Boletim Técnico da Escola Politécnica da USP

Departamento de Engenharia Eletrônica

BT/PEE/95-03

Utilização de Redes Neurais Artificiais para Detecção e Identificação de Falhas em Circuitos

Márcio Yukio Teruya
Roberto Amilton Bernardes Sória
Euvaldo Cabral Jr.

São Paulo - 1995

Teruya, Marcio Yukio
Utilização de redes neurais artificiais para detecção e identificacão de falhas em circuitos / M.Y. Teruya, R.A.B. Sória, E.F. Cabral Jr. -- São Paulo : EPUSP, 1995.
31p. -- (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia Eletrônica, BT/PEE/95-20)

1. Redes neurais (Inteligência artificial) 2. Circuitos eletrônicos I. Cabral Jr., Euvaldo Ferreira II. Sória, Roberto Amilton Bernardes III. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Eletrô nica IV. Título V. Série

CDD 006.3
621.3815

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA DETECÇÃO E IDENTIFICAÇÃO DE FALHAS EM CIRCUITOS

Márcio Yukio Teruya
Lab. de Micro-Eletrônica (LME), DEE, Escola Politécnica - USP
CAIXA POSTAL: 61548 CEP: 05424-970 - São Paulo - SP
e-mail: myteruya@lme.usp.br

Roberto Amilton Bernardes Sória
Lab. de Comunicações e Sinais (LCS), DEE, Escola Politécnica - USP
CAIXA POSTAL: 61548 CEP: 05424-970 - São Paulo - SP
e-mail: roberto@lcs.usp.br

Euvaldo Cabral Jr.
Lab. de Comunicações e Sinais (LCS), DEE, Escola Politécnica - USP
CAIXA POSTAL: 61548 CEP: 05424-970 - São Paulo - SP
e-mail: euvaldo@lcs.usp.br

Sumário

Este trabalho apresenta uma aplicação de redes neurais artificiais em detecção e identificação de falhas em circuitos eletrônicos. Um MLP, Multi Layer Perceptron, é usado para detectar as falhas e identificar o componente defeituoso em circuitos analógicos simples. Uma análise das saidas da rede em função da variação dos componentes é realizada. Os resultados obtidos foram satizfatórios, mostrando a eficácia da rede neural artificial para esse tipo de análise.

1. Introdução

A estratégia adotada foi de limitar o processo à análise DC e treinar a rede com tensões linearmente independentes, obtidas apartir da operação correta do circuito. Um circuito simples (poucos componentes) foi utilizado. A rede foi ensinada com valores discretos de tensões dentro da variação normal de cada componente (resistor e capacitor). Os transistores foram considerados defeituosos no caso de haver um curto circuito entre quaisquer dois terminais. Os "targets" foram escolhidos de forma a identificar o componente defeituoso. As falhas foram simuladas considerando valores aleatórios abaixo do mínimo e acima do máximo valor da variação normal de cada componente.

Inicialmente, encontra-se a descrição do modelo de falhas, das etapas de treinamento e os resultados obtidos para um primeiro circuito simples (amplificador de 1 transistor). Em seguida, é explorada a idéia de modularização de um circuito mais complexo (duplicador de frequência) que pode ser dividido em três circuitos mais simples, testaveis por redes pequenas. Uma rede neural relativamente pequena identifica qual o módulo defeituoso.

2. Modelo de falhas.

Pressuposto: o circuito está inicialmente funcionando; a rede opera a partir desta condição.

- Supõe-se que apenas UM componente de cada vez apresenta defeito.

- São previstos três tipos de falhas (SIMPLES) para os circuitos apresentados:

2.1 Falha em RESISTÊNCIA:

Chama-se de RN a resistência nominal. RNmin e RNmax são respectivamente os valores mínimo e máximo admissíveis para os valores da resistência no circuito. (RNmin e RNmax são determinados impondo-se uma faixa de variação aceitável do(s) ponto(s) de operação (tensão de nó(s) significativo(s))). O circuito foi simulado (SPICE) para variações de resistência:

_ entre 0.5RN e RNmin;
_ entre RNmax e infinito.

2.2 Falha em CAPACITOR:

_ estado de curto-circuito (simplificação);

2.3 Falha em TRANSÍSTOR:

_ curto BE;
_ curto CE;
_ curto BC.

3. Treino do MLP para o circuito 1.

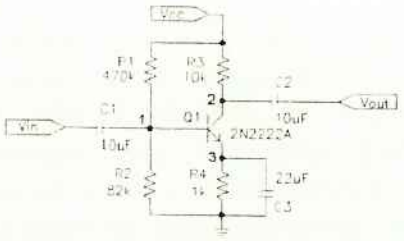

figura 1: Amplificador de 1 transitor. Os nós importantes estão identificados por números.

3.1) A rede neural artificial

É um MLP [Rumelhart] com apenas um hidden layer, treinado com a regra clássica "backpropagation".

3.1.1) Entradas da rede.

Neste circuito, pode-se observar que são apenas 3 os nós com tensões variantes; no caso, são os terminais do transistor. Escolheu-se então:

p1 = coletor,
p2 = base,
p3 = emissor,

porém, verifica-se que p1 e p3 são, aproximadamente, linearmente dependentes, conforme é visto pela figura 3. Sendo assim, no caso simplificado de análise DC, apenas p1 e p2 são significativos e, estes foram os pontos de observação escolhidos. O fato de serem apenas 2 os pontos observados simplifica a análise pois permite a visualização dos resultados em mapas bidimensionais.

3.1.2) Saídas da rede.

Um "output neuron" foi definido para cada tipo de defeito, obtendo-se então:

output 1 em zero: defeito em R1 (R1 < R1Nmin);
output 2 em zero: defeito em R1 (R1 > R1Nmax);
output 3 em zero: defeito em R2 (R2 < R2Nmin);
output 4 em zero: defeito em R2 (R2 > R2Nmax);
output 5 em zero: defeito em R3 (R3 < R3Nmin);
output 6 em zero: defeito em R3 (R3 > R3Nmax);
output 7 em zero: defeito em R4 (R4 < R4Nmin);
output 8 em zero: defeito em R4 (R4 > R4Nmax);
output 9 em zero: defeito no transistor (curto ce);
output 10 em zero: defeito no transistor (curto be);
output 11 em zero: defeito no transistor (curto bc);
output 12 em zero: defeito no capacitor (curto);

OBS.: Na situação de não-defeito o neuron correspondente fica em 1.

3.2) Geração dos padrões de treino:

Os padrões de treino foram gerados segundo o modelo de falhas de cada componente. No caso das resistências, a distribuição dos valores foi escolhida como abaixo (em frações do valor nominal):

R1
Sem falha: 0.98, 0.99, 1.00, 1.01, 1.02, 1.03
falha inferior: 0.50, 0.60, 0.70, 0.80, 0.90, 0.95
falha superior: 1.05, 1.10, 1.50, 2.00, 3.00, 4.00, 6.00, 10.0, 1e+31

R2
Sem falha: 0.96, 0.98, 1.00, 1.02, 1.04
falha inferior: 0.50, 0.60, 0.70, 0.80, 0.90, 0.94
falha superior: 1.06, 1.20, 1.50, 2.00, 3.00, 4.00, 6.00, 10.0, 1e+31

R3
Sem falha: 0.96, 0.98, 1.00, 1.02, 1.04
falha inferior: 0.50, 0.60, 0.70, 0.80, 0.90, 0.94
falha superior: 1.06, 1.20, 1.50, 2.00, 3.00, 4.00, 6.00, 10.0, 1e+31

R4
Sem falha: 0.93, 0.95, 0.97, 0.99,1.01, 1.03, 1.05, 1.07
falha inferior: 0.50, 0.60, 0.70, 0.80, 0.90
falha superior: 1.10, 1.50, 2.00, 3.00, 4.00, 6.00, 10.0, 1e+31

Obs.: Escolha da faixa nominal das resistências: determinada impondo-se um desvio máximo de 5% do ponto de operação no ponto 2 (coletor);

No caso das capacitâncias:

Sem falha: resistência 1e+31
Com falha: resistência nula

No caso do transístor:

Sem falha: resistência 1e+31 entre os terminais BE, CE ou BC.
Com falha: resistência nula entre os terminais BE, CE ou BC.

3.3) Resultados:

figura 4: gráfico do erro.
figuras 5, 6, 7, 8, 9, 10: mapas dos targets em função das entradas 1 e 2.
figuras 11, 12, 13, 14: respostas da rede a variações das resistências.

figuras 15, 16, 17, 18: respostas da rede aos curtos.
Sobre os mapas das figuras 5, 6, 7, 8, 9 e 10:

Legenda para as figuras 3, 5, 6, 7, 8, 9 e 10:
1 - operação nominal do circuito;
2 - falha de R1 (R1< R1Nmin);
3 - falha de R1 (R1 > R1Nmax);
4 - falha de R2 (R2 < R2Nmin);
5 - falha de R2 (R2 > R2Nmax);
6 - falha de R3 (R3 < R3Nmin);
7 - falha de R3 (R3 > R3Nmax);
8 - falha de R4 (R4 < R4Nmin);
9 - falha de R4 (R4 > R4Nmax);
A - falha no transístor (curto CE);
B - falha no transístor (curto BE);
C - falha no transístor (curto BC);
D - falha no capacitor.

