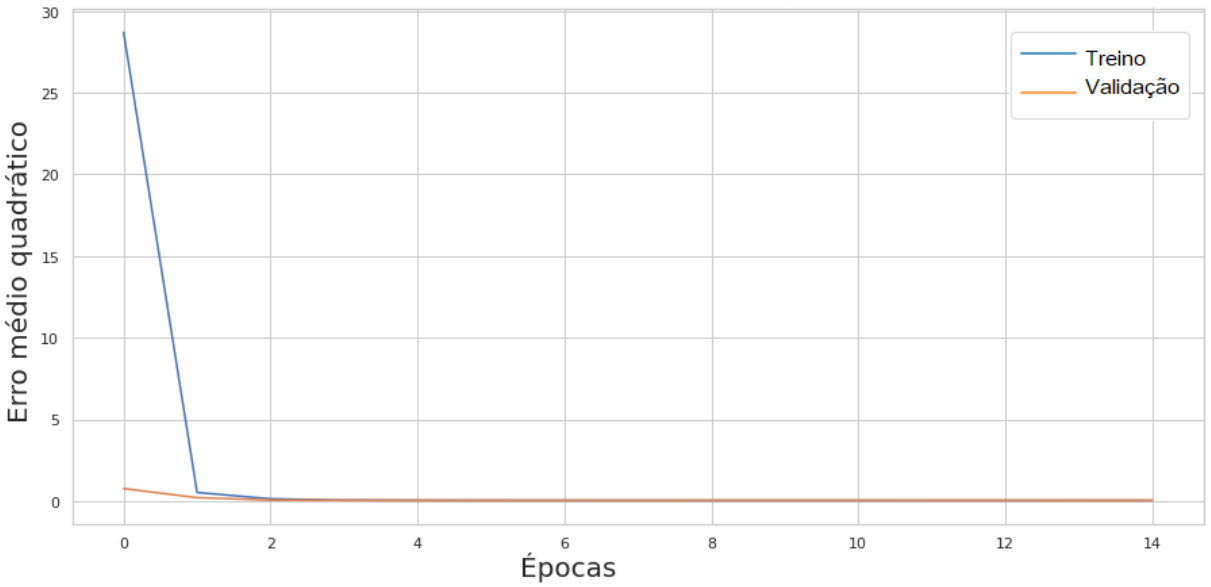

Figura 6.5: Curva de aprendizado para o Modelo 2.

De acordo com a Fig. 6.5, pode-se observar que, a partir da quarta época, os erros de treino e validação são praticamente iguais. Observou-se também que não há *overfitting* no modelo da RNA. O melhor resultado de MSE foi obtido na época 15, onde o erro foi igual a 4,0% para o conjunto de treino e 3,5% para o conjunto de validação.

O resultado do MSE para conjunto de testes ficou em torno de 3,5%. Levando-se em consideração que esses dados são desconhecidos pela rede e a pouca diferença observada nas medidas, foi possível concluir que a identificação dos parâmetros foi boa.

Foi possível também perceber que, no Modelo 2 foi preciso utilizar mais características na entrada da rede para obter um resultado satisfatório, enquanto no Modelo 1 utilizando menos entradas foi obtido um resultado melhor. Isto faz sentido, pois ao aumentar o número de parâmetros que a RNA deveria prever a identificação torna-se mais complexa, ou seja, ao oferecer mais informações de entrada na RNA melhora-se o desempenho da mesma.

Lembrando que a diferença entre o Modelo 1 e o Modelo 2 está no número de parâmetros da densidade espectral, ou seja, mais parâmetros precisavam ser identificados.

6.5 Modelo 3

A seguir tem-se a matriz de correlação da Fig. 6.6:

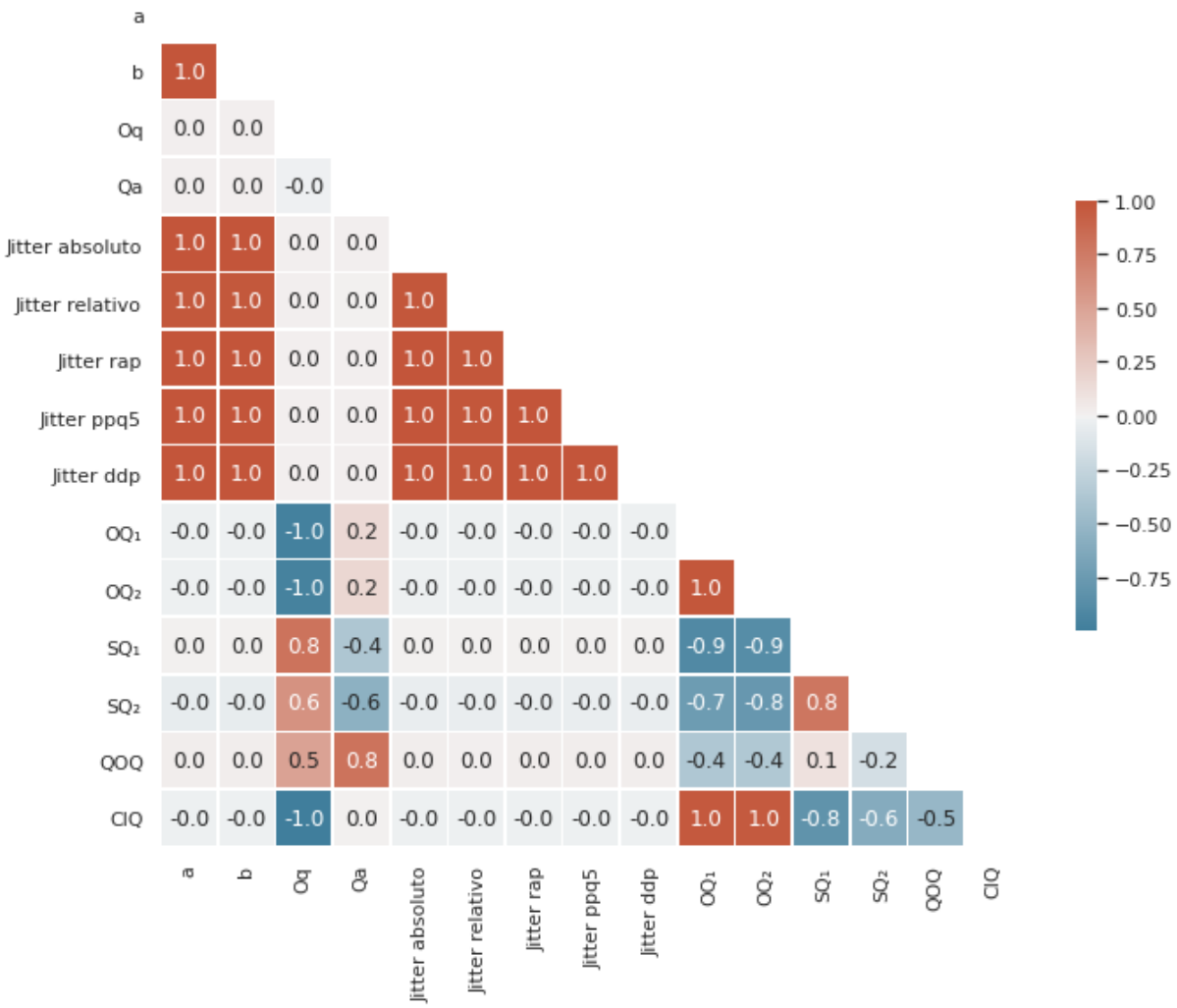

Figura 6.6: Matriz de correlação Modelo 3.

Pode-se perceber que todas as características de *jitter* e algumas de tempo: SQ_1, SQ_2 e QOQ apresentaram as maiores correlações, por isso foram utilizados como entrada da rede neural.

Após definir as características de entrada, foi realizado o treinamento da rede. A curva de aprendizado para os dados de treino e testes está ilustrada na Fig. 6.7.

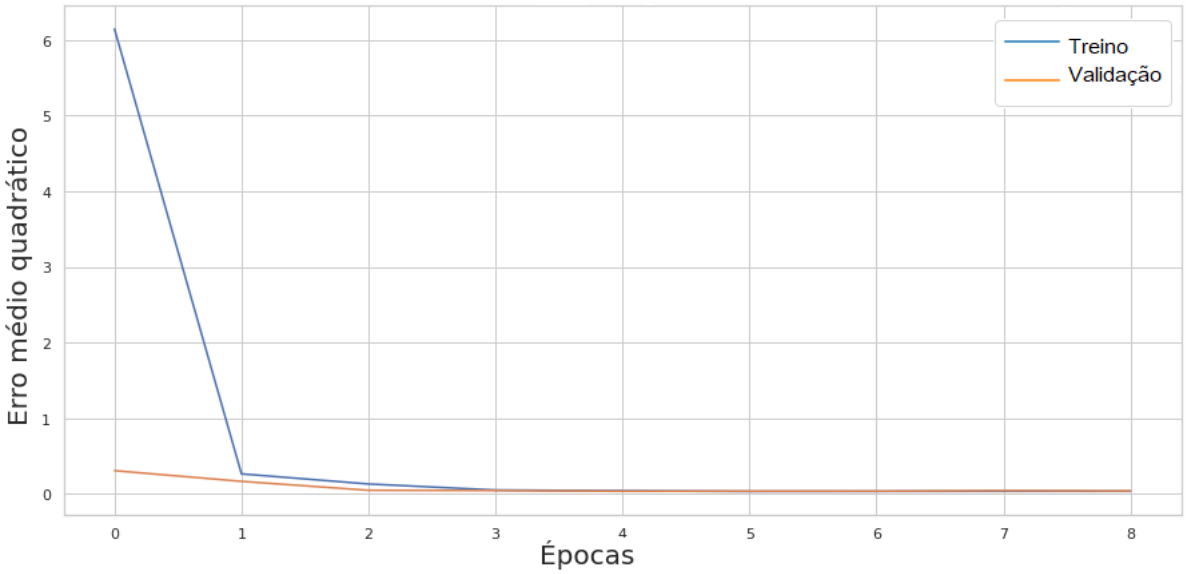

Figura 6.7: Curva de aprendizado para o Modelo 3.

De acordo com a Fig. 6.7, pode-se observar que a partir da quarta época o erro de treino e validação são praticamente iguais, logo não houve *overfitting* no treinamento do modelo. O melhor resultado de MSE foi obtido na época 9, onde o erro foi igual a 3,6% para o conjunto de treino e 3,9% para o conjunto de validação, ou seja, praticamente iguais.

Para o conjunto de testes o valor do MSE ficou em torno de 4,4%, pior do que o resultado de treino e validação, mas levando também em consideração que esses dados são novos para a rede pode-se concluir que a identificação dos parâmetros foi satisfatória.

Esse modelo usa o modelo LF como base, que é um modelo mais complexo. Portanto, é de se esperar que as caracaterísticas usadas para identificar os seus parâmetros sejam diferentes.

6.6 Modelo 4

Abaixo, a matriz de correlação da Fig. 6.8:

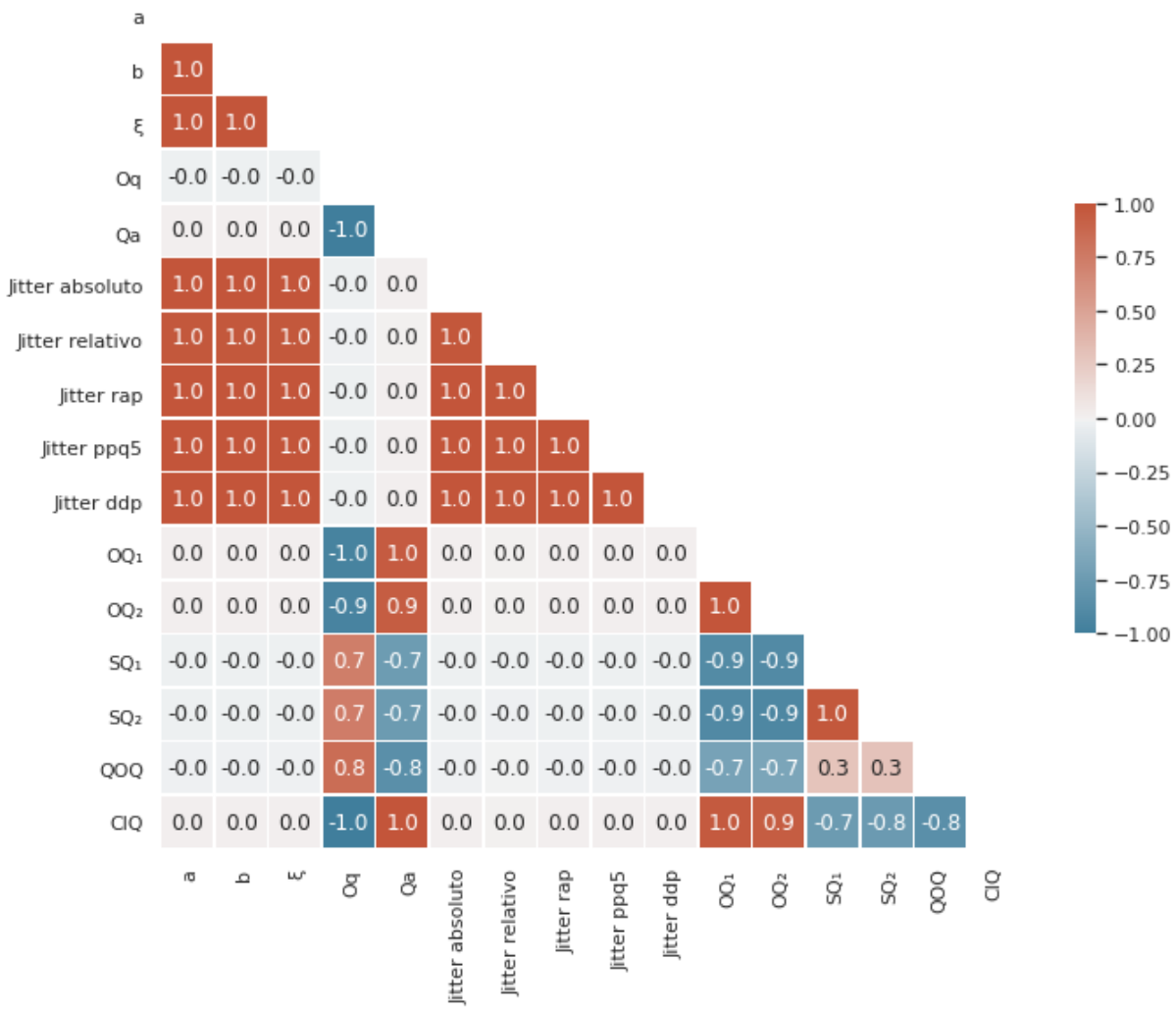

Figura 6.8: Matriz de correlação Modelo 4.

