Boletim Técnico da Escola Politécnica da USP

Departamento de Engenharia Eletrônica

BT/PEE/94-17

Redes Neurais Artificiais Aplicadas à Identificação de Pulsos Decádicos em Linhas Telefônicas

Antonio P. Timoszczuk
Euvaldo F. Cabral Jr.

São Paulo - 1994

Timoszczuk, Antonio P
Redes neurais artificiais aplicadas à identificação de pulsos decádicos em linhas telefônicas / A.P. Timoszczuk, E.F. Cabral Jr. -- São Paulo : EPUSP, 1994.
24p. -- (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia Eletrônica, BT/PEE/94-17)

1. Redes neurais (Inteligência artificial) 2. Sinais - Processamento digital 3. Telefonia I. Cabral Jr., Euvaldo F II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Eletrônica III. Título IV. Série

CDD 006.3
621.385
621.3822

REDES NEURAIS ARTIFICIAIS APLICADAS À IDENTIFICAÇÃO DE PULSOS DECÁDICOS EM LINHAS TELEFÔNICAS

Antonio P. Timoszczuk e Euvaldo F. Cabral Jr.
Laboratório de Comunicações e Sinais (LCS), DEE, EPUSP
Caixa Postal 61458 - CEP 05424-970 - São Paulo - S.P. - Brasil
e-mail: euvaldo@lcs.poli.usp.br

Resumo

Neste trabalho é estudada a aplicação de redes neurais artificiais para a classificação de pulsos telefônicos decádicos verdadeiros quando recebidos em meio a ruído. Este trabalho é uma continuidade de um trabalho anterior* para identificação de pulsos decádicos em linhas telefônicas, onde a partir de amostras no tempo, foram aplicadas técnicas de análise frequencial e "clustering" para separação dos pulsos válidos do ruído.

Este trabalho utiliza as mesmas amostras do trabalho anterior, e é dividido em duas partes. A primeira parte consiste em uma análise detalhada do sinal amostrado, sendo aplicados diversos métodos de pré-processamento com o objetivo de preparar as amostras para serem utilizadas como entradas de uma rede neural artificial adequadamente selecionada. Os métodos utilizados são mostrados, bem como os resultados obtidos. Na segunda parte, é apresentada a metodologia empregada para aplicação da rede neural artificial propriamente dita. Os resultados obtidos indicam a viabilidade do uso das redes neurais artificiais para execução desta tarefa de separação/classificação de pulsos válidos/ruído.

ARTIFICIAL NEURAL NETWORK TO IDENTIFY DECADIC PULSES ON TELEPHONE LINES

Antonio P. Timoszczuk and Euvaldo F. Cabral Jr.
Laboratório de Comunicações e Sinais (LCS), DEE, EPUSP
Caixa Postal 61458 - CEP 05424-970 - São Paulo - S.P. - Brasil
e-mail: euvaldo@lcs.poli.usp.br

Abstract

This work aims the application of an artificial neural network (ANN) paradigm to identify true decadic pulses from noise on telephone lines, as a follow on of a previous work* where the identification of true pulses was carried out using DFT with cluster analysis techniques.

At this point, a detailed analysis of the decadic pulses has been made, with the help of several pre-processing techniques, to prepare the samples to be used as inputs to a selected artificial neural network, following a description of the methodology used to train the network. The results obtained indicate that the use of artificial neural network to the identification task is promising.

---

* TIMOSZCZUK, A.P.; MATHIAS, M.A.; CABRAL JR., E.F. Identificação de pulsos decádicos em linhas telefônicas. **Boletim Técnico Escola Politécnica da USP**, BT/PEE/94-07, São Paulo, 1994.

Índice

1. Introdução

Com o objetivo de tornar a prestação do serviço ao cliente mais ágil e eficiente, o segmento de serviços oferecidos por parte de empresas tais como bancos, administradoras de cartão de crédito e afins experimentou nos últimos anos um vertiginoso crescimento do grau de informatização.

A utilização da rede telefônica como meio de comunicação entre o usuário e os serviços (automatizados) depende de dispositivos inteligentes que possuam interfaces homem-máquina capazes de suportar processos iterativos como, por exemplo, a solicitação por parte da máquina do número da conta corrente do usuário ou a senha de acesso do serviço desejado para fins de verificação da habilitação do indivíduo ao uso do serviço.

O meio mais simples que o usuário dispõe para acessar estes serviços é o próprio aparelho telefônico que, em sua versão mais comum, é do tipo decádico. Caso o aparelho telefônico seja do tipo multi-freqüencial, a possibilidade de ocorrer um problema de deteção correta dos dígitos por parte da máquina é pequena, pois os números teclados são convertidos em tons (pares de frequências) que por meio de filtros (digitais ou analógicos) são facilmente interpretáveis e a informação é recebida com grande confiabilidade.

A dificuldade ocorre quando o aparelho telefônico é do tipo decádico, que transmite os números discados/teclados por meio de aberturas e fechamentos de enlace ("loops" de corrente). Estes sinais são de difícil detecção pois podem ser facilmente confundidos com ruídos presentes no ambiente ou mesmo no meio de comunicação (rede telefônica, central de comutação), levando o dispositivo detector a uma interpretação errônea da informação fornecida pelo usuário.

Atualmente na rede telefônica brasileira, cerca de 98% dos aparelhos telefônicos são do tipo decádico, fato que motiva a busca de um método eficiente e confiável para o reconhecimento dos dígitos fornecidos pelo usuário através deste tipo de aparelho.

Neste trabalho são verificados diversos métodos de pré-processamento, com o objetivo de preparar as amostras para serem utilizadas como entradas para uma rede neural artificial adequadamente selecionada, sendo verificada a viabilidade de utilização das redes neurais artificiais para a identificação de pulsos decádicos verdadeiros.

2. Objetivos

O objetivo é, a partir das amostras de pulsos decádicos e ruído, efetuar diversos pré-processamentos para análise destas amostras. O resultado desta análise será então utilizado como entrada para o treinamento de uma rede neural artificial, que determinará quais amostras identificam os pulsos decádicos verdadeiros, diferenciando-os dos ruídos e dos pulsos causados pelo curto circuito e realimentação da cápsula microfônica.

3. Aquisição e pré-processamento

Para a análise das amostras, foram desenvolvidos programas utilizando MATLAB e linguagem de programação "C", sendo que as respectivas listagens são apresentadas no apêndice. A descrição da metodologia e equipamentos utilizados para

obtenção das amostras, bem como das características do sinal decádico, são descritos de forma resumida.

A partir deste ponto serão descritos os processamentos aplicados, com os respectivos resultados, buscando uma representação adequada para alimentação da rede neural artificial.

3.1 Sistema de aquisição

Para a aquisição dos dados foi montado o sistema apresentado na Figura 3.1.

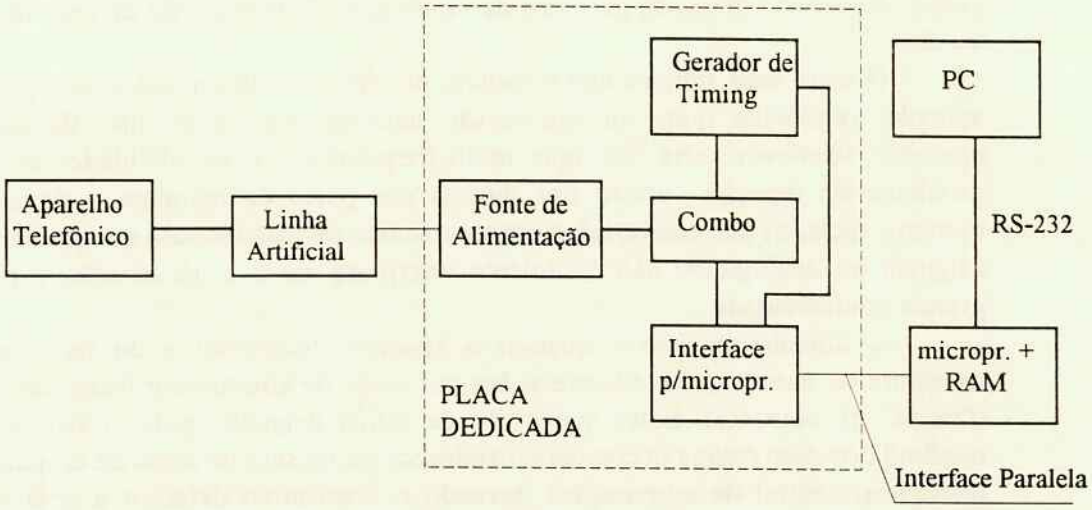


Figura 3.1 - Sistema de aquisição.

Os pulsos (dígitos) do aparelho telefônico, são transmitidos ao sistema de aquisição através de uma linha artificial (simulando 3Km de linha telefônica 0.4mm^2); o sinal analógico correspondente é então convertido para a forma digital segundo a codificação PCM-8bits a uma taxa de amostragem de 8KHz, obedecendo a Lei A de compressão.

O microprocessador dedicado, por meio de varredura temporal, efetua a leitura das amostras, que são armazenadas na memória RAM. Estas amostras são então enviadas para um computador tipo IBM-PC através de uma interface serial RS-232, onde são armazenadas em arquivos na forma binária.

3.2 Pulsos decádicos

Uma emissão de pulsos decádicos é feita através de um dispositivo (disco ou teclado) do telefone que provoca interrupções na corrente elétrica DC que circula na linha telefônica.

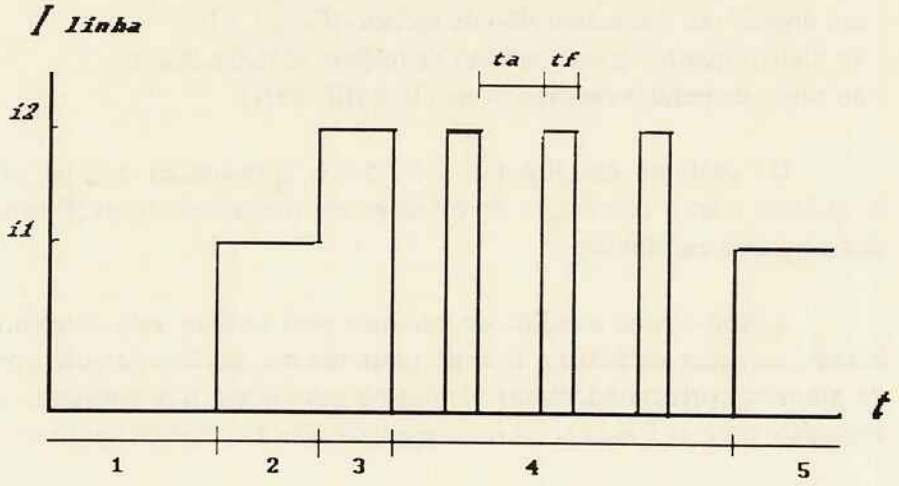
O sinal emitido pelo aparelho telefônico, ao ser pressionada uma tecla (ou acionado o disco), é um trem de pulsos que chaveará a linha um número de vezes igual ao dígito da tecla pressionada, sendo que o dígito zero corresponde a dez pulsos. Os pulsos têm freqüência nominal de 10Hz, relação de abertura/fechamento nominal de 2:1, e a tolerância para ambos os parâmetros, qualquer que seja a velocidade com que as teclas sejam pressionadas é a seguinte:

a) tempo de abertura: entre 58ms e 77ms;

b) tempo de fechamento: entre 28ms e 40ms.

A pausa inter digital é de no mínimo 700ms, e no máximo de 1.300ms, qualquer que seja a velocidade com que as teclas sejam pressionadas. A Figura 3.2, apresenta o evento dos pulsos decádicos ocorrendo na linha (de forma ideal). O período 1 representa o monofone em repouso (no gancho), e a corrente elétrica circulante na linha é nula. Quando o monofone é removido do gancho começa a circular na linha a corrente elétrica **i1**. Ao se girar o disco para a direita, a cápsula do microfone é curto-circuitada, o que diminui a impedância na linha e provoca portanto uma elevação na corrente circulante na linha, **i2**. Quando o disco é liberado ele tende a retornar a sua posição de repouso e, durante seu percurso de volta, contatos interruptores são abertos tantas vezes quanto o número do digito que se deseja transmitir.

No aparelho decádico de teclado (também chamado decádico eletrônico ou decádico compatível), o chaveamento da linha ocorre de maneira similar, porém no lugar de contatos mecânicos, são utilizadas chaves semicondutoras.



ta = tempo de abertura

tf = tempo de fechamento

Figura 3.2 - Pulsos decádicos.

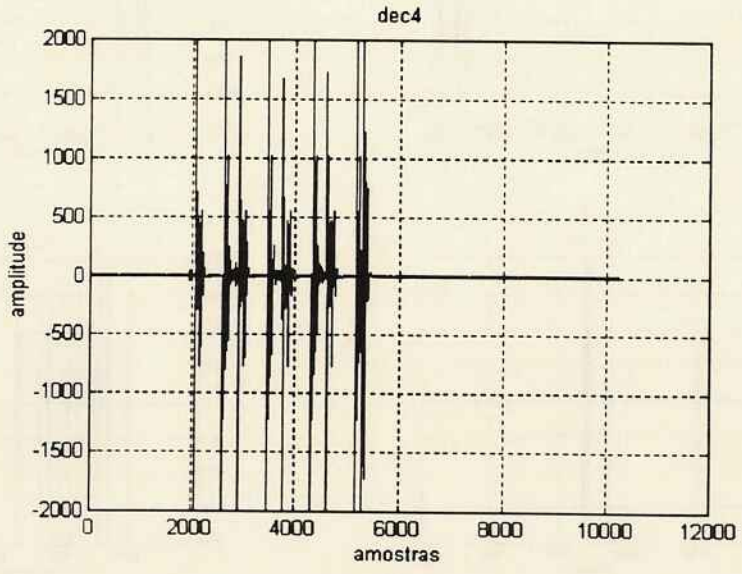


Figura 3.3 - Pulsos correspondentes ao dígito 4.

3.3 Análise do sinal

Foram usadas as amostras provenientes de um aparelho decádico de disco, de um aparelho decádico de teclas, amostras de ruídos na linha e amostras de ruído provocado por batidas no monofone.

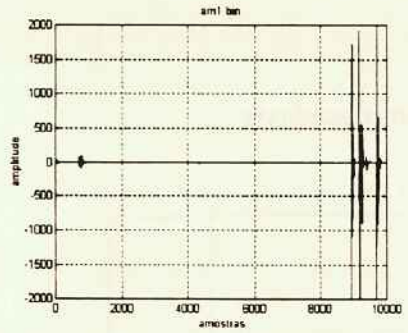
Escolheu-se arbitrariamente a amostra correspondente ao dígito "um" para aparelho de disco (AM1.BIN), tendo sido calculado o gráfico correspondente ao espectro em freqüência para o número total de amostras do arquivo de dados (10.000 amostras a 8KHz). Este cálculo foi efetuado através de uma DFT (implementada na rotina VERDFT.M: ver apêndice).

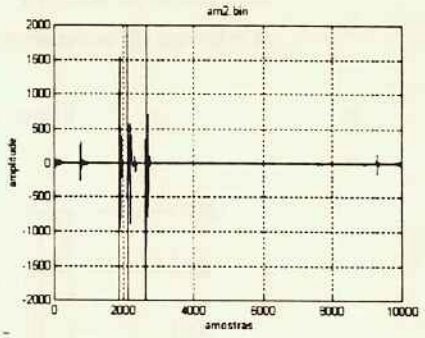
Com os dados obtidos através da DFT, foi calculado o gráfico de auto correlação do espectro em freqüência da amostra. Este cálculo teve por objetivo visualizar as regiões mais significativas do espectro em freqüência.