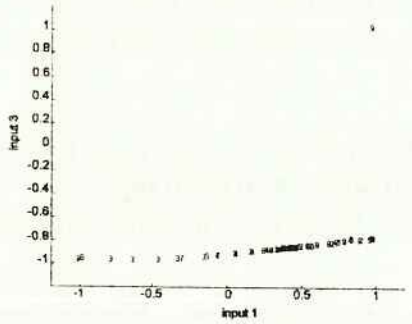

figura 3: Mapa dos targets em função das entradas 1 e 3.

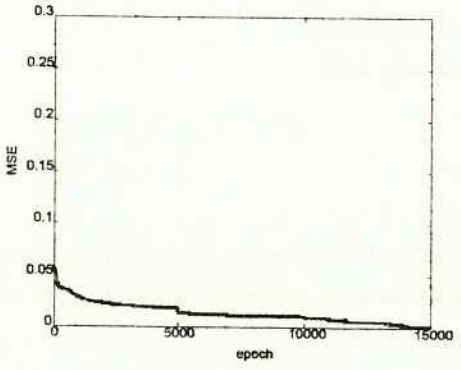

figura 4: Gráfico do decaimento do MSE (Mean Squared Error) em função do número de épocas para o circuito 1.

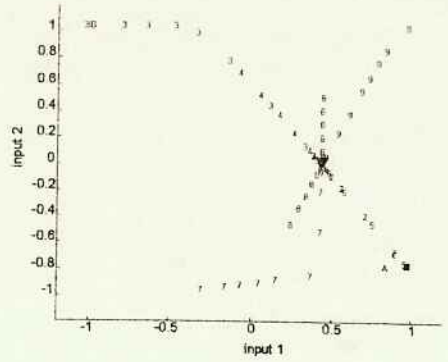

figura 5: mapa dos targets em função das entradas 1 e 2.

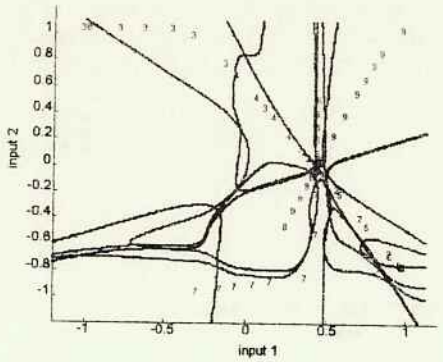

figura 6: mapa dos targets com a separação em regiões dada pela MLP.

Observação sobre a figura 6: Nota-se que o mapa das entradas é separado em várias regiões, cada região delimitada por linhas. Estas regiões identificam as respostas dadas pela rede. Por exemplo, nesta figura tem-se uma região para o target 1, uma para o target 7, etc.; tem-se ainda as regiões ambíguas que apontam para dois ou mais tipos de defeitos como, por exemplo, a região contendo os targets 3 e 4, isto é, a resposta da rede é:

target 3	= 101111111111	-	defeito em R1
	AND		
target 4	= 110111111111	-	defeito em R2
	=		
target 34	= 100111111111	-	defeito em R1 ou em R2.

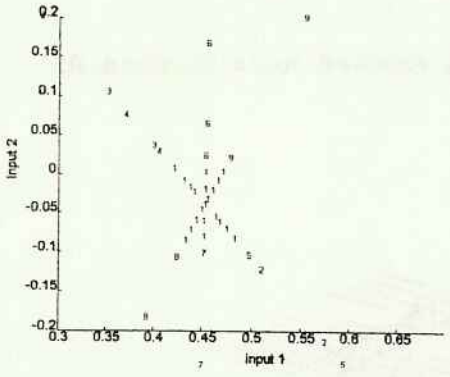

figura 7: zoom da figura 5.

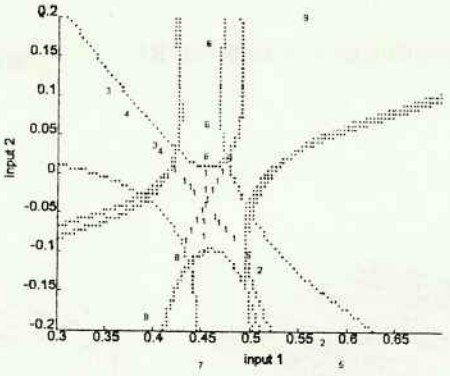

figura 8: zoom da figura 6.

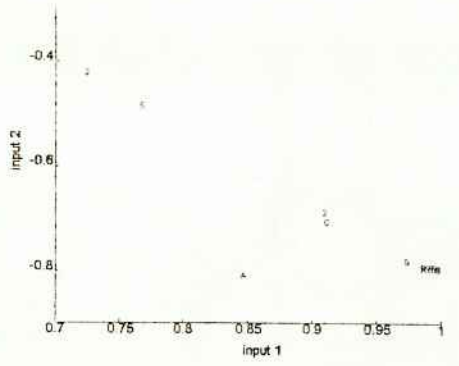

figura 9: zoom da figura 5.

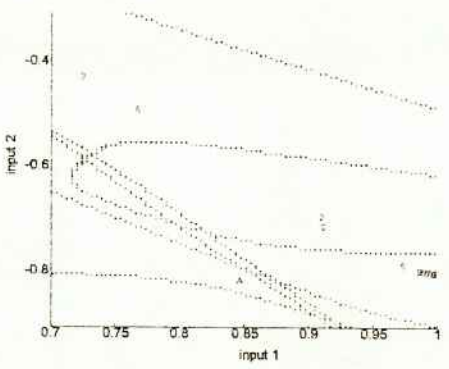

figura 10: zoom da figura 6.

Observação sobre as figuras 11 a 14: as figuras são tridimensionais; o eixo dos X indica o valor da resistência, o eixo dos Y indentifica a saída da rede e o eixo dos Z indica o nível de ativação das saídas da rede. As retas no plano z = 1 delimitam a faixa nominal para a resistência.

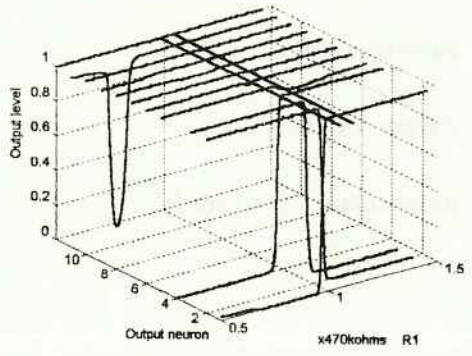

figura 11: resposta da rede a variações de R1

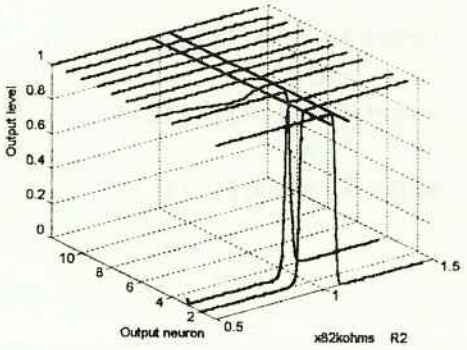

figura 12: resposta da rede a variações de R2

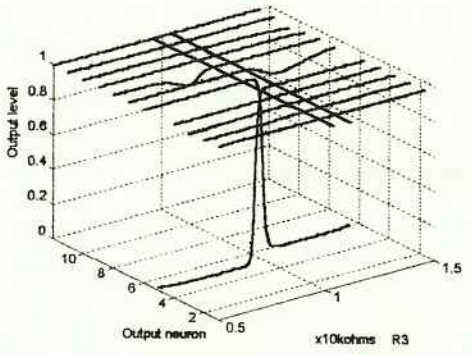

figura 13: resposta da rede a variações de R3

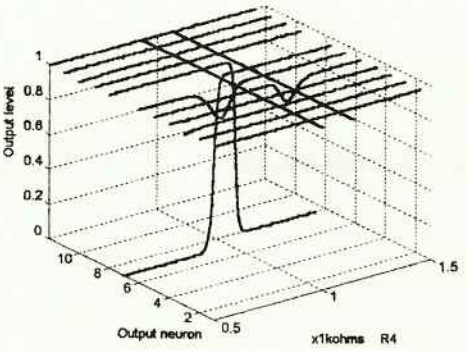

figura 14: resposta da rede a variações de R4.

Observação sobre as figuras 15 a 18: indicam a resposta da rede para defeitos em capacitor e transistor.

Observação sobre as figuras 16 a 18: Repare a resposta ambígua dada pela rede:

figura 16: neurons 2 e 10 indicam falha em R1 e curto BE.
figura 17: neurons 1, 4 e 11 indicam falha em R1, R2 e curto BC.
figura 18: neurons 6, 12 indicam falha em R3 e C3.

Estas ambiguidades originam-se da insuficiência de dados fornecidos à rede.

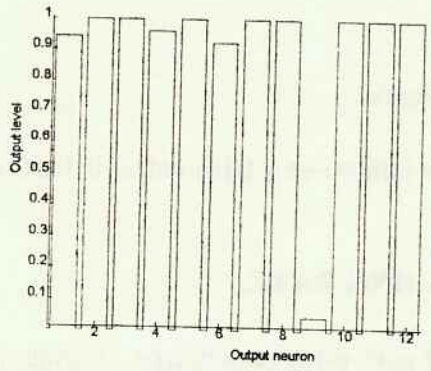

figura 15: resposta da rede para curto CE.

