Centro Universitário de Brasília – UniCEUB
Faculdade de Ciências Exatas e Tecnologia – FAET
Curso de Engenharia da Computação
Projeto Final

Aquisição e Tratamento de Dados em um Sistema de Orientação Acústica

Aluno: Mateus Silva Teixeira – R.A.: 20016207
Professor Orientador: MC. José Julimá Bezerra Júnior

Brasília/DF – Junho de 2005

Centro Universitário de Brasília – UniCEUB
Faculdade de Ciências Exatas e Tecnologia – FAET
Curso de Engenharia da Computação
Projeto Final

Aquisição e Tratamento de Dados em um Sistema de Orientação Acústica

Monografia, sob a orientação do Prof. MC. José Julimá Bezerra Júnior avaliado por uma Banca Examinadora do Curso de Engenharia da Computação da Faculdade de Ciências Exatas e Tecnologia - FAET do Centro Universitário de Brasília - UniCEUB e constituiu requisito para obtenção do Título de Bacharel em Engenharia da Computação.

Brasília/DF – Junho de 2005

A Sebastião José da Silva (*in memoriam*)

Agradecimentos

A Deus, pela vida.

Aos pais, pela oportunidade de poder estudar, e acreditarem nesse ideal.

Aos meus irmãos, pelo incentivo aos estudos.

A minha namorada, pelo carinho e afeto.

Aos parentes, pela força e confiança.

Aos amigos de faculdade. Tivemos muitos momentos de estudos que me ajudaram bastante. Diversão e muito estudo fizeram parte desse longo período.

Aos amigos da banda, pelo apoio, força e oração.

Ao professores MC. Wladimir Silva Meyer e MC. José Julimá Bezerra Júnior, pelo incentivo, confiança e dedicação.

Ao coordenador do curso de Engenharia da Computação, MC. Abiezer Amarília.

Aos monitores dos laboratórios, pela ajuda, atenção e paciência.

Ao professor da disciplina de Projeto Final, MC. Francisco Javier, por sempre incentivar-me a realizar este projeto.

Aos demais professores do Curso de Engenharia da Computação, por todo o conhecimento transmitido ao longo do curso.

E de modo especial aos grandes amigos: Júlio César, Henrique Rodrigues, Adriano Delfino, Fernanda Sakamoto e Cezário Neto.

Resumo

Este projeto consiste no desenvolvimento de um sistema capaz de determinar a direção angular de uma fonte de áudio, podendo posteriormente ser integrado a sistemas de outras naturezas, como sistemas de segurança e vídeo-conferência.

O uso da teoria da norma quadrática, *norma 2*, nos sinais de áudio discretizados, possibilita estabelecer uma característica destes sinais. Destaca-se, no modelo proposto, o uso de Rede Neural Artificial na estimação da direção angular da fonte acústica, rede esta implementada na linguagem C/C++, visando possibilitar sua portabilidade para plataformas móveis e independentes de um microcomputador.

Palavras-Chaves: Redes Neurais, Algoritmo *back-propagation*, orientação acústica, tratamento de sinais.

Abstract

This project consisted of the development of a system which is able to determine the direction of arrival angle of an acoustic source, which may later be integrated to other types of systems, such a videoconference systems and security systems.

The use of the quadratic norm theory, *norm 2*, in discrete audio signals, permitted the handling and use of the signal characteristics. In the proposed model, an artificial Neural Network was used to estimate the angular direction of the acoustic source, with the network developed in C/C++, allowing it to be portable to mobile and independent platforms in personal computers.

Keywords; neural networks, *back-propagation* algorithms, acoustic orientation, signal treatment.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos e Siglas

ASCII	*"American Standard Code for Information Interchange"*
BIT	*"Binary Digit*" – Menor unidade de informação
DARPA	*"Defense Advanced Research Projects Agency"*
fs	Freqüência do sinal
GB	Giga Byte
GHz	Giga Hertz
GMDH	*"Group Method Data Handling"*
GRNN	*"General Regression Neural Networks"*
IEEE	*"Institute of Electrical and Electronics Engineers"*
INNS	*"International Neural Networks Society"*
MB	Mega Byte
MHz	Mega Hertz
MLP	*"Multi-layer Perceptron*" – *Perceptron* multi-camadas
PC	*"Personal Computer"* – Computador Pessoal
PNN	*"Probabilistic Neural Networks"*
RAM	*"Radom-Access Memory"*
RNA	Rede Neural Artificial
.txt	Formato de arquivo texto que suporta somente caracteres

1 – Introdução

1.1 – Motivação

A aquisição e o tratamento de dados de áudio tem sido cada vez mais assunto de grandes pesquisas. Dentre estas linhas de pesquisa, destaca-se a coleta de um sinal de áudio, onde, através do tratamento desse sinal, pode-se determinar de qual direção o sinal foi emitido.

Como aplicabilidade, essa determinação pode ser utilizada em um ambiente de vídeo-conferência, pois sabendo a direção da fonte de áudio, tem-se a possibilidade de encontrar, mais precisamente, o locutor. Em sistemas de segurança, também se pode fazer uso dessa solução.

O uso de Redes Neurais Artificiais tem como objetivo, fazer com que o computador se comporte de forma semelhante ao ser humano, pois, crianças de 03 ou 04 anos têm a capacidade de distinguir perfeitamente uma cor da outra, mas um computador, que efetua grandiosos cálculos matemáticos, muitas vezes não tem a capacidade de definir tais situações. Ou seja, o objetivo principal das redes neurais é generalizar resultados a partir de conhecimento anterior que se tem do problema, conhecimento este utilizado para treinar a rede. [Siqueira, 2004]

1.2 – Objetivos e Considerações

O trabalho proposto tem como objetivo determinar de qual direção angular uma fonte de áudio emitiu um som, utilizando Redes Neurais Artificiais para generalizar as informações, considerando que:

- os sinais terão a mesma intensidade e freqüência;
- os sinais estão sendo comparados utilizando a mesma distância entre a fonte de áudio e o microfone;
- os sinais foram gravados e tratados para posteriormente serem treinados e testados na Rede Neural;
- o resultado será mostrado entre 0º e 90º.

Para realização do trabalho tem-se 4 (quatro) etapas:

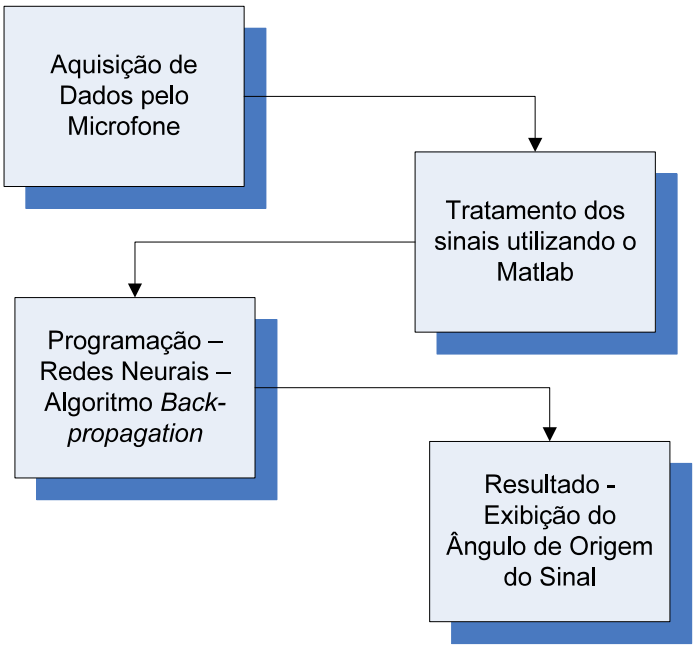

Figura 1.1 – Etapas do projeto.

Como mostra a Figura 1.1, será realizada a aquisição dos dados com a utilização de um microfone. Após, os dados serão discretizados e tratados pelo software Matlab[1]. Depois de tratados os dados, uma aplicação criada em C/C++[2] baseada em Redes Neurais irá determinar de qual direção o sinal foi gerado. E por último, será mostrado de qual direção angular, aproximadamente, foi gerado o sinal de áudio.

1.3 – Sinopse dos Capítulos

O Capítulo 2, suporte teórico, trata da história dos estudos sobre Redes Neurais, tipos, estruturas, topologias, aprendizagem, e sobre o algoritmo *back-propagation*. A norma quadrática e as funções utilizadas têm como objetivo preparar os dados para serem utilizados na aplicação

O Capítulo 3 tem como objetivo demonstrar de que forma é realizada a aquisição acústica para que se possa realizar o tratamento dos dados, as etapas e estrutura do protótipo utilizado na validação da estrutura proposta para a determinação da direção da fonte acústica, as principais funções utilizadas, como é realizado o treinamento e a classificação de novos padrões, material e passos para o uso da aplicação criada que tem

[1] Matlab: software matemático otimizado para executar cálculos científicos e de engenharia.
[2] C/C++: linguagem de programação.

como finalidade determinar, por simulação, qual direção angular foi gerado o sinal de áudio. Neste capítulo encontram-se também os passos para realizar a simulação utilizando a aplicação criada.

Por fim, no Capítulo 4, têm-se as considerações finais, os resultados obtidos, dificuldades encontradas para realização do projeto e as sugestões para trabalhos futuros.

2 – Suporte Teórico

Este capítulo tem como objetivo apresentar, de forma sucinta, o que são as redes neurais, suas principais características, aplicabilidades, tipos, como funcionam, etc.

A aplicação fará uso de uma rede neural utilizando o algoritmo *back-propagation* com o objetivo de "treinar" a rede para determinar a direção da fonte de áudio, e posteriormente, a rede ser capaz de determinar a direção dessa fonte para novas situações, ou seja, que não foram treinadas.

A norma quadrática e as funções utilizadas têm como finalidade tratar e preparar os dados para que a aplicação, criada em C/C++ baseada no algoritmo *back-propagation*, possa fazer uso dessas informações para determinar a direção da fonte de áudio.

2.1 – Redes Neurais

2.1.1 – O que são Redes Neurais Artificiais?

O cérebro humano é considerado o mais fascinante processador baseado em carbono existente, sendo composto por aproximadamente 10 bilhões de neurônios. Todas as funções e movimentos do organismo estão relacionados ao funcionamento destas pequenas células. Os neurônios estão conectados uns aos outros através de sinapses, e juntos formam uma grande rede, chamada *Rede Neural*. As sinapses transmitem estímulos através de diferentes concentrações de Na+ (Sódio) e K+ (Potássio), e o resultado disto pode ser estendido por todo o corpo humano. Esta grande rede proporciona uma fabulosa capacidade de processamento e armazenamento de informação.

O sistema nervoso é formado por um conjunto extremamente complexo de neurônios. Nos neurônios a comunicação é realizada através de impulsos, quando um impulso é recebido, o neurônio o processa, e passado um limite de ação, dispara um segundo impulso que produz uma substância neurotransmissora o qual flui do corpo celular para o axônio (que por sua vez pode ou não estar conectado a um dendrito de outra célula). O neurônio que transmite o pulso pode controlar a freqüência de pulsos aumentando ou diminuindo a polaridade na membrana pós-sináptica. Eles possuem um papel essencial na determinação do funcionamento, comportamento e do raciocínio do ser humano. Ao contrário das redes neurais artificiais, redes neurais naturais não transmitem sinais negativos, sua ativação é medida pela freqüência com que emitem pulsos, freqüência esta

de pulsos contínuos e positivos [Braga, Carvalho, Ludernir, 2000] [Tatibana, Kaetsu, 2004].

Os principais componentes dos neurônios são:

- ***Dentritos***, que têm por função, receber os estímulos transmitidos pelos outros neurônios;
- ***Corpo de neurônio***, também chamado de ***soma***, que é responsável por coletar e combinar informações vindas de outros neurônios;
- ***Axônio***, que é constituído de uma fibra tubular que pode alcançar até alguns metros, e é responsável por transmitir os estímulos para outras células. [Tatibana, Kaetsu, 2004]

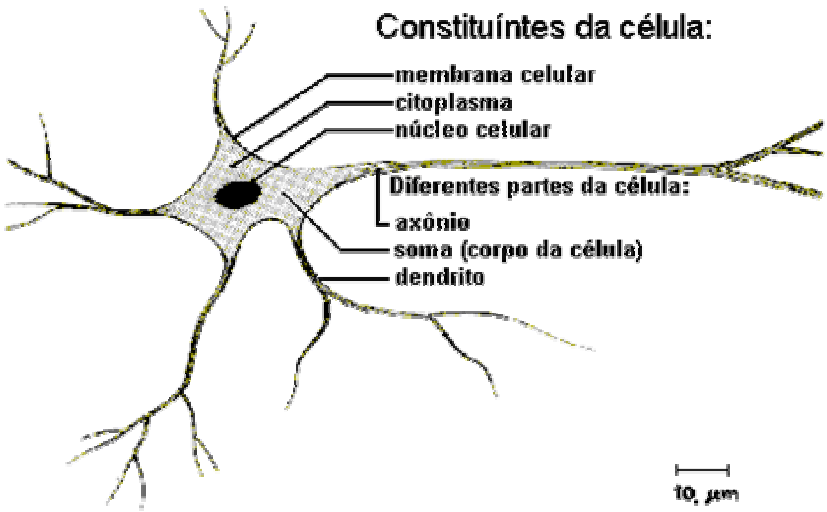

Figura 2.1 – Estrutura da célula neural.

A área de Redes Neurais Artificiais (RNAs), também conhecida como *conexionismo* ou sistemas de processamento paralelo ou distribuído, teve seus estudos iniciados por McCulloch e Pitts (1943).

Este modelo de computação não-algorítmica, é caracterizado por sistemas que, em certos níveis, relembram a estrutura do cérebro humano. Por não ser baseada em regras ou programas, a computação neural se constitui em uma alternativa à computação algorítmica convencional.

RNAs são sistemas paralelos distribuídos compostos por unidade de processamento simples (nodos) que calculam determinadas funções matemáticas, em sua grande parte, são

não-lineares. Essas unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, normalmente unidirecionais. Estas conexões, na maioria dos modelos, estão associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio da rede. Enfim, o funcionamento dessas redes é inspirado na estrutura do cérebro humano.

Para que as redes neurais venham a ter um bom desempenho, é necessária uma fase de aprendizagem, onde, um conjunto de exemplos é apresentado para a rede, a qual extrai automaticamente as características necessárias para representar a informação fornecida. Estas características são utilizadas posteriormente para gerar respostas para o problema.

Sem dúvidas, o atrativo principal da solução de problemas usando RNAs é a capacidade de *aprender* através de exemplos e de *generalizar* a informação aprendida. A generalização que está associada à capacidade de a rede aprender através de um conjunto reduzido de exemplos e posteriormente dar respostas coerentes para dados não-conhecidos, é uma demonstração de que a capacidade das redes neurais vai muito além do que simplesmente mapear relações de entrada e saída. As RNAs são capazes de extrair informações não-apresentadas de forma explicita através dos exemplos. [Braga, Carvalho, Ludernir, 2000] [Haykin, 2001]

2.1.2 – Histórico

Os primeiros estudos sobre redes neurais datam de 1943, através dos trabalhos e artigos de McCulloch e Pitts, em que sugeriam a construção de uma máquina baseada ou inspirada no cérebro humano. Muitos outros artigos e livros surgiram desde então, porém, por um longo período de tempo, pouco resultado foi obtido. Até que em 1949, Donald Hebb escreveu um livro intitulado *"The Organization of Behavior"* (A Organização do Comportamento) que perseguia a idéia de que o condicionamento psicológico clássico está presente em qualquer parte dos animais pelo fato de que esta é uma propriedade de neurônios individuais. Suas idéias não eram completamente novas, mas Hebb foi o primeiro a propor uma lei de aprendizagem específica para as sinapses dos neurônios. Este primeiro e corajoso passo serviram de inspiração para que muitos outros pesquisadores perseguissem a mesma idéia. E, embora muito tenha sido estudado e publicado nos anos que seguiram (1940-1950), estes serviram mais como base para desenvolvimento posterior do que para o próprio desenvolvimento.

Implementações de redes neurais através de circuitos analógicos já existiam nos anos 50. Para reproduzir o comportamento do cérebro humano, pensava-se que, bastaria

construir uma rede neural suficientemente grande. No entanto, uma detalhada análise matemática deixou claro o pouco poder computacional de um dos modelos de rede mais utilizados naquele período, o *perceptron*, que não é capaz, por exemplo, de ser aplicado com sucesso em problemas de classificação que não fossem linearmente separáveis.

Em 1956 no *"Darthmouth College"* nasceram os dois paradigmas da Inteligência Artificial, a *simbólica* e o *conexionista*.

A Inteligência Artificial Simbólica tenta simular o comportamento inteligente humano desconsiderando os mecanismos responsáveis por tal. Já a Inteligência Artificial Conexionista acredita que, construindo um sistema que simule a estrutura do cérebro, este sistema apresentará inteligência, ou seja, será capaz de aprender, assimilar, errar e aprender com seus erros.

O primeiro neuro computador a obter sucesso (*Mark I Perceptron*) surgiu entre 1957 e 1958, criado por Frank Rosemblatt, Charles Wightman e outros. Devido à profundidade de seus estudos, suas contribuições técnicas e de sua maneira moderna de pensar, muitos o vêem como o fundador da neuro computação na forma em que a temos hoje. Seu interesse inicial para a criação do *Perceptron* era o reconhecimento de padrões. Rosemblatt mostrou em seu livro (*Principles of Neurodynamics*) o modelo dos *"Perceptrons"*. Nele, os neurônios eram organizados em camada de entrada e saída, onde os pesos das conexões eram adaptados a fim de se atingir a eficiência sináptica. [Braga, Carvalho, Ludernir, 2000] [Tatibana, Kaetsu, 2004]

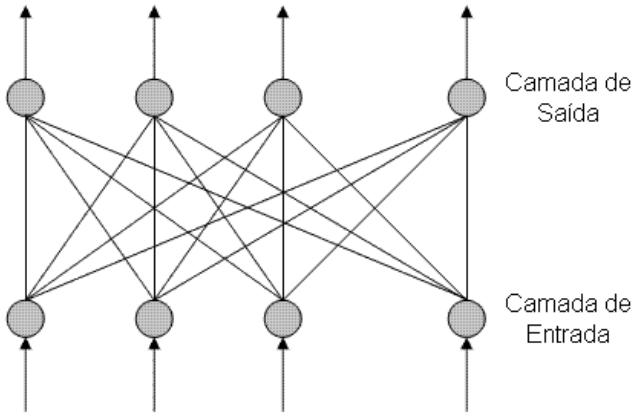

Figura 2.2 – Rede de perceptrons proposta por Rosemblatt.

Após Rosemblatt, Bernard Widrow, com a ajuda de alguns estudantes, desenvolveu um novo tipo de elemento de processamento de redes neurais chamado de Adaline, equipado com uma poderosa lei de aprendizado, que diferente do *Perceptron* que ainda

permanece em uso. Widrow também fundou a primeira companhia de hardware de neurocomputadores e componentes.

Infelizmente, os anos seguintes foram marcados por um entusiasmo exagerado de muitos pesquisadores, que passaram a publicar mais e mais artigos e livros que tinham uma previsão pouco confiável para a época, sobre máquinas tão poderosas quanto o cérebro humano que surgiriam em um curto espaço de tempo. Isto tirou quase toda a credibilidade dos estudos desta área e causou grandes aborrecimentos aos técnicos de outras áreas.

Um período de pesquisa silenciosa seguiu-se durante 1967 a 1982, quando poucas pesquisas foram publicadas devido aos fatos ocorridos anteriormente. Entretanto, aqueles que pesquisavam nesta época, e todos os que se seguiram no decorrer desses anos conseguiram, novamente, estabelecer um campo concreto para o renascimento da área.

Os anos 80 trouxeram de volta o interesse por essa fascinante área, principalmente em função do aparecimento de novos modelos de redes neurais e de algoritmos de aprendizado mais poderosos. Na literatura, podem ser encontrados diversos modelos de redes neurais e algoritmos de aprendizado. Entretanto, este documento irá se basear no modelo de aprendizado denominado retro-propagação do erro ou *error back-propagation*. No item 2.1.11 irá ser detalhando como esse algoritmo funciona.

Muitos pesquisadores foram bastante corajosos e passaram a publicar diversas propostas para a exploração de desenvolvimento de redes neurais bem como suas aplicações. Porém, o fato mais importante deste período ocorreu quando Ira Skurnick, um administrador de programas da DARPA (*Defense Advanced Research Projects Agency*) decidiu ouvir os argumentos da neuro computação e seus projetistas e, divergindo dos caminhos tradicionais dos conhecimentos convencionais, fundou em 1983 pesquisas em neuro computação. Este ato não só abriu as portas para a neuro computação, como também deu à DARPA o status de uma das líderes mundiais em se tratando de "moda" tecnológica.

Em 1987 ocorreu em São Francisco – Estado Unidos, a primeira conferência de redes neurais em tempos modernos, a *IEEE International Conference on Neural Networks*, e também foi formada a *International Neural Networks Society* (INNS). A partir destes acontecimentos, decorreu a fundação do *INNS Jornal* em 1989, seguido do *Neural Computation* e do *IEEE Transactions on Neural Networks* em 1990.

Desde 1987, muitas universidades anunciaram a formação de institutos de pesquisa e programas de educação em neuro computação. [Tatibana, Kaetsu, 2004] [Haykin, 2001]

2.1.3 – Tipos de Rede

Existem dois tipos básicos de rede neurais: ***supervisionadas*** e ***não supervisionadas***:

Redes supervisionadas

Constroem modelos, os quais classificam padrões ou executam predições de acordo com outros padrões de "entradas" e "saídas" que eles aprenderam.

Este tipo de rede apresenta a resposta mais razoável baseada em uma variedade de padrões de aprendizado. Em uma rede supervisionada, mostra-se à rede como fazer predições, classificações, ou decisões, fornecendo a ela um número de classificações corretas ou predições das quais ela pode aprender. Redes de Retropropagação, GRNN (*"General Regression Neural Networks"*), PNN (*"Probabilistic Neural Networks"*), e GMDH (*"Group Method Data Handling"*) são exemplos de redes supervisionadas. [Guahyba, 2004]

Redes não-supervisionadas

Pode classificar um conjunto de padrões de treinamento em um número especificado de categorias sem ser mostrado antes como categorizar. A rede faz isto agrupando os padrões de entrada. Ela os agrupa por sua proximidade em um espaço dimensional N, onde N é o número de "entradas". O usuário diz à rede o número máximo de categorias e ela geralmente agrupa os dados em certo número de categorias. Entretanto, a rede pode não ser capaz de separar os padrões naquelas muitas categorias opcionais. Redes Kohonen são exemplos de redes não supervisionadas. [Guahyba, 2004]

Nenhum tipo de rede garante uma resposta absolutamente correta, especialmente se os padrões são de alguma forma incompletos ou conflitantes.

Os resultados devem ser avaliados em termos da porcentagem de respostas corretas que resultaram do modelo.

A esse respeito, a tecnologia é similar ao funcionamento do neurônio biológico depois deste ter sido "projetado", e difere significantemente de todos outros programas convencionais de computador. Redes Neurais podem não trabalhar bem em algumas aplicações. Alguns problemas são bem apropriados para a capacidade de reconhecimento de padrões de uma rede neural e outros são mais bem resolvidos com métodos tradicionais. [Haykin, 2001] [Braga, Carvalho, Ludernir, 2000]

2.1.4 – A Estrutura da Rede

O bloco de construção básico da tecnologia de redes neurais é o **neurônio** simulado (descrito na Figura 2.3 como um círculo). Neurônios independentes são de pouco uso, a menos que eles estejam interconectados em uma rede de neurônios. A rede processa um número de **entradas** ("*inputs*") do "mundo externo" para produzir uma **saída** ("*output*"), que são as classificações da rede ou predições. Os neurônios são conectados por **pesos** (retratados como linhas), os quais são aplicados a valores passados de um neurônio para o outro.

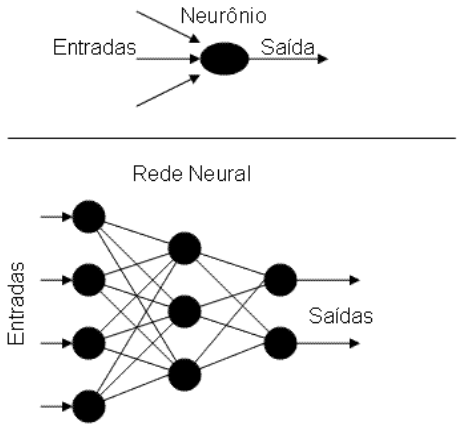

Figura 2.3 – Exemplo de estrutura de uma rede neural.

Um grupo de neurônios é chamado de ***slab***. Os neurônios também são agrupados em **camadas** por suas conexões com o "mundo externo". Por exemplo, se um neurônio recebe dados de fora da rede, ele é considerado como estando na **camada de entrada**. Se um neurônio contém as classificações ou predições da rede, ele está na **camada de saída**. Neurônios entre as camadas de entrada e saída estão na(s) **camada(s) oculta(s)** ou **escondida(s)**. Uma camada pode conter uma ou mais slabs de neurônios. [Guahyba, 2004]

2.1.5 – Topologias das Redes Neurais

Redes Feed-Forward

São redes com uma ou mais camadas de processadores, sendo que o fluxo de dados é sempre em uma única direção, ou seja, não existe realimentação.