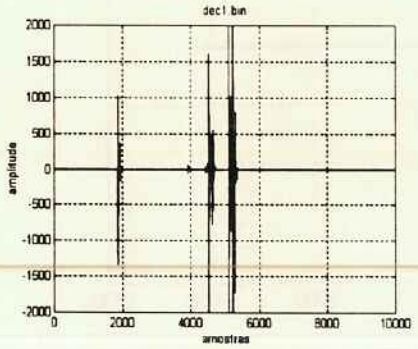
Este mesmo procedimento foi aplicado à outras amostras correspondentes a:
- um dígito "um" para aparelho de disco (AM2.BIN);
- um dígito "um" para aparelho de teclado (DEC1.BIN);
- ao dígito "quatro" para aparelho de teclado (MEC4.BIN);
- ao ruído de batidas no monofone (TAMBL.BIN).

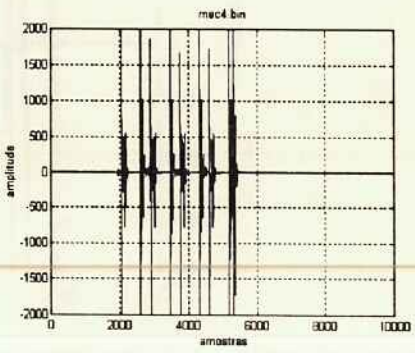
Os gráficos das Figuras 3.4-3.5-3.6 apresentam o sinal original, espectro em freqüência e auto correlação do espectro em freqüência respectivamente, para cada uma das amostras escolhidas.

Notou-se que a região de interesse para análise, está concentrada nas freqüências baixas, ao redor de 500Hz. É importante notar o gráfico de auto correlação do espectro da amostra correspondente as batidas no monofone, que apresenta uma auto correlação crescente para as freqüências mais altas, devido ao nível do ruído presente na amostra.









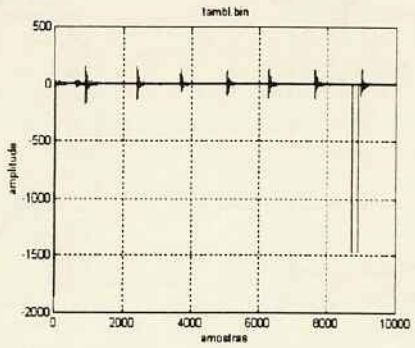


Figura 3.4 - Amostras no domínio do tempo.

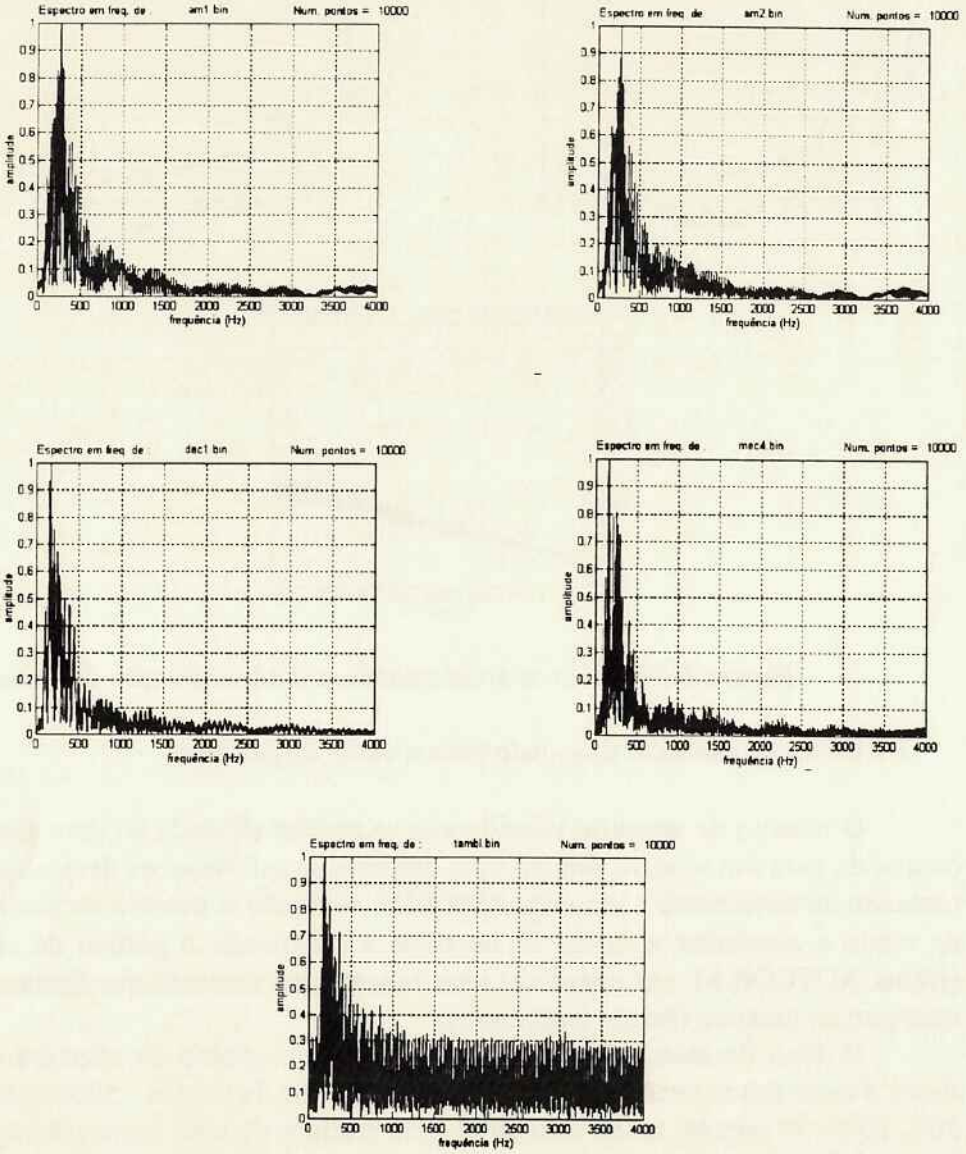


Figura 3.5 - Gráficos apresentando a DFT dos sinais.

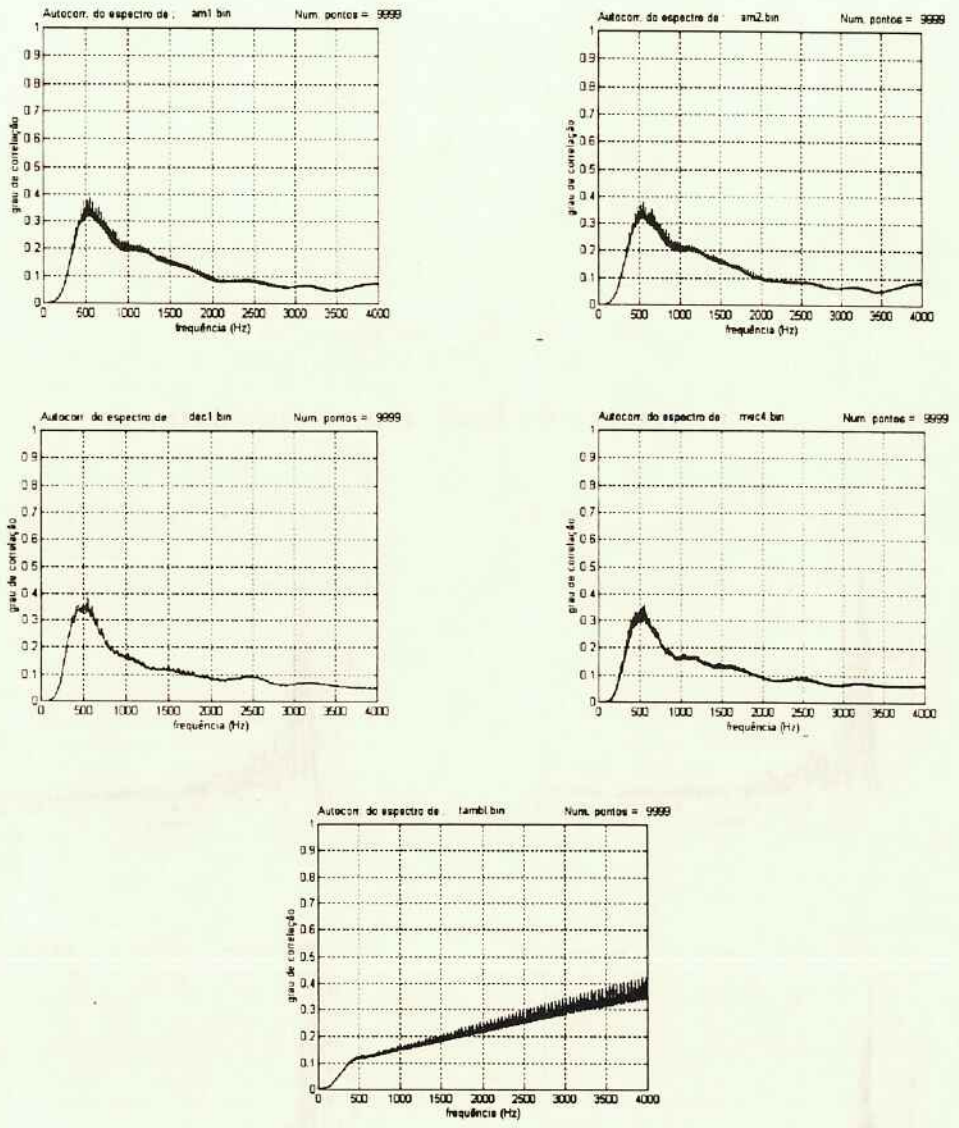


Figura 3.6 - Gráficos apresentando a auto correlação dos sinais.

3.4 Busca do tamanho adequado para o vetor de amostras

O número de amostras considerado na análise efetuada no item anterior, deve ser diminuída para um número prático, que preserve as informações desejadas. Isto foi feito tomando-se novamente a amostra AM1.BIN, variando o número de pontos da amostra de modo a acomodar a região de interesse e calculando o gráfico de auto correlação (rotina AUTCOR.M: ver apêndice) para buscar uma resposta que destaque a região de interesse ao máximo (baixas freqüências).

O foco de atenção foi concentrado sobre o evento de abertura de contato do disco. Foram feitos testes com os seguintes números de pontos : 5000, 2500, 1250, 625, 200, 100 e 50 pontos, tendo sido obtidos os gráficos de auto correlação apresentados na Figura 3.7.

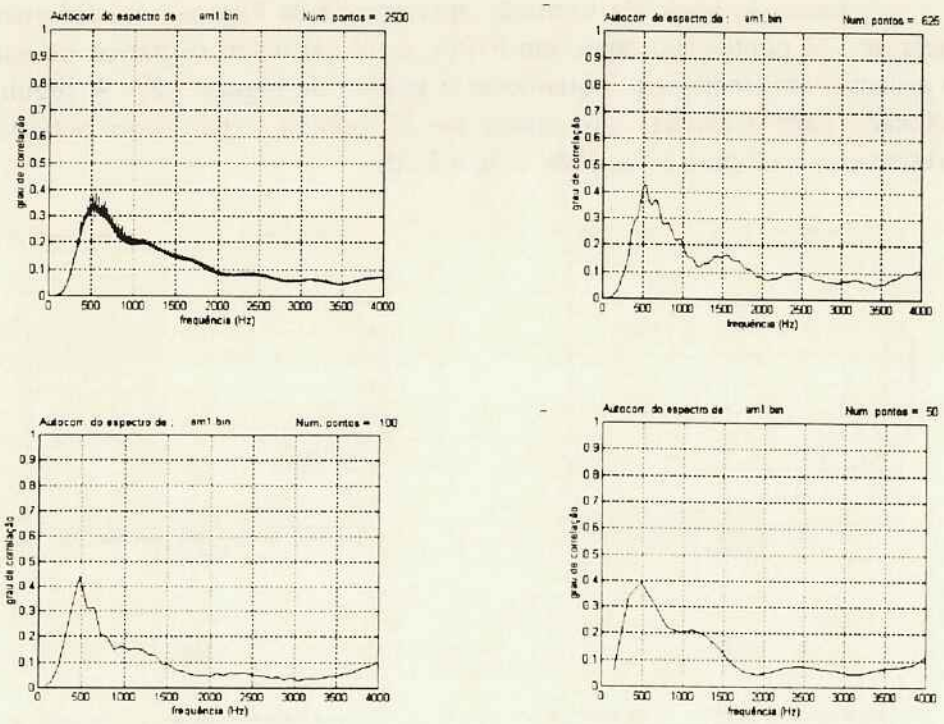
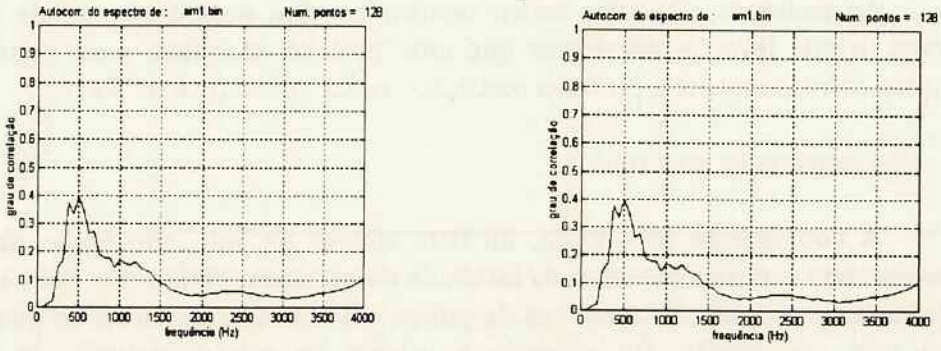


Figura 3.7 - Gráficos de auto correlação, variando o números de amostras.

O número de pontos que apresentou uma melhor resposta foi o correspondente a 100 pontos, onde a região de interesse aparece bem destacada das demais regiões do espectro. Como valor prático, foi escolhido 128 pontos para compor o vetor de amostras, o que irá facilitar sobremaneira a implementação prática.

3.5 Verificação da influência do deslocamento da janela de amostras no tempo

O fato de serem considerados 128 pontos para compor o vetor a ser analisado, equivale a uma janela retangular no tempo; esta janela foi deslocada sobre o pulso de abertura de contato da amostra AM1.BIN, cujo gráfico é apresentado na Figura 3.8a.



(a) (b)

Figura 3.8 - Pulso de abertura de contato.

A partir do início da transição apresentada na Figura 3.8a, foi posicionada uma janela de 128 pontos (iniciando em 9700), e foi calculado o gráfico de auto correlação do espectro em freqüência, obtendo-se o gráfico da Figura 3.8b. A seguir, a janela foi deslocada para a direita, em passos de 32 pontos, tendo sido obtidos os gráficos apresentados na Figura 3.9a, 3.9b, 3.9c e 3.9d.

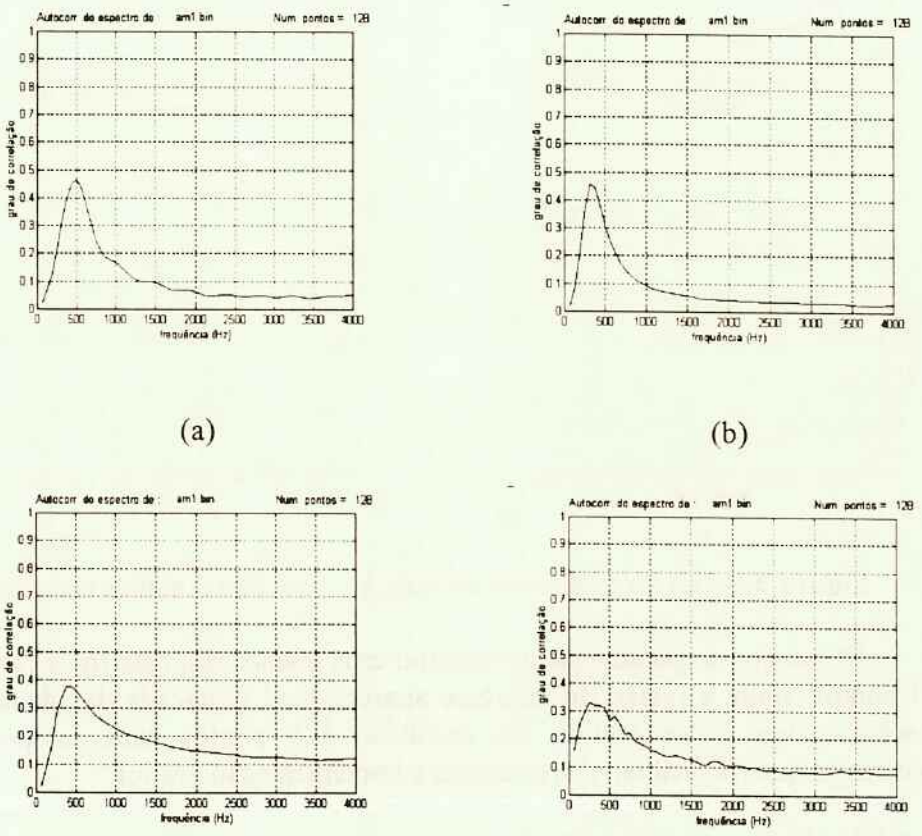


Figura 3.9 - Gráfico de auto correlação com deslocamento da janela.