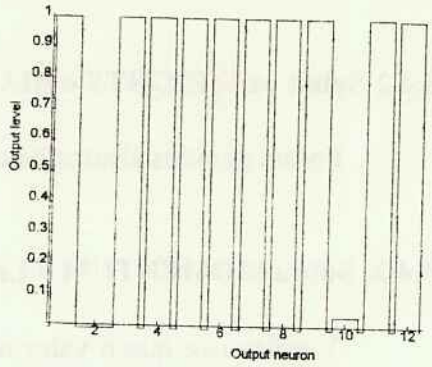

figura 16: resposta da rede para curto BE.

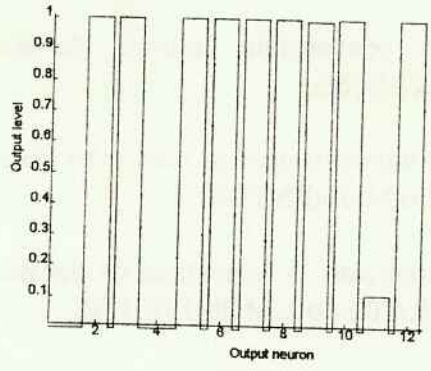

figura 17: resposta da rede para curto BC.

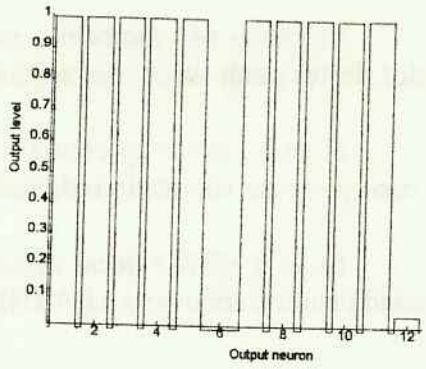

figura 18: resposta da rede para curto em C3.

3.4) Discussão do treinamento.

3.4.1. Número de "Hidden Neurons".

No caso dos treinamentos realizados, a regra da média geométrica aparenta ser valida, isto é, o número ideal de neurons na camada interna é aproximadamente

igual à média geométrica entre o número de neurons da primeira e última camada. A escolha de um número SUFICIENTE se mostrou fundamental, pois:

_ um número insuficiente implica na incapacidade da rede em aprender todos os targets;

_ um número exagerado implica na diminuição da capacidade de GENERALIZAÇÃO da rede;

Para os treinamentos realizados se escolheu:

num_hidden = sqr(num_in*num_out) + **2.**

O mapa da figura 6 mostra que o número de "hidden neurons" poderia ser um pouco menor, pois nota-se claramente que a rede exagerou (pouco) na complexidade das respostas.

3.4.2 Sobre os WEIGHTS e BIAS iniciais.

Foram gerados aleatoriamente e limitou-se a faixa entre -0.1 e 0.1.

3.4.3. Sobre MOMENTUM e LEARNING RATE.

Verificou-se que o valor inicial para estes parâmetros depende muito da configuração da rede e dos targets. Para o circuito em questão, os valores iniciais adotados foram:

MOMENTUM = 0.1
LEARNING RATE = 0.2

Verificou-se também que é conveniente alterar, durante o treinamento, o valor destes parâmetros nas seguintes condições:

a) se o ERRO apresenta comportamento caótico (erro não decaindo regularmente), o caos pode ser eliminado reduzindo-se o MOMENTUM;

b) se o ERRO decai muito lentamente, a velocidade de decaimento pode ser incrementada aumentando-se o LEARNING RATE ou o MOMENTUM.

3.4.4. Contradições nos dados de entrada.

Este foi o problema mais sério do treinamento. Algumas das entradas forneciadas como treinamento tinham significado dúbio, isto é, apontavam para um ou mais targets (pares (entradas, targets) com entradas iguais e targets diferentes).

Estas contradições podem ser vistas facilmente no mapa bidimensional de targets (figuras 5), onde se vê que as regiões determinadas por alguns

targets cruzam com regiões determinadas por outros targets (particularmente R1 e R2 com INTENSIDADE)

Foi necessário eliminar estas contradições para poder treinar adequadamente a rede. Esta eliminação foi feita trocando-se os targets superpostos por targets "ambíguos", isto é, que apontam para mais de um componente defeituoso. Exemplo:

target para R1 defeituoso:	0 1 1 1 1 1 1 1 1 1 1 1
target para R2 defeituoso:	1 1 1 0 1 1 1 1 1 1 1 1
target que substitui os anteriores para as entradas dúbias:	0 1 1 0 1 1 1 1 1 1 1 1

Este processo poderia ser feito manualmente com o auxílio do mapa bidimensional dos targets, porém como fazer com redes de maior número de entradas? Para resolver este problema foi desenvolvido o programa ERROS.EXE, o qual se baseia no seguinte fato:

Se os targets contraditórios para uma região do espaço de entradas estiverem em quantidade semelhante, isto é, se a quantidade de um determinado target não for muito maior que a de outro numa certa região, então os erros correspondentes a certos bits destes targets (em relação à resposta da rede) NUNCA irão a zero (o erro do bit do target em maior quantidade será menor).

O programa ERROS.EXE se aproveita deste fato para modificar os bits que apresentam erro exagerado após um elevado número de épocas (o número de épocas é considerado elevado quando a taxa de decaimento do ERRO é muito baixo).

O processo é ITERATIVO: o programa ERROS.EXE deve ser rodado sempre após um treinamento da rede, com número suficiente de épocas, até que o programa não consiga mais modificar os targets (aproveitando sempre os WEIGHTS e BIAS do treinamento anterior).

3.4.5. Fronteira RNmin/Falha ou RNmax/falha.

Um ponto muito específico e abrupto de transição entre componente bom/componente com falha pode ser conseguido.

Exemplificando: deseja-se que a rede indique falha para R1 quando este ultrapassa EXATAMENTE a faixa 1.100 a 1.101. Se o treinamento fosse feito com estes valores exatamente, o treinamento demoraria muito, talvez não fosse possível obter essa precisão ou o número de targets teria de ser muito elevado. A solução prática é iniciar o treino com uma faixa menos estreita como 1.05 a 1.15 e a seguir, quando a rede já estiver bem treinada, repetir o treinamento (aproveitando-se os WEIGHTS e BIAS obtidos anteriormente) dando à rede dados de "estreitamento", isto é, incluindo-se nos dados de treinamento pontos de estreitamento como 1.06, 1.07, 1.08, 1.09, 1.10, 1.11, ... e assim prossegue-se até obter-se um bom treinamento para a faixa desejada.

3.5) Conclusões.

a) A rede aprendeu EXATAMENTE o que se esperava dela depois de eliminadas as contradições, conforme se pode observar nos gráficos apresentados. A confiabilidade é alta.

b) A determinação exata de todas as falhas depende do número de dados fornecidos à rede; no caso em questão, apenas a informação de voltagens de nós não foi suficiente.

4. Modularização.

Este método viabiliza a análise de circuitos maiores por uma rede de tamanho razoável.

Definições:

módulo 0: é o circuito como um todo.

módulo n: é o enésimo subcircuito do circuito completo.

Seqüência de teste:

A rede observa, inicialmente, o módulo 0, recolhendo apenas uma fração dos dados totais do circuito completo, apenas o suficiente para apontar o módulo n defeituoso. A seguir a mesma rede (por uma troca de WEIGHTS, BIAS e INPUTs) pode ser utilizada para identificar o componente defeituoso dentro do módulo n.

5. Treino do MLP para o circuito 2.

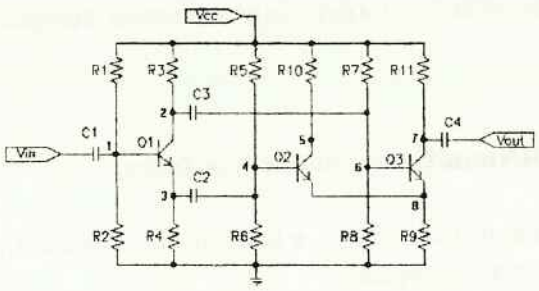

figura 2: Dobrador de freqüência. Os nós importantes estão identificados por números.

5.1) Modularização.

Módulo 0: o circuito completo.

Pontos de observação: 2, 5 e 7.

Módulo 1: subcircuito chefiado por Q1 (C1, R1, R2, R3, R4, Q1).

Pontos de observação: 1, 2 e 3.

Módulo 2: subcircuito chefiado por Q2 (C2, R5, R6, R10, R9 e Q2).

Pontos de observação:

Módulo 3: subcircuito chefiado por Q3 (C3, R7, R8, R11, R9 e Q3).

OBS: Neste item foi feito apenas o treinamento da rede para módulo 0 pois os módulos 1, 2 e 3 são simples amplificadores transitorizados semelhantes ao circuito 1.

5.2) A rede.

Uma MLP com apenas uma hidden layer, treinado com a regra clássica "backpropagation".

5.2.1) Entradas da rede.

Foram escolhidos os pontos 2, 5 e 7 pois estes são os nós que determinam o ponto de operação do módulos 1, 2 e 3 respectivamente:

input 1 = nó 2;
input 2 = nó 5;
input 3 = nó 7.

5.2.2) Saídas da rede.