Pode-se perceber que todas as características de *jitter* e de tempo apresentaram boas correlações, por isso foram utilizados como entrada da rede neural.

Esse é o modelo mais complexo e, dessa forma, é onde foram usadas mais características.

Após definir os parâmetros de entrada foi realizado o treinamento da rede, a curva de aprendizado para os dados de treino e testes está ilustrada na Fig. 6.9.

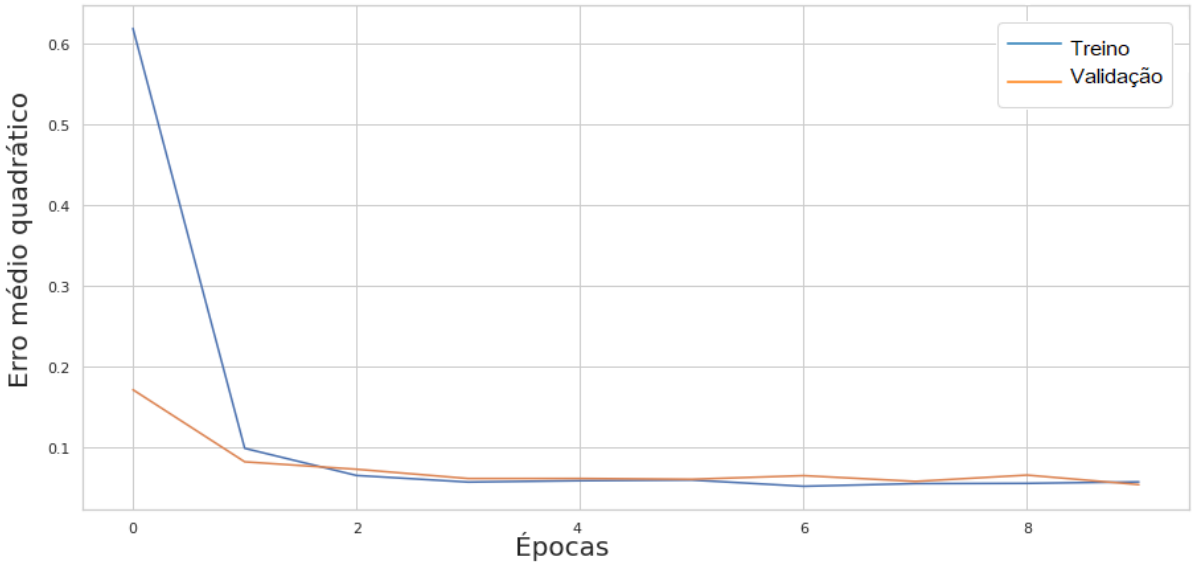

Figura 6.9: Curva de aprendizado para o Modelo 4.

De acordo com a Fig. 6.9, pode-se observar que a partir da nona época o erro de treino e validação são praticamente iguais. O melhor resultado de MSE foi obtido na época 10, onde o erro foi igual a 5,7% para o conjunto de treino e 5,4% para o conjunto de validação, ou seja, praticamente iguais.

O valor do MSE para o conjunto de testes ficou em torno de 4,6%. Este valor foi melhor do que o valor MSE de treino e validação. O Modelo 4 é o mais complexo entre todos estudados até aqui, por isso, esperava-se que a identificação de seus parâmetros fosse um pouco mais complicada. Sendo assim, entre todos os modelos, este foi o que retornou a pior taxa de erro, porém os valores não passaram de 6%, o que é um ótimo resultado.

6.7 Sinal de voz e características calculadas a partir da identificação de parâmetros da RNA

A ideia desta seção é a de gerar alguns sinais de voz com os parâmetros identificados, extrair características e comparar com as correspondentes entradas nas RNAs.

A rigor isso não faria sentido pois já existem as medidas de eficácia das RNAs, porém a ideia é um pouco mais abrangente e inclui as sínteses dos sinais de voz para que possam ser ouvidos e percebidos.

Foram gerada amostras de vozes pelo modelo fonte-filtro. Foi selecionada uma amostra de cada modelo e geradas as vogais a, e, i, o, u para o sinal original e o obtido pela RNA. Estes resultados se encontram no link `https://drive.google.com/drive/folders/1VmZEhPfud3d_Uqovk37IcVVQGNHODk91?usp=sharing`. Nas Tabelas 6.5, 6.6, 6.7, 6.8 pode-se observar os valores originais das características que foram utilizados na entrada da rede e o valor calculado a partir dos parâmetros identificados. Os cálculos foram os mesmos realizados na geração da base de dados para o treinamento da rede.

Tabela 6.5: Características do Modelo 1

Característica	**Original**	**RNA**
OQ_1	3129	2743
OQ_2	1597	1947
SQ_1	0.79	0.79
SQ_2	0.85	0.85
CIQ	15198	13439
QOQ	12399	12476
Jitter Absoluto	3.92e-05	4.14e-05
Jitter Relativo	116%	123%
Jitter RAP	0.22%	0.24%
Jitter PPQ5	0.24%	0.25%
Jitter DDP	0.67%	0.73%

Tabela 6.6: Características do Modelo 2

Característica	**Original**	**RNA**
OQ_1	3603	3242
OQ_2	2521	2161
SQ_1	0.79	0.79
SQ_2	0.86	0.86
CIQ	17837	15494
QOQ	11171	12528
Jitter Absoluto	2.33e-11	2.65e-11
Jitter Relativo	0.00023%	0.00026%
Jitter RAP	2.48e-07%	2.76e-07%
Jitter PPQ5	2.6e-07%	2.9e-07%
Jitter DDP	7.43e-07%	8.3e-07%

Tabela 6.7: Características do Modelo 3

Característica	**Original**	**RNA**
OQ_1	784	831
OQ_2	542	588
SQ_1	0.82	0.82
SQ_2	0.87	0.87
CIQ	4505	4722
QOQ	7001	7106
Jitter Absoluto	3.60e-05	3.34e-05
Jitter Relativo	107%	99%
Jitter RAP	0.20%	0.19%
Jitter PPQ5	0.23%	0.21%
Jitter DDP	0.61%	0.58%

Tabela 6.8: Características do Modelo 4

Característica	**Original**	**RNA**
OQ_1	1176	1176
OQ_2	784	784
SQ_1	0.80	0.80
SQ_2	0.86	0.86
CIQ	5980	5980
QOQ	7549	7450
Jitter Absoluto	5.91e-12	5.0e-12
Jitter Relativo	1.75-05%	1.48e-05%
Jitter RAP	3.31e-08%	2.71e-08%
Jitter PPQ5	3.52e-08%	3.16e-08%
Jitter DDP	9.94e-08%	8.15e-08%

Pode-se ver que as características calculadas a partir dos parâmetros identificados pela RNA estão bem próximas das características utilizadas na entrada da rede. Dessa maneira, consegue-se mostrar que a rede está identificando bem os parâmetros e os áudios fornecidos no drive também ficaram bem parecidos.

6.8 Resumo dos Resultados

Nesta seção serão resumidos os melhores resultados obtidos e a configuração utilizada na RNA. É importante ressaltar que a configuração mostrada a seguir foi utilizada para todos os modelos estudados.

Configuração da RNA utilizada:

- Função de ativação: relu;
- Otimizador: Adam;
- Taxa de aprendizado: 0.001;
- Número de camadas ocultas: 3;
- Número de perceptrons em cada camada oculta: 32;

Conforme a Tabela 6.9 tem-se o resumo dos melhores resultados obtidos para os modelos estudados com a configuração da RNA mostrada acima.

Tabela 6.9: Resumo dos resultados para os modelos estudados.

Modelo	**Épocas**	**Erro treino**	**Erro validação**	**Erro teste**
1	16	1,2%	1,2%	1,2%
2	8	4,0%	3,5%	3,5%
3	9	3,6%	3,9%	4,4%
4	10	5,7%	5,4%	4,6%

De acordo com a Tabela 6.9, pode-se perceber que o Modelo 1 teve o melhor resultado, e que o modelo 4 teve o pior desempenho em relação aos outros modelos. O Modelo 4 é um modelo mais completo e que respresenta melhor o sinal de voz, por isso é de se esperar que a identificação de seus parâmetros seja mais complicada, pois a quantidade de parâmetros a ser identificada é maior. Por sua vez, o modelo 1 é o mais simples de todos que foram tratados neste trabalho.

6.9 Identificação de parâmetros de um modelo considerando um sinal de voz real

Como uma extensão do estudo apresentado, o conteúdo desenvolvido será aplicado em uma situação real. Isto é, dado um sinal de voz, pretende-se identificar os parâmetros de um modelo matemático que poderia gerar esse sinal.

Para verificar o comportamento da RNA com vozes reais, foi gravado um áudio de um locutor pronunciando a vogal *a*, com duração de três segundos. Primeiramente foram utilizados os códigos desenvolvidos em [10] [11] para extrair as características do sinal glotal, as mesmas utilizadas nos modelos matemáticos. Além das características já utilizadas, foram adicionadas outras características como dH12, HRF e T_o.

Na tabela 6.10 tem-se os valores das características reais extraídas a partir do sinal de voz do locutor.

Tabela 6.10: Características extraídas do sinal glotal

Característica	**Valor calculado**
OQ_1	0.26
OQ_2	0.25
SQ_1	0.61
SQ_2	0.57
CIQ	0.14
QOQ	0.15
Jitter Absoluto	7e-06
Jitter Relativo	0.18%
Jitter RAP	0.09%
Jitter PPQ5	0.11%
dH12	6.75
HRF	4.85
T_o	0.003678

Onde T_o se refere ao período fundamental.

6.9.1 Modelo 1

Primeiramente, foram realizados testes com o Modelo 1, que obteve os melhores resultados anteriormente. Por isso, foi necessário gerar uma nova base de dados ajustando as características de acordo com a voz do locutor em questão, para que a rede tivesse um bom resultado. Isto é, foi preciso ajustar pelo menos alguns parâmetros do modelo, tal como frequência fundamental e outros, antes de fazer variações em torno desses valores para gerar o banco de dados.

Na tabela 6.11 tem-se os valores dos parâmetros utilizados para gerar esta nova base, para esta situação foi utilizado o Modelo 1.

Tabela 6.11: Variação dos parâmetros para o Modelo 1.

Parâmetros	**Valor inicial**	**Passo**	**Valor final**
a	5	0.1	25
b	10000	100	30000
α_1	40	1	90
α_2	20	1	70

O valores da Tabela 6.11 foram escolhidos de acordo com o resultado das características extraídas do sinal de voz real na Tabela 6.10, tentando chegar próximo aos valores do locutor real. Após a geração da nova base de dados, foi analisada a correlação entre estes dados e como resultado obteve-se a matriz da Fig. 6.10.

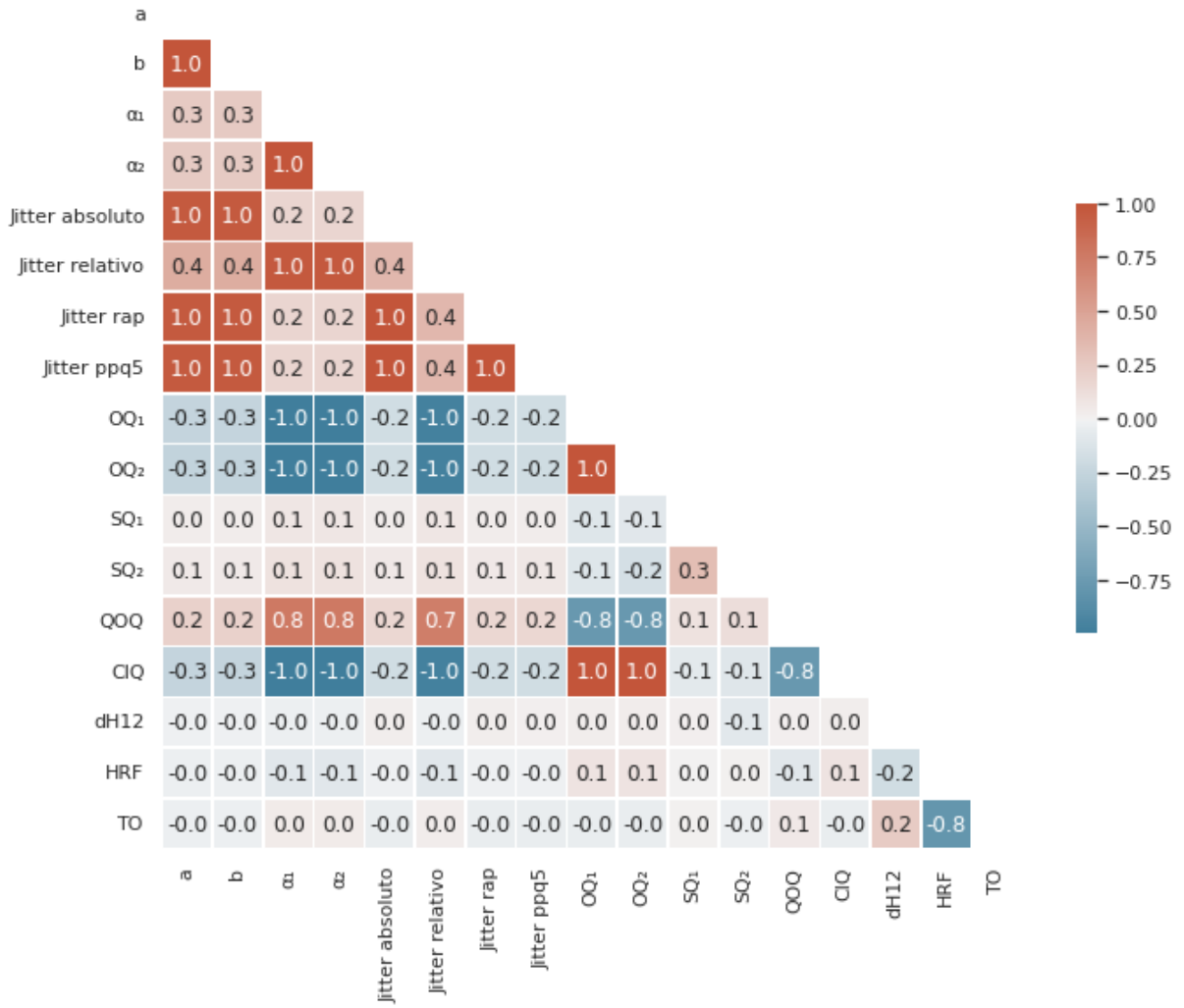

Figura 6.10: Matriz de correlação da nova base de dados.