As melhores respostas foram verificadas com deslocamentos de 32, 64 e 96 pontos, o que levou a considerar que uma posição adequada para a janela seria no instante correspondente a primeira transição brusca que ocorre no sinal.

3.6 Comparação com ruído

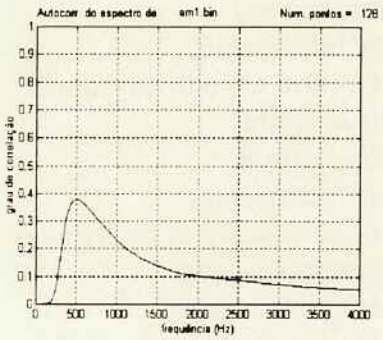
A comparação com ruído, foi feita através da auto correlação de cada vetor, considerando o posicionamento da janela de amostragem obtido nos itens anteriores, foi feita uma comparação de amostras de pulsos válidos com amostras de pulsos de ruído. Para esta comparação, foi utilizado o gráfico de auto correlação do espectro em frequência, tendo sido consideradas as seguintes amostras :

AM1.BIN - cápsula, abertura, fechamento;
AM2.BIN - cápsula, abertura, fechamento;
TAMBL.BIN - ruído 1 - pulso de batida no monofone;
- ruído 2 - pulso de batida no monofone;

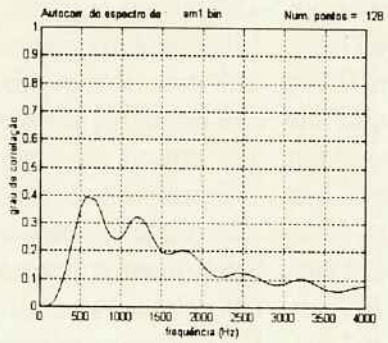
- ruído 3 - pulso de ruído na linha;
- ruído 4 - pulso de ruído na linha.

Tendo sido obtidos o gráficos do espectro em frequência e auto correlação correspondentes a estas amostras. A Figura 3.10, apresenta os gráficos de auto correlação correspondentes.

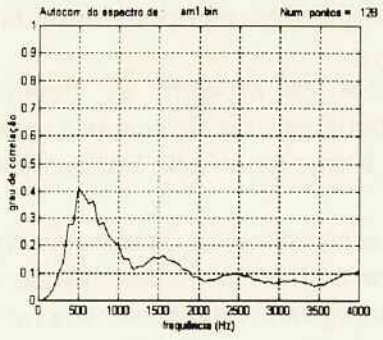
Através da comparação da auto correlação de cada amostra não foi possível conseguir distinguir satisfatoriamente o ruído dos pulsos válidos, uma vez que o gráfico de auto correlação do ruído provocado pela batida no monofone é muito semelhante àquela obtida de um pulso válido.



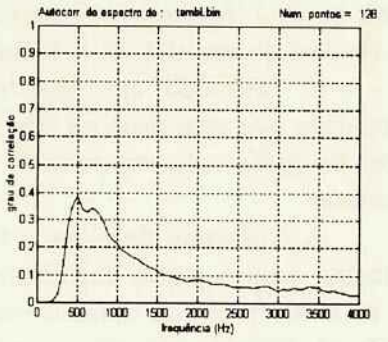
(a) cápsula



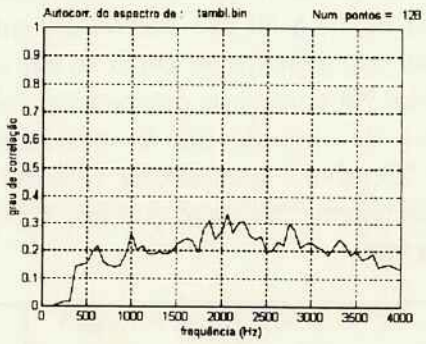
(b) abertura de contato



(c) fechamento



(d) batida no monofone



(e) ruído na linha

Figura 3.10 - Gráficos de auto correlação.

3.7 Análise de picos e vales

Foram separados os vetores de interesse para análise (128 pontos, obtidos através da rotina GRAVA1.M), compondo um conjunto de vetores das amostras da abertura de contato, fechamento de contato, curto/abertura de cápsula e ruídos, totalizando 85 vetores de amostras. A partir destes vetores, foram gerados os gráficos do espectro em freqüência de alguns destes vetores de amostras, para uma nova comparação. Foram analisados os seguintes eventos :

abep01.bin , 128 primeiros pontos do evento de abertura;
abes01.bin , 128 pontos seguintes do evento de abertura;
fech01.bin , 128 primeiros pontos do evento de fechamento;
caps01.bin , 128 primeiros pontos do evento de curto circuito da cápsula;
abep05.bin ,128 primeiros pontos do evento de abertura;
abes05.bin , 128 pontos seguintes do evento de abertura;
fech05.bin , 128 primeiros pontos do evento de fechamento;
caps05.bin , 128 primeiros pontos do evento de curto circuito da cápsula;
ruid01.bin , 128 primeiros pontos do evento de ruído na linha;
ruid03.bin , 128 primeiros pontos do evento de ruído na linha;
ruid07.bin , 128 primeiros pontos do evento de ruído do monofone;
ruid10.bin , 128 primeiros pontos do evento de ruído do monofone;

Nota-se que o evento de abertura possue dois conjuntos de 128 pontos, porém somente os 128 primeiros pontos contêm informações relevantes, desta forma o número de vetores de amostra foi reduzido para 55.

O resultado apresentado nos gráficos do espectro de freqüência, levou a conclusão que uma possível alternativa de identificação seria contar o número de picos e vales do gráfico do espectro em freqüência, comparando este número entre as diversas amostras.

A contagem de picos e vales foi feita comparando cada ponto de um vetor de amostra, com o ponto imediatamente anterior e posterior, determinando se é um ponto de máximo (pico) ou mínimo (vale). Esta contagem foi implementada (através da rotina PICVAL.M: vide apêndice), tendo sido aplicada a todas as 55 amostras consideradas. Os resultados obtido são apresentados na Tabela 3.1, onde são indicados os números de picos e vales para cada amostra.

É importante notar que as 15 primeiras amostras são correspondentes a abertura de contato, as amostras de 16 a 30 são correspondentes a fechamento de contato, as amostras 31 a 44 são correspondentes ao curto da cápsula e as amostras de 45 a 55 são correspondentes ao ruído. Na contagens dos picos e vales, foi introduzida uma variável, correspondente ao limiar de variação que é considerado para determinar um pico ou vale, equivalendo a um "threshold" que permite a consideração de grandes variações.

Através desta contagem conclui-se que não é possível determinar com segurança se um pulso é válido ou não.

Am.	T = 0.010		T = 0.005		T = 0.015		T = 0.020		T = 0.025	
	Picos	Vales	Picos	Vales	Picos	Vales	Picos	Vales	Picos	Vales
1	3	4	5	6	2	3	2	3	2	2
2	2	2	4	4	2	0	2	0	2	0
3	3	5	4	6	3	3	3	2	1	2
4	5	5	7	7	4	4	3	3	1	2

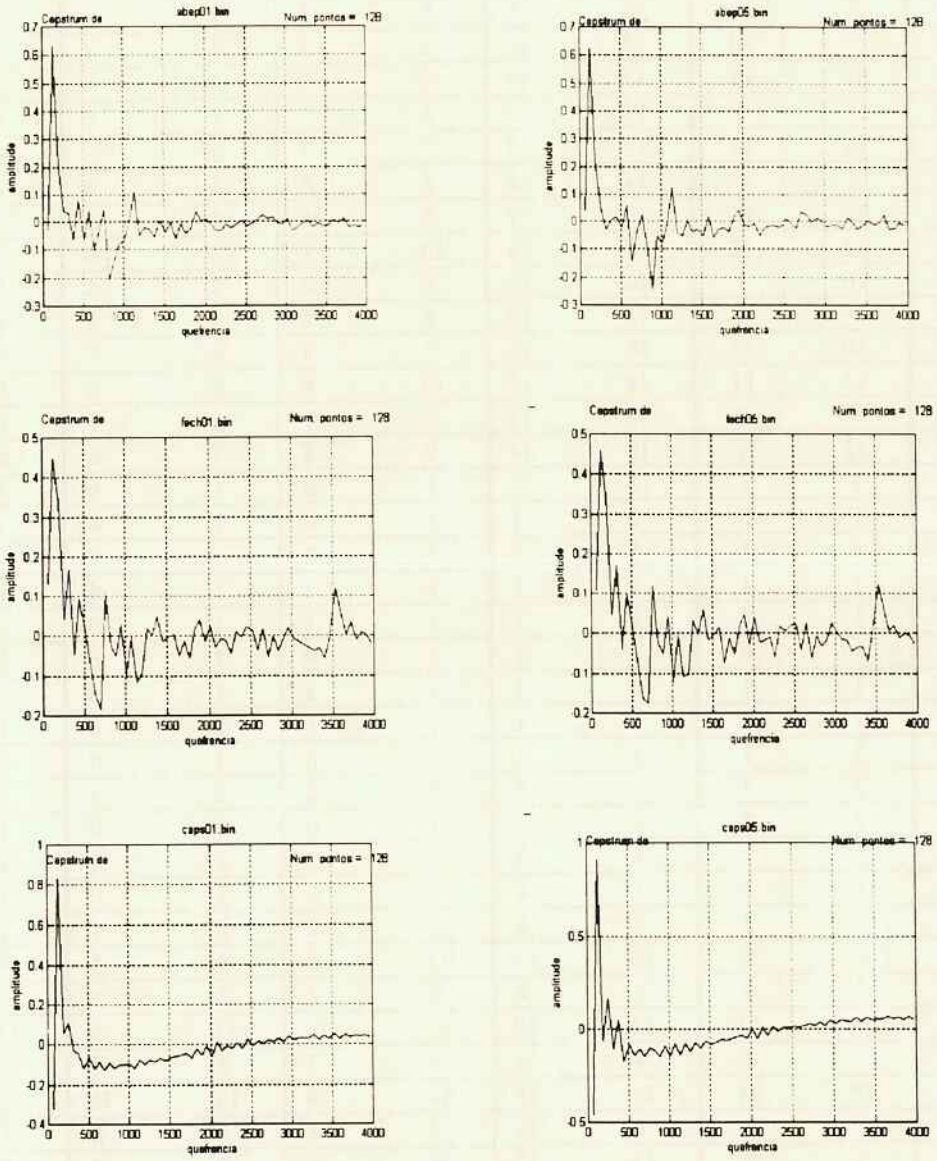
Am.	T = 0.010		T = 0.005		T = 0.015		T = 0.020		T = 0.025	
	Picos	Vales	Picos	Vales	Picos	Vales	Picos	Vales	Picos	Vales
8	5	5	6	6	4	3	2	3	2	2
9	2	3	3	6	2	3	2	2	1	1
10	2	4	4	5	1	3	1	1	1	1
11	3	5	3	5	1	2	1	1	1	1
12	4	4	5	6	3	2	1	1	1	1
13	11	11	13	13	8	7	6	4	4	3
14	2	6	4	6	2	4	2	3	2	1
15	3	3	5	4	3	3	3	2	2	1
16	7	10	12	13	5	5	4	5	4	4
17	7	11	16	18	5	5	4	4	4	4
18	7	10	15	16	5	5	4	4	4	4
19	10	11	15	14	6	8	4	4	3	4
20	7	9	17	17	4	5	4	5	4	4
21	12	14	17	17	7	7	6	6	5	4
22	7	6	14	14	5	5	5	4	4	4
23	12	14	19	19	8	10	5	5	4	4
24	11	12	13	14	10	8	7	6	6	6
25	10	5	10	10	8	4	7	4	7	4
26	9	6	14	10	5	5	5	3	4	3
27	10	12	16	16	7	9	6	7	6	6
28	12	11	15	13	10	8	8	6	6	5
29	9	9	12	13	7	7	6	5	6	5
30	12	12	12	14	9	8	7	6	6	6
31	1	1	1	1	1	0	1	0	1	0
32	1	0	1	1	1	0	1	0	1	0
33	1	1	1	1	1	0	1	0	1	0
34	1	1	1	1	1	1	1	0	1	0
35	1	1	1	1	1	1	1	0	1	0
36	1	0	1	1	1	0	1	0	1	0
37	1	1	1	1	1	1	1	0	1	0
38	1	1	1	1	1	1	1	0	1	0
39	1	1	1	1	1	0	1	0	1	0
40	2	3	2	4	2	2	2	2	2	1
41	1	0	1	1	1	0	1	0	1	0
42	2	2	4	4	2	2	2	2	2	1
43	1	1	1	1	1	0	1	0	1	0
44	2	2	3	4	2	2	2	2	2	1
45	13	9	16	15	12	9	9	8	9	8
46	2	2	7	4	2	2	2	2	2	2
47	11	13	14	17	11	11	11	9	11	9
48	3	4	4	5	2	3	2	2	1	2
49	17	14	21	15	14	10	14	10	13	9
50	7	5	9	10	5	4	5	2	5	2
51	9	9	13	12	8	6	8	4	8	3
52	9	10	15	13	7	9	7	9	5	9
53	13	11	15	12	12	9	9	8	7	6
54	10	9	16	14	7	6	7	4	5	4
55	12	12	18	18	7	7	5	4	4	3

Tabela 3.1 - Resultados da análise de picos e vales.

3.8 Análise por Cepstrum real

Aplicou-se às mesmas amostras citadas no item 3.7, a análise cepstral através da rotina VERCEP.M, obtendo os gráficos de amplitude cepstral, que foram normalizados pelo máximo de cada vetor. Esta técnica foi aplicada visando observar a existência de componentes convoluídas no vetor de amostra.

Alguns resultados obtidos são apresentados na Figura 3.11, tendo sido observado que a aplicação da técnica cepstral levou a uma normalização do espectro, não acrescentando nenhuma informação ao sinal.



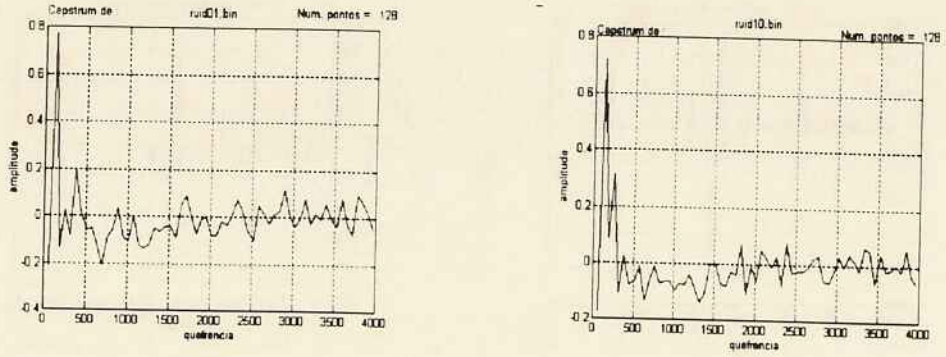


Figura 3.11 - Resultados das análises aplicando Cepstrum real.