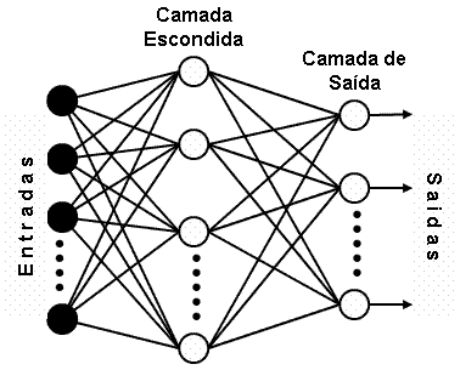

Figura 2.4 – Rede de múltiplas camadas do tipo Feed-Forward.

Redes Recorrentes

São redes com conexões entre processadores da mesma camada e/ou com processadores das camadas anteriores (realimentação). [Vellasco, 2004]

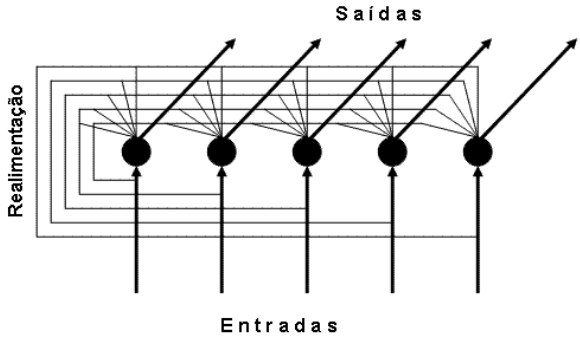

Figura 2.5 – Rede recorrente – com realimentação.

2.1.6 – Função de Ativação

A função de ativação, também conhecida como função restritiva, restringe a amplitude da saída de um neurônio. Essa função limita (restringe) o intervalo permissível de amplitude do sinal de saída de um valor finito. [Haykin, 2001].

Dentre as funções mais utilizadas, temos: [Meyer, 1997]

- Degrau: $$f(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases} \quad (2.1)$$

- Degrau Simétrico: $$f(x) = \begin{cases} 1, x > 0 \\ -1, x < 0 \end{cases} \quad (2.2)$$

- Linear: $$f(x) = x \quad (2.3)$$

- Sigmóide: $$f(x) = \frac{1}{1+e^{-x}} \quad (2.4)$$

- Tansigmóide: $$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2.5)$$

2.1.7 – Processamento de uma Rede Neural

O processamento de uma Rede Neural pode ser dividido em duas fases:

- *Learning* – processo de atualização dos pesos sinápticos para aquisição do conhecimento – aquisição de informação.

- *Recall* – processo de cálculo da saída da rede, dado certo padrão de entrada – recuperação de informação. [Vellasco, 2004]

2.1.8 – Aprendizado

Uma rede neural típica é uma Rede de Retropropagação, a qual geralmente tem 3 camadas de neurônios. Valores de entrada na primeira camada são pesados e passados para a segunda camada. Neurônios na camada oculta "atiram" ou produzem saídas que são baseadas na soma dos valores pesados passados por elas. A camada oculta passa valores à camada de saída da mesma forma, e a camada de saída produz os resultados desejados (predições ou classificações).

A rede "aprende", ajustando os pesos das interconexões entre as camadas. As respostas que a rede está produzindo são repetidamente comparadas com as respostas corretas, e cada vez os pesos das conexões são ajustados na direção das respostas corretas.

Eventualmente, se o problema pode ser aprendido, um conjunto estável de pesos adaptativamente evolui e irá produzir boas respostas para todas as decisões ou predições da

amostra. O real poder das redes neurais é evidente quando a rede treinada é capaz de produzir bons resultados para dados os quais ela nunca "viu" antes.

O maior segredo para construir com sucesso redes neurais é saber quando parar de treinar. Se você treina muito pouco, a rede não irá aprender os padrões. Se você treinar muito, a rede irá aprender com o problema ou memorizar os padrões de treinamento e não generalizar bem com novos padrões.

Dessa forma, concluí-se que “aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados através de uma forma continuada de estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular como ocorrem os ajustes realizados nos parâmetros.” [Braga, Carvalho, Ludernir, 2000]

Aprendizado supervisionado

Nesse tipo de aprendizado, a rede é treinada através do fornecimento dos valores de entrada e de seus respectivos valores desejados de saída, geralmente efetuado através do processo de minimização do erro calculado na saída. [Vellasco, 2004]

A entrada e saída desejada são fornecidas por um supervisor (professor) externo. O objetivo é ajustar os parâmetros da rede, de forma a encontrar uma ligação entre os pares de entrada e saída fornecidos.

Na Figura 2.6 é ilustrado o mecanismo de aprendizado supervisionado. O “professor” indica um comportamento bom ou ruim para a rede, visando a direcionar o processo. A rede tem sua saída corrente (calculada) comparada com a saída desejada, recebendo informações do supervisor sobre o erro da resposta final. A cada padrão de entrada submetido à rede, compara-se a resposta desejada (que representa uma ótima ação para ser realizada pela rede) com a resposta calculada, ajustando-se os pesos das conexões para minimizar o erro.

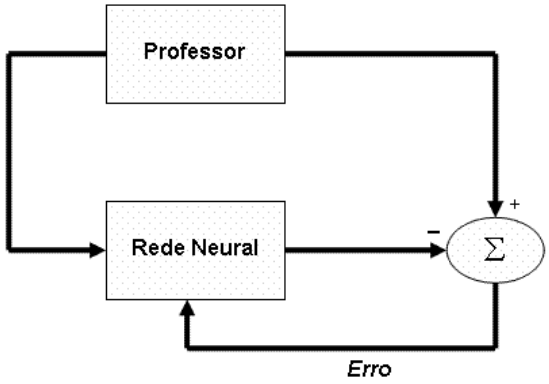

Figura 2.6 – Aprendizado supervisionado.

A desvantagem do aprendizado supervisionado é que, na ausência do professor, a rede não conseguirá aprender novas estratégias para novas situações não treinadas. Um exemplo de algoritmo para aprendizado supervisionado é o *back-propagation*. [Braga, Carvalho, Ludernir, 2000]

Aprendizado não-supervisionado

Como o próprio nome sugere, no aprendizado não-supervisionado não há um professor ou supervisor para acompanhar o processo de aprendizado. A Figura 2.7 ilustra este método.

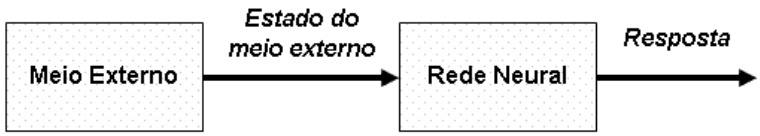

Figura 2.7 – Aprendizado não-supervisionado.

O aprendizado não supervisionado não requer o valor desejado de saída da rede. O sistema extrai as características do conjunto de padrões, agrupando-os em classes inerentes aos dados. [Braga, Carvalho, Ludernir, 2000]

As redes neurais artificiais caracterizam-se portanto:

- pelo grande número de elementos de processamento bem simples, inspirado no funcionamento de um neurônio;

- pelo grande número de conexões entre esses elementos de processamento. Cada conexão tem peso associado, este peso representa quão forte é a interação ou acoplamento entre elementos de processamento e se a sua natureza é excitatória ou inibitória;
- pelo controle altamente paralelo e distribuído;
- pela capacidade de um aprendizado através de observações de um conjunto de exemplos;
- pela manutenção do desempenho na presença de ruído e capacidade de lidar com dados incompletos.

2.1.9 – Uso da Rede Neural

Algumas utilizações atualmente empregando as Redes Neurais Artificiais: diagnóstico de doenças, identificação de compostos químicos, análise de testes médicos, controle de processo, predição de horas de trabalho necessárias para procedimentos industriais, identificação bacteriana, análise de custos, diagnóstico diferencial, otimizando resultados de experimentos biológicos, análise nutricional, experimentos agrícolas, gerenciamentos de recursos hídricos, análise a processamento de sinais, classificação de dados, controle de processos, análise de voz, análise de imagens, etc. [Braga, Carvalho, Ludernir, 2000] [Haykin, 2001]

2.1.10 – Perceptron

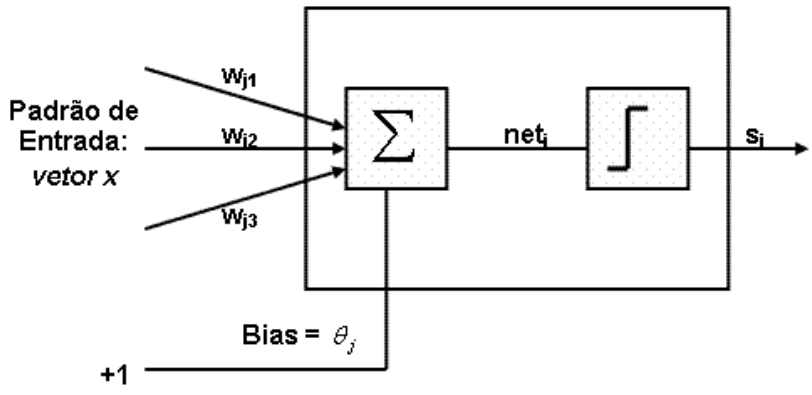

Figura 2.8 – Modelo Perceptron.

O modelo *Perceptron*, utiliza a função de ativação Degrau, tem a topologia em uma única camada. Normalmente, utiliza o aprendizado supervisionado para o treinamento da rede. Sua regra de propagação é baseada na seguinte equação:

$$net_j = \sum x_i . w_{ji} + \theta_j \tag{2.6}$$

onde:

x_i = vetor de entrada;

w_{ji} = peso atribuído em cada entrada;

θ_j = *bias*, desvio.

Uma das modelagens para o funcionamento de um neurônio consiste em um elemento de processamento que possuí um número *X* de entradas e uma única saída (s_j).

As conexões dos neurônios são conhecidas como sinapses, por onde os sinais gerados pelas saídas dos neurônios fluem. Cada sinapse possui associada a si um número real conhecido como peso, cujo módulo representa o ganho que o sinal recebe quando percorre a sinapse, enquanto o sinal do peso representa a natureza da ligação entre os neurônios: negativo para ligações ditas inibitórias e positivas para ligações ditas excitatórias.

Como visto anteriormente, uma rede neural consiste em um conjunto de neurônios conectados por sinapses. Dentre os tipos de redes neurais, neste trabalho, vamos nos deter a topologia conhecida como *Perceptron* multi-camadas (*Multi-layer Perceptron* – MLP).

As redes MLPs resolvem problemas mais complexos do que o *Perceptron* simples e Adaline, pois as camadas intermediárias simplificam o problema para a camada de saída. [Vellasco, 2004]

Esse tipo de rede, MLP, possui uma ou mais camadas intermediárias. Na Figura 2.9 é ilustrado um *perceptron* com 3 camadas. Os pontos de entrada não realizam nenhum processamento, eles simplesmente distribuem o sinal que recebem para os neurônios da camada posterior. Uma camada é conectada apenas às camadas adjacentes a ela, e a saída de cada neurônio é conectada a uma entrada de todos os neurônios na camada posterior, num arranjo conhecido como camadas completamente conectadas. O fluxo do sinal ocorre sempre no mesmo sentido, da entrada para a saída. Este é um exemplo de rede não-realimentada.

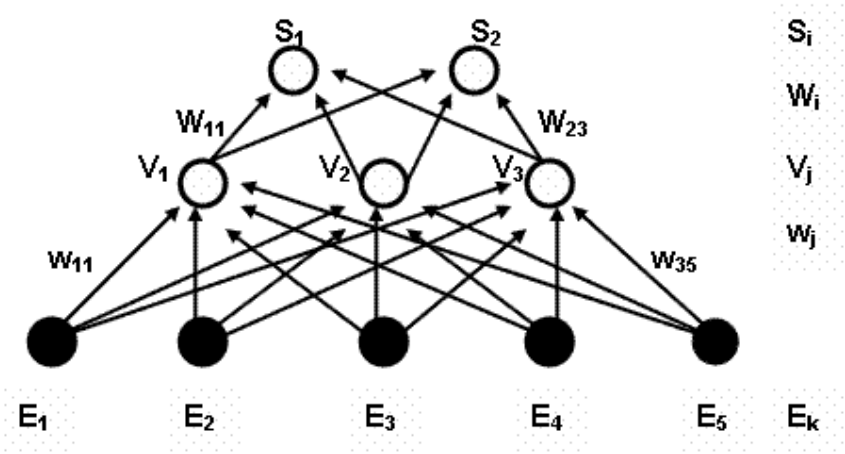

Figura 2.9 – *Perceptron* com uma camada oculta.

A camada *S* é a camada de saída da rede neural, enquanto *V* recebe o nome de camada escondida e *E* de camada de entrada. Uma rede pode possuir mais de uma camada escondida. E entre uma camada e outra, cada neurônio sofre a ação de um peso, *W*.

Seja *M* o número de neurônios da camada escondida *V*, e *N* o número de entradas de uma rede neural. Seja um vetor de entrada E^{μ} onde cada componente é um número real representado por E_k^{μ}, para $1 \le k \le N$. Se esse vetor E^{μ} for apresentado à rede, cada unidade na camada escondida V_j realiza o seguinte somatório:

$$h_j^{\mu} = \sum_{k=1}^{N} w_{jk} E_k^{w} \tag{2.7}$$

e produz uma saída:

$$V_j^{\mu} = G\left(h_j^{\mu}\right) = G\left(\sum_{k=1}^{N} w_{jk} E_k^{\mu}\right) \tag{2.8}$$

cada unidade na camada de saída realiza o somatório:

$$h_i^{\mu} = \sum W_{ij} V_j^{\mu} = \sum_{j=1}^{M} W_{ij} \left(\sum_{k=1}^{N} w_{jk} E_k^{\mu}\right) \tag{2.9}$$

e produz o resultado final:

$$S_i^{\mu} = G\left(h_i^{\mu}\right) = G\left(\sum_{j=1}^{M} W_{ij} G\left(\sum_{k=1}^{N} E_k^{\mu} w_{jk}\right)\right)$$

(2.10)

Nota-se que a saída S^{μ} gerada pela rede para uma entrada E^{μ} é uma função não linear de E^{μ}. O comportamento dessa função, uma vez fixada a função de ativação *G* de cada neurônio, depende exclusivamente do valor dos pesos das sinapses. [Liporace, Machado, Barbosa 1994]

É importante que a função de ativação *G* de cada neurônio seja não linear, já que isso irá determinar o mapeamento $E^{\mu} \rightarrow S^{\mu}$ realizado pela rede, e assim, capaz de representar mapeamentos mais complexos do que os representáveis por uma função linear. O objetivo é descobrir uma configuração de pesos que realize este mapeamento. Para tal procedimento, usaremos o treinamento usando o algoritmo *back-propagation*, descrito a seguir. [Liporace, Machado, Barbosa 1994]

2.1.11 – Perceptrons de Múltiplas Camadas (MLP)

As redes neurais conhecidas como *Perceptrons de Múltiplas Camadas - Multilayer Perceptron* (MLP), consistem de um conjunto de unidades sensoriais (nós de fonte) que constituem a camada de entrada, uma ou mais camadas ocultas, ou escondidas, e uma camada de saída. O sinal de entrada vai se propagando para frente através da rede, camada por camada.

Para resolver diversos problemas difíceis, os Perceptrons de Múltiplas Camadas têm sido aplicados com sucesso, sendo utilizado em seu treinamento de forma supervisionada o algoritmo de *retropropagação de erro (error back-propagation)*. [Haykin, 2001]

Treinamento de Redes MLP

Dentre os algoritmos existentes para o treinamento de redes neurais de múltiplas camadas, o mais conhecido é o algoritmo de retropropagação de erros, ou simplesmente, *back-propagation*. O algoritmo *back-propagation* é um algoritmo do tipo supervisionado que usa de pares de entrada e saída desejada para, por meio de um mecanismo de correção de erros, ajustarem os pesos da rede.

O seu treinamento ocorre em duas fases, em que cada fase percorre a rede em um sentido de ida e em um sentido de volta, sendo essas fases denominadas de fase *forward* e fase *backward*, respectivamente. Para um dado padrão de entrada, a fase *forward* é usada para definir sua saída na rede. Já a fase *backward*, usa saída fornecida e a saída desejada para realizar a atualização dos pesos e de suas conexões.

Na Figura 2.10 é ilustrado o fluxo do algoritmo *back-propagation*, no sentido *forward*, os dados vão da entrada para a saída, e no sentido *backward*, os erros, da saída para a entrada.

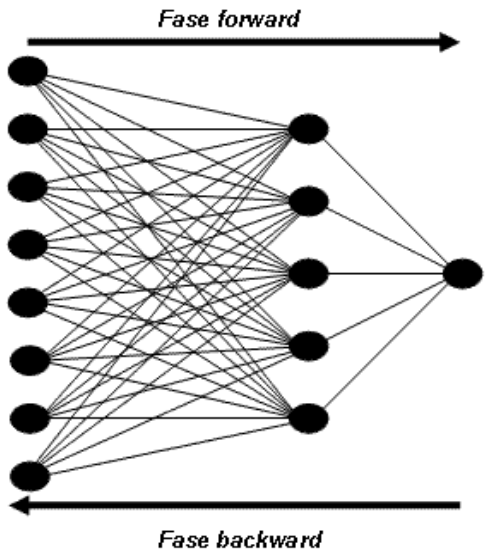

Figura 2.10 – Fluxo do algoritmo *back-propagation*.

Passos da fase forward, de acordo com [Braga, Carvalho, Ludernir, 2000] :

1. Entrada é apresentada a primeira camada da rede, camada C^0.
2. Para cada camada C^i a começar da camada de entrada:
   2.1. Após os nodos da camada C^i ($i > 0$) calcularem seus sinais de saída, estes servem como entrada para a definição das saídas produzidas pelos nodos da camada C^{i+1}.
3. As saídas produzidas pelos nodos da última camada são comparadas às saídas desejadas.

Passos da fase backward, de acordo com [Braga, Carvalho, Ludernir, 2000] :

1. A partir da última camada, até chegar à camada de entrada:
   1.1. Os nodos da camada atual ajustam seus pesos de forma a reduzir seus erros.

1.2. Os erros de um nodo das camadas intermediárias são calculados utilizando os erros dos nodos da camada seguinte conectados a ele, ponderados pelos pesos das conexões entre eles.

Sendo assim, o algoritmo *back-propagation*, que utiliza essas duas fases é apresentado a seguir: [Braga, Carvalho, Ludernir, 2000]

1. Inicializar pesos e parâmetros.
2. Repetir até os erros serem mínimos ou até a realização de um dado número de ciclos:
   2.1. Para cada padrão de treinamento X:
      2.1.1. Definir saída da rede através da fase *forward*.
      2.1.2. Comparar saídas produzidas com as saídas desejadas.
      2.1.3. Atualizar pesos dos nodos através da fase *backward*.

O algoritmo *back-propagation* baseia-se na regra delta, que foi proposta por Widrow e Hoff, sendo denominada *regra delta generalizada*, e é definida como:

$$\Delta\omega_{ij} = \eta\delta_p x_{pj} \tag{2.11}$$

onde,

$\Delta\omega_{ij}$ = variação no valor do peso que interliga os neurônios;

η = parâmetro da taxa de aprendizagem;

x_{pj} = sinal de entrada do neurônio j;

δ_p = gradiente local, definido como: $\delta p = \left(t_{pi} - a_{pi}\right)$, sendo:

t_{pi} = valor esperado na saída do neurônio *i* para um padrão *p* de entrada;

a_{pi} = valor calculado na saída do neurônio *i* para um padrão de entrada *p*.

Com a utilização desse algoritmo, é possível definir os erros dos nodos das camadas intermediárias, possibilitando o ajuste de seus pesos. Esse ajuste de peso é realizado usando o método do gradiente.

Taxas de aprendizado estão relacionadas ao passo de incremento (decremento) de seus pesos. Sendo que um valor inicial é definido pelo usuário ou é obtido através de algumas variações do algoritmo *back-propagation*. [Meyer, 1997]

Através da retropropagação do erro quadrático (Ep) das saídas, obteve-se a equação (2.11), por meio das seguintes equações:

$$\Delta\omega_{ij} = -\eta \frac{\partial E_p}{\partial \omega_{ij}} \tag{2.12}$$

$$E_p = \sum_i \left(t_{pi} - a_{pi}\right)^2 \tag{2.13}$$

Constitui o núcleo da regra de aprendizado, a derivada parcial do erro quadrático com relação a cada peso ω_{ij} das sinapses que chegam ao neurônio *i*. Se os pesos sofrem alteração sob a equação (2.11), esse erro quadrático se move na superfície de erro em direção a um ponto de mínimo, conforme é ilustrado na Figura 2.11.

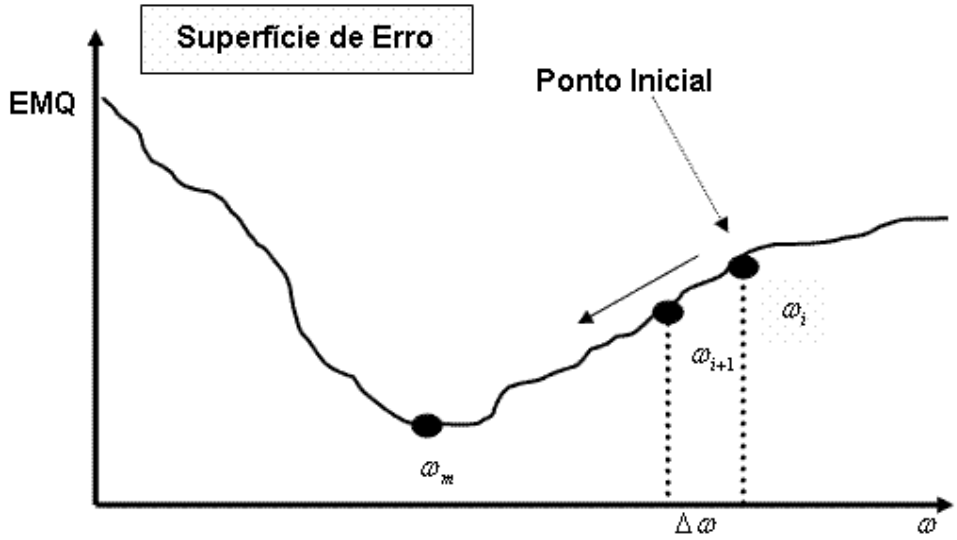

Figura 2.11 – Sentido da movimentação na superfície de erro.

Um dos problemas encontrados nas redes *back-propagation* é que a superfície de erro pode tender a um ponto de mínimo local, podendo o algoritmo convergir para um desses pontos ao invés de convergir para o mínimo geral. Para contornar esse problema, deve-se reiniciar o treinamento escolhendo outro conjunto de pesos iniciais, sendo que assim, o ponto de partida na superfície de erro é diferente do primeiro, tendo-se uma maior chance de evitar-se o ponto determinado ponto de mínimo local. Se o erro quadrático atinge valores toleráveis, dizemos então que a rede "*aprendeu*" a reconhecer os padrões que a ela foram apresentados e que está rede já está "*treinada*". [Meyer, 1997]

Com objetivo de melhorar o desempenho da rede no processo de treinamento e evitar mínimos locais, a adição de um termo *momentum* é uma das técnicas mais freqüentes. Está técnica é bastante utilizada por ser simples e efetiva. O termo *momentum* é representado pela equação (2.14): [Braga, Carvalho, Ludernir, 2000] [Meyer, 1997]

$$\Delta\omega_{ij}(n+1) = \eta(\delta_{pi}a_{pi}) + m\Delta\omega_{ij}(n) \tag{2.14}$$

Com a inclusão do termo *momentum*, o treinamento é acelerado em regiões muito planas da superfície de erros. Além disso, suprime oscilações de pesos em vales e ravinas. Na Figura 2.12 é ilustrado o efeito do uso do termo *momentum* no caminho seguido pela rede durante a fase de aprendizado. [Braga, Carvalho, Ludernir, 2000]

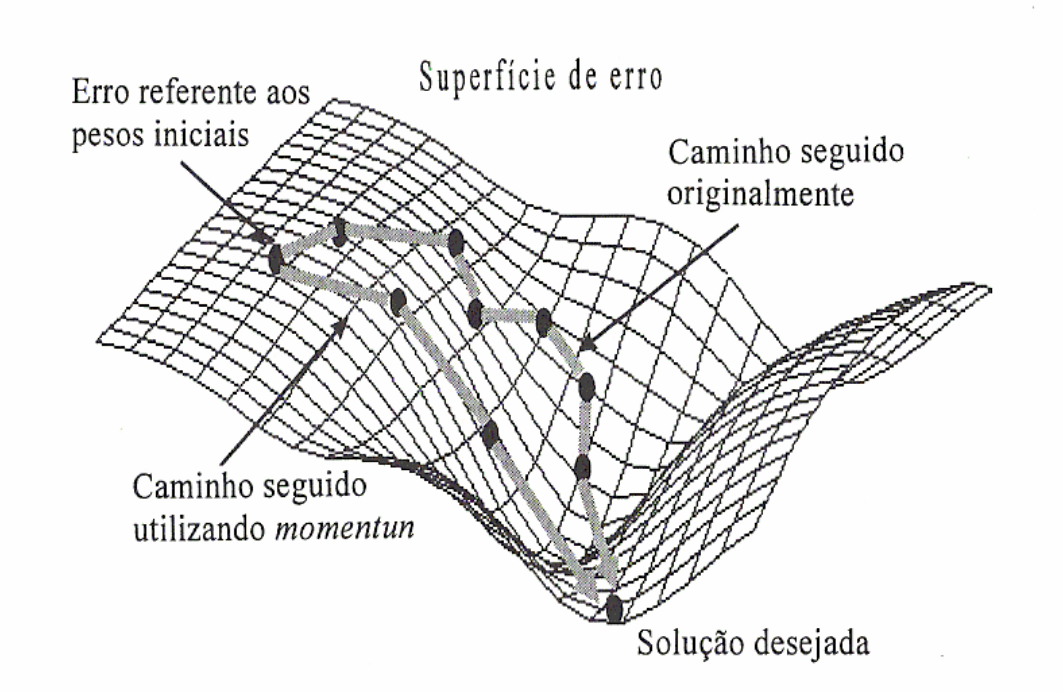

Figura 2.12 – Influência do termo *momentum*.