Baseado nisso, pode-se perceber que as características de *jiiter* continuam tendo alta correlação com os parâmetros a e b, porém α_1 e α_2 não tem alta correlação com as características de tempo, como foi visto no modelo matemático.

Sendo assim, o experimento foi dividido em duas etapas. Na primeira, utilizou-se apenas as características de *jitter* como entrada da rede neural, são elas: *Jitter* Absoluto, Relativo, RAP e PPQ5. Com isso foi possível prever os parâmetros a, b para o modelo 1 estudado anteriormente. O modelo 1 foi escolhido devido à melhor perfomance da rede e por ser o mais simples, dado que este é o começo de um estudo com a voz real. Nesta etapa fixou-se os parâmetros α_1 e α_2.

Na tabela 6.12 estão os parâmetros identificados para o modelo 1, utilizando a RNA.

Tabela 6.12: Parâmetros do Modelo 1.

Parâmetro	**RNA**
a	2
b	2171

De acordo com a Fig. 6.11, a rede foi treinada por 12 épocas, o valor do erro no conjunto de treino foi 9,6%, e no conjunto de validação 9,8%. Para o conjunto de testes o erro ficou em torno de 8,9%, o que foi um bom resultado.

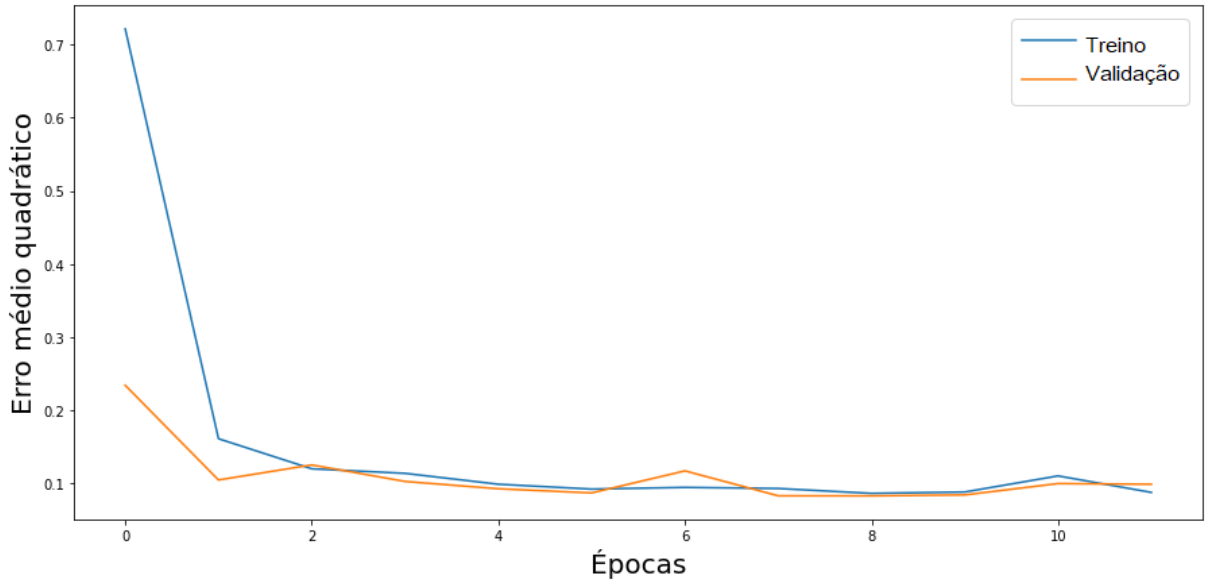

Figura 6.11: Curva de aprendizado.

Ao gerar o sinal glotal, a partir dos parâmetros identificados pela RNA para o modelo 1, calcula-se os valores das características para comparar com os valores do sinal de voz real, estes valores estão apresentando na Tabela 6.13.

Tabela 6.13: Características extraídas do sinal glotal - Modelo 1

Característica	**Valor calculado**
Jitter Absoluto	6e-06
Jitter Relativo	0.13%
Jitter RAP	0.09%
Jitter PPQ5	0.10%

Comparando as tabelas 6.10 e 6.13 pode-se ver que os valores para o sinal glotal gerado através do modelo estocástico ficaram bem próximos dos valores reais.

Para gerar o sinal de voz artificialmente os parâmetros α_1 e α_2 foram fixados em 60 e 20, respectivamente. A frequência fundamental utilizada foi a média das frequências fundamentais identificadas para este locutor, no valor de 271.87 Hz. As frequências das formantes foram substituídas [38], pois o locutor em questão é do sexo feminino, e estão apresentadas na Tabela 6.14. O áudio gerado está disponível em `https://drive.google.com/drive/folders/1BCJcrpA59HEzbTpxXMsUbvu3BqQkY7pq`.

Tabela 6.14: Frequência das três primeiras formantes [F1, F2, F3], para cada vogal do sexo feminino

Vogal	**F1**(Hz)	**F2**(Hz)	**F3**(Hz)
/a/	1002.9	1549.95	2959.7
/e/	437.03	2429.76	3087.09
/i/	361.9	2583.89	3378.14
/o/	444.89	914.26	2899.8
/u/	461.82	763.41	2902.55

Também foram realizados testes com algumas combinações das características dH12, HRF e T_o, mas como já era esperado devido a baixa correlação entre eles não houve melhora no desempenho da rede neural.

Na segunda parte do experimento, foram utilizadas todas as características calculadas como entrada da rede, são elas: OQ_1, OQ_2, SQ_1, SQ_2, CIQ, QOQ e os parâmetros de *jitter* usados anteriormente. Neste caso, a rede também irá prever os parâmetros α_1 e α_2. Na tabela 6.15 tem-se os parâmetros do modelo 1 identificados pela RNA.

Tabela 6.15: Parâmetros do Modelo 1

Parâmetro	**RNA**
a	20.4
b	218.5
α_1	1.21
α_2	1.56

Observou-se que os valores identificados de α_1 e α_2 ficaram bem distante dos valores utilizados para gerar a base de dados. Neste caso os resultados não foram bons, isso já era esperado devido a baixa correlação entre os parâmetros α_1, α_2 e as características de tempo.

Para o experimento realizado com o Modelo 1, foi possível observar que, diferentemente do que foi observado no experimento com o modelo matemático, o desempenho da rede piorou ao adicionar as características de tempo. A baixa correlação entre os parâmetros e estas características nessa nova base de dados gerada podem ter sido o problema. Para os próximos estudos deve-se explorar outras características de tempo para o sinal de voz real.

6.9.2 Modelo 4

Por último, realizou-se o mesmo estudo para o Modelo 4. Para isso, foi necessário gerar uma nova base de dados afim de ajustar os valores das características deste modelo ao do locutor real. Na tabela 6.16 tem-se os valores dos parâmetros utilizados para gerar esta nova base.

Tabela 6.16: Variação dos parâmetros para o Modelo 4.

Parâmetros	**Valor inicial**	**Passo**	**Valor final**
a	100	2	500
b	10000	200	50000
ξ	0.1	0.1	20
O_q	0.2	0.1	1
Q_a	0.1	0.1	0.9

Após isso, foi analisada a correlação entre estes dados e o resultado pode ser observado na matriz da Fig. 6.12.

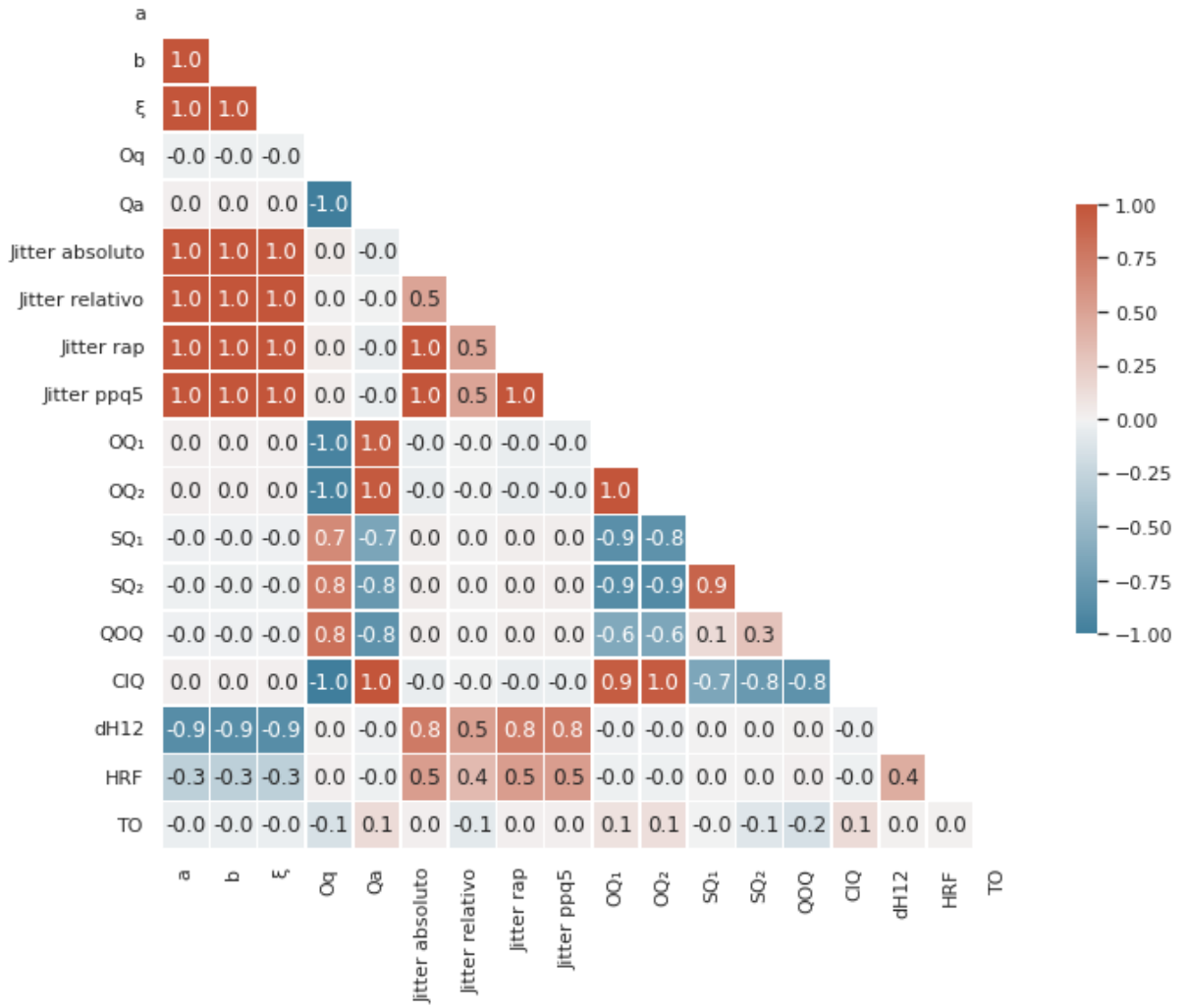

Figura 6.12: Matriz de correlação da nova base de dados.

Pode-se ver que a correlação entre os parâmetros a, b e ξ e as características de jitter é alta, como já era esperado. E, para os parâmetros de tempo há uma alta correlação com O_q e Q_a. Por isso, foram utilizados todos parâmetros como entrada da RNA.

Na tabela 6.18 estão os parâmetros identificados para o modelo 4, utilizando a RNA.

Tabela 6.17: Parâmetros do Modelo 4.

Parâmetro	**RNA**
a	2
b	1662
ξ	0.46
O_q	0.04
Q_a	0.05

Apesar de ter um bom resultado para os valores a, b e ξ, os parâmetros O_q e Q_a não estão dentro dos valores esperados, para que o sinal glotal neste modelo fique adequado é preciso que O_q seja maior do que Q_a. Sendo assim, foram realizados novos testes utilizando como entrada da RNA apenas as características de *jitter*.

Para isso, fixou-se o valor de O_q e Q_a em 0.8 e 0.3, respectivamente. Na tabela 6.18 estão os parâmetros identificados para o modelo 4, utilizando a RNA apenas com entradas de *jitter*.

Tabela 6.18: Parâmetros do Modelo 4.

Parâmetro	**RNA**
a	362
b	35896
ξ	12

Na Fig. 6.13 pode-se ver a curva de aprendizado para este caso. A rede foi treinada por 18 épocas, o valor do erro no conjunto de treino foi 5,2%, e no conjunto de validação 5,3%. Para o conjunto de testes o erro ficou em torno de 4,1%.