São necessárias apenas 3 saídas:

output 1 em zero = defeito no módulo 1;
output 2 em zero = defeito no módulo 2;
output 3 em zero = defeito no módulo 3.

OBS.: Na situação de não-defeito o neuron correspondente fica em 1.

5.3) Geração dos padrões de treino.

Processo idêntico ao utilizado para o circuito 1.

5.4) Treinamento da rede.

Foi necessário apenas um ajuste de contradição sobre R9 (aponta defeito para módulos 2 e 3). No caso bastaria que ele apontasse para 2 ou 3.

5.5) Resultados obtidos.

As figuras 19 a 40 são tridimensionais. O eixo dos X indica o valor da resistência, o eixo dos Y indentifica a saída da rede e o eixo dos Z indica o nível de ativação das saídas da rede. As retas no plano z = 1 delimitam a faixa nominal para a resitencia.

OBS: As figuras 41 a 51 são análogas às anteriores, porém sem variação de parâmetro do componente.

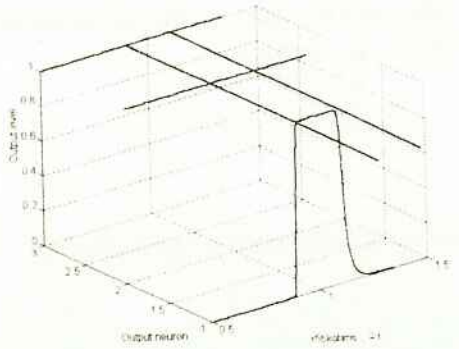

figura 19: resposta da rede a variações de R1.

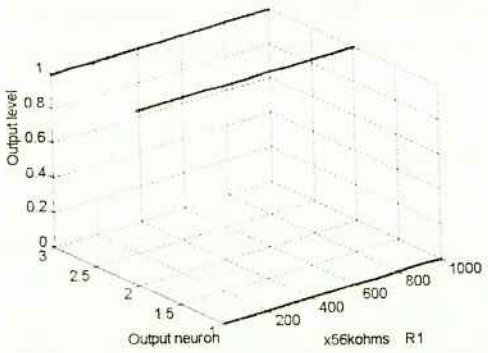

figura 20: idem, para valores superiores de R1.

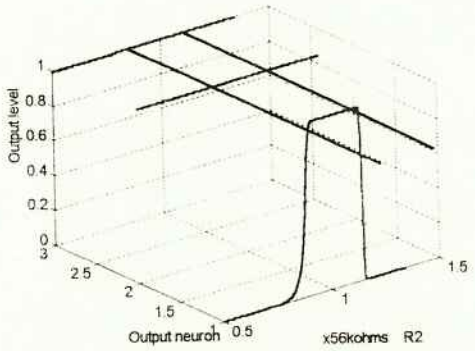

figura 21: resposta da rede a variações de R2.

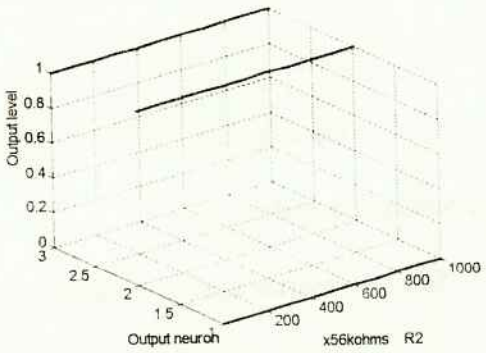

figura 22: idem, para valores superiores de R2.

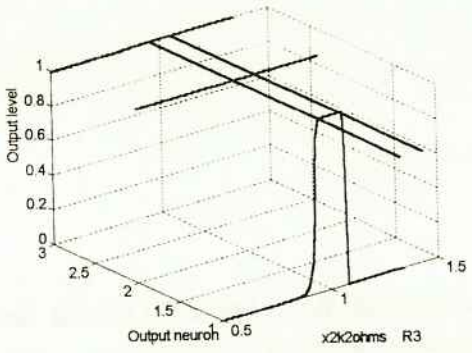

figura 23: resposta da rede a variações de R3.

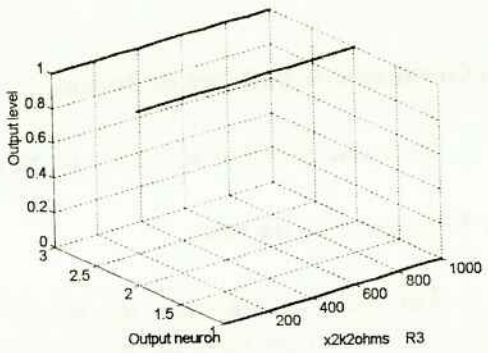

figura 24: idem, para valores superiores de R3.

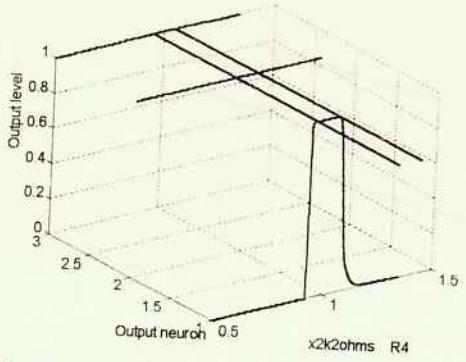

figura 25: resposta da rede a variações de R4.

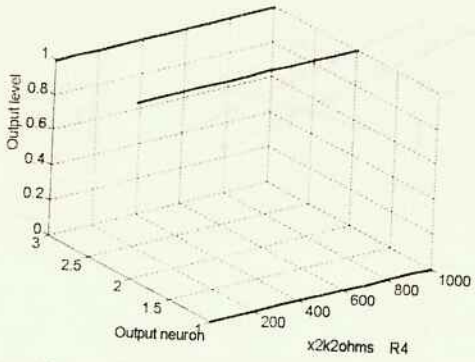

figura 26: idem, para valores superiores de R4.

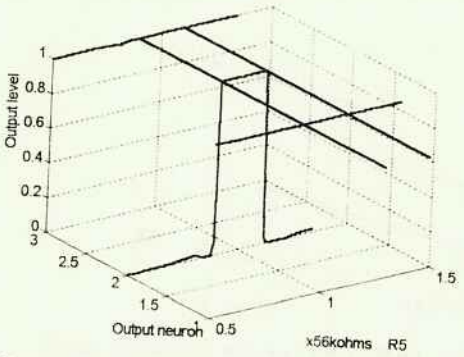

figura 27: resposta da rede a variações de R5.

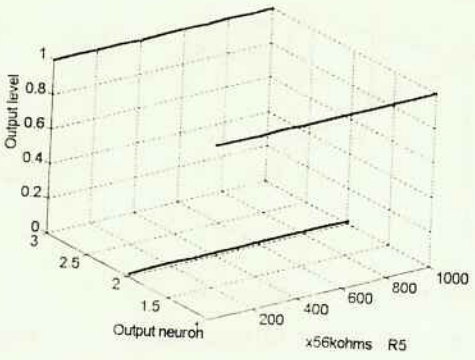

figura 28: idem, para valores superiores de R5.

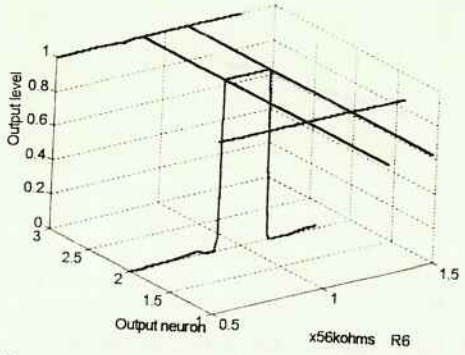

figura 29: resposta da rede a variações de R6.

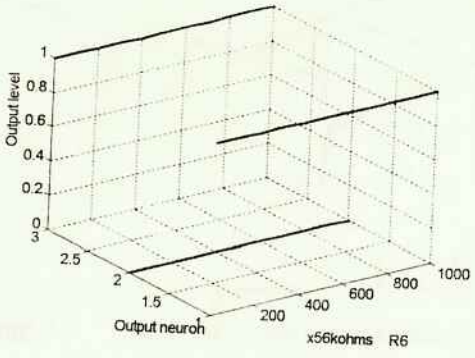

figura 30: idem, para valores superiores de R6.

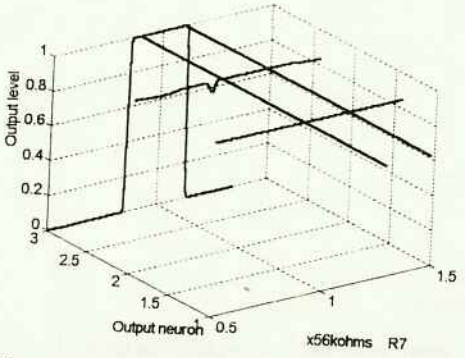

figura 31: resposta da rede a variações de R7.

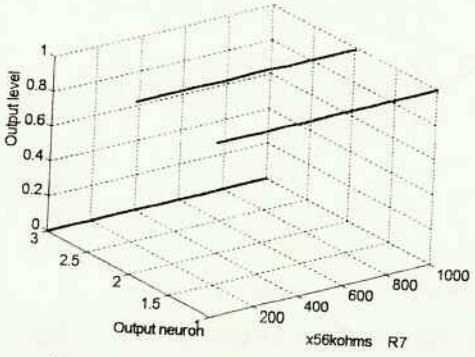

figura 32: idem, para valores superiores de R7.