3.9 Análise utilizando "Discrete Wavelet Transform" (DWT)

Foi aplicado às 55 amostras uma DWT (Discrete Wavelet Transform), através da rotina VERWLT.M, obtendo os gráficos de amplitude dos coeficientes ("mother functions" e "wavelets") para cada amostra. A transformação foi aplicada para a classe de funções de Daubenchis, utilizando quatro coeficientes, usualmente chamada de DAUB4. Esta transformação foi efetuada através do procedimento chamado de "pyramidal algorithm". Alguns gráficos dos resultados obtidos são apresentados nas Figura 3.12 - 3.13 - 3.14 - 3.15. A partir destes gráficos pode ser verificado que esta transformação permite distinguir perfeitamente o ruído dos pulsos válidos, embora não permita uma classificação dos pulsos válidos.

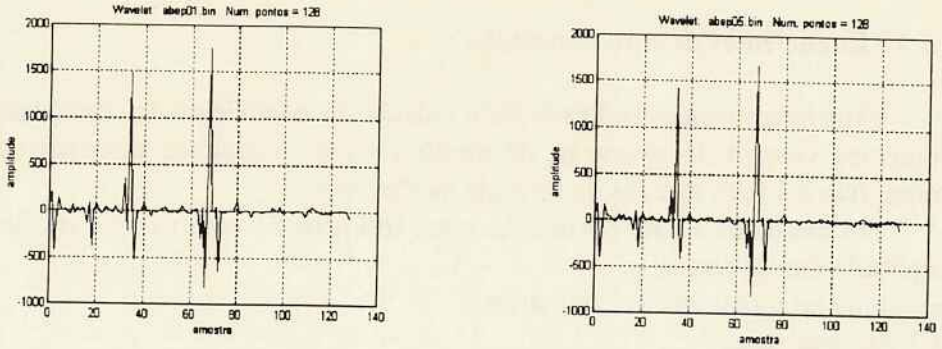


Figura 3.12 - DWT para abertura de contato.

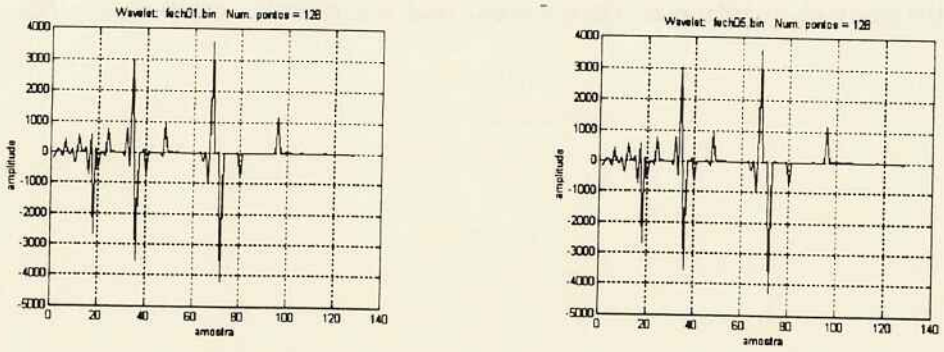


Figura 3.13 - DWT para fechamento de contato.

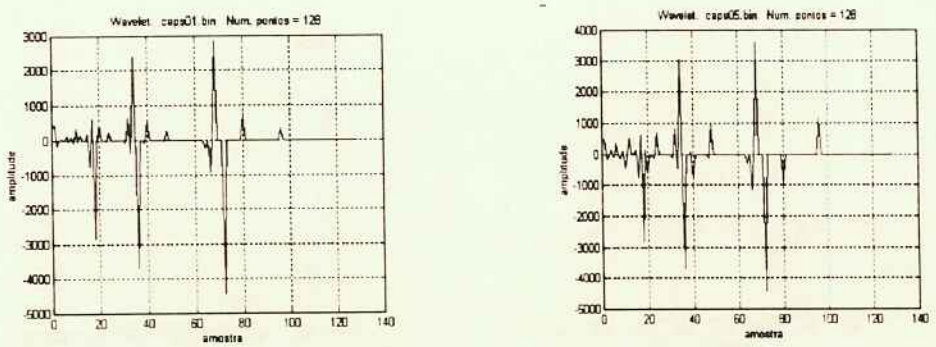


Figura 3.14 - DWT para cápsula.

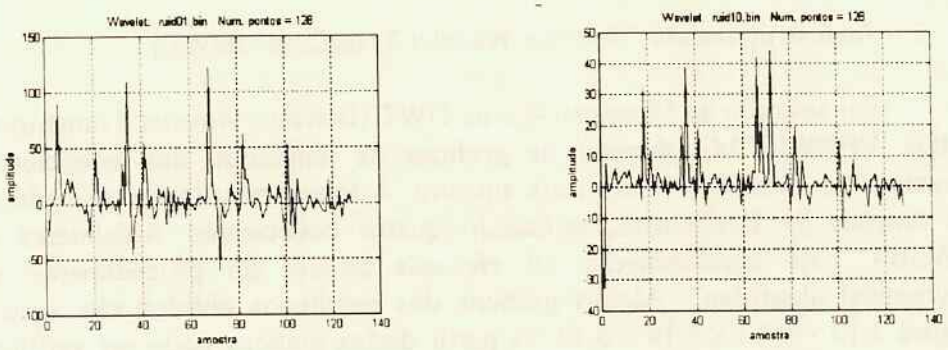


Figura 3.15 - DWT para ruído, esquerda ruído de linha e a direita ruído do monofone.

3.10 Coeficientes de auto correlação

Uma outra técnica utilizada foi o cálculo do coeficiente de correlação linear entre os diversos vetores de amostras, de modo a medir o grau de associatividade entre os vetores. Isto foi feito através da fórmula de Pearson.

Foi calculada a auto correlação entre todos os 55 vetores, nos seguintes casos:

- amplitude das amostras;
- amplitude ao quadrado das amostras;
- FFT das amostras;
- Cepstrum real das amostras;
- Wavelet das amostras.

Os resultados obtidos, são apresentados nas Figuras 3.16-3.17-3.18-3.19-3.20; nestes gráficos quanto mais clara a área, mais correlatados estão os vetores.

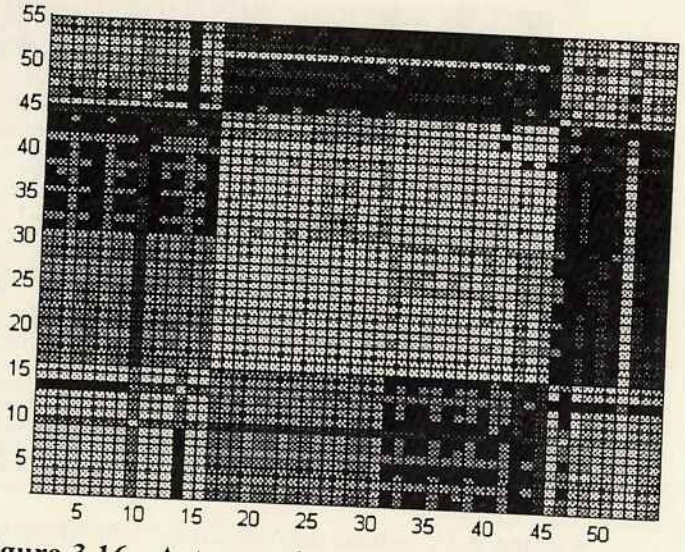


Figura 3.16 - Auto correlação da amplitude das amostras.

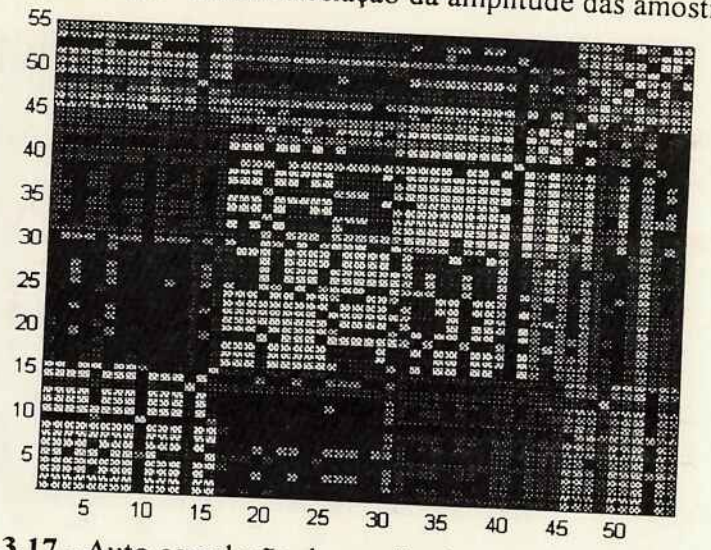


Figura 3.17 - Auto correlação da amplitude ao quadrado das amostras.

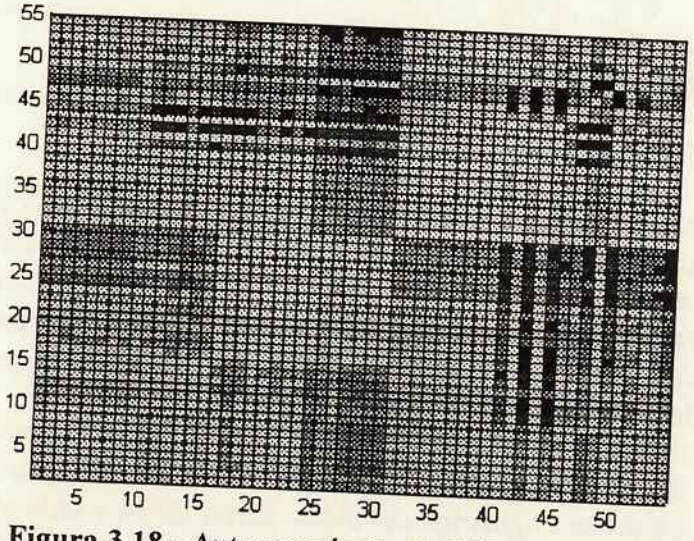


Figura 3.18 - Auto correlação da FFT das amostras.

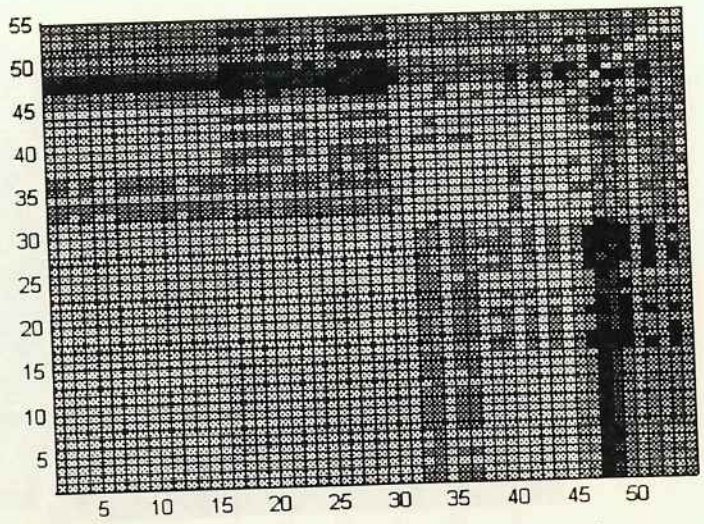


Figura 3.19 - Auto correlação do Cepstrum das amostras.

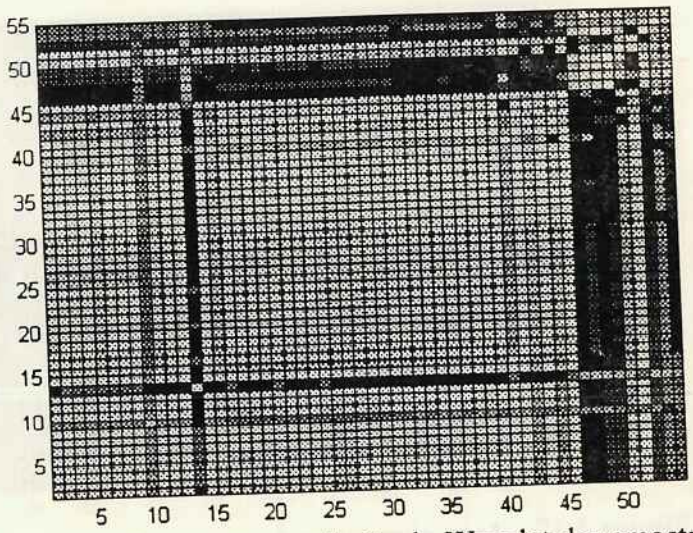


Figura 3.20 - Auto correlação do Wavelet das amostras.

4. Aplicação da rede neural artificial

4.1 Escolha da rede

A rede neural artificial escolhida para esta aplicação é constituída por um único elemento processador (neurônio) simples. A razão desta escolha se deve a simplicidade e eficiência para implementação deste tipo de rede em "hardware" convencional. Na configuração utilizada neste trabalho a rede possue uma única saída, que indicará se o pulso analisado é um pulso válido ou se é um pulso de ruído, e um número de entradas adequado ao tipo de pré-processamento aplicado.

Após análise dos gráficos de auto correlação, optou-se pelo pré-processamento através de FFT e Wavelets, por indicarem uma distinção bem clara entre os pulsos válidos e aqueles originários de ruídos. Desta forma a rede possuirá 64 entradas no caso do pré-processamento através de FFT e 128 entradas no caso de pré-processamento através de Wavelets.

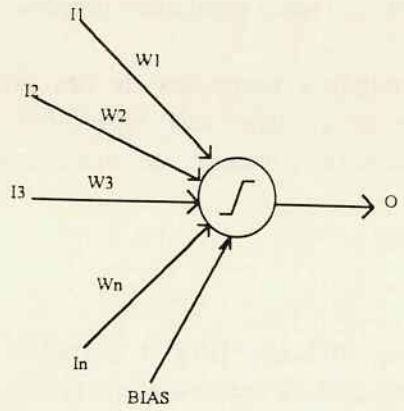


Figura 4.1 - Arquitetura da rede adotada.

A Figura 4.1, apresenta a arquitetura da rede adotada. Nesta figura I1..In corresponde as entradas da rede com os respectivos pesos W1..Wn; a entrada BIAS corresponde a um polarizador adicional para adequar a entrada da rede à função de transferência escolhida e, a saída única corresponde a O.

4.2 Preparação das amostras para a rede

A partir das amostras existentes, foram separados 55 vetores de 128 amostras cada segundo os critérios descritos anteriormente. Destes 55 vetores, os 15 primeiros vetores são correspondentes a abertura de contato, os vetores de 16 a 30 são correspondentes a fechamento de contato, os vetores de 31 a 44 são correspondentes ao curto da cápsula e os vetores de 45 a 55 são correspondentes ao ruído.

O segundo passo consistiu em calcular os vetores correspondentes a FFT e Wavelets de cada um dos vetores de entrada resultando em novos conjuntos de 55 vetores para cada caso.

O terceiro passo foi efetuar a seleção dos vetores a serem usadas nos conjuntos de treinamento e teste, de modo a ter-se um conjunto balanceado de vetores, ou seja, para maior eficiência no treinamento foram escolhidos vetores pertencentes a cada uma das classes de pulsos (abertura, fechamento, cápsula e ruído) distribuídas uniformemente dentro das respectivas classes. Para tal foi calculada a matriz de distâncias entre os diversos vetores em cada uma das classes existentes, utilizando o critério de distância Euclidiana. A partir destas matrizes foram escolhidos os vetores que se apresentaram equidistantes dentro de cada classe.