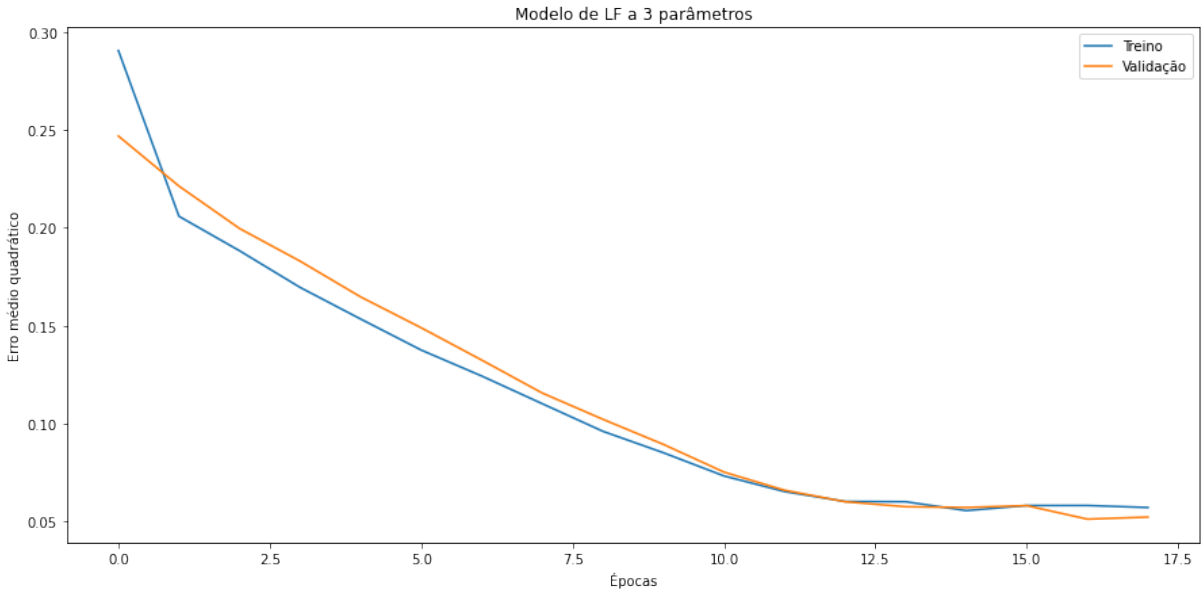

Figura 6.13: Curva de aprendizado.

Após a identificação dos parâmetros pela RNA, calcula-se os valores das características para comparar com os valores do sinal de voz real, estes valores estão apresentando na Tabela 6.19.

Tabela 6.19: Características extraídas do sinal glotal - Modelo 4

Característica	**Valor calculado**
Jitter Absoluto	2e-07
Jitter Relativo	0.07%
Jitter RAP	0.05%
Jitter PPQ5	0.05%

Os áudios gerados para esse modelo estão disponíveis em `https://drive.google.com/drive/folders/1BCJcrpA59HEzbTpxXMsUbvu3BqQkY7pq`, eles foram gerados utilizando a mesma regra aplicada ao testes do Modelo 1 desta seção.

Os parâmetros identificados através da RNA utilizando apenas entradas de *jitter* geraram valores de características próximos às do sinal de voz real. Os resultados do Modelo 1 ficaram um pouco melhores devido à complexidade do Modelo 4, mas mesmo assim os parâmetros puderam ser identificados.

Capítulo 7

Conclusões

Neste trabalho, foi possível mostrar que a identificação de parâmetros em modelos estocásticos do sinal glotal usando Redes Neurais Artificiais é possível e que, muito mais do que isso, funciona bem.

Os parâmetros identificados através da RNA geraram vozes muito parecidas com as vozes originais simuladas pelos modelos matemáticos. Os modelos 3 e 4 geraram sinais de vozes mais suaves, enquanto os modelos 1 e 2 tinham um som mais robótico, isso deve-se também ao tipo de modelo do sinal glotal, mais simples ou mais próximo da realidade.

Como consequência dos sinais de vozes gerados estarem muito parecidos, pode-se observar que as características originais e geradas pela RNA ficaram bem próximas. Dando destaque às características de tempo SQ_1 e SQ_2, que tiveram os valores iguais para todos os modelos estudados.

Com relação às características de *jitter* pode-se perceber que com os valores utilizados para os parâmetros, os modelos 1 e 3 foram os que responderam melhor a essas caractrísticas, pois obteve-se medidas mais significativas de *jitter*, podendo simular melhor o que foi proposto neste trabalho.

Apesar do modelo 4 ser o mais robusto em termos de produção do sinal de voz, foi o que teve o pior desempenho em termos de identificação dos parâmetros da RNA, tendo uma taxa de erro maior em relação aos outros modelos. Entretanto, ainda pode-se considerar que foi um bom resultado, pois as taxas de erro ficaram abaixo de 6%.

A RNA utilizada teve uma boa perfomance para todos os modelos utilizados, deixando em destaque os modelos 1 e 3 que tiveram as menores taxas de erro. No geral, a taxa de erro foi baixa para todos os modelos. O treinamento da RNA teve uma média de 10 épocas levando em consideração os 4 modelos.

No experimento que foi realizado com vozes reais verificou-se que, os valores de erro no treinamento da RNA para o Modelo 4 foram melhores em relação ao Modelo 1. Além disso, não foi possível obter valores satisfatórios dos parâmetros do Modelo 4, utilizando a combinação das características de tempo e *jiiter*, assim como no Modelo 1. Contudo, os áudios gerados para o Modelo 4 ficaram mais agradáveis, é preciso relembrar que se está trabalhando neste caso com uma frequência mais alta, o que faz os sons serem mais agudos. O Modelo 4 é o modelo mais complexo em relação ao sinal glotal, pois tem mais parâmetros a serem determinados, isso faz com que a RNA tenha mais dificuldades para identificar seus parâmetros.

7.1 Trabalhos futuros

Para trabalhos futuros pretende-se evoluir no estudo para os casos de vozes reais, utilizando mais vozes humanas. Para isso, é preciso criar uma base de dados robusta de vozes de locutores reais, de diferentes idades e gêneros, com e sem patologias vocais. Após isso, extrair o sinal glotal e realizar os mesmos procedimentos que foram feitos com os modelos matemáticos estocásticos. Por fim, estudar os parâmetros do modelo estocástico que mais se adequam a estas características. Além disso, pretende-se explorar outras características de tempo e a combinação cruzada das características de tempo e jitter, afim de melhorar o resultado da RNA com as mesmas.

Referências

[1] CATALDO E.; MONTEIRO, L. . S. C. A novel source-filter stochastic model for voice production. *Journal of voice*, v. 1, p. 1–8, 2021.

[2] JIANG W.; ZHENG, X. X. Q. Computational Modeling of Fluid–Structure–Acoustics Interaction during Voice Production. *Frontiers in Bioengineering and Biotechnology*, v. 5, p. 7, 2017.

[3] KOHLER, E. Rede neural para identificar nível de estresse na voz: uma abordagem testando parâmetros. *Trabalho de Conclusão de Curso - Universidade Federal de Santa Catarina.*, p. 1–89, 2021.

[4] CATALDO, E.; LIMA, R. Identificação de um pulso glotal determinístico através de seus parâmetros de tempo e frequência. *XXXVIII Simpósio Brasileiro de Telecomunicações e processamento de sinais - SBrT 2020, Florianópolis, SC*, 2020.

[5] SANTOS, J. C. Modelagem matemática-computacional das pregas vocais aplicada à síntese de vozes alteradas. *Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal de Sergipe*, p. 1–105, 2021.

[6] BENDEL, O. The synthetization of human voices. *AI Soc*, v. 34, p. 83–89, 2019.

[7] CATALDO, E.; SOIZE, C.; SILVA, R. L.; SILVA, J. M. Identification of a stochastic process modeling the stiffness of the vocal folds for a voice production model represented by a neural network. *XI International Conference on Structural Dynamics, EURODYN 2020*, v. 1, p. 3403–3412, 2020.

[8] DEGOTTEXD, G. Glottal Source and Vocal-Tract Separation - Estimation of glottal parameters, voice transformation and synthesis using a glottal model. *Tese de doutorado - Université Pierre et Marie Curiel, Paris VI (UPMC)*, 2010.

[9] LIN, J. L. G. F. Q. A Four Parameter Model of Glottal Flow. *STL-QPSR*, p. 1–13, 1985.

[10] FLORENTINO C; MOREIRA, F. Desenvolvimento de um sistema de verificação de locutor, usando modelos ocultos de Markov, unindo a técnica MFCC com parâmetros extraídos do sinal glotal. *Trabalho de Conclusão de Curso - Universidade Federal Fluminense, Niterói*, p. 1–87, 2017.

[11] SANTOS R; RAMOS, R. Desenvolvimento de um programa de verificação de locutor usando como entrada para hmms parâmetros extraídos do sinal de voz e do sinal glotal na emissão de palavras contendo conjuntamente sons vozeados e não-vozeados. *Trabalho de Conclusão de Curso - Universidade Federal Fluminense, Niterói*, p. 1–113, 2018.

[12] BAHIANO, D. Criação de modelos estocásticos para a síntese de vogais considerando os pulsos glotais de Rosenberg e de Liljencrants-Fant com parâmetros unificados. *Dissertação de Mestrado em Engenharia Elétrica e de Telecomunicações - Universidade Federal Fluminense, Niterói*, p. 1–94, 2020.

[13] BAHIANO D.; CATALDO, E. Criação de modelos estocásticos para a síntese de vogais considerando os pulsos glotais de Rosenberg e de Liljencrants-Fant com parâmetros unificados. *XXXVIII Simpósio brasileiro de telecomunicações e processamento de sinais - SBrT*, 2020.

[14] BRANDÃO A. ; CATALDO, E. . L. F. Classificacao de patologias em vozes naturais e em vozes sintetizadas por modelos mecanicos. *Congresso Nacional de Engenharia Mecanica, Recife. Anais do IV CONEM.*, 2006.

[15] BRANDÃO, A. Modelagem acústica da produção da voz utilizando técnicas de visualização de imagens médicas associadas a métodos numéricos. *Tese de Doutorado em Engenharia Mecânica - Universidade Federal Fluminense, Niterói*, 2011.

[16] SILVA, M. A. B. da. Modelos de redes neurais spiking evoluídas com inspiração quântica aplicado ao pré-diagnóstico de envelhecimento vocal.

[17] CHARTPAD, A. *Diagram Of Respiratory System A Clear Diagram Of The Human Respiratory System Anatomy Body.* 2021. `https://anatomychartpad.com/diagram-of-respiratorysystem`.

[18] MONGIA T. K.; SHARMA, R. K. Estimation and Statistical Analysis of Human Voice Parameters to Investigate the Influence of Psychological Stress and to Determine the Vocal Tract Transfer Function of an Individual. *Journal of Computer Networks and Communications*, v. 2014, 2014.

[19] DIAS, S. O. Estimation of the glotal pulse from speech or singing voice. *Tese de Doutorado em engenharia biomédica - Universidade do Porto, Portugal*, p. 1–140, 2012.

[20] PRAAT, S. *Universidade de Amsterdam.* 2009. `http://www.fon.hum.uva.nl/praat/`.

[21] HENRICH, N. Etude de la source glottique en voix parlée et chantée : modélisation et estimation, mesures acoustiques et électroglottographiques, perception. *Tese de doutorado - Université Pierre et Marie Curie-Paris VI (UPMC)*, 2001.

[22] GUNNING, D.; STEFIK, M.; CHOI, J.; MILLER, T.; STUMPF, S.; YANG, G.-Z. Xai-explainable artificial intelligence. *Science Robotics*, 2019.

[23] ZHANG, X.-D. Machine learning. *A Matrix Algebra Approach to Artificial Intelligence*, Springer Singapore, Singapore, p. 223–440, 2020.

[24] WANG, S.-C. Artificial neural network. *Interdisciplinary Computing in Java Programming*, Springer US, Boston, MA, p. 81–100, 2003.

[25] FETZER, J. H. What is artificial intelligence? *Artificial Intelligence: Its Scope and Limits*, Springer Netherlands, Dordrecht, p. 3–27, 1990.

[26] MARX, V. The big challenges of big data. . *Nature*, v. 498, p. 255–260, 2013.

[27] ACADEMY, D. S. *Deep Learning Book*. 2021. https://www.deeplearningbook.com.br/.

[28] HAHNLOSER, R.; SARPESHKAR, R.; MAHOWALD, M.; DOUGLAS, R. Digital selection and analog amplification co-exist in an electronic circuit inspired by neocortex. *Nature*, 01 2000.

[29] KERLIRZIN, P.; VALLET, F. Robustness in multilayer perceptrons. *Neural Computation*, v. 5, p. 473–482, 1993.

[30] FUKUMIZU, K. Statistical active learning in multilayer perceptrons. *IEEE Transactions on Neural Networks*, v. 11, p. 17–26, 2000.

[31] LILLICRAP T.P., S. A. M. L. e. a. Backpropagation and the brain. *Nat Rev Neurosci*, v. 21, p. 335–346, 2020.

[32] ROJAS, R. The backpropagation algorithm. *Neural Networks: A Systematic Introduction*, Springer Berlin Heidelberg, Berlin, Heidelberg, p. 149–182, 1996.

[33] ACADEMY, D. S. *Capítulo 12 – Aprendizado Com a Descida do Gradiente*. 2021. https://www.deeplearningbook.com.br/aprendizado-com-a-descida-do-gradiente/.

[34] KERAS. *EarlyStopping*. 2021. https://keras.io/api/callbacks/early_stopping/.

[35] YING, X. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, v. 1168, p. 022022, 2019.

[36] DEVELOPERS keras. *Keras Layers API*. 2021. https://keras.io/api/layers/.

[37] KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego*, 2015.

[38] GONCALVES, M. I. e. a. Brazilian journal of otorhinolaryngology. v. 75, p. 680–684, 2009.