Durante o treinamento de uma rede MLP é possível à ocorrência de *overfitting*, ou seja, a rede em vez de melhorar o seu desempenho, a rede começa a ter a sua generalização degrada, a sua taxa de acertos para padrões diferentes daqueles utilizados para os ajustes de pesos piora. Dizemos então que a rede "memorizou" os padrões de treinamento. As alternativas mais usadas para reduzir a ocorrência de *overfitting* são:

- encerrar o treinamento mais cedo (quando o erro de validação começa a subir);
- podar (*punning*) os pesos mais cedo.

Modos de Treinamento Seqüencial (Estocástico) e por Lote

Na aplicação prática do algoritmo de retropropagação, o aprendizado se resulta das várias apresentações de um determinado conjunto de exemplos de treinamento para o perceptron de múltiplas camadas. Como citado em [Haykin, 2001], "uma apresentação completa do conjunto de treinamento inteiro é denominado *época.*"

O processo de aprendizagem é mantido em uma base de época em época até os pesos sinápticos e os níveis de *bias* (desvio) se estabilizarem e o erro médio quadrado sobre todo conjunto de treinamento e convergir para um valor mínimo. [Haykin, 2001]

Em um determinado conjunto de treinamento de uma rede *back-propagation*, a aprendizagem por retropropagação pode proceder de uma entre duas formas:

1. ***Treinamento Seqüencial***: também conhecido como estocástico. Neste tipo de treinamento, a atualização dos pesos se dá após a apresentação de cada exemplo que é apresentado a rede. Este tipo de treinamento é utilizado em problemas em que temos uma grande quantidade de padrões de treino, pois os pesos, por serem modificados rapidamente, podem levar a uma convergência mais rápida. [Haykin, 2001] [Meyer, 1997]
2. ***Treinamento por Lote***: no treinamento por lote, o ajuste dos pesos é realizado após a apresentação de todos os exemplos de treinamento que constituem uma época. Este tipo de treinamento exige mais tempo para que os pesos sejam atualizados, mas é mais usado com algoritmos que alteram a taxa de aprendizagem dinamicamente. [Haykin, 2001]

2.2 – Norma Quadrática e Funções Utilizadas

Para realizar o tratamento dos dados, fez-se necessário a utilização do software Matlab 7.0, onde foi realizada a discretização dos sinais. O Matlab é um software especializado e otimizado para realizar cálculos científicos e de engenharia. É um programa de fácil utilização que contém diversas funções matemáticas que facilitam várias operações. [Chapman, 2003]

Dentre as funções que estão sendo usadas nesse projeto, destaca-se:

- *wavread()*[3] ;
- *norma 2;*
- *dec2bin();*

A função *wavread()* é utilizada para realizar a leitura dos sinais de áudio, a função *norma 2 (norm)* calcula o comprimento de um vetor; e a função *dec2bin()* transforma um número decimal em um número binário. [Matsumoto, 2004]

Com os dados do sinal que foi lido utilizando a função *wavread()*, forma-se um vetor. Portanto, o sinal de áudio é tratado como um vetor de dimensão R^n, em que n é o número de amostras do sinal. Após a leitura dos sinais, é necessário determinar o comprimento do vetor. Tal comprimento é calculado utilizando-se a função *norma 2*. [Boldrini, Costa, Figueiredo, Wetzler, 1981]

O cálculo da *norma 2* é dado pela expressão abaixo:

$$\|P_1,...,P_n\|_2 = \sqrt{|P_1|^2 + |P_2|^2 + ... + |P_n|^2} \tag{2.15}$$

onde,

$P_1,...,P_n$ = elementos do vetor (amostra de dados);

n = quantidade de amostras de dados.

O resultado obtido com o uso da função *norma 2* é convertido para binário através da função *dec2bin()*. A transformação para binário se faz necessário, pois o algoritmo criado em C/C++ trabalha com números binários.

Após a conversão dos dados para binário, esses valores são armazenados em um arquivo de extensão *.txt* para que o algoritmo possa realizar a leitura dos dados e treinar a rede neural.

Para que possa ser realizada a determinação da direção da fonte de áudio, utiliza-se um programa baseado em Redes Neurais Artificiais , na configuração *back-propagation*.

Foi necessário o uso da função *norma 2* para determinar uma característica dos sinais de áudio gravados, nesse caso o comprimento de cada vetor, e assim, poder converte

[3] w*avread()*: função que realiza a leitura de um sinal .wav no software Matlab.

esses dados em binário para que a rede possa determinar a direção angular da fonte de áudio.

Os dados tratados são usados para o treinamento da rede neural, a fim de capacitar a rede a generalizar resultados para novos dados apresentados.

Este resultado é mostrado através da aplicação criada para a avaliação dos resultados, onde um algoritmo foi desenvolvido utilizando a linguagem de programação C/C++. [Schildt, 1997] [Meyer, 1997]

3 – Determinação da Direção da Fonte de Áudio

Este capítulo tem como escopo a exposição das etapas do projeto, onde foi realizada a aquisição dos sinais de áudio, tratamento dos dados, simulação, treinamento e testes da aplicação criada com intuito de realizar a determinação da direção da fonte de áudio.

3.1 – Gravação dos Sinais

Para elaboração do projeto, fez-se necessário a gravação de sinais de áudio para realização de uma simulação. Os sinais foram gravados em um ambiente simulado, composto por um tabuleiro (Figura 3.1) e um microfone.

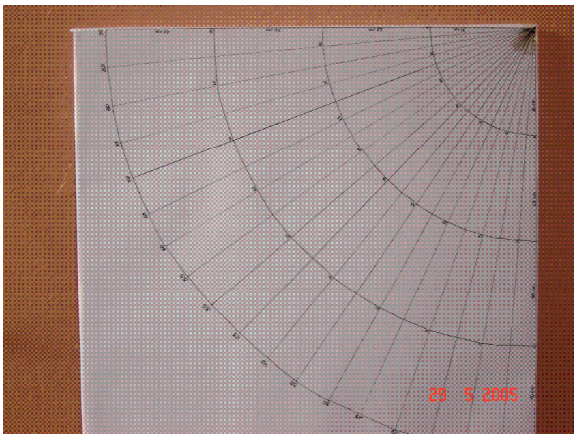

Figura 3.1 – Tabuleiro usado na simulação.

O tabuleiro foi divido em quatro linhas, distantes a 10 cm, 20 cm, 30 cm e 40 cm do microfone, e essas linhas dividas em ângulos entre 0° e 90° graus, separados de 5 em 5 graus.

Durante a gravação dos sinais, realizaram-se experimentos com 3 (três) microfones. O primeiro microfone (Figura 3.2), com abertura nas laterais dificultou o tratamento dos dados, pois essas aberturas capturam o ruído externo em várias direções.

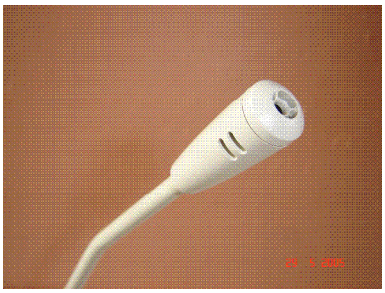

Figura 3.2 – Microfone com abertura nas laterais.

O segundo microfone (Figura 3.3), microfone de Lapela ML-70 Le Son, teve como problema a sua alta sensibilidade, capturando assim um ruído externo que prejudicaria o tratamento dos dados.

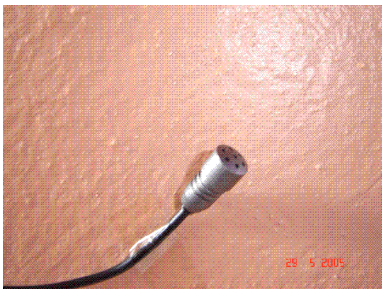

Figura 3.3 – Microfone de Lapela.

Já o terceiro microfone (Figura 3.4) testado durante a gravação dos sinais, foi que apresentou o melhor resultado, e assim o escolhido para ser usado no projeto. O microfone escolhido possui como característica ser um microfone unidirecional, e sem aberturas laterais como as existentes no primeiro microfone testado.

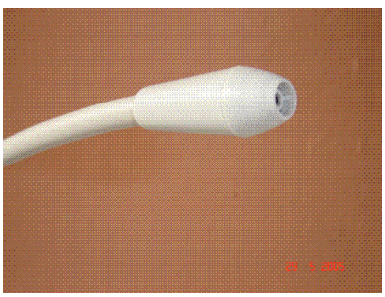

Figura 3.4 – Microfone utilizado no projeto.

Foram gravados cinco sinais de áudio em cada ângulo e em cada linha, totalizando 380 (trezentos e oitenta) sinais de áudio.

Como fonte de áudio foi utilizado um aparelho celular Nokia 1100 (Figura 3.5). Através do uso do aparelho celular como fonte de áudio, tem-se a possibilidade de garantir que o sinal terá sempre a mesma intensidade, freqüência e distância em relação ao microfone.

Figura 3.5 – Aparelho celular Nokia 1100, usado como fonte de áudio para simulação.

Os sinais de áudio foram gravados pelo software Cool Edit versão 1.53 fornecido pela Syntrillium Software Corporation. Após a gravação dos sinais, foi coletada uma parte desse sinal, 1 (um) segundo, para serem tratados e testados.

3.2 – Tratamento de Dados

3.2.1 – Tratamento dos Sinais Gravados

Após a gravação e a extração de uma parte de cada sinal, com duração de 1 (um) segundo, esses sinais foram tratados pelo software Matlab 7.0.

Primeiramente, realizou-se a leitura dos sinais de áudio utilizando a função *wavread()*. Como o sinal, após ser lido, está sendo tratado como um vetor, a função *norma 2*, descrita no tópico 2.2 calcula o comprimento desse vetor com objetivo de dar uma característica desse sinal para que a aplicação possa realizar a determinação da direção angular.

3.2.2 – Gráficos da Função *Norma 2*

Com o uso da função *norma 2*, nota-se que o comprimento do sinal varia de acordo com o ângulo em que o sinal foi gerado. Através dos gráficos abaixo, vê-se que quanto maior o ângulo da fonte de áudio em relação ao microfone, menor o comprimento do sinal.

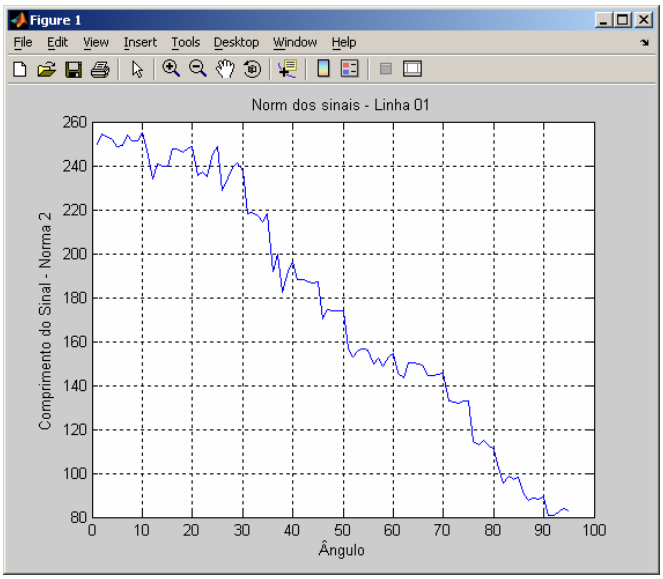

Figura 3.6 – Gráfico da função *norma 2* dos sinais gravados na linha 01.

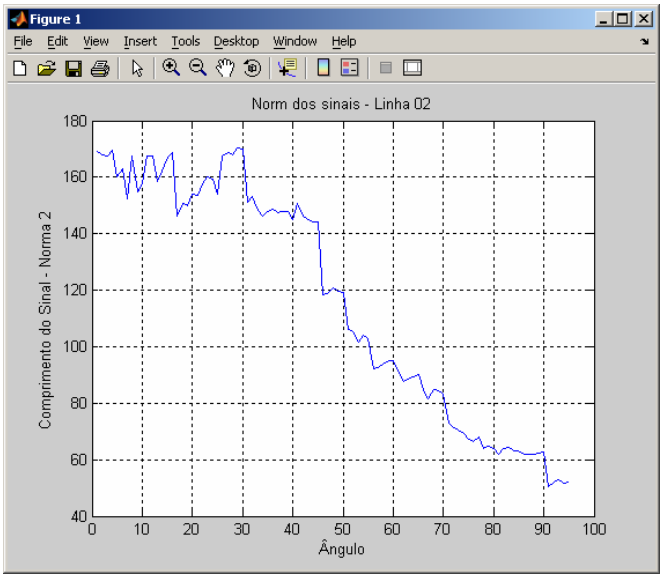

Figura 3.7 – Gráfico da função *norma 2* dos sinais gravados na linha 02.

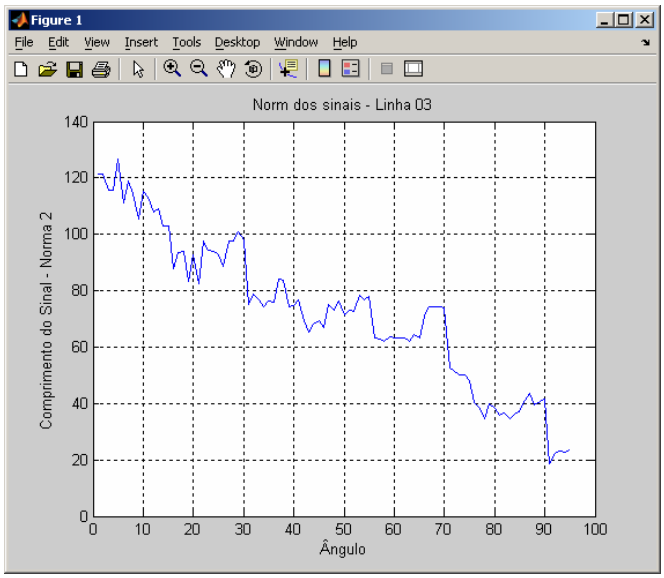

Figura 3.8 – Gráfico da função *norma 2* dos sinais gravados na linha 03.

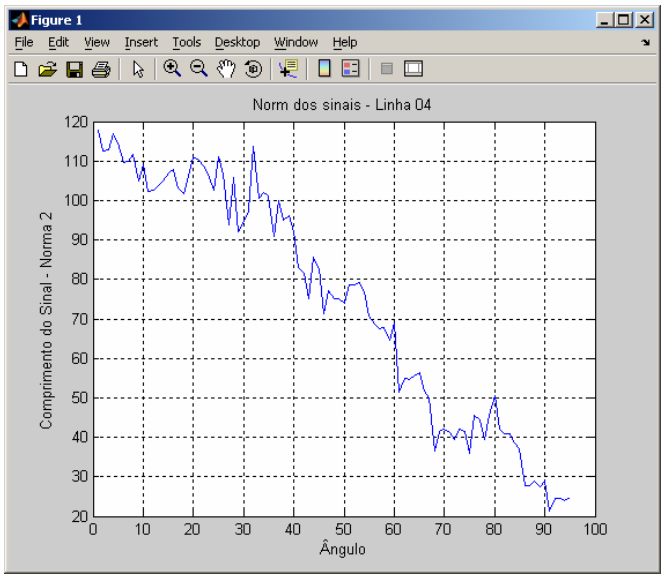

Figura 3.9 – Gráfico da função *norma 2* dos sinais gravados na linha 04.

Pelos gráficos acima, verifica-se que o comprimento do sinal, além de diminuir com aumento do ângulo da fonte de áudio em relação ao microfone, também varia com o aumento da distância entre o microfone e a fonte de áudio. Com isso, pode-se dizer que, a *norma 2* (comprimento) do sinal está relacionada com a sensibilidade do microfone em captar os sinais sonoros emitidos.

Na Tabela 3.1 são mostrados os comprimentos máximos e mínimos obtidos com a gravação dos sinais em cada linha.

Tabela 3.1 – Comprimento máximo e mínimo dos sinais em cada linha.

Linha	Distância entre fonte de áudio e microfone	Comprimento máximo do sinal	Comprimento mínimo do sinal
01	10 cm	254.8013	80.5730
02	20 cm	170.2843	50.3649
03	30 cm	126.8136	18.6193
04	40 cm	117.9248	21.2236

Com o aumento da distância da fonte de áudio em relação ao microfone, houve uma diminuição do comprimento do sinal, pois como mostrado na Tabela 3.1, quando a fonte de áudio estava a 10 cm do microfone, o comprimento máximo foi de 254.8013, e quando estava a 40 cm, o comprimento máximo foi de 117.9248.

Através das Figuras 3.6 a 3.9, também se pode notar que existem alguns "picos" que podem acarretar divergências na determinação da fonte de áudio, mas nota-se que a *norma 2* tende a diminuir com o aumento do ângulo da fonte de áudio em relação ao microfone. Tais "picos" devem ter ocorridos devido à variação do ambiente e ruídos externos existentes durante a gravação dos sinais.

3.2.3 – Resumo do Código para Tratamento dos Dados

A seguir é apresentado o fluxograma do algoritmo usado para o tratamento dos dados.

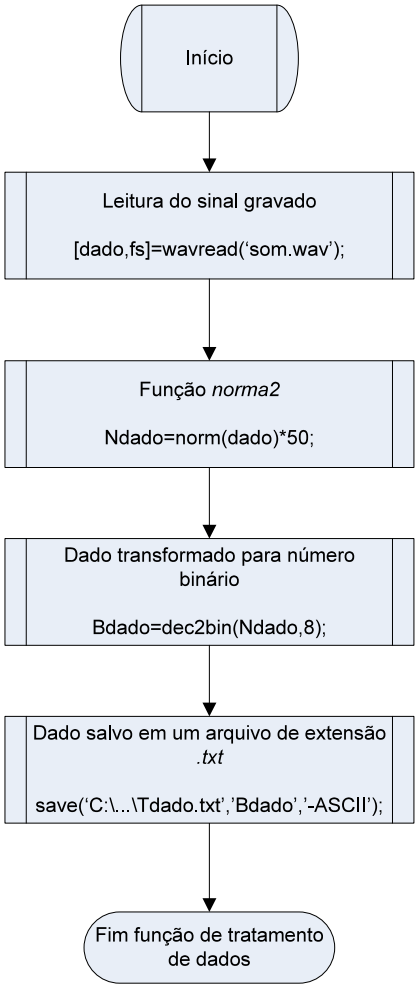

Figura 3.10 – Fluxograma do algoritmo de tratamento de dados.

O fluxograma acima descreve de forma resumida as etapas de tratamento de dados. O algoritmo completo encontra-se no Anexo A.

De acordo com a Figura 3.10, nota-se que a valor da função *norma 2 (norm)* foi multiplicado por 50, isso ocorreu porque os valores obtidos com o uso dessa função eram baixos, assim multiplicou-se todos os valores do comprimento do sinal com o intuito de obter-se valores que melhor se ajustasse ao projeto em execução. Tal procedimento foi adotado, de multiplicar o valor da *norma 2* por uma constante, para que os dados, quando transformados em binário, tivessem uma faixa de valores aceitáveis para serem tratados pela aplicação. Essa faixa de valores deve ficar entre 0 e 255, que corresponde em número binário, 00000000 e 11111111, respectivamente.

Deve-se ressaltar que este algoritmo foi usado para cada linha distante do microfone, ou seja, por quatro vezes, alterando os dados de entrada (som gravado), intermediários (*norm* e número binário) e os dados de saída (arquivo de extensão *.txt* com informações sobre os sinais gravados).

Fez-se necessário a conversão dos dados para número binário, pois a primeira proposta do projeto utilizava um circuito que tinha como objetivo, capturar, tratar e converter os sinais analógicos para digital utilizando um conversor Analógico/Digital e enviar esses dados para o PC via porta paralela.

A aplicação foi criada utilizando como entrada os números 0 e 1, e determinou-se que as entradas da rede teriam 8 bits, pois é a quantidade de saídas fornecidas pelo conversor.

Dessa forma, com a substituição do circuito pela placa de som e os softwares Matlab e Cool Edit, fez-se necessário a conversão dos dados para binário com intuito de utilizar a aplicação criada.

Ressalva-se que, mesmo com as alterações ocorridas, o objetivo do projeto permaneceu, o de determinar por simulação a direção angular da fonte de áudio.

Informações sobre o circuito que estava sendo criado e sobre o conversor analógico/digital estão nos anexos B e C, respectivamente.

3.3 – Estrutura do Algoritmo de Aplicação

Após o tratamento dos dados pelo software Matlab, onde foram gerados arquivos de extensão *.txt* com os dados de cada sinal, uma aplicação irá treinar a rede neural e classificar novos padrões apresentados à rede. A linguagem de programação adotada foi C/C++, e o compilador utilizado o Borland C++ Builder 5. [Schildt, 1997]

O algoritmo é composto pelos seguintes procedimentos: [Meyer, 1997]

- **InicializaPeso() –** Este procedimento atribui pesos às sinapses da rede. Os pesos são valores aleatórios variando entre -0,1 e 0,1;
- **InicializaBias() –** Este processo atribui valores às entradas *bias* (desvio) dos neurônios. Da mesma forma como o procedimento InicializaPeso(), este procedimento atribui valores aleatórios entre -0,1 e 0,1;
- **Treinamento() –** Efetua a atualização dos pesos e *bias* com base na função delta generalizada – equação (2.11);
- **TestaRede() –** Aplica um padrão à rede para fins de classificação;
- **Ativa() –** Para um dado padrão de entrada, esse procedimento calcula a saída de cada neurônio baseada na função sigmóide – equação (2.4);
- **GravaParametros() –** Após o treinamento da rede, este procedimento grava os arquivos compostos pelos parâmetros do estado final da rede;

- **LeParametros() –** Realiza a leitura dos parâmetros dos arquivos gerados pela função GravaParametros;

3.3.1 – Principais Funções Utilizadas no Algoritmo (Pseudocódigo)

A seguir têm-se os fluxogramas das principais funções utilizadas no algoritmo: [Meyer, 1997]

Função Principal():

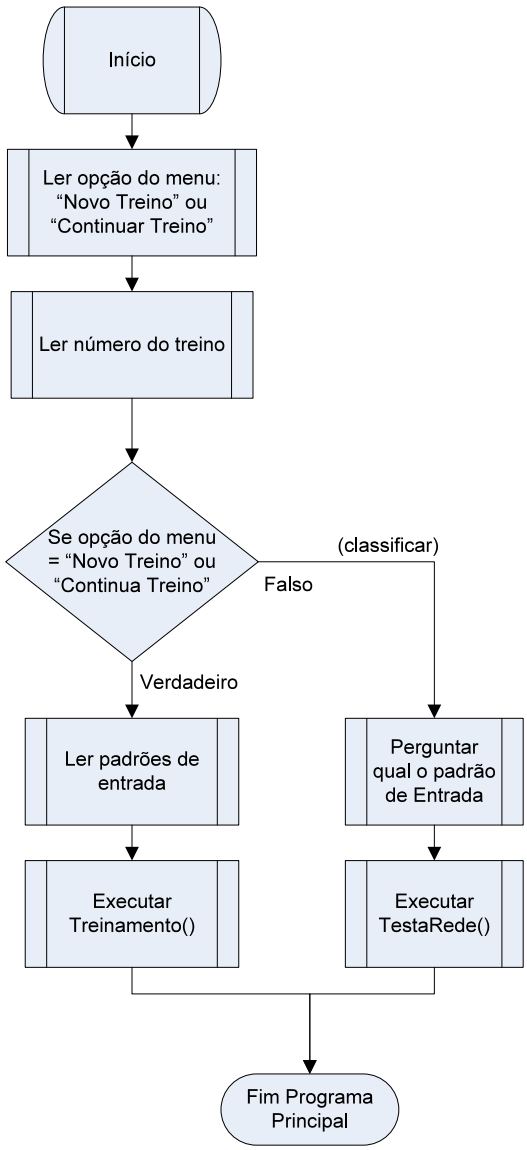

Figura 3.11 – Fluxograma da função principal.

Função Ativa():

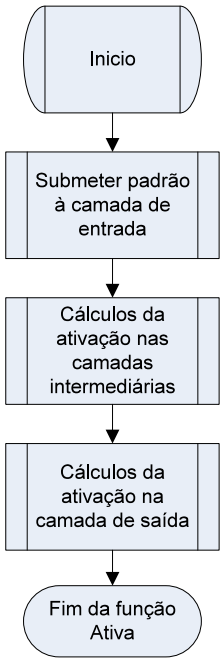

Figura 3.12 – Fluxograma da função ativa.

Função TestaRede():

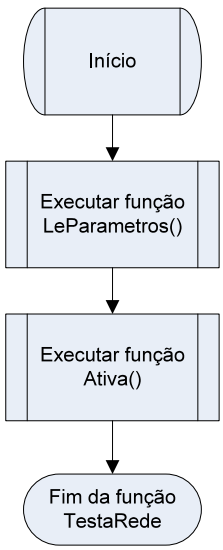

Figura 3.13 – Fluxograma da função testa rede.