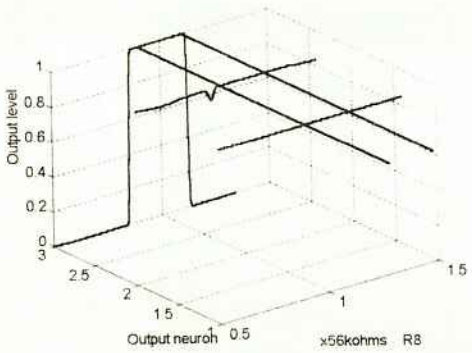

figura 33: resposta da rede a variações de R8.

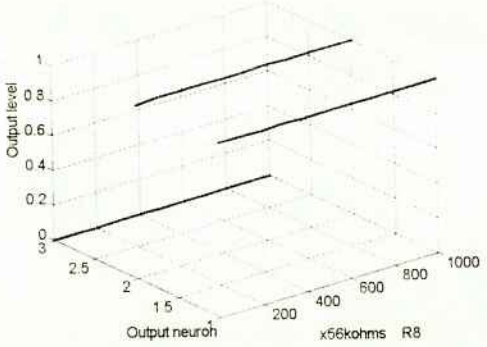

figura 34: idem, para valores superiores de R8.

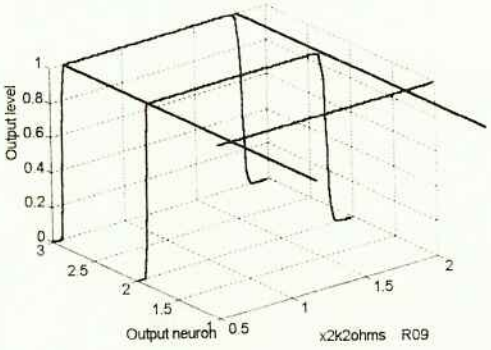

figura 35: resposta da rede a variações de R9.

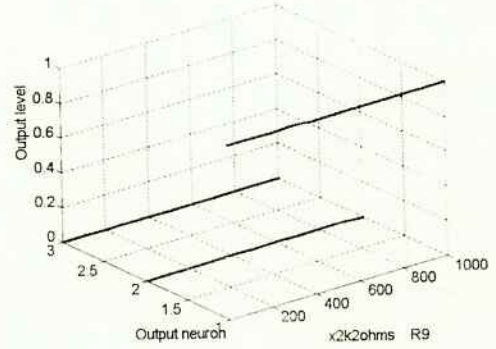

figura 36: idem, para valores superiores de R9.

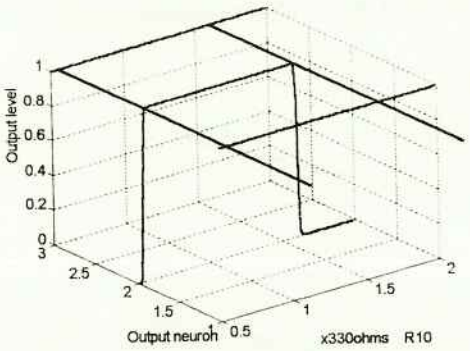

figura 37: resposta da rede a variações de R10.

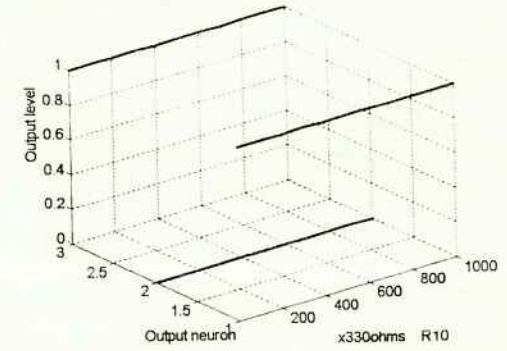

figura 38: idem, para valores superiores de R10.

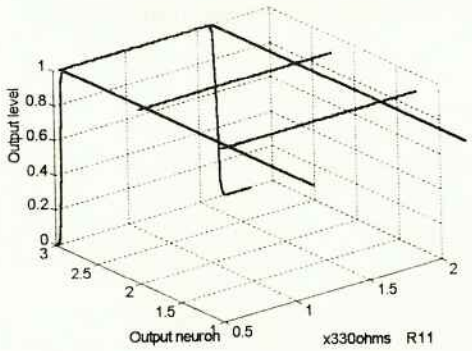

figura 39: resposta da rede a variações de R11.

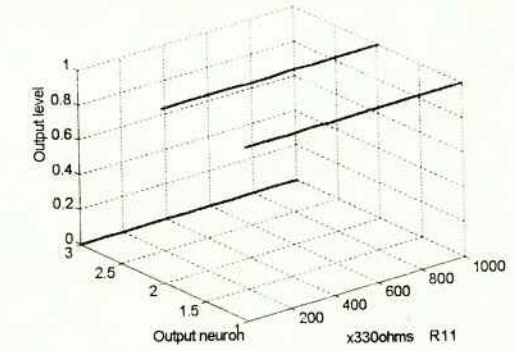

figura 40: idem, para valores superiores de R11.

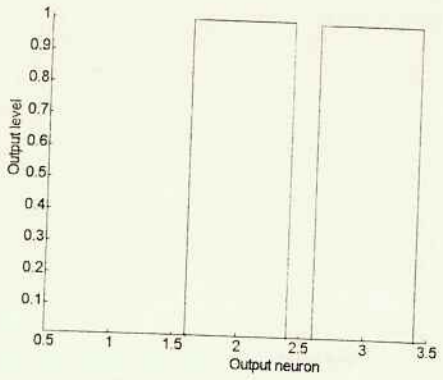

figura 41: resposta da rede para curto CE, Q1.

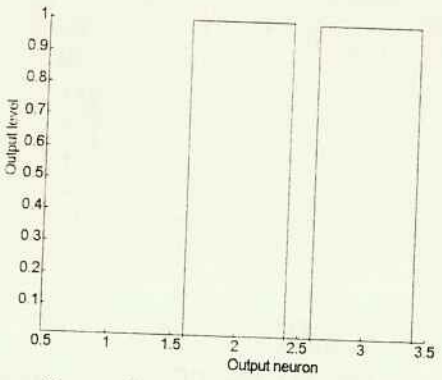

figura 42: resposta da rede para curto BE, Q1.

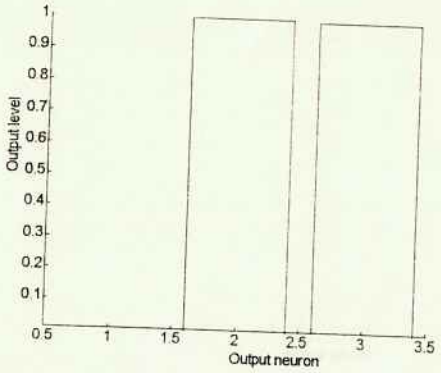

figura 43: resposta da rede para curto BC, Q1

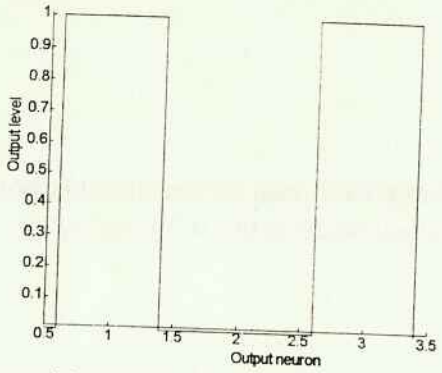

figura 44: resposta da rede para curto CE, Q2

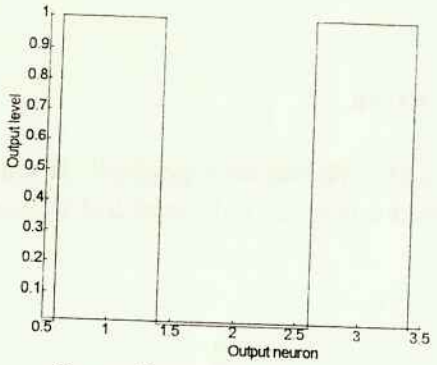

figura 45: resposta da rede para curto BE, Q2.

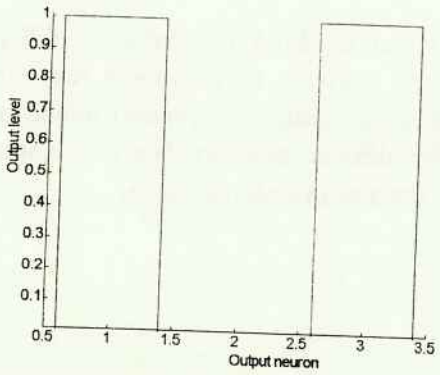

figura 46: resposta da rede para curto BC, Q2.

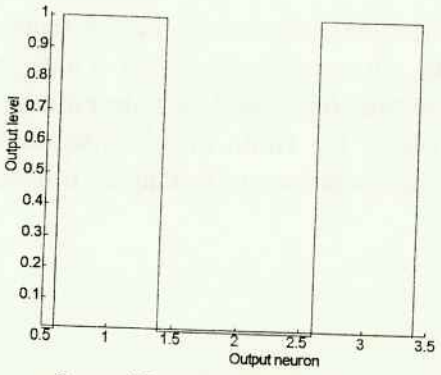

figura 47: resposta da rede para curto em C2.