APÊNDICE A – Modelo 1: Pulso glotal de Rosenberg e densidade espectral a 2 parâmetros.

```
import os
import matplotlib.pyplot as plt
import numpy as np
from numpy import cos,exp, log10,pi
from scipy.signal import lfilter, freqz
from random import random
from bisect import bisect_left

#Pulso Unitario com Processo Estocastico
def ros_uno_uni_normal(Av,FO,fm,alpha1,alpha2,abar,bbar):

    Tm=1/float(fm) #intervalo de amostragem do pulso
    TO=1/float(FO) #periodo fundamental do ciclo
    TP =(alpha1/float(100))*TO
    TN =a(alpha2/float(100))*TO
    xbar=(1.-abar**2/(2*bbar))**0.5
    kbar=10000
    beg=5000
    xx=([])
    xini=0
    ko=1/40000.0
    for ii in range(kbar):
        xfin=([((-float(bbar)*Tm/2.+1.)*xini+abar*np.random.randn()*Tm
    **0.5)/(1+float(bbar)*Tm/2.)])
        xx=np.concatenate((xx,xfin))
        xini=float(xfin[-1])
    dT=ko+(Tm-ko)*(xbar+xx[beg:])**2
    TOfin=sum(dT[0:int(TO/Tm)])
    TOvetor=TOfin*np.ones(int(TO/Tm)+1)
```

```
1028      t=([0])
          for iii in range(len(dT)):
1030          t=np.concatenate((t,([t[-1]+dT[iii]])))
          t=t[0:int(TO/Tm)+1]
1032      x=([])
          k=0
1034      n=0
          TP =(alpha1/float(100))*TOfin
1036      TN =(alpha2/float(100))*TOfin
          for i in range(len(t)):
1038          if t[i]<=TP:
                  k+=1
1040          if t[i]<=TP+TN:
                  n+=1
1042
         #criacao do pulso de Rosenberg com delta t(t) processo estoc stico
1044      for i in range(k):
              x1 = [(Av*0.5*(1-np.cos(np.pi*t[i]/(TP))))]
1046          x=np.concatenate((x,x1))
          for i in range(k,n):
1048          x2 = [(Av*(np.cos(np.pi*(t[i]-TP)/(2*TN))))]
              x=np.concatenate((x,x2))
1050      for i in range(n,len(t)):
              x3 = [(0)]
1052          x=np.concatenate((x,x3))
          imp=np.zeros(int(TOfin/Tm)+2) #fun   o impulso delta(t)
1054      imp[0]=1
          z = np.convolve(x,imp)
1056      return x,t,TOfin

1058  def find_nearest(array, value):
          idx = (np.abs(array - value)).argmin()
1060      return array[idx]

1062  #gerando 200 amostras de a, b, alpha1 e alpha2
      a=[]
1064  for i in range(100, 500, 2):
        a.append(i)
1066  print(len(a))
      b=[]
1068  for i in range(1000000, 5000000, 20000):
        b.append(i)
1070  print(len(b))
```

```
alpha1=[]
for i in range(1,5):
  for i in range(50, 100, 1):
    alpha1.append(i)
print(len(alpha1))
alpha2=[]
for i in range(1,5):
  for i in range(20, 70, 1):
    alpha2.append(i)
print(len(alpha2))

FO=180 #frequencia funcamental
fm=20000 #frequencia de amostragem
Tm=1/float(fm) #periodo de amostragem
Av=7 #amplitude de vozeamento
Tfs=3 #tempo total de sustentacao da vogal
TO=1/float(FO) #periodo fundamental
x1=([]) #criacao do vetor de pulsos LF com TO(t) processo estocastico
g1=([]) #criacao do vetor de pulsos LF Determin stico
jit=([])
todas=([])
t=([])
tz=0
jitabs_final=[]
jitrel_final=[]
rap_final=[]
ppq5_final=[]
ddp_final=[]
oq1=[]
oq2=[]
sq1=[]
sq2=[]
ciq=[]
qoq=[]
oq1_final=[]
oq2_final=[]
sq1_final=[]
sq2_final=[]
ciq_final=[]
qoq_final=[]
dH12_final=[]
HRF_final=[]
Harmonico_final=[]
```

```
ValorHarmonico_final=[]
for k in range(len(a)):
  #cria e concatena os pulsos
  for i in range(int(Tfs/TO)+10):
      #cria um pulso com TO(t) processo estocastico
      x,ty,TOfin=ros_uno_uni_normal(Av,FO,fm,alpha1[k],alpha2[k],a[k],b[k])
      list_x = x.tolist()
      tmax=list_x.index(max(list_x))
      #TP =alpha1[k]*TO  #to
      tc=alpha2[k]*TO  #tc
      x_a=x[0:tmax]
      x_d=x[tmax:len(x)]
      value_to1=find_nearest(x_a,0.1*(max(x))).tolist()
      value_to2=find_nearest(x_a,0.05*(max(x))).tolist()
      value_tqo=find_nearest(x_a,0.5*(max(x))).tolist()
      value_tqc=find_nearest(x_d,0.5*(max(x))).tolist()
      to1=list_x.index(value_to1, 0, tmax)
      to2=list_x.index(value_to2, 0, tmax)
      tqo=list_x.index(value_tqo, 0, tmax)
      tqc=list_x.index(value_tqc, tmax, len(list_x))
      oq1.append((tc-to1)/TOfin)
      oq2.append((tc-to2)/TOfin)
      sq1.append((tmax-to1)/(tc-tmax))
      sq2.append((tmax-to2)/(tc-tmax))
      ciq.append((tc-tmax)/TOfin)
      qoq.append((tqc-tqo)/TOfin)
      x1=np.concatenate((x1,x))#concatena os pulsos
      tt=ty+tz
      t=np.concatenate((t,tt))
      tz=t[-1]
      jit=np.concatenate((jit,([TOfin])))
  x=x1

  oq1_final.append(np.mean(oq1))
  oq2_final.append(np.mean(oq2))
  sq1_final.append(np.mean(sq1))
  sq2_final.append(np.mean(sq2))
  ciq_final.append(np.mean(ciq))
  qoq_final.append(np.mean(qoq))
  oq1=[]
  oq2=[]
  sq1=[]
  sq2=[]
```

```
    ciq=[]
1158 qoq=[]
    ValorHarmonico=[]
1160 Harmonico=[]
    dH12=[]
1162 HRF=[]
    VH=[]
1164 H=[]
    plt.plot(x[1:1000])
1166
    #Calcula FFT
1168 NN=int(Tfs/TOfin)+10;
    Fs=20000;
1170 deltaT=1/Fs;
    YT=fft(x,NN);
1172 PY=abs(YT);
    f=(Fs)*(np.arange(int(NN/2)))/NN;
1174 P=20*log10(PY);

1176 #S  considera o lado das frequ ncias positivas
    import math
1178 PP=P[1:math.floor(int(NN/2))+1];
    PP2=PP[5:len(PP)];
1180
    #Calcula os picos dos harm nicos
1182
    V2=max(PP2);
1184 PP2_list=PP2.tolist()
    I2=PP2_list.index(max(PP2_list))
1186
    ValorHarmonico.append(V2);
1188 Harmonico.append(f[I2+4]);
    deltaf=(Fs/(2*len(f)));
1190 Harmonic1=f[I2+4];

1192 #Determina o n mero de formantes que ser o considerados

1194 NumFormantes=10;

1196 for l in range(1,NumFormantes):

1198    novafreq=l*Harmonic1;
```

```
1200      #usa uma margem em torno de cada m ximo
          margem=50;
1202
          maxnovafreq=novafreq+margem;
1204      minnovafreq=novafreq-margem;

1206      #Determina o in cio e o fim do per odo no espectro
          #para determinar os m ximos
1208
          Ninic=math.floor(minnovafreq/deltaf);
1210      if (Ninic < 1):
            Ninic = 1;
1212      Nfim=math.floor(maxnovafreq/deltaf);

1214      vetmax=PP[Ninic:Nfim];

1216      valormax=max(vetmax);
          vetmax_list=vetmax.tolist()
1218      indmax=vetmax_list.index(max(vetmax_list))

1220      #Calcula as Amaxs
          ValorHarmonico.append(valormax);
1222      Harmonico.append(f[Ninic+indmax-1]);

1224    soma=0;

1226      #Calcula HRF - com NumFormantes
          #Escolhemos 10
1228
        for l in range(1,NumFormantes):
1230      soma=soma+ValorHarmonico[l];

1232      #Calcula dH12
        dH12=ValorHarmonico[0]-ValorHarmonico[1];
1234
        HRF=soma/ValorHarmonico[0];
1236
        VH.append(ValorHarmonico)
1238    H.append(Harmonico)

1240    dH12_final.append(dH12)
        HRF_final.append(HRF)
1242
```

```
    ValorHarmonico_final.append(np.mean(VH))
1244 Harmonico_final.append(np.mean(H))

1246 j=len(jit)
    jitabs=0
1248
    for i in range(j-1):
1250   jitabs+=abs((jit[i]-jit[i+1]))
    jitabs_final.append(jitabs/(j-1))
1252
    jitrel=jitabs/(sum(jit)/j)
1254 jitrel_final.append(jitrel*100)

1256 #novos parametros
    soma1=0
1258 for i in range(1,j-1):
        soma1+=abs((2*jit[i]/3.-jit[i-1]/3.-jit[i+1]/3.))
1260 soma1=soma1/(j-2)
    rap=soma1/(sum(jit)/j)
1262 rap_final.append(rap*100)

1264 soma2=0
    for i in range(2,j-2):
1266     soma2+=abs((4*jit[i]/5.-jit[i-1]/5.-jit[i-2]/5.-jit[i+1]/5.-jit[i
      +2]/5.))
    soma2=soma2/(j-4)
1268 ppq5=soma2/(sum(jit)/j)
    ppq5_final.append(ppq5*100)
1270
    soma3=0
1272 for i in range(1,j-1):
        soma3+=abs((jit[i+1]+jit[i-1]-2*jit[i]))
1274 soma3=soma3/(j-2)
    ddp=soma3/(sum(jit)/j)
1276 ddp_final.append(ddp*100)

1278 print(k)
    print(TOfin)
```