Função Treinamento():

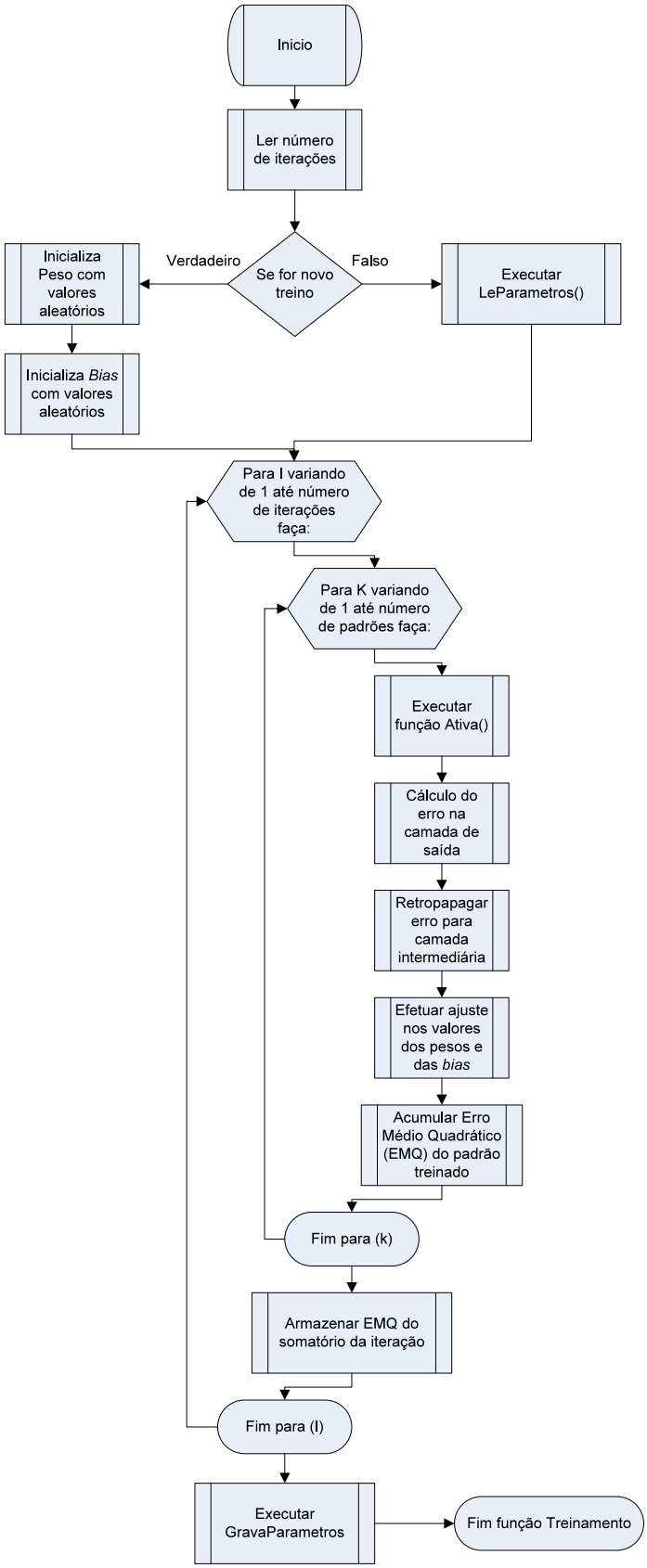

Figura 3.14 – Fluxograma da função treinamento.

3.3.2 – Treinamento

Durante a fase de treinamento, foram apresentados N padrões representando M ângulos diferentes e ajustados os parâmetros como a taxa de aprendizagem, *momentum* e tamanho da camada intermediária.

Para realização do treinamento, são requeridos os seguintes parâmetros:

- Número máximo de iterações;
- Amostras a serem treinadas;
- Número do treino.

A rede neural utilizada no projeto é do tipo supervisionada, utilizando a função de ativação sigmóide. O aprendizado é do tipo supervisionado e o seu treinamento do tipo estocástico, ou seqüencial.

No treinamento, usou-se uma matriz 35x9 (Tabela 3.2) onde as linhas representam quantidade de amostras utilizadas no treinamento e as colunas são cada *bit* que a rede deve atribuir um peso, sendo que a última coluna representa o resultado desejado para saída.

Os dados de entrada para o treinamento foram escolhidos considerando que, essas amostras são possíveis entradas que a rede obterá. Assim, com esses exemplos, foi realizado o treinamento da rede com objetivo de gerar uma matriz de peso para que, quando fossem realizados os testes, com novos dados inseridos na rede, esses dados pudessem ser influenciados pelos oriundos do treinamento e assim gerar os resultados de acordo com o treinamento realizado.

Tabela 3.2 – Dados para treinamento da rede neural.

Padrões de entrada para treinamento								Saída Desejada
0	1	1	1	1	0	0	1	0°
0	1	1	1	1	0	0	1	0°
0	1	1	1	0	0	1	1	0°
0	1	1	1	0	0	1	1	0°
0	1	1	1	1	1	1	0	0°
0	1	0	1	0	1	1	1	15°
0	1	0	1	1	1	0	1	15°
0	1	0	1	1	1	1	0	15°
0	1	0	1	0	0	1	1	15°
0	1	1	0	1	0	1	0	15°
0	1	0	0	1	0	1	0	30°
0	1	0	0	1	1	1	0	30°
0	1	0	0	1	1	0	0	30°
0	1	0	0	1	0	1	0	30°
0	1	0	0	1	1	0	0	30°
0	1	0	0	0	0	1	1	45°
0	1	0	0	1	0	1	1	45°
0	1	0	0	1	0	0	1	45°
0	1	0	0	0	1	0	0	45°
0	1	0	0	0	1	1	1	45°
0	0	1	1	1	1	1	1	60°
0	0	1	1	1	1	1	1	60°
0	0	1	1	1	1	1	0	60°
0	1	0	0	0	0	0	0	60°
0	0	1	1	1	1	1	1	60°
0	0	1	0	1	0	0	0	75°
0	0	1	0	0	1	1	0	75°
0	0	1	0	0	0	1	0	75°
0	0	1	0	0	1	1	1	75°
0	0	1	0	0	1	1	0	75°
0	0	0	1	0	0	1	0	90°
0	0	0	1	0	1	1	0	90°
0	0	0	1	0	1	1	1	90°
0	0	0	1	0	1	1	0	90°
0	0	0	1	0	1	1	1	90°

Na tabela 3.2, encontra-se os dados referentes a gravação realizada na terceira linha do tabuleiro, a 30 cm do microfone.

Para cada treinamento, é possível alterar os padrões de entrada, permitindo um novo tipo de generalização e uma nova matriz de peso. Com a alteração dos padrões de entrada entre um treinamento e outro, têm-se uma diferença entre os pesos da matriz e o erro médio quadrático, que irá variar dependendo do número de iterações que a rede sofrerá.

Durante a fase de treinamento, uma das dificuldades encontradas foi determinar a quantidade de iterações para realização do treinamento, pois como foi descrito no capítulo 2, o treinamento da rede influência no seu desempenho para fornecer as respostas corretas. Dessa forma, caso a rede seja “pouco” treinada, não irá apresentar bons resultados, mas caso a rede venha a ser “muito” treinada, a rede tende a “memorizar” os resultados ao invés de “aprender”. Assim, foram realizados diversos testes com a intenção de chegar-se a uma quantidade de iterações que fornecesse um bom resultado quando for realizado o teste da rede com intuito de verificar qual a direção da fonte de áudio.

A quantidade de épocas realizadas no treinamento, variou entre 1.000 e 100.000 iterações. Esses valores foram obtidos após uma série de testes realizados.

Para realização do treinamento, utilizou-se, aproximadamente, 52% das amostras para treinamento e o restante para validação.

Verificou-se, que para alguns casos, essa quantidade de amostras para o treinamento foi satisfatória, mas, como sugestão para outros trabalhos, sugere-se a utilização de quantidades diferentes de amostras para verificar o comportamento da rede.

Após uma série de treinamentos e testes, optou-se por parar o treinamento quando chega-se a 100.000 épocas, pois com uma faixa para treinamento entre 1.000 e 100.000 épocas, foi possível verificar o comportamento da rede de acordo com o número de épocas realizadas durante o treinamento.

Com a realização do treinamento e testes, verificou-se que em alguns ângulos a rede apresentava um melhor desempenho. Deve-se levar em conta que os ruídos externos existentes nas gravações estão influenciando no treinamento da rede neural, pois o tratamento realizado teve como objetivo “preparar” os dados para serem treinados e testados pela aplicação, não sendo realizado nenhum tipo de filtragem nos sinais. Dessa forma, possíveis erros e divergências relacionados a determinação angular ocorrem devido aos problemas encontrados na gravação dos sinais.

3.3.3 – Classificação de Novos Padrões

Para classificar novos padrões são informados para rede quais são esses novos dados a serem classificados. A rede realiza a leitura dos parâmetros gravados, como por exemplo, do peso associado ao treino realizado, e executa a função ativa (Figura 3.12) para determinar em qual direção localiza-se a fonte de áudio.

Essa direção será mostrada ao usuário em forma de ângulo aproximado. Assim, para cada novo padrão que a rede tenha que classificar, será informado ao usuário o ângulo aproximado de onde foi gerado o sinal de áudio.

Considerando o ruído ambiente e o erro aceitável da rede neural, esses valores podem variar em uma faixa de acordo com os gráficos da função *norma 2* mostrados no Capitulo 2.

Deve-se ressaltar, que o erro existente na determinação acústica, na prática pode não influenciar de uma forma que venha a prejudicar o uso da aplicação, pois como já foi comentado anteriormente, tem-se como possibilidade a aplicação desse projeto em sistemas de vídeo-conferência e sistemas de segurança, e em ambos os sistemas, podem-se fazer uso de câmeras filmadoras, e ao ângulo de visão de grande parte das câmeras está em torno de 54°. Assim, mesmo que existindo uma divergência entre o ângulo testado e o resultado apresentado, essa diferença pode ser compensada pelo ângulo de visão da câmera filmadora. [Souza, 2002]

3.4 – Configuração do Ambiente de Homologação

Para implementação do protótipo e realização de testes e simulações, necessitou-se dos seguintes equipamentos e softwares:

- Tabuleiro (Figura 3.1) divido em quatro linhas distantes a 10 cm, 20 cm, 30 cm e 40 cm do microfone;
- Microfone (Figura 3.4);
- Aparelho celular Nokia 1100 como fonte de áudio (Figura 3.5);
- Microcomputador com a seguinte configuração:
    - Pentium III 750 Mhz;
    - 384 MB de memória RAM;
    - 50 GB de disco rígido;
- Softwares utilizados:
    - Sistema Operacional Microsoft Windows XP;
    - Cool Edit versão 1.53;
    - Matlab 7.0;
    - Bloco de notas;
    - Borland C/C++ Builder 5.

3.5 – Métodos Utilizados

Após a gravação, tratamento, e treinamento dos dados, foi realizada uma simulação para determinar de qual direção o sinal foi gerado. Para tal objetivo, utiliza-se a aplicação criada em C/C++ descrita no item 3.3.

O primeiro passo para realização da simulação consiste em escolher em qual linha será comparado os ângulos. Caso ainda não tenha sido realizado o treinamento da rede, deve-se escolher a quantidade de interações que se deseja para treinar a rede. Após o treinamento, podem-se realizar os testes escolhendo o ângulo a ser verificado e conferir o resultado apresentado.

O item 3.6 mostra os passos para realizar a simulação que determina a direção da fonte de áudio.

3.6 – Passos para Realização do Treinamento e Testes para Determinação da Fonte de Áudio

A seguir têm-se os passos para realizar o treinamento e testes da rede neural utilizando a aplicação criada. São apresentadas as telas, suas funcionalidades e os procedimentos para desempenhar a simulação.

Figura 3.15 – Tela de apresentação da aplicação.

A Figura 3.15 mostra a **Tela de Apresentação** do programa criado usando a linguagem de programação C. A Figura 3.16, exibi a **Tela Principal**.

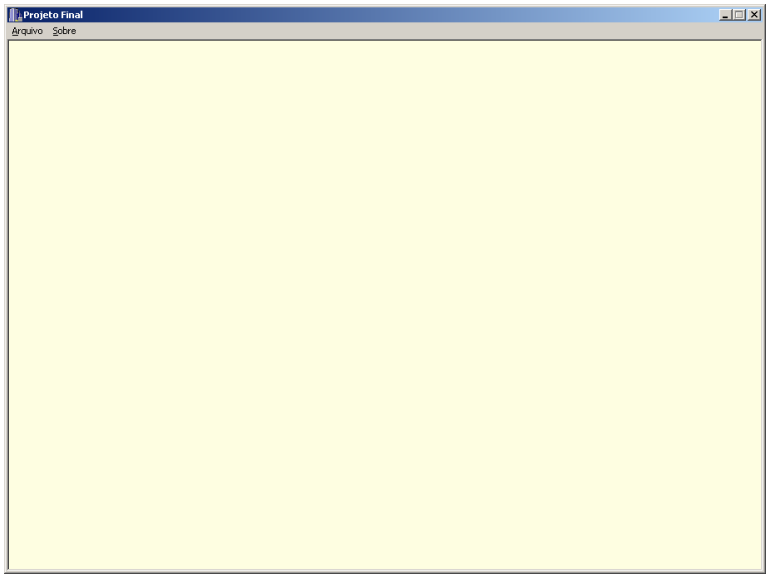

Figura 3.16 – Tela principal.

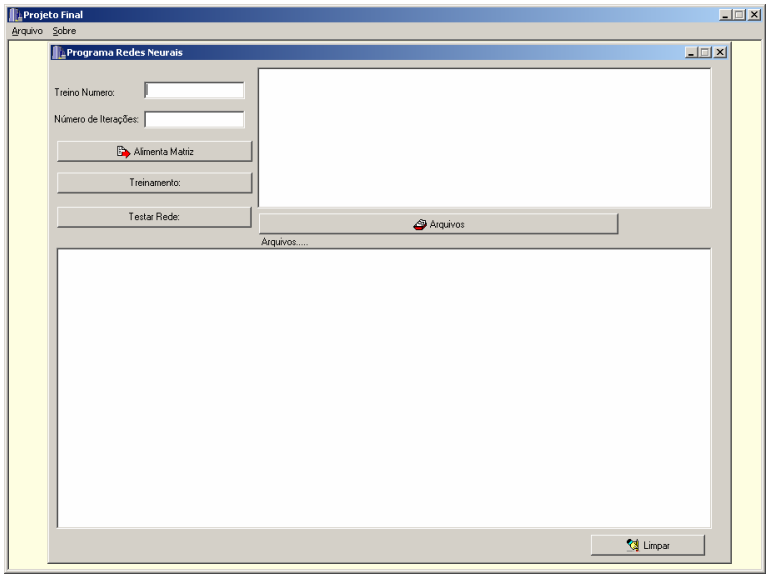

Figura 3.17 – Tela de simulação.

Clicando em **Arquivo**, Figura 3.16, e selecionando a opção **Simulação**, será mostrada a **Tela de Simulação**, Figura 3.17, onde poderá ser escolhido se deseja realizar um novo treinamento ou realizar os testes para verificar de qual direção está localizada a fonte de áudio.

Para realizar um novo treinamento, primeiramente devem-se escolher as dados que serão treinados. Tal procedimento é realizado clicando no botão **Arquivos**, Figura 3.17,

escolhendo a linha a ser treinada, Figura 3.18, e selecionando o primeiro arquivo de extensão *.txt* que aparece na Figura 3.19.

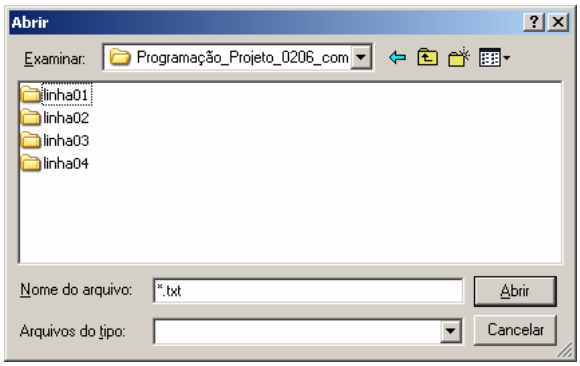

Figura 3.18 – Escolhendo a linha a ser treinada.

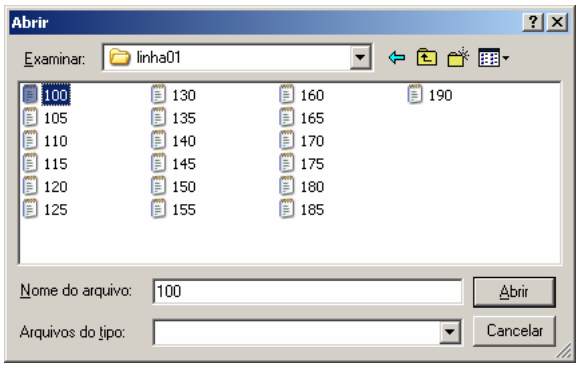

Figura 3.19 – Selecionando os dados para serem treinados.

Em seguida, deve-se indicar o ângulo que deverá ser treinado. No primeiro quadro da Figura 3.20, escolhe-se qual ângulo irá ser treinado. Em cada ângulo existem cinco amostras do sinal. Dessa forma, quando se escolhe o ângulo no primeiro quadro, no segundo quadro aparece cada uma das amostras que poderá ser utilizada para treinamento.

Nota-se que as amostras estão escritas da seguinte forma:

11111001: Amostra 0 da Linha – 1 do Ang. de 00 Graus

que quer dizer que essa amostra contém os dados do som gravado na linha 01 no ângulo 0°, e essa é a amostra da primeira gravação realizada nesse ângulo, e os primeiros números mostra o dado na forma de número binário.

Como foi dito no capítulo 2, foram gravados cinco sinais em cada ângulo, dessa forma, têm-se cinco amostras para cada ângulo, enumeradas de 0 a 4.

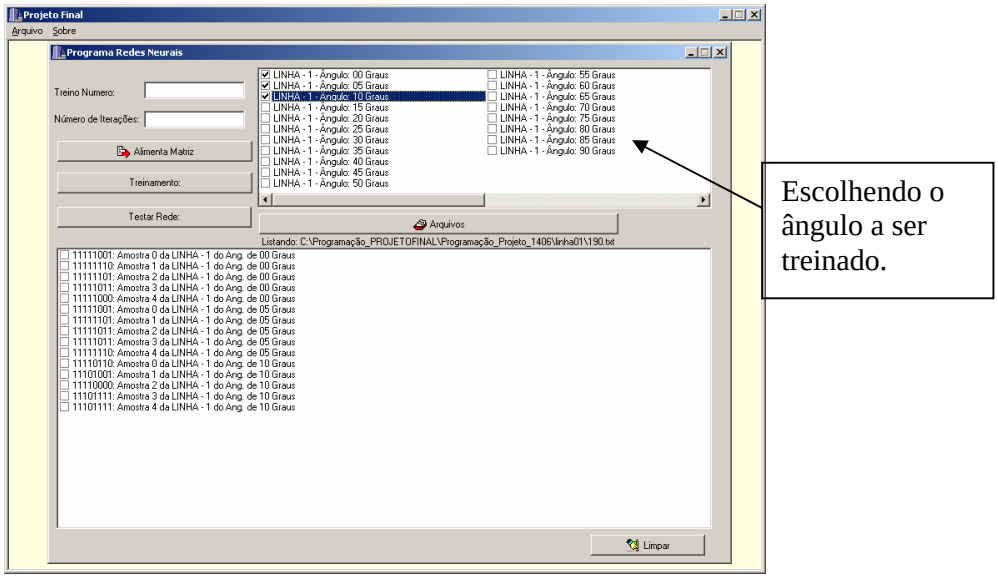

Figura 3.20 – Escolhendo ângulos a serem treinados.

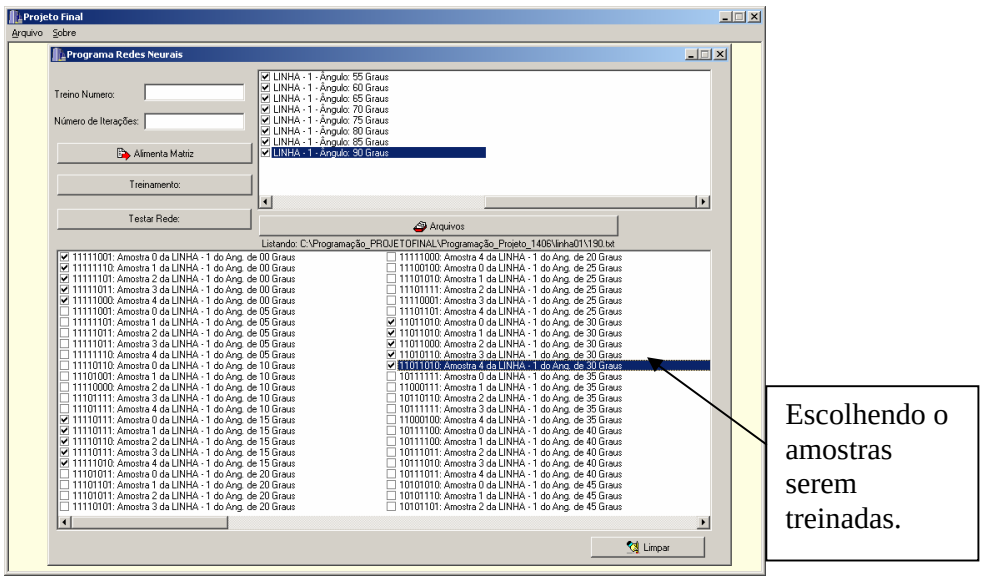

Figura 3.21 – Escolhendo amostras dos ângulos a serem treinadas.

Após a escolha dos ângulos e amostras a serem treinadas, deve-se alimentar a matriz de teste clicando no botão **Alimenta Matriz**, Figura 3.21, para que o treinamento possa ser realizado.

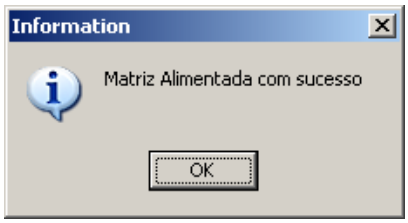

Figura 3.22 – Informando que a matriz de teste foi alimentada com sucesso.

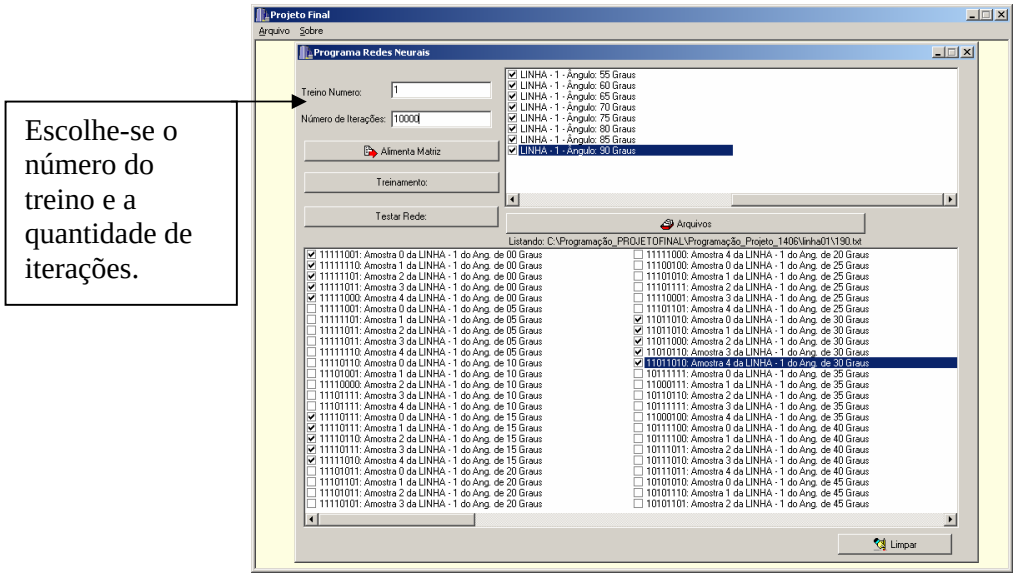

Figura 3.23 – Escolhendo o número do treino e a quantidade de iterações (épocas).

Depois de escolhida as amostras a serem treinadas, a alimentada a matriz, é preciso informar o número do treino a ser realizado e a quantidade de iterações que deverá existir no treinamento.

Informado o número do treino e a quantidade de iterações, basta clicar no botão **Treinamento** para que seja realizado o treinamento da rede.

Caso deseja-se realizar outro treinamento com as mesmas amostras, mas com a quantidade de iterações diferentes, basta informar o número do novo treino e a quantidade de iterações.

Caso almeje realizar o treinamento dos dados de outra linha, faz-se necessário excluir os dados da tela de simulação, para realizar essa ação basta clicar no botão **Limpar**, e seguir os passos a partir da Figura 3.17.

Para realizar o teste de verificação de qual ângulo o sinal de áudio foi emitido, devem-se escolher as amostras que serão utilizadas clicando no botão **Arquivos** da **Tela de Simulação**, selecionar a linha que deseja testar e escolher o primeiro arquivo, mostrado na Figura 3.19.

Após, deve-se selecionar a amostra a ser testada, informar o número do treino e clicar no botão **Testar Rede**, Figura 3.24.