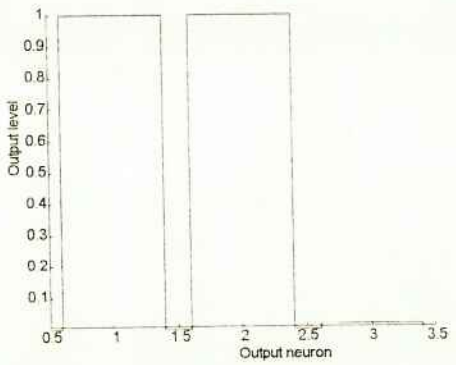

figura 48: resposta da rede para curto CE, Q3.

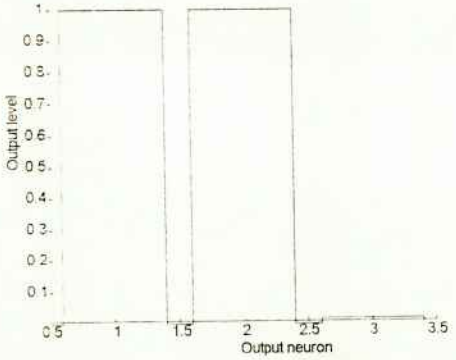

figura 49: resposta da rede para curto BE, Q3.

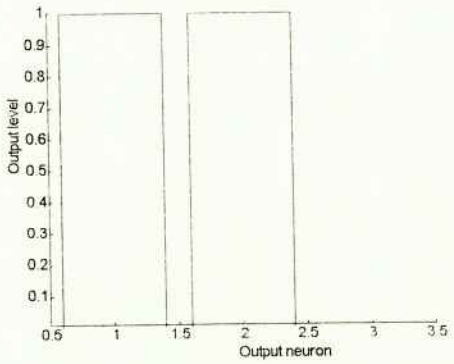

figura 50: resposta da rede para curto BC, Q3.

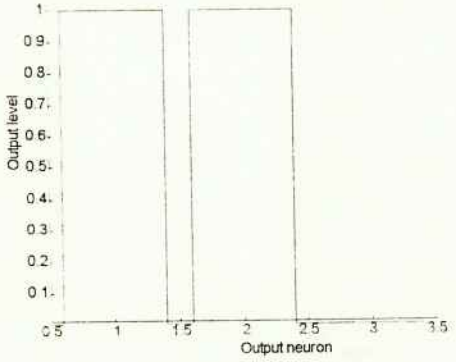

figura 51: resposta da rede para curto em C3.

5.6) Discussão.

A modularização é possível. A maior dificuldade está na escolha dos pontos de observação para o módulo 0. Esta escolha exige conhecimento sobre o funcionamento do circuito.

6. Conclusão

Os resultados deste trabalho mostram que o uso do MLP na análise de falhas é simples e robusto para circuitos simples o suficiente para permitir ao projetista resolver o problema das contradições dos dados de entrada. Futuros estudos objetivando melhorar a acuidade podem levar em conta as informações AC (freqüência, módulo, fase), os estados para circuitos seqüenciais, e a utilização de um modelo mais detalhado de falhas.

APÊNDICE

Este apêndice descreve os programas gerados e lista os mais importantes.

1) Simulador de falhas/gerador de padrões. O nome do programa é VARPAR7.EXE. Ele varia os valores dos componentes e faz uma simulação a cada variação gerando, no final, um arquivo contendo os pares (tensões_de_nós, targets) para cada simulação.

2) Treinador de rede (backpropagation com momentum e learning rate). Programa BACKNNT.EXE .(Este programa pode ser encontrado na literatura especializada)

3) Gerador de pesos. Programa WEIGHTS.EXE .

4) Normalizador de entradas.O nome do programa é CE.EXE; a normalização é feita pelo ajuste linear dos dados de entrada entre os valores -1 e 1. Ele normaliza os dados contidos em SIM.OUT e escreve o resultado em SIMX.OUT.

5) Visualizadores de resultados.

_ MAPA3.EXE: permite (para uma rede de 2 entradas):
a) Visualizar os TARGETS em função das entradas;
b) Visualizar a resposta da rede -> separação do plano de entrada nas diversas regiões determinadas pela saida da rede.

_ MAPA0.M: desenha o MAPA em MATLAB.

_ TEST.M: em MATLAB, exibe a resposta da rede em função dos valores dos parâmetros dos componentes em teste.

6) Eliminador de contradições dos dados injetados na rede. Programa ERROS.EXE.