APÊNDICE B – Modelo 2: Pulso glotal de Rosenberg e densidade espectral a 3 parâmetros.

```
1000 import os
     import matplotlib.pyplot as plt
1002 import numpy as np
     from numpy import cos,exp, log10,pi
1004 from scipy.signal import lfilter, freqz
     from random import random
1006 from bisect import bisect_left


1008
     #Pulso Unitario com Processo Estocastico
1010 def ros_uno_uni_normal(Av,FO,fm,alpha1,alpha2,abar,bbar,eta):
         alpha=np.array([[0.,1.],[-float(bbar)**2,-2*eta*float(bbar)]])
1012     beta=np.array([[0],[abar]])
         Tm=1/float(fm)
1014     TO=1/float(FO)
         xbar=(1.-abar**2/(4*eta*bbar**3))**0.5
1016     TP =(alpha1/float(100))*TO
         TN =(alpha2/float(100))*TO
1018     kbar=10000
         beg=3000
1020     xx=([])
         xini=np.array([[0],[0]])
1022     ko=1/40000.0
         for ii in range(kbar):
1024         xfin=(-(alpha)*Tm/2.+np.ones((2,2))).dot(xini)
             xfin=(xfin+beta*np.random.randn()*Tm**0.5)
1026         xfin=inv((np.ones((2,2))+alpha*Tm/2.)).dot(xfin)
             kfin=([xfin[0][0]])
1028         xx=np.concatenate((xx,kfin))
```

```
            xini=xfin
    dT=ko+(Tm-ko)*(xbar+xx[beg:])**2
    TOfin=sum(dT[0:int(TO/Tm)])
    TOvetor=TOfin*np.ones(int(TO/Tm)+1)
    t=([0])
    for iii in range(len(dT)):
        t=np.concatenate((t,([t[-1]+dT[iii]])))
    t=t[0:int(TO/Tm)+1]
    x=([])
    k=0
    n=0
    TP=(alpha1/float(100))*TOfin
    TN=(alpha2/float(100))*TOfin
    for i in range(len(t)):
        if t[i]<=TP:
            k+=1
        if t[i]<=TP+TN:
            n+=1

    #criacao do pulso de Ros com delta t(t) processo estoc stico
    for i in range(k):
        x1 = [(Av*0.5*(1-np.cos(np.pi*t[i]/(TP))))]
        x=np.concatenate((x,x1))
    for i in range(k,n):
        x2 = [(Av*(np.cos(np.pi*(t[i]-TP)/(2*TN))))]
        x=np.concatenate((x,x2))
    for i in range(n,len(t)):
        x3 = [(0)]
        x=np.concatenate((x,x3))
    imp=np.zeros(int(TOfin/Tm)+2)
    imp[0]=1
    z = np.convolve(x,imp)
    return x,t,TOfin

def find_nearest(array, value):
    idx = (np.abs(array - value)).argmin()
    return array[idx]

#gerando 200 amostras de a, b, alpha1 e alpha2
a=[]
for i in range(100, 500, 2):
  a.append(i)
```

```
print(len(a))
b=[]
for i in range(1000000, 5000000, 20000):
  b.append(i)

print(len(b))
#b=100
alpha1=[]
for i in range(1,5):
  for i in range(50, 100, 1):
    alpha1.append(i)

print(len(alpha1))
#alpha1=60
alpha2=[]
for i in range(1,5):
  for i in range(20, 70, 1):
    alpha2.append(i)
print(len(alpha2))
#alpha2=20

eta=[]
for i in range(1,51):
  for i in range(1, 5, 1):
      eta.append(i/10)
print(len(eta))

FO=180 #frequencia funcamental
fm=20000 #frequencia de amostragem
Tm=1/float(fm) #periodo de amostragem
Av=7 #amplitude de vozeamento
Tfs=3 #tempo total de sustentacao da vogal
TO=1/float(FO) #periodo fundamental
x1=([]) #criacao do vetor de pulsos LF com TO(t) processo estocastico
g1=([]) #criacao do vetor de pulsos LF Determin stico
#eta=20
jit=([])
todas=([])
t=([])
tz=0
jitabs_final=[]
jitrel_final=[]
rap_final=[]
```

```
ppq5_final=[]
ddp_final=[]
oq1=[]
oq2=[]
sq1=[]
sq2=[]
ciq=[]
qoq=[]
oq1_final=[]
oq2_final=[]
sq1_final=[]
sq2_final=[]
ciq_final=[]
qoq_final=[]
dH12_final=[]
HRF_final=[]
Harmonico_final=[]
ValorHarmonico_final=[]

for k in range(len(a)):
  #cria e concatena os pulsos
  for i in range(int(Tfs/TO)+10):
      x,ty,TOfin=ros_uno_uni_normal(Av,FO,fm,alpha1[k],alpha2[k],a[k],b[k],
    eta[k])
      list_x = x.tolist()
      tmax=list_x.index(max(list_x))
      #TP =alpha1[k]*TO  #to
      tc=alpha2[k]*TO  #tc
      x_a=x[0:tmax]
      x_d=x[tmax:len(x)]
      value_to1=find_nearest(x_a,0.1*(max(x))).tolist()
      value_to2=find_nearest(x_a,0.05*(max(x))).tolist()
      value_tqo=find_nearest(x_a,0.5*(max(x))).tolist()
      value_tqc=find_nearest(x_d,0.5*(max(x))).tolist()
      to1=list_x.index(value_to1, 0, tmax)
      to2=list_x.index(value_to2, 0, tmax)
      tqo=list_x.index(value_tqo, 0, tmax)
      tqc=list_x.index(value_tqc, tmax, len(list_x))
      oq1.append((tc-to1)/TOfin)
      oq2.append((tc-to2)/TOfin)
      sq1.append((tmax-to1)/(tc-tmax))
      sq2.append((tmax-to2)/(tc-tmax))
      ciq.append((tc-tmax)/TOfin)
```

```
        qoq.append((tqc-tqo)/TOfin)
        x1=np.concatenate((x1,x))#concatena os pulsos
        tt=ty+tz
        t=np.concatenate((t,tt))
        tz=t[-1]
        jit=np.concatenate((jit,([TOfin])))
    x=x1

    oq1_final.append(np.mean(oq1))
    oq2_final.append(np.mean(oq2))
    sq1_final.append(np.mean(sq1))
    sq2_final.append(np.mean(sq2))
    ciq_final.append(np.mean(ciq))
    qoq_final.append(np.mean(qoq))
    oq1=[]
    oq2=[]
    sq1=[]
    sq2=[]
    ciq=[]
    qoq=[]
    ValorHarmonico=[]
    Harmonico=[]
    dH12=[]
    HRF=[]
    VH=[]
    H=[]
    plt.plot(x[1:1000])

    #Calcula FFT
    NN=int(Tfs/TOfin)+10;
    Fs=20000;
    deltaT=1/Fs;
    YT=fft(x,NN);
    PY=abs(YT);
    f=(Fs)*(np.arange(int(NN/2)))/NN;
    P=20*log10(PY);

    #S  considera o lado das frequ ncias positivas
    import math
    PP=P[1:math.floor(int(NN/2))+1];
    PP2=PP[5:len(PP)];

    #Calcula os picos dos harm nicos
```

```
1200
    V2=max(PP2);
1202 PP2_list=PP2.tolist()
    I2=PP2_list.index(max(PP2_list))
1204
    ValorHarmonico.append(V2);
1206 Harmonico.append(f[I2+4]);
    deltaf=(Fs/(2*len(f)));
1208 Harmonic1=f[I2+4];

1210 #Determina o n mero de formantes que ser o considerados

1212 NumFormantes=10;

1214 for l in range(1,NumFormantes):

1216    novafreq=l*Harmonic1;

1218    #usa uma margem em torno de cada m ximo
       margem=50;
1220
       maxnovafreq=novafreq+margem;
1222    minnovafreq=novafreq-margem;

1224    #Determina o in cio e o fim do per odo no espectro
       #para determinar os m ximos
1226
       Ninic=math.floor(minnovafreq/deltaf);
1228    if (Ninic < 1):
          Ninic = 1;
1230    Nfim=math.floor(maxnovafreq/deltaf);

1232    vetmax=PP[Ninic:Nfim];

1234    valormax=max(vetmax);
       vetmax_list=vetmax.tolist()
1236    indmax=vetmax_list.index(max(vetmax_list))

1238    #Calcula as Amaxs
       ValorHarmonico.append(valormax);
1240    Harmonico.append(f[Ninic+indmax-1]);

1242 soma=0;
```

```
1244      #Calcula HRF - com NumFormantes
          #Escolhemos 10
1246
      for l in range(1,NumFormantes):
1248      soma=soma+ValorHarmonico[l];

1250      #Calcula dH12
      dH12=ValorHarmonico[0]-ValorHarmonico[1];
1252
      HRF=soma/ValorHarmonico[0];
1254
      VH.append(ValorHarmonico)
1256  H.append(Harmonico)

1258  dH12_final.append(dH12)
      HRF_final.append(HRF)
1260
      ValorHarmonico_final.append(np.mean(VH))
1262  Harmonico_final.append(np.mean(H))

1264  j=len(jit)
      jitabs=0
1266
      for i in range(j-1):
1268      jitabs+=abs((jit[i]-jit[i+1]))
      jitabs_final.append(jitabs/(j-1))
1270
      jitrel=jitabs/(sum(jit)/j)
1272  jitrel_final.append(jitrel*100)

1274  #novos parmetros
      soma1=0
1276  for i in range(1,j-1):
          soma1+=abs((2*jit[i]/3.-jit[i-1]/3.-jit[i+1]/3.))
1278  soma1=soma1/(j-2)
      rap=soma1/(sum(jit)/j)
1280  rap_final.append(rap*100)

1282  soma2=0
      for i in range(2,j-2):
1284      soma2+=abs((4*jit[i]/5.-jit[i-1]/5.-jit[i-2]/5.-jit[i+1]/5.-jit[i
        +2]/5.))
```

```
    soma2=soma2/(j-4)
1286 ppq5=soma2/(sum(jit)/j)
    ppq5_final.append(ppq5*100)
1288
    soma3=0
1290 for i in range(1,j-1):
        soma3+=abs((jit[i+1]+jit[i-1]-2*jit[i]))
1292 soma3=soma3/(j-2)
    ddp=soma3/(sum(jit)/j)
1294 ddp_final.append(ddp*100)

1296 print(k)
    print(TOfin)
```

APÊNDICE C – Modelo 3: Pulso glotal de LF e densidade espectral a 2 parâmetros.

```
import os
import matplotlib.pyplot as plt
import numpy as np
from numpy import cos,exp, log10,pi
from scipy.signal import lfilter, freqz
from random import random
from bisect import bisect_left

#Pulso Unitario com Processo Estocastico
def LF_uno_uni_normal(Av,FO,fm,Oq,m,Qa,abar,bbar):

    Tm=1/float(fm)
    TO=1/float(FO)
    xbar=(1.-abar**2/(2*bbar))**0.5
    kbar=10000
    beg=5000
    xx=([])
    xini=0
    ko=1/40000.0
    for ii in range(kbar):
        xfin=([((-float(bbar)*Tm/2.+1.)*xini+abar*np.random.randn()*Tm
    **0.5)/(1+float(bbar)*Tm/2.)])
        xx=np.concatenate((xx,xfin))
        xini=float(xfin[-1])
    dT=ko+(Tm-ko)*(xbar+xx[beg:])**2
    TOfin=sum(dT[0:int(TO/Tm)])
    TOvetor=TOfin*np.ones(int(TO/Tm)+1)
    t=([0])
    for iii in range(len(dT)):
        t=np.concatenate((t,([t[-1]+dT[iii]])))
    t=t[0:int(TO/Tm)+1]
```

```
    x=([])
    k=0
    for i in range(len(t)):
        if t[i]<=Oq*TOfin:
            k+=1

    #criacao do pulso de LF com delta t(t) processo estoc stico
    for i in range(k):
        def f(E):
            return E*Qa*(1-Oq)*TOvetor[i]-1+exp(-E*(1-Oq)*TOvetor[i])
        E=fsolve(f,1000000)
        x0=pi/(m*Oq*TOvetor[i])
        x1=cos(pi/m)
        x2=sin(pi/m)
        x3=TOvetor[i]*(1-Oq)
        x4=exp(E*TOvetor[i]*(1-Oq))
        x5=1/E
        def g(a):
            return(1/(a**2+x0**2))*(a+x0*((exp(-a*Oq*TOvetor[i])-x1)/x2))-
x3/(x4-1)+x5
        a=fsolve(g,0)
        x6=-Av*sin(pi/m)*(a**2+(pi/(m*Oq*TOvetor[i]))**2)/exp(-a*Oq*TOvetor
[i])
        x7=pi/(m*Oq*TOvetor[i])+a*exp(a*Oq*TOvetor[i]*m)*sin(pi)-(pi/(m*Oq*
TOvetor[i]))*exp(a*Oq*TOvetor[i]*m)*cos(pi)
        Ee=x6/x7
        x8=(-Ee*exp(-a*Oq*TOvetor[i])*((pi/(m*Oq*TOvetor[i]))+a*exp(a*t[i])
*sin(pi*t[i]/(Oq*TOvetor[i]*m))-(pi/(m*Oq*TOvetor[i]))*exp(a*t[i])*cos(
pi*t[i]/(Oq*TOvetor[i]*m))))/((sin(pi/m)*(a**2+(pi/(m*Oq*TOvetor[i]))
**2)))
        x=np.concatenate((x,x8))
    for i in range(k,len(t)):
        def f(E):
            return E*Qa*(1-Oq)*TOvetor[i]-1+exp(-E*(1-Oq)*TOvetor[i])
        E=fsolve(f,1000000)
        x0=pi/(m*Oq*TOvetor[i])
        x1=cos(pi/m)
        x2=sin(pi/m)
        x3=TOvetor[i]*(1-Oq)
        x4=exp(E*TOvetor[i]*(1-Oq))
        x5=1/E
        def g(a):
```

```
1066                return(1/(a**2+x0**2))*(a+x0*((exp(-a*Oq*TOvetor[i])-x1)/x2))-
    x3/(x4-1)+x5
          a=fsolve(g,0)
1068      x6=-Av*sin(pi/m)*(a**2+(pi/(m*Oq*TOvetor[i]))**2)/exp(-a*Oq*TOvetor
    [i])
          x7=pi/(m*Oq*TOvetor[i])+a*exp(a*Oq*TOvetor[i]*m)*sin(pi)-(pi/(m*Oq*
    TOvetor[i]))*exp(a*Oq*TOvetor[i]*m)*cos(pi)
1070      Ee=x6/x7
          x9=-Ee*(1/(E*Qa*(1-Oq)*TOvetor[i])-1)*(TOvetor[i]-t[i]+(1-exp(E*(
    TOvetor[i]-t[i])))/E)
1072      x=np.concatenate((x,x9))
     imp=np.zeros(int(TOfin/Tm)+2)
1074 imp[0]=1
     z = np.convolve(x,imp)
1076 return x,t,TOfin

1078 def find_nearest(array, value):
     idx = (np.abs(array - value)).argmin()
1080 return array[idx]

1082 #gerando 200 amostras de a, b, alpha1 e alpha2
     a=[]
1084 for i in range(100, 500, 2):
       a.append(i)
1086 print(len(a))
     b=[]
1088 for i in range(1000000, 5000000, 20000):
       b.append(i)
1090 print(len(b))

1092 Qa=[]
     for i in range(0,23):
1094   for i in range(1, 10,1):
         Qa.append(i/10)
1096 Qa=Qa[0:200]
     print(len(Qa))
1098 Oq=[]
     for i in range(0,23):
1100   for i in range(10,1,-1):
         Oq.append(i/10)
1102 Oq=Oq[0:200]
     print(len(Oq))
1104
```

```
FO=180 #frequencia funcamental
fm=20000 #frequencia de amostragem
Tm=1/float(fm) #periodo de amostragem
Av=7 #amplitude de vozeamento
Tfs=3 #tempo total de sustentacao da vogal
TO=1/float(FO) #periodo fundamental
#Oq=0.8
m=0.75
#Qa=0.3
x1=([]) #criacao do vetor de pulsos LF com TO(t) processo estocastico
g1=([]) #criacao do vetor de pulsos LF Determin stico
jit=([])
todas=([])
t=([])
tz=0
jitabs_final=[]
jitrel_final=[]
rap_final=[]
ppq5_final=[]
ddp_final=[]
oq1=[]
oq2=[]
sq1=[]
sq2=[]
ciq=[]
qoq=[]
oq1_final=[]
oq2_final=[]
sq1_final=[]
sq2_final=[]
ciq_final=[]
qoq_final=[]
dH12_final=[]
HRF_final=[]
Harmonico_final=[]
ValorHarmonico_final=[]
for k in range(len(a)):
  #cria e concatena os pulsos
   for i in range(int(Tfs/TO)+10):
       x,ty,TOfin=LF_uno_uni_normal(Av,FO,fm,Oq[k],m,Qa[k],a[k],b[k]) #cria
    um pulso com TO(t) processo estocastico
       list_x = x.tolist()
       tmax=list_x.index(max(list_x))
```

```
        tc=Oq[k]*TO  #tc
        x_a=x[0:tmax]
        x_d=x[tmax:len(x)]
        value_to1=find_nearest(x_a,0.1*(max(x))).tolist()
        value_to2=find_nearest(x_a,0.05*(max(x))).tolist()
        value_tqo=find_nearest(x_a,0.5*(max(x))).tolist()
        value_tqc=find_nearest(x_d,0.5*(max(x))).tolist()
        to1=list_x.index(value_to1, 0, tmax)
        to2=list_x.index(value_to2, 0, tmax)
        tqo=list_x.index(value_tqo, 0, tmax)
        tqc=list_x.index(value_tqc, tmax, len(list_x))
        oq1.append((tc-to1)/TOfin)
        oq2.append((tc-to2)/TOfin)
        sq1.append((tmax-to1)/(tc-tmax))
        sq2.append((tmax-to2)/(tc-tmax))
        ciq.append((tc-tmax)/TOfin)
        qoq.append((tqc-tqo)/TOfin)
        x1=np.concatenate((x1,x))#concatena os pulsos
        tt=ty+tz
        t=np.concatenate((t,tt))
        tz=t[-1]
        jit=np.concatenate((jit,([TOfin])))
    x=x1

    oq1_final.append(np.mean(oq1))
    oq2_final.append(np.mean(oq2))
    sq1_final.append(np.mean(sq1))
    sq2_final.append(np.mean(sq2))
    ciq_final.append(np.mean(ciq))
    qoq_final.append(np.mean(qoq))

    oq1=[]
    oq2=[]
    sq1=[]
    sq2=[]
    ciq=[]
    qoq=[]
    ValorHarmonico=[]
    Harmonico=[]
    dH12=[]
    HRF=[]
    VH=[]
    H=[]
```

```
1190
    #Calcula FFT
1192 NN=int(Tfs/TOfin)+10;
    Fs=20000;
1194 deltaT=1/Fs;
    YT=fft(x,NN);
1196 PY=abs(YT);
    f=(Fs)*(np.arange(int(NN/2)))/NN;
1198 P=20*log10(PY);

1200 #S  considera o lado das frequ ncias positivas
    import math
1202 PP=P[1:math.floor(int(NN/2))+1];
    PP2=PP[5:len(PP)];
1204
    #Calcula os picos dos harm nicos
1206
    V2=max(PP2);
1208 PP2_list=PP2.tolist()
    I2=PP2_list.index(max(PP2_list))
1210
    ValorHarmonico.append(V2);
1212 Harmonico.append(f[I2+4]);
    deltaf=(Fs/(2*len(f)));
1214 Harmonic1=f[I2+4];

1216 #Determina o n mero de formantes que ser o considerados

1218 NumFormantes=10;

1220 for l in range(1,NumFormantes):

1222    novafreq=l*Harmonic1;

1224    #usa uma margem em torno de cada m ximo
       margem=50;
1226
       maxnovafreq=novafreq+margem;
1228    minnovafreq=novafreq-margem;

1230    #Determina o in cio e o fim do per odo no espectro
       #para determinar os m ximos
1232
```

```
    Ninic=math.floor(minnovafreq/deltaf);
    if (Ninic < 1):
      Ninic = 1;
    Nfim=math.floor(maxnovafreq/deltaf);

    vetmax=PP[Ninic:Nfim];

    valormax=max(vetmax);
    vetmax_list=vetmax.tolist()
    indmax=vetmax_list.index(max(vetmax_list))

    #Calcula as Amaxs
    ValorHarmonico.append(valormax);
    Harmonico.append(f[Ninic+indmax-1]);

  soma=0;

    #Calcula HRF - com NumFormantes
    #Escolhemos 10

  for l in range(1,NumFormantes):
    soma=soma+ValorHarmonico[l];

    #Calcula dH12
  dH12=ValorHarmonico[0]-ValorHarmonico[1];

  HRF=soma/ValorHarmonico[0];

  VH.append(ValorHarmonico)
  H.append(Harmonico)

  dH12_final.append(dH12)
  HRF_final.append(HRF)

  ValorHarmonico_final.append(np.mean(VH))
  Harmonico_final.append(np.mean(H))

  j=len(jit)
  jitabs=0

  for i in range(j-1):
    jitabs+=abs((jit[i]-jit[i+1]))
  jitabs_final.append(jitabs/(j-1))
```

```
1276

    jitrel=jitabs/(sum(jit)/j)
1278    jitrel_final.append(jitrel*100)

1280    #novos parametros
    soma1=0
1282    for i in range(1,j-1):
        soma1+=abs((2*jit[i]/3.-jit[i-1]/3.-jit[i+1]/3.))
1284    soma1=soma1/(j-2)
    rap=soma1/(sum(jit)/j)
1286    rap_final.append(rap*100)

1288    soma2=0
    for i in range(2,j-2):
1290        soma2+=abs((4*jit[i]/5.-jit[i-1]/5.-jit[i-2]/5.-jit[i+1]/5.-jit[i
      +2]/5.))
    soma2=soma2/(j-4)
1292    ppq5=soma2/(sum(jit)/j)
    ppq5_final.append(ppq5*100)
1294

    soma3=0
1296    for i in range(1,j-1):
        soma3+=abs((jit[i+1]+jit[i-1]-2*jit[i]))
1298    soma3=soma3/(j-2)
    ddp=soma3/(sum(jit)/j)
1300    ddp_final.append(ddp*100)

1302    plt.plot(x[1:1000])
    print(k)
```