O quarto passo consistiu em montar os conjuntos de treinamento e teste, utilizando os vetores escolhidos no passo anterior. Os conjuntos de treinamento e teste, foram montados na forma de matrizes onde cada linha corresponde a um vetor com 64 ou 128 elementos. No caso da matriz de treinamento, esta possue uma última coluna adicional que corresponde ao valor desejado para a saída ("target"). Os conjuntos de treinamento e teste, foram montados para FFT e Wavelets na seguinte ordem:

- dois vetores correspondendo aos pulsos de abertura de contato;
- dois vetores correspondendo aos pulsos de fechamento de contato;
- dois vetores correspondendo aos pulsos gerados pela cápsula; e
- três vetores correspondendo aos pulsos gerados por ruído.

Um outro conjunto de teste foi montado com os 55 vetores para cada um dos pré-processamentos, com objetivo de um teste final. Estas matrizes foram montadas através dos programas GERAMTX1.M e GERAMTX.M (vide apêndice).

4.3 Treinamento da rede e resultados obtidos

A rede montada é composta de um único elemento processador, tendo sido utilizada a função de transferência Sigmoidal implementada através de uma forma genérica com coeficientes ajustáveis, segundo a expressão a seguir.

$$O = \frac{a}{1+e^{(C\cdot I+\theta)}} - d$$

O algoritmo utilizado para a correção dos erros é o Adaline ou correção quadrática, cuja expressão é apresentada a seguir:

$$E_j = \sqrt{\sum_n \left(O_{jn} - t_{jn}\right)^2}$$

onde "n" representa o número de passos aguardados para corrigir o erro, "j" corresponde ao passo de treinamento e, "t" corresponde a saída desejada.

Os programas PERCEF.M e PERCETF.M (vide apêndice) implementam a rede para treinamento e teste utilizando os vetores resultantes da FFT; e os programas PERCEW.M e PERCETW.M implementam a rede para treinamento e teste utilizando os vetores resultantes dos Wavelets.

Para o treinamento a rede teve seus pesos iniciados com valores pequenos e pseudo-randômicos, em seguida foram apresentados os vetores de treinamento gravados na matriz de treinamento. A escolha da ordem de apresentação dos vetores, obedeceu a um processo de "sorteio" pseudo-randômico. Cada apresentação de vetor de entrada foi denominada passo ou "step", sendo que a cada passo foi calculada a saída e o respectivo erro. Em caso de erro grande, os valores dos pesos foram corrigidos segundo a expressão abaixo:

$$\Delta W_{i,j} = I_i \cdot E_j \cdot \alpha$$
$$W_{i,j}^{new} = W_{i,j}^{old} + \Delta W_{i,j}$$

onde $\Delta W_{i,j}$ corresponde ao quanto é corrigido o peso da entrada "i" após o passo "j", E_j corresponde ao erro apresentado na saída após o passo "j" e α corresponde ao "learning rate".

4.3.1 Utilização dos vetores FFT

O conjunto de treinamento resultante da FFT foi apresentado à rede, sendo que as saídas desejadas correspondem a 45 para pulsos válidos e -45 para pulsos de ruído. Foram ajustados os parâmetros da rede até obter-se o resultado apresentado na Figura 4.1.

Na Figura 4.1a é apresentado o erro durante o treinamento, tendo o mesmo sido considerado satisfatório após 300 passos de treinamento. A Figura 4.1b apresenta o resultado da rede para o conjunto de teste, sendo que a rede classificou corretamente todas as 9 amostras.

Em seguida foi feito um teste para verificar a robustez da rede, que consistiu em inverter os conjuntos de treino e teste. Após esta inversão a rede foi novamente treinada,

tendo convergido em 300 passos; a Figura 4.1c apresenta o resultado da rede para o novo conjunto de teste, sendo que a rede novamente classificou corretamente o conjunto de teste.

Como último teste foi apresentada a matriz de teste contendo todos os vetores. A Figura 4.1d apresenta os resultados, tendo sido notado que a rede diferenciou os vetores de ruído dos vetores válidos.

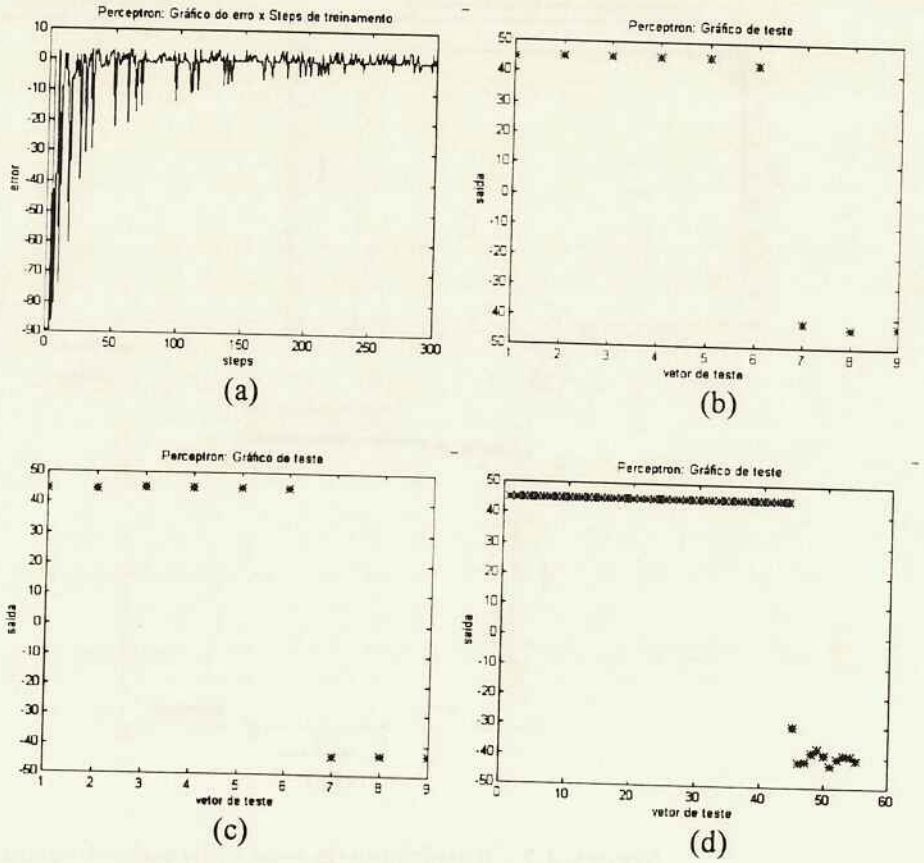


Figura 4.1 - Resultados da rede utilizando FFT.

4.3.2 Utilização dos Wavelets

Da mesma foram o conjunto de treinamento resultante da FFT foi apresentado à rede, sendo que as saídas desejadas correspondem a 45 para pulsos válidos e -45 para pulsos de ruído. Foram ajustados os parâmetros da rede até obter-se o resultado apresentado na Figura 4.2.

Na Figura 4.2a .apresenta o erro durante o treinamento, tendo o mesmo sido considerado satisfatório após 500 passos de treinamento. A Figura 4.2b apresenta o resultado da rede para o conjunto de teste, sendo que a rede classificou corretamente todas as 9 amostras. O teste de robustez foi efetuado conforme descrito no item 4.3.1, porém não foi obtido 100% de classificações corretas.

Como último teste foi apresentada a matriz de teste contendo todos os vetores. A Figura 4.2c apresenta os resultados, tendo sido notado que a rede diferenciou os vetores de ruído dos vetores válidos, porém classificou de forma errada alguns pulsos válidos.

Foi efetuado um novo treinamento da rede desta vez com um número maior de passos (2000) um conjunto maior de teste. Desta vez foram usados 25 vetores de

treinamento segundo o mesmo critério já estabelecido, sendo os mesmos distribuídos da seguinte forma:

- seis vetores correspondendo aos pulsos de abertura de contato;
- seis vetores correspondendo aos pulsos de fechamento de contato;
- seis vetores correspondendo aos pulsos gerados pela cápsula; e
- sete vetores correspondendo aos pulsos gerados por ruído.

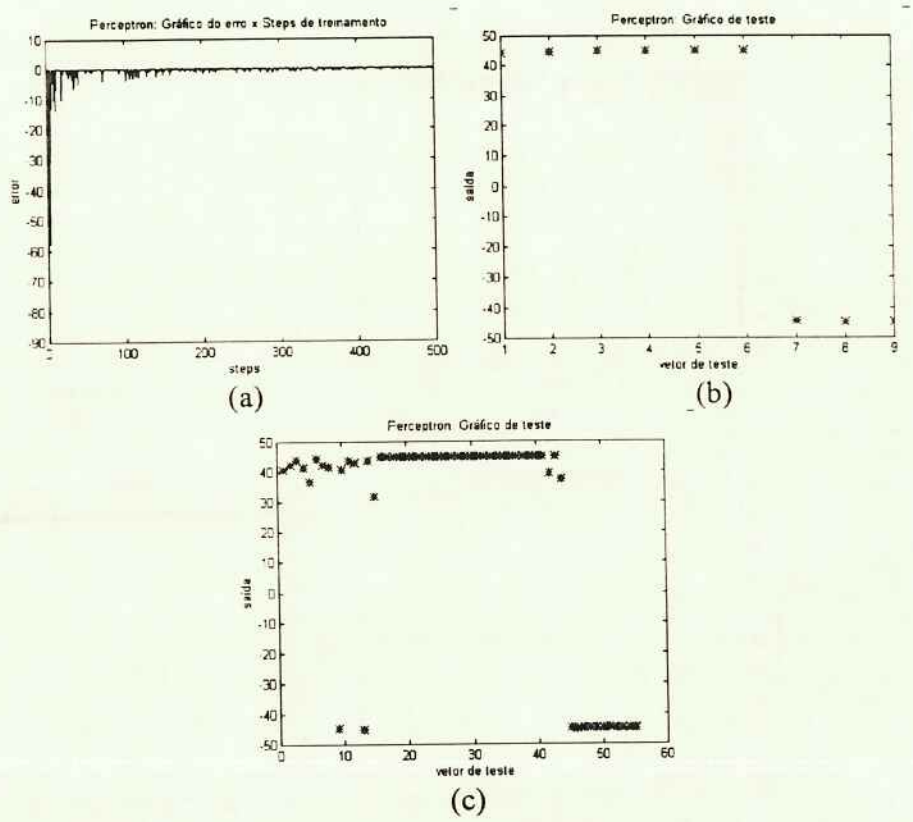
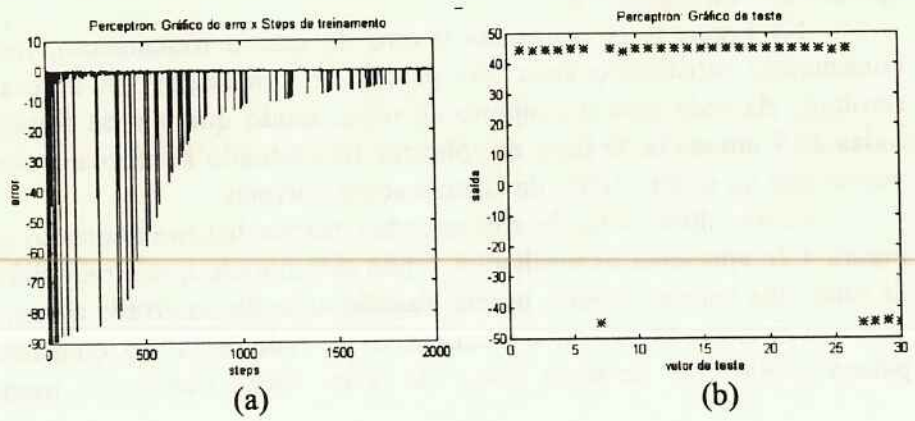
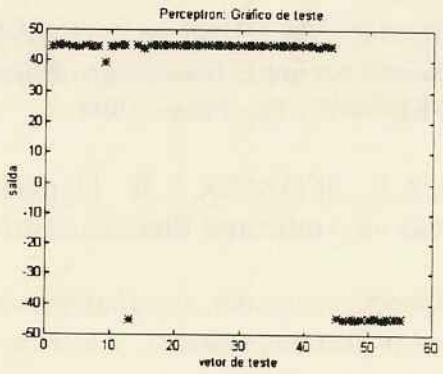


Figura 4.2 - Resultados da rede utilizando Wavelets.

O conjunto de teste, por sua vez, foi montado com os 30 vetores restantes. O resultado obtido é apresentado na Figura 4.3. Pode ser observado através da Figura 4.3a que após 2000 passos o erro de saída foi diminuído para um mínimo; a Figura 4.3b apresenta o resultado da classificação do conjunto de teste. A rede distinguiu o ruído, porém também classificou de forma errada um pulso válido.

Finalmente foi apresentado todo o conjunto, tendo sido observada uma classificação errada, conforme apresentado na Figura 4.3c.





(c)

Figura 4.3 - Resultados da rede utilizando Wavelets.

5. Conclusão

Neste trabalho foi apresentada uma comparação do resultado de diversos tipos de pré-processamento para a identificação dos pulsos. Foi observado que para este tipo de sinal as análises através da FFT e Wavelets mostraram-se mais adequadas para a identificação dos pulsos, sugerindo inclusive a possibilidade de classifica-los de acordo com o tipo de fenômeno que gerou o pulso.

O objetivo de identificar os pulsos válidos dos pulsos causados por ruído, foi atingido através da aplicação da FFT como pré-processamento para uma rede neural extremamente simples, comprovando a aplicabilidade desta técnica de processamento ao problema.

Os resultados obtidos através da rede, utilizando os Wavelets como pré-processamento, vem ratificar os resultados do mapa de "clusters" obtido no trabalho anterior, no qual foi mostrado que os "clusters" de alguns tipos de pulso sobrepõe-se a outros, dificultando o reconhecimento. Foi observado também que através da ampliação do conjunto de treinamento e aumento do número de passos de treinamento, pode ser melhorada a capacidade de classificação da rede.

Foi buscado neste trabalho também uma classificação dos pulsos, através do método de "piece wise" aplicado à distribuição de valores de "targets" do perceptron, para distinguir os diversos tipos de pulso existentes. Este objetivo não foi atingido pois os vetores de entrada estão muito próximos entre si, não permitindo o correto mapeamento através da função de transferência neural utilizada. Para este tipo de classificação é sugerida a utilização de redes mais complexas.

A identificação, da forma como foi efetuada neste trabalho, deve ser complementada com técnicas de janelas definidas de acordo com a temporização esperada para as sequências de pulsos, de forma a permitir sua correta identificação da seqüência temporal.

Como continuidade para este trabalho, será buscada uma topologia de rede neural artificial mais adequada para a classificação destes pulsos de acordo com a causa do pulso, também deverá ser estudada a aplicação de redes neurais artificiais que levam em consideração o aspecto temporal dos sinais, permitindo identificar toda a seqüência de eventos correspondente aos pulsos decádicos.

6. Bibliografia

[1] TIMOSZCZUK, A.P.; MATHIAS, M.A.; CABRAL JR., E.F. Identificação de pulsos decádicos em linhas telefônicas. **Boletim Técnico Escola Politécnica da USP**, BT/PEE/94-07, São Paulo, 1994.

[2] OPPENHEIM, A.V.; SCHAFER, R.W. Homomorphic analysis of speech. **IEEE Transactions on Audio and Electroacustics**, v. AU-16, n. 2, Jun. 1968.

[3] BURR, D.J. Speech recognition experiments with perceptrons. In: **Neural information processing systems.** American Institute of Physics, 1988. p.144-53.

[4] PRESS, W.H.; et al. **Numerical recipes in C**. New York, Cambridge University Press, 1994.

[5] **MATLAB user's guide**. The Mathworks, Inc., Natick, Mass., Feb. 1993.