VARPAR7.PAS

```
{O programa varia parametros de um arquivo descritivo de circuito padrao
PSPICE, simulando o circuito a cada variacao e recolhendo dos arquivos de
saida informacoes de interesse para treinamento de redes neurais}
{regras basicas da linguagem de modificacao:
    Comandos de variacao:
    #SV v1 t1 v2 t2 ... vn tn  -> Seta o valor do componente
                                  sucessivamente para V*v1, V*v2, ... V*vn,
                                  sendo V o valor nominal;
                                  targets correspondentes t1, t2, t3, etc.
  _ Os comandos de variacao devem estar contidos em linhas-comentario
da listagem .CIR na linha imediatamente anterior a linha que contem
parametro a ser variado;
  _ o parametro a ser variado deve ser o ultimo numero da linha de
comando PSPICE e deve ser precedido de um espaco;
  _ os numeros nao podem ser expressos com o auxilio dos multiplos
literais (k,Meg,n,etc.);}
{Comentario sobre andamento da programacao: Praticamente completo e funcional para combinacoes de 1
a 1;Pequenas modificacoes serao necessarias dentro de variador para combinacoes maiores}
```

```
Program variador_de_parametros_e_simulador;
uses crt,dos;
const
    num_simu_max = 210;
    num_linhas = 160;
    num_colunas = 95;
    num_com = 50;
    num_col_com=50;
    num_ele = 50;
    nnos = 15;
    num_max_patterns = 210;
type
    vetor_vnos = array[1..nnos] of real;
    matriz_das_comb = array[1..num_ele] of integer;
    tipo_linha = string[num_colunas];
    matriz_texto = array[1..num_linhas] of tipo_linha;
    tipo_de_comando = (nada,SV);
    vetor_de_valores = array[1..2*num_simu_max] of real;
    linha_com = record
                    tipo:tipo_de_comando;
                    target:integer;
                    i:integer;
                    np:integer; npa:integer;
                    posc:integer; posl:integer;
                    vo:real; {valor original}
                end;
    matriz_com = array[1..num_com] of linha_com;
    tipo_nome = string[20];
var
    cd:matriz_texto;
    nlin,i,j:integer;
    mc:matriz_com;
    vv:vetor_de_valores;  {valores setados para componentes para simulacao
                    e valores de targets correspondentes}
    ncomt      {numero total de comandos inseridos em mc}
    :integer;
    n,p:integer;
    mcomb:matriz_das_comb;
    nome_arq:tipo_linha;
{$M 65500,0,0}
procedure beep(f:integer);
begin
    sound(f);
    delay(100);
    nosound;
end;
procedure erro(erro,lc,cc:integer);
begin
    case erro of
        1:writeln('Arquivo texto muito grande.');
        2:writeln('Parametro numerico incorreto.');
        3:writeln('Device variante nao conhecido.');
        4:writeln('Comando nao conhecido.');
        5:writeln('Falta de parametros no comando.');
```

```
            6:writeln('Erro no comando de incremento.');
            7:writeln('FIM');
            8:writeln('Numero de simulacoes excessivo.');
            end;
        writeln('Erro na posicao: lc=',lc,', cc=',cc);
        readln;
        halt(1);
    end;
    {lc arquivo texto e o coloca na matriz m}
    procedure leitor(nome:tipo_linha; var m:matriz_texto; var nlin:integer);
    var arq:text;
    begin
        nlin:=0;
        assign(arq,nome);
        reset(arq);
        while not eof(arq) do
            begin
                nlin:=nlin+1;
                readln(arq,m[nlin]);
            end;
        if nlin>num_linhas then erro(1,0,0);
        close(arq);
    end;
    procedure escritor(nome:tipo_linha; var m:matriz_texto; var nlin:integer);
    var arq:text;
        i:integer;
    begin
        assign(arq,nome);
        rewrite(arq);
        for i:=1 to nlin do writeln(arq,m[i]);
        close(arq);
    end;
    procedure proc_com(var lc,cc:integer);
    begin
        while ((cd[lc,cc]<>'#')and(lc<=nlin)) do
            begin
                while ((cd[lc,cc]<>'#')and(cc<num_colunas)) do cc:=cc+1;
                if cd[lc,cc]<>'#' then
                    begin
                        lc:=lc+1;
                        cc:=1;
                    end;
            end;
        if cd[lc,cc]='#' then cc:=cc+1;
    end;
    procedure prox_palavra(var lc,cc:integer; var cd:matriz_texto; var nlin:integer);
    begin
        while ((cd[lc,cc]<>' ')and(lc<=nlin)) do
            begin
                while ((cd[lc,cc]<>' ')and(cc<num_colunas)) do cc:=cc+1;
                if cc>=num_colunas then
                    begin lc:=lc+1; cc:=1; end;
            end;
        while ((cd[lc,cc]=' ')and(lc<=nlin)) do
```

```
        begin
          while ((cd[lc,cc]=' ')and(cc<num_colunas)) do cc:=cc+1;
          if cc>=num_colunas then
            begin lc:=lc+1; cc:=1; end;
        end;
end;
function leia_num(lc,cc:integer; var cd:matriz_texto):real;
var ns:string[20];
   i,err:integer;
   num:real;
begin
    for i:=1 to 20 do ns[i]:=' ';
    i:=1;
    repeat
        ns[i]:=cd[lc,cc];
        cc:=cc+1;
        i:=i+1;
    until ((cd[lc,cc]<'0')or(cd[lc,cc]>'9'))and
        (upcase(cd[lc,cc])<>'E')and(cd[lc,cc]<>'.')and
        (cd[lc,cc]<>'-')and(cd[lc,cc]<>'+');
    ns[0]:=chr(i-1);
    val(ns,num,err);
    if err<>0 then erro(2,lc,cc);
    leia_num:=num;
end;
procedure leia_parametros(lc,cc:integer; var lp:vetor_de_valores; var npar:integer);
var fim:boolean;
begin
    npar:=0;
    fim:=false;
    repeat
        prox_palavra(lc,cc,cd,nlin);
        if cd[lc,cc] = '*' then prox_palavra(lc,cc,cd,nlin);
        if (ord(cd[lc,cc])>=48)and(ord(cd[lc,cc])<=ord('9'))
          then begin
              npar:=npar+1;
              lp[npar]:=leia_num(lc,cc,cd);
              end
          else fim:=true;
    until fim;
end;
function det_valor_comp(lc,cc:integer; var l_par,c_par:integer):real;
var l,c,i:integer;
    valor:real;
    procedure procure_ultimo;
    var lt:integer;
    begin
        repeat
            lt:=l;
            if (cd[l,c]>='0')and(cd[l,c]<='9')and(lt=l) then
              begin
                  l_par:=l; c_par:=c;
              end;
            prox_palavra(l,c,cd,nlin);
```

```
            until lt<>l;
        end;
    begin
        l:=lc;
        while cd[l,1]='*' do l:=l+1;
        c:=1;
        procure_ultimo;
        valor:=leia_num(l_par,c_par,cd);
        det_valor_comp:=valor;
    end;
    procedure seta_valores(var lc,cc:integer;
                           lmc:integer;
                           var ivv:integer;
                           var mc:matriz_com;
                           var vv:vetor_de_valores);
    var l_parv,c_parv,npar,i:integer;
        lp:vetor_de_valores;
        valor:real;
    begin
        leia_parametros(lc,cc,lp,npar);
        if npar<1 then erro(5,lc,cc);
        valor:=det_valor_comp(lc,cc,l_parv,c_parv);
        for i:=1 to npar div 2 do
            begin
                vv[ivv+2*(i-1)]:=lp[2*i-1]*valor;
                vv[ivv+2*(i-1)+1]:=lp[2*i];
            end;
        mc[lmc].tipo:=SV;
        mc[lmc].i:=ivv;
        mc[lmc].np:=npar div 2;
        mc[lmc].npa:=0;
        mc[lmc].posl:=l_parv;
        mc[lmc].posc:=c_parv;
        mc[lmc].vo:=valor;
        ivv:=ivv+npar;
    end;
    procedure codifique_comandos(var lmc:integer;
                           var mc:matriz_com;
                           var vv:vetor_de_valores);
    var i,j,ivv,lc,cc:integer;
        com:string[2];
    begin
        lc:=1;
        cc:=1;
        lmc:=1;
        ivv:=1;

        while (lc<=nlin) do
            begin
            proc_com(lc,cc);
            if lc<=nlin then
                begin
                    com:=upcase(cd[lc,cc])+upcase(cd[lc,cc+1]);
                    if com='SV'
```

```
                then seta_valores(lc,cc,lmc,ivv,mc,vv)
                else erro(4,lc,cc);
              lmc:=lmc+1;
              if ivv>2*num_simu_max then erro(8,lc,cc);
            end;
          end;
    lmc:=lmc-1;
end;
procedure modifique_CIR(lp,cp:integer; valor:real);
var l,num:tipo_linha;
    i:integer;
begin
     l:=cd[lp];
     l[0]:=chr(cp-2);
     str(valor,num);
     l:=l+num;
     for i:=1 to num_colunas do cd[lp][i]:=' ';
     cd[lp]:=l;
     escritor(nome_arq+'.cir',cd,nlin);
end;
procedure valor_original(ncom:integer; var mc:matriz_com);
var lp,cp:integer;
begin
     lp:=mc[ncom].posl;
     cp:=mc[ncom].posc;
     mc[ncom].npa:=0;
     modifique_cir(lp,cp,mc[ncom].vo);
end;
procedure modifique_e_incremente(ncom:integer; var mc:matriz_com; var fim:boolean);
var lp,cp:integer;
    valor:real;
begin
     lp:=mc[ncom].posl;
     cp:=mc[ncom].posc;
     fim:=false;
     if mc[ncom].npa<mc[ncom].np
          then begin
               case mc[ncom].tipo of
                    SV:valor:=vv[mc[ncom].i+2*mc[ncom].npa];
                    end;
               mc[ncom].npa:=mc[ncom].npa+1;
               end
          else begin
               valor_original(ncom,mc);
               fim:=true;
               end;
        if not fim then modifique_cir(lp,cp,valor);
    end;
    function comb(n,p:integer):real;
    var
    aux1,aux2,aux3:real;
       function fat(num:integer):real;
        var
        i:integer;
```

```
        aux:real;
       begin
        if (num = 0) then
        fat:=1
        else
         begin
          aux:=1;
          for i:=1 to num do
           begin
            aux:=aux*i;
           end;
          fat:=aux;
         end;
      end;
  begin
  aux1:=fat(n);
  aux2:=fat(p);
  aux3:=fat(n-p);
  comb:=aux1/(aux2*aux3);
  end;
  procedure gcomb(var mcomb:matriz_das_comb; n,p:integer; var fim:boolean);
  var i,j:integer;
     m:matriz_das_comb;
     sair:boolean;
     function seg_final(i:integer):boolean;
     var j:integer;
        cond:boolean;
     begin
         cond:=true;
         if m[p]=n then
                 begin
                 for j:=p-1 downto i+1 do
                        if m[j]+1 <> m[j+1] then cond:=false;
                 end
                 else cond:=false;
        if i=p then cond:=true;
        seg_final:=cond;
    end;
    function final(i:integer):boolean;
    begin
       final:=(m[i]=n-(p-i));
    end;
begin
    m:=mcomb;
    sair:=false;
    for i:=p downto 1 do
       begin
          if seg_final(i) and not final(i)
            then begin
               m[i]:=m[i]+1;
               if i<p then
                     for j:=i+1 to p do m[j]:=m[j-1]+1;
               sair:=true;
               end;
```

```
            mcomb:=m;
            if sair then exit;
          end;
      fim:=true;
      for i:=1 to p do if not final(i) then fim:=false;
end;
procedure lcout(var vnos:vetor_vnos; var nnos:integer);
var cdo:matriz_texto;
    nlin,l,c,i,j:integer;
    st:tipo_linha;
    nno:integer;
    valor:real;
    fim:boolean;

        procedure proc_str(st:tipo_linha; var l:integer);
        var i:integer;
        begin
            l:=0;
            i:=ord(st[0]);
            repeat
                if l>1 then cdo[l-1][0]:=chr(128);
                l:=l+1;
                cdo[l][0]:=chr(i);
            until (l>nlin) or (st=cdo[l]);
        end;
        procedure proc_no(var l,c,nno:integer; var valor:real; var fim:boolean);
        begin
            repeat
                repeat
                    repeat
                        c:=c+1;
                    until (cdo[l,c]='(')or(cdo[l,c]='*')or
                        (ord(cdo[l,c])=13)or(c>127);
                    if (ord(cdo[l,c])=13) or (c>127)
                      then begin l:=l+1; c:=0; end;
            until (l>=nlin-1) or (cdo[l,c]='(') or (cdo[l,c]='*');

            fim:=(l>nlin-3) or (cdo[l,c]='*');

            if not fim then
              begin
                  prox_palavra(l,c,cdo,nlin);
                  nno:=trunc(leia_num(l,c,cdo));
                  prox_palavra(l,c,cdo,nlin);
                  valor:=leia_num(l,c,cdo);
              end;
        end;
begin
    st:=' ****     SMALL SIGNAL BIAS SOLUTION';
    for i:=1 to num_linhas do
      for j:=1 to num_colunas do cdo[i,j]:=' ';
    leitor(nome_arq+'.out',cdo,nlin);
    proc_str(st,l);
    l:=l+5;
    c:=0;
```

```
      i:=1;
      repeat
          proc_no(l,c,nno,valor,fim);
          if not fim then
            begin
              vnos[i]:=valor;
              i:=i+1;
            end;
      until fim;
      nnos:=i-1;
end;
procedure variador(n,p:integer);
var i,nsim:integer;
   mcomb:matriz_das_comb;
   fim1,fim2:boolean;
   arq:text;
   NPattern:integer;
   MPattern:array[1..Num_max_patterns] of integer;

  procedure simule_e_imprima;
  var i,k,m:integer;
     vnos:vetor_vnos;
     nnos:integer;
     fim:boolean;
     valor,vo,v:real;
  begin
     clrscr;
     write('Simulacao ',nsim,' Elementos ');
     for k:=1 to p do write(mcomb[k],' ');
     writeln;
    for k:=1 to ncomt do writeln(cd[mc[k].posl]);
     swapvectors;
     exec('pspice1.exe',nome_arq+'.cir');
     swapvectors;
     leout(vnos,nnos);
     {write(arq,vnos[1]:1:4,' ');}
     write(arq,vnos[2]:1:4,' ');
     {write(arq,vnos[3]:1:4,' ');
    write(arq,vnos[5]:1:4,' ');
    write(arq,vnos[6]:1:4,' ');
    write(arq,vnos[7]:1:4,' '); }
    write(arq,vnos[8]:1:4,' ');
    write(arq,vnos[9]:1:4,' ');
    writeln(arq);
    { Feito para variacao um a um }
    m:=1;
    for k:=1 to ncomt do
      begin
        vo:=mc[k].vo;
        if mc[k].npa=0
          then
            begin
            valor:=vo;
            write(arq,'1 ');
```

```
                end
              else
                begin
                case mc[k].tipo of
                    SV:valor:=vv[mc[k].i+2*(mc[k].npa-1)];
                    end;
                v:=vv[mc[k].i+2*(mc[k].npa-1)+1];
                write(arq,v:1:0,' ');
                if vv[mc[k].i+2*(mc[k].npa-1)+1]<1e-30 then m:=k+1;
                end;
          end;
        writeln(arq,m);
        nsim:=nsim+1;
    end;
begin
    assign(arq,'sim.out');
    rewrite(arq);
    for i:=1 to num_ele do mcomb[i]:=0;
    for i:=1 to p do mcomb[i]:=i;
    nsim:=1;
    fim1:=false;
    NPattern:=1;
    simule_e_imprima;
    repeat
        repeat
            i:=1;
            repeat
                modifique_e_incremente(mcomb[i],mc,fim2);
                if not fim2 then simule_e_imprima;
            until fim2;
            if p>1 then
              begin
                repeat
                    i:=i+1;
                    modifique_e_incremente(mcomb[i],mc,fim2);
                until (not fim2) or (fim2 and (i=p));
                if not fim2 then simule_e_imprima;
              end;
        until (fim2 and (i=p));
        gcomb(mcomb,n,p,fim1);
      until fim1;
      close(arq);
      for i:=1 to n do valor_original(i,mc);
    end;
    begin
      nome_arq:='amp';
      for i:=1 to num_simu_max do vv[i]:=0;
      for i:=1 to num_linhas do
        for j:=1 to num_colunas do
          cd[i,j]:=' ';
      leitor(nome_arq+'.cir',cd,nlin);
      clrscr;
      codifique_comandos(ncomt,mc,vv);
      variador(ncomt,1);
```

```
    {
     for i:=1 to ncomt do
        begin
           writeln(mc[i].vr,' ');
           if mc[i].vr then writeln(mc[i].vpi,' ',mc[i].vpp,' ',mc[i].vpn,' ',mc[i].vmi,' ',mc[i].vmp,'
',mc[i].vmn);
           writeln(mc[i].ca,' ');
           writeln(mc[i].cc,' ',mc[i].rcc);
           writeln(mc[i].posl,' ',mc[i].posc);
           writeln(mc[i].vo);
           writeln;
           readln;
        end;
     readln;
     }
     writeln('Fim das simulacoes.');
end.
```