APÊNDICE D – Modelo 4: Pulso glotal de LF e densidade espectral a 3 parâmetros.

```
import os
import matplotlib.pyplot as plt
import numpy as np
from numpy import cos,exp, log10,pi
from scipy.signal import lfilter, freqz
from random import random
from bisect import bisect_left

#Pulso Unitario com Processo Estocastico
def LF_uno_uni_normal(Av,FO,fm,Oq,m,Qa,abar,bbar,eta):

    alpha=np.array([[0.,1.],[-float(bbar)**2,-2*eta*float(bbar)]])
    beta=np.array([[0],[abar]])
    Tm=1/float(fm)
    TO=1/float(FO)
    xbar=(1.-abar**2/(4*eta*bbar**3))**0.5
    kbar=10000
    beg=3000
    xx=([])
    ko=1/40000.0
    xini=np.array([[0],[0]])
    for ii in range(kbar):
        xfin=(-(alpha)*Tm/2.+np.ones((2,2))).dot(xini)
        xfin=(xfin+beta*np.random.randn()*Tm**0.5)
        xfin=inv((np.ones((2,2))+alpha*Tm/2.)).dot(xfin)
        kfin=([xfin[0][0]])
        xx=np.concatenate((xx,kfin))
        xini=xfin
    dT=ko+(Tm-ko)*(xbar+xx[beg:])**2
    TOfin=sum(dT[0:int(TO/Tm)])
    TOvetor=TOfin*np.ones(int(TO/Tm)+1)
```

```
    t=([0])
    for iii in range(len(dT)):
        t=np.concatenate((t,([t[-1]+dT[iii]])))
    t=t[0:int(TO/Tm)+1]
    x=([])
    k=0
    for i in range(len(t)):
        if t[i]<=Oq*TOfin:
            k+=1

    #criacao do pulso de LF com delta t(t) processo estoc stico
    for i in range(k):
        def f(E):
            return E*Qa*(1-Oq)*TOvetor[i]-1+exp(-E*(1-Oq)*TOvetor[i])
        E=fsolve(f,1000000)
        x0=pi/(m*Oq*TOvetor[i])
        x1=cos(pi/m)
        x2=sin(pi/m)
        x3=TOvetor[i]*(1-Oq)
        x4=exp(E*TOvetor[i]*(1-Oq))
        x5=1/E
        def g(a):
            return(1/(a**2+x0**2))*(a+x0*((exp(-a*Oq*TOvetor[i])-x1)/x2))-
x3/(x4-1)+x5
        a=fsolve(g,0)
        x6=-Av*sin(pi/m)*(a**2+(pi/(m*Oq*TOvetor[i]))**2)/exp(-a*Oq*TOvetor
[i])
        x7=pi/(m*Oq*TOvetor[i])+a*exp(a*Oq*TOvetor[i]*m)*sin(pi)-(pi/(m*Oq*
TOvetor[i]))*exp(a*Oq*TOvetor[i]*m)*cos(pi)
        Ee=x6/x7
        x8=(-Ee*exp(-a*Oq*TOvetor[i])*((pi/(m*Oq*TOvetor[i]))+a*exp(a*t[i])
*sin(pi*t[i]/(Oq*TOvetor[i]*m))-(pi/(m*Oq*TOvetor[i]))*exp(a*t[i])*cos(
pi*t[i]/(Oq*TOvetor[i]*m))))/((sin(pi/m)*(a**2+(pi/(m*Oq*TOvetor[i]))
**2)))
        x=np.concatenate((x,x8))
    for i in range(k,len(t)):
        def f(E):
            return E*Qa*(1-Oq)*TOvetor[i]-1+exp(-E*(1-Oq)*TOvetor[i])
        E=fsolve(f,1000000)
        x0=pi/(m*Oq*TOvetor[i])
        x1=cos(pi/m)
        x2=sin(pi/m)
        x3=TOvetor[i]*(1-Oq)
```

```
1068            x4=exp(E*TOvetor[i]*(1-Oq))
                x5=1/E
1070            def g(a):
                    return(1/(a**2+x0**2))*(a+x0*((exp(-a*Oq*TOvetor[i])-x1)/x2))-
    x3/(x4-1)+x5
1072            a=fsolve(g,0)
                x6=-Av*sin(pi/m)*(a**2+(pi/(m*Oq*TOvetor[i]))**2)/exp(-a*Oq*TOvetor
    [i])
1074            x7=pi/(m*Oq*TOvetor[i])+a*exp(a*Oq*TOvetor[i]*m)*sin(pi)-(pi/(m*Oq*
    TOvetor[i]))*exp(a*Oq*TOvetor[i]*m)*cos(pi)
                Ee=x6/x7
1076            x9=-Ee*(1/(E*Qa*(1-Oq)*TOvetor[i])-1)*(TOvetor[i]-t[i]+(1-exp(E*(
    TOvetor[i]-t[i])))/E)
                x=np.concatenate((x,x9))
1078        imp=np.zeros(int(TOfin/Tm)+2)
            imp[0]=1
1080        z = np.convolve(x,imp)
            return x,t,TOfin
1082

    def find_nearest(array, value):
1084        idx = (np.abs(array - value)).argmin()
            return array[idx]
1086

    #gerando 200 amostras de a, b, alpha1 e alpha2
1088 a=[]
    for i in range(100, 500, 2):
1090    a.append(i)
    a=a[101:200]
1092 print(len(a))
    b=[]
1094 for i in range(1000000, 5000000, 20000):
    b.append(i)
1096 b=b[101:200]
    print(len(b))
1098

    eta=[]
1100 for i in range(1, 201, 1):
        eta.append(i/10)
1102 print(eta)
    eta=eta[101:200]
1104 print(len(eta))

1106 Qa=[]
```

```
for i in range(0,23):
  for i in range(1, 10,1):
    Qa.append(i/10)
Qa=Qa[101:200]
print(len(Qa))
Oq=[]
for i in range(0,23):
  for i in range(10,1,-1):
    Oq.append(i/10)
Oq=Oq[101:200]
print(len(Oq))

FO=180 #frequencia funcamental
fm=20000 #frequencia de amostragem
Tm=1/float(fm) #periodo de amostragem
Av=7 #amplitude de vozeamento
Tfs=3 #tempo total de sustentacao da vogal
TO=1/float(FO) #periodo fundamental
#Oq=0.8
m=0.75
#Qa=0.3
x1=([]) #criacao do vetor de pulsos LF com TO(t) processo estocastico
g1=([]) #criacao do vetor de pulsos LF Determin stico
jit=([])
todas=([])
t=([])
tz=0
jitabs_final=[]
jitrel_final=[]
rap_final=[]
ppq5_final=[]
ddp_final=[]
oq1=[]
oq2=[]
sq1=[]
sq2=[]
ciq=[]
qoq=[]
oq1_final=[]
oq2_final=[]
sq1_final=[]
sq2_final=[]
ciq_final=[]
```

```
qoq_final=[]
dH12_final=[]
HRF_final=[]
Harmonico_final=[]
ValorHarmonico_final=[]
for k in range(len(a)):
  #cria e concatena os pulsos
  for i in range(int(Tfs/TO)+10):
      x,ty,TOfin=LF_uno_uni_normal(Av,FO,fm,Oq[k],m,Qa[k],a[k],b[k],eta[k])
    #cria um pulso com TO(t) processo estocastico
      list_x = x.tolist()
      tmax=list_x.index(max(list_x))
      tc=Oq[k]*TO  #tc
      x_a=x[0:tmax]
      x_d=x[tmax:len(x)]
      value_to1=find_nearest(x_a,0.1*(max(x))).tolist()
      value_to2=find_nearest(x_a,0.05*(max(x))).tolist()
      value_tqo=find_nearest(x_a,0.5*(max(x))).tolist()
      value_tqc=find_nearest(x_d,0.5*(max(x))).tolist()
      to1=list_x.index(value_to1, 0, tmax)
      to2=list_x.index(value_to2, 0, tmax)
      tqo=list_x.index(value_tqo, 0, tmax)
      tqc=list_x.index(value_tqc, tmax, len(list_x))
      oq1.append((tc-to1)/TOfin)
      oq2.append((tc-to2)/TOfin)
      sq1.append((tmax-to1)/(tc-tmax))
      sq2.append((tmax-to2)/(tc-tmax))
      ciq.append((tc-tmax)/TOfin)
      qoq.append((tqc-tqo)/TOfin)
      x1=np.concatenate((x1,x))#concatena os pulsos
      tt=ty+tz
      t=np.concatenate((t,tt))
      tz=t[-1]
      jit=np.concatenate((jit,([TOfin])))
  x=x1

  oq1_final.append(np.mean(oq1))
  oq2_final.append(np.mean(oq2))
  sq1_final.append(np.mean(sq1))
  sq2_final.append(np.mean(sq2))
  ciq_final.append(np.mean(ciq))
  qoq_final.append(np.mean(qoq))
  oq1=[]
```

```
1192    oq2=[]
        sq1=[]
1194    sq2=[]
        ciq=[]
1196    qoq=[]
        ValorHarmonico=[]
1198    Harmonico=[]
        dH12=[]
1200    HRF=[]
        VH=[]
1202    H=[]
        plt.plot(x[1:1000])
1204
        #Calcula FFT
1206    NN=int(Tfs/TOfin)+10;
        Fs=20000;
1208    deltaT=1/Fs;
        YT=fft(x,NN);
1210    PY=abs(YT);
        f=(Fs)*(np.arange(int(NN/2)))/NN;
1212    P=20*log10(PY);

1214    #S  considera o lado das frequ ncias positivas
        import math
1216    PP=P[1:math.floor(int(NN/2))+1];
        PP2=PP[5:len(PP)];
1218
        #Calcula os picos dos harm nicos
1220
        V2=max(PP2);
1222    PP2_list=PP2.tolist()
        I2=PP2_list.index(max(PP2_list))
1224
        ValorHarmonico.append(V2);
1226    Harmonico.append(f[I2+4]);
        deltaf=(Fs/(2*len(f)));
1228    Harmonic1=f[I2+4];

1230    #Determina o n mero de formantes que ser o considerados

1232    NumFormantes=10;

1234    for l in range(1,NumFormantes):
```

```
1236      novafreq=l*Harmonic1;

1238      #usa uma margem em torno de cada m ximo
          margem=50;
1240
          maxnovafreq=novafreq+margem;
1242      minnovafreq=novafreq-margem;

1244      #Determina o in cio e o fim do per odo no espectro
          #para determinar os m ximos
1246
          Ninic=math.floor(minnovafreq/deltaf);
1248      if (Ninic < 1):
            Ninic = 1;
1250      Nfim=math.floor(maxnovafreq/deltaf);

1252      vetmax=PP[Ninic:Nfim];

1254      valormax=max(vetmax);
          vetmax_list=vetmax.tolist()
1256      indmax=vetmax_list.index(max(vetmax_list))

1258      #Calcula as Amaxs
          ValorHarmonico.append(valormax);
1260      Harmonico.append(f[Ninic+indmax-1]);

1262    soma=0;

1264      #Calcula HRF - com NumFormantes
          #Escolhemos 10
1266
        for l in range(1,NumFormantes):
1268      soma=soma+ValorHarmonico[l];

1270      #Calcula dH12
        dH12=ValorHarmonico[0]-ValorHarmonico[1];
1272
        HRF=soma/ValorHarmonico[0];
1274
        VH.append(ValorHarmonico)
1276    H.append(Harmonico)
```

```
1278    dH12_final.append(dH12)
        HRF_final.append(HRF)
1280
        ValorHarmonico_final.append(np.mean(VH))
1282    Harmonico_final.append(np.mean(H))

1284    j=len(jit)
        jitabs=0
1286
        for i in range(j-1):
1288      jitabs+=abs((jit[i]-jit[i+1]))
        jitabs_final.append(jitabs/(j-1))
1290
        jitrel=jitabs/(sum(jit)/j)
1292    jitrel_final.append(jitrel*100)

1294    #novos parametros
        soma1=0
1296    for i in range(1,j-1):
            soma1+=abs((2*jit[i]/3.-jit[i-1]/3.-jit[i+1]/3.))
1298    soma1=soma1/(j-2)
        rap=soma1/(sum(jit)/j)
1300    rap_final.append(rap*100)

1302    soma2=0
        for i in range(2,j-2):
1304        soma2+=abs((4*jit[i]/5.-jit[i-1]/5.-jit[i-2]/5.-jit[i+1]/5.-jit[i
          +2]/5.))
        soma2=soma2/(j-4)
1306    ppq5=soma2/(sum(jit)/j)
        ppq5_final.append(ppq5*100)
1308
        soma3=0
1310    for i in range(1,j-1):
            soma3+=abs((jit[i+1]+jit[i-1]-2*jit[i]))
1312    soma3=soma3/(j-2)
        ddp=soma3/(sum(jit)/j)
1314    ddp_final.append(ddp*100)

1316    print(k)
```