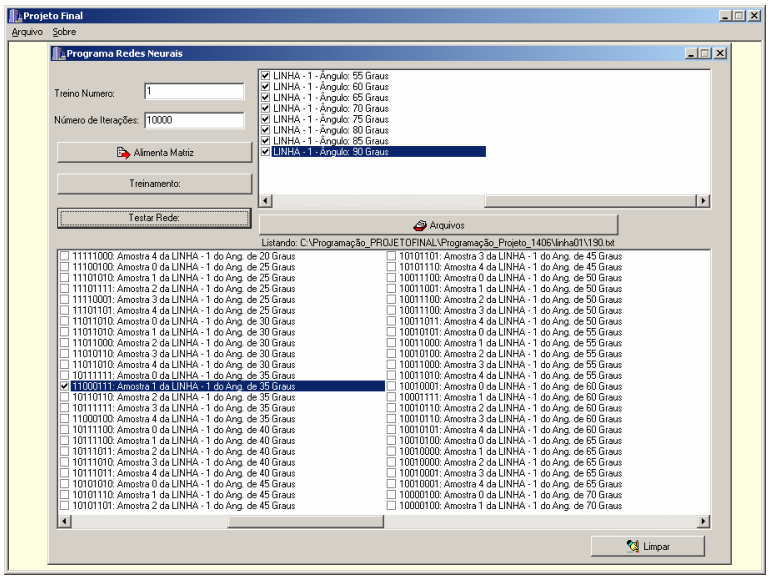

Figura 3.24 – Escolhendo amostra para avaliar.

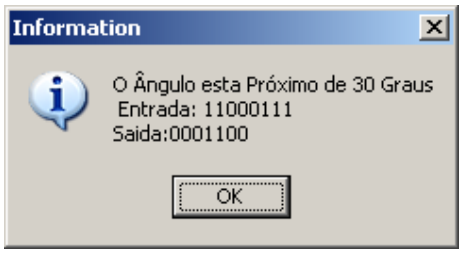

Figura 3.25 – Resultado da simulação.

Dessa forma, será informado qual o ângulo aproximado o sinal foi gerado, Figura 3.25. Também é informado no resultado final, a entrada em forma de número binário e a saída padrão da rede neural.

3.7 – Resultados e Considerações

Através dos testes realizados, nota-se que existiram algumas divergências entre o ângulo que foi submetido para teste e a saída esperada. Nota-se também que a quantidade de iterações do treinamento influência em boa parte nessa determinação.

Contudo, verifica-se que as prováveis causas para os erros na determinação tenham sido ocasionadas por ruídos externos ocorridos durante a gravação dos sinais, pois como foi dito anteriormente, os sinais gravados não sofreram nenhum tipo de filtragem.

Dessa forma, os resultados apresentados pela aplicação podem apresentar algumas divergências em relação ao ângulo solicitado para teste. Mas como foi mencionado, na prática essa diferença pode ser compensada pelo ângulo de visão de uma filmadora, que está em torno de 54°. [Souza, 2002]

Portanto, admite-se uma divergência entre o ângulo do sinal submetido a teste e o ângulo apresentado pela aplicação de no máximo 45°. Se essa diferença estiver nessa faixa, considera-se que a rede está apresentando, aproximadamente, a direção angular correta.

4 – Conclusão

4.1 – Considerações Finais

Através da realização deste projeto, foi possível determinar, aproximadamente, a direção de uma fonte de áudio em um sistema acústico.

Para que fosse possível a determinação da direção angular da fonte de áudio, fez-se necessário o tratamento dos sinais gravados. Para isso, foi calculada a *norma 2* do sinal discretizado (vetor), que oferece como resultado o comprimento do vetor, possibilitando extrair uma característica para cada sinal de áudio.

O uso de Redes Neurais Artificiais possibilitou uma generalização da determinação da fonte de áudio. O algoritmo, depois de treinado, tem a capacidade de responder por situações que jamais teve conhecimento. E, através do algoritmo *back-propagation*, chega-se, em uma boa parte das amostras a um erro aceitável, na determinação da fonte de áudio, permitindo definir, aproximadamente, de qual ângulo o sinal foi gerado.

O treinamento da rede teve como objetivo fazer com que a mesma "aprendesse" a distinguir novas situações não treinadas, ou seja, é passado para rede informações de uma nova localização da fonte de áudio, e a rede tem a capacidade de informar a localização aproximada dessa fonte.

Dessa forma, o uso de redes neurais possibilitou uma generalização das informações, permitindo que a mesma seja capaz de definir a posição de uma fonte de áudio para novos locais aonde não ocorreu o treinamento.

Portanto, com uma característica de cada sinal, fornecido pelo uso da *norma 2*, e o uso da aplicação desenvolvida baseada em redes neurais, foi possível determinar, aproximadamente, de qual direção angular foi gravado o sinal de áudio.

4.2 – Dificuldades Encontradas

Dentre as principais dificuldades encontradas para elaboração do projeto, destacam-se:

1ª - criação do algoritmo *back-propagation* em C/C++;

2ª - gravação e tratamento dos sinais de áudio.

A criação do algoritmo *back-propagation* em C/C++ tem como uma particularidade, o uso de números binários para entrada dos dados.

A gravação dos sinais foi a parte mais complicada do projeto, já que durante a gravação dos sinais, existiram diversos problemas, como: sensibilidade do microfone e ruído do ambiente.

Uma característica observada durante a gravação dos sinais de áudio trata-se da sensibilidade do microfone de acordo com o ângulo que o sinal foi gerado. Notou-se que quanto maior o ângulo, menor a sensibilidade do microfone usado na gravação dos sinais.

Deve-se ressaltar que existiram algumas alterações no decorrer da elaboração do projeto. Dentre as alterações realizadas, a que mais se destaca é a troca do circuito eletrônico que tinha como objetivo capturar, amplificar, converter o sinal de áudio e enviar o dado para o PC por meio da porta paralela. Tal circuito foi substituído e optou-se pelo uso da placa de som do PC, e os softwares Cool Edit e Matlab, para gravação e tratamento dos sinais, respectivamente. Detalhes sobre circuito projetado encontram-se no Anexo B.

4.3 – Sugestões para trabalhos futuros

Através dos conceitos apresentados neste projeto, pode-se ter como inspiração para o futuro:

- a construção de um protótipo sem utilização do PC, realizando a programação utilizando um Microcontrolador, deixando o protótipo independente de um microcomputador;
- o movimento de uma base giratória, que poderá ter como objetivo focalizar a fonte de áudio, podendo ser utilizado em sistemas de vídeo-conferência e sistemas de segurança;
- uso da voz humana como fonte de áudio;
- utilização de outro método para determinar a direção da fonte acústica, como por exemplo, lógica de fuzzy ou interpolação;
- gravação dos sinais de áudio utilizando outra fonte de áudio e em outro ambiente, como por exemplo, um estúdio de gravação.

5 – Referências Bibliográficas

ALLDATASHEET, **AllDataSheet**. Disponível em <http://www.alldatasheet.com>. Acessado em 10 de novembro de 2004.

ALMEIDA, **Almeida Home page**. Disponível em <http://planeta.terra.com.br/educacao/almeida/pg09.htm>. Acessado em 24 de novembro de 2004.

BOGART, Theodore F. Jr.. **Dispositivos e Circuitos Eletrônicos –** Volume II. 3ª edição. . São Paulo: Pearson Makron Books, 2001.

BOLDRINI, José Luiz; COSTA, Sueli I. Rodrigues; FIGUEIREDO, Vera Lúcia; WETZLER, Henry G.. **Algebra Linear –** 3ª Edição **–** São Paulo: Editora Harbra Ltda, 1981.

BOYLESTAD, Robert L.; NASHELSKY, Louis. **Dispositivos Eletrônicos e Teoria de Circuitos –** Sexta Edição **–** Rio de Janeiro: LTC Editora, 1999.

BRAGA, Antônio de Pádua; CARVALHO André Carlos P. de Leon F; LUDERNIR, Teresa Bernarda. **Redes Neurais Artificiais –** Teoria e Aplicações. LTC, 2000

CHAPMAN, Stephen J. **Programação em MATLAB para Engenheiros –** São Paulo: Pioneira Thomson Learning, 2003.

GUAHYBA; Adriano. **Visão Geral sobre Redes Neurais**. Disponível em <http://www.guahyba.vet.br/avicultura/neurais.htm>. Acessado em 17 de agosto de 2004.

HAYKIN, Simon. **Redes Neurais: Princípios e Prática**. Trad. Paulo Martins Engel. **–** 2ª edição **–** Porto Alegre: Bookman, 2001

INGLE, Vinay K; PROAKIS, John G. **Digital Signal Processing using MATLAB**. **–** BookWare Companio Series, 2000

LIPORACE, Frederico dos Santos; MACHADO, Ricardo José; BARBOSA Valmir Carneiro. **Um Sistema Neural para Monitoração do Desflorestamento na Região**

Amazônica Utilizando Imagens do Landsat. Centro Científico Rio – Relatório Técnico CCR-161 – IBM, 1994.

MATSUMOTO, Élia Yathie. **MATLAB 6**: Fundamentos de Programação – São Paulo: Érica, 2001

MATSUMOTO, Élia Yathie. **MATLAB 7**: Fundamentos de Programação – São Paulo: Érica, 2004

MESSIAS, Antônio Rogério. **Características de Funcionamento do Conversor Analógico Digital ADC0804 de 8 bits**. Disponível em <http://www.rogercom.com/pparalela/ConversorADC0804.htm> Acessado em 27 de setembro de 2004

MEYER, Wladimir da Silva Meyer. **Metodologias para Classificação de Texturas e Consulta a Bases de Imagens**. Instituto Militar de Engenharia – Rio de Janeiro, 1997.

NATIONAL. **National Semicondutor**. Disponível em <www.national.com>. Último acesso em 10 de novembro de 2004.

RUSSEL, Stuart; NORVIG, Peter. **Artificial Intelligence – A Modern Approach**. Pearson Education by Prentice-Hall, New Jersey – EUA.

SCHILDT, Herbet. **C, Completo e Total –** 3ª edição. São Paulo: Pearson Makron Books, 1997.

SIQUEIRA, José Antônio Lerosa. **Inteligência Artificial Usando Redes Neurais**. Disponível em <http://www.caetano.eng.br/pef411/index.htm>. Acessado em 30 de agosto de 2004

SOUZA, Adílio Ricardo M.. **A escolha adequada das câmeras.** Able Alarm Serviços Ltda. De 20 de agosto de 2002. Disponível em <http://www.ablealarm.com.br/artigo20082002.htm>. Acessado em 16 de junho de 2005.

STROUSTRUP, Bjarne. **A Linguagem de Programação C++ –** 3ª edição. Porto Alegre: Bookman, 2000.

TATIBANA, Cássia Yuri; KAETSU Deisi Yuki. **Uma Introdução às Redes Neurais**. Disponível em <http://www.din.uem.br/ia/neurais/>. Acessado em 17 de agosto de 2004 e 10 de novembro de 2004.

VELLASCO, Marley Maria B. R.. **Sistemas Inteligentes – Redes Neurais**. Disponível em: <www.ica.ele.puc-rio.br>. Acessado em 17 de agosto de 2004.

ZUBEN, Fernando J. Von. Uma **Caricatura Funcional de Redes Neurais Artificiais**. Revista da Sociedade Brasileira de Redes Neurais, Vol.: 1 Nº. 2, pp. – 77-87, 2003. Disponível em <http://www.sbrn.org.br>. Acessado em 27 de agosto de 2004.

Anexo A – Algoritmos para Tratamento de Sinais

A seguir têm-se os algoritmos utilizados para realizar o tratamento dos sinais gravados. O tratamento foi realizado em cada linha onde o sinal foi gravado.

Primeiramente faz-se a leitura dos sinais. Em seguida é calculado o comprimento do vetor utilizando a função *norm*. Após o cálculo do comprimento do vetor, esse dado é transformado em número binário através da função *dec2bin()*. Depois da conversão dos dados para binário, esses dados são salvos em um arquivo de extensão.*txt*.