CE.PAS

```
program ce;
uses crt,turbo3;
const cols = 20;
type matriz = array[1..220] of array[1..cols] of real;
var arq:text;
   i,j,nl,np,nq:integer;
   p:matriz;
   q:array[1..220] of array[1..cols] of single;
   mins:array[1..cols] of real;
   maxs:array[1..cols] of real;
   nome:string[20];
   car:char;
function max(var v:matriz; j,n:integer):real;
var i:integer;
   m:real;
begin
   m:=0;
   for i:=1 to n do
      if v[i][j] > m then m:=v[i][j];
   max:=m;
end;
function min(var v:matriz; j,n:integer):real;
var i:integer;
   m:real;
begin
   m:=1e31;
   for i:=1 to n do
      if v[i][j] < m then m:=v[i][j];
   min:=m;
end;
begin
   clrscr;
   nome:='sim';
   assign(arq,nome+'.out');
```

```
    reset(arq);
    i:=1;
    repeat
       j:=1;
       repeat
            read(arq,p[i][j]); j:=j+1;
       until SeekEoln(arq);
       np:=j-1;
end.       j:=1;
       writeln;
       repeat
            read(arq,q[i][j]); j:=j+1;
       until SeekEoln(arq);
       nq:=j-1;
       if not eof(arq) then i:=i+1;
    until SeekEof(arq);
    nl:=i-1;
    close(arq);
    writeln(nl,' ',np,' ',nq);
    readln;

    assign(arq,'mm.dat');
    writeln('Normalizar com dados atuais ou anteriores? (ESC=anteriores)');
    while not keypressed do;
    read(kbd,car);
    if ord(car) <> 27
      then
        begin
        rewrite(arq);
        for j:=1 to np do
          begin
          mins[j]:=min(p,j,nl);
          maxs[j]:=max(p,j,nl);
          writeln(mins[j],' ',maxs[j]);
          writeln(arq,mins[j],maxs[j]);
          end;
        end
      else
        begin
        reset(arq);
        for j:=1 to np do
          begin
          readln(arq,mins[j],maxs[j]);
          end;
        end;
    close(arq);

    for i:=1 to nl do
      begin
      for j:=1 to np do
        begin
        p[i][j] := 2*(p[i][j] - mins[j])/(maxs[j] - mins[j]) - 1;
        write(p[i][j],' ');
        end;
```

```
        writeln;
        end;
      assign(arq,nome+'x.out');
      rewrite(arq);
      for i:=1 to nl do
        begi
        for j:=1 to np do
          write(arq,p[i][j]:1:4,' ');
        writeln(arq);
        for j:=1 to nq do
          write(arq,q[i][j]:1:0,' ');
        writeln(arq);
        end;
      close(arq);
      writeln('Arquivo convertido para ',nome+'x.out');
      readln;
```

MAPA0.M

```
%
% Desenha gráfico dos targets.
% Entradas em P e Targets em T.
%
whitebg;
graymon;
set(0,'defaulttextfontsize',7);
set(0,'defaulttexthorizontalalignment','center');
targets;
%axis([-1.2 1.2 -1.2 1.2]);
axis([0.2 0.4 -0.97 -0.9]);
for i = 1:size(T,1),
  l = T(i,size(T,2));
  x = P(i,2);   % Abcissa
  y = P(i,6);         % Ordenada
  if l == 1   text(x,y,'a');
    elseif l == 2   text(x,y,'b');
    elseif l == 3   text(x,y,'c');
    elseif l == 4   text(x,y,'d');
    elseif l == 5   text(x,y,'e');
    elseif l == 6   text(x,y,'f');
    elseif l == 7   text(x,y,'g');
    elseif l == 8   text(x,y,'h');
    elseif l == 9   text(x,y,'i');
    elseif l == 10  text(x,y,'j');
    elseif l == 11  text(x,y,'k');
    elseif l == 12  text(x,y,'l');
    elseif l == 13  text(x,y,'m');
    elseif l == 14  text(x,y,'n');
    elseif l == 15  text(x,y,'o');
    elseif l == 16   text(x,y,'p');
    elseif l == 17   text(x,y,'q');
    elseif l == 18   text(x,y,'r');
    elseif l == 19   text(x,y,'s');
```

```
    elseif l == 20   text(x,y,'t');
    elseif l == 21   text(x,y,'u');
    elseif l == 22   text(x,y,'v');
    elseif l == 23   text(x,y,'x');
    elseif l == 24  text(x,y,'y');
    elseif l == 25  text(x,y,'z');
    elseif l == 26  text(x,y,'1');
    elseif l == 27  text(x,y,'2');
    elseif l == 28  text(x,y,'3');
    elseif l == 29  text(x,y,'4');
    elseif l == 30   text(x,y,'5');
    elseif l == 31   text(x,y,'6');
    elseif l == 32   text(x,y,'7');
    elseif l == 33   text(x,y,'8');
    elseif l == 34   text(x,y,'9');
    elseif l == 35   text(x,y,'!');
    elseif l == 36   text(x,y,'@');
    elseif l == 37   text(x,y,'#');
    elseif l == 38  text(x,y,'$');
    elseif l == 39  text(x,y,'%');
    elseif l == 40  text(x,y,'&');
    elseif l == 41  text(x,y,'*');
    elseif l == 42  text(x,y,'(');
    elseif l == 43  text(x,y,')');
  end;
end;
```

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 Tipo 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
BT/PEE/94-17 - Redes Neurais Artificiais Aplicadas à Identificação de Pulsos Decádicos em Linhas Telefônicas - ANTONIO P. TIMOSZCZUK, EUVALDO F. CABRAL JR.
BT/PEE/95-01 - Estudo Comparativo de Métodos de Cálculo da Frequência Fundamental - MARCOS COSTA HUNOLD, EUVALDO F. CABRAL JR.
BT/PEE/95-02 - Combinando Técnicas de Redes Neurais Artifíciais e Informações de Excitação no Reconhecimento Automático do Locutor - ANDRÉ BORDIN MAGNI, EUVALDO F. CABRAL JR.