APÊNDICE E – Rede Neural Artificial

```
1000 def rna(n_output,n_input,num_epochs,learning_rate,X_train,y_train,X_test,
       y_test):

1002   callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)

1004   # Estrutura da RNA
       modelo_v1 = tf.keras.Sequential()
1006   modelo_v1.add(tf.keras.layers.Dense(32, input_shape = (n_input,) ,
         kernel_initializer = 'normal', activation = 'relu'))
       modelo_v1.add(tf.keras.layers.Dense(32,  kernel_initializer = 'normal',
         activation = 'relu'))
1008   modelo_v1.add(tf.keras.layers.Dense(32,  kernel_initializer = 'normal',
         activation = 'relu'))
       modelo_v1.add(tf.keras.layers.Dense(32,  kernel_initializer = 'normal',
         activation = 'relu'))
1010   modelo_v1.add(tf.keras.layers.Dense(n_output))

1012   # Compliando e treinando o modelo
       modelo_v1.compile(loss = tf.keras.losses.MSE, optimizer = tf.keras.
         optimizers.Adam(learning_rate))
1014   history = modelo_v1.fit(X_train, y_train, validation_split=0.2, epochs=
         num_epochs, batch_size=10, callbacks=[callback])

1016   # Sum rio do modelo
       print("\nSum rio do modelo:")
1018   modelo_v1.summary()

1020   # Avaliando a performance em treino
       scores_treino = modelo_v1.evaluate(X_train, y_train, verbose = 0)
1022   print("\nErro Final em Treino: "+str(scores_treino))

1024   # Fazendo previs es
       print("\nTestando o Modelo...")
```

```
1026    y_pred = modelo_v1.predict(X_test)

1028    # Avaliando a performance em teste
        scores_teste = modelo_v1.evaluate(X_test, y_test, verbose = 0)
1030    print("\nErro Final em Teste: "+str(scores_teste))

1032    return history,y_pred,scores_teste,scores_treino
```

APÊNDICE F – Matriz de correlação

```
1000 def m_correlacao(dados):

1002   import seaborn as sns

1004   corr = df.corr()

1006   sns.set_theme(style="whitegrid")

1008   f, ax = plt.subplots(figsize=(11, 9))

1010   cmap = sns.diverging_palette(230, 20, as_cmap=True)

1012   mask = np.triu(np.ones_like(corr, dtype=bool))

1014   sns.heatmap(corr, annot=True, mask=mask, cmap=cmap, fmt=".1f",linewidths
         =.6, vmax=1, square=True, cbar_kws={"shrink": .5})


1016

#construindo matriz correlacao
1018 df = pd.DataFrame([a,b,Oq,Qa,jitabs_final,jitrel_final,rap_final,ppq5_final
       ,ddp_final,oq1_final,oq2_final,sq1_final,sq2_final,qoq_final,ciq_final])
     df=df.transpose()
1020 df.columns = ['a','b','Oq','Qa','Jitt_{abs}','Jitt_{relativo}','Jitt_{rap}'
       ,'Jitt_{ppq5}','Jit_{ddp}','OQ_{1}','OQ_{2}','SQ_{1}','SQ_{2}','QOQ','
       CIQ']


1022 m_correlacao(df)
```

APÊNDICE G - Treinamento dos modelos

```
import numpy as np
import matplotlib.pyplot as plt
import random
import tensorflow as tf
import os
import pandas as pd
from sklearn.model_selection import train_test_split
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

# Definindo input e output da RNA

norm_a = [float(i)/max(a) for i in a]
norm_b = [float(i)/max(b) for i in b]
norm_Oq = [float(i)/max(Oq) for i in Oq]
norm_Qa = [float(i)/max(Qa) for i in Qa]

input = pd.DataFrame([jitabs_final,jitrel_final,rap_final,ppq5_final,
    ddp_final,sq1_final,sq2_final,qoq_final])
input = input.transpose()

output = pd.DataFrame([norm_a,norm_b,norm_Oq,norm_Qa])
output=output.transpose()

X_train, X_test, y_train, y_test = train_test_split(input, output,
    test_size = 0.2)

# Hiperpar metros
num_epochs = 100
learning_rate = 0.001
n_output = 4
n_input = 8

history,y_pred,scores_teste,scores_treino = rna(n_output, n_input,
    num_epochs,learning_rate, X_train,y_train,X_test,y_test)
```

```
1032 #plotando MSE para valida   o e treino
     plt.subplots(figsize=(15, 7))
1034 plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
1036 plt.title('Modelo de LF a 2 par metros')
     plt.xlabel(' pocas ')
1038 plt.ylabel('Erro m dio quadr tico')
     plt.legend(['Treino', 'Valida   o'])
```

APÊNDICE H – Gerando vogais

```
import os
import shutil
import matplotlib.pyplot as plt
import numpy as np
from numpy import cos,exp, log10,pi, abs
from scipy.signal import lfilter, freqz
import soundfile as sf
from Pulso_Unitario_LF import LF_uno_uni_normal
from random import random
import time

ini = time.time()

FO=98
a= 255
b= 2560000
Oq= 0.3
Qa= 0.7

try:
    os.mkdir('a='+str(a)[0:6]+'_b='+str(b)+'_F0='+str(FO)+'Hz')
except OSError:
    shutil.rmtree('a='+str(a)[0:6]+'_b='+str(b)+'_F0='+str(FO)+'Hz')
    os.mkdir('a='+str(a)[0:6]+'_b='+str(b)+'_F0='+str(FO)+'Hz')
os.chdir(os.getcwd()+'/a='+str(a)[0:6]+'_b='+str(b)+'_F0='+str(FO)+'Hz')

fm=20000
Tm=1/float(fm)
Av=7
Tfs=3
TO=1/float(FO)
m=0.75
x1=([])
```

```
jit=([])
todas=([])
t=([])
tz=0
for i in range(int(Tfs/TO)):
    x,ty,TOfin=LF_uno_uni_normal(Av,FO,fm,Oq,m,Qa,a,b)
    x1=np.concatenate((x1,x))
    tt=ty+tz
    t=np.concatenate((t,tt))
    tz=t[-1]
    jit=np.concatenate((jit,([TOfin])))
x=x1

#Medidas de jitter
j=len(jit)
jitabs=0
for i in range(j-1):
    jitabs+=abs((jit[i]-jit[i+1]))
jitabs=jitabs/(j-1)
jitrel=jitabs/(sum(jit)/j)
jitrel=jitrel*100
print (jitrel)
soma1=0
for i in range(1,j-1):
    soma1+=abs((2*jit[i]/3.-jit[i-1]/3.-jit[i+1]/3.))
soma1=soma1/(j-2)
rap=soma1/(sum(jit)/j)
rap=rap*100
soma2=0
for i in range(2,j-2):
    soma2+=abs((4*jit[i]/5.-jit[i-1]/5.-jit[i-2]/5.-jit[i+1]/5.-jit[i
    +2]/5.))
soma2=soma2/(j-4)
ppq5=soma2/(sum(jit)/j)
ppq5=ppq5*100
soma3=0
for i in range(1,j-1):
    soma3+=abs((jit[i+1]+jit[i-1]-2*jit[i]))
soma3=soma3/(j-2)
ddp=soma3/(sum(jit)/j)
ddp=ddp*100
arquivo = open('parametros','w')
arquivo.write('jit = '+str(jitrel)+'% \n')
```

```
arquivo.write('rap = '+str(rap)+'% \n')
arquivo.write('ppq5 = '+str(ppq5)+'% \n')
arquivo.write('ddp = '+str(ddp)+'% \n')
arquivo.close()

#Convolucao entre o sinal glotal e a funcao impulso
imp = np.zeros(len(t))
imp[0] = 1
z1 = np.convolve(x,imp)
plt.figure(figsize=(10.0,2.0))
plt.axis((0,1000*sum(jit[0:5]),0,Av+0.1))
plt.plot(1000*t[0:len(x)],z1[0:len(x)],'k',label=u'Delta t(t) Stochastic
    process')
plt.ylabel('Amplitude')
plt.xlabel('Time (ms)')
plt.title(u'Glottal Signal')
plt.savefig('PULSOS_LF_'+'_a='+str(a)+'_b='+str(b)+'_jit'+str(jitrel)[0:5]+
    '%.png', format='png',bbox_inches='tight')
# plt.show()

#Trato Vocal
for H in range(1,6):
    L=['A','E','I','O','U']
    plt.subplot(2,1,1)
    plt.axis((0,3500*TO,0,Av+0.1))
    plt.plot(1000*t,z1[0:len(t)],'b')
    plt.ylabel('Amplitude')
    plt.title(u'EXCITACAO GLOTAL ANTES E DEPOIS DO FILTRO - VOGAL '+str(L[H
    -1]))

    #formantes da categoria de cantor baixo
    formants=[[900,1300,2000,2200,2500],
              [450,1700,2000,2200,2310],
              [300,1900,2100,2200,2490],
              [500,800,2150,2200,2490],
              [360,700,2170,2200,2330]]
    bandwidths=[41,52,70,32,100]
    fmts=formants[H-1]
    if FO>fmts[0]:
        fmts[0]=FO+10
    impulse = np.zeros(500)
    impulse[0] = 1
    xin=impulse
```

```
1116    for resonance in range(5):
            f=fmts[resonance]
1118        bw=bandwidths[resonance]
            num=np.array([1-2*exp(-bw*2*pi/fm)*cos(2*pi*f/fm)+exp(-4*bw*pi/fm)
        ])
1120        den=np.array([1,-2*exp(-bw*2*pi/fm)*cos(2*pi*f/fm),exp(-4*bw*pi/fm)
        ])
            yout=lfilter(num,den,xin,axis=0)
1122        xin=yout

1124    #Radiacao dos Labios (Aproximacao)
        r=0.95
1126    B1=np.array([1,-r])
        A1=np.array([1])
1128    yout=lfilter(B1,A1,np.squeeze(xin))

1130    #convolucao entre os pulso glotais e os filtros do trato vocal
        zout=np.convolve(yout,z1)
1132    ym=zout[0:len(x)]
        ym=ym/max(np.abs(ym))
1134
        #salvar em  udio  cada vogal sintetizada pelo modelo fonte-filtro
1136    sf.write('LF_'+str(L[H-1])+'_a='+str(a)+'_b='+str(b)+'_jitter='+str(
        jitrel)[0:5]+'%.wav', ym, fm)
```