Os algoritmos foram criados utilizando o software Matlab.

```
%*****************************************************************
%Centro Universitário de Brasília – UniCEUB
%Faculdade de Ciências Exatas e Tecnologia – FAET
%Curso de Engenharia da Computação
%Projeto Final
%Prof. Orientador: José Julimá
%Aluno: Mateus Silva Teixeira - RA.: 20016207
%*****************************************************************
%AQUISIÇÃO E TRATAMENTO DE DADOS EM UM SISTEMA DE ORIENTAÇÃO ACÚSTICA
%*****************************************************************
%Tratamento de dados - Linha 01
%*****************************************************************
%Legenda
%   S = som
%   R = read - sinal lido e armazenado em uma variável
%   N = norm - comprimento do vetor
%   B = binário
%   primeiro número (3) = microfone utilizado (3)
%   segundo número (1) = linha onde está a fonte de áudio (1)
%   terceiro e quarto número (00) = ângulo da fonte de áudio (00)
%   quinto número (1) = gravação
%       Obs.: foram realizadas 5 (cinco) gravações em cada ângulo
%*****************************************************************

%Realizando a leitura dos sinais
%ângulo 0
[R31001,fs]=wavread('S31001.wav');
[R31002,fs]=wavread('S31002.wav');
[R31003,fs]=wavread('S31003.wav');
[R31004,fs]=wavread('S31004.wav');
[R31005,fs]=wavread('S31005.wav');

%ângulo 05
[R31051,fs]=wavread('S31051.wav');
[R31052,fs]=wavread('S31052.wav');
[R31053,fs]=wavread('S31053.wav');
[R31054,fs]=wavread('S31054.wav');
[R31055,fs]=wavread('S31055.wav');

%ângulo 10
[R31101,fs]=wavread('S31101.wav');
[R31102,fs]=wavread('S31102.wav');
[R31103,fs]=wavread('S31103.wav');
[R31104,fs]=wavread('S31104.wav');
[R31105,fs]=wavread('S31105.wav');

%ângulo 15
[R31151,fs]=wavread('S31151.wav');
[R31152,fs]=wavread('S31152.wav');
```

```
[R31153,fs]=wavread('S31153.wav');
[R31154,fs]=wavread('S31154.wav');
[R31155,fs]=wavread('S31155.wav');

%ângulo 20
[R31201,fs]=wavread('S31201.wav');
[R31202,fs]=wavread('S31202.wav');
[R31203,fs]=wavread('S31203.wav');
[R31204,fs]=wavread('S31204.wav');
[R31205,fs]=wavread('S31205.wav');

%ângulo 25
[R31251,fs]=wavread('S31251.wav');
[R31252,fs]=wavread('S31252.wav');
[R31253,fs]=wavread('S31253.wav');
[R31254,fs]=wavread('S31254.wav');
[R31255,fs]=wavread('S31255.wav');

%ângulo 30
[R31301,fs]=wavread('S31301.wav');
[R31302,fs]=wavread('S31302.wav');
[R31303,fs]=wavread('S31303.wav');
[R31304,fs]=wavread('S31304.wav');
[R31305,fs]=wavread('S31305.wav');

%ângulo 35
[R31351,fs]=wavread('S31351.wav');
[R31352,fs]=wavread('S31352.wav');
[R31353,fs]=wavread('S31353.wav');
[R31354,fs]=wavread('S31354.wav');
[R31355,fs]=wavread('S31355.wav');

%ângulo 40
[R31401,fs]=wavread('S31401.wav');
[R31402,fs]=wavread('S31402.wav');
[R31403,fs]=wavread('S31403.wav');
[R31404,fs]=wavread('S31404.wav');
[R31405,fs]=wavread('S31405.wav');

%ângulo 45
[R31451,fs]=wavread('S31451.wav');
[R31452,fs]=wavread('S31452.wav');
[R31453,fs]=wavread('S31453.wav');
[R31454,fs]=wavread('S31454.wav');
[R31455,fs]=wavread('S31455.wav');

%ângulo 50
[R31501,fs]=wavread('S31501.wav');
[R31502,fs]=wavread('S31502.wav');
[R31503,fs]=wavread('S31503.wav');
[R31504,fs]=wavread('S31504.wav');
[R31505,fs]=wavread('S31505.wav');

%ângulo 55
[R31551,fs]=wavread('S31551.wav');
[R31552,fs]=wavread('S31552.wav');
[R31553,fs]=wavread('S31553.wav');
[R31554,fs]=wavread('S31554.wav');
[R31555,fs]=wavread('S31555.wav');

%ângulo 60
[R31601,fs]=wavread('S31601.wav');
[R31602,fs]=wavread('S31602.wav');
[R31603,fs]=wavread('S31603.wav');
[R31604,fs]=wavread('S31604.wav');
[R31605,fs]=wavread('S31605.wav');

%ângulo 65
[R31651,fs]=wavread('S31651.wav');
[R31652,fs]=wavread('S31652.wav');
[R31653,fs]=wavread('S31653.wav');
```

```
[R31654,fs]=wavread('S31654.wav');
[R31655,fs]=wavread('S31655.wav');

%ângulo 70
[R31701,fs]=wavread('S31701.wav');
[R31702,fs]=wavread('S31702.wav');
[R31703,fs]=wavread('S31703.wav');
[R31704,fs]=wavread('S31704.wav');
[R31705,fs]=wavread('S31705.wav');

%ângulo 75
[R31751,fs]=wavread('S31751.wav');
[R31752,fs]=wavread('S31752.wav');
[R31753,fs]=wavread('S31753.wav');
[R31754,fs]=wavread('S31754.wav');
[R31755,fs]=wavread('S31755.wav');

%ângulo 80
[R31801,fs]=wavread('S31801.wav');
[R31802,fs]=wavread('S31802.wav');
[R31803,fs]=wavread('S31803.wav');
[R31804,fs]=wavread('S31804.wav');
[R31805,fs]=wavread('S31805.wav');

%ângulo 85
[R31851,fs]=wavread('S31851.wav');
[R31852,fs]=wavread('S31852.wav');
[R31853,fs]=wavread('S31853.wav');
[R31854,fs]=wavread('S31854.wav');
[R31855,fs]=wavread('S31855.wav');

%ângulo 90
[R31901,fs]=wavread('S31901.wav');
[R31902,fs]=wavread('S31902.wav');
[R31903,fs]=wavread('S31903.wav');
[R31904,fs]=wavread('S31904.wav');
[R31905,fs]=wavread('S31905.wav');

%Fazendo a norm de cada sinal
%norm ang. 0
N31001=norm(R31001)*50
N31002=norm(R31002)*50
N31003=norm(R31003)*50
N31004=norm(R31004)*50
N31005=norm(R31005)*50

%norm ang. 05
N31051=norm(R31051)*50
N31052=norm(R31052)*50
N31053=norm(R31053)*50
N31054=norm(R31054)*50
N31055=norm(R31055)*50

%norm ang. 10
N31101=norm(R31101)*50
N31102=norm(R31102)*50
N31103=norm(R31103)*50
N31104=norm(R31104)*50
N31105=norm(R31105)*50

%norm ang. 15
N31151=norm(R31151)*50
N31152=norm(R31152)*50
N31153=norm(R31153)*50
N31154=norm(R31154)*50
N31155=norm(R31155)*50

%norm ang. 20
N31201=norm(R31201)*50
N31202=norm(R31202)*50
N31203=norm(R31203)*50
```

```
N31204=norm(R31204)*50
N31205=norm(R31205)*50

%norm ang. 25
N31251=norm(R31251)*50
N31252=norm(R31252)*50
N31253=norm(R31253)*50
N31254=norm(R31254)*50
N31255=norm(R31255)*50

%norm ang. 30
N31301=norm(R31301)*50
N31302=norm(R31302)*50
N31303=norm(R31303)*50
N31304=norm(R31304)*50
N31305=norm(R31305)*50

%norm ang. 35
N31351=norm(R31351)*50
N31352=norm(R31352)*50
N31353=norm(R31353)*50
N31354=norm(R31354)*50
N31355=norm(R31355)*50

%norm ang. 40
N31401=norm(R31401)*50
N31402=norm(R31402)*50
N31403=norm(R31403)*50
N31404=norm(R31404)*50
N31405=norm(R31405)*50

%norm ang. 45
N31451=norm(R31451)*50
N31452=norm(R31452)*50
N31453=norm(R31453)*50
N31454=norm(R31454)*50
N31455=norm(R31455)*50

%norm ang. 50
N31501=norm(R31501)*50
N31502=norm(R31502)*50
N31503=norm(R31503)*50
N31504=norm(R31504)*50
N31505=norm(R31505)*50

%norm ang. 55
N31551=norm(R31551)*50
N31552=norm(R31552)*50
N31553=norm(R31553)*50
N31554=norm(R31554)*50
N31555=norm(R31555)*50

%norm ang. 60
N31601=norm(R31601)*50
N31602=norm(R31602)*50
N31603=norm(R31603)*50
N31604=norm(R31604)*50
N31605=norm(R31605)*50

%norm ang. 65
N31651=norm(R31651)*50
N31652=norm(R31652)*50
N31653=norm(R31653)*50
N31654=norm(R31654)*50
N31655=norm(R31655)*50

%norm ang. 70
N31701=norm(R31701)*50
N31702=norm(R31702)*50
N31703=norm(R31703)*50
N31704=norm(R31704)*50
```

```
N31705=norm(R31705)*50

%norm ang. 75
N31751=norm(R31751)*50
N31752=norm(R31752)*50
N31753=norm(R31753)*50
N31754=norm(R31754)*50
N31755=norm(R31755)*50

%norm ang. 80
N31801=norm(R31801)*50
N31802=norm(R31802)*50
N31803=norm(R31803)*50
N31804=norm(R31804)*50
N31805=norm(R31805)*50

%norm ang. 85
N31851=norm(R31851)*50
N31852=norm(R31852)*50
N31853=norm(R31853)*50
N31854=norm(R31854)*50
N31855=norm(R31855)*50

%norm ang. 90
N31901=norm(R31901)*50
N31902=norm(R31902)*50
N31903=norm(R31903)*50
N31904=norm(R31904)*50
N31905=norm(R31905)*50

%Convertendo em binário
%bin ang. 0
B31001=dec2bin(N31001,8)
B31002=dec2bin(N31002,8)
B31003=dec2bin(N31003,8)
B31004=dec2bin(N31004,8)
B31005=dec2bin(N31005,8)

%bin ang. 05
B31051=dec2bin(N31051,8)
B31052=dec2bin(N31052,8)
B31053=dec2bin(N31053,8)
B31054=dec2bin(N31054,8)
B31055=dec2bin(N31055,8)

%bin ang. 10
B31101=dec2bin(N31101,8)
B31102=dec2bin(N31102,8)
B31103=dec2bin(N31103,8)
B31104=dec2bin(N31104,8)
B31105=dec2bin(N31105,8)

%bin ang. 15
B31151=dec2bin(N31151,8)
B31152=dec2bin(N31152,8)
B31153=dec2bin(N31153,8)
B31154=dec2bin(N31154,8)
B31155=dec2bin(N31155,8)

%bin ang. 20
B31201=dec2bin(N31201,8)
B31202=dec2bin(N31202,8)
B31203=dec2bin(N31203,8)
B31204=dec2bin(N31204,8)
B31205=dec2bin(N31205,8)

%bin ang. 25
B31251=dec2bin(N31251,8)
B31252=dec2bin(N31252,8)
B31253=dec2bin(N31253,8)
B31254=dec2bin(N31254,8)
```

```
B31255=dec2bin(N31255,8)

%bin ang. 30
B31301=dec2bin(N31301,8)
B31302=dec2bin(N31302,8)
B31303=dec2bin(N31303,8)
B31304=dec2bin(N31304,8)
B31305=dec2bin(N31305,8)

%bin ang. 35
B31351=dec2bin(N31351,8)
B31352=dec2bin(N31352,8)
B31353=dec2bin(N31353,8)
B31354=dec2bin(N31354,8)
B31355=dec2bin(N31355,8)

%bin ang. 40
B31401=dec2bin(N31401,8)
B31402=dec2bin(N31402,8)
B31403=dec2bin(N31403,8)
B31404=dec2bin(N31404,8)
B31405=dec2bin(N31405,8)

%bin ang. 45
B31451=dec2bin(N31451,8)
B31452=dec2bin(N31452,8)
B31453=dec2bin(N31453,8)
B31454=dec2bin(N31454,8)
B31455=dec2bin(N31455,8)

%bin ang. 50
B31501=dec2bin(N31501,8)
B31502=dec2bin(N31502,8)
B31503=dec2bin(N31503,8)
B31504=dec2bin(N31504,8)
B31505=dec2bin(N31505,8)

%bin ang. 55
B31551=dec2bin(N31551,8)
B31552=dec2bin(N31552,8)
B31553=dec2bin(N31553,8)
B31554=dec2bin(N31554,8)
B31555=dec2bin(N31555,8)

%bin ang. 60
B31601=dec2bin(N31601,8)
B31602=dec2bin(N31602,8)
B31603=dec2bin(N31603,8)
B31604=dec2bin(N31604,8)
B31605=dec2bin(N31605,8)

%bin ang. 65
B31651=dec2bin(N31651,8)
B31652=dec2bin(N31652,8)
B31653=dec2bin(N31653,8)
B31654=dec2bin(N31654,8)
B31655=dec2bin(N31655,8)

%bin ang. 70
B31701=dec2bin(N31701,8)
B31702=dec2bin(N31702,8)
B31703=dec2bin(N31703,8)
B31704=dec2bin(N31704,8)
B31705=dec2bin(N31705,8)

%bin ang. 75
B31751=dec2bin(N31751,8)
B31752=dec2bin(N31752,8)
B31753=dec2bin(N31753,8)
B31754=dec2bin(N31754,8)
B31755=dec2bin(N31755,8)
```

```
%bin ang. 80
B31801=dec2bin(N31801,8)
B31802=dec2bin(N31802,8)
B31803=dec2bin(N31803,8)
B31804=dec2bin(N31804,8)
B31805=dec2bin(N31805,8)

%bin ang. 85
B31851=dec2bin(N31851,8)
B31852=dec2bin(N31852,8)
B31853=dec2bin(N31853,8)
B31854=dec2bin(N31854,8)
B31855=dec2bin(N31855,8)

%bin ang. 90
B31901=dec2bin(N31901,8)
B31902=dec2bin(N31902,8)
B31903=dec2bin(N31903,8)
B31904=dec2bin(N31904,8)
B31905=dec2bin(N31905,8)

%Plotando os gráficos do comprimento dos sinais
P31 = [N31001 N31002 N31003 N31004 N31005 N31051 N31052 N31053 N31054 N31055 N31101 N31102 N31103
N31104 N31105 N31151 N31152 N31153 N31154 N31205 N31201 N31202 N31203 N31204 N31205 N31251 N31252
N31253 N31254 N31255 N31301 N31302 N31303 N31304 N31305 N31351 N31352 N31353 N31354 N31355 N31401
N31402 N31403 N31404 N31405 N31451 N31452 N31453 N31454 N31455 N31501 N31502 N31503 N31504 N31505
N31551 N31552 N31553 N31554 N31555 N31601 N31602 N31603 N31604 N31605 N31651 N31652 N31653 N31654
N31655 N31701 N31702 N31703 N31704 N31705 N31751 N31752 N31753 N31754 N31755 N31801 N31802 N31803
N31804 N31805 N31851 N31852 N31853 N31854 N31855 N31901 N31902 N31903 N31904 N31905]
plot(P31),title('Norm dos sinais - Linha 01'),xlabel('Ângulo'),ylabel('Comprimento do Sinal - Norma 2')
grid on

%Salvando em .txt
%save ang. 0
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31001.txt','B31001','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31002.txt','B31002','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31003.txt','B31003','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31004.txt','B31004','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31005.txt','B31005','-ASCII');

%save ang. 05
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31051.txt','B31051','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31052.txt','B31052','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31053.txt','B31053','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31054.txt','B31054','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31055.txt','B31055','-ASCII');

%save ang. 10
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31101.txt','B31101','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31102.txt','B31102','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31103.txt','B31103','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31104.txt','B31104','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31105.txt','B31105','-ASCII');

%save ang. 15
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31151.txt','B31151','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31152.txt','B31152','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31153.txt','B31153','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31154.txt','B31154','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31155.txt','B31155','-ASCII');

%save ang. 20
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31201.txt','B31201','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31202.txt','B31202','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31203.txt','B31203','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31204.txt','B31204','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31205.txt','B31205','-ASCII');

%save ang. 25
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31251.txt','B31251','-ASCII');
```

```
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31252.txt','B31252','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31253.txt','B31253','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31254.txt','B31254','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31255.txt','B31255','-ASCII');

%save ang. 30
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31301.txt','B31301','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31302.txt','B31302','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31303.txt','B31303','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31304.txt','B31304','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31305.txt','B31305','-ASCII');

%save ang. 35
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31351.txt','B31351','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31352.txt','B31352','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31353.txt','B31353','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31354.txt','B31354','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31355.txt','B31355','-ASCII');

%save ang. 40
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31401.txt','B31401','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31402.txt','B31402','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31403.txt','B31403','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31404.txt','B31404','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31405.txt','B31405','-ASCII');

%save ang. 45
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31451.txt','B31451','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31452.txt','B31452','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31453.txt','B31453','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31454.txt','B31454','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31455.txt','B31455','-ASCII');

%save ang. 50
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31501.txt','B31501','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31502.txt','B31502','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31503.txt','B31503','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31504.txt','B31504','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31505.txt','B31505','-ASCII');

%save ang. 55
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31551.txt','B31551','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31552.txt','B31552','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31553.txt','B31553','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31554.txt','B31554','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31555.txt','B31555','-ASCII');

%save ang. 60
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31601.txt','B31601','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31602.txt','B31602','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31603.txt','B31603','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31604.txt','B31604','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31605.txt','B31605','-ASCII');

%save ang. 65
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31651.txt','B31651','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31652.txt','B31652','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31653.txt','B31653','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31654.txt','B31654','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31655.txt','B31655','-ASCII');

%save ang. 70
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31701.txt','B31701','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31702.txt','B31702','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31703.txt','B31703','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31704.txt','B31704','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31705.txt','B31705','-ASCII');

%save ang. 75
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31751.txt','B31751','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31752.txt','B31752','-ASCII');
```

```
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31753.txt','B31753','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31754.txt','B31754','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31755.txt','B31755','-ASCII');

%save ang. 80
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31801.txt','B31801','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31802.txt','B31802','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31803.txt','B31803','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31804.txt','B31804','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31805.txt','B31805','-ASCII');

%save ang. 85
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31851.txt','B31851','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31852.txt','B31852','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31853.txt','B31853','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31854.txt','B31854','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31855.txt','B31855','-ASCII');

%save ang. 90
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31901.txt','B31901','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31902.txt','B31902','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31903.txt','B31903','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31904.txt','B31904','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T31905.txt','B31905','-ASCII');


%****************************************************************
%Centro Universitário de Brasília – UniCEUB
%Faculdade de Ciências Exatas e Tecnologia – FAET
%Curso de Engenharia da Computação
%Projeto Final
%Prof. Orientador: José Julimá
%Aluno: Mateus Silva Teixeira - RA.: 20016207
%****************************************************************
%AQUISIÇÃO E TRATAMENTO DE DADOS EM UM SISTEMA DE ORIENTAÇÃO ACÚSTICA
%****************************************************************
%Tratamento de dados - Linha 02
%****************************************************************
%Legenda
%   S = som
%   R = read - sinal lido e armazenado em uma variável
%   N = norm - comprimento do vetor
%   B = binário
%   primeiro número (3) = microfone utilizado (3)
%   segundo número (2) = linha onde está a fonte de áudio (2)
%   terceiro e quarto número (00) = ângulo da fonte de áudio (00)
%   quinto número (1) = gravação
%       Obs.: foram realizadas 5 (cinco) gravações em cada ângulo
%****************************************************************

%Realizando a leitura dos sinais
%ângulo 0
[R32001,fs]=wavread('S32001.wav');
[R32002,fs]=wavread('S32002.wav');
[R32003,fs]=wavread('S32003.wav');
[R32004,fs]=wavread('S32004.wav');
[R32005,fs]=wavread('S32005.wav');

%ângulo 05
[R32051,fs]=wavread('S32051.wav');
[R32052,fs]=wavread('S32052.wav');
[R32053,fs]=wavread('S32053.wav');
[R32054,fs]=wavread('S32054.wav');
[R32055,fs]=wavread('S32055.wav');

%ângulo 10
[R32101,fs]=wavread('S32101.wav');
[R32102,fs]=wavread('S32102.wav');
[R32103,fs]=wavread('S32103.wav');
[R32104,fs]=wavread('S32104.wav');
[R32105,fs]=wavread('S32105.wav');
```

```
%ângulo 15
[R32151,fs]=wavread('S32151.wav');
[R32152,fs]=wavread('S32152.wav');
[R32153,fs]=wavread('S32153.wav');
[R32154,fs]=wavread('S32154.wav');
[R32155,fs]=wavread('S32155.wav');

%ângulo 20
[R32201,fs]=wavread('S32201.wav');
[R32202,fs]=wavread('S32202.wav');
[R32203,fs]=wavread('S32203.wav');
[R32204,fs]=wavread('S32204.wav');
[R32205,fs]=wavread('S32205.wav');

%ângulo 25
[R32251,fs]=wavread('S32251.wav');
[R32252,fs]=wavread('S32252.wav');
[R32253,fs]=wavread('S32253.wav');
[R32254,fs]=wavread('S32254.wav');
[R32255,fs]=wavread('S32255.wav');

%ângulo 30
[R32301,fs]=wavread('S32301.wav');
[R32302,fs]=wavread('S32302.wav');
[R32303,fs]=wavread('S32303.wav');
[R32304,fs]=wavread('S32304.wav');
[R32305,fs]=wavread('S32305.wav');

%ângulo 35
[R32351,fs]=wavread('S32351.wav');
[R32352,fs]=wavread('S32352.wav');
[R32353,fs]=wavread('S32353.wav');
[R32354,fs]=wavread('S32354.wav');
[R32355,fs]=wavread('S32355.wav');

%ângulo 40
[R32401,fs]=wavread('S32401.wav');
[R32402,fs]=wavread('S32402.wav');
[R32403,fs]=wavread('S32403.wav');
[R32404,fs]=wavread('S32404.wav');
[R32405,fs]=wavread('S32405.wav');

%ângulo 45
[R32451,fs]=wavread('S32451.wav');
[R32452,fs]=wavread('S32452.wav');
[R32453,fs]=wavread('S32453.wav');
[R32454,fs]=wavread('S32454.wav');
[R32455,fs]=wavread('S32455.wav');

%ângulo 50
[R32501,fs]=wavread('S32501.wav');
[R32502,fs]=wavread('S32502.wav');
[R32503,fs]=wavread('S32503.wav');
[R32504,fs]=wavread('S32504.wav');
[R32505,fs]=wavread('S32505.wav');

%ângulo 55
[R32551,fs]=wavread('S32551.wav');
[R32552,fs]=wavread('S32552.wav');
[R32553,fs]=wavread('S32553.wav');
[R32554,fs]=wavread('S32554.wav');
[R32555,fs]=wavread('S32555.wav');

%ângulo 60
[R32601,fs]=wavread('S32601.wav');
[R32602,fs]=wavread('S32602.wav');
[R32603,fs]=wavread('S32603.wav');
[R32604,fs]=wavread('S32604.wav');
[R32605,fs]=wavread('S32605.wav');
```

```
%ângulo 65
[R32651,fs]=wavread('S32651.wav');
[R32652,fs]=wavread('S32652.wav');
[R32653,fs]=wavread('S32653.wav');
[R32654,fs]=wavread('S32654.wav');
[R32655,fs]=wavread('S32655.wav');

%ângulo 70
[R32701,fs]=wavread('S32701.wav');
[R32702,fs]=wavread('S32702.wav');
[R32703,fs]=wavread('S32703.wav');
[R32704,fs]=wavread('S32704.wav');
[R32705,fs]=wavread('S32705.wav');

%ângulo 75
[R32751,fs]=wavread('S32751.wav');
[R32752,fs]=wavread('S32752.wav');
[R32753,fs]=wavread('S32753.wav');
[R32754,fs]=wavread('S32754.wav');
[R32755,fs]=wavread('S32755.wav');

%ângulo 80
[R32801,fs]=wavread('S32801.wav');
[R32802,fs]=wavread('S32802.wav');
[R32803,fs]=wavread('S32803.wav');
[R32804,fs]=wavread('S32804.wav');
[R32805,fs]=wavread('S32805.wav');

%ângulo 85
[R32851,fs]=wavread('S32851.wav');
[R32852,fs]=wavread('S32852.wav');
[R32853,fs]=wavread('S32853.wav');
[R32854,fs]=wavread('S32854.wav');
[R32855,fs]=wavread('S32855.wav');

%ângulo 90
[R32901,fs]=wavread('S32901.wav');
[R32902,fs]=wavread('S32902.wav');
[R32903,fs]=wavread('S32903.wav');
[R32904,fs]=wavread('S32904.wav');
[R32905,fs]=wavread('S32905.wav');

%Fazendo a norm de cada sinal
%norm ang. 0
N32001=norm(R32001)*50
N32002=norm(R32002)*50
N32003=norm(R32003)*50
N32004=norm(R32004)*50
N32005=norm(R32005)*50

%norm ang. 05
N32051=norm(R32051)*50
N32052=norm(R32052)*50
N32053=norm(R32053)*50
N32054=norm(R32054)*50
N32055=norm(R32055)*50

%norm ang. 10
N32101=norm(R32101)*50
N32102=norm(R32102)*50
N32103=norm(R32103)*50
N32104=norm(R32104)*50
N32105=norm(R32105)*50

%norm ang. 15
N32151=norm(R32151)*50
N32152=norm(R32152)*50
N32153=norm(R32153)*50
N32154=norm(R32154)*50
N32155=norm(R32155)*50
```

```
%norm ang. 20
N32201=norm(R32201)*50
N32202=norm(R32202)*50
N32203=norm(R32203)*50
N32204=norm(R32204)*50
N32205=norm(R32205)*50

%norm ang. 25
N32251=norm(R32251)*50
N32252=norm(R32252)*50
N32253=norm(R32253)*50
N32254=norm(R32254)*50
N32255=norm(R32255)*50

%norm ang. 30
N32301=norm(R32301)*50
N32302=norm(R32302)*50
N32303=norm(R32303)*50
N32304=norm(R32304)*50
N32305=norm(R32305)*50

%norm ang. 35
N32351=norm(R32351)*50
N32352=norm(R32352)*50
N32353=norm(R32353)*50
N32354=norm(R32354)*50
N32355=norm(R32355)*50

%norm ang. 40
N32401=norm(R32401)*50
N32402=norm(R32402)*50
N32403=norm(R32403)*50
N32404=norm(R32404)*50
N32405=norm(R32405)*50

%norm ang. 45
N32451=norm(R32451)*50
N32452=norm(R32452)*50
N32453=norm(R32453)*50
N32454=norm(R32454)*50
N32455=norm(R32455)*50

%norm ang. 50
N32501=norm(R32501)*50
N32502=norm(R32502)*50
N32503=norm(R32503)*50
N32504=norm(R32504)*50
N32505=norm(R32505)*50

%norm ang. 55
N32551=norm(R32551)*50
N32552=norm(R32552)*50
N32553=norm(R32553)*50
N32554=norm(R32554)*50
N32555=norm(R32555)*50

%norm ang. 60
N32601=norm(R32601)*50
N32602=norm(R32602)*50
N32603=norm(R32603)*50
N32604=norm(R32604)*50
N32605=norm(R32605)*50

%norm ang. 65
N32651=norm(R32651)*50
N32652=norm(R32652)*50
N32653=norm(R32653)*50
N32654=norm(R32654)*50
N32655=norm(R32655)*50

%norm ang. 70
```

```
N32701=norm(R32701)*50
N32702=norm(R32702)*50
N32703=norm(R32703)*50
N32704=norm(R32704)*50
N32705=norm(R32705)*50

%norm ang. 75
N32751=norm(R32751)*50
N32752=norm(R32752)*50
N32753=norm(R32753)*50
N32754=norm(R32754)*50
N32755=norm(R32755)*50

%norm ang. 80
N32801=norm(R32801)*50
N32802=norm(R32802)*50
N32803=norm(R32803)*50
N32804=norm(R32804)*50
N32805=norm(R32805)*50

%norm ang. 85
N32851=norm(R32851)*50
N32852=norm(R32852)*50
N32853=norm(R32853)*50
N32854=norm(R32854)*50
N32855=norm(R32855)*50

%norm ang. 90
N32901=norm(R32901)*50
N32902=norm(R32902)*50
N32903=norm(R32903)*50
N32904=norm(R32904)*50
N32905=norm(R32905)*50

%Convertendo em binário
%bin ang. 0
B32001=dec2bin(N32001,8)
B32002=dec2bin(N32002,8)
B32003=dec2bin(N32003,8)
B32004=dec2bin(N32004,8)
B32005=dec2bin(N32005,8)

%bin ang. 05
B32051=dec2bin(N32051,8)
B32052=dec2bin(N32052,8)
B32053=dec2bin(N32053,8)
B32054=dec2bin(N32054,8)
B32055=dec2bin(N32055,8)

%bin ang. 10
B32101=dec2bin(N32101,8)
B32102=dec2bin(N32102,8)
B32103=dec2bin(N32103,8)
B32104=dec2bin(N32104,8)
B32105=dec2bin(N32105,8)

%bin ang. 15
B32151=dec2bin(N32151,8)
B32152=dec2bin(N32152,8)
B32153=dec2bin(N32153,8)
B32154=dec2bin(N32154,8)
B32155=dec2bin(N32155,8)

%bin ang. 20
B32201=dec2bin(N32201,8)
B32202=dec2bin(N32202,8)
B32203=dec2bin(N32203,8)
B32204=dec2bin(N32204,8)
B32205=dec2bin(N32205,8)

%bin ang. 25
```

```
B32251=dec2bin(N32251,8)
B32252=dec2bin(N32252,8)
B32253=dec2bin(N32253,8)
B32254=dec2bin(N32254,8)
B32255=dec2bin(N32255,8)

%bin ang. 30
B32301=dec2bin(N32301,8)
B32302=dec2bin(N32302,8)
B32303=dec2bin(N32303,8)
B32304=dec2bin(N32304,8)
B32305=dec2bin(N32305,8)

%bin ang. 35
B32351=dec2bin(N32351,8)
B32352=dec2bin(N32352,8)
B32353=dec2bin(N32353,8)
B32354=dec2bin(N32354,8)
B32355=dec2bin(N32355,8)

%bin ang. 40
B32401=dec2bin(N32401,8)
B32402=dec2bin(N32402,8)
B32403=dec2bin(N32403,8)
B32404=dec2bin(N32404,8)
B32405=dec2bin(N32405,8)

%bin ang. 45
B32451=dec2bin(N32451,8)
B32452=dec2bin(N32452,8)
B32453=dec2bin(N32453,8)
B32454=dec2bin(N32454,8)
B32455=dec2bin(N32455,8)

%bin ang. 50
B32501=dec2bin(N32501,8)
B32502=dec2bin(N32502,8)
B32503=dec2bin(N32503,8)
B32504=dec2bin(N32504,8)
B32505=dec2bin(N32505,8)

%bin ang. 55
B32551=dec2bin(N32551,8)
B32552=dec2bin(N32552,8)
B32553=dec2bin(N32553,8)
B32554=dec2bin(N32554,8)
B32555=dec2bin(N32555,8)

%bin ang. 60
B32601=dec2bin(N32601,8)
B32602=dec2bin(N32602,8)
B32603=dec2bin(N32603,8)
B32604=dec2bin(N32604,8)
B32605=dec2bin(N32605,8)

%bin ang. 65
B32651=dec2bin(N32651,8)
B32652=dec2bin(N32652,8)
B32653=dec2bin(N32653,8)
B32654=dec2bin(N32654,8)
B32655=dec2bin(N32655,8)

%bin ang. 70
B32701=dec2bin(N32701,8)
B32702=dec2bin(N32702,8)
B32703=dec2bin(N32703,8)
B32704=dec2bin(N32704,8)
B32705=dec2bin(N32705,8)

%bin ang. 75
B32751=dec2bin(N32751,8)
```

```
B32752=dec2bin(N32752,8)
B32753=dec2bin(N32753,8)
B32754=dec2bin(N32754,8)
B32755=dec2bin(N32755,8)

%bin ang. 80
B32801=dec2bin(N32801,8)
B32802=dec2bin(N32802,8)
B32803=dec2bin(N32803,8)
B32804=dec2bin(N32804,8)
B32805=dec2bin(N32805,8)

%bin ang. 85
B32851=dec2bin(N32851,8)
B32852=dec2bin(N32852,8)
B32853=dec2bin(N32853,8)
B32854=dec2bin(N32854,8)
B32855=dec2bin(N32855,8)

%bin ang. 90
B32901=dec2bin(N32901,8)
B32902=dec2bin(N32902,8)
B32903=dec2bin(N32903,8)
B32904=dec2bin(N32904,8)
B32905=dec2bin(N32905,8)

%Plotando os gráficos do comprimento dos sinais
P32 = [N32001 N32002 N32003 N32004 N32005 N32051 N32052 N32053 N32054 N32055 N32101 N32102 N32103
N32104 N32105 N32151 N32152 N32153 N32154 N32205 N32201 N32202 N32203 N32204 N32205 N32251 N32252
N32253 N32254 N32255 N32301 N32302 N32303 N32304 N32305 N32351 N32352 N32353 N32354 N32355 N32401
N32402 N32403 N32404 N32405 N32451 N32452 N32453 N32454 N32455 N32501 N32502 N32503 N32504 N32505
N32551 N32552 N32553 N32554 N32555 N32601 N32602 N32603 N32604 N32605 N32651 N32652 N32653 N32654
N32655 N32701 N32702 N32703 N32704 N32705 N32751 N32752 N32753 N32754 N32755 N32801 N32802 N32803
N32804 N32805 N32851 N32852 N32853 N32854 N32855 N32901 N32902 N32903 N32904 N32905]
plot (P32),title('Norm dos sinais - Linha 02'),xlabel('Ângulo'),ylabel('Comprimento do Sinal - Norma 2')
grid on

%Salvando em .txt
%save ang. 0
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32001.txt','B32001','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32002.txt','B32002','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32003.txt','B32003','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32004.txt','B32004','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32005.txt','B32005','-ASCII');

%save ang. 05
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32051.txt','B32051','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32052.txt','B32052','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32053.txt','B32053','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32054.txt','B32054','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32055.txt','B32055','-ASCII');

%save ang. 10
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32101.txt','B32101','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32102.txt','B32102','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32103.txt','B32103','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32104.txt','B32104','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32105.txt','B32105','-ASCII');

%save ang. 15
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32151.txt','B32151','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32152.txt','B32152','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32153.txt','B32153','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32154.txt','B32154','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32155.txt','B32155','-ASCII');

%save ang. 20
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32201.txt','B32201','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32202.txt','B32202','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32203.txt','B32203','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32204.txt','B32204','-ASCII');
```

```
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32205.txt','B32205','-ASCII');

%save ang. 25
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32251.txt','B32251','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32252.txt','B32252','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32253.txt','B32253','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32254.txt','B32254','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32255.txt','B32255','-ASCII');

%save ang. 30
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32301.txt','B32301','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32302.txt','B32302','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32303.txt','B32303','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32304.txt','B32304','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32305.txt','B32305','-ASCII');

%save ang. 35
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32351.txt','B32351','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32352.txt','B32352','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32353.txt','B32353','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32354.txt','B32354','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32355.txt','B32355','-ASCII');

%save ang. 40
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32401.txt','B32401','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32402.txt','B32402','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32403.txt','B32403','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32404.txt','B32404','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32405.txt','B32405','-ASCII');

%save ang. 45
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32451.txt','B32451','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32452.txt','B32452','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32453.txt','B32453','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32454.txt','B32454','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32455.txt','B32455','-ASCII');

%save ang. 50
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32501.txt','B32501','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32502.txt','B32502','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32503.txt','B32503','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32504.txt','B32504','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32505.txt','B32505','-ASCII');

%save ang. 55
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32551.txt','B32551','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32552.txt','B32552','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32553.txt','B32553','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32554.txt','B32554','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32555.txt','B32555','-ASCII');

%save ang. 60
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32601.txt','B32601','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32602.txt','B32602','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32603.txt','B32603','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32604.txt','B32604','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32605.txt','B32605','-ASCII');

%save ang. 65
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32651.txt','B32651','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32652.txt','B32652','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32653.txt','B32653','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32654.txt','B32654','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32655.txt','B32655','-ASCII');

%save ang. 70
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32701.txt','B32701','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32702.txt','B32702','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32703.txt','B32703','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32704.txt','B32704','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32705.txt','B32705','-ASCII');
```

```
%save ang. 75
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32751.txt','B32751','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32752.txt','B32752','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32753.txt','B32753','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32754.txt','B32754','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32755.txt','B32755','-ASCII');

%save ang. 80
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32801.txt','B32801','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32802.txt','B32802','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32803.txt','B32803','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32804.txt','B32804','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32805.txt','B32805','-ASCII');

%save ang. 85
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32851.txt','B32851','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32852.txt','B32852','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32853.txt','B32853','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32854.txt','B32854','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32855.txt','B32855','-ASCII');

%save ang. 90
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32901.txt','B32901','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32902.txt','B32902','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32903.txt','B32903','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32904.txt','B32904','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T32905.txt','B32905','-ASCII');