Apêndice

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: AUTCOR.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para calcular a autocorrelação de um vetor (conjunto de amostras) resultante de uma FFT.
% Os resultados são apresentados na forma gráfica.
%
% O usuário deve fornecer :
% - o nome dos arquivo binário a ser utilizado <nome.bin>;
% - o início do intervalo a ser usado no cálculo, correspondendo a <amostra inicial>, cujo valor minimo
é 1;
% - o final do intervalo a ser usado no cálculo, correspondendo a <amostra final>, cujo valor máximo é
% igual ao número total de amostras.
% O número de amostras total deve ser conhecido de forma a permitir uma primeira visualização, sendo
% que o intervalo de interesse pode ser determinado pela visualização das amostras (vide programa de
% visualização das amostras).
%
% Frequência de amostragem de 8000 Hz.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Efetua a leitura do primeiro arquivo a ser tratado

fnamedat = input('Nome do arquivo : ','s');
fid= fopen(fnamedat,'r');
a=fread(fid,Inf,'short');
fclose(fid);

% Efetua o cálculo da fft de cada arquivo segundo o intervalo fornecido.
% Observações : b(x) é um vetor com indice=1 ... tamanho;
%               c(x) toma o módulo;
%               d(x) é o vetor c(x) normalizado pelo máximo;
%               e(x) é o vetor de autocorrelação de d(x);
%               f(x) é o vetor e(x) normalizado pelo máximo.

inicio = input('Forneça o inicio do intervalo : ');
final = input('Forneça o final do intervalo : ');
tamanho = final-inicio;
x = inicio:1:final;
b = a(x);
c = abs(fft(b));
d = c/(max(c));
e = xcorr(d);
f = e/(max(e));

% Traça o gráfico da autocorrelação.
% Observação : xn corresponde a escala normalizada em frequência.

x = 1:1:tamanho/2;
xn = (4000/(tamanho/2))*x;
plot(xn,f(x));

% Completa o gráfico com informações úteis.
```

```
grid
axis ([0 4000 0 1]);
title(fnamedat)
text (0,1.05,'Autocorr. do espectro de :')
text (3000,1.05,'Num. pontos =')
text (4000,1.05,int2str(tamanho))
xlabel('frequência (Hz)')
ylabel('grau de correlação')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: GERAMTX.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Este programa gera as matrizes de treinamento e de teste para serem usadas com uma rede neural
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Gera a matriz de treinamento

NVTRAIN = input('Forneça o número de vetores de treino: ');
NPOINT = input('Forneça o número de pontos do vetor de treino: ');
NOUT = input('Forneça o número de saídas por vetor de treino: ');

for n=1:NVTRAIN
  VTRAIN = input('Forneça o nome do vetor de treino: ');

  % Efetua a leitura de um vetor

  fid = fopen(int2str(VTRAIN),'r')';
  a=fread(fid,Inf,'short');
  fclose(fid);

  % Efetua a leitura dos valores para o target

  for m=1:NOUT
    OUT = input('Forneça o valor da saída: ');
    TARGET(n,m) = OUT;
  end

  % Monta os valores do vetor de treino e do target na matriz de treinamento

  for i=1:NPOINT % Monta o vetor de treino em trainmtx
    trainmtx(n,i) = a(i);
  end
end

  % Normaliza a matriz pelo máximo global dos vetores de treino

  % Determina os índices "i" e "j", que apontam para o maior valor dentro da matriz

  maximo1 = max (trainmtx);
  maximod = max (maximo1);
  for i = 1:NVTRAIN;
    for j = 1:NPOINT;
```

```
        dif = maximod - trainmtx(i,j);
        if dif == 0
          maxmtx = trainmtx(i,j);
        end
      end
    end

    trainmtx = trainmtx/maxmtx;

  % Monta os valores do vetor de target na matriz de treino

  for n=1:NVTRAIN
    for j=1:NOUT  % Monta o vetor target em trainmtx
      trainmtx(n,NPOINT+j) = TARGET(n,j);
    end
  end

  % Salva a matriz de treinamento no formato .MAT

  save trainmtx trainmtx

  % Gera a matriz de teste

  NVTEST = input('Forneça o número de vetores de teste: ');
  NPOINT = input('Forneça o número de pontos do vetor de teste: ');
  for n=1:NVTEST
    VTEST = input('Forneça o nome do vetor de teste: ');

    % Efetua a leitura de um vetor

    fid = fopen(int2str(VTEST),'r')';
    a=fread(fid,Inf,'short');
    fclose(fid);

    % Monta os valores do vetor de teste na matriz de teste

    for i=1:NPOINT % Monta o vetor de teste em testmtx
      testmtx(n,i) = a(i);
    end
  end

    % Normaliza a matriz pelo máximo global dos vetores de treino

    % Determina os índices "i" e "j", que apontam para o maior valor dentro da matriz

    maximo1 = max (testmtx);
    maximod = max (maximo1);
    for i = 1:NVTEST;
      for j = 1:NPOINT;
        dif = maximod - testmtx(i,j);
        if dif == 0
          maxmtx = testmtx(i,j);
        end
      end
    end
    testmtx = testmtx/maxmtx;

% Salva a matriz de teste no formato .MAT
```

```
save testmtx testmtx

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: GERAMTX1.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Este programa gera as matrizes de treinamento e de teste
% para serem usadas com uma rede neural
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Gera a matriz de treinamento

clear NVTRAIN;
clear NPOINT;
clear NOUT;
NVTRAIN = input('Forneça o número de vetores de treino: ');
NPOINT = input('Forneça o número de pontos do vetor de treino: ');
NOUT = input('Forneça o número de saídas por vetor de treino: ');

for n=1:NVTRAIN
  VTRAIN = input('Forneça o nome do vetor de treino: ');

  % Efetua a leitura de um vetor

  fid = fopen(int2str(VTRAIN),'r')';
  a=fread(fid,Inf,'short');
  fclose(fid);

  % Efetua o cálculo da fft de cada arquivo segundo o intervalo fornecido.
  % Observações :
  %      a é o vetor com indice=1 até L;
  %      b é o vetor linha correspondente ao vetor a coluna;
  %      c é o vetor resultante da FFT, considerado em módulo;
  %      e é o vetor FFT tomado pela metade (devido a simetria).

  % Converte a(vetor coluna) em b(vetor linha)

  x = 1:1:NPOINT;
    b = a(x);

  % Calcula o vetor correspondente a FFT

  c = abs(fft(b));
  x = 1:1:NPOINT/2;
    e = c(x);

  % Efetua a leitura dos valores para o target

  for m=1:NOUT
    OUT = input('Forneça o valor da saída: ');
    TARGET(n,m) = OUT;
  end

  % Monta os valores do vetor de treino na matriz de treinamento
```

```
  for i=1:NPOINT/2 % Monta o vetor de treino em trainmtx
    trainmtx(n,i) = e(i);
  end
end

  % Normaliza a matriz pelo máximo global dos vetores de treino

  % Determina os índices "i" e "j", que apontam para o maior valor dentro da matriz

  maximo1 = max (trainmtx);
  maximod = max (maximo1);
  for i = 1:NVTRAIN;
    for j = 1:NPOINT/2;
      dif = maximod - trainmtx(i,j);
      if dif == 0
         maxmtx = trainmtx(i,j);
      end
    end
  end
  trainmtx = trainmtx/maxmtx;

% Monta os valores do vetor de target na matriz de teste

for n=1:NVTRAIN
  for j=1:NOUT  % Monta o vetor target em trainmtx
    trainmtx(n,(NPOINT/2)+j) = TARGET(n,j);
  end
end

% Salva a matriz de treinamento no formato .MAT

save trainmtx trainmtx

% Gera a matriz de teste

clear NVTEST;
clear NPOINT;
NVTEST = input('Forneça o número de vetores de teste: ');
NPOINT = input('Forneça o número de pontos do vetor de teste: ');
for n=1:NVTEST
  VTEST = input('Forneça o nome do vetor de teste: ');

  % Efetua a leitura de um vetor

  fid = fopen(int2str(VTEST),'r')';
  a=fread(fid,Inf,'short');
  fclose(fid);

  % Efetua o cálculo da fft de cada arquivo segundo o intervalo fornecido.
  % Observações :
  %      a é o vetor com indice=1 até L;
  %      b é o vetor linha correspondente ao vetor a coluna;
  %      c é o vetor resultante da FFT, considerado em módulo;
  %      e é o vetor FFT tomado pela metade (devido a simetria).

  % Converte a(vetor coluna) em b(vetor linha)

  x = 1:1:NPOINT;
```

```
    b = a(x);

  % Calcula o vetor correspondente a FFT

  c = abs(fft(b));
  x = 1:1:NPOINT/2;
    e = c(x);

  % Monta os valores do vetor de teste e do target na matriz de teste

  for i=1:NPOINT/2 % Monta o vetor de teste em testmtx
    testmtx(n,i) = e(i);
  end
end

  % Normaliza a matriz pelo máximo global dos vetores de treino

  % Determina os índices "i" e "j", que apontam para o maior valor dentro da matriz

  maximo1 = max (testmtx);
  maximod = max (maximo1);
  for i = 1:NVTRAIN;
    for j = 1:NPOINT/2;
      dif = maximod - testmtx(i,j);
      if dif == 0
         maxmtx = testmtx(i,j);
      end
    end
  end
  testmtx = testmtx/maxmtx;

% Salva a matriz de teste no formato .MAT

save testmtx testmtx


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: GERAWVLT.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para calcular a "Discrete Wavelet Transform" dos vetores das amostras consideradas. É
% utilizada a classe de Daubechies, a quatro coeficientes (C0, C1, C2 e c3), DAUB4.
% Valores dos coeficientes:
%                          C0 = 0.4829629131445341
%                          C1 = 0.8365163037378079
%                          C2 = 0.2241438680420134
%                          C3 = -0.1294095225512604
%
%
% O usuário deve fornecer :
% - o nome do arquivo contendo o vetor a ser usado no cálculo;
% - o tamanho do vetor de amostras;
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Solicita ao usuário o nome e tamanho do vetor de amostras;

for N=1:55  % Número de vetores a ser utilizado (num.=nome).
  N=N
  L = 128; %input('Forneça o tamanho do vetor de amostras : ');

  % Efetua a leitura do vetor (arquivo) a ser usado no cálculo, montando o vetor v(n)

  % Efetua a leitura de um vetor

  fid = fopen(int2str(N),'r')';
  v=fread(fid,Inf,'short');
  fclose(fid);

  % Cálculo dos novos vetores transformados, a partir dos vetores v(n).

  C0 = 0.4829629131445341;
  C1 = 0.8365163037378079;
  C2 = 0.2241438680420134;
  C3 = -0.1294095225512604;
  NN = L;
  NH = L/2;
  while (NN >= 4)
    i = 1;
    for j = 1:2:L-3
      w(i) = C0*v(j)+C1*v(j+1)+C2*v(j+2)+C3*v(j+3);
      w(i+NH) = C3*v(j)-C2*v(j+1)+C1*v(j+2)-C0*v(j+3);
      i = i+1;
    end
    w(i) = C0*v(L-1)+C1*v(L)+C2*v(1)+C3*v(2);
    w(i+NH) = C3*v(L-1)-C2*v(L)+C1*v(1)-C0*v(2);
    v=w;
    NH = NH/2;
    NN = NN/2;
  end

  % Salva o vetor v(n) como arquivo binário, para uso posterior

  fname = [int2str(N)];
  fwriteid = fopen(fname,'w');
  count = fwrite(fwriteid,v,'short');
  status = fclose(fwriteid);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: GRAVA1.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para criar os arquivos "*.bin", que contém os dados a serem analisados.
% O usuário deve fornecer :
% - o nome do arquivo binário a ser utilizado <nome.bin>;
% - o início do intervalo a ser utilizado <amostra inicial>, cujo valor minimo é 1;
% - o final do intervalo a ser utilizado <amostra final>, cujo valor máximo é igual ao número total de
% amostras.
% - o nome do arquivo destino dos dados a serem gravados <nome.bin>.
```

```
% O número de amostras total deve ser conhecido.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Efetua a leitura do arquivo a ser tratado, criando o vetor b.

fnamedat = input('Nome do arquivo de entrada : ','s');
frid= fopen(fnamedat,'r');
b=fread(frid,Inf,'short');
fclose(frid);

% Busca amplitudes maiores do que 100, para montar vetor de dados com 128 bytes de comprimento
"c".

y=129;
npulso=0;
for x=10:1:10000;
  if b(x)>100              % se amplitude > 100, considera como pulso
    if x-y>128;   % verifica se andou o suficiente para não pegar o mesmo pulso
      y=x-10
          npulso=npulso+1;

                                % gera o novo vetor

      for z=1:1:128;
        c(z)=b(y+z);
       end
                                % Grava o vetor "ann" em disco

        fwid = fopen(int2str(npulso),'w');
        count = fwrite(fwid,c,'short');
        status = fclose(fwid);
     end
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: MPICVAL.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%           Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para calcular o número de picos e vales em cada uma das amostras consideradas. O
conjunto
% de amostras, é resultado de uma FFT.
%
% O nome dos arquivos binários a serem utilizados <nome.bin>, deve ser 1.bin, 2.bin até N.bin (onde N
é
% a quantidade de amostras);
%
% O usuário deve fornecer :
% - o tamanho dos vetores de amostras, a serem usados no cálculo;
% - o número de amostras a ser usado no cálculo;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Solicita ao usuário o número de vetores de amostras (arquivos);
```

```
N = input('Forneça o número de vetores a ser usado : ');
L = input('Forneça o tamanho do vetor de amostras : ');

% Efetua a leitura dos vetores (arquivos) a serem usados no cálculo, montando a matriz de entrada
% (M(N,L)) com N linhas e L colunas.
% Cada coluna corresponde a uma variável temporal(ou frequencial) x(1), x(2)...x(L), e cada linha
% corresponde a um vetor de amostras (conjunto de variáveis de 1 a 128).

for n=1:1:N;

 % Efetua a leitura de um vetor para ser incorporado à matriz M

 fid = fopen(int2str(n),'r')';
 a=fread(fid,Inf,'short');
 fclose(fid);

 % Efetua o cálculo da fft de cada arquivo segundo o intervalo
 % fornecido.
 % Observações : a é o vetor com indice=1 até L;
 %          b é o vetor linha correspondente ao vetor a coluna;
 %          c é o vetor resultante da FFT, considerado em módulo;
 %          d é o vetor "c" normalizado pelo seu máximo;
 %           e é o vetor FFT tomado pela metade (devido a simetria).

 % Converte a(vetor coluna) em b(vetor linha)

 x = 1:1:L;
 b(x) = a(x);

 % Calcula o vetor correspondente a FFT

 c = abs(fft(b));
 d = c/(max(c));
 x = 1:1:L/2;
  e = d(x);

 % Incorpora o vetor gerado "c" como uma linha adicional à matriz M

 M = [M; e];
end

% Cálculo do número de picos e vales para cada amostra.
% Cada amostra corresponde a uma linha da matriz "M".

% Seta limite

T = 0.025;
for n=1:1:N;
 C(1,n)=0;
 C(2,n)=0;
 for l=2:1:(L/2)-1;

  % Verifica picos

  if (M(n,l) - M(n,l-1)) > T;
   if (M(n,l) - M(n,l+1)) > T;
    C(1,n) = C(1,n)+1;
     else
     end
```

```
% Verifica vales

elseif (M(n,l-1) - M(n,l)) > T;
   if (M(n,l+1) - M(n,l)) > T;
      C(2,n) = C(2,n)+1;
   else
   end
  else
  end
 end
end

% Salva a matriz "C" como arquivo texto, para posterior impressão

fid = fopen('picval.txt','w');
fprintf(fid,'T = %1.4f\n\n\r',T);
fprintf(fid,'Picos   Vales\n\n\r');
fprintf(fid,'%3.0f    %3.0f\n\r',C);
status = fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: MXCOR.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para calcular a matriz dos coeficientes de correlação entre as amostras consideradas.
% O conjunto de amostras, é % resultado de uma FFT.
%
% O nome dos arquivos binários a serem utilizados <nome.bin>, deve ser 1.bin, 2.bin até N.bin (onde
% N é a quantidade de amostras);
%
% O usuário deve fornecer :
% - o tamanho dos vetores de amostras, a serem usados no cálculo;
% - o número de amostras a ser usado no cálculo;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Solicita ao usuário o número de vetores de amostras (arquivos);

N = input('Forneça o número de vetores a ser usado : ');
L = input('Forneça o tamanho do vetor de amostras : ');

% Efetua a leitura dos vetores (arquivos) a serem usados no cálculo, montando os vetores v1...v(n)

for n=1:1:N;

  % Efetua a leitura de um vetor

  fid = fopen(int2str(n),'r')';
  a=fread(fid,Inf,'short');
  fclose(fid);

  % Efetua o cálculo da fft de cada arquivo segundo o intervalo fornecido.
  % Observações : a é o vetor com indice=1 até L;
  %         b é o vetor linha correspondente ao vetor a coluna;
```

```
%       c é o vetor resultante da FFT, considerado em módulo;
%       d é o vetor "c" normalizado pelo seu máximo;
%        e é o vetor FFT tomado pela metade (devido a simetria).