%****************************************************************
%Centro Universitário de Brasília – UniCEUB
%Faculdade de Ciências Exatas e Tecnologia – FAET
%Curso de Engenharia da Computação
%Projeto Final
%Prof. Orientador: José Julimá
%Aluno: Mateus Silva Teixeira - RA.: 20016207
%****************************************************************
%AQUISIÇÃO E TRATAMENTO DE DADOS EM UM SISTEMA DE ORIENTAÇÃO ACÚSTICA
%****************************************************************
%Tratamento de dados - Linha 03
%****************************************************************
%Legenda
%   S = som
%   R = read - sinal lido e armazenado em uma variável
%   N = norm - comprimento do vetor
%   B = binário
%   primeiro número (3) = microfone utilizado (3)
%   segundo número (3) = linha onde está a fonte de áudio (3)
%   terceiro e quarto número (00) = ângulo da fonte de áudio (00°)
%   quinto número (1) = gravação
%       Obs.: foram realizadas 5 (cinco) gravações em cada ângulo
%****************************************************************

%Realizando a leitura dos sinais em cada ângulo gravado
%ângulo 0
[R33001,fs]=wavread('S33001.wav');
[R33002,fs]=wavread('S33002.wav');
[R33003,fs]=wavread('S33003.wav');
[R33004,fs]=wavread('S33004.wav');
[R33005,fs]=wavread('S33005.wav');

%ângulo 05
[R33051,fs]=wavread('S33051.wav');
[R33052,fs]=wavread('S33052.wav');
[R33053,fs]=wavread('S33053.wav');
[R33054,fs]=wavread('S33054.wav');
[R33055,fs]=wavread('S33055.wav');

%ângulo 10
[R33101,fs]=wavread('S33101.wav');
```

```
[R33102,fs]=wavread('S33102.wav');
[R33103,fs]=wavread('S33103.wav');
[R33104,fs]=wavread('S33104.wav');
[R33105,fs]=wavread('S33105.wav');

%ângulo 15
[R33151,fs]=wavread('S33151.wav');
[R33152,fs]=wavread('S33152.wav');
[R33153,fs]=wavread('S33153.wav');
[R33154,fs]=wavread('S33154.wav');
[R33155,fs]=wavread('S33155.wav');

%ângulo 20
[R33201,fs]=wavread('S33201.wav');
[R33202,fs]=wavread('S33202.wav');
[R33203,fs]=wavread('S33203.wav');
[R33204,fs]=wavread('S33204.wav');
[R33205,fs]=wavread('S33205.wav');

%ângulo 25
[R33251,fs]=wavread('S33251.wav');
[R33252,fs]=wavread('S33252.wav');
[R33253,fs]=wavread('S33253.wav');
[R33254,fs]=wavread('S33254.wav');
[R33255,fs]=wavread('S33255.wav');

%ângulo 30
[R33301,fs]=wavread('S33301.wav');
[R33302,fs]=wavread('S33302.wav');
[R33303,fs]=wavread('S33303.wav');
[R33304,fs]=wavread('S33304.wav');
[R33305,fs]=wavread('S33305.wav');

%ângulo 35
[R33351,fs]=wavread('S33351.wav');
[R33352,fs]=wavread('S33352.wav');
[R33353,fs]=wavread('S33353.wav');
[R33354,fs]=wavread('S33354.wav');
[R33355,fs]=wavread('S33355.wav');

%ângulo 40
[R33401,fs]=wavread('S33401.wav');
[R33402,fs]=wavread('S33402.wav');
[R33403,fs]=wavread('S33403.wav');
[R33404,fs]=wavread('S33404.wav');
[R33405,fs]=wavread('S33405.wav');

%ângulo 45
[R33451,fs]=wavread('S33451.wav');
[R33452,fs]=wavread('S33452.wav');
[R33453,fs]=wavread('S33453.wav');
[R33454,fs]=wavread('S33454.wav');
[R33455,fs]=wavread('S33455.wav');

%ângulo 50
[R33501,fs]=wavread('S33501.wav');
[R33502,fs]=wavread('S33502.wav');
[R33503,fs]=wavread('S33503.wav');
[R33504,fs]=wavread('S33504.wav');
[R33505,fs]=wavread('S33505.wav');

%ângulo 55
[R33551,fs]=wavread('S33551.wav');
[R33552,fs]=wavread('S33552.wav');
[R33553,fs]=wavread('S33553.wav');
[R33554,fs]=wavread('S33554.wav');
[R33555,fs]=wavread('S33555.wav');

%ângulo 60
[R33601,fs]=wavread('S33601.wav');
[R33602,fs]=wavread('S33602.wav');
```

```
[R33603,fs]=wavread('S33603.wav');
[R33604,fs]=wavread('S33604.wav');
[R33605,fs]=wavread('S33605.wav');

%ângulo 65
[R33651,fs]=wavread('S33651.wav');
[R33652,fs]=wavread('S33652.wav');
[R33653,fs]=wavread('S33653.wav');
[R33654,fs]=wavread('S33654.wav');
[R33655,fs]=wavread('S33655.wav');

%ângulo 70
[R33701,fs]=wavread('S33701.wav');
[R33702,fs]=wavread('S33702.wav');
[R33703,fs]=wavread('S33703.wav');
[R33704,fs]=wavread('S33704.wav');
[R33705,fs]=wavread('S33705.wav');

%ângulo 75
[R33751,fs]=wavread('S33751.wav');
[R33752,fs]=wavread('S33752.wav');
[R33753,fs]=wavread('S33753.wav');
[R33754,fs]=wavread('S33754.wav');
[R33755,fs]=wavread('S33755.wav');

%ângulo 80
[R33801,fs]=wavread('S33801.wav');
[R33802,fs]=wavread('S33802.wav');
[R33803,fs]=wavread('S33803.wav');
[R33804,fs]=wavread('S33804.wav');
[R33805,fs]=wavread('S33805.wav');

%ângulo 85
[R33851,fs]=wavread('S33851.wav');
[R33852,fs]=wavread('S33852.wav');
[R33853,fs]=wavread('S33853.wav');
[R33854,fs]=wavread('S33854.wav');
[R33855,fs]=wavread('S33855.wav');

%ângulo 90
[R33901,fs]=wavread('S33901.wav');
[R33902,fs]=wavread('S33902.wav');
[R33903,fs]=wavread('S33903.wav');
[R33904,fs]=wavread('S33904.wav');
[R33905,fs]=wavread('S33905.wav');

%Fazendo a norm de cada sinal
%norm ang. 0
N33001=norm(R33001)*50
N33002=norm(R33002)*50
N33003=norm(R33003)*50
N33004=norm(R33004)*50
N33005=norm(R33005)*50

%norm ang. 05
N33051=norm(R33051)*50
N33052=norm(R33052)*50
N33053=norm(R33053)*50
N33054=norm(R33054)*50
N33055=norm(R33055)*50

%norm ang. 10
N33101=norm(R33101)*50
N33102=norm(R33102)*50
N33103=norm(R33103)*50
N33104=norm(R33104)*50
N33105=norm(R33105)*50

%norm ang. 15
N33151=norm(R33151)*50
N33152=norm(R33152)*50
```

```
N33153=norm(R33153)*50
N33154=norm(R33154)*50
N33155=norm(R33155)*50

%norm ang. 20
N33201=norm(R33201)*50
N33202=norm(R33202)*50
N33203=norm(R33203)*50
N33204=norm(R33204)*50
N33205=norm(R33205)*50

%norm ang. 25
N33251=norm(R33251)*50
N33252=norm(R33252)*50
N33253=norm(R33253)*50
N33254=norm(R33254)*50
N33255=norm(R33255)*50

%norm ang. 30
N33301=norm(R33301)*50
N33302=norm(R33302)*50
N33303=norm(R33303)*50
N33304=norm(R33304)*50
N33305=norm(R33305)*50

%norm ang. 35
N33351=norm(R33351)*50
N33352=norm(R33352)*50
N33353=norm(R33353)*50
N33354=norm(R33354)*50
N33355=norm(R33355)*50

%norm ang. 40
N33401=norm(R33401)*50
N33402=norm(R33402)*50
N33403=norm(R33403)*50
N33404=norm(R33404)*50
N33405=norm(R33405)*50

%norm ang. 45
N33451=norm(R33451)*50
N33452=norm(R33452)*50
N33453=norm(R33453)*50
N33454=norm(R33454)*50
N33455=norm(R33455)*50

%norm ang. 50
N33501=norm(R33501)*50
N33502=norm(R33502)*50
N33503=norm(R33503)*50
N33504=norm(R33504)*50
N33505=norm(R33505)*50

%norm ang. 55
N33551=norm(R33551)*50
N33552=norm(R33552)*50
N33553=norm(R33553)*50
N33554=norm(R33554)*50
N33555=norm(R33555)*50

%norm ang. 60
N33601=norm(R33601)*50
N33602=norm(R33602)*50
N33603=norm(R33603)*50
N33604=norm(R33604)*50
N33605=norm(R33605)*50

%norm ang. 65
N33651=norm(R33651)*50
N33652=norm(R33652)*50
N33653=norm(R33653)*50
```

```
N33654=norm(R33654)*50
N33655=norm(R33655)*50

%norm ang. 70
N33701=norm(R33701)*50
N33702=norm(R33702)*50
N33703=norm(R33703)*50
N33704=norm(R33704)*50
N33705=norm(R33705)*50

%norm ang. 75
N33751=norm(R33751)*50
N33752=norm(R33752)*50
N33753=norm(R33753)*50
N33754=norm(R33754)*50
N33755=norm(R33755)*50

%norm ang. 80
N33801=norm(R33801)*50
N33802=norm(R33802)*50
N33803=norm(R33803)*50
N33804=norm(R33804)*50
N33805=norm(R33805)*50

%norm ang. 85
N33851=norm(R33851)*50
N33852=norm(R33852)*50
N33853=norm(R33853)*50
N33854=norm(R33854)*50
N33855=norm(R33855)*50

%norm ang. 90
N33901=norm(R33901)*50
N33902=norm(R33902)*50
N33903=norm(R33903)*50
N33904=norm(R33904)*50
N33905=norm(R33905)*50

%Convertendo em binário
%bin ang. 0
B33001=dec2bin(N33001,8)
B33002=dec2bin(N33002,8)
B33003=dec2bin(N33003,8)
B33004=dec2bin(N33004,8)
B33005=dec2bin(N33005,8)

%bin ang. 05
B33051=dec2bin(N33051,8)
B33052=dec2bin(N33052,8)
B33053=dec2bin(N33053,8)
B33054=dec2bin(N33054,8)
B33055=dec2bin(N33055,8)

%bin ang. 10
B33101=dec2bin(N33101,8)
B33102=dec2bin(N33102,8)
B33103=dec2bin(N33103,8)
B33104=dec2bin(N33104,8)
B33105=dec2bin(N33105,8)

%bin ang. 15
B33151=dec2bin(N33151,8)
B33152=dec2bin(N33152,8)
B33153=dec2bin(N33153,8)
B33154=dec2bin(N33154,8)
B33155=dec2bin(N33155,8)

%bin ang. 20
B33201=dec2bin(N33201,8)
B33202=dec2bin(N33202,8)
B33203=dec2bin(N33203,8)
```

```
B33204=dec2bin(N33204,8)
B33205=dec2bin(N33205,8)

%bin ang. 25
B33251=dec2bin(N33251,8)
B33252=dec2bin(N33252,8)
B33253=dec2bin(N33253,8)
B33254=dec2bin(N33254,8)
B33255=dec2bin(N33255,8)

%bin ang. 30
B33301=dec2bin(N33301,8)
B33302=dec2bin(N33302,8)
B33303=dec2bin(N33303,8)
B33304=dec2bin(N33304,8)
B33305=dec2bin(N33305,8)

%bin ang. 35
B33351=dec2bin(N33351,8)
B33352=dec2bin(N33352,8)
B33353=dec2bin(N33353,8)
B33354=dec2bin(N33354,8)
B33355=dec2bin(N33355,8)

%bin ang. 40
B33401=dec2bin(N33401,8)
B33402=dec2bin(N33402,8)
B33403=dec2bin(N33403,8)
B33404=dec2bin(N33404,8)
B33405=dec2bin(N33405,8)

%bin ang. 45
B33451=dec2bin(N33451,8)
B33452=dec2bin(N33452,8)
B33453=dec2bin(N33453,8)
B33454=dec2bin(N33454,8)
B33455=dec2bin(N33455,8)

%bin ang. 50
B33501=dec2bin(N33501,8)
B33502=dec2bin(N33502,8)
B33503=dec2bin(N33503,8)
B33504=dec2bin(N33504,8)
B33505=dec2bin(N33505,8)

%bin ang. 55
B33551=dec2bin(N33551,8)
B33552=dec2bin(N33552,8)
B33553=dec2bin(N33553,8)
B33554=dec2bin(N33554,8)
B33555=dec2bin(N33555,8)

%bin ang. 60
B33601=dec2bin(N33601,8)
B33602=dec2bin(N33602,8)
B33603=dec2bin(N33603,8)
B33604=dec2bin(N33604,8)
B33605=dec2bin(N33605,8)

%bin ang. 65
B33651=dec2bin(N33651,8)
B33652=dec2bin(N33652,8)
B33653=dec2bin(N33653,8)
B33654=dec2bin(N33654,8)
B33655=dec2bin(N33655,8)

%bin ang. 70
B33701=dec2bin(N33701,8)
B33702=dec2bin(N33702,8)
B33703=dec2bin(N33703,8)
B33704=dec2bin(N33704,8)
```

```
B33705=dec2bin(N33705,8)

%bin ang. 75
B33751=dec2bin(N33751,8)
B33752=dec2bin(N33752,8)
B33753=dec2bin(N33753,8)
B33754=dec2bin(N33754,8)
B33755=dec2bin(N33755,8)

%bin ang. 80
B33801=dec2bin(N33801,8)
B33802=dec2bin(N33802,8)
B33803=dec2bin(N33803,8)
B33804=dec2bin(N33804,8)
B33805=dec2bin(N33805,8)

%bin ang. 85
B33851=dec2bin(N33851,8)
B33852=dec2bin(N33852,8)
B33853=dec2bin(N33853,8)
B33854=dec2bin(N33854,8)
B33855=dec2bin(N33855,8)

%bin ang. 90
B33901=dec2bin(N33901,8)
B33902=dec2bin(N33902,8)
B33903=dec2bin(N33903,8)
B33904=dec2bin(N33904,8)
B33905=dec2bin(N33905,8)

%Plotando os gráficos do comprimento dos sinais
P33 = [N33001 N33002 N33003 N33004 N33005 N33051 N33052 N33053 N33054 N33055 N33101 N33102 N33103
N33104 N33105 N33151 N33152 N33153 N33154 N33205 N33201 N33202 N33203 N33204 N33205 N33251 N33252
N33253 N33254 N33255 N33301 N33302 N33303 N33304 N33305 N33351 N33352 N33353 N33354 N33355 N33401
N33402 N33403 N33404 N33405 N33451 N33452 N33453 N33454 N33455 N33501 N33502 N33503 N33504 N33505
N33551 N33552 N33553 N33554 N33555 N33601 N33602 N33603 N33604 N33605 N33651 N33652 N33653 N33654
N33655 N33701 N33702 N33703 N33704 N33705 N33751 N33752 N33753 N33754 N33755 N33801 N33802 N33803
N33804 N33805 N33851 N33852 N33853 N33854 N33855 N33901 N33902 N33903 N33904 N33905]
plot (P33),title('Norm dos sinais - Linha 03'),xlabel('Ângulo'),ylabel('Comprimento do Sinal - Norma 2')
grid on

%Salvando em .txt
%save ang. 0
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33001.txt','B33001','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33002.txt','B33002','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33003.txt','B33003','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33004.txt','B33004','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33005.txt','B33005','-ASCII');

%save ang. 05
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33051.txt','B33051','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33052.txt','B33052','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33053.txt','B33053','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33054.txt','B33054','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33055.txt','B33055','-ASCII');

%save ang. 10
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33101.txt','B33101','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33102.txt','B33102','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33103.txt','B33103','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33104.txt','B33104','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33105.txt','B33105','-ASCII');

%save ang. 15
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33151.txt','B33151','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33152.txt','B33152','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33153.txt','B33153','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33154.txt','B33154','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33155.txt','B33155','-ASCII');

%save ang. 20
```

```
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33201.txt','B33201','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33202.txt','B33202','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33203.txt','B33203','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33204.txt','B33204','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33205.txt','B33205','-ASCII');

%save ang. 25
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33251.txt','B33251','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33252.txt','B33252','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33253.txt','B33253','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33254.txt','B33254','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33255.txt','B33255','-ASCII');

%save ang. 30
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33301.txt','B33301','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33302.txt','B33302','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33303.txt','B33303','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33304.txt','B33304','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33305.txt','B33305','-ASCII');

%save ang. 35
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33351.txt','B33351','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33352.txt','B33352','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33353.txt','B33353','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33354.txt','B33354','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33355.txt','B33355','-ASCII');

%save ang. 40
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33401.txt','B33401','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33402.txt','B33402','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33403.txt','B33403','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33404.txt','B33404','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33405.txt','B33405','-ASCII');

%save ang. 45
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33451.txt','B33451','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33452.txt','B33452','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33453.txt','B33453','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33454.txt','B33454','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33455.txt','B33455','-ASCII');

%save ang. 50
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33501.txt','B33501','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33502.txt','B33502','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33503.txt','B33503','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33504.txt','B33504','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33505.txt','B33505','-ASCII');

%save ang. 55
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33551.txt','B33551','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33552.txt','B33552','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33553.txt','B33553','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33554.txt','B33554','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33555.txt','B33555','-ASCII');

%save ang. 60
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33601.txt','B33601','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33602.txt','B33602','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33603.txt','B33603','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33604.txt','B33604','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33605.txt','B33605','-ASCII');

%save ang. 65
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33651.txt','B33651','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33652.txt','B33652','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33653.txt','B33653','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33654.txt','B33654','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33655.txt','B33655','-ASCII');

%save ang. 70
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33701.txt','B33701','-ASCII');
```

```
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33702.txt','B33702','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33703.txt','B33703','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33704.txt','B33704','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33705.txt','B33705','-ASCII');

%save ang. 75
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33751.txt','B33751','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33752.txt','B33752','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33753.txt','B33753','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33754.txt','B33754','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33755.txt','B33755','-ASCII');

%save ang. 80
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33801.txt','B33801','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33802.txt','B33802','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33803.txt','B33803','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33804.txt','B33804','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33805.txt','B33805','-ASCII');

%save ang. 85
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33851.txt','B33851','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33852.txt','B33852','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33853.txt','B33853','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33854.txt','B33854','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33855.txt','B33855','-ASCII');

%save ang. 90
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33901.txt','B33901','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33902.txt','B33902','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33903.txt','B33903','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33904.txt','B33904','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T33905.txt','B33905','-ASCII');


%***************************************************************
%Centro Universitário de Brasília – UniCEUB
%Faculdade de Ciências Exatas e Tecnologia – FAET
%Curso de Engenharia da Computação
%Projeto Final
%Prof. Orientador: José Julimá
%Aluno: Mateus Silva Teixeira - RA.: 20016207
%***************************************************************
%AQUISIÇÃO E TRATAMENTO DE DADOS EM UM SISTEMA DE ORIENTAÇÃO ACÚSTICA
%***************************************************************
%Tratamento de dados - Linha 04
%***************************************************************
%Legenda
%   S = som
%   R = read - sinal lido e armazenado em uma variável
%   N = norm - comprimento do vetor
%   B = binário
%   primeiro número (3) = microfone utilizado (3)
%   segundo número (4) = linha onde está a fonte de áudio (4)
%   terceiro e quarto número (00) = ângulo da fonte de áudio (00)
%   quinto número (1) = gravação
%       Obs.: foram realizadas 5 (cinco) gravações em cada ângulo
%***************************************************************

%Realizando a leitura dos sinais
%ângulo 0
[R34001,fs]=wavread('S34001.wav');
[R34002,fs]=wavread('S34002.wav');
[R34003,fs]=wavread('S34003.wav');
[R34004,fs]=wavread('S34004.wav');
[R34005,fs]=wavread('S34005.wav');

%ângulo 05
[R34051,fs]=wavread('S34051.wav');
[R34052,fs]=wavread('S34052.wav');
[R34053,fs]=wavread('S34053.wav');
[R34054,fs]=wavread('S34054.wav');
```

```
[R34055,fs]=wavread('S34055.wav');

%ângulo 10
[R34101,fs]=wavread('S34101.wav');
[R34102,fs]=wavread('S34102.wav');
[R34103,fs]=wavread('S34103.wav');
[R34104,fs]=wavread('S34104.wav');
[R34105,fs]=wavread('S34105.wav');

%ângulo 15
[R34151,fs]=wavread('S34151.wav');
[R34152,fs]=wavread('S34152.wav');
[R34153,fs]=wavread('S34153.wav');
[R34154,fs]=wavread('S34154.wav');
[R34155,fs]=wavread('S34155.wav');

%ângulo 20
[R34201,fs]=wavread('S34201.wav');
[R34202,fs]=wavread('S34202.wav');
[R34203,fs]=wavread('S34203.wav');
[R34204,fs]=wavread('S34204.wav');
[R34205,fs]=wavread('S34205.wav');

%ângulo 25
[R34251,fs]=wavread('S34251.wav');
[R34252,fs]=wavread('S34252.wav');
[R34253,fs]=wavread('S34253.wav');
[R34254,fs]=wavread('S34254.wav');
[R34255,fs]=wavread('S34255.wav');

%ângulo 30
[R34301,fs]=wavread('S34301.wav');
[R34302,fs]=wavread('S34302.wav');
[R34303,fs]=wavread('S34303.wav');
[R34304,fs]=wavread('S34304.wav');
[R34305,fs]=wavread('S34305.wav');

%ângulo 35
[R34351,fs]=wavread('S34351.wav');
[R34352,fs]=wavread('S34352.wav');
[R34353,fs]=wavread('S34353.wav');
[R34354,fs]=wavread('S34354.wav');
[R34355,fs]=wavread('S34355.wav');

%ângulo 40
[R34401,fs]=wavread('S34401.wav');
[R34402,fs]=wavread('S34402.wav');
[R34403,fs]=wavread('S34403.wav');
[R34404,fs]=wavread('S34404.wav');
[R34405,fs]=wavread('S34405.wav');

%ângulo 45
[R34451,fs]=wavread('S34451.wav');
[R34452,fs]=wavread('S34452.wav');
[R34453,fs]=wavread('S34453.wav');
[R34454,fs]=wavread('S34454.wav');
[R34455,fs]=wavread('S34455.wav');

%ângulo 50
[R34501,fs]=wavread('S34501.wav');
[R34502,fs]=wavread('S34502.wav');
[R34503,fs]=wavread('S34503.wav');
[R34504,fs]=wavread('S34504.wav');
[R34505,fs]=wavread('S34505.wav');

%ângulo 55
[R34551,fs]=wavread('S34551.wav');
[R34552,fs]=wavread('S34552.wav');
[R34553,fs]=wavread('S34553.wav');
[R34554,fs]=wavread('S34554.wav');
[R34555,fs]=wavread('S34555.wav');
```

```
%ângulo 60
[R34601,fs]=wavread('S34601.wav');
[R34602,fs]=wavread('S34602.wav');
[R34603,fs]=wavread('S34603.wav');
[R34604,fs]=wavread('S34604.wav');
[R34605,fs]=wavread('S34605.wav');

%ângulo 65
[R34651,fs]=wavread('S34651.wav');
[R34652,fs]=wavread('S34652.wav');
[R34653,fs]=wavread('S34653.wav');
[R34654,fs]=wavread('S34654.wav');
[R34655,fs]=wavread('S34655.wav');

%ângulo 70
[R34701,fs]=wavread('S34701.wav');
[R34702,fs]=wavread('S34702.wav');
[R34703,fs]=wavread('S34703.wav');
[R34704,fs]=wavread('S34704.wav');
[R34705,fs]=wavread('S34705.wav');

%ângulo 75
[R34751,fs]=wavread('S34751.wav');
[R34752,fs]=wavread('S34752.wav');
[R34753,fs]=wavread('S34753.wav');
[R34754,fs]=wavread('S34754.wav');
[R34755,fs]=wavread('S34755.wav');

%ângulo 80
[R34801,fs]=wavread('S34801.wav');
[R34802,fs]=wavread('S34802.wav');
[R34803,fs]=wavread('S34803.wav');
[R34804,fs]=wavread('S34804.wav');
[R34805,fs]=wavread('S34805.wav');

%ângulo 85
[R34851,fs]=wavread('S34851.wav');
[R34852,fs]=wavread('S34852.wav');
[R34853,fs]=wavread('S34853.wav');
[R34854,fs]=wavread('S34854.wav');
[R34855,fs]=wavread('S34855.wav');

%ângulo 90
[R34901,fs]=wavread('S34901.wav');
[R34902,fs]=wavread('S34902.wav');
[R34903,fs]=wavread('S34903.wav');
[R34904,fs]=wavread('S34904.wav');
[R34905,fs]=wavread('S34905.wav');

%Fazendo a norm de cada sinal
%norm ang. 0
N34001=norm(R34001)*50
N34002=norm(R34002)*50
N34003=norm(R34003)*50
N34004=norm(R34004)*50
N34005=norm(R34005)*50

%norm ang. 05
N34051=norm(R34051)*50
N34052=norm(R34052)*50
N34053=norm(R34053)*50
N34054=norm(R34054)*50
N34055=norm(R34055)*50

%norm ang. 10
N34101=norm(R34101)*50
N34102=norm(R34102)*50
N34103=norm(R34103)*50
N34104=norm(R34104)*50
N34105=norm(R34105)*50
```

```
%norm ang. 15
N34151=norm(R34151)*50
N34152=norm(R34152)*50
N34153=norm(R34153)*50
N34154=norm(R34154)*50
N34155=norm(R34155)*50

%norm ang. 20
N34201=norm(R34201)*50
N34202=norm(R34202)*50
N34203=norm(R34203)*50
N34204=norm(R34204)*50
N34205=norm(R34205)*50

%norm ang. 25
N34251=norm(R34251)*50
N34252=norm(R34252)*50
N34253=norm(R34253)*50
N34254=norm(R34254)*50
N34255=norm(R34255)*50

%norm ang. 30
N34301=norm(R34301)*50
N34302=norm(R34302)*50
N34303=norm(R34303)*50
N34304=norm(R34304)*50
N34305=norm(R34305)*50

%norm ang. 35
N34351=norm(R34351)*50
N34352=norm(R34352)*50
N34353=norm(R34353)*50
N34354=norm(R34354)*50
N34355=norm(R34355)*50

%norm ang. 40
N34401=norm(R34401)*50
N34402=norm(R34402)*50
N34403=norm(R34403)*50
N34404=norm(R34404)*50
N34405=norm(R34405)*50

%norm ang. 45
N34451=norm(R34451)*50
N34452=norm(R34452)*50
N34453=norm(R34453)*50
N34454=norm(R34454)*50
N34455=norm(R34455)*50

%norm ang. 50
N34501=norm(R34501)*50
N34502=norm(R34502)*50
N34503=norm(R34503)*50
N34504=norm(R34504)*50
N34505=norm(R34505)*50

%norm ang. 55
N34551=norm(R34551)*50
N34552=norm(R34552)*50
N34553=norm(R34553)*50
N34554=norm(R34554)*50
N34555=norm(R34555)*50

%norm ang. 60
N34601=norm(R34601)*50
N34602=norm(R34602)*50
N34603=norm(R34603)*50
N34604=norm(R34604)*50
N34605=norm(R34605)*50
```

```
%norm ang. 65
N34651=norm(R34651)*50
N34652=norm(R34652)*50
N34653=norm(R34653)*50
N34654=norm(R34654)*50
N34655=norm(R34655)*50

%norm ang. 70
N34701=norm(R34701)*50
N34702=norm(R34702)*50
N34703=norm(R34703)*50
N34704=norm(R34704)*50
N34705=norm(R34705)*50

%norm ang. 75
N34751=norm(R34751)*50
N34752=norm(R34752)*50
N34753=norm(R34753)*50
N34754=norm(R34754)*50
N34755=norm(R34755)*50

%norm ang. 80
N34801=norm(R34801)*50
N34802=norm(R34802)*50
N34803=norm(R34803)*50
N34804=norm(R34804)*50
N34805=norm(R34805)*50

%norm ang. 85
N34851=norm(R34851)*50
N34852=norm(R34852)*50
N34853=norm(R34853)*50
N34854=norm(R34854)*50
N34855=norm(R34855)*50

%norm ang. 90
N34901=norm(R34901)*50
N34902=norm(R34902)*50
N34903=norm(R34903)*50
N34904=norm(R34904)*50
N34905=norm(R34905)*50

%Convertendo em binário
%bin ang. 0
B34001=dec2bin(N34001,8)
B34002=dec2bin(N34002,8)
B34003=dec2bin(N34003,8)
B34004=dec2bin(N34004,8)
B34005=dec2bin(N34005,8)

%bin ang. 05
B34051=dec2bin(N34051,8)
B34052=dec2bin(N34052,8)
B34053=dec2bin(N34053,8)
B34054=dec2bin(N34054,8)
B34055=dec2bin(N34055,8)

%bin ang. 10
B34101=dec2bin(N34101,8)
B34102=dec2bin(N34102,8)
B34103=dec2bin(N34103,8)
B34104=dec2bin(N34104,8)
B34105=dec2bin(N34105,8)

%bin ang. 15
B34151=dec2bin(N34151,8)
B34152=dec2bin(N34152,8)
B34153=dec2bin(N34153,8)
B34154=dec2bin(N34154,8)
B34155=dec2bin(N34155,8)
```

```
%bin ang. 20
B34201=dec2bin(N34201,8)
B34202=dec2bin(N34202,8)
B34203=dec2bin(N34203,8)
B34204=dec2bin(N34204,8)
B34205=dec2bin(N34205,8)

%bin ang. 25
B34251=dec2bin(N34251,8)
B34252=dec2bin(N34252,8)
B34253=dec2bin(N34253,8)
B34254=dec2bin(N34254,8)
B34255=dec2bin(N34255,8)

%bin ang. 30
B34301=dec2bin(N34301,8)
B34302=dec2bin(N34302,8)
B34303=dec2bin(N34303,8)
B34304=dec2bin(N34304,8)
B34305=dec2bin(N34305,8)

%bin ang. 35
B34351=dec2bin(N34351,8)
B34352=dec2bin(N34352,8)
B34353=dec2bin(N34353,8)
B34354=dec2bin(N34354,8)
B34355=dec2bin(N34355,8)

%bin ang. 40
B34401=dec2bin(N34401,8)
B34402=dec2bin(N34402,8)
B34403=dec2bin(N34403,8)
B34404=dec2bin(N34404,8)
B34405=dec2bin(N34405,8)

%bin ang. 45
B34451=dec2bin(N34451,8)
B34452=dec2bin(N34452,8)
B34453=dec2bin(N34453,8)
B34454=dec2bin(N34454,8)
B34455=dec2bin(N34455,8)

%bin ang. 50
B34501=dec2bin(N34501,8)
B34502=dec2bin(N34502,8)
B34503=dec2bin(N34503,8)
B34504=dec2bin(N34504,8)
B34505=dec2bin(N34505,8)

%bin ang. 55
B34551=dec2bin(N34551,8)
B34552=dec2bin(N34552,8)
B34553=dec2bin(N34553,8)
B34554=dec2bin(N34554,8)
B34555=dec2bin(N34555,8)

%bin ang. 60
B34601=dec2bin(N34601,8)
B34602=dec2bin(N34602,8)
B34603=dec2bin(N34603,8)
B34604=dec2bin(N34604,8)
B34605=dec2bin(N34605,8)

%bin ang. 65
B34651=dec2bin(N34651,8)
B34652=dec2bin(N34652,8)
B34653=dec2bin(N34653,8)
B34654=dec2bin(N34654,8)
B34655=dec2bin(N34655,8)

%bin ang. 70
```

```
B34701=dec2bin(N34701,8)
B34702=dec2bin(N34702,8)
B34703=dec2bin(N34703,8)
B34704=dec2bin(N34704,8)
B34705=dec2bin(N34705,8)

%bin ang. 75
B34751=dec2bin(N34751,8)
B34752=dec2bin(N34752,8)
B34753=dec2bin(N34753,8)
B34754=dec2bin(N34754,8)
B34755=dec2bin(N34755,8)

%bin ang. 80
B34801=dec2bin(N34801,8)
B34802=dec2bin(N34802,8)
B34803=dec2bin(N34803,8)
B34804=dec2bin(N34804,8)
B34805=dec2bin(N34805,8)

%bin ang. 85
B34851=dec2bin(N34851,8)
B34852=dec2bin(N34852,8)
B34853=dec2bin(N34853,8)
B34854=dec2bin(N34854,8)
B34855=dec2bin(N34855,8)

%bin ang. 90
B34901=dec2bin(N34901,8)
B34902=dec2bin(N34902,8)
B34903=dec2bin(N34903,8)
B34904=dec2bin(N34904,8)
B34905=dec2bin(N34905,8)

%Plotando os gráficos do comprimento dos sinais
P34 = [N34001 N34002 N34003 N34004 N34005 N34051 N34052 N34053 N34054 N34055 N34101 N34102 N34103
N34104 N34105 N34151 N34152 N34153 N34154 N34205 N34201 N34202 N34203 N34204 N34205 N34251 N34252
N34253 N34254 N34255 N34301 N34302 N34303 N34304 N34305 N34351 N34352 N34353 N34354 N34355 N34401
N34402 N34403 N34404 N34405 N34451 N34452 N34453 N34454 N34455 N34501 N34502 N34503 N34504 N34505
N34551 N34552 N34553 N34554 N34555 N34601 N34602 N34603 N34604 N34605 N34651 N34652 N34653 N34654
N34655 N34701 N34702 N34703 N34704 N34705 N34751 N34752 N34753 N34754 N34755 N34801 N34802 N34803
N34804 N34805 N34851 N34852 N34853 N34854 N34855 N34901 N34902 N34903 N34904 N34905]
plot(P34),title('Norm dos sinais - Linha 04'),xlabel('Ângulo'),ylabel('Comprimento do Sinal - Norma 2')
grid on

%Salvando em .txt
%save ang. 0
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34001.txt','B34001','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34002.txt','B34002','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34003.txt','B34003','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34004.txt','B34004','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34005.txt','B34005','-ASCII');

%save ang. 05
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34051.txt','B34051','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34052.txt','B34052','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34053.txt','B34053','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34054.txt','B34054','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34055.txt','B34055','-ASCII');

%save ang. 10
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34101.txt','B34101','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34102.txt','B34102','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34103.txt','B34103','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34104.txt','B34104','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34105.txt','B34105','-ASCII');

%save ang. 15
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34151.txt','B34151','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34152.txt','B34152','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34153.txt','B34153','-ASCII');
```

```
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34154.txt','B34154','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34155.txt','B34155','-ASCII');

%save ang. 20
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34201.txt','B34201','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34202.txt','B34202','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34203.txt','B34203','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34204.txt','B34204','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34205.txt','B34205','-ASCII');

%save ang. 25
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34251.txt','B34251','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34252.txt','B34252','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34253.txt','B34253','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34254.txt','B34254','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34255.txt','B34255','-ASCII');

%save ang. 30
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34301.txt','B34301','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34302.txt','B34302','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34303.txt','B34303','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34304.txt','B34304','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34305.txt','B34305','-ASCII');

%save ang. 35
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34351.txt','B34351','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34352.txt','B34352','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34353.txt','B34353','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34354.txt','B34354','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34355.txt','B34355','-ASCII');

%save ang. 40
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34401.txt','B34401','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34402.txt','B34402','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34403.txt','B34403','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34404.txt','B34404','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34405.txt','B34405','-ASCII');

%save ang. 45
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34451.txt','B34451','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34452.txt','B34452','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34453.txt','B34453','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34454.txt','B34454','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34455.txt','B34455','-ASCII');

%save ang. 50
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34501.txt','B34501','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34502.txt','B34502','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34503.txt','B34503','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34504.txt','B34504','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34505.txt','B34505','-ASCII');

%save ang. 55
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34551.txt','B34551','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34552.txt','B34552','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34553.txt','B34553','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34554.txt','B34554','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34555.txt','B34555','-ASCII');

%save ang. 60
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34601.txt','B34601','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34602.txt','B34602','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34603.txt','B34603','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34604.txt','B34604','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34605.txt','B34605','-ASCII');

%save ang. 65
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34651.txt','B34651','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34652.txt','B34652','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34653.txt','B34653','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34654.txt','B34654','-ASCII');
```

```
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34655.txt','B34655','-ASCII');

%save ang. 70
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34701.txt','B34701','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34702.txt','B34702','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34703.txt','B34703','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34704.txt','B34704','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34705.txt','B34705','-ASCII');

%save ang. 75
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34751.txt','B34751','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34752.txt','B34752','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34753.txt','B34753','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34754.txt','B34754','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34755.txt','B34755','-ASCII');

%save ang. 80
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34801.txt','B34801','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34802.txt','B34802','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34803.txt','B34803','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34804.txt','B34804','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34805.txt','B34805','-ASCII');

%save ang. 85
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34851.txt','B34851','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34852.txt','B34852','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34853.txt','B34853','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34854.txt','B34854','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34855.txt','B34855','-ASCII');

%save ang. 90
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34901.txt','B34901','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34902.txt','B34902','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34903.txt','B34903','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34904.txt','B34904','-ASCII');
save ('C:\PROJETO_FINAL\Matlab-RedeNeural\filem\T34905.txt','B34905','-ASCII');
```