% Converte a(vetor coluna) em b(vetor linha)

x = 1:1:L;
b(x) = a(x);

% Calcula o vetor correspondente a FFT

c = abs(fft(b));
d = c/(max(c));
x = 1:1:L/2;
  e = d(x);

% Salva o vetor resultante da fft no vetor v(n).

  eval(['v',int2str(n),'=e;']);
end

% Cálculo da matriz de coeficientes de autocorrelação "C" entre os diversos vetores v(n).

for l = 1:1:N;   % Varre os valores de linha da matriz.
  eval(['A=v',int2str(l),';']);
  mediaA = sum(A)/64;
  for c = 1:1:N;   % Varre os valores de coluna da matriz.
    eval(['B=v',int2str(c),';']);
    mediaB = sum(B)/64;
    aux1 = 0;
    aux2 = 0;
    aux3 = 0;
    for i=1:1:64;
      aux1 = aux1 + (A(i)-mediaA)*(B(i)-mediaB);
      aux2 = aux2 + (A(i)-mediaA)^2;
      aux3 = aux3 + (B(i)-mediaB)^2;
    end
    C(l,c) = aux1 / sqrt(aux2*aux3);
  end
end

% Salva a matriz "C" como arquivo texto, para posterior impressão

save c.dat C -ascii


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: MXCORAMP.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para calcular a matriz dos coeficientes de correlação entre as amostras consideradas.
%
% O nome dos arquivos binários a serem utilizados <nome.bin>, deve ser 1.bin, 2.bin até N.bin (onde N
% é a quantidade de amostras);
%
% O usuário deve fornecer :
```

```
% - o tamanho dos vetores de amostras, a serem usados no cálculo;
% - o número de amostras a ser usado no cálculo;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Solicita ao usuário o número de vetores de amostras (arquivos);

N = input('Forneça o número de vetores a ser usado : ');
L = input('Forneça o tamanho do vetor de amostras : ');

% Efetua a leitura dos vetores (arquivos) a serem usados no cálculo, montando os vetores v1...v(n)

for n=1:1:N;

  % Efetua a leitura de um vetor

  fid = fopen(int2str(n),'r')';
  a=fread(fid,Inf,'short');
  fclose(fid);

  % Efetua o cálculo da fft de cada arquivo segundo o intervalo fornecido.
  % Observações : a é o vetor com indice=1 até L;
  %          b é o vetor linha correspondente ao vetor a coluna;

  % Converte a(vetor coluna) em b(vetor linha)

  x = 1:1:L;
  b(x) = a(x);

  % Salva o vetor resultante no vetor v(n).

  eval(['v',int2str(n),'=b;']);

end

% Cálculo da matriz de coeficientes de autocorrelação "C" entre os diversos vetores v(n).

for l = 1:1:N;  % Varre os valores de linha da matriz.
  eval(['A=v',int2str(l),';']);
  mediaA = sum(A)/64;
  for c = 1:1:N;  % Varre os valores de coluna da matriz.
    eval(['B=v',int2str(c),';']);
    mediaB = sum(B)/64;
    aux1 = 0;
    aux2 = 0;
    aux3 = 0;
    for i=1:1:64;
      aux1 = aux1 + (A(i)-mediaA)*(B(i)-mediaB);
      aux2 = aux2 + (A(i)-mediaA)^2;
      aux3 = aux3 + (B(i)-mediaB)^2;
    end
    C(l,c) = aux1 / sqrt(aux2*aux3);
  end
end

% Salva a matriz "C" como arquivo texto, para posterior impressão

save c.dat C -ascii
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: MXCORCEP.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para calcular a matriz dos coeficientes de correlação entre as amostras consideradas.
% O conjunto de amostras, é resultado de um real cepstrum.
%
% O nome dos arquivos binários a serem utilizados <nome.bin>, deve ser 1.bin, 2.bin até N.bin (onde N
% é a quantidade de amostras);
%
% O usuário deve fornecer :
% - o tamanho dos vetores de amostras, a serem usados no cálculo;
% - o número de amostras a ser usado no cálculo;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Solicita ao usuário o número de vetores de amostras (arquivos);

N = input('Forneça o número de vetores a ser usado : ');
L = input('Forneça o tamanho do vetor de amostras : ');

% Efetua a leitura dos vetores (arquivos) a serem usados no cálculo, montando os vetores v1...v(n)

for n=1:1:N;

 % Efetua a leitura de um vetor

 fid = fopen(int2str(n),'r)';
 a=fread(fid,Inf,'short');
 fclose(fid);

 % Efetua o cálculo da fft de cada arquivo segundo o intervalo fornecido.
 % Observações : a é o vetor com indice=1 até L;
 %          b é o vetor linha correspondente ao vetor a coluna;
 %          c é o vetor resultante apos o cepstrum
 %          d é o vetor "c" normalizado pelo seu máximo;
 %          e é o vetor cepstrum normalizado tomado pela metade
 %          (devido a simetria).

 % Converte a(vetor coluna) em b(vetor linha)

  x = 1:1:L;
  b(x) = a(x);

 % Calcula o vetor correspondente ao cepstrum real (e)

  c = rceps(b);
  d = c/(max(c));
   x = 1:1:L/2;
   e = d(x);

 % Salva o vetor resultante do cepstrum no vetor v(n).

  eval(['v',int2str(n),'=e;']);
end
```

```
% Cálculo da matriz de coeficientes de autocorrelação "C" entre os diversos vetores v(n).

for l = 1:1:N;  % Varre os valores de linha da matriz.
  eval(['A=v',int2str(l),';']);
  mediaA = sum(A)/64;
  for c = 1:1:N;  % Varre os valores de coluna da matriz.
    eval(['B=v',int2str(c),';']);
    mediaB = sum(B)/64;
    aux1 = 0;
    aux2 = 0;
    aux3 = 0;
    for i=1:1:64;
      aux1 = aux1 + (A(i)-mediaA)*(B(i)-mediaB);
      aux2 = aux2 + (A(i)-mediaA)^2;
      aux3 = aux3 + (B(i)-mediaB)^2;
    end
    C(l,c) = aux1 / sqrt(aux2*aux3);
  end
end

% Salva a matriz "C" como arquivo texto, para posterior impressão

save c.dat C -ascii

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: MXCORPOT.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%         Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para calcular a matriz dos coeficientes de correlação entre as amostras consideradas.
% O conjunto de amostras, é resultado do quadrado do valor de cada amostra.
%
% O nome dos arquivos binários a serem utilizados <nome.bin>, deve ser 1.bin, 2.bin até N.bin (onde N
% é a quantidade de amostras);
%
% O usuário deve fornecer :
% - o tamanho dos vetores de amostras, a serem usados no cálculo;
% - o número de amostras a ser usado no cálculo;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Solicita ao usuário o número de vetores de amostras (arquivos);

N = input('Forneça o número de vetores a ser usado : ');
L = input('Forneça o tamanho do vetor de amostras : ');

% Efetua a leitura dos vetores (arquivos) a serem usados no cálculo, montando os vetores v1...v(n)

for n=1:1:N;

  % Efetua a leitura de um vetor

  fid = fopen(int2str(n),'r)';
  a=fread(fid,Inf,'short');
  fclose(fid);
```

```
% Efetua o cálculo do quadrado de cada vetor segundo o intervalo fornecido.
% Observações : a é o vetor com indice=1 até L;
%             b é o vetor linha correspondente ao vetor a coluna;
%             c é o vetor com os valores quadrados

% Converte a(vetor coluna) em b(vetor linha)

x = 1:1:L;
b(x) = a(x);

% Calcula o vetor correspondente ao quadrado dos valores de "b".

for i=1:1:L;
  c(i) = (b(i))^2;
end
d = c/(max(c));
 x = 1:1:L/2;
 e = d(x);

% Salva o vetor resultante do cepstrum no vetor v(n).

 eval(['v',int2str(n),'=e;']);
end

% Cálculo da matriz de coeficientes de autocorrelação "C" entre os diversos vetores v(n).

for l = 1:1:N;  % Varre os valores de linha da matriz.
  eval(['A=v',int2str(l),';']);
  mediaA = sum(A)/64;
  for c = 1:1:N;  % Varre os valores de coluna da matriz.
    eval(['B=v',int2str(c),';']);
    mediaB = sum(B)/64;
    aux1 = 0;
    aux2 = 0;
    aux3 = 0;
    for i=1:1:64;
      aux1 = aux1 + (A(i)-mediaA)*(B(i)-mediaB);
      aux2 = aux2 + (A(i)-mediaA)^2;
      aux3 = aux3 + (B(i)-mediaB)^2;
    end
    C(l,c) = aux1 / sqrt(aux2*aux3);
  end
end

% Salva a matriz "C" como arquivo texto, para posterior impressão

save c.dat C -ascii


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: PERCEF.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa que implementa treinamento do Perceptron, com numero de entradas variável.
% W(NINP) - vetor de pesos do neurônio deltaW(NINP) - vetor diferença dos pesos
```

```
% output - saída do Perceptron
% error(TSTEPS) - vetor de erro durante o treinamento
% trainmtx(NTRAINVEC,NINP+1) - matriz de entrada usada para treinamento; o número de colunas é
% "+1", pois o último valor é o "target"
% inputmtx(NINP) - vetor de entrada a ser analisado
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Definição de algumas constantes

NINP = 64;         % Número de entradas do Perceptron
BIAS = -5;         % Output threshold
ALFA = 0.06;       % Learning rate
TSTEPS = 300;      % Número de passos do treinamento
ERSTEP = 3;        % Número de passos de erro antes da correção
NTRAINVEC = 9;           % Número de vetores de treinamento
WEIGHTFAC = 0.01; % Fator multiplicador dos pesos
INERTIA = 0; % Fator de inércia para atualização dos pesos
DINERTIA = 0; % Delta inércia

% Inicialização do vetor de pesos, com valores pseudo-randômicos pequenos

'Inicializando...'
W = randn(1,NINP+1)*WEIGHTFAC;      % Inicializa os pesos, gera vetor de pesos
deltaW(NINP+1) = 0;        % Inicializa os deltas
deltaWold(NINP+1) = 0;  % Inicializa os deltas anteriores
error(TSTEPS) = 0; % Inicializa o vetor de erro
erstep = 0; % Inicializa contador de steps para corrigir pesos

% Treinamento do Perceptron, segundo a regra Adaline

% Efetua a leitura da matriz de treinamento

load trainmtx;

% Treinamento do Perceptron

'Treinando...'

for step=1:TSTEPS

   % Efetua o "sorteio" do vetor de entrada/saída a ser aplicado

   chose = fix(rand*NTRAINVEC+1); % Calcula o número do vetor escolhido
   if chose > NTRAINVEC, chose = chose-1, end

   % Calcula o valor total da entrada, e determina o target atual

   pinput = 0;
   for i=1:NINP
      pinput = pinput + W(i)*(trainmtx(chose,i)); % Calcula o input parcial
   end
   input(step) = pinput - BIAS; % Calcula o input total, considerando BIAS
   target = trainmtx(chose,NINP+1); % O target é o último elemento

   % Calcula a saída atual

   a = 45; % Coeficiente "a" da neural function
   b = 45; % Coeficiente "b" da neural function
```

```
    c = -0.2; % Coeficiente "c" da neural function
    teta = 6; % Coeficiente "teta" da neural function (shift)
    output(step) = (2*a/(1 + exp(c*input(step)+teta)))-b; % Função genérica ajustável

    % Calcula o erro da saída; o cálculo é feito a cada "ERSTEP"

    error(step) = output(step)-target;
    erstep = erstep+1;
    if erstep == ERSTEP
      toterror = 0;
      for k=step-ERSTEP+1:step
        l = error(k);
        toterror = toterror + l^2;
      end
      finerror = sqrt(toterror);

      % Atualiza os pesos das conexões de entrada

      for i=1:NINP
        deltaW(i) = trainmtx(chose,i)*finerror*ALFA; % Cálculo do incr. W
        Wnew(i) = W(i)+deltaW(i)+INERTIA*deltaWold(i);
        deltaWold(i)=deltaW(i);
      end
      deltaW(NINP+1) = BIAS*finerror*ALFA; % Cálculo da correção do peso do BIAS
      Wnew(NINP+1) = W(NINP+1)+deltaW(NINP+1)+INERTIA*deltaWold(NINP+1);
      W = Wnew;
      erstep = 0;
    end
  end

  plot(error);
  title('Perceptron: Gráfico do erro x Steps de treinamento');
  xlabel('steps');
  ylabel('error');


  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
  % Título: PERCETF.M
  % Versão: 25/11/94
  % Autor: Antonio Pedro Timoszczuk
  %        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
  %
  % Programa que verifica o funcionamento do Perceptron, com numero de entradas variável.
  % W(NINP) - vetor de pesos do neurônio
  % deltaW(NINP) - vetor diferença dos pesos
  % output - saída do Perceptron
  % error(TSTEPS) - vetor de erro durante o treinamento
  % testmtx(NINP,NTESTVEC) - matriz de entrada usada para treinamento.
  % inputmtx(NINP) - vetor de entrada a ser analisado
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Definição de algumas constantes

  NTESTVEC = 9;% Número de vetores de teste

  output(NTESTVEC)=0;
```

```
% Efetua a leitura da matriz de teste

load testmtx;

% Teste do Perceptron

'Testando...'

for steps=1:NTESTVEC

  % Calcula o valor da entrada

  tpinput = 0;
  for i=1:NINP
    tpinput = tpinput + W(i)*testmtx(steps,i); % Calcula o input parcial
  end
  tinput = tpinput - BIAS; % Calcula o input total, considerando BIAS
  toutput(steps) = (2*a/(1+exp(c*tinput+teta)))-b; % Função genérica ajustável
end

plot(toutput,'*');
title('Perceptron: Gráfico de teste');
xlabel('vetor de teste');
ylabel('saída');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: PERCEW.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa que implementa treinamento do Perceptron, com numero de entradas variável.
% W(NINP) - vetor de pesos do neurônio
% deltaW(NINP) - vetor diferença dos pesos
% output - saída do Perceptron
% error(TSTEPS) - vetor de erro durante o treinamento
% trainmtx(NTRAINVEC,NINP+1) - matriz de entrada usada para treinamento; o número de colunas
% é "+1", pois o último valor é o "target"
% inputmtx(NINP) - vetor de entrada a ser analisado
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Definição de algumas constantes

  NINP = 128;      % Número de entradas do Perceptron
  BIAS = 0;        % Output threshold
  ALFA = 0.4;      % Learning rate
  TSTEPS = 2000; % Número de passos do treinamento
  ERSTEP = 1;      % Número de passos de erro antes da correção
  NTRAINVEC = 25;        % Número de vetores de treinamento
  WEIGHTFAC = 0.01; % Fator multiplicador dos pesos
  INERTIA = 0; % Fator de inércia para atualização dos pesos