Anexo B – Circuito Proposto Anteriormente

A primeira proposta deste projeto tinha como um dos fins a criação de um circuito que fosse capaz de tratar o sinal capturado pelos microfones.

O circuito tinha como objetivo subtrair o sinal capturado pelos microfones, amplificar, converter para binário e enviar os dados do sinal de áudio para o microcomputador pela porta paralela do PC.

Esse circuito foi substituído pela placa de som do PC e pelo softwares Cool Edit e Matlab.

O circuito foi construído até a etapa do conversor analógico digital, conjunto 05 – Figura B.1, mas devido as dificuldades de comunicação do circuito com o computador via porta paralela, optou-se pela substituição do mesmo.

Ressalta-se que essa alteração não interferiu no objetivo do projeto, de determinar por simulação, a direção angular de uma fonte de áudio.

A seguir tem-se a descrição do circuito que seria utilizado:

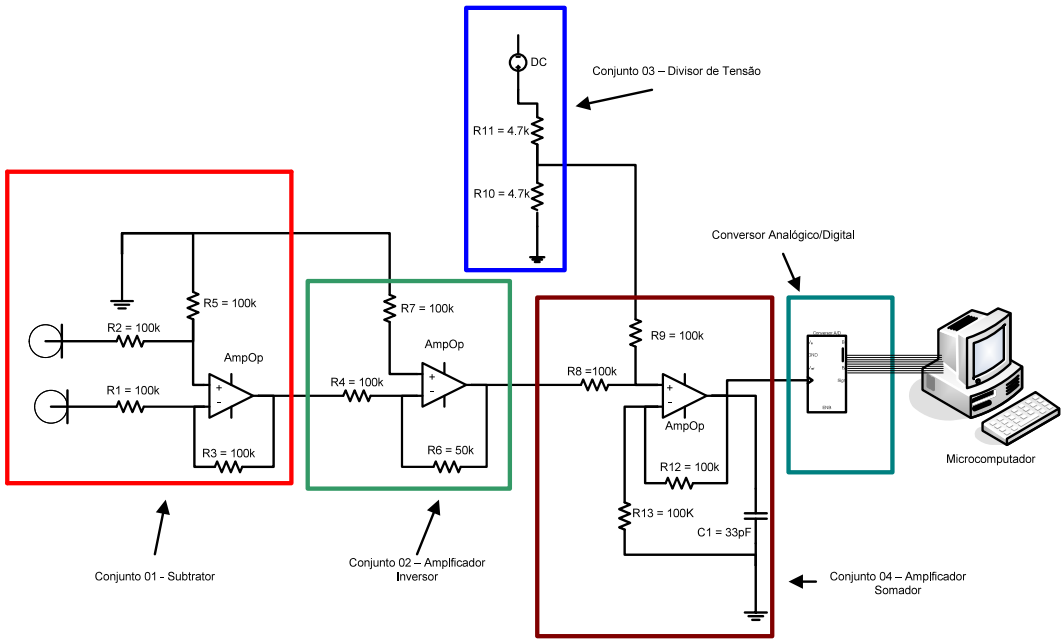

Figura B.1 – Esquema geral do circuito.

O circuito é composto por 2 microfones, amplificadores operacionais a FET, conversor analógico/digital, capacitores, resistores e fontes de tensão contínuas.

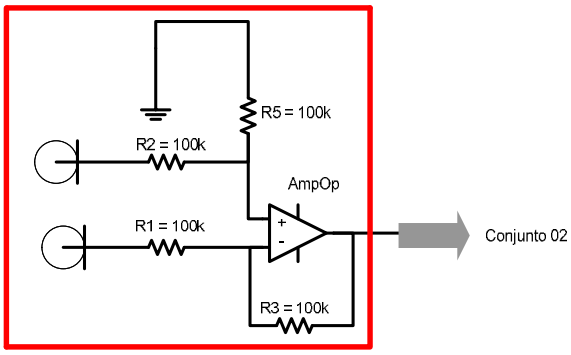

Figura B.2 – Conjunto 01: subtrator operacional.

O conjunto 01 (Figura B.2), subtrator, tem como alvo principal, “subtrair“ os sinais que serão captados pelos dois microfones. Estes sinais poderão assumir um valor máximo de 50mV, em virtude dos microfones que estavam sendo utilizados. O sinal de saída do amplificador subtrator assumirá valores positivos ou negativos, dependendo da posição relativa da fonte com relação aos microfones. A equação (B.1) nos mostra a fórmula do amplificador subtrator. [Almeida, 2004]

$$V_s = \frac{R_3}{R_1}(V_2 - V_1) \tag{B.1}$$

onde:

V_s = tensão de saída;

R_3 = resistor de realimentação;

R_1 = resistor de entrada;

V_1 = sinal de entrada 1, proveniente do microfone 01.

V_2 = sinal de entrada 2, proveniente do microfone 02.

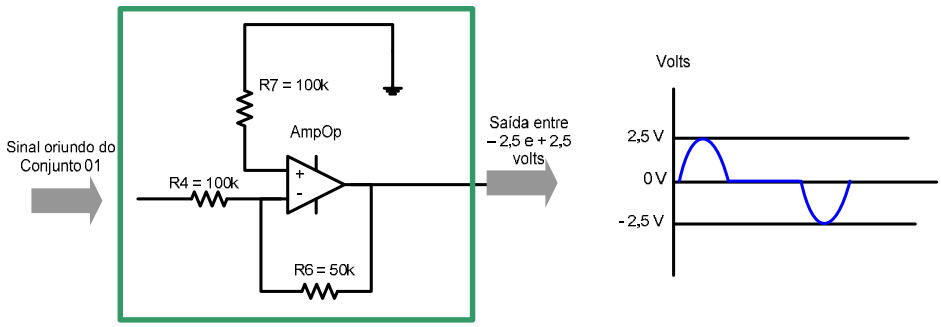

Figura B.3 – Conjunto 02: amplificador inversor.

O conjunto 02 (Figura B.3), um amplificador inversor realiza a amplificação do sinal, possibilitando um “ganho” no mesmo. Esse ganho deverá permitir uma tensão máxima de 2,5 Volts, pois o conjunto 03 (Figura B.4), divisor de tensão, tem como objetivo, fazer com que a saída máxima do circuito não ultrapasse 5 volts, uma vez que o conversor analógico/digital utilizado suporta uma tensão de entrada entre 0 e 5 volts.

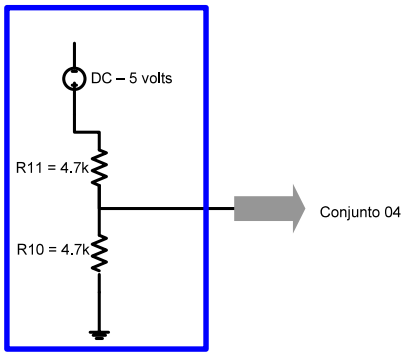

Figura B.4 – Conjunto 03: divisor de tensão.

A equação (B.2) mostra como é realizado o cálculo no amplificar inversor. [Boylestad, Nashelsky, 1999] [Almeida, 2004]

$$V_0 = -\frac{R_6}{R_4} V_1 \tag{B.2}$$

onde:

V_0 = tensão de saída;

V_1 = tensão de entrada;

R_4 = resistor de entrada;

R_6 = resistor de realimentação.

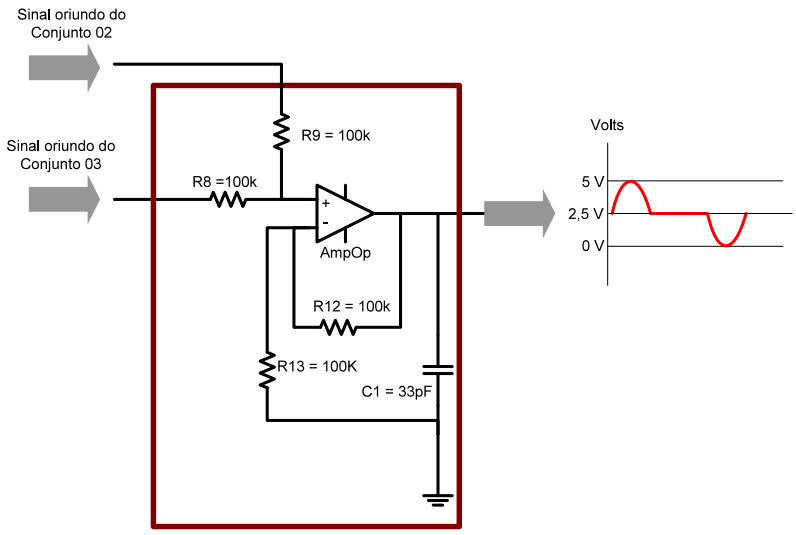

Figura B.5 – Conjunto 04: amplificador somador.

O conjunto 04 (Figura B.5), amplificador somador, realiza a soma dos sinais oriundos do divisor de tensão e do conjunto 02. A saída desse conjunto deve estar entre 0 e 5 volts, pois é essa a faixa de tensão que o conversor ADC0804 suporta para o seu perfeito funcionamento.

O conversor analógico/digital tinha como objetivo no projeto, converter os sinais analógicos em dados digitais e enviá-los ao PC via porta paralela.

Anexo C – Conversor Analógico/Digital

Para projetar o circuito descrito no Anexo B, fez-se necessário um estudo sobre o conversor analógico ADC 0804. A seguir será mostrado algumas características desse conversor.

Características

Um conversor Analógico/Digital é um circuito integrado que converte grandezas analógicas em dados binários (0 e 1).

Como um bom exemplo, temos a placa de som de um computador, que tem um conversor Analógico/Digital (ADC), de 16 ou mais bits, que tem a função de converter os sinais sonoros captados pelo transdutor (microfone) e os transforma em dígitos binários, que terão a possibilidade de serem armazenados em arquivos, para uso e manipulações futuras.

O conversor Analógico Digital ADC0804 é um Circuito Integrado da National Semicondutores [National] capaz de converter uma amostra analógica entre 0 e 5 Volts, em um valor binário de 8 bits.

Para entender a resolução do conversor, é necessário saber o valor máximo que a entrada suporta e o tamanho máximo da saída em bits. Como exemplo vamos fazer os cálculos para o ADC0804.

Cálculo da resolução do ADC0804:

$$resolução = \frac{valor\_analógico}{2^8} \tag{C.1}$$

$$resolução = \frac{5Volts}{256}$$

$$resolução = 0{,}0195Volts\_ou\_19{,}5mV.$$

Tabela C.1 – Conversão volts para binário no ADC0804.

Volts	*Decimal*	*Binário*
0,0195	1	00000001
0,0390	2	00000010
0,0585	3	00000011
0,0780	4	00000100
. . .	. . .	. . .
4,9920	255	11111111

Observando, a tabela acima, entende-se que para cada amostra convertida, através do ADC0804, sua saída poderá assumir valores entre 0 e 255. Isso devido ao fato que com 8 bits, pode-se obter 256 combinações diferentes. [Messias, 2004]

Com o Conversor Analógico ADC0804 pode-se converter amostras de uma vasta gama de eventos e grandezas físicas com o auxílio de sensores como: temperatura, velocidade, umidade, pressão, luminosidade, som, etc.

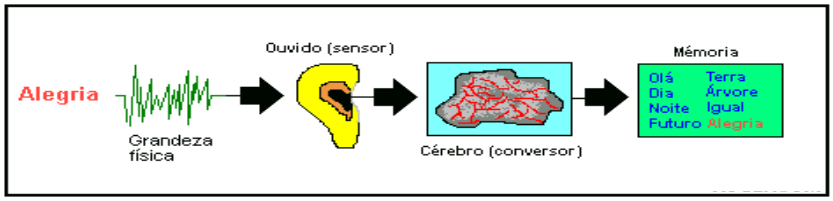

Figura C.1 – Conversor analógico humano.

Na Figura C.1, o ouvido humano (sensor) recebe os sinais sonoros do ambiente, por exemplo, a palavra Alegria, que é convertida, através do cérebro, em dados. Com esses dados convertidos e armazenados na memória podemos compará-los com uma vasta gama de informações, pré-armazenadas até o momento atual de nosso aprendizado. [Messias, 2004]

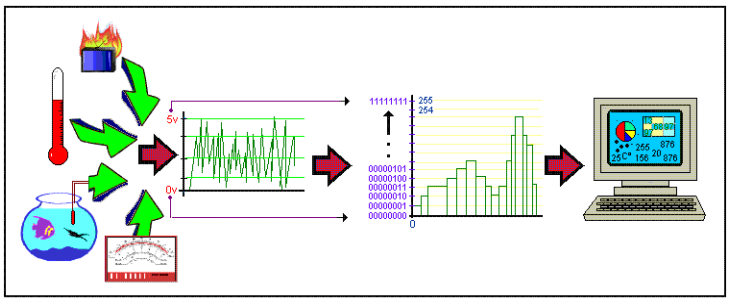

Figura C.2 – Conversor analógico digital de 8 bits, ADC0804.

Observa-se na Figura C.2 as grandezas físicas convertidas em tensões elétricas. Através de sensores essas tensões elétricas serão convertidas em dígitos binários e, por fim, os dados armazenados e interpretados pelo computador em formas de gráficos, números, etc. [Messias, 2004].

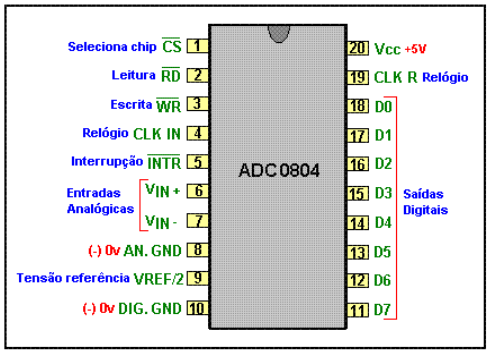

Figura C.3 – Características da pinagem do conversor ADC0804.

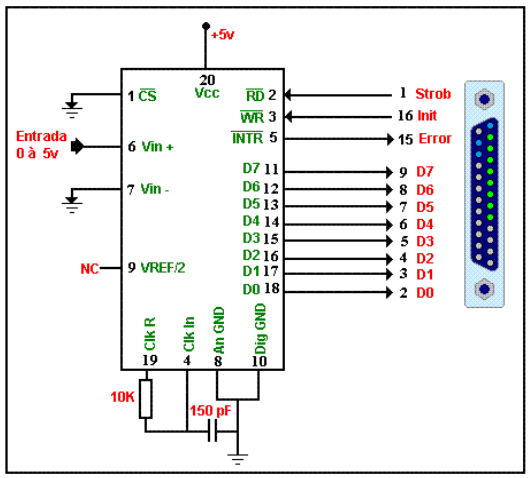

Figura C.4 – Configuração usada com o conversor ADC0804.

Pinos de controle do ADC0804

A seguir, estão descritos os principais pinos do ADC0804:

- Pino 1: CS – Aterrado;
- Pino 2: RD (Leitura) – Disponibiliza os dados convertidos nos pinos 11 a 18. Para iniciar a leitura envia um sinal 0 lógico;
- Pino 3: WR (Escrita) – Dá a ordem de início da conversão. Para iniciar a conversão envie um sinal 0 lógico;

- Pino 5: INTR (Interrupção) – Quando a conversão acaba, esse pino ficará ativo 0 (lógica inversa);
- Pino 6: $V_{IN}(+)$ – Entrada entre 0 e 5 volts;
- Pino 7: $V_{IN}(-)$ – Pino aterrado;
- Pino 8: GND;
- Pino 9: NC – Não conectado;
- Pinos 11 a 18 – Ligados à porta paralela;
- Pino 20: V_{CC} – Ligado a uma tensão continua de 5 volts. [Messias, 2004] [National]