  % Inicialização do vetor de pesos, com valores pseudo-randômicos pequenos

  'Inicializando...'
```

```
W = randn(1,NINP+1)*WEIGHTFAC;      % Inicializa os pesos, gera vetor de pesos
deltaW(NINP+1) = 0;      % Inicializa os deltas
deltaWold(NINP+1) = 0;  % Inicializa os deltas anteriores
error(TSTEPS) = 0; % Inicializa o vetor de erro
erstep = 0; % Inicializa contador de steps para corrigir pesos

% Treinamento do Perceptron, segundo a regra Adaline

% Efetua a leitura da matriz de treinamento

load trainmtx;

% Treinamento do Perceptron

'Treinando...'

for step=1:TSTEPS

 % Efetua o "sorteio" do vetor de entrada/saida a ser aplicado

 chose = fix(rand*NTRAINVEC+1); % Calcula o número do vetor escolhido
 if chose > NTRAINVEC, chose = chose-1, end

 % Calcula o valor total da entrada, e determina o target atual

 pinput = 0;
 for i=1:NINP
   pinput = pinput + W(i)*(trainmtx(chose,i)); % Calcula o input parcial
 end
 input(step) = (pinput - BIAS); % Calcula o input total, considerando BIAS
 target = trainmtx(chose,NINP+1); % O target é o último elemento

 % Calcula a saída atual

 a = 45; % Coeficiente "a" da neural function
 b = 45; % Coeficiente "b" da neural function
 c = -0.1; % Coeficiente "c" da neural function
 teta = 6; % Coeficiente "teta" da neural function (shift)
 output(step) = (2*a/(1 + exp(c*input(step)+teta)))-b; % Função genérica ajustável
 % output(step) = input^2/1+input^2;

 % Calcula o erro da saída; o cálculo é feito a cada "ERSTEP"

 error(step) = output(step)-target;
 erstep = erstep+1;
 if erstep == ERSTEP

  toterror = 0;
  for k=step-ERSTEP+1:step
    l = error(k);
    toterror = toterror + l^2;
  end
  finerror = sqrt(toterror);

 % Atualiza os pesos das conexões de entrada

 for i=1:NINP
  deltaW(i) = trainmtx(chose,i)*finerror*ALFA; % Cálculo do incr. W
  Wnew(i) = W(i)+deltaW(i)+INERTIA*deltaWold(i);
```

```
        deltaWold(i)=deltaW(i);
      end
      deltaW(NINP+1) = BIAS*finerror*ALFA; % Cálculo da correção do peso do BIAS
      Wnew(NINP+1) = W(NINP+1)+deltaW(NINP+1)+INERTIA*deltaWold(NINP+1);
      W = Wnew;
      erstep = 0;
    end
  end

  plot(error);
  title('Perceptron: Gráfico do erro x Steps de treinamento');
  xlabel('steps');
  ylabel('error');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: PERCETW.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa que verifica o funcionamento do Perceptron, com numero de entradas variável.
% W(NINP) - vetor de pesos do neurônio
% deltaW(NINP) - vetor diferença dos pesos
% output - saida do Perceptron
% error(TSTEPS) - vetor de erro durante o treinamento
% testmtx(NINP,NTESTVEC) - matriz de entrada usada para treinamento.
% inputmtx(NINP) - vetor de entrada a ser analisado
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Definição de algumas constantes

NTESTVEC = 30;            % Número de vetores de teste
output(NTESTVEC)=0;

% Efetua a leitura da matriz de teste

load testmtx;

% Teste do Perceptron

'Testando...'

for steps=1:NTESTVEC

   % Calcula o valor da entrada

   tpinput = 0;
   for i=1:NINP
     tpinput = tpinput + W(i)*testmtx(steps,i); % Calcula o input parcial
   end
   tinput = tpinput - BIAS; % Calcula o input total, considerando BIAS

   % Calcula a saída atual

   toutput(steps) = (2*a/(1+exp(c*tinput+teta)))-b; % Função genérica ajustável
end
```

```
plot(toutput,'*');
title('Perceptron: Gráfico de teste');
xlabel('vetor de teste');
ylabel('saída');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: VERBIN.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%         Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para visualizar os arquivos "*.bin", que contém os dados a serem analisados. Traça o
% gráfico de amplitude em função das amostras.
% O usuário deve fornecer :
% - o nome do arquivo binário a ser visualizado <nome.bin>;
% - o início do intervalo a ser visualizado <amostra inicial>,cujo valor mínimo é 1;
% - o final do intervalo a ser visualizado <amostra final>, cujo valor máximo é igual ao número total de
% amostras.
% O número de amostras total deve ser conhecido de forma a permitir uma primeira visualização.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Efetua a leitura do arquivo a ser tratado, criando o vetor a.

fnamedat = input('Nome do arquivo : ','s');
fid= fopen(fnamedat,'r');
a=fread(fid,Inf,'short');
fclose(fid);

% Traça o gráfico dos dados no intervalo definido por inicio e final.

inicio = input('Forneça o inicio do intervalo : ');
final = input('Forneça o final do intervalo : ');
x=inicio:1:final;
plot(x,a(x));

% Completa o gráfico com informações úteis.

grid
title(fnamedat)
xlabel('amostras')
ylabel('amplitude')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: VERCEP.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%         Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
% Programa para calcular o RealCepstrum dos arquivos "*.bin", que contém os dados a serem
analisados.
% O usuário deve fornecer :
% - o nome do arquivo binário a ser visualizado <nome.bin>;
```

```
% - o início do intervalo a ser usado no cálculo, correspondendo a <amostra inicial>, cujo valor minimo
é 1;
% - o final do intervalo a ser usado no cálculo, correspondendo a <amostra final>, cujo valor máximo é
% igual ao número total de amostras.
% O número de amostras total deve ser conhecido de forma a permitir uma primeira visualização, sendo
% que o intervalo de interesse pode ser determinado pela visualização das amostras (vide programa de
% visualização das amostras).
%
% Frequência de amostragem de 4000 Hz.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Efetua a leitura do arquivo a ser tratado

fnamedat = input('Nome do arquivo : ','s');
fid= fopen(fnamedat,'r');
a=fread(fid,Inf,'short');
fclose(fid);

% Efetua o cálculo do cepstrum de cada arquivo segundo o intervalo fornecido.
% Observações : b(x) é um vetor com indice=1 ... tamanho;
%               d(x) é o vetor normalizado pelo máximo de b(x).

inicio = input('Forneça o inicio do intervalo : ');
final = input('Forneça o final do intervalo : ');
tamanho = final-inicio;
x = inicio:1:final;
b = a(x);
d = b/(max(b));

% Calcula o RealCepstrum da sequencia d

e = rceps(d);

% Traça o gráfico do cepstrum calculado.
% Observação : xn corresponde a escala normalizada em frequência.

x = 1:1:tamanho/2;
xn = (4000/(tamanho/2))*x;
plot(xn,e(x));

% Completa o gráfico com informações úteis.

grid
title(fnamedat)
text (0,max(e)+0.1,'Cepstrum de :')
text (3000,max(e)+0.1,'Num. pontos =')
text (4000,max(e)+0.1,int2str(tamanho+1))
xlabel('quefrencia')
ylabel('amplitude')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: VERDFT.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
%
```

```
% Programa para calcular a FFT dos arquivos "*.bin", que contémos dados a serem analisados.
% O usuário deve fornecer :
% - o nome do arquivo binário a ser visualizado <nome.bin>;
% - o início do intervalo a ser usado no cálculo, correspondendo a <amostra inicial>, cujo valor minimo
é 1;
% - o final do intervalo a ser usado no cálculo, correspondendo a <amostra final>, cujo valor máximo é
% igual ao número total de amostras.
% O número de amostras total deve ser conhecido de forma a permitir uma primeira visualização, sendo
% que o intervalo de interesse pode ser determinado pela visualização das amostras (vide programa de
% visualização das amostras).
%
% Frequência de amostragem de 4000 Hz.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Efetua a leitura do arquivo a ser tratado

fnamedat = input('Nome do arquivo : ','s');
fid= fopen(fnamedat,'r');
a=fread(fid,Inf,'short');
fclose(fid);

% Efetua o cálculo da fft de cada arquivo segundo o intervalo fornecido.
% Observações : b(x) é um vetor com indice=1 ... tamanho;
%               c(x) toma o módulo;
%               d(x) é o vetor normalizado pelo máximo de c(x).

inicio = input('Forneça o inicio do intervalo : ');
final = input('Forneça o final do intervalo : ');
tamanho = final-inicio;
x = inicio:1:final;
b = a(x);
c = abs(fft(b));
d = c/(max(c));

% Traça o gráfico da fft calculada.
% Observação : xn corresponde a escala normalizada em frequência.

x = 1:1:tamanho/2;
xn = (4000/(tamanho/2))*x;
plot(xn,d(x));

% Completa o gráfico com informações úteis.

grid
title(fnamedat)
text (0,1.05,'Espectro em freq. de :')
text (3000,1.05,'Num. pontos =')
text (4000,1.05,int2str(tamanho+1))
xlabel('frequência (Hz)')
ylabel('amplitude')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Título: VERWAVLT.M
% Versão: 25/11/94
% Autor: Antonio Pedro Timoszczuk
%        Escola Politécnica da USP-DEE - LCS - São Paulo - S.P.
```

```
%
% Programa para visualizar a DWT (Discrete Wavelet Transform) dos arquivos "*.bin", que contém os
% dados.
% Traça o gráfico de amplitude em função das amostras.
% O usuário deve fornecer :
% - o nome do arquivo binário a ser visualizado <nome.bin>;
% - o início do intervalo a ser visualizado <amostra inicial>, cujo valor minimo é 1;
% - o final do intervalo a ser visualizado <amostra final>, cujo valor máximo é igual ao número total de
% amostras.
% O número de amostras total deve ser conhecido de forma a permitir uma primeira visualização.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Efetua a leitura do arquivo a ser tratado, criando o vetor a.

fnamedat = input('Nome do arquivo : ','s');
fid= fopen(fnamedat,'r');
v=fread(fid,Inf,'short');
fclose(fid);
inicio = input('Forneça o inicio do intervalo : ');
final = input('Forneça o final do intervalo : ');
L=final;

% Cálculo dos novos vetores transformados, a partir dos vetores v(n).

C0 = 0.4829629131445341;
C1 = 0.8365163037378079;
C2 = 0.2241438680420134;
C3 = -0.1294095225512604;
NN = L;
NH = L/2;
  while (NN >= 4)
    i = 1;
    for j = 1:2:L-3
      w(i) = C0*v(j)+C1*v(j+1)+C2*v(j+2)+C3*v(j+3);
      w(i+NH) = C3*v(j)-C2*v(j+1)+C1*v(j+2)-C0*v(j+3);
      i = i+1;
    end
    w(i) = C0*v(L-1)+C1*v(L)+C2*v(1)+C3*v(2);
    w(i+NH) = C3*v(L-1)-C2*v(L)+C1*v(1)-C0*v(2);
    v=w;
    NH = NH/2;
    NN = NN/2;
  end

% Traça o gráfico dos dados no intervalo definido por inicio
% e final.

x=inicio:1:final;
plot(x,v(x));

% Completa o gráfico com informações úteis.

grid
s=['Wavelet:  ',fnamedat,'  Num. pontos = ',int2str(final)];
title(s)
xlabel('amostra')
ylabel('amplitude')
```

BOLETINS TÉCNICOS - TEXTOS PUBLICADOS

BT/PEE/93-01 - Oscilador a HEMT - 10 GHz - FÁTIMA S. CORRERA, EDMAR CAMARGO

BT/PEE/93-02 - Representação Senoidal da Voz através dos Polos do Filtro Preditor - MARCELO B. JOAQUIM, NORMONDS ALENS

BT/PEE/93-03 - Blindagens por Grades Condutoras: Cálculo do Campo Próximo - LUIZ CEZAR TRINTINALIA, ANTONIO ROBERTO PANICALI

BT/PEE/93-04 - Sistema de Otimização e Controle de Produção em Minas de Pequeno e Médio Porte - TSEN CHUNG KANG, VITOR MARQUES PINTO LEITE

BT/PEE/94-01 - Determinação das Frases de Aplicação Forense para o projeto NESPER e Tese de Mestrado IME/94, com Base em Estudos Fonéticos - MARCONI DOS REIS BEZERRA, EUVALDO F. CABRAL JUNIOR

BT/PEE/94-02 - Implementação e Teste de uma Rede Neural Artificial do Tlpo KSON (Kohonen Self-Organizing Network) com Entradas Bidimensionais - MARCELO YASSUNORI MATUDA, EUVALDO F. CABRAL JR.

BT/PEE/94-03 - Transformada de Walsh e Haar Aplicadas no Processamento de Voz - ALEXANDRE AUGUSTO OTTATI NOGUEIRA, THIAGO ANTONIO GRANDI DE TOLOSA, EUVALDO F. CABRAL JÚNIOR

BT/PEE/94-04 - Aplicação de Redes Neurais ao Problema de Reconhecimento de Padrões por um Sonar Ativo - ALEXANDRE RIBEIRO MORRONE, CRISTINA COELHO DE ABREU, EDUARDO KOITI KIUKAWA, EUVALDO F. CABRAL JR.

BT/PEE/94-05 - Tudo que se Precisa Saber sobre a Prática da FFT - Transformada Rápida de Fourier (Inclui Software) - ROGÉRIO CASAGRANDE, EUVALDO F. CABRAL JR.

BT/PEE/94-06 - A Survey on Speech Enhancement Techniques of Interest to Speaker Recognition - CELSO S. KURASHIMA, EUVALDO F. CABRAL JR.

BT/PEE/94-07 - Identificação de Pulsos Decádicos em Linhas Telefônicas - ANTONIO P. TIMOSZCZUK, MÁRCIO A. MATHIAS, EUVALDO F. CABRAL JR.

BT/PEE/94-08 - Implementação e Teste de Filtros do Tipo Adaptativo e "Notch" para a Remoção de Interferência de 60 Hz em Sinais de Eletrocardiograma - FLÁVIO ANTÔNIO MENEGOLA, JOSÉ AUGUSTO DE MATTOS, JOSÉ GOMES G. FILHO, SIDNEY SILVA VIANA, EUVALDO F. CABRAL JR.

BT/PEE/94-09 - Compressão de Sinais de Voz utilizando Transformadas de Karhunen-Loève, Fourier e Hadamard - IVAN LUIS VIEIRA, LUIZ FERNANDO STEIN WETZEL, EUVALDO F. CABRAL JR.

BT/PEE/94-10 - "Ray Tracing" Paralelo - EDUARDO TOLEDO SANTOS, JOÃO ANTONIO ZUFFO

BT/PEE/94-11 - Implementação de uma Ferramenta Posicionador para "Gate-Arrays" Tipo Mar de Portas - JORGE W. PERLAZA PRADO, WILHELMUS A. M. VAN NOIJE

BT/PEE/94-12 - Tudo que se Precisa Saber Sobre a Teoria da FFT - Transformada Rápida de Fourier - FÁBIO LUÍS ROMÃO, REINALDO SILVEIRA, ROGÉRIO CASAGRANDE, EUVALDO CABRAL JR.

BT/PEE/94-13 - Análise do Ruído Sonoro em uma Sala de Aquisição de Amostras de Som com Microcomputador - FÁBIO LUÍS ROMÃO, REINALDO SILVEIRA, EUVALDO CABRAL JR.

BT/PEE/94-14 - Cor: Aspectos Relevantes para Visualização de Dados - SÍLVIA DELGADO OLABARRIAGA

BT/PEE/94-15 - Projeto de Filtros Digitais IIR com Fase Aproximadamente Linear Utilizando Redução de Ordem - IVAN F. J. RODRIGUES, MAX GERKEN

BT/PEE/94-16 - GERAFILTRO: Sistema para Projeto Automático de Filtros Digitais "IIR" (da especificação em alto nível ao leiaute do "ASIC") - RICARDO PIRES, JOSÉ VIEIRA DO VALE NETO